Unsupervised Learning in Human Activity Recognition: A First Foray into Clustering Data Gathered from Wearable Sensors

Niklas Weber

Supervisors:
Dr. Aki Härmä, Philips Research Eindhoven
Prof. Dr. Tom Heskes, Radboud University Nijmegen

January 2016 through February 2017

Abstract

Human Activity Recognition is the task of determining what kind of activity led to measurements in a sensor data stream. The main body of work surrounding this task focuses on the supervised and semi-supervised scenarios, leaving the important use-case of completely unlabeled data untreated. This thesis is a foray into tackling this unsupervised case, with an additional focus on the impact of missing data. We build a theoretical foundation for unsupervised human activity recognition, link prior research of the supervised setting to other studies regarding the interaction between missing data handling strategies and clustering and evaluate the clustering performance as a function of the amount of missing data of K-Means, Minibatch K-Means, Gaussian Mixture Models and Dirichlet Process Gaussian Mixture models on both synthetic and real-world datasets. Based on our findings we provide first practical guidelines on how to approach this novel problem and recommend K-Means and Gaussian Mixture Models as good all-round solutions.

Contents

Acknowledgement

None of the work presented here would have been even remotely possible without the help of a host of wonderful people. It is my pleasure to express my gratitude towards them.

First and foremost I would like to thank my supervisors Tom Heskes and Aki Härmä for their guidance, leadership, support and for offering me their vast treasures of knowledge.

I also thank my former colleagues at Philips Research Eindhoven who were always willing to help and who created a place I was more than happy to work at.

Further thanks go to all the members of the Philips Intern Committee in particular and all Philips interns in general. Your hard work inspired me, your adventures balanced me and your friendship made this place home. Fuso Gang forever.

I wish, from my heart, to extend these thanks to all of my friends, near and far. Without your continued support I would probably still write my Bachelor's thesis. You mean the world to me.

The same holds true for my father, my mother and my brother. Thanks for believing in me – even when I did not.

Finally, I would like to thank two people in particular. Luísa van der Linden, for her unending patience and for letting me see the light in darkness. And Jakob Bleier, for his wonderful friendship, endless inspiration and boundless wisdom. You should have the beard, not me.

You are the Gandalfs, the Éowyns, the Gimlis, the Sams and even the Eagles to my Frodo. And we all know Frodo would not have survived for five minutes without them.

Guren glassui

Chapter 1

Introduction

As computing hardware gets smaller, faster and cheaper it also becomes more pervasive. So much so that nowadays nearly every adult carries a smartphone for the better part of the day, potentially accompanied by a smart watch, a step counter and other accessories. These devices are usually equipped with sensors, detecting measurements such as, among others, acceleration, temperature or heart rate. There are many more examples, including sensors not attached to one's body. Consider CCTV cameras, RFID tags on a lot of items or the sensors embedded into modern buildings.

As a direct consequence of an increase in availability of this sensor data there is an increase in trying to make sense of it all. The overall field, capturing the grand total of all sensors and detection tasks, is sometimes called ubiquitous sensing (Puccinelli and Haenggi 2005). Within this field a multitude of tasks and challenges are considered, for example vehicle detection (Yoo 2013), identifying pedestrian group motion (Yücel et al. 2013), building smart cities (Hancke, Silva, and Hancke 2012), or, on an underlying level, devising energy-efficient algorithms for wireless sensors (Xiong et al. 2013). This work focuses on a task at the intermediate scale: Human Activity Recognition (HAR).

HAR is the task of inferring what a person does based on a stream of sensor data. Let us, for now, not be bothered by a formal definition, which will follow later. Fundamentally, one thus tries to find a mapping from what the sensors tell you to what a person is doing at the moment. 'Sensors' here is a very broad term and, in principle, can encompass the whole array of sensing machinery at humanity's disposal. Traditionally they have been grouped into two broad classes: external and wearable (Lara and Labrador 2013).

External sensors are placed outside of the person doing the activities. This includes cameras, touch or heat sensors among others. These are placed in objects of interest such as, for example, furniture or kitchen appliances.

A common application of these kinds of sensors is, for example, in smart homes/buildings (Hong and Ohtsuki 2011; Sarkar, Y.-K. Lee, S. Lee, et al. 2011; Tolstikov et al. 2011; Van Kasteren, Englebienne, and Kröse 2010; Yang, J. Lee, and Choi 2011). Due to the high specificity of the data collected, i.e. it is known not only how the person moves but also what things she interacts with, the scale of recognized activities can be rather fine-grained and complex. One could for example distinguish cleaning the fridge from cleaning a cupboard, or distinguish reading a book from sitting. However, all such detection is limited to the sensor-equipped environment and depends upon the person using the appropriate items.

Wearable sensors, on the other hand, are attached to the user directly or at least carried around by her. There is a wide variety of such sensors. Lara and Labrador (2013) provide a taxonomy of wearable sensors. They discriminate them based on a number of properties. Let us get a sense of the landscape by considering two of these properties.

On the one hand there is obtrusiveness. Some low-obtrusion systems only rely on phones (Berchtold et al. 2010; Reddy et al. 2010) or watches (Maurer et al. 2006a). On the other hand, the designs of Pärkkä et al. (2006) require a whole rucksack of equipment, which is vastly more obtrusive.

On the other hand there is a combined property called selection of attributes and sensors. Looking at attributes and sensors is also helpful to get a feel for how people approach this problem. One popular setup is the use of several accelerometers, often bundled with additional sensors (Bao and Intille 2004; Ermes, Pärkkä, and Cluitmans 2008; Tapia et al. 2007). This yields a wide variety of data. Some examples include, ambient noise and light, humidity, ambient temperature (Maurer et al. 2006b; Pärkkä et al. 2006), body temperature, acceleration (Bao and Intille 2004; Hanai, Nishimura, and Kuroda 2009; He and Jin 2008) or location (Reddy et al. 2010; Riboni and Bettini 2011).

As we can see, this is a wide field with lots of variation. In this work, we focus our attention on the data gathered by wearable sensors, with as little obtrusiveness as possible. One of our goals is making this task feasible in a way that is non-disturbing and as transparent as possible to the user.

However, there is one critical qualification that still needs to be made. All of the studies cited above worked in a supervised, or at least semi-supervised, setting. This means, that for all (supervised) or some (semi-supervised) amount of data, ground truth, i.e. the actual tasks carried out in that moment, is known. In many scenarios this is neither true nor feasible. For example, consider the user who just downloaded an application intended to give him relevant information based on his activities - but now has to hand-label these activities laboriously until the underlying model is working

well enough. Furthermore, there are huge sets of such data, but without known labels. It might still be possible to find structure in this data and do something useful with it.

In fact, little such unsupervised work has been done so far at all. This thesis tries to address this knowledge gap and make existing unlabeled data useful by taking a first step towards Unsupervised Human Activity Recognition. The whole of this task straightforwardly decomposes into two subtasks: Firstly, discover *that* there is something, i.e. identify similar patterns of sensor data. Secondly, identify *what* there is, i.e. try to generate useful labels for these clusters.

The work presented in this thesis solely focuses on the first subproblem: finding clusters.

The remainder of this thesis is structured as follows:

Chapter 2 provides background and theoretical information. It reviews the prior literature in the field of supervised human activity recognition, describing typical sensor setups, processing pipelines and formalizations. Furthermore it provides a novel formalization of the unsupervised case. Subsequently, it goes on to give a broad overview of clustering in general to provide a theoretical backdrop for the rest of this work. It also contains an overview of different types of missing data and their impact on analyses. This is followed by an overview of strategies for handling such missing data, informing us in which scenario what strategy is appropriate. The background section concludes by reviewing research on the interaction of these missing data handling strategies and clustering, bringing together all of the aforementioned topics.

Chapter 3 describes an experiment on discovering subjects in the real-world PAMAP2 dataset. Insights from this experiment inform the need for investigating different scenarios regarding the relationship between subjects and activities.

Chapter 4 contains this investigation, detailing three scenarios: all activities are carried out by all subjects, every activity is unique to one single subject and a mixture of these two. For each scenario a synthetic dataset fitting this situation is presented. Furthermore, for each scenario experimental results are detailed regarding the clustering performance of various models for this dataset, as a function of the amount of missing values.

Chapter 5 describes how the knowledge gathered in the previous section was used to inform experiments focused on discovering activities in PAMAP2. It displays experimental results of clustering performance as a function of the amount of missing values. Finally it contains first recommendations for practical model selection for unsupervised human activity recognition.

Chapter 6 holds a conclusion of the work presented here and provides pointers for promising future research.

Let us dive right in by getting a better feel what human activity recognition entails and how it is normally being tackled.

Chapter 2

Problem Setting and Background

2.1 The Problem Setting

To figure out how to solve this problem we have set ourselves, let us devise a way of defining it formally. Recall, that our informal definition was that we wish to figure out what a person does based on a stream of data. To this end, it is helpful to have a look at what this data is that we wish to cluster.

2.1.1 Where the data comes from

Let us begin at the beginning: the origin of the data. In this work, we only consider data generated by wearable devices. A typical data collection/processing pipeline is depicted in Figure 2.1 (Lara and Labrador 2013, based on their Figure 2, p.1194). Sensors are attached to the subject under consideration. Typical sensors include accelerometers, thermometers, heartbeat sensors, GPS and others. We shall talk about the different kinds of data involved in more detail later. This data is then often sent to a local collection/fusion device such as a smart phone (Oh, Park, and Cho 2010; Omatu et al. 2009) or other portable device (Kao, Lin, and Wang 2009; Maurer et al. 2006b; Pärkkä et al. 2006). This devices bundles the data streams. The amount of calculation done here varies between setups. It ranges from none at all, over some pre-processing to whole model fitting and prediction (Berchtold et al. 2010; Riboni and Bettini 2011). If there is any need for additional processing or more permanent storage, relevant data may then be sent to a remote server (Lara, Pérez, et al. 2012).

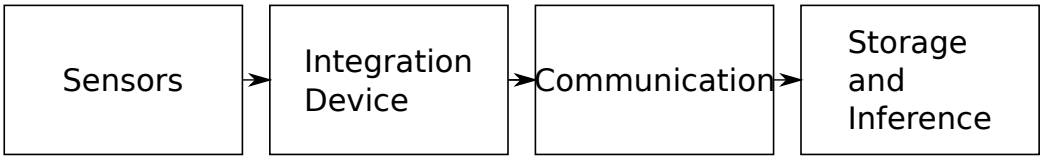

Figure 2.1: A typical data gathering/processing setup for wearable sensors

2.1.2 Formal definition of the problem

With the source of the data described and a detection goal outlined, we can now formally define the problem we wish to solve. We have found two formal definitions in the literature, both in Lara and Labrador (2013, p. 2). The first one follows straightforwardly from the setting outlined so far:

Definition 1. *Definition: Human Activity Recognition Problem (HARP). Given a set $S = \{S_0, \ldots, S_{k-1}\}$ of k time series, each one from a particular measured attribute, and all defined within time interval $I = [t_\alpha, t_\omega]$, the goal is to find a temporal partition $\langle I_0, \ldots, I_{r-1} \rangle$ of I, based on the data in S, and a set of labels representing the activity performed during each interval I_j (e.g. sitting, walking, etc). This implies that time intervals I_j are consecutive, non-empty, non-overlapping, and such that $\bigcup_{j=0}^{r-1} I_j = I$.*

This, however, does not account for hierarchical activities, nor for overlapping activities. Furthermore, it is infeasible in general, with the number of possible temporal partitions exploding quickly.

This gives rise to a relaxation and reformulation of the problem. The time series now get grouped into windows, for which at least partial labels are known. The objective changes to finding a function that maps time windows to activity labels as well as possible. Again we are following the definition of Lara and Labrador.

Definition 2. *Relaxed HARP (RHARP): Given (1) a set $W = \{W_0, \ldots, W_{m-1}\}$ of m equally-sized time windows, totally or partially labeled, and such that each W_i contains a set of time series $S_i = \{S_{i,0}, \ldots, S_{i,k-1}\}$ from each of the k measured attributes, and (2) a set of $A = \{a_0, \ldots, a_{n-1}\}$ of activity labels, the goal is to find a mapping function $f : S_i \to A$ that can be evaluated for all possible values of S_i, such that $f(S_i)$ is as similar as possible for the actual activity performed during W_i.*

However, this definition is not quite applicable to what we are doing. As we are working in the unsupervised setting, there is no set of labels A available

to us. Furthermore, we cannot maximize the objective function of trying to match existing labels as closely as possible. As such, we must reformulate the problem. We propose the following definition:

Definition 3. *Unsupervised Relaxed HARP (URHARP): Given (1) an unlabeled set* $W = \{W_0, \ldots, W_{m-1}\}$ *of* m *equally-sized time windows, such that each* W_i *contains a set of time series* $S_i = \{S_{i,0}, \ldots, S_{i,k-1}\}$ *from each of the* k *measured attributes, and (2) a set of* $A = \{0, \ldots, n-1\}$ *of arbitrary activity labels, the goal is to find a mapping function* $f : S_i \to A$ *that can be evaluated for all possible values of* S_i, S_j*, such that* $f(S_i) = f(S_j)$ *if and only if the true activities carried out during* W_i *is equal to the true activity carried out during* W_j*.*

A few key things are different to the previous definition. Firstly, all windows are assumed to be unlabeled. Secondly, activity labels are arbitrary as long as they are internally consistent. The reason for this is that we do not try to identify elements of a fixed set of activities, such as running or swimming, but instead are interesting in finding blocks of activity in general. Linking these to actual activities is a different task which we are not considering here. We decide to still keep a fixed range on A as often the number of activities we are trying to find is fixed. Thirdly, the goal is now to assign labels in such a way that two windows should have the same label iff they were generated during the same true activity.

This is problematic as we, by definition, do not possess this information. As such it is very hard to say how good a particular solution to an instance of URHARP is. This, however, is to be expected. Fundamentally this is due to the fact that clustering is hard to evaluate because of the missing ground truth. Evaluation is one of the main issues in clustering. Consequently, it is also major hurdle in this thesis. Within this work, we will circumvent this problem by using datasets that do have labels available. We will not use these labels to build our models, but we will use them to evaluate the goodness of our clustering.

Now that we have defined the problem to solve, let us have a look at the actual model fitting pipeline involved.

2.1.3 The model fitting pipeline

A model fitting pipeline for wearable-sensor data is commonly designed similar to this the schema depicted in Figure 2.2: One begins with the raw data. This data then passes through pre-processing, which can differ per data channel. For example, it might make sense to extract an overall direction and intensity change for acceleration, but it might be more useful to take a simple average

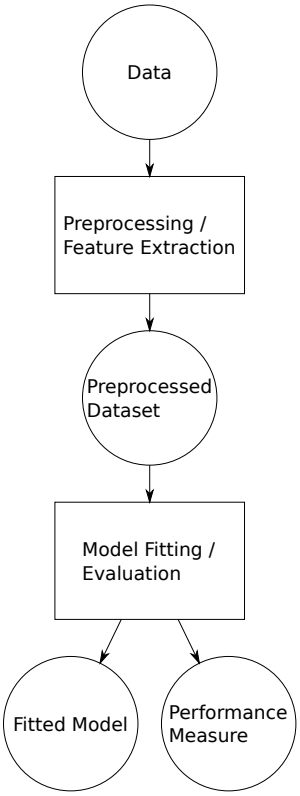

Figure 2.2: A typical model fitting pipeline.

for heart rate or temperature. The whole of this process can also be referred to as the feature extraction stage. Also commonly included in this stage is the binning of data into time windows. In such a case, multiple consecutive data points get grouped to form one new data point in the resulting dataset.

The following block is the model fitting and evaluation stage. This is intentionally vague, as it critically depends on the whole setup of model fitting and evaluation. The idea is, that a whole modeling procedure is evaluated. In our case it means a setup consisting of resampling, hyperparameter tuning and model fitting. Other setups could simply be a train/test set split followed by model fitting and evaluation, or even just a single clustering run. The pipeline described so far is a valid architecture for unsupervised, semi-supervised and supervised learning, though of course not all will have the same implementation.

Now we have learned what human activity recognition is about. We also saw that we are working in the unsupervised setting here. This means we approach it as a clustering problem. Consequently the next chapter briefly outlines the most important topics in clustering.

2.2 Clustering

In this chapter we will give an outline of clustering in general. We will give a general overview of the whole process, formally define the problem and then zoom in on the different components of the overall process. The bulk of this information stems from Jain and Dubes (1988), supplemented with additional sources.

2.2.1 Clustering: an overview

Clustering in general is the problem of trying to find structure in a dataset without the availability of class labels. Jain and Dubes (ibid.) use it interchangeably with "unsupervised learning" and so do I. Clustering can also be seen as classification without knowing any class-labels to train on. It is fundamentally a data exploration technique.

There are multiple ways of phrasing what a cluster is and different approaches on how to generate a clustering. Seeing that this whole problem is somewhat slippery it makes sense to approach it from different, but related angels, so as to get a better grip on it. Thus, let us have a look at some ideas of clusters:

A cluster can be described as a set of entities which are similar, and entities in other clusters are dissimilar to these. An alternative formulation sees samples as points in space: Clusters are sets of point such that their distances to each other are small, but to points on other clusters are large. A third formulation phrases it in terms of density: Clusters can be described as regions of high density. They are separated from other clusters by regions of low density.

It is worth noticing that which clusters that can be identified depends on the granularity of inspection. To illustrate: Imagine yourself inspecting a map of a country. You will probably find some larger structures within this country, for example administrative regions or large natural landscapes. Now, imagine zooming in on a smaller part of the map, for example a city. Smaller-scale structures will be revealed, but the overall pattern might be lost. One might be able to identify quarters and parts of the city, but not regions of the country anymore, even when moving around on the zoomed-in map.

By virtue of this property and the fact that distance — directly, but also indirectly through the notions of density and similarity — is part of all definitions of a cluster we can see that how proximity of data points is defined is crucial to what clusters we can discover. This is mirrored in the fact that overlap between clusters is one of the most critical factors in being able to

discriminate them, as we will discuss in Chapter 2.5.2.

There are two fundamental approaches to cluster discovery: hierarchical and partitional. In the partitional approach one tries to find one single assignment of class labels to data points.

The hierarchical approach is again split into agglomerative and divisive clustering. In the agglomerative setting all data points start out in their own one-element clusters. This means that for n data points there are n clusters.

In the next step, two of these clusters are joined. This results in a clustering with $n - 1$ clusters, one of which has 2 elements. Using this newly found clustering as a basis, the merging procedure is applied again. This procedure is repeated until all data points belong to one single cluster.

This process as a whole yields $n - 1$ different clusterings of the same dataset, one after each merge step.

One can then select which of these clusterings to use based on the number of clusters desired.

The divisive hierarchical clustering approach also produces a series of clusterings of the same dataset. However, it starts with all data points being in one cluster which is then iteratively divided.

In this current work we have only used agglomerative algorithms, thus we limit further exposure to this kind of hierarchical clustering.

These two approaches require two different formalizations, which is why they will be given in their respective chapters.

Within these two fundamental approaches there are multiple methods, different ways of defining proximity and also varying implementations of these methods. Unfortunately, "empirical evidence seems to be the only practical guide to the selection of clustering methods" (Jain and Dubes 1988).

To make matters a bit more difficult, most of these methods also work worse when the data do not obey some assumptions. For example, everything based on Euclidean distances has problems with data which have features on different scales. Consequently, some pre-processing is necessary to make the data usable in our analysis. We will look at this in more detail below.

So you have applied a clustering algorithm and found some clusters, or even a whole hierarchy of them, in your dataset. How do you know that these results are actually meaningful? This is the question of cluster validation, and crucial to clustering analysis. Unfortunately it is also one of the most difficult, as the notion of what a cluster is itself is very vague. Combined with the fact that defining a workable, testable, rational criterion for what kind of clustering is good is very hard makes the validation of such structures very difficult. We will tackle this topic in its own section below.

In general, one can identify the following scheme when trying to do cluster analysis:

1. Determine data representation
2. Preprocess data
3. Cluster data
4. Validate results
5. Iterate

This is an iterative process. After finishing some analysis we might learn more about the data, changing our fundamental ideas of what is going on, requiring updates to data representation or the cluster analysis, leading to different outcomes and so forth.

Let us now zoom in on the different building blocks. We will proceed in the order in which these steps normally are carried out: data representation and preprocessing, clustering, and then validation.

2.2.2 Data Representation and preprocessing

The data under consideration lies at the very heart of our cluster analysis. The way we choose to represent it and its particular properties will determine what kind of structure we can find. As such, a careful inspection of these factors is crucial. In this section we will investigate how data is typically represented in this problem setting. Furthermore, many of the clustering methods presented later make assumptions regarding the form of the data, e.g. a consistent scale across features. We will discuss preprocessing steps necessary to abide by these assumptions. The treatment of missing data is not included here due to its extent, see Chapter 2.3 for this topic.

Let us begin at the beginning: What are the fundamental notions when talking about data in clustering?

Fundamental data structures

How do we represent data? We shall call each thing we wish to assign a class to a "pattern". Names used elsewhere are "datum", "data point", "sample" or similar. Each pattern is a d-dimensional row vector, with d being the number of features recorded for this pattern. d is assumed to be the same for all patterns.

All patterns taken together, i.e. our dataset, is then represented by a pattern matrix:

Definition 4. *Pattern matrix: A $n \times d$ matrix. Each of the n rows is one of the patterns, each of the d columns one of the data dimensions.*

Let us call the space in which these datapoints lie the *pattern space*.

Given the pattern matrix and any measure of proximity, we can generate the *proximity matrix*.

Definition 5. *Proximity matrix:* $n \times n$ *symmetrical matrix. Element* $i, j = d(i, j)$ *for some distance measure* $d()$ *and patterns* i, j*.*

The proximity matrix can either be a similarity matrix or a dissimilarity matrix, depending on the measure that was used. Many methods use this matrix as their input, meaning that given the proximity matrix, knowledge of the actual pattern matrix might be unnecessary for clustering.

Data scale and type

What kind of data is recorded in these matrices? Data has two fundamental modes: *type* and *scale*.

The type of the data tells us something about permissible values. We discriminate between categorical ('running', 'cooking', 'singing'), binary (0, 1), discrete $(\dots, -2, -1, 0, 1, 2, \dots)$ and continuous (R) features.

It is worth noting that proximity matrices, too, have types. They can be binary (are data points in the same subset?), discrete (rank order among patterns) or continuous (e.g. Euclidean distance).

The scale of the data tells us something about the relationship between different values for this feature. It can be divided as follows:

1. Qualitative
   (a) Nominal: Numbers are arbitrary and code for categories
   (b) Ordinal: Numbers are arbitrary, but their ordering is meaningful
2. Quantitative
   (a) Interval: Numbers are measured on a scale and are meaningful with respect to this scale
   (b) Ratio: Numbers have absolute meaning. The ratio between numbers is meaningful. Example: distance in meter, temperature in Kelvin.

Proximity indices

There are plenty of different ways of computing entries in a proximity matrix. Let us call each such way *proximity index*. Again, following Jain and Dubes (1988), let us denote a proximity index as $d(i, k)$.

Most relevant to our use-case at hand are ratio-based proximity measures. A whole family of these is specified by the Minkowski distance:

$$d(i,k) = \left(\sum_{j=1}^{d} |x_{ij} - x_{kj}|^r \right)^{1/r} \tag{2.1}$$

For different values of r we get various useful measures:

1. $r = 1$: Manhattan distance. For binary patterns this is the Hamming distance
2. $r = 2$: Euclidean distance
3. $r \to \infty$: "sup" distance

Note that proximities might not be metrics (Tversky 1977).

Normalization

Normalization is a common step in preprocessing. It is made necessary by the fact that common clustering approaches, e.g. partitional clustering using Euclidean distances, are thrown off by uneven scaling of features. In this specific case, features on a larger scale would receive more weight when computing distances between points. In general, all preprocessing should be matched to the rest of the pipeline, but normalization is so commonplace that it deserves mentioned nevertheless.

Commonly, normalization is done by centering the data, i.e. shifting it so that its mean is 0, and scaling it such that its variance is 1. This is done on a per-feature basis. Let m_j be the sample mean and s_j^2 be the sample variance of feature j:

$$m_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} \tag{2.2}$$

$$s_j^2 = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - m_j)^2 \tag{2.3}$$

Then centering the data is done by:

$$x_{ij} = x_{ij} - m_j \tag{2.4}$$

And scaling and centering together is done as follows:

$$x_{ij} = \frac{x_{ij} - m_j}{s_j} \tag{2.5}$$

While this kind of normalization is very common it is neither the only available choice nor is it always appropriate. Other kinds of normalization involve scaling other factors. Examples include the range (Carmichael and Julius 1968) or a measure of heterogeneity (Hall et al. 1973). Normalization can be detrimental if it muddies the separation between natural clusters, compare Jain and Dubes (1988, Figure 2.5 on page 25) for an example.

2.2.3 Clustering methods and approaches

Introduction

In the previous section we have seen how to bring data into the required form to actually carry out any sort of clustering at all. In this section we dive right into the heart of the matter. We will give formal definitions for the different approaches to clustering. We will investigate multiple methods to find clusters and how they relate to error measures and properties of the data.

Clustering "is a special kind of classification" (ibid., p. 56). A clustering algorithm tries to assign each pattern to, typically, one out of k clusters. To gain an overview it is helpful to sort the many possible ways of doing so into a taxonomy. This taxonomy has the following levels: *approach*, *method*, *algorithm* and *implementation*. The meaning of these levels will become clear in the following paragraphs.

As Jain and Dubes (ibid.) stress, a method is different from an algorithm. A method describes, on an abstract level, how to cluster things. An algorithm gives concrete steps. An implementation, finally, is an actual computer program that can be run. These differences are important: one method can be described by different algorithms, yielding different results. And two implementations of the same algorithm again might yield different clusterings, e.g. based on the way ties in proximity are broken.

At the highest level, there are two common approaches: hierarchical clustering and partitional clustering. Partitional clustering finds exactly one partitioning of the data (however, patterns might belong to more than one cluster or belong to clusters to varying degrees depending on the method used). In contrast, hierarchical clustering produces a nested structure of these individual partitionings. Starting from one individual cluster per pattern these clusters are gradually combined until one cluster for the whole dataset remains. Each intermediate step is a partitioning at a different level of granularity. Our focus will lie on partitional clustering, as this is the tool of choice for our problem at hand.

Partitional clustering

Partitional clustering is the task of finding one partitioning of the data into clusters, in contrast to the multiple nested partitionings discovered by hierarchical clustering. Again opposed to hierarchical clustering it usually does not operate on a proximity matrix, but a pattern matrix. Let us then formally define the problem we wish to solve in terms of these patterns Following Jain and Dubes (1988, p. 90):

Definition 6. *Partitional Clustering Problem: Given n patterns in d-dimensional metric space, determine a partition $C = (C_1, \ldots, C_K)$ into K groups, such that the patterns in one cluster are more similar to each other than to patterns in different clusters.*

The principal difference between partitional clustering methods stems from the use of different *clustering criteria*, which measure the goodness of such a partitioning. This means, it needs to somehow capture the meaning behind "more similar to each other than to patterns in different clusters". We divide these criteria into local ones and global ones (ibid., p. 90). Global criteria try to find a partition through the use of prototypes. Each pattern is assigned to, and consequently represented by, such a prototype. Local criteria, on the other hand, inspect the region around each individual pattern. Properties of this, as of now somewhat vaguely-defined, region might be the density or the classes assigned to neighboring points. Choosing a criterion is always problem-dependent, and finding the right one can be a major hurdle in cluster analysis.

As an abstract, theoretical problem the Partitional Clustering Problem is straightforward to solve. Simply try each possible clustering and choose the one maximizing our clustering criterion of choice. One could find the optimal clustering C_O under criterion $f()$ and pattern matrix X as:

$$C_O = \arg\max_C f(C, X) \tag{2.6}$$

However, this is infeasible for real problems. As Jain and Dubes (ibid.), citing Fortier and Solomon (1966) and Jensen (1969), show, the number of possible clusterings is given by Stirling numbers of the second kind. They demonstrate that even for only 19 objects and four clusters the number of possible clusterings lies at 11,259,666,000. It is obvious that simply searching through all possible solutions is not manageable.

So what to do instead? Faced with this problem we turn to iterative optimization methods. There are alternatives that try to limit the search space, e.g. by dynamic programming (Jensen 1969). However, they still run

into complexity issues. As such we shall focus on iterative, hill-climbing-based methods. They commonly work by starting out with some initial partition and then modifying clusters, e.g. by transferring a point from one cluster to another, or by merging two clusters into one. Again, choices have to be made in a large space, and so heuristics exist both for choosing initial partitions and for modifying clusters. These steps are typically iterated until some sort of stopping criterion is satisfied, e.g. a change in the performance falls under a certain threshold or clusterings do not differ a lot across various steps.

Unfortunately, this method also inherits all of the typical problems found in iterative solutions to non-convex optimization problems. It can get stuck in local optima, it is somewhat hard to judge when a solution is good enough and the runtime necessary to find such a suitable solution is hard to predict. The usual tactics for dealing with these apply, e.g. doing multiple runs with different initializations or setting a hard limit on the number of iterations.

Now that we have some idea of how to optimize a criterion we are puzzled by the question: How do we choose which one to use? Unfortunately, '[t]here is no single "best" criterion for obtaining a partition because no precise and workable definition of "cluster" exists' (Jain and Dubes 1988, p. 91). The problem is as follows. In each new analysis, the structure we wish to identify in the data, if it exists at all, might look differently. At the very least it depends on how the data is encoded and scaled and on the nature of the grouping we wish to discover. Unfortunately, only if we use a criterion that discovers structures close to the true groups we are looking for is it possible to discover them.

There exists an unending multitude of available criteria. Only a few of them are well-understood, intuitively and mathematically. Many criteria are rediscovered across different fields, or are equivalent to ones that are already known (Urquhart 1982).

In this chapter we briefly reviewed the essential points of clustering, viz. how the overall process typically works, how data is represented and processed and which kinds of clustering methods and approaches there are.

All of this tacitly assumes a complete dataset, i.e. one without missing values. Or, at the very least, a complete distance matrix. However, real world data is often not in this neat state. Sensors malfunction, connections are dropped and readings are corrupted. To properly understand how to deal with these missing values it is crucial to first understand why it is missing, as this determines the impact that the missingness has on downstream analyses. The next chapter consequently outlines the three categories of missing data.

2.3 Missing Data

Data goes missing due to a lot of different reasons. In our specific case, sensors might malfunction, or the transmission anywhere along the data collection/processing pipeline might be faulty. To make matters a bit more difficult, missing values might not always be clearly identifiable as such: a sensor might just produce bogus readings for a while. But let us focus on the case that missing data has been detected. What do we do with it?[1].

The answer to this very much depends on the mechanism behind the missing data. There are three main classes to be discriminated: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Each of these is described in more detail in their own chapter below.

Globally speaking, data that is not missing completely at random introduces a bias into every subsequent analysis if not being approached properly. On the other hand MCAR data does not have this effect and may be dealt with by simply eliminating patterns with missing data. Why is this so? A closer look at these types will tell us.

2.3.1 Missing Completely At Random

MCAR data is one where whether a data point is missing or not does not depend on the data at all. For example, if all sensors are afflicted by random wireless outages to the same degree. This means that missingness is related to neither the actual sensor data nor any of the missing data. Thus whatever data is left is just as representative a sample as the complete data set. However, it does reduce the power of any following analysis. In this scenario, it is generally safe to simply delete all data points having missing entries. This is known as list-wise deletion.

Let's introduce some notation for this. Let Y be the data we wanted to collect. This decomposes into Y_o, the data we actually observed, and Y_m, data that is missing. Let $P(r)$ be the probability that data is missing. Then missing completely at random can be formalized as follows:

$$P(r|Y_o, Y_m) = P(r) \tag{2.7}$$

[1]The global structure of this section is based on Wikipedia entry for missing data `https://en.wikipedia.org/w/index.php?title=Missing_data&oldid=757211169` and on the "Introduction to missing data" by the London School of Hygiene and Tropical Medicine `http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=category&id=40:missingness-mechanisms&Itemid=96&layout=default`.

The latter is also origin of the formal definitions. Other sources are cited separately.

Meaning, that the probability of data missing is both independent of the observed and the missing data.

2.3.2 Missing At Random

MAR has somewhat less strict of a condition. Data is missing at random, if missingness is independent of the missing data. Formally:

$$P(r|Y_o, Y_m) = P(r|Y_o) \tag{2.8}$$

This is somewhat harder to grasp intuitively. Building upon the wireless signal outage example from above, imagine that signal strength depended on location. So for location A, there would be no or little missing data, whereas for location B missing data might abound. In this case, data is not missing completely at random. However, it is random for any given location. Meaning: given the location, missingness is random.

In practice, one hardly discriminates between MCAR and MAR. In both cases, list-wise deletion can be appropriate tool given that enough data is left to conduct a meaningful analysis.

2.3.3 Missing Not At Random

Finally, data can be missing not at random (MNAR). This covers all other cases. Say, for example, that for some activities like swimming users never wear a step tracker or always leave their smartphone at home. This means discarding this data would strongly bias every resulting model against finding such activities. In the case of an exploratory analysis it would even hide that these activities had taken place at all. Thus, list-wise deletion is not an appropriate option.

In this chapter we found that the missingness mechanism critically determines what effect missing values have on subsequent analyses. However, we did not answer the question of how to actually handle missing values given that we know how they came about. This is the topic of the following chapter.

2.4 Strategies for Dealing with Missing Data

Once missing data and their underlying missingness mechanism has been identified one can begin to actually do something about this and enable further analyses. This chapter outlines some of the most important strategies for doing so and links them to the previously discussed missingness mechanisms.

Jain and Dubes (1988) lists Kittler (1976), Sneath, Sokal, et al. (1973), and Zagoruiko and Yolkina (1982) as sources of treatments for missing data. A useful summary of approaches is given in Dixon (1979). More modern takes can be found in Baraldi and Enders (2010) and Schafer and Graham (2002).

Broadly speaking, we can identify three main ways of handling missing data for the context of this thesis:

1. Delete data
2. Use a model which can directly take missingness into account
3. Fill missing data by imputation

Let us have a look at each of these in turn and see how they would apply to our current scenario of finding activities in sensor data.

2.4.1 Deleting Data

Deleting data can be done in multiple ways. One may decide to drop all samples with one or more missing features. This is known as list-wise deletion or as complete-case analysis. If data is missing at random or completely at random this can be done without introducing any bias into the analysis. However, the lower number of samples leads to a reduced power of said analysis. In our concrete case this could mean not being able to detect an activity in the sensor data, even if it is in there. List-wise deletion is very easy to implement and in many cases the standard behavior of software packages. It is widely used. However, it is essential to consider the mechanism behind missing values to determine whether it is actually appropriate.

It is also possible to delete any feature that has missing values. However, great care must be taken in this case to not delete essential, highly informative features. Determining this, however, can be non-trivial. This is due to the fact that many methods for determining how valuable a feature is, e.g. Chi square tests, themselves require the data to be complete, i.e. without missing values. For the current use-case in this paper this also hardly seems like a good option. Assume, for example, that there was a short signal loss for the sensor units connected to a participant's leg. Following this approach for handling missing data we would now delete all features this sensor generated, losing all information about leg movement.

Finally, there is also the option of selectively excluding missing values only for analyses where this particular features is actually used. This is called pairwise deletion or available case analysis.

2.4.2 Direct Modeling

It is also possible to approach datasets with missing values using modeling approaches which can intrinsically handle them. Consider for example the work done in Wilson (2015, p. 24).

The authors applies a Gaussian Mixture Model to a dataset with missing values. To do so, the EM process employed during model fitting is adapted to be able to handle these missing values. They exploit the fact that marginalizing a Gaussian distribution yields another Gaussian in return. This means that the rest of the procedure can stay the same, it will only be carried out for a lower number of features.

This essentially means that for incomplete samples, only known values are being used, without inferring missing ones. Complete samples are used in their entirety. This could also be seen as a way of integrating pairwise deletion directly into a model.

The success of using this method will be discussed in more detail in Chapter 2.5.

Alternatively, one might also explicitly include a missingness mechanism in a model. For example, belonging to a class k could bring with it both a distribution over possible values and Poisson distribution determining the number of missing values. Only if a draw from the Poisson distribution indicates that the value is not missing will that value actually be sampled.

2.4.3 Imputation

Imputation means filling the missing values in the data. There is a wide variety of ways of doing this.

Static Value

One may simply choose an arbitrary static value that represents missing data and use this to fill missing values. If the feature to be filled happens to be categorical one effectively introduces a new 'missing' category. However, in all other cases this strategy will introduce a bias towards the chosen value and is thus not suitable. For example, if one were to replace all missing values in the X-axis acceleration of a sensor by the arbitrary value 7 one would distort all future analyses to also include this value, or place a higher-than-justifiable weight on it.

Imputation Based on Known Values of The Same Feature

As an alternative one may choose to fill missing values with a measure of the average of that features, for example the mean or the median. In doing so one can be sure not change said average, but might still over-represent it. There is another issue regarding this approach for our current problem domain. Sensor data is inherently time-dependent. As such, filling with the overall mean will blot out any local temporal trends, which is exactly what we are trying to detect.

To account for this, one can use methods such as rolling window functions or forward-filling. Windowing functions means taking a contiguous sample of w data points, where w is the window length. Within this window, a function of interest, for example the mean, is computed. This mean is then used to fill missing values within the current window.

There are at least two interrelated points to consider for this method. Firstly, the choice of the window length w. Smaller values for w lead to capturing more local trends. However, this might also mean more sudden jumps in the imputed value. Larger values for w will generate a smoother imputation, but might miss local trends.

Secondly, there might be windows for which all (or most) data is missing. This is more likely to occur for shorter windows. In these cases, estimates become less reliable. This can be avoided by choosing w larger than the longest streak of missing values, again at the cost of capturing local trends.

Forward-filling means taking the last known value and using this to fill all missing values until another known datum is discovered. Backward-filling follows the same definition, *mutatis mutandis*. These have the advantage of being able to deal with long streaks of missing value automatically. However, care must be taken to not blindly accept these values. If the last known value is far removed in time from the currently to-be-filled missing datum it might not be an accurate representation. What constitutes 'far' is relative to the problem domain. For measurements of outside temperature three minutes might not make a huge difference. For measurements of acceleration during activities even three seconds might yield vastly different values. Generally speaking, forwards-/backwards-filling are reasonable approaches for an inherently smooth signal.

Imputation Based on Known Values of Other Features

As an alternative one may try to infer missing value of one feature based on known values of other ones. For example, one might build a linear regression model based on X and Y acceleration to predict missing values in

Z acceleration. However, this also means constructing feature values as direct linear combinations of other features, thereby increasing correlation between the filled values and the known values it is based on.

Multiple Imputation

A shortcoming of all of these techniques is that they do not take take the uncertainty into account that is introduced by estimating missing values. After imputation, filled values look just like measured ones and are given the same weight in any subsequent analyses. A way of overcoming this problem and including uncertainty information into analyses is using multiple imputation, initially introduced by Rubin Donald (1987).

The central idea behind multiple imputation is to capture and quantify the uncertainty associated with imputed values and take it into account for analysis outcomes. The general approach is to impute values K times, each time by drawing samples from a distribution. This will lead to K complete datasets. The analysis is then carried out once for each of these datasets, leading to K results, each with its own measure of uncertainty. These will then be combined into one final result and uncertainty measure.

Let's consider an example: our independent variables are X, Y and Z acceleration, as measured by a sensor attached to a person's leg. From this we wish to infer heart rate using some model M. Assume that X and Y data are complete, but that Z has missing values.

In a single-imputation paradigm we could now, e.g., build a regression model of Z, dependent on X and Y. We would use this to infer values for Z, leading to one complete dataset. On this we could then fit M, giving us estimates for the heart rate.

In a multiple-imputation paradigm the procedure would change as follows. Instead of building a single regression model connecting X, Y and Z one would instead build K different ones. For each such model the intercept and coefficients would be drawn from a multivariate distribution.[2]

Missing values would then be computed by first drawing from the Gaussian distribution of residuals and then adding these draws to the regression result.

We could then use each of these K resulting datasets as we normally would, yielding K different heart rate predictions and variances for each X, Y, Z pair.

[2]The explanation of multiple imputation is based on the London School for Hygiene and Tropical Medicine's "Introduction to multiple imputation" `http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=category&id=43:introduction-to-multiple-imputation&Itemid=96&layout=default`.

Results are then combined in the following way: The final prediction H_{final} is the average of the K individual predictions.

$$H_{final} = \frac{1}{K} \sum_{k=1}^{K} H_k \tag{2.9}$$

The variance of H_{final} is composed of the within-imputation variance σ_w^2 and between-imputation variance σ_b^2, defined as follows.

$$\sigma_w^2 = \frac{1}{K} \sum_{k=1}^{K} \sigma_k^2 \tag{2.10}$$

$$\sigma_b^2 = \frac{1}{K-1} \sum_{k=1}^{K} (H_k - H_{final})^2 \tag{2.11}$$

Leading to the final variance:

$$\sigma_{final}^2 = \left(1 + \frac{1}{K}\right) \sigma_b^2 + \sigma_w^2 \tag{2.12}$$

For more information information see also Schafer and Graham 2002.

In this chapter we saw various ways of handling missing data and also learned when they are appropriate given the missingness mechanism. But how do they impact the resulting clustering in practice?

The next chapter reviews existing research into exactly this question.

2.5 Missing Data and Clustering

Little work has been done on the intersection of clustering and strategies for dealing with missing data. By this we mean systematic investigations into the effects of various missing data strategies on clustering of the resulting datasets. This is an understandable development: cluster evaluation is a non-trivial problem by itself. Layering missing data and cluster evaluation makes it difficult to be confident about any findings.

Thankfully, Wilson (2015) braved these challenges. Their work compares the following missing data strategies on both artificial and real-world datasets:

1. Complete case analysis/list-wise deletion (followed by Gaussian Mixture modeling as implemented in the mclust package for R(Fraley and Raftery 2002))

2. 45-times multiple imputation, as implemented in the MICE package for R(Buuren and Groothuis-Oudshoorn 2011)

3. Direct modeling: hierarchical clustering in the form of Ward Clustering

4. Direct modeling: Adapted Gaussian Mixture Model

The authors present a modified version of Gaussian Mixture Models whose Expectation Maximization (EM) fitting procedure has been adapted to be able to handle missing data.

They investigate the effects of various problem parameters on clustering performance, for example the fraction of data that is missing or how much clusters overlap.

This directly applies to the research at hand, warranting a longer exposure. In what follows we will first have a brief look at how the EM procedure was changed. This enables us to actually interpret that paper's results. Subsequently, we shall provide a succinct summary of the work's main results.

2.5.1 The Adapted Gaussian Mixture Model

The authors apply a Gaussian Mixture Model to a dataset with missing values. This model is typically fit using an EM process. To be able to fit the model in the presence of missing values, this EM process is adapted. During the E step, densities of belonging to the different clusters are being computed. If a sample with missing features is encountered the rest of the computation is carried out using a marginalized version of the joint distribution over all features.

How is this done formally? The authors illustrate this on Page 24 of their paper:

Imagine a case for three features $1, 2$ and 3. Normally, one would compute the densities using the mean vector

$$\mu = [\mu_1, \mu_2, \mu_3] \tag{2.13}$$

and covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{2.14}$$

However, assume now that a pattern is encountered for which the second feature dimension is missing. In this case one would simply drop the missing feature dimension from all computations. This means one would use the following mean vector:

$$\mu = [\mu_1, \mu_3] \tag{2.15}$$

Furthermore, the covariance matrix would be transformed to the following shape:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{13} \\ \sigma_{31} & \sigma_{33} \end{bmatrix} \tag{2.16}$$

This means that for incomplete samples cluster assignment is based only on the known values. However, the authors point out that this does not bias in favor of any of the clusters, as assignment to all of the clusters is based on this reduced information.

The changes to the M step are somewhat more involved, but follow in the same spirit. In general a fair intuition about this algorithm is that it defaults to a lower-dimensional mixture model for all samples with missing values. We refer the interested reader to Chapter 5 of Wilson (2015).

2.5.2 Experiments, Insights and Practical Advise

What insights do the authors gather from studying the relationship between missing data strategies and clustering? Before we dive in a qualification is in order: results only apply to multivariate, quantitative, continuous data where data is assumed to be missing at random.

The study contains two primary experiments:

1. Cluster discovery
2. Classification accuracy

The findings for both experiments agree. Consequently, let us have a look at each of these experiments in turn before turning to the outcomes.

Cluster Discovery Experimental Setup

The cluster discovery experiment remains somewhat nebulous. The description, found on Page 33 of Wilson (ibid.) simply states that "[t]hree methods were tested for their ability to correctly identify the number of components of the data in the presence of MCAR missing values". This does not specify how these methods actually identify the number of components.

The answer lies in the clustering methods used. They employ their self-developed Adapted Gaussian Mixture, which seems to be capable of providing an estimate of the number of components. In addition they also make use of the R package mclust Fraley, Raftery, et al. (2012). This package implements Gaussian Mixture Models and also enables the estimation of the number of components.

These methods are used in combination with three different ways of handing missing data, resulting in the following approaches:

1. Complete case analysis used in conjunction with mclust
2. 45-times multiple imputation used with mclust, where the most common inferred number of components was used as the final result
3. Adapted Gaussian Mixture Model

In the experiment a data set was presented to each of the three methods. Performance was measured as the method's ability to correctly identify the number of true components in a synthetically generated dataset.

Each method was run multiple times for slight variations in the parameters of the dataset. This means that for each dataset quality of interest the other parameters are held constant. This parameter, e.g. overlap between clusters, is then varied. The method under examination, say complete case analysis, is then run for each setting of this parameter, yielding one measurement point.

To properly interpret results it is necessary to also take into account the dataset which was used to generate these measurements. Let us have a brief look.

Dataset

The dataset parameters which are varied between clustering runs are:

1. Dimensionality, using values 3, 5, 8, and 12
2. Amount of missing data, using values 10%, 15%, 20%, 25%, and 30%
3. Cluster separation, using values 0.001 (well-separated clusters), 0.015 (intermediate), and 0.05 (large overlap)

The dataset has been generated using the R package MixSim (Melnykov, Chen, and Maitra 2012). It generates data drawn from a mixture of Gaussians, allowing for the control the dataset parameters mentioned above. Note that this implies that the Gaussian Mixture Models tested by the authors actually exactly match the true method of data generation.

Furthermore it is noteworthy of the dataset that clusters are non-spherical, making them harder to detect. However, all components are homogenous, decreasing difficulty. The true number of components was always fixed to 3. Interestingly, the number of datapoints was set to 60.

In modern terms this is a tiny dataset, in terms of dimensionality, number of components and especially number of samples. The experimental results

presented in this paper make intuitive sense, but should always be seen in light of this fact. They might not generalize to scenarios in even higher dimensions, much larger datasets or a highly increased count of clusters as is not uncommon.

With this established, let us briefly touch upon the setup for the classification accuracy experiment before diving into the results.

Classification Accuracy

The goal in the classification accuracy experiment was to sort all samples into the correct clusters, given the true number of components. They compare the following three methods:

1. Hierarchical clustering
2. 45-times multiple imputation, where the final performance is the mean of all 45 performance measurements
3. Using the novel Adapted Gaussian Mixture Model they developed

A point that remains unclear is how they compare cluster indices to ground truth classes. The issue here is that cluster index 3 might perfectly correspond to true class label 1, which means just comparing their numerical identity is not meaningful. They briefly mention "solving the relabeling issue" (p. 34) as part of a sentence but do not explain this in any more detail.

Our best guess is that they tried all possible mappings from cluster indices to class labels and used the one which lead to the highest accuracy.

For the remainder this experiment operated on the same dataset as the cluster identification one and varied the same parameters.

Having established the setup and the introduced the dataset let us now have a look at what the authors of that study learned.

Experimental Results

In what follows we will give a short outline of the primary conclusions the authors draw from the experiments described in the sections immediately preceding this one.

What did they find out regarding the most important factor on clustering performance? They found that overlap had the strongest effect on performance for all missing data strategies, for both experiments. Higher overlap between clusters always severely degrades performance.

Based on this, we would expect feature engineering to be critical to the overall success of any cluster analysis. It might help in alleviating the problem

of overlapping clusters, for example by adding another dimension in which clusters are easily separable.

If additional features are not available then using kernel-based techniques, which internally project data into higher-dimensional spaces, might be successful due to the same reasons.

How do the different missing data strategies deal with changes in overlap? Between strategies, using the adapted version of the mixture models showed the steepest decrease in performance for increased overlap in clusters. Their hypothesis is that the information available without imputation is insufficient for properly assigning patterns to clusters in this scenario.

Using the complete case approach samples with missing data is deleted. This approach still suffers less from increasing cluster overlap than the novel Adapted Gaussian Mixtures. Based on this the authors hypothesize that not having any information at all can be better than making a clustering decision based on incomplete data. They argue that incomplete data can actually be misleading.

Multiple imputation, on the other hand, handles an increase in cluster overlap the best. The authors offer the explanation that multiple imputation might borrow information from complete data samples to fill the gaps. If the complete data samples are informative enough this borrowed information might help in clustering.

This makes sense but is also not a very insightful explanation. In fact, the very idea behind basically all methods of imputation is to estimate missing values from complete data samples, implicitly assuming that this might be useful for a future analysis. As such they simply state that they think imputation as a principle might work.

The impact of increasing dimensionality as well as increasing the fraction of missing data is not as severe and the insights not as deep. Consequently, we would like to only present the most interesting bits of that study's results.

Firstly, let us have a look at the impact of dimensionality and sample size on complete case analysis. Complete case analysis shows disproportionately bad performance when the sample size is small and dimensionality is high. To clarify: all methods show bad performance in such a scenario. However, list-wise deletion suffers more than the Adapted Gaussian Mixture Model or multiple imputation in terms of relative decrease of performance. However, so the authors, real-world data might have a large sample size and small dimensionality, making list-wise deletion a viable choice and easy-to-use for data that is missing at random.

We would like to comment that it is questionable if this is actually true. While real-world datasets are growing fast in the days of big data analysis so are the feature sets and thus the dimensionality. Building models with

hundreds of features is not uncommon, especially when also considering interaction terms.

Secondly, let us touch briefly upon the merits and shortcomings of hierarchical clustering. Hierarchical clustering can deal with missing values as it operates purely on a distance matrix, not the data itself. It is, however, not apparent from the paper how the distance matrix was computed while values were missing. The authors find that hierarchical methods perform extremely poorly on a real-world dataset they tested in addition to the synthetic one discussed above.

Finally, we summarize the practical advise the authors distill from their experiments. In conclusion, the authors find that multiple imputation offers superior performance to all other methods, followed by complete case analysis and finally their direct modeling approach of using adapted Gaussian Mixture Models.

In this chapter we outlined prior clustering research for datasets with missing values. This concludes the background section of this work. The next chapter dives into the experimental work by discussing the discovery of subjects in PAMAP2 through clustering.

Chapter 3

Discovering Subjects in PAMAP2

When faced with the task of unsupervised activity recognition a crucial issue is that in practical applications, ground truth labels are not available. However, subject IDs are. Based on this, our first venture into clustering sensor data is actually an experiment trying to identify subjects rather than activities. The fundamental idea here is that subjects carry out activities characteristic to them, leading to personalized patterns of sensor activity. Activities, so to say, can act as fingerprints, identifying subjects.

For proper context this section contains a description of the dataset, followed by an analysis of the different stages of our experiments.

3.1 The PAMAP2 Dataset

PAMAP2 (Reiss and Stricker 2012) consists of streams of wearable-sensor-data. Nine subjects (101 through 109) each followed the same protocol of activities, prescribing what activities to do, for how long, in what order. Activities ranged across standing, watching TV, playing soccer, driving a car and others for a total of 18 activities. While following this protocol subjects were each equipped with three Colibri wireless IMUs and a heart rate monitor. Each of these positions held one IMU: wrist of the dominant arm, the chest and the dominant side's ankle.

Heart rate was measured at circa 9Hz. The IMUs supplied the following measurements at 100Hz:

1. temperature
2. 3D acceleration data (x, y and z directions) at 16g scale

3. 3D acceleration data at 6g scale

4. 3D gyroscope data

5. 3D magnetometer data

6. orientation (invalid in this dataset)

Across all subjects this leaves us with 376417 samples and 52 features (when not counting meta-features such as activity ID or timestamp).

Following common practice a windowing function was applied to this dataset using a 1-second, non-overlapping window. This effectively yields a new dataset, on which the experiments described below were based. These are the features of the new, windowed, dataset:

1. For each 16g sensor, for each direction in x,y,z: take the mean of values over this window. The new feature could, e.g. be: ankle_x_mean.

2. Same, but take the standard deviation instead

3. For heart rate: Upsample by forward-filling with most recent non-missing value.

4. For each sensor in hand, ankle, chest: Create a new feature as the sum of the x,y,z standard deviations in this window, serving as an overall indicator of activity.

A window can potentially cross the boundary between activities. It gets assigned to the most common activity present in the window. Windowing was done per subject, so windows do not 'roll over' from one subject to the next one.

All other features (acceleration at 6g, orientation, magnetic resistance, temperature and gyrometer) are excluded. Out of these, 6g acceleration and orientation are recommended to be removed by the official description of the dataset. The 6g sensors often get oversaturated in fast movements. Orientation is invalid in this experiment. The other features were excluded as acceleration is better understood and easier to preprocess and interpret.

In this experiment the aim was to capture subjects as clusters. Our motivation for this lies in cluster validation. While true activities are not always available (and, in fact, are not in the primary applications of this research), subject identifiers are. The idea was thus to use sensor data akin to fingerprints, trying to predict what subject generated these sensor readings. We also investigated the effect of missing data on this task.

3.2 Experimental Setup

3.2.1 General Setup

The general setup was as follows: The whole dataset was used at once. Using all available features, we induced a pre-defined amount of missing data, $p\%$. This was done by randomly setting $p\%$ of the samples per feature to missing. This protocol leads to data that is Missing Completely At Random (MCAR). Subsequently, we imputed missing data by replacing it with the mean of the available data. The dataset was then standardized.

The thusly prepared dataset was then clustered. We used the true number of clusters, i.e. the number of subjects, as a parameter for the clustering algorithms. This yielded a cluster index for each sample. Performance was calculated by comparing these cluster indices with the ground truth given by the subject ID.

3.2.2 Cluster Evaluation

We used the Adjusted Rand Score as a performance measure. Its ubiquitous use through this paper warrants a closer look. As the name would indicate the Adjusted Rand Score is based on the Rand Score. Understanding this will make understanding the adjusted version a straightforward extension.

Let us begin with the intuition of the Rand Score (Rand 1971) before going towards a more formal treatment. This score measures the agreement between two clusterings. It is, as it were, an extension of the idea of accuracy used in supervised learning to the unsupervised realm.[1]

It measures agreement by comparing all pairs of data points. For each pair of points a clustering can either sort them into the same or into different clusters. Two clusterings agree if they both decide the same, i.e. both clusterings group this pair of points into a cluster, or both clusterings split this pair of points into two separate clusters.

But how can this measure then deal with the fact that two clusterings might even find the same clusters, but assign different indices to them?

It is essential to note that the particular names, indices or identities of the clusters do not matter. Two points could both be grouped into cluster 5 under clustering C_A and into cluster 7 under clustering C_b. These indices are completely independent of each other - whether they are the same or different carries no additional meaning.

What matters is that under both clusterings, the pair gets either sorted into one single cluster, or split apart.

[1] `http://i11www.iti.uni-karlsruhe.de/extra/publications/ww-cco-06.pdf`

The Rand Score is then computed as the fraction of pairs on which the two clusterings agree. In our case, one of these two clusterings is actual ground truth, meaning we compare the clustering found by our model to true activities. More formally, the Rand Score is then:

$$R = \frac{\#agreements}{\#pairs} \tag{3.1}$$

This score ranges from 0 for perfect disagreement to 1 for perfect agreement.

While intuitively useful, it has some shortcomings. Firstly, it does not account for agreements that occur randomly. Secondly, the expected value of the score increases with the number of clusters.

To correct for these facts, Hubert and Arabie (1985) extended this measure to the Adjusted Rand Score. Again, let us begin with the intuition.

Conceptually, the whole change is that agreement is reduced by the expected agreement under random cluster assignment. This happens both for the measure agreement and for the maximally possible agreement, i.e. both for numerator and denominator of Equation 3.1.

As such, our score now reflects how much two clusterings agree compared to random clustering of the dataset.

Formally, this is done by assuming that the two clusterings are both drawn from a generalized hypergeometric distribution. It is assumed that there are fixed numbers of clusters with fixed sizes.

Under this assumption one can now calculate the expected Rand Score for random clusterings $E[R_{random}]$. Using this, we can formulate the Adjusted Rand Score AR as:

$$AR = \frac{R - E[R_{random}]}{R_{perfect} - E[R_{random}]} \tag{3.2}$$

Here, $R_{perfect}$ is the Rand Score for perfect agreement, i.e. 1.

The resulting measure AR has a range of 0 through 1. Here 0 indicates that agreement between the two clusterings is in line with random assignment. In our case, where one clustering is the ground truth, it means that results of the clustering model are no better than random. On the other end of the scale 1 indicates perfect agreement between ground truth and cluster model.

It is possible in theory to obtain a negative Adjusted Rand Score.

When using the Adjusted Rand Score it is important to keep in mind that it is based on assumptions regarding the clustering procedure, as outlined above. These are not necessarily true. In fact, it often happens that clusters are not evenly sized, violating one of the assumptions.

It has nevertheless proven to be a useful measure in practice, having both the advantages of being conceptually similar to the well-understood accuracy and of taking into account random clustering agreements.

3.3 Models

We employed the following models: K-Means (KM), Mini-batch K-Means (MBKM)(Sculley 2010), and a Gaussian Mixture Model (GMM) The following paragraph contains short descriptions of each of these so that results can be explained properly in the light of the nature of these algorithms. In what follows, we will give brief outlines of all of these models as well as how they are fitted to the data.

3.3.1 K-Means

K-Means minimizes within-cluster distance yielded by assigning each datapoint to one of k cluster centers. As this problem in general is NP-hard it is solved by heuristics, most commonly by choosing k random initial cluster center positions. The algorithm then iterates between two steps: First, all datapoints are assigned to their closest cluster center. Secondly, a new cluster center for each cluster is computed based on the updated assignment of points. This cycle continues until a stopping condition is reached such as a stable clustering, small changes between iterations or the maximum number of iterations. This way of implementing KM is also known as Lloyd's Algorithm (Lloyd 1982). KM is biased towards evenly-sized, spherically-shaped clusters. It cannot detect non-spherical clusters. Due to its greedy and heuristic nature it will find locally optimal solutions, depending on the positions of the initial guesses of cluster centers. To counteract this it is typically initialized multiple times and the overall best run is used, as measured by within-cluster distance across all datapoints.

After having outlined the general idea, let us formalize this to rule out any ambiguity.

KM has as a goal to minimize within-cluster distances, meaning the sum of distances between cluster centers and all points associated to that respective center:

$$\text{within-cluster distance} = \sum_{c}^{C} \sum_{i \in c} ||x_i - \mu_c||^2 \tag{3.3}$$

Here c is one particular cluster out of all clusters C. $i \in c$ indexes all

points in cluster c, with x_i being one such point in particular. μ_c is the mean of cluster c.

Once at the start of the algorithm k initial cluster centers are assigned. Within the thesis we use the "k-means++" scheme to do so (Arthur and Vassilvitskii 2007). It promises a better initial seed by picking points which are farther apart than a random selection of data points.

Following this initial assignment, two update steps follow until convergence. First, for each cluster a new center is calculated as the mean of all points assigned to it:

$$\mu_c = \frac{1}{|c|} \sum_{i \in c} x_i \tag{3.4}$$

Here, $|c|$ is the number of points in cluster c.

Secondly, all points are assigned to their closest cluster median, based on the updated values computed in the previous step.

$$x_i \in c_a \iff \forall c_b \in C : ||x_i - \mu_c^a||^2 \leq ||x_i - \mu_c^b||^2 \tag{3.5}$$

3.3.2 MiniBatchK-Means

MiniBatchK-Means is an extension of KM with the goal of reducing use of computational resources. There are two principal differences to regular KM. First, mini-batches are drawn from the dataset. Typically one draws hundreds or thousands of these mini-batches. For each minibatch updating the cluster means is then done on a per-datapoint basis. This means that for each datapoint in the minibatch the following two steps are carried out, which constitute the update rule for this algorithm:

1. Determine the nearest cluster center c_i to the current datapoint x_j
2. Update c_i as a streaming average with x_j

An overall effect of this is that the rate of change for each cluster median is decreased. MBKM is considerably faster than KM. The same caveats apply. MBKM is a promising solution for using K-Means at a large scale. However, it demonstrated inferior performance to K-Means in some of our test cases, which is why we chose to make it part of the overall analysis.

3.3.3 Gaussian Mixture Models

Gaussian Mixture Models assumes the data to be generated by a mixture of (possibly multivariate) Gaussian distributions, also called components. This can be written as

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

Here K is the total number of components, π_k is the mixing coefficient for component k, and μ_k and Σ_k are the mean and covariance parameters for that component.[2]

To properly introduce inference in this model we need to rephrase it in terms of a latent variable z. It can be useful to think of z as a vector indicating, for a given observation, what component generated this observation. Consequently, it is a K-dimensional binary random vector for which holds that exactly one of its values is 1, the other ones being 0. It is linked to the mixing coefficients π as follows:

$$p(z_k = 1) = \pi_k \tag{3.7}$$

With this in place we can now introduce a term representing the responsibility of component k for explaining an observation x. This is the conditional likelihood of z given x:

$$\gamma(z_k) \equiv p(z_k = 1|x) \tag{3.8}$$

This gives us all the setup we need to employ the Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977; McLachlan and Krishnan 2007) to infer parameters of a Gaussian Mixture Model.

The EM algorithm can be used as a way of finding maximum likelihood solutions to latent-variable models. Let us first have a look at the general outline of the algorithm. A more formal description of the update rules for Gaussian Mixture Models will be provided afterwards.

The parameters we wish to estimate for a Gaussian Mixture Model are the mixing coefficients π_k, the Gaussian means μ_k, and the Gaussian covariance matrices Σ_k.

The EM algorithm is initiated by setting some arbitrary starting values for these parameters. These can be random but could also be informed by, e.g., a prior clustering run using a less expensive model such as K-Means. In this

[2]Formulae and explanations regarding GMMs are taken from Bishop (2006), pages 430 through 439. Derivations are given in vastly greater detail and with more eloquence there.

case, K-Means cluster means translate naturally to GMM component means, K-Means cluster variances to GMM variance matrices and the distribution of samples across K-Means clusters to GMM mixing coefficients.

After setting these value, two steps are iterated until convergence: the E step and the M step. Let us consider both in turn.

During the E step, component responsibilities $\gamma(z_k)$ are calculated using the current values of the model parameters.

Consequently during the M step these responsibilities are used to find new values for means, covariances and mixing coefficients, in this order.

Each application of this set of two steps is guaranteed to increase the log-likelihood of the model, i.e.

$$\log p(X|\mu, \Sigma, \pi) \tag{3.9}$$

One can thus iterate application of the E and M step until the increase in log-likelihood falls under some threshold, or a maximum number of iterations is reached.

Formally, we can write this a follows:

1. Initialize the parameters, randomly or for example using the outcomes of a run of K-Means

2. E: Calculate the component responsibilities using the current parameter values
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{3.10}$$

3. M: Use these responsibilities to update current parameter estimates
$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \tag{3.11}$$
$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T \tag{3.12}$$
$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{3.13}$$

   where
$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{3.14}$$

4. Evaluate log-likelihood

$$\log p(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \mu_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\} \tag{3.15}$$

In conclusion of our discussion of GMMs, it is worth noting that using maximum likelihood methods of estimating GMM parameters brings with it some inherent risk. If there are insufficient observations per component it can happen that the optimization process ends up in singularities, finding solutions with infinite likelihood. This can avoided by e.g. using a Bayesian approach to GMMs.

3.4 Outcomes

Running this experiment on PAMAP2 for range of of values of missing data and using the models described above yields the results depicted in Figure 3.1. The X-axis indicates how much data is missing, with the origin being data without missing values. The Y-axis is performance as measured by the Adjusted Rand Score. Different colors/lines indicate different clustering models.

This plot shows that finding subjects through clustering does not work at all in this way. Even when there is no missing data at all cluster indices are random when compared to ground truth, i.e. subject IDs. There is some fluctuation in the performance, but the differences between points on the X-axis are negligible. The same holds true for differences between models. Why is this the case? The reason for this lies in the nature of PAMAP2: as all subjects carry out the same activities, inter-subject distances will be small, while inter-activity distances will be large. We see that taking the nature of the data into account is critical.

This brings us to the topic of data models. In particular it seems that subjects are completely hidden. In the next chapter we will investigate three broad classes of data models and their properties. These differ in the relationship between subjects and activities: are activities unique to individual subjects, or are they shared by multiple subjects? These insights guided our follow-up experiments with PAMAP2, which are described in Section 5.

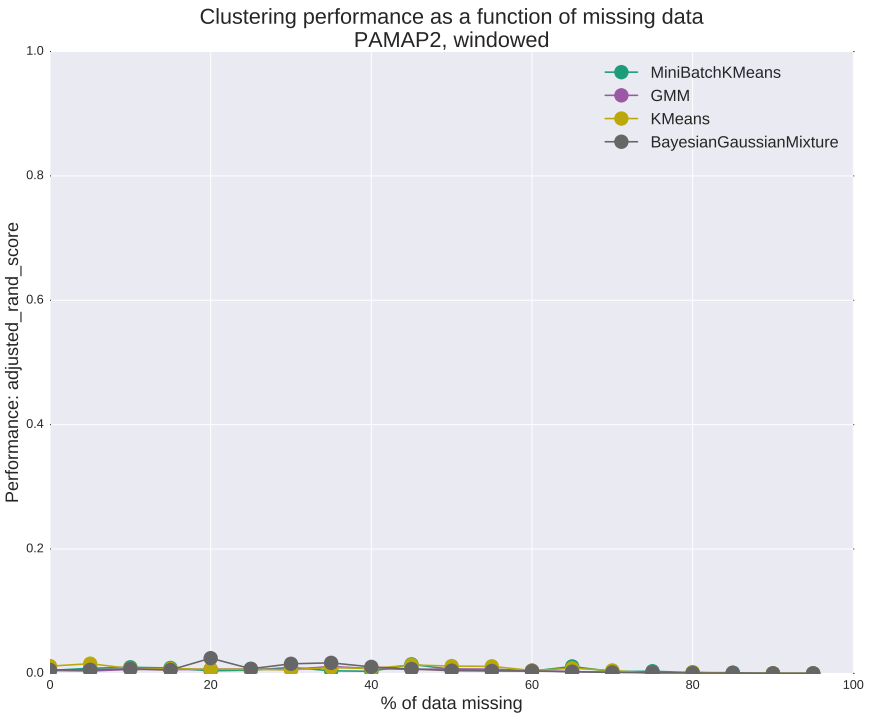

Figure 3.1: Capturing subjects as clusters in windowed PAMAP2 for various amounts of missing data.

Chapter 4

Clustering Performance: Inter-Subject Distance, Inter-Activity Distance And Missing Data

In Section 3 we noticed that the poor performance when trying to recover subjects in PAMAP2 might be explained by our data model not fitting this dataset. Our hypothesis is that there are a few common scenarios regarding the relation of subjects, activities and their clusters: activity-centric, subject-centric and mixed. We have created synthetic datasets according to these scenarios and used them to run experiments exploring the impact of missing data on clustering performance. In what follows we will detail our datasets and findings per scenario.

4.1 Activity-Centric Scenario

4.1.1 Overview

To get a first intuition of this scenario consider PAMAP2. Subjects all, more or less, followed the prescribed activity protocol. This also means that they all carried out only the activities which are part of this protocol. As such, we can view it in such a way as there being one universal pool of activities and all subjects sample from this pool. And while there may be differences between subjects A(lice) and B(ob) running, sensor readings from these two events will still look pretty similar. However, sensor readings between Alice either running or standing will look vastly different. Put differently, inter-activity

differences will be greater than inter-subject differences. Figures 4.1 and 4.2 illustrates this scenario. This is a synthetic dataset we created to investigated the effect of such a data distribution on clustering performance. Let us call this the *activity-centric scenario*, as on the highest level there is one big cluster per activity.

This dataset has the following characteristics:

1. 8000 patterns, evenly split across activities and subjects
2. 4 activity IDs, providing top-level clusters
3. 8 subject IDs, all are present in each top-level cluster and divide each of those into sub-clusters

The data comes from 32 (4 activities times 8 subjects) 2-dimensional Gaussian cluster centers. For each activity ID a common center is chosen. This yields the 4 top-level clusters. For each subject a slight deviation from this center is defined. This yields the 8 sub-clusters per activity. Top-level cluster centers and offsets are chosen manually.

4.1.2 Activity recognition performance

Figure 4.3 illustrates performance when trying to find activities through clustering in the activity-centric synthetic dataset. The following experimental setup was used: Starting from the complete dataset, missing data was induced. Missing values were then estimated as the mean of known values, per feature. The resulting dataset was standardized. This standardized dataset was then used to fit various cluster models, using the true number of activities as the number of clusters if applicable. Predictions of each model were compared to true activity IDs using the Adjusted Rand Score, which is the performance measure shown in the plot. This procedure was iterated for 0% thru 95% of missing data, in 5% increments.

In addition to the models used before, compare Chapter 3.3, we are also also employing Dirichlet Process Gaussian Mixture Models here.

Dirichlet Process Gaussian Mixture Models are an extension of GMMs (Rasmussen 2000). They belong to the class of the Non-parametric Bayesian Models (Orbanz and Teh 2011). Instead of defining the number of components to be identified, DGPMMs estimate these based on the data. The idea is that more complex data leads to a more complex model. The number of components is made part of the model and also inferred during model fitting.

The setup of this model alleviates the modeler of the need of defining the number of clusters in advance. However, it is still necessary to provide a value

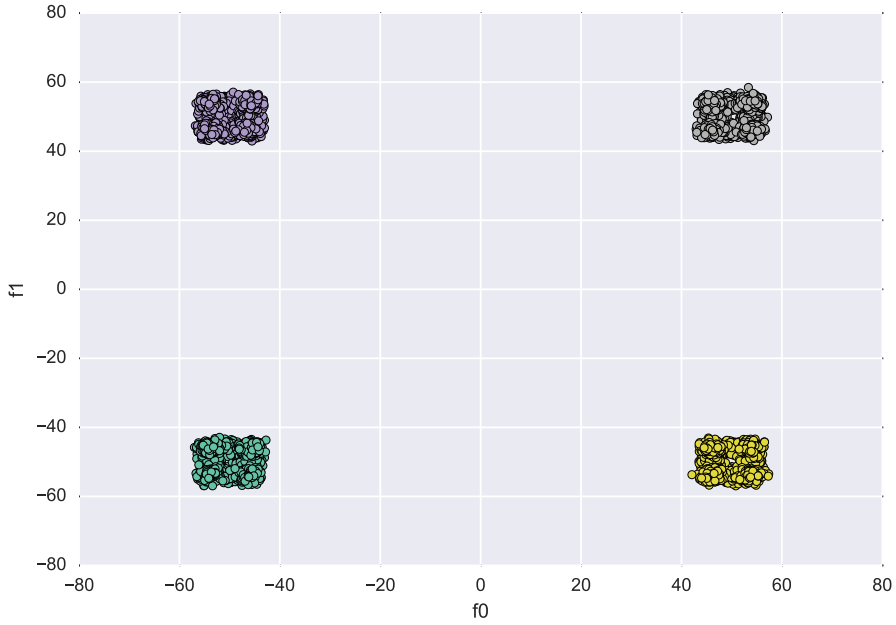

Figure 4.1: An illustration of the activity-centric scenario: Differences between activities are greater than differences between subjects. Colors indicate activities.

Figure 4.2: An illustration of the activity-centric scenario: Differences between activities are greater than differences between subjects. Colors indicate subjects.

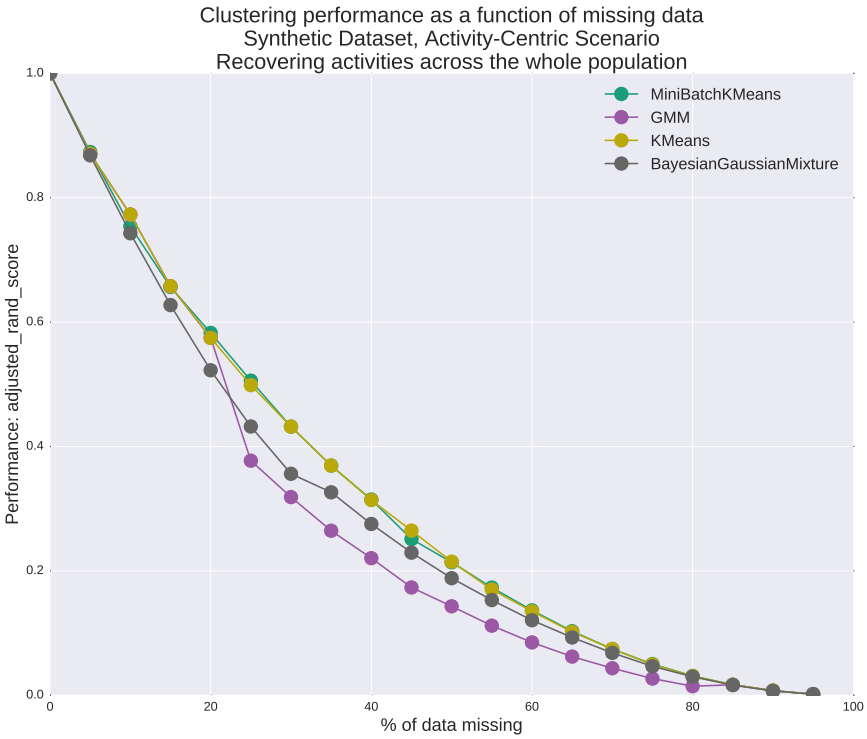

Figure 4.3: An illustration of clustering performance in the activity-centric scenario. X-axis: percentage of missing data, y-axis: adjusted rand index.

for the concentration parameter α, which influences the expected number of clusters. As such, expectations or prior knowledge regarding the granularity of the clustering can – or rather: must – be expressed.

All models started out with perfect clustering for no missing data, which then declined in a sort of exponential decay as more data goes missing. KMeans and MiniBatchKMeans performed the best across the whole range of missing values. They closely tracked each other's performance to the point of being indistinguishable. GMM started out the same, but suffered more performance penalties as more data went missing. DPGMM (indicated as Bayesian Gaussian Mixture in the legend) lies in-between these two groups, mostly faring better than GMM but on average slightly worse than KMeans and MiniBatchKMeans.

The curves were in line with expectations, as activity clusters were chosen in such a way as to be easily separable.

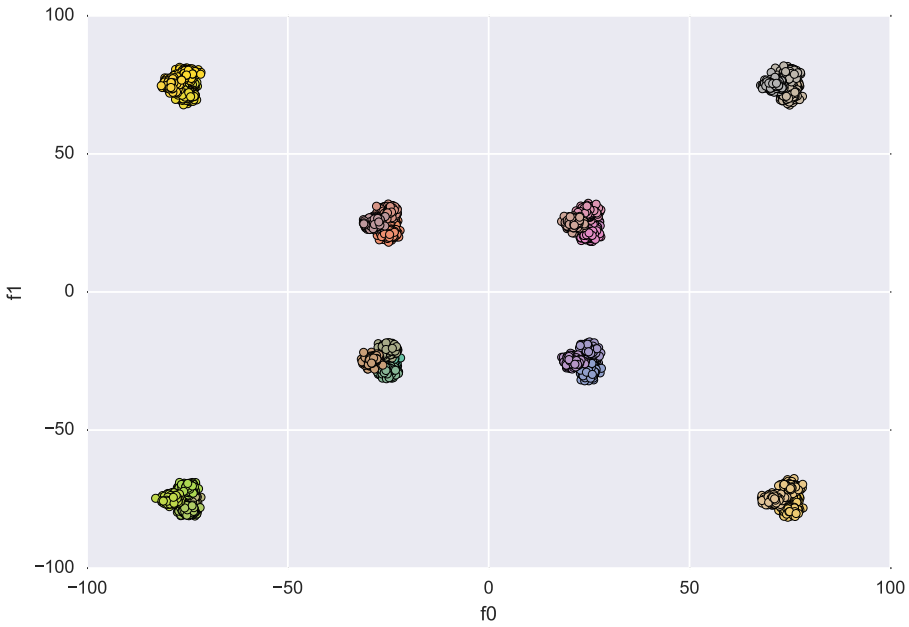

Figure 4.4: An illustration of the subject-centric scenario: Differences between subjects are greater than differences between activities. Colors indicate activities. Though somewhat hard to see, there are three activities per top-level cluster.

4.2 Subject-Centric Scenario

4.2.1 Overview

In a certain sense the subject-centric scenario is the inverse of the activity-centric one. In this case we assume that inter-subject are big, while inter-activity differences comparatively are small. This could, for example, reflect a situation in which subjects tend to have unique activities not shared by any other subjects. More concretely, Alice could enjoy riding her motorcycle and snowboarding, whereas Bob is more into crocheting. This is illustrated in Figures 4.4 and 4.5, a synthetic dataset created to illustrate this kind of distribution. Let us call this the *subject-centric scenario*, owed to the fact that subjects form the top-level clusters.

This has dataset the following characteristics:

1. 8000 patterns, spread evenly across activities and subjects
2. 8 subjects, providing top-level clusters

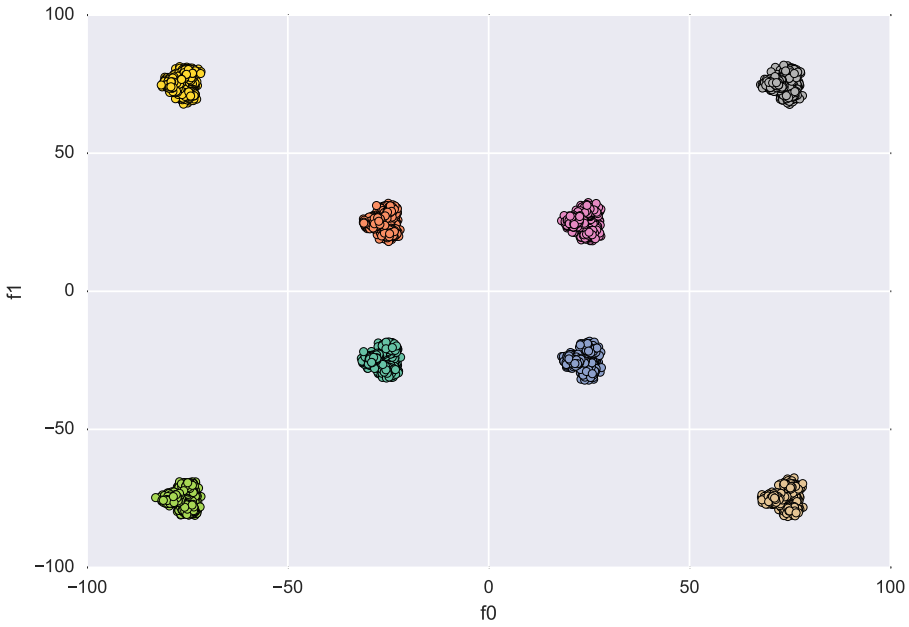

Figure 4.5: An illustration of the subject-centric scenario: Differences between subjects are greater than differences between activities. Colors indicate subjects.

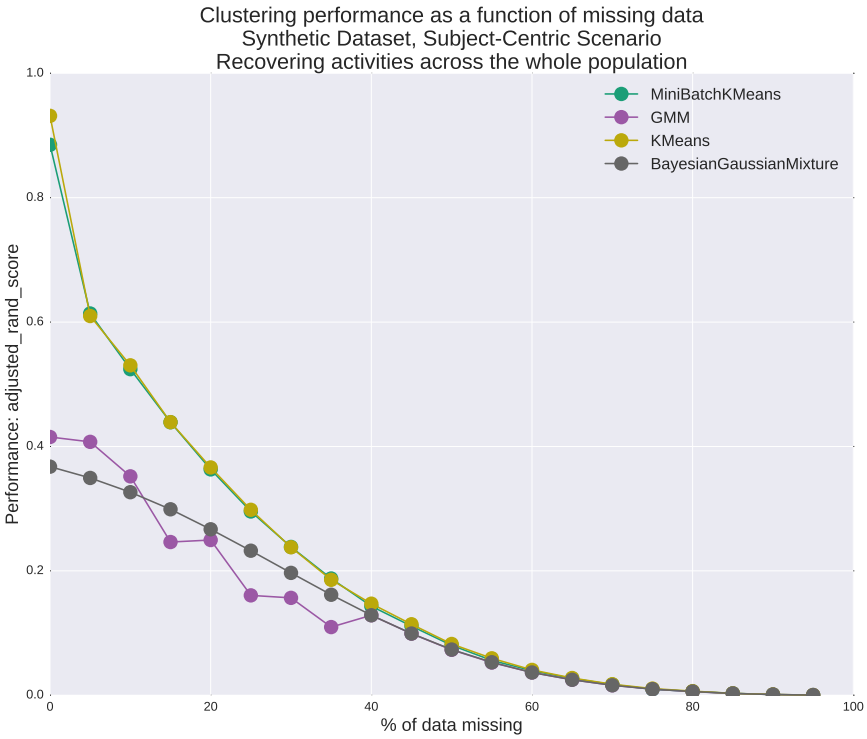

Figure 4.6: An illustration of clustering performance in the subject-centric scenario. X-axis: percentage of missing data, y-axis: adjusted rand index.

3. 32 activities, providing sub-clusters

32 clusters were created by setting 8 cluster centers and shifting them by 4 offsets each. As such we end up with 8 super-clusters, each representing one subject, each of which can be split into 4 sub-clusters, each of which represents an activity this subject carries out.

4.2.2 Activity recognition performance

Figure 4.6 illustrates performance when trying to find activities through clustering in the activity-centric synthetic dataset. The experimental setup was the same as for the activity-centric case, see Section 4.1.2

Again we found (MiniBatch)KMeans outperforming the other approaches. Performance in general was lower than for the activity-centric scenario as one would expect. In fact worse performance could have been expected given that top-level structures were grouped by subjects. This however, might be due to the fact that this was a very regular dataset, easy to structure once the right

level of detail is found. Furthermore, true clusters were globular, which fits the (MB)KM cluster model. This also contributed to these models consistently taking the lead, which might change for different cluster shapes. GMMs showed worse performance than (MB)KM models. They also performed worse than in the activity-centric scenario. It is noteworthy that GMMs should have intuitively fit just as well as KMeans. Their underlying model is the factual method of data generation. Furthermore, the covariance type, which determines the shapes of the Gaussians modeled, was set to "diagonal", yielding circular clusters. DPGMMs (Bayesian Gaussian Mixtures), again, yielded performance somewhere in-between GMMs and KMeans. They start out slightly weaker than GMMs, catch up around 10% missing data and from there to 40% tend to outperform GMMs slightly.

4.3 Mixed Scenario

4.3.1 Overview

Finally, there is a also the *mixed scenario*, which is a combination of both the aforementioned cases. Some activities are shared by many subjects, this constitutes the activity-centric portion of this data. However, there are also many activities which are unique to single subjects. This scenario is closest to typical free-living data. There will be many shared activities, such as walking or eating, but most people will also have some unique/rare ones such as, e.g., base jumping or unicycling. We have created a synthetic dataset to illustrate this scenario, as depicted in Figures 4.7 and 4.8.

This dataset has the following characteristics:

1. 16000 patterns, spread evenly across activities and subjects
2. 8 subjects, providing top-level clusters and sub-clusters
3. 36 activities, providing top-level clusters and sub-clusters

This dataset is the union of the activity-centric and the subject-centric synthetic datasets. Please refer to their descriptions in Section 4.1 and Section 4.2 to see how the base for this dataset was created.

Activity IDs for both base sets were shifted to not collide. This resulted in $4 + 32 = 36$ activity IDs. Each of the 8 subjects takes part in the 4 shared activities and also has 4 unique activities not shared with any other subject. Looking from the outside there are $4 + 8 = 12$ top-level clusters. 4 of these represent activities and decompose into subjects, vice versa for the remaining 8. None of the top-level clusters overlap.

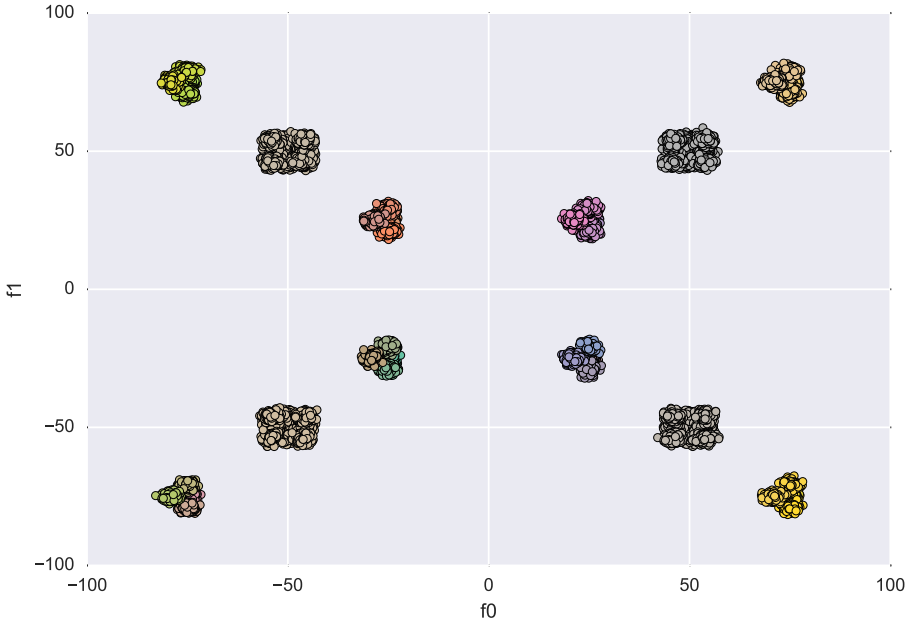

Figure 4.7: An illustration of the mixed scenario: Some activities are unique to individual subjects (high inter-activity distance), other activities are shared between subjects (high inter-subject distance). Colors indicate activities.

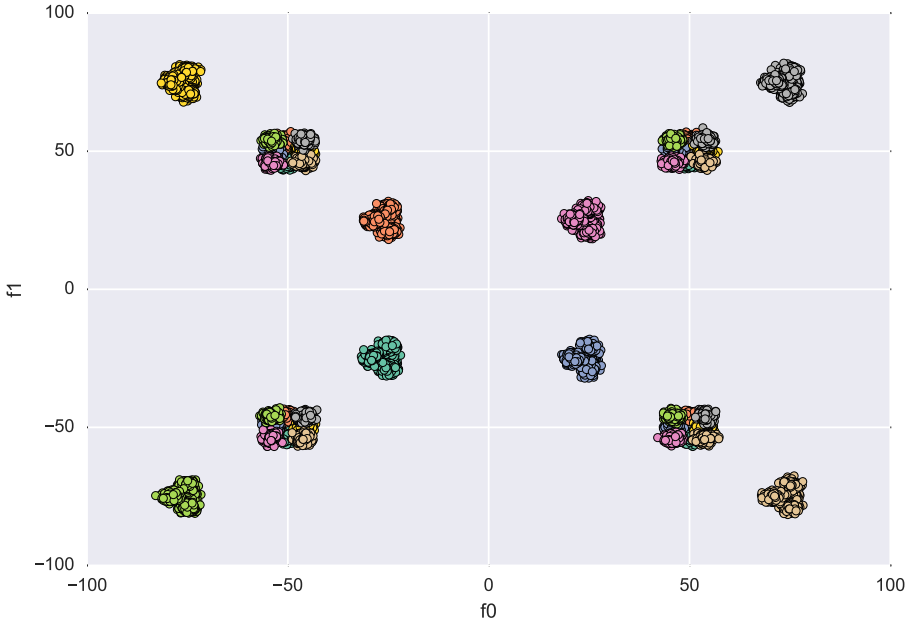

Figure 4.8: An illustration of the mixed scenario: Some activities are unique to individual subjects (high inter-activity distance), other activities are shared between subjects (high inter-subject distance). Colors indicate subjects.

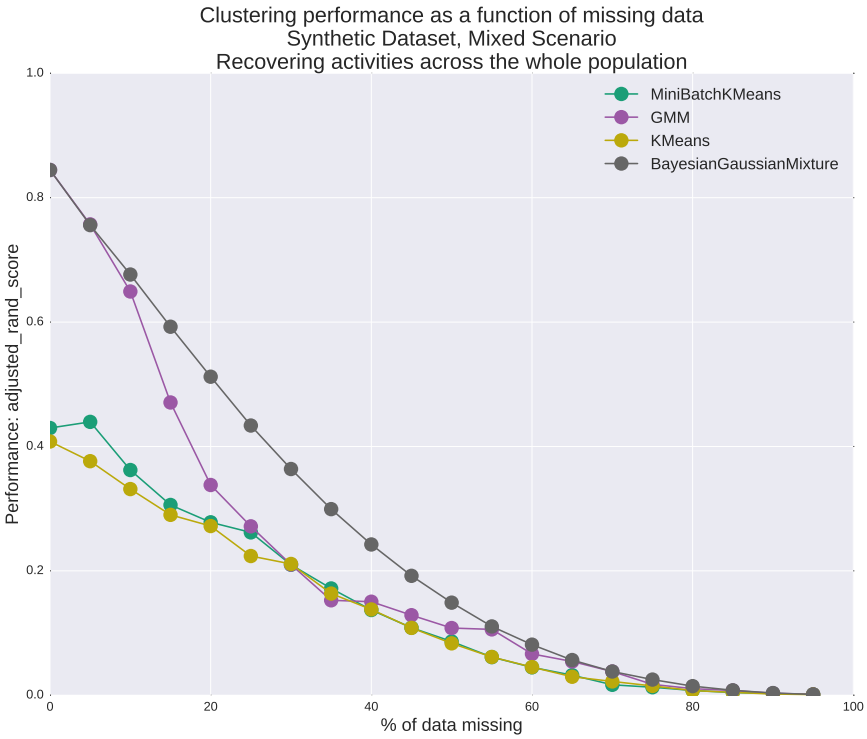

Figure 4.9: An illustration of clustering performance in the mixed scenario. X-axis: percentage of missing data, y-axis: adjusted rand index.

4.3.2 Activity recognition performance

Figure 4.9 illustrates performance when trying to find activities through clustering in the mixed synthetic dataset. The experimental setup was the same as for the activity-centric case, see Section 4.1.2

In the mixed scenario DPGMM (indicated as Bayesian Gaussian Mixture) surprisingly outperformed all other clusterers. For the range of 0% to 10% of missing data GMM shows mostly the same performance. However, between 15% and 50% of missing data it clearly outperforms both GMM and KMeans-based models. (MB)KMeans severely dropped off in performance compared to both other scenarios. It started at around 0.5 for the Adjusted Rand Score, compared to 0.9 in the subject-centric scenario or 1.0 in the activity-centric one. The gap between GMM and KMeans performance gradually closed as more data went missing, with the point of convergence being around 25% till 30%. From this moment on, those models showed similar performance, with GMM performing slightly better at around 50%.

PAMAP2 seems to be a prototypical example of the activity-centric

scenario. The experiments in Section 5 were influenced by this hypothesis.

Chapter 5

Finding Activities Through Clustering in PAMAP2

With the consideration of different data models in mind, another experiment presented itself: Try to find activities in the whole dataset by clustering. This experiment closely followed the procedure outlined above, except for two points:

1. The number of clusters given to models as a parameter is now the number of unique activities

2. Performance is evaluated as the agreement between cluster indices and activity index.

Imputation of missing values again was done by using the mean of known values for a given feature. Note that an activity, e.g. running, has the same ID across all subjects.

Figure 5.1 shows results of this experiment. Overall performance was much higher with Adjusted Rand Scores starting out at about 0.4-0.5 for no missing data. As more data went missing performance gradually decreased, which is in alignment with expectations. GMM demonstrated slightly better performance than the K-Means-based models in the 20%-35% range. MBKM mostly followed KM, but sometimes dropped below that line. For the single dip at 65% this might be explained by being more susceptible to problems of local minima, though on an earlier stretch it performed consistently worse than KM. DPGMM (indicated as Bayesian Gaussian Mixture) performance was mostly in line with other models. In the 35% to 65% missing data range it performed worse the others.

Based on this result some tweaking was carried out to establish a higher baseline of performance. The task changed in the following way:

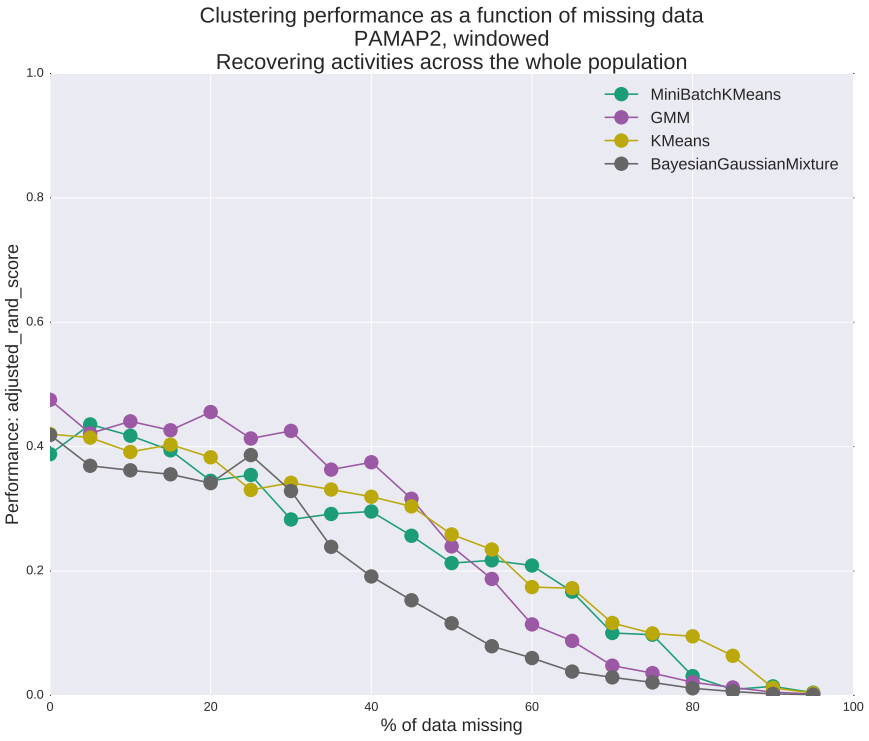

Figure 5.1: Clustering performance as a function of missing data when trying to find activities as clusters.

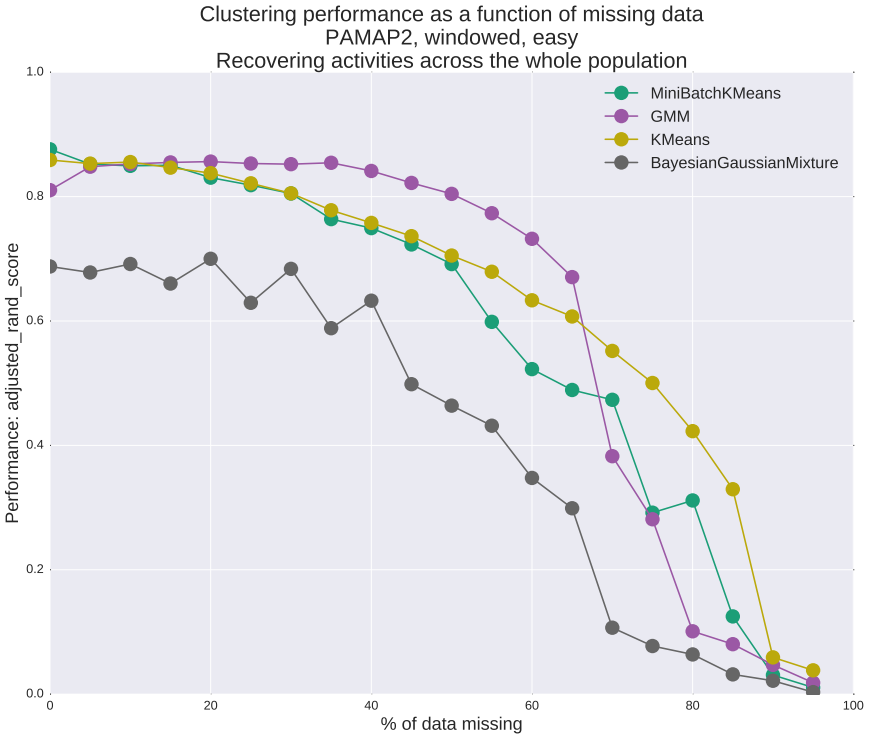

Figure 5.2: Clustering performance as a function of missing data when trying to find subjects as clusters. Here, PAMAP2 has been restricted to a subset of easily-distinguishable activities.

1. All samples with activity ID being 0, indicating "transient activity" were removed. This yielded nearly no improvement

2. Subjects 103 and 109 were removed. These showed atypical behavior. 109, for example, mainly drove her car around, ignoring other activities. No clear change in performance.

3. Limiting the data to samples (windows) belonging to a set of easily-distinguished activities, viz. lying, sitting, walking and running. This greatly increased performance.

Carrying out the experiment yielded significantly improved performance, as depicted in Figure 5.2. Our hypothesis is that this improvement mainly comes from less overlap between the clusters. This is based on the insight of Wilson (2015), who determined this to be the primary factor in clustering performance. Please compare Chapter 2.4 for an extended discussion.

KM, MBKM and GMM now started at an Adjusted Rand Score of around 0.8 to 0.9, meaning close to perfect agreement between cluster indices and activity IDs. Trends established in the previous experiment came out more clearly: GMM outperformed the other models in the 25%-65% range. However, at that point it significantly dropped off, being outperformed by K-Means. MBKM first tracked KM, then sharply dropped, probably due to a local optimum. After 55% it consistently scored lower than regular KM. DPGMM (indicated by Bayesian Gaussian Mixture) yielded consistently lower performance than any of the other models.

It is difficult to hypothesize what might be the root cause for these results. Nevertheless, one can use these data to give some recommendations for selecting a model when facing controlled, real-world data with missing values, in which activities are shared among subjects.

If no data at all is missing K-Means and MiniBatch K-Means offer the best performance. For moderate amounts of missing values between 0% and 60% GMMs demonstrate superior clustering results. If more than 60% of data is missing all model performance deteriorates sharply. However, K-Means holds up the best. DPGMMs are generally not advisable in this setting.

This concludes the experimental section of this thesis. In what follows, we conclude and review the work presented here and discuss pointers for possible promising future work.

Chapter 6

Discussion and Future Work

This work is a first foray into unsupervised Human Activity Recognition (HAR) in wearable sensor data. It thus fills a critical gap in earlier research, paving the way for analyzing a new class of rich datasets and applications utilizing these insights.

To achieve this, we have reviewed the existing HAR literature. Based on the formal definition of the HAR problem we have derived UHAR, the Unsupervised Human Activity Recognition problem, yielding theoretical foundations and common terminology for future research.

We have furthermore connected this field of inquiry with prior research regarding the interaction of missing values and clustering. Due to the nature of unsupervised learning and missing values research on the intersection of these areas is quite sparse. Nevertheless results in this area helped explain some of our empirical findings and provide guidelines for approaching missing value handling when trying to discover clusters of activities without ground truth labels.

We furthermore identified possible scenarios regarding the relationship between subjects and activities, i.e. whether activities are shared between subjects, unique to subjects or a mix of those two. We introduced these as the activity-centric, subject-centric and mixed scenarios, respectively.

These central assumptions also lead a different natural hierarchy of clusters in the data: In the activity-centric scenario, subjects look alike, i.e. there is low inter-subject distance in the data. On the flipside, activities are quite different from one another, i.e. there is high inter-activity distance. This leads to activities being more easily identified as clusters than subjects. The complement of this holds true for the subject-centric case, with high inter-subject distances and low inter-activity distances.

Each of these scenarios was explored by creating matching synthetic dataset. Each such dataset was then used to run experiments revealing

clustering performance of a variety of models, viz. K-Means, MiniBatch K-Means, Gaussian Mixture Models and Infinite/Dirichlet Process Gaussian Mixture Models. Clustering performance is determined as a function of how much data was missing. These results shed light on how well the tested models deal with missing values, in the context of the different activity/subject scenarios. They might thus provide practical insight for choosing narrowing down model choices when one can identify which scenario most closely matches the dataset at hand and how many values are missing.

Furthermore, experiments regarding both activity and subject identification were conducted on the PAMAP2 dataset. This is a real-world wearable sensor dataset, in which subjects follow a pre-defined protocol of activities. This setup makes it less complex and chaotic than free living data, but still provides sensor readings as they might be encountered in the wild. Furthermore, it is labeled with high accuracy. These properties make it into an excellent test case for clustering methods.

Our experiments on this dataset reveal how the degree of missing data impacts clustering performance for a range of models. They provide an estimate of real-world viability of these methods and provide guidelines concerning the choice of model given the amount of missing values. They also reconfirm findings of earlier research regarding missing values and unsupervised learning in which cluster overlap was identified as the key factor in being able to correctly identify said clusters.

As an overall conclusion and practical recommendation we would like to make the following points about the models under investigation and their usefulness. DPGMMs demonstrated inferior performance for non-synthetic data. It remains to be determined if this can be alleviated with more optimal hyperparameter choices, but as a first recommendation we would advise against employing them for unsupervised human activity recognition.

K-Means as a model offers good overall performance. It generally trumps other models when no data is missing. For intermediate to high, say 5% to 60%, amounts of missing data it shows fair performance. Beyond the 60% mark it shows the least step decrease in performance across all models. In spite of its simplicity it is still a highly useful model across the board.

MiniBatch K-Means can serve as a drop-in replacement for K-Means. It is far superior in terms of computational complexity, making it feasible for much larger datasets. However, we have seen empirical evidence of its clustering performance never being better than that of K-Means but being far less regular. This means that there is a higher risk of receiving an unusually bad clustering due to chance. If this model is to be used we recommend taking the best out of a considerable number of runs. The random jumps in performance were still visible for at least 10 iterations.

Gaussian Mixture Models evaluated the best for intermediate to high amounts of missing data, outperforming K-Means in this region. Without missing data they still came close. For very high amounts of missing data, i.e. more than 60%, they suffered more severe performance losses than K-Means. Based on this we can generally recommend them as models in unsupervised human activity recognition if the amount of missing data does not exceed 60%.

The investigation presented in this thesis leaves quite a few questions open which we wished to have also studied and which are promising candidates for future research.

First and foremost would be the comparison of the impact of various missing data strategies on the clustering performance. We became aware of the existing research in this area after finalizing all experimental work. We are convinced that trying different strategies will have an enormous impact on the clustering results. Essential strategies to test would include:

1. Forward-filling of the last known value[1]
2. Regression based on known values
3. Multiple imputation
4. Modeling the missingness process directly

Furthermore, it appears promising to us to investigate models which might be able to inherently handle missing data well, such as Unsupervised Extreme Learning Machines (Huang et al. 2014).

One may also consider different ways of handling missing values. Abdella and Marwala (2005) demonstrate a promising approach for imputation using genetic algorithms and neural networks. Another interesting option is the use of (deep) autoencoders (Vincent et al. 2008), which can be trained on complete patterns and then used to impute missing values for incomplete data points.

Finally, these autoencoders can also be used as a clustering model themselves, as described in the work of Le (2013).

Based on the observation that overlap between clusters is an essential factor for clustering performance it would also make sense to investigate better feature engineering. These could be part of the pre-processing, e.g. doing Fourier Transformations on sensor readings to decompose it into more elemental components. Alternatively, kernel-based models could be employed

[1]This has already been employed to downsample heart rate data, but not to complete missing observations

which could potentially separate clusters more easily by projection into a higher-dimensional space.

Finally, our empirical results also warrant a deeper dive. Identifying their root causes might lead to both theoretical and practical insight. For example we observed that GMMs had non-optimal performance on synthetic data sets in spite of being the true model of the underlying data-generating process. This might hint at interesting challenges.

We hope to have provided some interesting foundations, observations and practical guidelines for getting started in unsupervised human activity recognition.

Bibliography

Abdella, Mussa and Tshilidzi Marwala (2005). "The use of genetic algorithms and neural networks to approximate missing data in database". In: *IEEE 3rd International Conference on Computational Cybernetics, 2005. ICCC 2005.* Pp. 207–212. DOI: 10.1109/ICCCYB.2005.1511574.

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 1027–1035. URL: http://dl.acm.org/citation.cfm?id=128338 (visited on 01/27/2017).

Bao, Ling and Stephen S. Intille (2004). "Activity recognition from user-annotated acceleration data". In: *Pervasive computing*. Springer, pp. 1–17. URL: http://link.springer.com/chapter/10.1007/978-3-540-24646-6_1 (visited on 02/17/2016).

Baraldi, Amanda N. and Craig K. Enders (2010). "An introduction to modern missing data analyses". In: *Journal of School Psychology* 48.1, pp. 5–37. URL: http://www.sciencedirect.com/science/article/pii/S0022440509000661 (visited on 01/08/2017).

Berchtold, Martin et al. (2010). "Actiserv: Activity recognition service for mobile phones". In: *Wearable Computers (ISWC), 2010 International Symposium on*. IEEE, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5665868 (visited on 02/17/2016).

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN: 978-0-387-31073-2.

Buuren, Stef van and Karin Groothuis-Oudshoorn (2011). "mice: Multivariate Imputation by Chained Equations in R". In: *Journal of Statistical Software* 45.3, pp. 1–67. URL: http://www.jstatsoft.org/v45/i03/.

Carmichael, J. W. and R. S. Julius (1968). "Finding natural clusters". In: *Systematic Biology* 17.2, pp. 144–150. URL: http://sysbio.oxfordjournals.org/content/17/2/144.short (visited on 08/05/2016).

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38. URL: http://www.jstor.org/stable/2984875 (visited on 01/29/2017).

Dixon, J. K. (1979). "Pattern Recognition with Partly Missing Data". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.10, pp. 617–621. ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310090.

Ermes, Miikka, Juha Pärkkä, and Luc Cluitmans (2008). "Advancing from offline to online activity recognition with wearable sensors". In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, pp. 4451–4454. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4650199 (visited on 02/17/2016).

Fortier, J. J. and Ho Solomon (1966). "Clustering procedures". In: *Multivariate Analysis* 62.

Fraley, Chris and Adrian E. Raftery (2002). "Model-based Clustering, Discriminant Analysis and Density Estimation". In: *Journal of the American Statistical Association* 97, pp. 611–631.

Fraley, Chris, Adrian E. Raftery, et al. (2012). *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*. 597. Department of Statistics, University of Washington.

Hall, D. J. et al. (1973). *Development of New Pattern-Recognition Methods*. Tech. rep. DTIC Document. URL: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0772614 (visited on 08/05/2016).

Hanai, Yuya, Jun Nishimura, and Tadahiro Kuroda (2009). "Haar-like filtering for human activity recognition using 3d accelerometer". In: *Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009. IEEE 13th*. IEEE, pp. 675–678. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4786008 (visited on 02/17/2016).

Hancke, Gerhard, Bruno Silva, and Gerhard Hancke Jr. (2012). "The Role of Advanced Sensing in Smart Cities". en. In: *Sensors* 13.1, pp. 393–425. ISSN: 1424-8220. DOI: 10.3390/s130100393. URL: http://www.mdpi.com/1424-8220/13/1/393/ (visited on 02/17/2016).

He, Zhen-Yu and Lian-Wen Jin (2008). "Activity recognition from acceleration data using AR model representation and SVM". In: *Machine Learning and Cybernetics, 2008 International Conference on*. Vol. 4. IEEE, pp. 2245–2250. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4620779 (visited on 02/17/2016).

Hong, Jihoon and Tomoaki Ohtsuki (2011). "A state classification method based on space-time signal processing using SVM for wireless monitoring systems". In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*. IEEE, pp. 2229–2233. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6139913 (visited on 02/17/2016).

Huang, Gao et al. (2014). "Semi-supervised and unsupervised extreme learning machines". In: *Cybernetics, IEEE Transactions on* 44.12, pp. 2405–2417. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6766243 (visited on 03/16/2016).

Hubert, Lawrence and Phipps Arabie (1985). "Comparing partitions". en. In: *Journal of Classification* 2.1, pp. 193–218. ISSN: 0176-4268, 1432-1343. DOI: 10.1007/BF01908075. URL: http://link.springer.com/article/10.1007/BF01908075 (visited on 01/11/2017).

Jain, Anil K. and Richard C. Dubes (1988). *Algorithms for Clustering Data*. en. Prentice Hall PTR. ISBN: 978-0-13-022278-7.

Jensen, Robert E. (1969). "A dynamic programming algorithm for cluster analysis". In: *Operations Research* 17.6, pp. 1034–1057. URL: http://pubsonline.informs.org/doi/abs/10.1287/opre.17.6.1034 (visited on 08/05/2016).

Kao, Tzu-Ping, Che-Wei Lin, and Jeen-Shing Wang (2009). "Development of a portable activity detector for daily activity recognition". In: *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*. IEEE, pp. 115–120. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5222001 (visited on 02/17/2016).

Kittler, Josef (1976). "A locally sensitive method for cluster analysis". In: *Pattern Recognition* 8.1, pp. 23–33. URL: http://www.sciencedirect.com/science/article/pii/0031320376900261 (visited on 08/05/2016).

Lara, Óscar D. and Miguel A. Labrador (2013). "A survey on human activity recognition using wearable sensors". In: *Communications Surveys & Tutorials, IEEE* 15.3, pp. 1192–1209. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6365160 (visited on 02/04/2016).

Lara, Óscar D., Alfredo J. Pérez, et al. (2012). "Centinela: A human activity recognition system based on acceleration and vital sign data". In: *Pervasive and mobile computing* 8.5, pp. 717–729. URL: http://www.sciencedirect.com/science/article/pii/S1574119211000794 (visited on 02/17/2016).

Le, Q. V. (2013). "Building high-level features using large scale unsupervised learning". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8595–8598. DOI: 10.1109/ICASSP.2013.6639343.

Lloyd, S. (1982). "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489.

Maurer, Uwe et al. (2006a). "Activity recognition and monitoring using multiple sensors on different body positions". In: *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*. IEEE, 4–pp. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1612909 (visited on 02/17/2016).

— (2006b). "Activity recognition and monitoring using multiple sensors on different body positions". In: *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*. IEEE, 4–pp. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1612909 (visited on 02/17/2016).

McLachlan, Geoffrey and Thriyambakam Krishnan (2007). *The EM algorithm and extensions*. Vol. 382. John Wiley & Sons. URL: https://books.google.com/books?hl=en&lr=&id=NBawzaWoWa8C&oi=fnd&pg=PR3&dq=mclachlan+em+algorithm&ots=tne7SQ_AvN&sig=OC9EwFEf6TAcDduSlCR4_pP1DZw (visited on 01/29/2017).

Melnykov, Volodymyr, Wei-Chen Chen, and Ranjan Maitra (2012). "MixSim: An R Package for Simulating Data to Study Performance of Clustering Algorithms". In: *Journal of Statistical Software* 51.12, pp. 1–25. URL: http://www.jstatsoft.org/v51/i12/.

Oh, Keunhyun, Han-Saem Park, and Sung-Bae Cho (2010). "A mobile context sharing system using activity and emotion recognition with Bayesian networks". In: *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on*. IEEE, pp. 244–249. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5667193 (visited on 02/17/2016).

Omatu, Sigeru et al. (2009). *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living: 10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops, Salamanca, Spain, June 10-12, 2009. Proceedings*. Vol. 5518. Springer. URL: https://books.google.nl/books?hl=en&lr=&id=I8dqCQAAQBAJ&oi=fnd&pg=PR3&dq=+T.+Brezmes,+J.-L.+Gorricho+and+J.+Cotrina+,++Distributed+Computing,+Artificial+Intelligence,+Bioinformatics,+Soft+Computing,+and+Ambient+Assisted+Living+,+vol.+5518+,+pp.796+-799+,+2009+:Springer+Berlin+&ots=cY9Rm9V9a9&sig=mQLOlVgRtXOLpMzRIEggapKiDak (visited on 02/17/2016).

Orbanz, Peter and Yee Whye Teh (2011). "Bayesian nonparametric models". In: *Encyclopedia of Machine Learning*. Springer, pp. 81–89. URL: http:

//link.springer.com/10.1007/978-0-387-30164-8_66 (visited on 08/28/2016).

Pärkkä, Juha et al. (2006). "Activity classification using realistic data from wearable sensors". In: *Information Technology in Biomedicine, IEEE Transactions on* 10.1, pp. 119–128. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1573714 (visited on 02/17/2016).

Puccinelli, Daniele and Martin Haenggi (2005). "Wireless sensor networks: applications and challenges of ubiquitous sensing". In: *IEEE Circuits and Systems Magazine* 5.3, pp. 19–31. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1507522 (visited on 08/31/2016).

Rand, William M. (1971). "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical Association* 66.336, pp. 846–850. ISSN: 0162-1459. DOI: 10.2307/2284239. URL: http://www.jstor.org/stable/2284239 (visited on 01/16/2017).

Rasmussen, Carl Edward (2000). "The Infinite Gaussian Mixture Model". In: *Advances in Neural Information Processing Systems 12*. Ed. by S. A. Solla, T. K. Leen, and K. Müller. MIT Press, pp. 554–560. URL: http://papers.nips.cc/paper/1745-the-infinite-gaussian-mixture-model.pdf.

Reddy, Sasank et al. (2010). "Using mobile phones to determine transportation modes". In: *ACM Transactions on Sensor Networks (TOSN)* 6.2, p. 13. URL: http://dl.acm.org/citation.cfm?id=1689243 (visited on 02/17/2016).

Reiss, A. and D. Stricker (2012). "Introducing a New Benchmarked Dataset for Activity Monitoring". In: *2012 16th International Symposium on Wearable Computers*, pp. 108–109. DOI: 10.1109/ISWC.2012.13.

Riboni, Daniele and Claudio Bettini (2011). "COSAR: hybrid reasoning for context-aware activity recognition". In: *Personal and Ubiquitous Computing* 15.3, pp. 271–289. URL: http://link.springer.com/article/10.1007/s00779-010-0331-7 (visited on 02/17/2016).

Rubin Donald, B. (1987). *Multiple imputation for nonresponse in surveys*. New York: Wiley.

Sarkar, Jehad, Young-Koo Lee, Sungyoung Lee, et al. (2011). "GPARS: A general-purpose activity recognition system". In: *Applied Intelligence* 35.2, pp. 242–259. URL: http://link.springer.com/article/10.1007/s10489-010-0217-4 (visited on 02/17/2016).

Schafer, Joseph L. and John W. Graham (2002). "Missing data: our view of the state of the art." In: *Psychological Methods* 7.2, p. 147. URL: http://psycnet.apa.org/journals/met/7/2/147/ (visited on 01/08/2017).

Sculley, David (2010). "Web-scale k-means clustering". In: *Proceedings of the 19th international conference on World wide web*. ACM, pp. 1177–

1178. URL: http://dl.acm.org/citation.cfm?id=1772862 (visited on 01/11/2017).

Sneath, Peter HA, Robert R. Sokal, et al. (1973). *Numerical taxonomy. The principles and practice of numerical classification.* W H Freeman & Co (Sd). ISBN: 978-0-7167-0697-7.

Tapia, Emmanuel Munguia et al. (2007). "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor". In: *Wearable Computers, 2007 11th IEEE International Symposium on*. IEEE, pp. 37–40. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4373774 (visited on 02/17/2016).

Tolstikov, Andrei et al. (2011). "Comparison of fusion methods based on dst and dbn in human activity recognition". In: *Journal of Control Theory and Applications* 9.1, pp. 18–27. URL: http://link.springer.com/article/10.1007/s11768-011-0260-7 (visited on 02/17/2016).

Tversky, Amos (1977). "Features of similarity." In: *Psychological review* 84.4, p. 327. URL: http://psycnet.apa.org/journals/rev/84/4/327/ (visited on 08/05/2016).

Urquhart, Roderick (1982). "Graph theoretical clustering based on limited neighbourhood sets". In: *Pattern recognition* 15.3, pp. 173–187. URL: http://www.sciencedirect.com/science/article/pii/0031320382900693 (visited on 08/05/2016).

Van Kasteren, T. L. M., Gwenn Englebienne, and Ben JA Kröse (2010). "An activity monitoring system for elderly care using generative and discriminative models". In: *Personal and ubiquitous computing* 14.6, pp. 489–498. URL: http://link.springer.com/article/10.1007/s00779-009-0277-9 (visited on 02/17/2016).

Vincent, Pascal et al. (2008). "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: ACM, pp. 1096–1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294. URL: http://doi.acm.org/10.1145/1390156.1390294 (visited on 01/25/2017).

Wilson, Stefan E. (2015). "Methods for Clustering Data with Missing Values". In: URL: https://www.math.leidenuniv.nl/scripties/MasterWilson.pdf (visited on 11/02/2016).

Xiong, Naixue et al. (2013). "Energy-Efficient Algorithm for Broadcasting in Ad Hoc Wireless Sensor Networks". en. In: *Sensors* 13.4, pp. 4922–4946. ISSN: 1424-8220. DOI: 10.3390/s130404922. URL: http://www.mdpi.com/1424-8220/13/4/4922/ (visited on 02/17/2016).

Yang, Jaeyoung, Joonwhan Lee, and Joongmin Choi (2011). "Activity recognition based on RFID object usage for smart mobile devices". In: *Jour-*

nal of Computer Science and Technology 26.2, pp. 239–246. URL: http://link.springer.com/article/10.1007/s11390-011-9430-9 (visited on 02/17/2016).

Yoo, Seong-eun (2013). "A Wireless Sensor Network-Based Portable Vehicle Detector Evaluation System". en. In: *Sensors* 13.1, pp. 1160–1182. ISSN: 1424-8220. DOI: 10.3390/s130101160. URL: http://www.mdpi.com/1424-8220/13/1/1160/ (visited on 02/17/2016).

Yücel, Zeynep et al. (2013). "Deciphering the Crowd: Modeling and Identification of Pedestrian Group Motion". en. In: *Sensors* 13.1, pp. 875–897. ISSN: 1424-8220. DOI: 10.3390/s130100875. URL: http://www.mdpi.com/1424-8220/13/1/875/ (visited on 02/17/2016).

Zagoruiko, N. G. and V. N. Yolkina (1982). "22 Inference and data tables with missing values". In: *Handbook of Statistics* 2, pp. 493–500. URL: http://www.sciencedirect.com/science/article/pii/S0169716182020252 (visited on 08/05/2016).