Methods and Tools for Mining Multivariate Time Series

by

Ricardo Emanuel de Gouveia da Costa **Cachucho**

Methods and Tools for Mining Multivariate Time Series

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op maandag 10 december 2018
klokke 15.00 uur

door

Ricardo Emanuel de Gouveia da Costa **Cachucho**

geboren te Madeira, Portugal
in 1986

Front and back cover patterns by Ana Cachucho.
Printing: Ridderprint BV, the Netherlands

Para a Cecilia,
que me faz acreditar todos os dias.

Contents

Chapter 1

Introduction

We live an era of unprecedented data challenges. These challenges have been fuelled by technological developments in storage, communication and collection of data. A good example of such developments can be recognized in sensor networks. Most people are not aware of such sources of data but they are around us, in our houses, means of transportation and even with ourselves, fitted into smartphones, watches and other wearables. These sensors have been inserted ubiquitously into our daily lives, producing large amounts of data with the potential for extensive analysis. If seen as a whole, most sensor systems produce multiple variables, derived from one or more sensors. This source of data, that we refer to as *multivariate time series*, forms the main focus of this thesis.

Mining multivariate time series is both the title and the core of this thesis. As a data mining thesis, there is a focus on the algorithmic solutions for the challenges related to multivariate time series. We envision these tasks in three different dimensions: developing methods that solve a particular mining task, providing tools that embody data mining methodologies and developing applications were temporal data plays a central role. Each dimension poses its unique challenges on how to process multivariate time series data.

1.1 Motivation

We start by introducing the main motivation behind our research in multivariate time series and the challenges that we found for the field of data mining as a field that needs to interact with other disciplines.

1.1.1 Data Science

Data-related challenges represent a major scientific effort these days. Such challenges cut across many disciplines, making the data paradigm the broadest of nowadays in terms of disciplines it can potentially affect. This context encourages the blooming of buzzwords and research hypes, such as big data and data science. These buzzwords are so strong that they boosted the focus of computer scientist towards data. As an example, over the last fifteen years, my own institute, the Leiden Institute of Advance Computer Science redirected its educational and research focus mostly towards challenges related to data, such as optimization, data mining, bioinformatics, and computer vision. More recently, terms such as big data fuelled multidisciplinary partnerships and a general understanding that not only statistics, but also computer science needs to support other scientific fields in their data analysis.

Let us start with the term *big data*. There are no strict definitions for what big data really is. This can cause considerable ontological confusion, when different disciplines get together to discuss this term and its challenges. Without pretending to have the best definition of big data, here is a (somewhat simplistic) attempt: *Big data refers to the challenges created by the fact that our present technological capacity to measure and collect data outperforms in great measure our capacity to explore and exploit it.* But then, how big is big data for most? Furthermore, does our present incapacity to deal with big data represent all the challenges around the paradigm of data?

Firstly, when talking about data, size matters. Let us start with the most extreme case, the World Wide Web. Today, if we were to collect and store all the data transferred during one day over all the World Wide Web, this would amount to a pile of burned CDs of data (roughly 700 Mb each) that would stack from Earth to Mars and back [1]. But this is not the case in the majority of data science challenges, where the scale of what is considered to be a challenging amount of data is determined by the tools available. Most of the data mining research around the term big data is about developing new computational methodologies that can upscale fairly standard analyses to Gbs or Tbs of data. Without reducing the merits of this line of research, in fact most fields of research are struggling with datasets in the order of Mbs. This struggle is in many cases due to the usage of standard tools, which are incapable of loading a large file for analysis or are simply not able to plot the data for a first inspection. Research projects need better tools, either for general purposes or in some cases tailored for a specific project.

[1] `http://www.imdb.com/title/tt5275828/`

So is there more to the challenges surrounding data than scalability? I believe there are, and they are multidisciplinary in nature. For a complete approach to any current data project, the scientist's background should include a good domain of databases design and management, algorithm design and performance, statistics, machine learning systems, and optimization, just to name a few. The broader field that involves this multidisciplinary knowledge has been commonly referred as *data science*.

The multidisciplinary nature of data science can be seen as a natural extension of data mining. Data mining is a field that marries methods both from mathematics such as machine learning and statistics with computer science techniques such as databases and design of efficient algorithms. The primary goal of data mining is to extract valuable information from data and turn it into a models that can be further used. Data science extends this modeling challenge both with challenges upstream and downstream.

Upstream data science challenges include turning the attention towards better data collection protocols, efficient data collection systems, faster data communication and data storage solutions (short period, long term storage, work-in-progress storage, multiple user access). For example, recently most of the principal funding agencies of Dutch research institutes (e.g. NWO, H2020 and ERC), started asking project applicants to write a data management plan for each proposal. Increasingly, domain experts in many areas of research are dealing with large amounts of generated data. Among the different research communities, one can find the social sciences, life sciences, arts and design, among others. Many of these domain experts may not have a computer science background, but often they find themselves dealing with data challenges.

Consider now the challenges downstream. They include the development of tools that include other data analysis. Such tools need to be easily available and designed for a broad audience. Additionally, there is a need to transform results of models (predictive and descriptive) into decision support systems, such that results turn into possible actions.

This thesis seeks to provide methods and methodologies for domain experts using sensor data, by building data mining algorithms tailored to sensor data. We are also developing easy-to-use standalone tools where visualization, analysis and building models of time series does not require programming skills.

The research of this thesis has been developed in multiple sensor-related projects. Referencing such projects should enforce the idea that data mining can be the common ground for multidisciplinary projects. The following sections relate to projects in the areas of civil engineering, medical sciences, life sciences (sports) and social sciences.

InfraWatch

Infrastructures such as bridges are built to endure harsh conditions, for instance as heavy traffic or extreme weather conditions. However, it is known that they do not last forever: in the long run, infrastructures deteriorate very gradually and eventually they need to be repaired or replaced. In order to keep traffic moving, bridge owners need to do maintenance. How to predict the necessity for maintenance? The current approach is to do *ad-hoc* external inspections.

InfraWatch is a project that joins asset managers, contractors and academia (both civil engineering and computer sciences) to change the current practice of bridge maintenance. A large sensor network (approximately 145 sensors) was installed on a highway bridge near Amsterdam, collecting data since 2008 at a sampling rate of 100 Hz, generating big data. At the Leiden Institute of Advanced Computer Science, we have been developing algorithms that build data-driven models of the bridge [102, 103, 67, 15] and new visualization tools [101, 65, 13], for large multivariate time series.

Leiden Longevity Study

The baby boomers of the fifties and sixties are aging and as a consequence in Europe, health has become a major concern for people themselves [96], policy makers [92] and science [104, 94]. The *Leiden Longevity Study* is designed to identify genetic and phenotype markers that relate to longevity. A total of 421 families were recruited, consisting of long-lived white siblings, their offspring and the offspring's partners (who act as controls). The collection of data was vast and for some cohort studies, the data collection protocol included unsupervized sensor data (accelerometers and bioharness) for long periods of free living. The question became then: how to extract valuable information about physical activity that can help explaining the aging process.

In order to explore the unsupervized data, first a validation study (GOTOv) produced a dataset with a set of activities that have been annotated (hence a supervised dataset). The participants of this study have the same characteristics as the population as the Leiden Longevity Study studies. Using a method described in this thesis [14], we have been able to create tools (see Chapter 5) and highly reliable activity recognition models [74]. This will allow us to explore the data from the cohorts that have unsupervized data.

Social Competence

Social interactions in the playground have been considered important learning opportunities, for children to learn social skills at preschool years. Specifically, all forms of social play (fantasy, role, exercise, or rough-and-tumble) have been related to children's social competence [21]. The research team Focus on Emotions, of the Faculty of Social and Behavioural Sciences of Leiden University and the Human Motion Faculty of the University of Lisbon have been studying child's play in the playground.

As data science partners, we designed a new data collection protocol to measure interactions in the playground, based on Radio-Frequency Identification Devices. Using active RFIDs as badges, the interactions could be measured for all the children at the same time at a fairly high sampling rate (4 Hz). This partnership resulted in several new analyses [99, 100], as well as a first place in a smart city competition [1]. Collection of additional datasets is ongoing.

Sports Analytics: MASS project

The MASS project focuses on the monitoring and analysis of elite Dutch speed skaters during several training seasons. The goal is to consider a wide range of aspects, in order to find effective patterns in their training routines, which might lead to more effective training and better competition performance. The project builds on a unique collection of daily data about Dutch speed skaters. Collected over the last ten years, this data comes from within LottoNL-Jumbo, the top-level team of coach Jac Orie [73].

In order to mine this data, first we set up a database infrastructure to store the data and developed a web-based tool where one can make data and model analysis on the fly with ease. The research resulted in a high profile article [53] (see Chapter 6) as well as considerable publicity.

1.1.2 Research Questions

The applied projects outlined above, as well as many similar projects producing complex, multivariate time series, offer many opportunities for new research in the area of data science. The following is a list of research questions inspired by these projects that motivate the work outlined in this thesis. The first two questions have to do with the multivariate nature of the data itself:

> **Q1** How can we find unsupervised patterns in multivariate time series?
>
> **Q2** Can we find dependencies between variables in multivariate data that occur intermittently?

These two questions relate to applications where the observed system can be in different states with different observed behaviour, but it is not quite clear yet which are the states it can be in. In other words, this is a so-called unsupervised setting, where one would like the data to be clustered into a finite number of states, but the states should be reasonably steady, and not switch from one time point to the next. This setting is for example relevant in the Leiden Longevity Study, where one would like to recognise various activities in a person's behaviour. Additionally, research question Q2 relates to the possibility that different states show different phenomena with respect to the measured variables involved. Research questions Q1 and Q2 are addressed in Chapter 2.

The unsupervised setting is attractive if one is unfamiliar with the dataset and domain it captures. However, in many cases, one is dealing with a supervised task, and either regression or classification models need to be induced from multivariate data. A specific challenge, that was also present in some of the projects mentioned above, is that the supervised 'labels' are not always available at the same rate as the detailed multivariate data was recorded. For example, it is easy to capture activity data at high sampling rates, e.g. by means of an accelerometer, but obtaining the activity of a person can realistically not be recorded more often than once per second, if not much slower. This inspires the following research questions, which are addressed in Chapters 4, 5 and 6:

> **Q3** How can we learn classification and regression models from multivariate time series with mixed sampling rates, taking into account the different temporal scales at which dependencies might hold?

Q4 How can we automatically derive good time series features that take into account delays and temporal aggregation?

In the field of sports analytics, one is interested in reliable models, but also in finding patterns that are actionable and interpretable for a coach, in order to optimise the effectiveness of their training routines. The MASS project above motivated research question Q5:

Q5 How can interpretable and simple patterns be extracted from multivariate training data, while considering the complexities of elite sports and of how the human body responds to various training impulses?

The above research questions relate to key data science methods that are required to deal with the data challenges described, but they over-simplify the process of data science. In practice, a project requires a methodology and tools to support the analysis of complex data. The following research question addresses this issue, which is also the topic of Chapters 3, 5, and parts of Chapter 6.

Q6 How can the non-trivial analysis of multivariate times series by supported by a methodology and a tool?

1.2 Multivariate Time Series

In this section, we introduce the fundamental concepts used throughout this thesis. To start, consider a univariate time series, defined as follows:
Definition 1. *A* univariate time series $\mathbf{t}$ *is a finite sequence of values* $(t_1, \ldots, t_n)$ *ordered in time, where i represents an index of the sequence* $i \in \{1, \ldots, n\}$ *and t_i is a real value:* $\forall i \;\; t_i \in \mathbb{R}$.

Consider now the properties of a single time series, $\mathbf{t}$. In this univariate setting, the sequence of data points $t_i, i \in \{1, \ldots, n\}$ represents a process measured over a period of time.

Across all the projects mentioned above and chapters following this introduction, there is a data structure that is common to all: the *multivariate* time series. Continue to assume that our time series is of finite length n. The extension to the definition above is that now we have m variables to consider. We formalize a multivariate time series as a matrix $\mathbf{T}$ of size $n \times m$, as follows:

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}_{1,1} & \mathbf{T}_{1,2} & \cdots & \mathbf{T}_{1,m} \\ \mathbf{T}_{2,1} & \mathbf{T}_{2,2} & \cdots & \mathbf{T}_{2,m} \\ \vdots & \vdots & \mathbf{T}_{i,j} & \vdots \\ \mathbf{T}_{n,1} & \mathbf{T}_{n,2} & \cdots & \mathbf{T}_{n,2} \end{pmatrix}$$

Please note that in this matrix, there is a temporal order: $\mathbf{T}_{i,j}$ represents a measurement at time point i for variable j. The temporal order of the data can be represented as a sequence $\{1, 2, \ldots, i, \ldots, n\}$. Note that this representation assumes that all measurements for the different variables j are performed at the same time.

1.2.1 Sampling Rates

Definition 1 is an abstract formalization of a time series without specifications of the exact timing of the data. It is abstract because the series is just ordered by an index from $\{1, \ldots, n\}$. In reality, however, each row in the series is actually measured at a specific moment in time, which we refer to as the *timestamps* of the measurement. A timestamp can be seen as a function $\omega(i)$ that maps the index i to a specific point in time.

Having the timestamps, one important property to know is which is the temporal spacing between consecutive measurements. For consecutive measurements, this spacing is simply:

$$\omega(i) - \omega(i-1), \forall i \in \{2, \ldots, n\}.$$

A more informative measure of how measurements are spaced is actually the *sampling rate*, which is the average number of samples per unit of time:

$$f = \frac{n-1}{\omega(n) - \omega(1)}.$$

The sampling rate is a frequency with unit s^{-1} (per second), more generally referred to as Herz (Hz). The sampling rate f is an average statistic that holds for the entire series. In contrast, the sampling rate can also be computed locally, as follows:

$$f_i = \frac{1}{\omega(i) - \omega(i-1)}, \forall i \in \{2, \ldots, n\}.$$

Fixed Sampling Rate The most desirable situation is when the sampling rate is fixed, so the temporal spacing between measurements is always the same. Note that well-behaved periodicity of the data will allow us to ignore time stamps and simply use indexes. We refer to *fixed sampling rate* when the sampling rate f_i is always constant, such that

$$f = f_i = \frac{1}{\omega(i) - \omega(i-1)}, \forall i \in \{2, \dots, n\}.$$

Variable Sampling Rate When the statement above does not hold, we refer to it as *variable sampling rate*. Notice that variable sampling rate is often the case, sometimes due to imprecision in measuring, other times because the measured phenomena does not have a constant periodicity.

There are multiple standard approaches to deal with variable sampling rates. If the differences between sampling rates f_i is very small, then one can just opt to ignore it. Alternatively, one could find solutions either from the data perspective or from the design of the algorithm. From the perspective of the data, one can have a pre-processing step to make the sampling rate constant, by using standard techniques such as interpolation or repeating values [105, 83]. From the algorithmic perspective, one can design and implement solutions to deal with variable sampling rates, with techniques such as buffering [9].

1.2.2 Machine Learning Tasks

The machine learning literature identifies a range of standard tasks. These differ in terms of nature of the data, such as the traditional tabular data, as opposed to structured data (e.g. time series). Machine learning tasks can also differ in terms of aim, such as classification, regression, biclustering, subgroup discovery and segmentation. This suggests a landscape of problems and solutions where it is easy to get lost.

One good way to categorize machine learning tasks is to divide them into two learning paradigms: unsupervized and supervized. This nomenclature separates tasks with a clear, known and measurable target (supervized) and tasks that try to make sense of data without having a prior target (unsupervized). This thesis focuses on tasks from both paradigms that are relevant for multivariate time series, as mentioned bellow.

Unsupervized Learning

The field of unsupervized learning is vast, and reflects the different ways one could extract meaningful information from data. As for time series, examples of well-known machine learning tasks are *whole time series clustering* [22, 49, 2], *segmentation* [35, 46, 47] and *motif discovery* [72, 20, 71, 103, 88, 70, 93].

Whole time series clustering applies when one is facing a dataset where the same variable was measured in multiple systems, under the same protocol. Take as an example measuring the core body temperature of a person. If this would be done for multiple persons, then we could cluster the different types of metabolism in terms of temperature [22]. Although common, this setting does not apply to our definition of multivariate time series, where it is assumed that we are measuring only one system.

Let us now focus on the situation of measuring one system over time. In such situations, the most well-known tasks are *segmentation* and *motif discovery.* On the one side, segmentation aims to cut a time series into a set of segments, where the data inside each segment is homogeneous. On the other side, motif discovery finds recurrent patterns or motifs in the time series, allowing the possibility of selecting only some parts of the time series. Both segmentation and motif discovery are tasks that typically are applied to univariate data, not easily to multivariate time series.

In this thesis, we introduce an unsupervized task that is applicable to multivariate data collected from a single system: biclustering of multivariate time series. Biclustering is a well-known task in the machine learning literature, but cannot be readily applied to time series data. We reinterpret this task into a scenario of multivariate time series. An introduction to this task, a formal definition, an algorithmic solution and tool can be found in Chapters 2 and 3.

Supervized Learning

Consider now that we have a known and measurable target to model. In such a case, we are dealing with a different paradigm from the one above, known as supervized learning. In such a case, one wants to predict a target. In the case of this thesis, we will focus on predictions in a multivariate scenario.

Before moving on to the definition of the target, let us make an important distinction between prediction of future data (forecast a system) and prediction of current and past data (understand a system). Forecasting is to estimate future values given past values of a time series. Please note that forecasting can be done both in univariate and multivariate settings. Unlike forecasting, prediction of existing data seeks ways to explain some target in terms of other variables. This sort of prediction can be divided into regression or classification problems. Such models are invariably multivariate. The fundamental difference to forecast resides in the error. The regression or classification error is the difference between the actual value of the estimation and the prediction of such value. We focus on this set of tasks and formalize them bellow.

Consider that besides the m independent variables present in $\mathbf{T}$, we have a dependent variable $\mathbf{r}$. Having $\mathbf{r}$ as a target, the general task in this situation is to find a model $\mathcal{F}$, such that:

$$\mathbf{r}_i = \mathcal{F}(\mathbf{T}_i) + e_i$$

Please note that e_i represents the error of the model for the prediction of existing data as mentioned a couple of paragraphs above. For both classification and regression models, the objective is to minimize this error.

Classification A problem can be categorized as a classification problem when the dependent variable $\mathbf{r}$ is categorical. When categorical, the domain of $\mathbf{r}$ assumes a finite number c of possible, such that $\mathbf{r}_i \in \{1, \ldots, c\}$. To evaluate a classification model, normally one would use accuracy as a metric. To define the accuracy, please consider a binary classification problem ($c = 2$), with the following terminology in terms of prediction.

$$accuracy = \frac{number\ of\ cases\ correctly\ predicted}{number\ of\ cases}$$

Please note that this accuracy rate can be generalized to a multiclass classification problem where $c > 2$. The accuracy reflects the quality of a classification model, due to its capability to correctly predict the classes. The closer to unity, the better are the predictions performed by the classifier.

Regression Next to classification problems, we have regression problems. A regression problem is recognized as such when the dependent variable $\mathbf{r}$ is numeric, such that $\mathbf{r}_i \in \mathbb{R}$. In such a numeric setting, the error can be either an overestimation or an underestimation and is calculated as follows:

$$e_i = \sqrt{(\mathcal{F}(\mathbf{T}_i) - \mathbf{r}_i)^2}$$

Please note that as measure of quality for a regression model, one would want a normalized measure to be able to compare methods across different datasets or just to avoid the interpretation using the domain size of the dependent variable $\mathbf{r}$. For such a measure, we normally adopt r-squared (R^2), which is defined as follows:

$$R^2 = \frac{\sum_{i=1}^{n} {e_i}^2}{\sum_{i=1}^{n} (\mathbf{r}_i - \bar{\mathbf{r}})^2},$$

where $\bar{\mathbf{r}}$ is the average of the dependent variable. R-squared is a measure of quality of the regression model by comparison with a simple average estimation. The closer to unity, the best is the model and the further is from being an average estimation.

Mixed sampling rates In many cases, the sampling rate for $\mathbf{T}$ is very high or at least higher than desired as an output. Here as output, we refer to the result of a regression or classification task as defined above. As an example, please consider the InfraWatch bridge mentioned previously, with an installed sensor network that measures at 100 Hz and where at best, we want to know the traffic intensity every minute. For such situations, we aim to deal with modeling tasks where the sampling rate of the dependent variable $f_{\mathbf{r}}$ is (much) lower than that of the independent variables $\mathbf{T}$

$$f_{\mathbf{r}} \leq f_{\mathbf{T}}.$$

Please note that a consequence of such an assumption is that the length of $\mathbf{r}$ is not n anymore but $|\mathbf{r}|$. The differences of sampling rates imply differences in the number of samples between $\mathbf{T}$ and $\mathbf{r}$, where $|\mathbf{r}| < n$. The regression or classification models as stated above won't hold under such circumstance and one would need to change the data such that for each value of the dependent variable $\mathbf{r}$, there is only one vector of values as independent variables.

The transformation of independent variables we normally call feature construction. As described above, in the supervized settings we are looking for a function $\mathcal{F}$ that is able to explain $\mathbf{r}$. In the case of mixed sampling rates, we have a situation that $\mathbf{T}$ cannot be used to model $\mathbf{r}$ directly. To model $\mathbf{r}$, we need to transform $\mathbf{T}$ into a feature set $\mathbf{F}$, such that for any of the features that belong to the set $\mathbf{F}$ have length $|\mathbf{r}|$. We approach this transformation of $\mathbf{T}$ by means of *aggregation*, which is described below.

1.2.3 Aggregation of Time Series

In this section, we discuss the relevance of so-called *aggregation functions* to deal with time series. Starting from the beginning, one could ask what an aggregation function does. In short, it takes a vector of numbers and transforms it into a lower dimension, typically a single value. As an example, let's consider a univariate time series $\mathbf{t}$ according to definition 1. An aggregation function summarizes $\mathbf{t}$ as follows:

$$a : \mathbb{R}^n \rightarrow \mathbb{R}^1.$$

Please note that the usage of an aggregation function such as a, intends not only to perform a dimensionality reduction but also to summarize some particular informative aspect of $\mathbf{t}$.

There are many ways to summarize sequences of measurements. To consider such variety, we assume a set of aggregation functions A, where $a \in A$. Although the set A can theoretically be defined to include an infinite number of functions, we will restrict A to a very manageable number. We consider A to be a family of well-known aggregation functions, normally designated as *descriptive statistics*, such as a minimum, a maximum or an average. More on A can be found in Chapter 4.

Let's now focus on how to properly use aggregation functions in the context of time series. Above, we considered applying such functions to summarize the whole time series $\mathbf{t}$. Although it works well as an example, this is not how aggregation is applied in most practical cases. More commonly, a is applied to subsequences of a time series, informally referred to as a *window*. A time series subsequence is defined as follows:

Definition 2. *Given a univariate time series* $\mathbf{t}$ *as in Definition 1, a subsequence* $\mathbf{s}_{\mathbf{t},i,l}$ *is a subset of* $1, \ldots, n$*, containing* l *contiguous values. We consider* $\mathbf{s}_{\mathbf{t},i,l}$ *to be represented by a vector* $(t_{i-l+1}, \ldots, t_{i-1}, t_i)$*, and:*

- $i \in \{l, \ldots, n\}$,
- $\forall i \;\; t_i \in \mathbb{R}$,
- $1 \leq l \leq n$.

Aggregation functions, when applied to time series subsequences, will summarize a particular period of time. Such usage of aggregation function in subsequences of time series is normally intended for feature construction. In the context of this thesis, the need for feature construction has already been motivated above, while discussing how to mine multivariate time series with mixed sampling rates.

One could now consider how a feature is constructed in the context of a time series. For the purpose of simplicity, let us start by considering a feature that has the same sampling rate as the original time series. This implies that the aggregation function slides orderly through the subsequence (one data value in, one out). A feature can then be constructed as follows:

$$f_{t,a,l}[i] = a(\mathbf{s}_{\mathbf{t},i,l})$$

A more generic definition of time series feature (aggregate feature) is given in Chapter 4. Please note that we assumed for now that features are constructed with subsequences of the same length l. Such an assumption works fine when we have a fixed sampling rate. As presented in Section 1.2.1, it can also occur that one faces data with variable sampling rates. In such a scenario, there are two solutions to consider:

- If the variation of the sampling rate is considerable, one could fix a time interval and include all the data that falls into that window of time.

- If the variation of the sampling rate is insignificant, one could just do some pre-processing to adjust the sampling rate and still fix the size of the window based on the index.

A feature as defined above captures some particular aspect of a time series. Following this idea, we identify the following applications of aggregate features:

- Feature engineering: capture or extract some specific aspect of the time series that can potentially be informative, and is not apparent in the original time series.

- Smoothing: transform the time series into a smoother version of itself. Here the idea is to reduce the noise that naturally occurs in data.

- Integration over time: with integration over time we mean extracting some aspect of the data that exhibits an influence over some time.

- Delays: the idea of capturing delays is important in time series because sometimes the causal effect is not instantaneous. Being able to capture information with delays might help to do such integration over time.

1.3 Main Contributions

In this section, we describe the main contributions of this thesis. The contributions will be organized by chapters that follow this introduction. Each chapter corresponds to one article. Three of the chapters have been published in the proceedings of computer sciences conferences, one in a high impact journal and a remaining one that is under submission process.

biclusTS: Biclustering Multivariate Time Series In Chapter 2, we introduce biclustering as a relevant task for multivariate time series data. Although biclustering is well-established as a machine learning task, it is primarily considered in the context of classical tabular data [19, 61, 79]. Existing solutions do not work with multivariate time series because such algorithms may pick single rows (in this context, timestamps). When applied to multivariate time series, a classical algorithm will produce biclusters with cluster members scattered across all the time series, with little or no relevance when seen from a temporal perspective.

Our algorithm uses as coherence measure between segments, namely a one-step estimation of the density difference between distributions of values within different segments. Such an estimation of the density difference differs from the traditional two-step approach that first estimates the probability density functions and then calculates the estimated density difference of the two distributions. The work in this chapter was published as follows:

> Ricardo Cachucho, Siegfried Nijssen, Arno Knobbe, *Biclustering Multivariate Time Series*, in Proceedings of Intelligent Data Analysis (IDA), 2017

Bipeline: Bisclustering tool In Chapter 3, we introduce a working tool, easily accessible to everyone that wants to experiment with biclustering methods in multivariate time series. We called this tool *Bipeline*, due to the main idea of allowing users to follow easily an experimental pipeline. Such a pipeline means that this tool can be used to load, visualize, preprocess and bicluster multivariate time series. In terms of biclustering, the intention of Bipeline is not only to share the algorithm biclusTS (Chapter 2), but also to compare it to other standard biclustering algorithms. The following paper describes the tool own detail:

> Ricardo Cachucho, Kaihua Liu, Siegfried Nijssen, Arno Knobbe, *Bipeline: a Web-based Visualization Tool for Biclustering of Multivariate Time Series*, in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2016

Accordion: Mining Mixed Sampling Rates In Chapter 4, we change the learning paradigm to a supervized scenario. Here, we introduce a new method that deals with multivariate time series with *mixed sampling rates*. We named this algorithm *Accordion*. Accordion automatically searches for good aggregate features for a given dataset with mixed sampling rates. As a result, Accordion returns a set of aggregate features at the sampling rate of the dependent variable.

What sets our method apart from many has to do with the way good descriptive features are searched automatically. While the majority of classification and regression algorithms see the feature construction as an independent preprocessing step, we see it as a machine learning problem. The construction of good aggregate features is an optimization procedure, and as a result, good features will lead to good models. The obvious question is then, what is a good feature? We consider a feature to be good, when it is able to properly discriminate target classes (classification) or when it has a good correlation with a numeric target (regression). This work was publishes as:

> Ricardo Cachucho, Marvin Meeng, Ugo Vespier, Siegfried Nijssen, Arno Knobbe, *Mining Multivariate Time Series with Mixed Sampling Rates*, in Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp), 2014

ClaRe: Classification and Regression tool In Chapter 5, we introduce a web-based tool (ClaRe) to build regression and classification models using features from the Accordion algorithm (Chapter 4).

Our tool aims to make Accordion accessible in a SaaS (Software as a Service) environment. Given a multivariate time series dataset, it allows users to easily access Accordion's features. The upside of such a SaaS is the ability to access Accordion embedded in a data mining methodology: load, pre-process, visualize and evaluate. The following paper is under submission:

> Ricardo Cachucho, Stelios Paraschiakos, Kaihua Liu, Benjamin van der Burgh, Arno Knobbe, *ClaRe: Classification and Regression Tool for Multivariate Time Series*, in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2018

Speed Skating Analytics Chapter 6 reports on a cooperation between an elite speed skating team (LottoNL-Jumbo) and Leiden University, where the objective is to analyze and optimize the speed skating performance of individual athletes. To look into performance, we accessed daily training and competition results data, collected over a period of over 15 years.

From a modeling perspective, the challenge was to find meaningful aggregate features to avoid situations of under- or overtraining in the athletes' training schedules. The challenge from the data science perspective was on how to weigh such aggregate periods and on how to deal with variable sampling rates in multivariate time series. The models presented represent a mixture of linear models and subgroup discovery (step functions) that should provide a platform of analysis for personalized performance optimization. The content of this chapter appears in the following publication:

> Arno Knobbe, Jac Orie, Nico Hofman, Benjamin van der Burgh, Ricardo Cachucho, *Sports Analytics for Professional Speed Skating*, in journal of the Data Mining and Knowledge Discovery, Volume 31, Issue 6, pp 1872–1902, Springer, 2017

Please note that, in the the case of this publication I am not the first author. In this case, I designed and developed the application where the experiments ran, conducted the experiments to find meaningful aggregate features used in the performance models and lead LIACS data scientist.

1.4 Thesis outline

Following this introductory chapter, this thesis presents a series of papers. These are papers that have been published and peer reviewed, with the exception of Chapter 5 that has been submitted for publication. The papers are presented in the form of self-contained chapters. Although being self-contained chapters, one could divide the work into chapters on unsupervized tasks and chapters on supervized tasks.

Unsupervized learning Chapter 2, Biclustering Multivariate Time Series [15], presents a method on how to select subsets of rows (time periods) and subsets of columns (variables) under a coherence measure. This paper was published at the Intelligent Data Analysis 2017 conference.

Chapter 3, Bipeline: a Web-based Visualization Tool for Biclustering of Multivariate Time Series [13], presents a web-based tool where users can test different biclustering algorithms in multivariate time series. This paper was published at the ECML-PKDD 2016 conference.

Supervized learning In Chapter 4, we change to a supervized learning setting. This chapter, Mining Multivariate Time Series with Mixed Sampling Rates [14], presents a method to search for good aggregate features to build decision trees and linear regression models. This paper has been submitted and presented at the Ubiquitous Computing 2014 conference.

Chapter 5, ClaRe: Classification and Regression Tool for Multivariate Time Series, presents a web-based tool where the users can experiment with aggregate features as presented in Chapter 4 within a data mining methodology framework. This paper was published at the ECML-PKDD 2016 conference.

As for Chapter 6, Sports Analytics for Professional Speed Skating [53], we present an empirical study on how to apply data science techniques to a professional sports environment. This paper was published in the Data Mining and Knowledge Discovery journal.

Finally, in Chapter 7 we give an overview of the findings and contributions that this thesis proposes.

Chapter 2

Biclustering Multivariate Time Series

Ricardo Cachucho, Siegfried Nijssen, Arno Knobbe

in Proceedings of Intelligent Data Analysis (IDA), 2017

Abstract

Sensor networks are able to generate large amounts of unsupervized multivariate time series data. Understanding this data is a non-trivial task: not only patterns in the time series for individual variables can be of interest, it can also be important to understand the relations between patterns in different variables. In this paper, we present a novel data mining task that aims for a better understanding of the prominent patterns present in multivariate time series: multivariate time series biclustering*. This task involves the discovery of subsets of variables that show consistent behavior in a number of shared time segments. We present a biclustering method, BiclusTS, to solve this task. Extensive experimental results show that, in contrast to several traditional biclustering methods, with our method the discovered biclusters respect the temporal nature of the data. In the spirit of reproducible research, code, datasets and an experimentation tool are made publicly available to help the dissemination of the method.*

2.1 Introduction

Multivariate time series are becoming increasingly available, mostly through the rapid development of measurement systems. More and more, machinery, infrastructures and humans use sensors, collecting data synchronously over time. The continuous measuring of these systems also means that most datasets produced by them are unsupervized. These datasets contain a range of phenomena, from recurring phenomena recognizable across all the sensors, to some phenomena that are only recognizable in some of the signals or even random events that do not reoccur at all. The nature of multivariate time series data offers a big opportunity for pattern recognition. As a consequence, there is a considerable need for unsupervized methods that can provide insight in this data.

Among unsupervized tasks that have been studied in the time series field are the *segmentation* and *motif discovery* tasks [103]. These tasks aim to either identify recurring patterns in the time series, or to partition the time series into segments. Most of this earlier work is however biased towards univariate time series. Unlike the univariate setting, mining multivariate time series poses the following challenges:

- understanding the relationship between different variables is highly important;
- the relationships between different variables may alternate at different moments in time;

As a result, there is a need for new methods that identify subsets of variables with consistent behavior over subsets of time. Which methods can identify such subsets? We consider *biclustering* to be a good approach to solve this pattern recognition problem.

Biclustering is a well-known task in the literature [19, 61, 79, 51, 7, 91, 72, 42, 43]. Essentially, it aims at discovering subsets of rows with corresponding subsets of columns, such that the submatrix selected by these rows and columns satisfies a certain cohesiveness requirement. Biclustering can be seen as a generalization of clustering where not only the rows are clustered, but the columns as well. Please note that one bicluster is the result of the biclustering task. Furthermore, biclusters can overlap each other and not all rows or columns need to be included in the biclusters. However, as we will point out throughout the paper, traditional biclustering methods are not well-suited for analysing multivariate time series.

The traditional methods for biclustering ignore the temporal aspect of the data in the analysis. The possibility to shuffle the rows and include rows individually, imply that the time series will no longer be ordered. Traditional biclustering algorithms can include an arbitrary subset of time points in a bicluster; there is no guarantee that contiguous time points are included in the bicluster. As a result, very small segments of the time series, or even individual time points, may be selected in the bicluster.

We argue that discovering biclusters is also useful on time series data. Using the smartphone as an example, we know that people will walk and might cycle at some point through the day. To capture walking, we only need accelerometry data, while for cycling, we would likely to benefit from additional GPS data. However, as we will point out throughout the paper, traditional biclustering methods are not well-suited for analysing multivariate time series, as they ignore in the analysis the temporal aspect of data. They can include an arbitrary subset of time points in a bicluster; there is no guarantee that contiguous time points are included in the bicluster. As a result, very small segments of the time series may be included in the bicluster.

In this paper, we propose a new biclustering method that finds recurring patterns over time for multivariate time series. The distinguishing feature of our biclustering method is that it finds biclusters spanning sufficiently long segments of time. As a result, the biclusters that are found can be see as recurring motifs in multivariate time series data. Our algorithm includes:

- a search strategy for finding biclusters in time;
- the definition of a coherence measure to score the quality of biclusters in time;

Key advantages of our method compared to other methods include:

- the ability to deal with numeric data in a continuous scale (no discretization is needed);
- interpretability of the results.

In the following sections, we define the biclustering problem for multivariate time series, present our biclustering method and experimental results and at last a conclusion. Our experiments show promising results, both in terms of the quality of the results and scalability of the method. For reproducibility purposes, we make our code and datasets freely available. Additionally, we also developed and published an experimentation tool [13] with intuitive GUI, where all the experiments described can be easily tested and applied to various datasets.

2.2 Preliminaries

In this section, we define the main concepts of our problem: multivariate time series (Section 4.2.1), followed by the problem statement of biclustering multivariate time series (Section 4.2.3) and the definition of the similarity measure (Section 2.2.3) used both to segmentation and biclustering of multivariate time series.

2.2.1 Multivariate Time Series

We assume that we are given a multivariate time series of length n over m variables. This time series is represented in an $n \times m$ matrix $\mathbf{T}$. In this matrix, $\mathbf{T}_{ij}$ represents the measurement at time point i for variable j.

We will often need to identify parts of this data matrix. We introduce some notation to make this easier. Let $I \subseteq \{1, \ldots, n\}$ be a subset of time points and let $J \subseteq \{1, \ldots, m\}$ be a subset of variables, then $\mathbf{T}_{IJ}$ defines the following submatrix:

$$\mathbf{T}_{IJ} = \begin{pmatrix} \mathbf{T}_{I_1J_1} & \mathbf{T}_{I_1J_2} & \cdots & \mathbf{T}_{I_1J_k} \\ \mathbf{T}_{I_2J_1} & \mathbf{T}_{I_2J_2} & \cdots & \mathbf{T}_{I_2J_k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{I_\ell J_1} & \mathbf{T}_{I_\ell J_2} & \cdots & \mathbf{T}_{I_\ell J_k} \end{pmatrix},$$

where I_u is the uth element in the set I, $k = |J|$ and $\ell = |I|$. Furthermore, we will use $\mathbf{T}_{i\bullet}$ as a shorthand for T_{IJ} with $I = \{i\}$ and $J = \{1, \ldots, m\}$ and $\mathbf{T}_{\bullet j}$ as a shorthand for T_{IJ} with $I = \{1, \ldots, n\}$ and $J = \{j\}$.

2.2.2 Problem Statement

As discussed in the introduction, we are interested in identifying biclusters in time series data. More formally, we define the problem at hand as follows.

Definition 3. *One* bicluster in time series data *consists of:*

- *selected segments $\mathcal{I} = \{I_1, \ldots, I_q\}$, where each segment consists of contiguous measurements $I_x = \{a_x, a_x + 1, \ldots, b_x\}$, for some $a_x, b_x \in \{1, \ldots, n\}$ and $I_x \cap I_y = \varnothing$ if $x \neq y$;*
- *selected variables $J \subseteq \{1, \ldots, m\}$,*

such that:

- *the selected segments and variables satisfy the following requirement:*

  $$H(\mathcal{I}, J) < \delta,$$

  where

  $$H(\mathcal{I}, J) = \frac{1}{|\mathcal{I}||J|} \sum_{j \in J} \sum_{I \in \mathcal{I}} d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup \mathcal{I} j}); \tag{2.1}$$

  here $\cup\mathcal{I} = \cup_{I \in \mathcal{I}} I$ *and* $d(\mathbf{t}_1, \mathbf{t}_2)$ *measures how similar two time series* $\mathbf{t}_1$ *and* $\mathbf{t}_2$ *are; i.e., it is required that each selected segment is similar to the union of the other selected segments, for all chosen variables;*

- J *is maximal: no variable can be added such that the similarity constraint remains satisfied;*

- $\mathcal{I}$ *is maximal: no segment can be added such that the similarity constraint remains satisfied;*

- *each segment* $I \in \mathcal{I}$ *has a sufficient length, i.e.,* $|I| \geq \ell$*.*

Notice that $H(I, J)$ is set to capture the coherence for each column independently. This allows biclusters to be coherent, even if two variables do not share their distribution. This way, two variables can measure the same phenomenon in different ways, depending on the system characteristics and measuring system setup. As an example, high levels of body activity can be measured by high levels of heart rate or sinusoidal patterns of acceleration. For each selected variable in a bicluster, each selected segment is *similar* to the other selected segments for that variable, and we are interested in the largest such bicluster.

2.2.3 Probabilistic divergence score

Many alternatives are possible for the definition of the similarity, $d(\mathbf{t}_1, \mathbf{t}_2)$, between two subsequences. In this paper, we propose to measure the similarity of two subsequences by determining the divergence between the *distributions* of measurements in these subsequences.

Definition 4. *Given two probability densities* $p(x)$ *and* $p'(x)$ *over the same domain, a* divergence score $d(p, p')$ *is a score with the following properties:*

- $d(p, p') \geq 0$*;*
- $d(p, p') = 0$ *iff* $p(x) = p'(x)$ *for all* x*.*

Definition 5. *Given two time series subsequences* $\mathbf{t}_1$ *and* $\mathbf{t}_2$*, where* $\mathbf{t}_1$ *contains samples from probability density* $p(x)$ *and* $\mathbf{t}_2$ *contains samples from probability density* $p'(x)$*, a divergence score for these subsequences is an* estimate *of the divergence of these distributions, i.e.,* $d(\mathbf{t}_1, \mathbf{t}_2)$ *is an estimate of* $d(p, p')$ *as calculated from* $\mathbf{t}_1$ *and* $\mathbf{t}_2$*.*

2.3 Biclustering Multivariate Time Series

The problem formalized in the previous section is hard to solve exactly. In order to solve it, we propose a heuristic method. Our algorithm iteratively removes segments or columns until a bicluster is obtained with the desired quality. Unlike the traditional biclustering algorithm of Cheng and Church [19], our method is designed for multivariate time series, stressing solutions for the temporal aspect of multivariate time series data in a bicluster.

In our setting, given the length of most time series, removing individual rows is not a feasible approach; converging on a bicluster would take too long. Furthermore, it is unlikely that a heuristic that removes rows one by one is likely to lead to segments of good quality. For these reasons, we propose a new approach that works as follows:

1. Identify segment boundaries in the time series of each variable.
2. Combine the segment boundaries of the different attributes to obtain segment boundaries across all variables.
3. Perform a node deletion biclustering algorithm where either columns and all rows between segment boundaries are removed at the same time.

Below, we will discuss the details of this approach.

2.3.1 Time Series Segmentation

The process we propose for segmentation is detailed in Alg. 1. This algorithm computes the density differences between consecutive subsequences of time series using a sliding window approach. The user-defined parameters for this algorithm are as follows. The size of each subsequence (window) is defined by the parameter w. The window is moved over the time series in steps of size *jump*. This parameter is intuitively bounded by $1 \leq jump \leq w$. Note that by setting $jump \geq \ell$, we can ensure that segments are never of length shorter than ℓ.

Algorithm 1 Segmentation.

```
Input: multivariate time series $\mathbf{t}$, threshold for local maximum selection $\theta$,
  window size $w$, jump between consecutive windows $jump, 1 \leq jump \leq w$.
  $\mathcal{C} \leftarrow \{1\}$
  for each column of $\mathbf{T}$ do
    for each $i \in \{0, 1, \ldots, \lfloor (n-w)/jump - 1 \rfloor\}$ do
      $s \leftarrow i \cdot jump$
      $d[i] \leftarrow d(\mathbf{t}[s+1, \ldots, s+w], \mathbf{t}[s+jump+1, \ldots, s+jump+w])$
    end for
    for each $i \in \{2, \ldots, \lfloor (n-w)/jump - 2 \rfloor\}$ do
      if $d[i-1] \leq d[i] \wedge d[i+1] \leq d[i] \wedge d[i] \geq \theta$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{i \cdot jump + w\}$
    end for
  end for
  return segment start indices $\mathcal{C}$
```

Having decided on how to go through the data, one needs to calculate the divergence scores between consecutive windows of data, $d(\mathbf{t}_1, \mathbf{t}_2)$. A solution for this divergence score is proposed in Section 2.3.3. After calculating the divergence score for all consecutive windows, we extract all local maxima to find the segment boundaries. The risk of using local maxima is that too many boundaries might be found, creating too many segments. For this purpose, a threshold θ for the selection of local maximum divergences is introduced. Only local maxima above θ will be selected as segment boundaries. A reasonable solution for this threshold is to normalize each variable and have the boundaries rescaled between 0 and 1 such that $0 \leq \theta \leq 1$.

Alg. 1 returns a set $\mathcal{C}$ consisting of the segment boundaries. This set is the union of all the segments found for each of the m variables. Note that we segment the variables independently of each other; this is important as in biclustering, we assume that the behavior of different variables over time may be different. To consider independency between variables, we need a univariate divergence measure. Alternatively, one could consider a multivariate version of the divergence measure. This would be suitable if the purpose would be to only segment multivariate time series ($\frac{1}{m}\sum_{1}^{m} divergence_{i,j}$). However, a multivariate divergence seems counter-intuitive in the case of biclustering, once we are looking both for subsets of segments and subsets of variables to include in the matrix $\mathbf{T}_{IJ}$. Experiments in Section 2.4.1 present the difference between univariate and multivariate divergence scores.

2.3.2 Biclustering

In this section, we present the *BiclusTS* algorithm to solve the problem of finding multiple biclusters given a multivariate time series. We will discuss the main challenge of recognizing interesting subsets of rows and columns ($\mathbf{T}_{\mathcal{I}J}$), while respecting the temporal nature of time series. We then move into the details of how to find a bicluster and how to explore the data in order to find multiple biclusters.

BiclusTS: Single node deletion

The BiclusTS algorithm is described in Algorithm 2. The algorithm assumes an initial set of segments in the data, as computed by Alg. 1. Then, a greedy process removes segments and variables (columns) that present the largest divergence to the bicluster. During this repeated process, the difference $H(\mathcal{I}, J)$ reduces monotonically until it drops below the acceptability bound δ. The remaining subset of segments $\mathcal{I}$ and columns J is returned as a bicluster. The difference measure $H(\mathcal{I}, J)$ is defined as follows:

$$H(I, J) = \frac{1}{|\mathcal{I}||J|} \sum_{j \in J} \sum_{I \in \mathcal{I}} d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup \mathcal{I} j}) \tag{2.2}$$

Notice that $H(I, J)$ is a probabilistic measure, not interested in the correlation between variables. Instead, it is set to account coherence of each column independently (see Definition 2.1). This allows biclusters to be coherent, even if the behavior between variables is not correlated, as we consider desirable. This is because time series can measure the same phenomenon in different ways, depending on the system characteristics and measuring system setup. As an example, high levels of body activity can be measured both by high levels of heart rate and sinusoidal patterns of acceleration.

The requirements of our method to find a bicluster is to provide a segmented time series, composed of the time series $\mathbf{T}$ itself and the boundaries of each segment (as produced by Alg. 1). This will ensure a faster biclustering procedure and results consistent with the temporal aspects of the multivariate time series. Another requirement of Alg. 2 is a parameter δ that ensures a certain similarity for all segments within each column of the bicluster.

Algorithm 2 BiclusTS: Find One Bicluster.

```
Input: initial set of segments $\mathcal{I}$, acceptability threshold $\delta$.
  let $J$ be the set of all variables
  calculate $H(\mathcal{I}, J)$
  while $H(\mathcal{I}, J) > \delta$ do
    for all segments $I$ and variables $j$ do
      calculate $d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup\mathcal{I}j})$
    end for
    for each segment $I$ of $\mathcal{I}$ do
      calculate $\frac{1}{|J|}\sum_{j\in J} d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup\mathcal{I}j})$
    end for
    for each variable $j$ of $J$ do
      calculate $\frac{1}{|\mathcal{I}|}\sum_{I\in\mathcal{I}} d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup\mathcal{I}j})$
    end for
    find maximum margin divergence; remove the corresponding segment or variable
    recalculate $H(\mathcal{I}, J)$
  end while
  return $\mathbf{T}_{\mathcal{I}J}$, a bicluster that is a submatrix of $\mathbf{T}$
```

Finding a given number of biclusters

Having described the process of finding one bicluster, the challenge of finding a number of biclusters remains. For this task, we propose Alg. 3, which finds k non-overlapping biclusters by iteratively looking for biclusters in those segments that have not been selected yet. As input, we must have an initial segmented multivariate time series T and acceptability bound δ already introduced in Section 2.3.2. Additionally, this algorithm requires k, which is the number of potential biclusters to be found.

In the first iteration, Alg. 3 starts using all the segments $\mathcal{I}$ to find the first bicluster. After finding a bicluster, we add it to the set of biclusters $\mathcal{B}$ and remove all the segments that have been biclustered from the initial set of segments, $\mathcal{I}$. This process is iterated until no segments are left to be included in a new bicluster or the number of potential biclusters, k is reached. We use the word “potential” because there are situations where k is not reached, due to unavailable segments to bicluster, or when the acceptability bound δ is too low to produce any results.

Algorithm 3 Find k biclusters.

```
Input: initial set of segments $\mathcal{I}$, acceptability threshold $\delta$, the desirable
  number of biclusters $k$.
  $\mathcal{B} \leftarrow \varnothing$, an empty set of biclusters
  while $|\mathcal{B}| \leq k$ and $\mathcal{I} \neq \varnothing$ do
    $\mathcal{B} \leftarrow \mathcal{B} \cup BiclusTS(\mathcal{I}, \delta)$
    Remove all segments in $\mathcal{B}$ from $\mathcal{I}$
  end while
  return $\mathcal{B}$, a set of biclusters found in matrix $T$
```

2.3.3 Density-Difference Estimation (LSDD)

An important choice that remains to be specified is which divergence score to use. Time series can assume many different shapes, depending on the phenomena and measurement system, making the comparison between subsequences a non-trivial problem. Obvious solutions for comparing time series would be two-step approaches. For instance, one could first estimate the probability density distributions (PDFs) of both subsequences, and then compare these using an f-divergence measure. As pointed out by [86], the drawback of such approaches is that good estimations will smoothen the PDFs and thus result in under-estimations of density-differences.

Instead of taking such a step-wise approach, we here propose to use a more direct approach, based on calculating a least-squares density-difference (LSDD) [86]. LSDD measures the similarity between two time series by directly estimating density-differences ($f(x)$) between time series subsequences. This method does not require a separate estimation of the time series distributions. LSDD directly estimates the density-difference between two samples, $f(x)$, by fitting a density-difference model $g_\theta(x)$ that minimizes:

$$\underset{\theta}{\operatorname{argmin}} \int \Big(g_\theta(x) - f(x)\Big)^2 \, dt + \lambda \theta^T \theta. \tag{2.3}$$

Note that the second term in this formula is a regularization term. The model $g_\theta(t)$ that is used to estimate the difference is a mixture model of Gaussians:

$$g(x) = \sum_{\ell=1}^{|\mathbf{c}|} \theta_\ell \exp(-\frac{||x - c_\ell||}{2\sigma^2}),$$

where $\mathbf{c}$ is a random sample of measurements in both time series $\mathbf{t}_1$ and $\mathbf{t}_2$.

To fit the model, the equivalence of Equation 2.3 with the following formula is exploited:

$$\underset{\theta}{\operatorname{argmin}} \int g_\theta^2(x)\, dx - 2 \int g_\theta(x) f(x)\, dx + \lambda \theta^T \theta. \tag{2.4}$$

Given that $f(x)$ is unknown, an empirical estimate is used for the second term:

$$\sum_{\ell=1}^{|\mathbf{c}|} \frac{\theta_\ell}{|\mathbf{t}_1|} \sum_{i=1}^{|\mathbf{t}_1|} \exp\Big(- \frac{||t_{1i} - c_\ell||}{2\sigma^2}\Big) - \frac{\theta_\ell}{|\mathbf{t}_2|} \sum_{i=1}^{|\mathbf{t}_2|} \exp\Big(- \frac{||t_{2i} - c_\ell||}{2\sigma^2}\Big).$$

The resulting minimization problem can be solved analytically, as shown by the authors of LSDD [86]. Note that this model fitting procedure has two parameters: the Gaussian kernel width σ and the regularization parameter λ. The authors of LSDD propose to optimize these parameters using cross-validation. When LSDD is used on a large scale, this cross-validation becomes too demanding.

After a study of LSDD's behaviour, reported in Section 2.4.1, in which we compare different subsequences of the same time series, we found that the parameters of LSDD are very stable, i.e., the optimal choice for the parameters values does not change very often for the same time series. Thus, we propose that the parameters are estimated with cross-validation a certain number of times at the beginning of the segmentation task (Alg. 1). Then, we fix these parameters for the rest of the LSDD calculation processes, thus speeding up computational efficiency while keeping the quality as a density difference estimation measure.

LSDD is also considered now the biclustering process for the calculation of $H(\mathcal{I}, J)$. The advantage of using LSDD for computing $H(\mathcal{I}, J)$ is that it allows rich descriptions of the segments to be taken into account in the process of finding each bicluster. From the computational perspective, fixing the parameters of LSDD is beneficial to speed up the process of finding each bicluster, due to the extensive amount of calculations of LSDD in Algorithm 2. This can be seen in Alg. 2, where for all the segments and variables LSDD is used as a divergence measure between two time series subsequences (e.g. $d(\mathbf{T}_{Ij}, \mathbf{T}_{\cup\mathcal{I}j})$). Thus, making of $H(\mathcal{I}, J)$ a mean density difference score.

2.4 Experiments

In this section, we divide the experiments in two parts: segmentation and biclustering. We present experimental results of the segmentation task including how to do LSDD estimation (normalization and cross-validation of parameters) and univariate versus multivariate segmentation. The second part is about biclustering with several experiments comparing traditional biclustering with our proposed method.

We evaluate our method on four datasets, details of which are given in the table below. The datasets were selected for their length and their multivariate nature (with datasets having up to 119 variables). Except for *Accelerometry*, the datasets also have variables that can be grouped in different categories, such that each group will show considerably different behaviour. For example, the *InfraWatch* data [103] is collected from three types of sensor (each sensitive to different phenomena and time scales): strain gauges, vibration sensors, and temperature sensors.

dataset	# variables	# time points	sampling rate	duration
Accelerometry	3	176 700	85 Hz	34.6 min
Snowboarding	21	21 180	1 Hz	5.88 hrs
Running	6	951 200	100 Hz	2.64 hrs
InfraWatch	119	17 996	1/3600 Hz	749.8 days

All experiments were performed with an implementation in R. A demo tool called Bipeline [13], demonstrates this implementation and is easily accessible online (`http://fr.liacs.nl:7000/`). Using Bipeline, one can replicate experiments and can try the various settings and choices. The code is also made available (`https://github.com/kainliu/ShinyDashboard`). The four datasets mentioned above have also been made available, (in accordance with reproducible research standards) and can be found at:

- `www.openml.org/data/download/1854941/accelerometry.csv`
- `www.openml.org/data/download/1854942/infrawatch.csv`
- `www.openml.org/data/download/1854943/running.csv`
- `www.openml.org/data/download/1854944/snowboard.csv`.

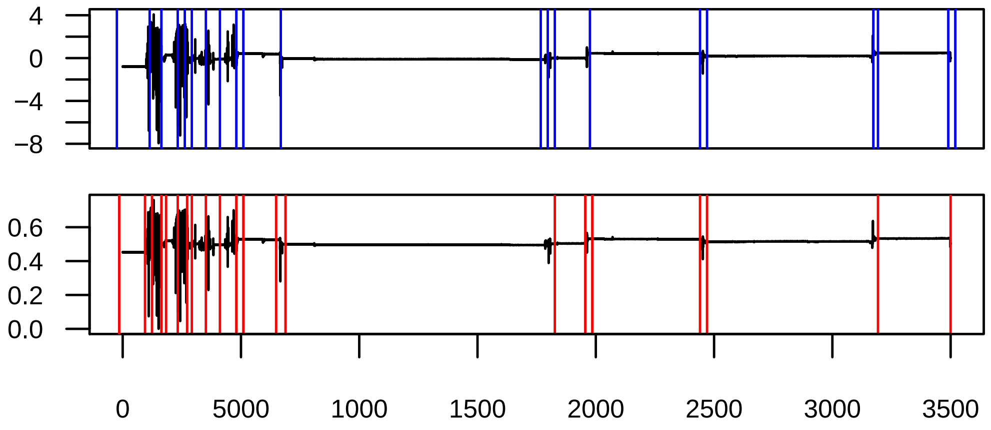


Figure 2.1: Two graphs showing the segmentation obtained on the *X-axis* variable of *Accelerometry* dataset, for the original time series (above) and the normalized one.

2.4.1 Segmentation

In Section 2.3.1, we proposed a solution to segment multivariate time series in order to bicluster them. Here, we present the experimental results that support our decisions on how to solve this segmentation task.

Normalization

Before considering aspects of the actual segmentation, we examine the effect of normalization. As was argued in Section 2.3.3, the usage of LSDD estimation as a divergence score in the multivariate setting requires normalization of each variable. To segment multivariate time series as described in Alg. 1, all the datasets were normalized and the parameter θ was fixed at a value of 0.75. Here, we study the effect of normalization on Alg. 1 results.

In this experiment, we performed comparisons on each variable, to see whether the produced segmentation is notably different, before and after normalization. These experiments demonstrated that this effect is marginal. Fig. 2.1 shows two segmentations produced on the X-axis accelerometer variable of dataset *Accelerometry*. Note that the differences in segmentations in the two settings are minimal, thus indicating that one could normalize the time series as a pre-processing step, ensuring comparable LSDD results across variables during the tasks of segmentation and biclustering.

LSDD estimation

The method we used to compare consecutive subsequences is a single-shot estimation of the difference between probability densities (LSDD), defined in Section 2.3.3. This method is based on fitting a Gaussian model where two parameters, λ and σ, need to be estimated. As was mentioned in Section 2.3.3, we estimate reasonable values for these parameters prior to the LSDD estimation for the entire time series, and then work with these fixed values. Clearly, we are trading off computational speed over accuracy of setting λ and σ always with cross-validation.

Here, the experiments consist of running Alg. 1 for all the datasets, with non-overlapping data windows of 100 samples ($w = jump = 100$), and considering two setups. In one setup, for each iteration we estimate LSDD with cross-validation. All the parameter estimations and computation times are saved. In the other setup, we cross-validate LSDD the first 100 iterations and then fix the parameters, by choosing the median σ and λ.

First, we consider the penalty in time produced by the cross-validation at each subsequence, compared to fixing the parameters at the start. For each dataset, the penalty of computing LSDD with cross-validation was calculated. For each setup of running Alg. 1, all the LSDD estimation times are summed up over all LSDD estimations per variable, and over all variables. Thus, the penalty is the time ratio of computing LSDD with cross-validation to LSDD with fixed parameters (so how many times faster the second is). In other words, how many times slower it is to perform cross-validation as opposed to fixing the parameters. The following time penalties were obtained for the four datasets:

Dataset	Penalty
Accelerometry	172
Snowboarding	62
Running	91
InfraWatch	93

Clearly, doing cross-validation at each subsequence is prohibitively expensive, while our choice to fix parameters at the start of each LSDD estimation is much more realistic.

While clearly being much faster, there is the risk of producing sub-optimal values for λ and σ. We test this end of the trade-off by examining the stability of parameter values when LSDD is estimated using cross-validation. If these values remain mostly the same throughout the LSDD estimation, then we can safely pre-compute the values and fix them. For this experiment, we test parameter settings from a fixed range, identical to that proposed originally [86]: $\lambda \in \{0.001, 0.003, 0.01, 0.031, 0.1, 0.316, 1, 3.162, 10\}$ and $\sigma \in \{0.25, 0.5, 0.75, 1, 1.2, 1.5, 2, 3, 5\}$. For the four datasets and both parameters, the percentage of values found that deviate from the median were obtained, and averaged over all variables:

Dataset	Deviation
Accelerometry	1.2%
Snowboarding	1.5%
Running	0.7%
InfraWatch	3.1%

As seen in the table above, the LSDD parameters (λ and σ) are extremely stable. Here, we do not argue the fact that this solution to fix the parameters is a sub-optimal heuristic, when considering an alternative such as cross-validation. Our argument for the decision of fixing both λ and σ is related with the trade-off between accuracy and computational efficiency. The lost in accuracy is low, with a residual deviation from the cross-corelated parameters and on the other hand the computational efficiency gains are in orders of magnitude of at least two digits. To conclute, the parameters are extremely stable and can be safely estimated and fixed prior to LSDD estimation, with considerable efficiency gains.

Univariate or multivariate segmentation

One choice that influences the segmentation (that can be configured in our online tool) is whether to choose segment boundaries based on density differences in individual variables, or whether to compute them from the average density distances over all variables. Figs. 2.2 and 2.3 demonstrate the effect that this choice has on the boundaries produced. Clearly, segmentation over the combined density differences produces fewer boundaries, and only in locations where clear changes are visible in the majority of the time series. The multivariate segmentation, on the other hand, is much more sensitive to changes only observable in individual time series, and as a result produces more boundaries and thus smaller segments on average. Although either option is available and has its advantages and disadvantages, we feel that the univariate, more detailed approach to segmentation is more in line with the philosophy of biclustering and our goals to discover temporal phenomena in subsets of the variables. For that reason, we adopt the univariate option for all further experiments.

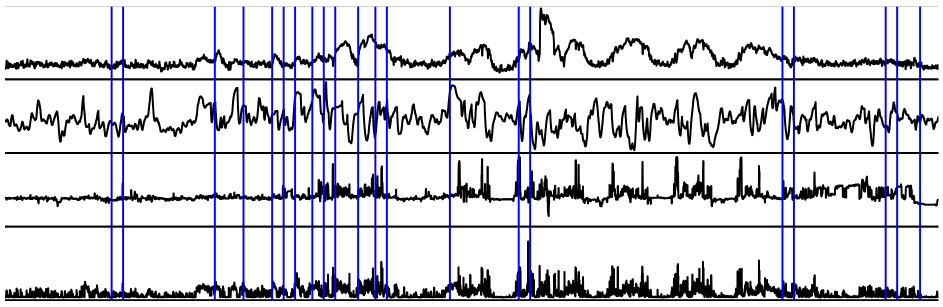


Figure 2.2: Multivariate segmentation example (*Snowboarding* data).

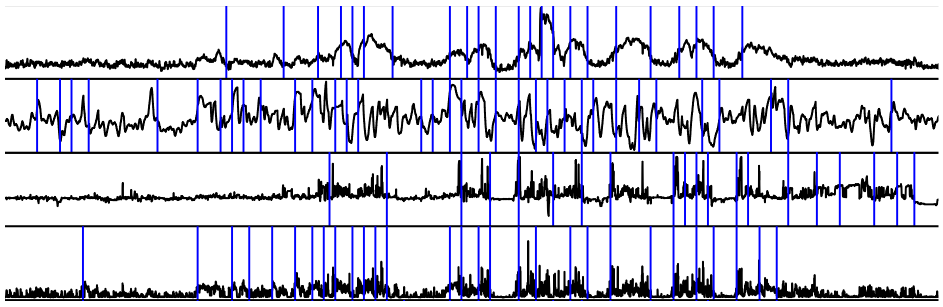


Figure 2.3: Univariate segmentation example.

2.4.2 Biclustering

As a reference point for reproduction of experiments, the parameter settings of our proposed method is presented in the table below.

Datasets	Window	Jump	Delta
Accelerometry	500	100	0.01
Snowboarding	90	30	0.01
Running	500	100	0.01
InfraWatch	120	24	0.01

For all the experimental settings, all parameters defaults can be found in the online tool. Additionally, we present results to further compare with other methods. We observe how many segments were created and the segment's average size. Notice that we want a relatively small number of segments with rather large sizes.

Datasets	Duration	Number segments	Average segment size
Accelerometry	7.55 s	14	642.5 (0.36%)
Snowboarding	81.5 s	24	81.5 (0.38%)
Running	16.7 s	103	1671.8 (0.18%)
InfraWatch	36.2 hrs	21	36.2 (0.20%)

At this point, it is important to understand the differences between the results produced by our approach and those produced by traditional biclustering algorithms, such as the Cheng & Church algorithm. In this experiment, we apply to our four datasets all the available algorithms in the `biclust` package [42] in R. This means that experiments were run to compare BiclusTS with the following biclustering algorithms: Cheng & Church (C&C) [19], the Xmotifs biclustering algorithm [72], the Plaid model [91], Bimax [79], and Questmotif [43, 72].

In the table below, we present the number of segments created by each algorithm. Please note that some algorithms are not even able to deal with large datasets (represented by -). The numbers of segments are in most case more than those resulting from BiclusTS, showing that fragmentation is a systematic problem of the traditional biclustering algorithms.

	Number segments					
Datasets	C&C	Xmotifs	Plaid	Bimax	Questmotif	BiclustTS
Accelerometry	4 034	1 863	1 641	141	3 338	14
Snowboarding	22	547	1	-	13	24
Running	3 852	3 852	9	-	34 791	103
InfraWatch	17	64	-	1	98	21

Complementary to the number of segments, one should look at the average size of the segments biclustered. The average size of the segments created by the traditional biclustering algorithms is in most cases very small. Interestingly, in two cases (indicated by the *), the biclustering algorithms produced segments stretching the entire length of the time series, thus failing to identify any meaningful segmentation into different activities. Notice that Plaid has some exceptions. Still, take as an example the *Snowboarding* dataset. For this dataset, Plaid created a bicluster containing all the data.

	Average segments size					
Datasets	C&C	Xmotifs	Plaid	Bimax	Quest	BiclustTS
Accelerometry	13.6	16.8	44.5	1.1	20.4	642.5
Snowboarding	2.8	15.1	21 180*	-	1420.4	81.5
Running	1.5	2.8	105 688	-	2.7	1671.8
InfraWatch	5.7	2.3	-	17 996*	83.1	36.2

Capturing distributions

The biclustering component of our method compares different segments, in order to cluster similar ones. Central to our approach is the choice to capture such similarity by differences in the distribution within each segment, by means of LSDD. A distribution is a fairly rich way to describe a set of measurements, and one could argue that simpler descriptions of segments could work as well, and might be more efficient also. In this experiment, rather than relying on the one-shot LSDD, we describe each segment of a time series simply by the average value, and apply biclustering to this simplified intermediate representation.

As can be expected, representing segments with only an average tends to cluster together segments that have very little in common, except for an average value that happens to be similar. Fig. 2.4 shows two examples of segments that were clustered in such an undesirable way. Note that for steady time series, this solution will actually work just fine. The problem only occurs with sinusoidal or other more complex periodic shapes, which are in fact very common in time series. The corresponding clustering by BiclusTS does not produce these meaningless groupings of segments.

Demonstration

Our method was designed to find biclusters that avoid very short segments of consecutive time points. However, having good subsets of time periods represented in the bicluster is not enough. BiclusTS also aims to capture interesting phenomena involving complex patterns. These patterns can be observed visually.

In order to show what can be expected from BiclusTS, we applied it to the *Snowboarding* dataset. With 21 variables, this dataset measures a person using a BioHarness chest sensor while riding a snowboard. The sensor measures vital signs such as heart rate, breath rate and body temperature, as

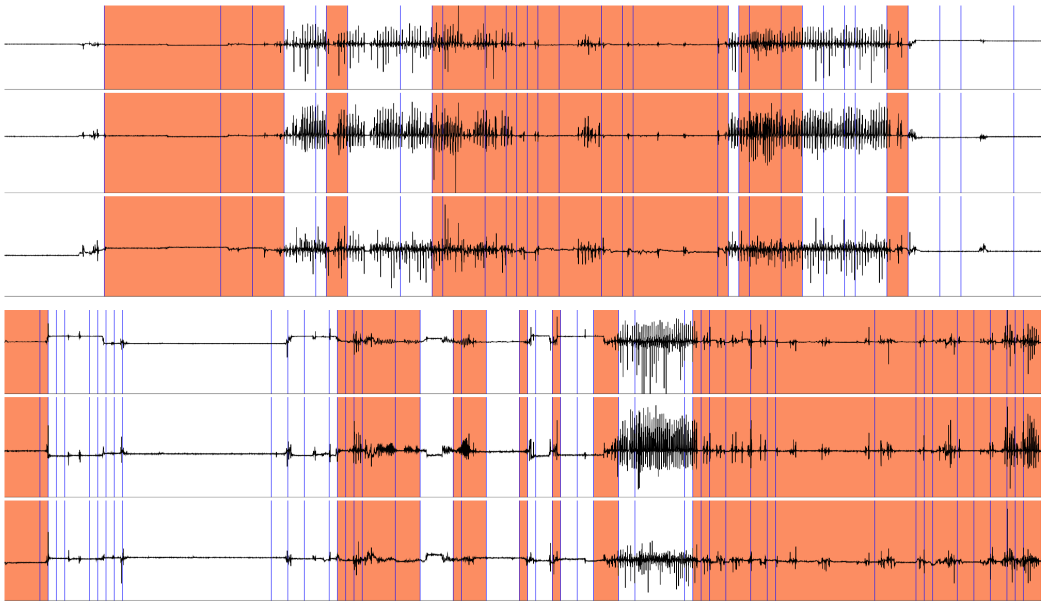


Figure 2.4: Two examples of undesirable clustering of segments.

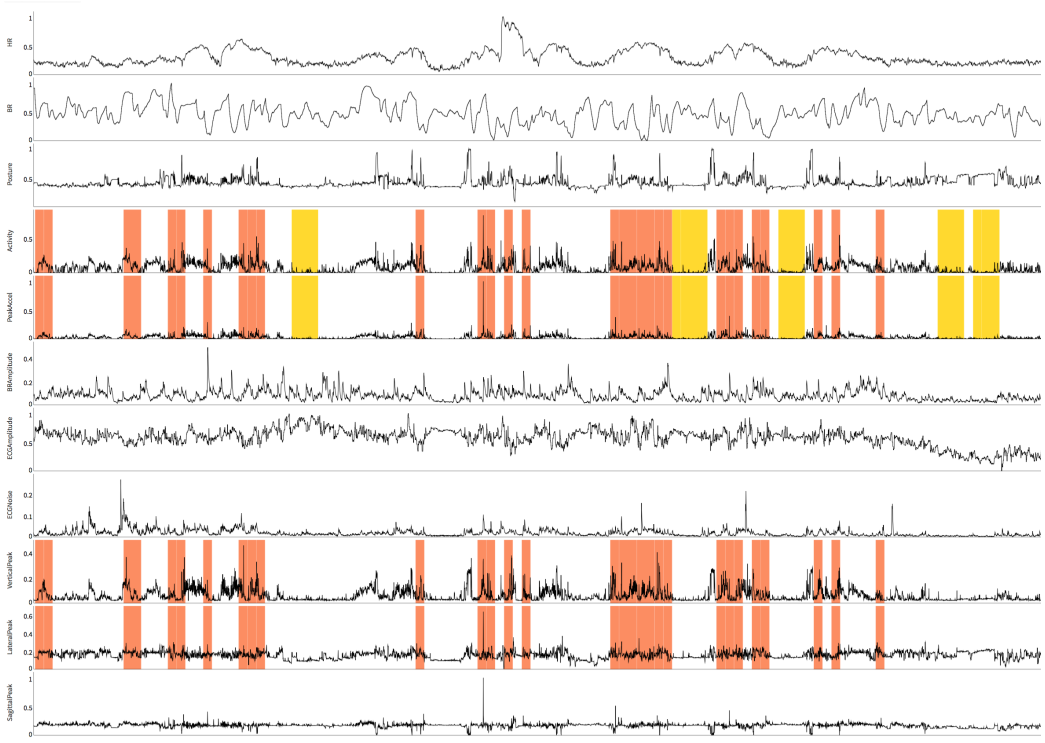


Figure 2.5: Examples of biclusters produced with BiclusTS.

well as acceleration. The dataset covers various sorts of activities common for a day of snowboarding in the high mountains.

Figure 2.5 shows two examples of biclusters related to alternating time periods of rest and downhill snowboard activities. As for the selected variables, one bicluster shows high levels of physical activity and different postures, during the periods of snowboarding. The other bicluster identifies resting periods in between snowboarding. These resting periods are recognized only using the activity levels measured by the sensor system (*Activity* and *PeakAcceleration*). As expected, our biclustering method makes use of different subsets of variables to describe different phenomena.

2.5 Related Work

Biclustering has received a lot of attention in the last decade and has become a well established task [19, 51, 61, 79, 29], with important real-world application, especially in the bioinformatics research field. The most cited biclustering algorithm is the Cheng & Church algorithm [19], making it an obvious benchmark. All these algorithms were not build to consider time series and perform badly due to the shuffling of rows without concern for the temporal nature of time series.

Adaptations for time series have been proposed [108, 62, 29], specifically for gene expression data. These adaptations focus on finding biclusters that consist of one contiguous segment in time, representing similar amounts of concentration for a given period of measurements. This means that these biclusters are mostly shaped-based and not repeatable over time. Adaptations of biclustering for general time series have not been proposed yet.

We believe our work presents novelty when by designing a generic solution to bicluster multivariate time series. The focus of our study is on defining a good optimization criterion for a new problem setting, and studying how to evaluate this evaluation criterion efficiently; developing a search algorithm for this criterion is just one of the components of our study. The difference with standard biclustering is that our data is ordered in time. In our setting, a bicluster is required to cover sufficiently long consecutive stretches of time. Also, our optimization criterion is different: it evaluates similarities in distributions. Moreover, the proposed clustering algorithm (part of a larger approach) is designed specifically to work with segments of time series and to allow the calculation of H(I, J) as described in Definition 1.

Still considering biclustering, one could also consider to carry out a comparison between our heuristic method and a complete enumeration approach. We considered this option, but initial results were such that complete enumeration was not a feasible approach for the datasets of realistic size and complexity. The proposed evaluation criterion has a high computational complexity; evaluating it for an exponential number of biclusters is not feasible within reasonable runtimes. Consider that our largest data has 951 200 time measurements, for 6 attributes. A naive enumeration of all subsets of time measurements is clearly not possible. We would also like to point out that, while exhaustive solutions to biclustering have been studied in the literature, they are not commonly used or accepted. Most common are greedy algorithms, such as the one studied in our work.

The task of motif discovery is similar to the task studied here [20, 71, 103]. In most cases, motif discovery algorithms focus on the univariate setting. Nevertheless the tasks are similar, as both motif discovery and biclustering aim at finding subsequences in a time series with a certain similarity. The difference is that motif discovery methods are mostly shape-based [71, 103]. The most cited approach [20] also uses probabilistic metrics, but requires the discretization of data for efficiency reasons. We propose to make full use of the richness of the time series by using LSDD [86] to compare segments.

A number of studies have considered the multivariate setting [88, 70, 93]. These also propose to find non-overlapping subsequences in a multivariate setting. The difference here is that [88, 93] propose a two-step approach, where the univariate setting is extended to the multivariate setting by grouping motifs using principle component analysis or coincidence rates. As for [70], they propose shape-based motif discovery in the multivariate setting, without considering the independence between variables. With our biclustering approach, we allow the discovery of subsets of both time periods and variables, without discretizing the measurements, thus allowing richer probabilistic descriptions to find more complex patterns.

One step further is to look into the problem of updating projections in a stream of tensors [87]; it incrementally maintains projections in these streams and is not capable of finding patterns that reoccur for the same sensors in different unconnected fragments of time. The optimization criterion that [87] used is a global optimization criterion that favors obtaining an as accurate representation of the data as possible, while instead, we focus on finding local patterns. Arguably, here, our problem setting compares to [87] problem setting as biclustering compares to matrix factorization: the problems are related, but have a different focus that justifies both's existence.

2.6 Conclusions and Future Work

This paper introduced the task of biclustering multivariate time series. In this context, respecting the temporal order is absolutely critical. Given that the traditional biclustering setting assumes that all samples are independent, the shuffling process of finding subsets of rows and columns will lead to useless biclusters in the time series context. We showed the importance of an algorithmic solution to bicluster multivariate time series. First, we proposed the creation of segments of sufficient length; second, we argued for the use of an LSDD divergence score [86] to ensure that for each selected variable, all selected segments have a similar distribution.

We presented an algorithm for solving this biclustering task. It consists of two stages: first, a stage in which the time series are segmented in segments of sufficient length; second, a stage in which a selection of segments and variables is made. In both the first and second stage, we used LSDD with fixed parameters. Experiments showed that LSDD produces stable results, and that the probabilistic descriptions of each segment can be used to accept, or reject a bicluster during the node deletion process.

This paper introduced the task of biclustering multivariate time series. In this context, respecting the temporal order is absolutely critical. Given that the traditional biclustering setting assumes that all samples are independent, the shuffling process of finding subsets of rows and columns will lead to useless biclusters in the time series context. With this paper, we show the importance of an algorithmic adaptation of traditional methods when biclustering multivariate time series.

We proposed two modifications of the traditional biclustering task. First, we proposed the creation of segments of sufficient length; second, we argued for the use of an LSDD divergence score [86] to ensure that for each selected variable, all selected segments have a similar distribution.

We presented an algorithm for solving this biclustering task. In this algorithm, we distinguished two stages: first, a stage in which the time series are segmented in segments of sufficient length; second, a stage in which a selection of segments and variables is made. Both in the first and in the second stage we used LSDD with fixed parameters. Experiments showed that LSDD produces stable results, and that the probabilistic descriptions of each segment can be used to accept, or reject a bicluster during the node deletion process.

As for the problem of setting multiple parameters, although the algorithms have some parameters, several of them are either fixed or estimated automatic. As for the parameters presented in Algorithms 1, 2 and 3, we have decided to fix most of them to the same value in all the experiments, except for window size, w, and jump between consecutive windows, $jump$. All the other parameters are fixed, because the results are not very sensitive to their setting. The default values can be found in the online tool (see Chapter 3). Additionally, we would like to point out that the parameters from Section 2.3.3 (θ and σ) are estimated automatically, and justified experimentally in Section 2.4.1 (LSDD estimation).

As future work, other types of biclustering in time series data can be studied. Comparisons between segments can be based on other descriptions, such as Fourier transforms with scores of spectral similarities, ARMA models coefficients and functional fits, both with scores on standard errors. These representations could be effective at capturing specific phenomena in the data that are not recognized by comparing distributions or other representations. Our algorithm can be improved as well. One could consider the use of multiple node deletion and addition to speed-up the search, and could modify it to allow for biclustering. This would be highly beneficial to both speed up the biclustering procedure and capture overlapping phenomena.

Chapter 3

Bipeline: a Web-based Visualization Tool for Biclustering of Multivariate Time Series

Ricardo Cachucho, Kaihua Liu, Siegfried Nijssen, Arno Knobbe

in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2016

Abstract

Large amounts of multivariate time series data are being generated every day. Understanding this data and finding patterns in it became a contemporary and relevant task. To find prominent patterns present in multivariate time series, one can use biclustering, that is looking for patterns both in subsets of variables that show coherent behavior and in a number of time periods. For this, an experimental tool is needed.
Here, we present Bipeline*, a web-based visualization tool that provides both experts and non-experts with a pipeline for experimenting with multivariate time series biclustering. With* Bipeline*, it is straightforward to save experiments and try different biclustering algorithms, enabling users to intuitively go from pre-processing to visual analysis of biclusters.*

3.1 Introduction

The development of sensing technology lead to an explosion of sensor-based applications, commonly known as the internet of things. Such applications strive mainly due to factors such as, flexibility of design (smaller sensors), lower costs of production (cheaper sensors) and ease in terms of deployment and communication (interactive sensors). In many cases, such sensor systems measure complex phenomena without any sort of supervision, where variables are collected synchronously over time. As a result, there is an explosion of unsupervized multivariate time series.

As a motivating example take the case of a monitoring project for a highway bridge in the Netherlands [52, 102]. In this project, about 150 sensors were deployed on one span of the Hollandsebrug bridge, during an overall refurbish procedure to increase the life time of an important highway bridge. The intent of the project is to develop structural health monitoring methodologies for such a concrete bridge and find key performance indicators (KPIs) that could lead to a predictive maintenance. The fact is that the bridge is exposed to environmental elements (temperature, wind, rain, salt...) and is also subject to multiple events due to the traffic passing on the bridge. All these factors put together, result in a complex phenomena that were measured and materialized as multivariate time series. To discover patterns in such a multivariate and unsupervized setting, one would need sophisticated pattern recognition methods.

Pattern mining of multivariate time series is becoming highly relevant, both in scientific research and industrial applications. There are multiple tasks to deal with pattern mining for time series, such as segmentation and motif discovery. In the case of motif discovery, the task is set to find recurrent patterns over time. The motif discovery solutions normally focuses on the univariate case. Note that in the multivariate setting, not only recurrent patterns in one variable over time are relevant, but also relationships between multiple variables could provide useful insights. This task, is both clustering for time periods and variables, also know as biclustering [19, 61, 15].

Given a multivariate time series, it could be useful to try different biclustering algorithms. Also, one needs to optimize parameters across different steps, such as pre-processing, segmentation and biclustering itself. For each of these steps, there are many parameters to be optimized, leading to a large number of experiments. Furthermore, at each step, visual inspection is highly important for researchers to validate their findings. However, there is a lack of tools for this process.

We propose *Bipeline*, a web-based visualization tool that provides a pipeline for applying biclustering to multivariate time series. This tool is readily accessible to anyone via a web-based interface, allowing them to navigate through multiple experimental settings. Parameters can be interactively tuned, with web components such as checkboxes, sliders and drop-down menus. At each step of the biclustering process, feedback is provided be means of visualizations, with plots such as pre-processed time series, segmentation boundaries and biclusters. One or more biclusters can be plotted with a simple selection procedure.

3.2 Related Work

Until now, biclustering software tools with a graphical user interface have been developed to deal with biological gene expression data. BicOverlapper [82] is a tool for visual inspection of gene expression biclusters, introducing a novel visualization algorithm *Overlapper* to represent biclusters. Similarly, BiCluster Viewer [34] is a visualization tool for efficient and interactive analysis of large gene expression datasets. BicAT [7] implements multiple biclustering algorithms, for visualization and analysis of biclusters for expression data. BiGGEsTS [29] provides an environment for biclustering time series gene expression data.

All tools mentioned above integrate techniques for pre-processing and biclustering analysis, specifically for gene expression data. Their main purpose is to support biologists with the analysis and exploration of the gene expression data. However, these tools do not support biclustering analysis for multivariate time series. Also, most of them do not provide a pipeline experiment environment. Bipeline provides such a pipeline, where intermediate results can be inspected and saved. Using a friendly and interactive plotting environment, both non-experts and experts can pre-process, segment and analyze biclusters for multivariate time series.

Another class of tools are the machine learning experimental tool, where one can compare algorithms and decide on which is the best solution for a particular dataset. Examples of such tools are Weka [32], Moa [10] or KNIME [8]. From the multivariate time series perspective, the setback of such tools is that they are not tailored to experiment on biclustering or not tailored to analyze time series. On the other hand, Bipeline is an environment where compare multiple traditional biclustering algorithms and compare them to the biclustering algorithm proposed in the previous chapter.

3.3 Tool Overview

Bipeline is a web-based application that provides a pipeline to pre-process, segment and bicluster multivariate time series. An online version is available[1], which is compatible with all modern web browsers and across different client platforms. Both the user interface in the web browser and the server are implemented using R Shiny package [17]. In Figure 3.1, the system architecture illustrates the experimental pipeline and how each individual step relates to the other steps:

Importing Users can upload datasets and have a first view of the data table and descriptive statistics (*minimum*, *maximum*, *mean*, . . .). This first inspection, although useful, is not enough to assess the quality of the data.

Plotting To gain further insight into the time series, it is crucial to have a visual inspection of the time series. The plotting panel includes multiple interactive plotting views, using a plotting R package dygraphs [97]. An example of these plots is illustrated in Figure 3.2. These interactive plots allow zoom in and out functionality, which is a highly desirable functionality for visual inspection of large time series.

This process of visual inspection is important across multiple phases of the biclustering process. This need for plotting is specially important in tasks that need human evaluation. Considering that there are multiple steps involved (pre-processing, segmentation and biclustering), Bipeline gives the user the flexibility to vizualize the data multiple times across different stages.

[1]http://fr.liacs.nl:7000

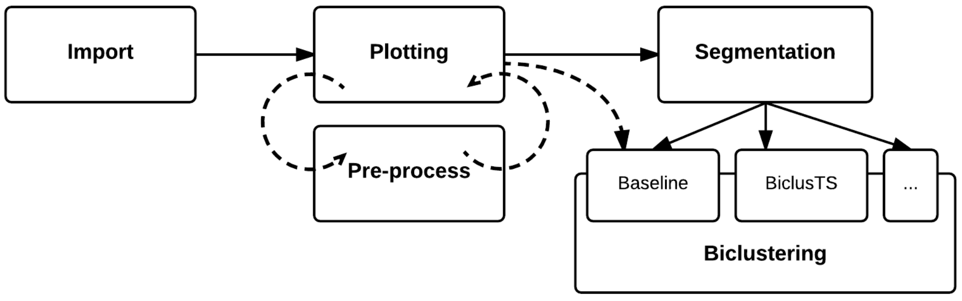
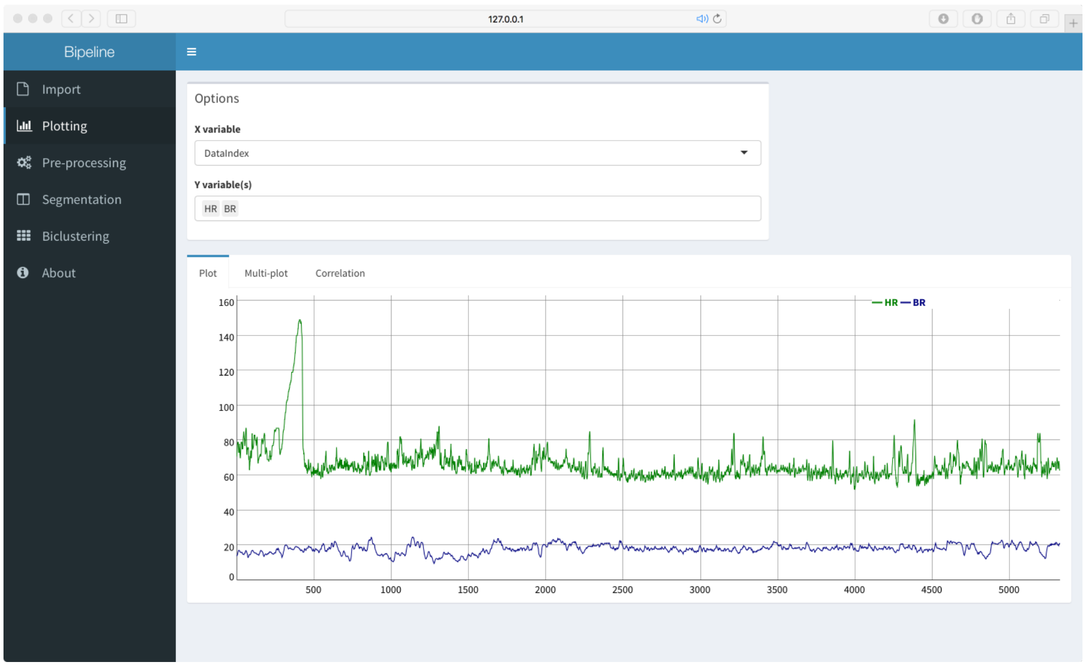


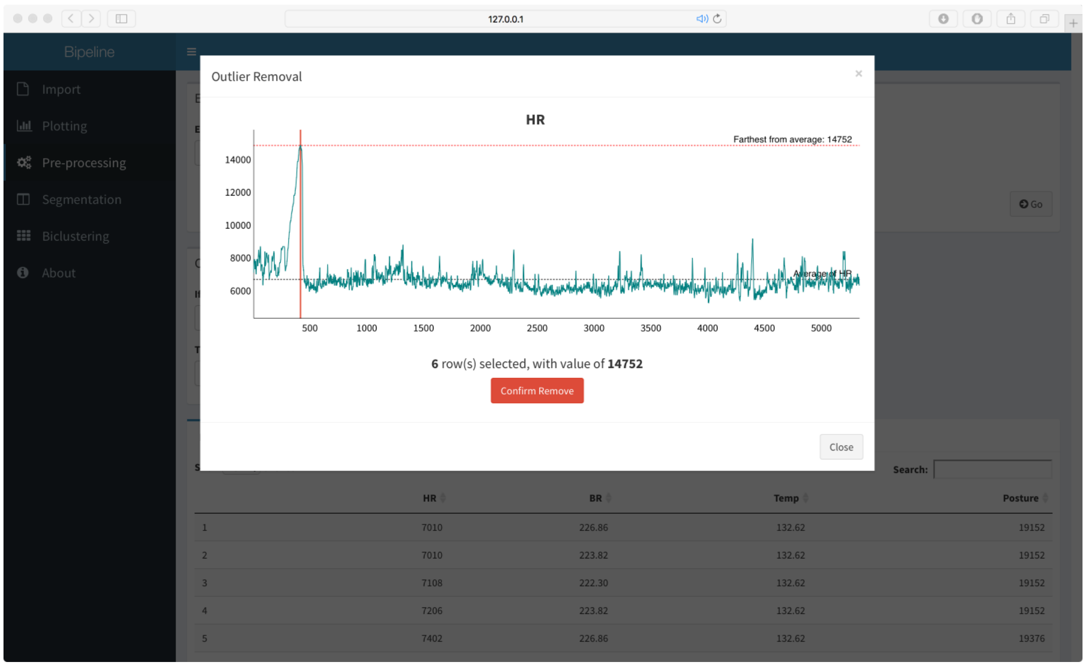
Figure 3.1: A overview of *Bipeline* architecture.



X variable: DataIndex

Y variable(s): HR BR

Figure 3.2: *Bipeline* user interface: Plotting menu.



6 row(s) selected, with value of **14752**

	HR	BR	Temp	Posture
1	7010	226.86	132.62	19152
2	7010	223.82	132.62	19152
3	7108	222.30	132.62	19152
4	7206	223.82	132.62	19152
5	7402	226.86	132.62	19376

Figure 3.3: *Bipeline* user interface: Pre-processing menu.

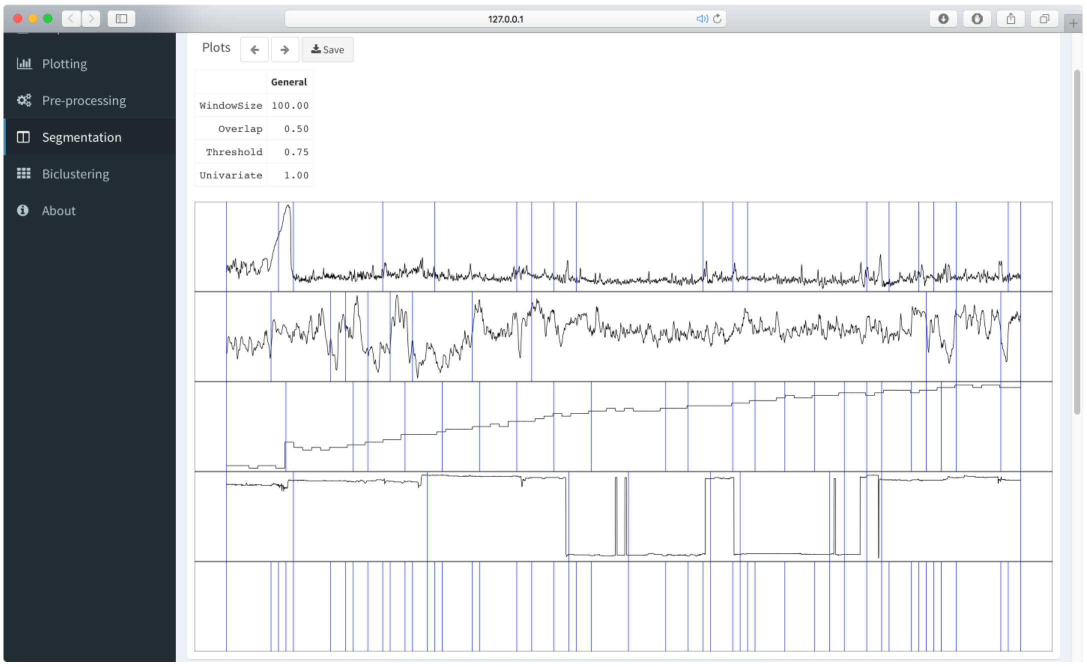
Pre-processing This panel allows preliminary handling of data such as: excluding variables, normalization, conditional removal and replacement of data, and outlier removal. Users can alternate between *plotting* (Figure 3.2), and *pre-processing* (Figure 3.3 until satisfied, then export the pre-processed data by clicking the *Save* button.

Segmentation This allows segmentation of the data, one of the steps necessary for the biclustering as suggested by the method presented in the previous chapter. By default, all variables share the same parameter settings: *window size*, *overlap* and *threshold* can be easily tuned. For greater flexibility, the user can dynamically create new tabs to set the parameters for individual variables. Additionally, a *minimum segment size* is customizable, and the tool will merge short segments to its most similar contiguous segment. Segmentation results can be visualized (Figure 3.4, saved and (re-)loaded, allowing the results to be used during the next step (biclustering) and over multiple user sessions.

Biclustering In *Bipeline*, we implement a number of biclustering algorithms, grouped in three categories. The *baseline algorithms* allow users to try well-known biclustering algorithms (e.g., Cheng & Church) [19, 61], that have been implemented using R package biclust [44]. *Segmentation + Baseline* biclusters the time series using an average representation of each segment, instead of using individual rows. *Segmentation + BiclusTS* is a novel algorithm [15] introduced to recognize similarities between segments, using probability density-difference estimation [86]. All biclusters are plotted in colored blocks, as shown in Fig.3.5. Users can select the biclusters they want to see, and the plot will respond with a real-time update.

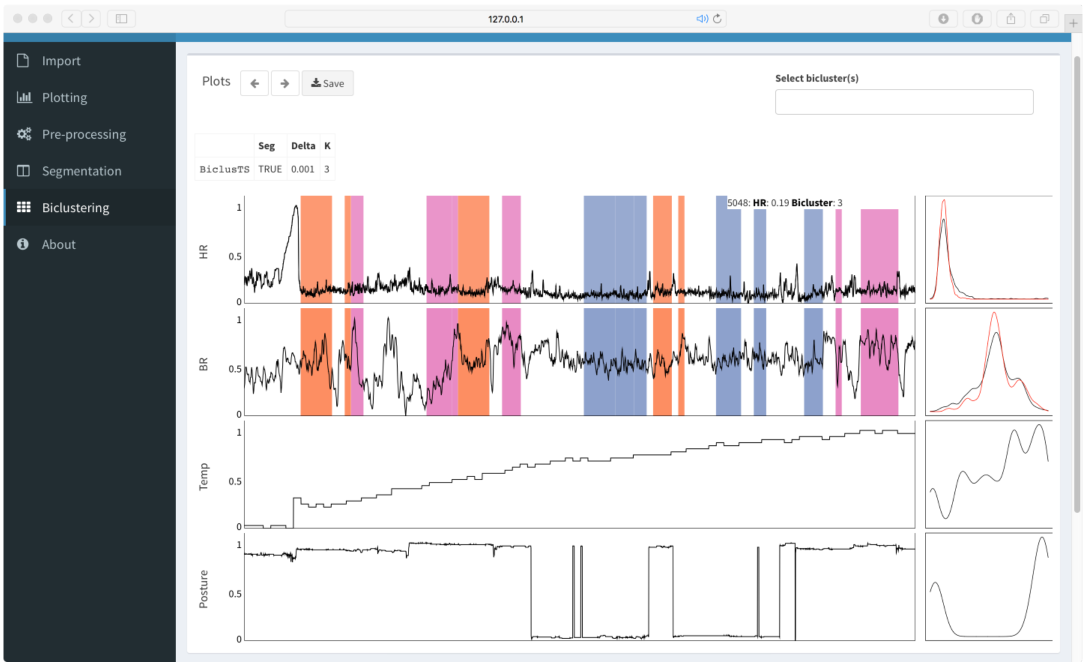
The consideration of multiple biclustering algorithms in the tool, give the possibility to experiment different settings and interpret the different results visually. These different experimental options, place *Bipeline* in the group of tools that can be used for extensive biclustering experiments.

To allow extensive experiments, multiple features are shared by both *Segmentation* and *Biclustering.* Plots and parameter tables from different experiments are kept in history, allowing users to navigate back and forth to compare results and optimize parameters. During computationally expensive tasks, the front-end displays a progress bar, while the back-end server is busy carrying out the calculations. Furthermore, interactive web components can be saved into images with a single click.



	General
WindowSize	100.00
Overlap	0.50
Threshold	0.75
Univariate	1.00

Figure 3.4: *Bipeline* user interface: Segmentation menu.



	Seg	Delta	K
BiclusTS	TRUE	0.001	3

Figure 3.5: *Bipeline* user interface: Biclustering menu.

3.4 Conclusion

In this chapter we propose *Bipeline*, a web-based visualization tool, which provides a pipeline for applying biclustering to multivariate time series. Its main features include: visual inspection at multiple stages, interactive zoom in and out plotting, easy navigation, storage of results, and saving plots and experimental settings using a single click.

Bipeline's intuitive web-based design, makes it accessible both to experts and non-experts, and compatible across platforms. From a user perspective, the implementation of this tool can be found online[2]. Additionally to the online tool, for the users with a computer science background, both the biclusTS algorithm (see Chapter 2) and the tool implementation are open sourced and freely available[3].

[2]http://fr.liacs.nl:7000
[3]https://github.com/kainliu/Bipeline

Chapter 4

Mining Multivariate Time Series with Mixed Sampling Rates

Ricardo Cachucho, Marvin Meeng, Ugo Vespier, Siegfried Nijssen, Arno Knobbe

in Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp), 2014

Abstract

Fitting sensors to humans and physical structures is becoming more and more common. These developments provide many opportunities for ubiquitous computing, as well as challenges for analyzing the resulting sensor data. From these challenges, an underappreciated problem arises: modeling multivariate time series with mixed sampling rates*. Although mentioned in several application papers using sensor systems, this problem has been left almost unexplored, often hidden in a pre-processing step or solved manually as a one-pass procedure (feature extraction/construction). This leaves an opportunity to formalize and develop methods that address mixed sampling rates in an automatic fashion.*
We approach the problem of dealing with multiple sampling rates from an aggregation perspective. We propose Accordion, a new embedded method that constructs and selects aggregate features iteratively, in a memory-conscious fashion. Our algorithms works on both classification and regression problems. We describe three experiments on real-world time series datasets.

4.1 Introduction

This paper presents a practical modeling task in the field of multivariate time series analysis, and algorithms to solve this task. In real-world applications involving time series, specifically those produced by multiple sensors, one is often confronted with the challenge of analyzing data captured at various sampling rates. This might occur when one wants to include sensors that measure processes at various rates, for example the vibration (high sampling rate) and temperature (low rate) of a large windmill. In this paper, we analyze a specific instantiation of such a problem, where the aim is to model a target time series, that is captured at (much) lower rates than the remaining series. To model the target series in terms of the remaining ones, we will somehow have to 'slow down' the high-frequency measurements in order to match the target.

As a motivating example, consider the problem of *activity recognition*. In this problem, the task is to classify a person's activities into a finite set of classes, typically at a fairly slow rate. The activity will be predicted using body-worn or environmental sensors, for example measuring physiological parameters (e.g. heart rate), acceleration or position in space. The practical problem here is that modern sensors tend to measure at high sampling rates (typically 1 Hz or higher), whereas activity is registered at much lower rates (e.g. once every 60 seconds). Therefore, each period to be classified is described by many measurements (per sensor). The most obvious solution is to combine multiple measurements into a single value characterising the period, for example, the average heart rate or the highest acceleration experienced over this period.

In this paper, we approach this challenge from an aggregation perspective: for a given sensor and a given time interval (a window of data), an aggregate function will summarize a sequence of sensor readings into a single numeric value. An aggregate feature is composed of three main components: a sensor measuring at high rates (the *predictor*), a window over which values of the sensor are aggregated, and finally an *aggregate function* that combines these values into a single outcome. Assuming a target series sampled at low frequency $f_\mathbf{r}$ and the remaining time series at higher frequency $f_\mathbf{S}$, the proportion $f_\mathbf{S}/f_\mathbf{r}$ denotes the number of measurements per sensor that correspond to a single target value. To allow for phenomena that involve unknown degrees of integration over time, our algorithm will be allowed to consider windows both longer and shorter than $f_\mathbf{S}/f_\mathbf{r}$.

To illustrate how the optimal window size may be somewhat larger or smaller than the proportion $f_{\mathbf{S}}/f_{\mathbf{r}}$, consider the challenge of modeling a person's sleep quality (one of our applications mentioned in the experimental section) from various sensors, both body-worn and placed in the environment. In this case, 24 hours of sensor data naturally correspond to a single target value describing the sleep quality of one night. However, one can imagine that a feature capturing the amount of strenuous activity during the four hours prior to sleep might play an important role, which corresponds to a window size of only 1/6th of the 'natural' window. Similarly, windows covering more than 24 hours are imaginable, such as those related to the nutrition over the last 48 hours. Clearly, a fixed window size based on the mentioned proportion will not guarantee optimal results, and we will have to include and optimize the size of the window as a parameter in the definition of features.

Although a feature construction step (even including aggregation) is the backbone of many activity recognition projects [25], all too often this step is presented as a one-pass process [6, 69, 58, 75], such that only a fixed set of features becomes available for the actual modeling step. The resulting features are static and constructed manually, either based on some domain knowledge about the physics involved, or by making default choices. It is not hard to imagine that this step is, in fact, the result of several iterations of trial and error. Moreover, this fixed set is required to be relatively small, for reasons of memory or storage. The iterative method we propose, Accordion, is an embedded approach that does both feature construction (building candidate aggregate features), and feature selection automatically, allowing for features to be created dynamically during the search process.

At each iteration in the search process, Accordion transforms high frequency predictors into a set of candidate aggregate features at the lower frequency of the target, searching for the best combination of the components that compose an aggregate feature. From this set of aggregate features, only the most promising candidate feature is selected and materialized, in a greedy fashion. Therefore we categorize our algorithms as *memory-conscious*. With the dynamic construction of features proposed here, we aim to solve both the issue of choosing the right features and estimating their parameters, as well as the varying requirements for the informative features that occur while modeling the data (for example, further down in a decision tree). In order to do so, the feature construction and selection steps are closely tied with the final modeling process, in both the regression and classification setting. Inspired by Brush's challenge of enhancing reproducibility and clarity [12], our algorithms and activity recognition datasets are made available[1].

When thinking of the aggregate feature construction possibilities, it is good to note that the search space is potentially very large, due to the choices of sensor, aggregate function and window size (which may vary substantially, as noted). Therefore, we search for (candidate) aggregate features heuristically. We propose algorithms that consider a feasible set of candidate features by a) limiting the actual choice of aggregate functions to a small set (*min*, *max*, *avg*, . . .), b) performing a moderate search over the possible window sizes, and c) selecting the final aggregate features at different degrees of greediness. On top of these choices, we tackle the potentially large size of the final dataset by materializing only the selected features.

In general, we distinguish between two types of applications, one where the slow target series is numeric, and we are effectively dealing with a *regression* model, and one where the target is nominal and we need a *classification* model. The feature selection algorithm works differently for either setting, but the essence of constructing sets of candidate features using aggregation is identical. The main difference between the two versions is the kind of modeling they mimic: in the regression case, the feature construction algorithm effectively builds a linear model in a greedy fashion. In the case of classification, we construct a decision tree of aggregate features along the lines of C4.5 [80].

The main contributions of this paper are as follows:

- Present and formalize a common task in the modeling of multivariate time series, related to the target being measured at a lower rate than the remaining series.
- Propose an embedded algorithm for the proposed task, that dynamically constructs, selects and models (all in one solution), using a manageable set of aggregate features in the contexts of both classification and regression.
- Describe how both algorithms are memory-conscious, materializing only a limited set of aggregate features, and potentially increasing the possible search space for good features.
- Algorithm implementation and datasets are made available to the research community.

[1]`http://www.liacs.nl/~cachucho/publications.html`

4.2 Preliminaries

4.2.1 Multivariate Time Series with Mixed Sampling Rates

The data we consider is assumed to come from *sensor systems*. We assume that our sensor system $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_p, \mathbf{r}\}$ consists of $p+1$ sensors. The first p sensors will act as *predictors*, while the last sensor $\mathbf{r}$, the *response*, will be treated as the target sensor that we wish to predict or explain. $|\mathbf{s}|$ and $|\mathbf{r}|$ indicate the length (number of data points) of $\mathbf{s}$ and $\mathbf{r}$, respectively. While the domain for the predictors is always the set of real numbers $\mathbb{R}$, the domain of $\mathbf{r}$ is either $\mathbb{R}$ (regression setting), or a finite set of classes (classification setting).

We assume that all sensors register measurements synchronously and at the same fixed sampling rate, *except* for the response, which is registering at a lower sampling rate. We also assume that the predictor sampling rate is an integer $q > 1$ multiple of the sampling rate of the response: $f_{\mathbf{S}} = q \cdot f_{\mathbf{r}}$. This leads to the following definition.

Definition 6. *A time series dataset with mixed sampling rates is assumed to consist of:*

- *A set of time series* $\mathbf{S}$*, representing the predictor variables, where* $\mathbf{S}$ *is materialized as a matrix of size* $|\mathbf{s}| \times p$*. Each time series* $\mathbf{s}$ *in* $\mathbf{S}$ *is a vector of real numbers, where* $\mathbf{s}_i, i = 1, \ldots, |\mathbf{s}|$ *is the ith element of* $\mathbf{s}$*.*

- *A time series* $\mathbf{r}$*, representing the response variable. This time series has a length of* $|\mathbf{r}| = |\mathbf{s}|/q$*, where* $q \in \mathbb{N}^+ \wedge q > 1$*; the ith element in* $\mathbf{r}$ *is assumed to have been measured at the same time as the* $i \cdot q$*th element in* $\mathbf{s}$*.*

Note that this implies that the measurements of $\mathbf{S}$ and $\mathbf{r}$ do not start at the same time (see Figure 4.1).

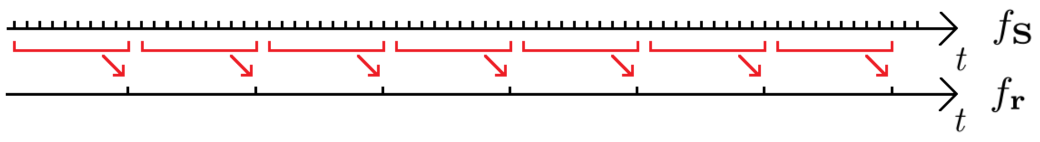


Figure 4.1: Relation between high ($f_{\mathbf{S}}$) and low ($f_{\mathbf{r}}$) sampling rates.

4.2.2 Feature Construction with Mixed Sampling Rates

As discussed, our aim is to model the *response variable* in terms of the *predictors*, over time. As the two are sampled at different rates, we will 'slow down' the high frequency measurements of $\mathbf{S}$ using aggregate functions, and turn them into features that are available at the sampling rate of $\mathbf{r}$. In other words, we will be taking a *feature construction* approach. In order to transform the high frequency measurements into lower frequency ones, we employ the notion of a *window*:

Definition 7 (Window)**.** *Given a window length w and an index $1 \leq w \leq i \leq |\mathbf{s}|$ in a predictor time series $\mathbf{s}$, a window of length w at index i consists of the time series of measurements $\mathbf{s}[i-w+1,\dots,i] = \mathbf{s}_{i-w+1}, \mathbf{s}_{i-w+2}, \dots, \mathbf{s}_i$.*

For a response series measured q times as slow as the predictors, it could make sense to choose w such that $w = q$. Figure 4.1 depicts such a situation. However, experimental evaluation reveiled that this choice may not always be optimal. Both window lengths $w > q$ and $w < q$ could also be argued for. Therefore, we simply assume that $w \in \mathbb{N}^+$. Note that when $w > q$, each consecutive window will have the following fraction of overlap: $1 - q/w$.

We will employ *aggregate functions* to summarize the measurements in a window into a single value. An aggregate function $a \in A$ takes as input a time series of measurements $\mathbf{m}$, and produces a single numeric value $a(\mathbf{m}) \in \mathbb{R}$. The fixed set of aggregate functions $A = \{min, max, avg, \dots\}$ we use will be described in more detail in the next section.

We can now define aggregate features as follows.

Definition 8 (Aggregate Feature)**.** *Given a choice of window size w, an aggregate function a and a predictor time series $\mathbf{s} \in \mathbf{S}$, the aggregate feature $\boldsymbol{af}_{\mathbf{s},a,w}$ is a vector of length $|\mathbf{s}|/q$, defined as follows:*

$$\boldsymbol{af}_{\mathbf{s},a,w}[i'] = a(\mathbf{s}[\max(1, i' \cdot q - w + 1), \dots,' \cdot q])$$

where $1 \leq i' \leq |\mathbf{s}|/q$.

An aggregate feature for a given dataset can hence be specified by a tuple of parameters $(\mathbf{s}, a, w)$. Sometimes we will refer to these features without reference to their parameters, just as a generic aggregate feature $\mathbf{f}$.

A set of aggregate features $\mathcal{F}$ together with the vector of response values $\mathbf{r}$ can be used to create a new data matrix. Each aggregate feature corresponds to a column of this matrix. The number of rows in this matrix corresponds exactly to the length of $\mathbf{r}$, $|\mathbf{r}| = |\mathbf{s}|/q$.

More formally, a data matrix $\mathbf{S}'$ of dimension $|\mathbf{r}| \times (|\mathcal{F}| + \mathbf{1})$ is obtained, where $\mathbf{S}'_{t,f}$ is the feature calculated for the tth target instance using the fth feature descriptor in $\mathcal{F}$.

4.2.3 Problem Statement

Our main task is to find good aggregate features for time series datasets with mixed sampling rates. More formally, we assume we are given a time series dataset as introduced in Definition 6, as well as a function $score(\mathcal{F}, \mathbf{r})$ that can evaluate the quality of a set of features with respect to response variable $\mathbf{r}$. The task is to find a set of aggregate features $\mathcal{F}$, such that each feature is described by:

- A predictor time series $\mathbf{s} \in \mathbf{S}$.
- An aggregate function $a \in A$.
- A window size w.

Furthermore, the feature set $\mathcal{F}$ should optimize the scoring function $score(\mathcal{F}, \mathbf{r})$. Scoring functions in this paper can be based on regression or classification models. The details of this will be discussed in the next section.

4.3 The Accordion method

4.3.1 Aggregation of Time Series

Aggregate functions provide a means for summarizing a series of measurements in a window, in various ways, as illustrated in Figure 4.2. Different aggregate functions capture different aspects of the measurements within a window. Although the space of aggregate functions is conceivably very large, we have opted for a relatively small collection of functions that represent common statistics of sets of values. The set of aggregate functions, A, considered in this paper is composed of:

- *avg*: the mean value,
- *med*: the median,
- *max*: the maximum value,
- *min*: the minimum value,

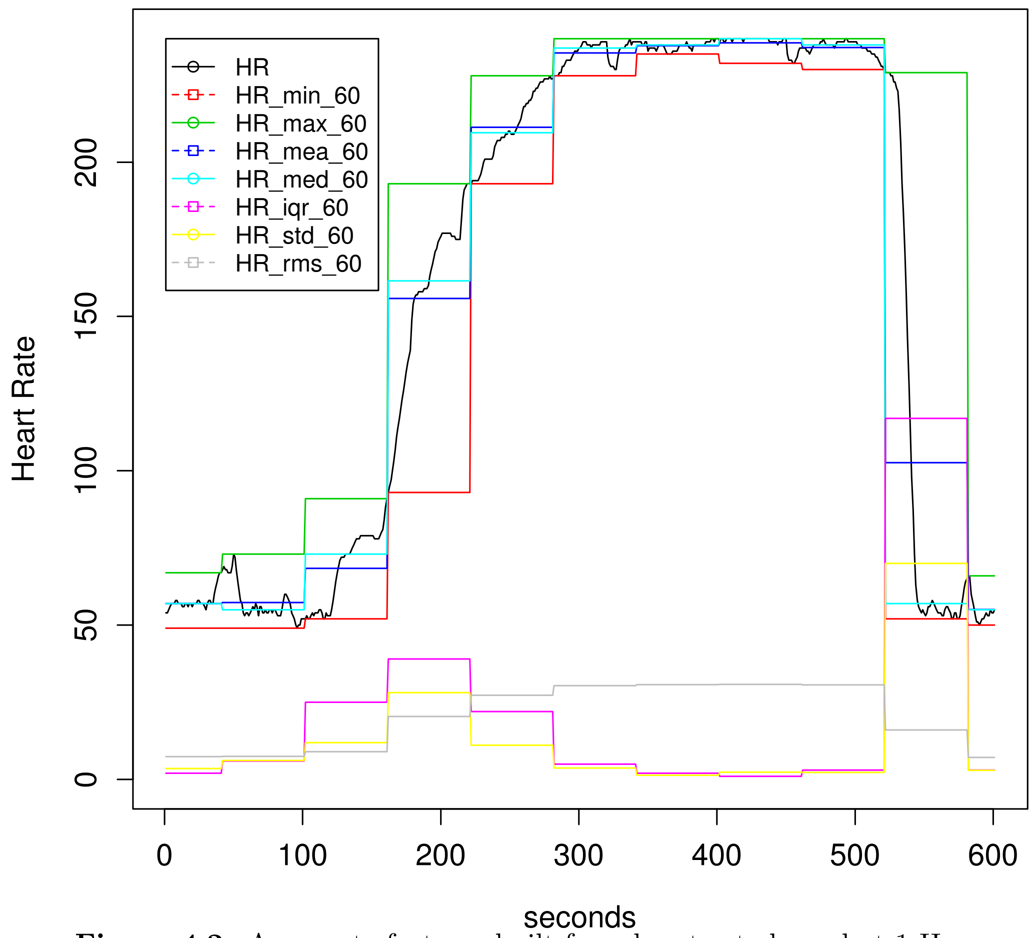


Figure 4.2: Aggregate features built from heart rate logged at 1 Hz.

- *stdv*: the standard deviation,
- the inter-quartile range: $IQR = inf\{x \in \mathbb{R} : 0.75 \leq P(X \leq x)\} - inf\{x \in \mathbb{R} : 0.25 \leq P(X \leq x)\}$,
- the root mean squared: $RMS = \sqrt{\frac{1}{n}\sum\limits_{i=1}^{n} x_i^2}$.

One could argue that features of windows from the *frequency domain*, such as properties of the spectrum of the data, could also be interpreted as aggregate functions: they take a set of measurements (in fact, a sequence), and summarize them into a single value. Such features would capture more periodic properties of the data in the window. Because our algorithms are open source we argue the set of aggregate functions, A, should be seen as mutable. Domain knowledge could be used to change it. If inclined to do so, one can change (add and remove) the aggregate functions that compose A, in order to capture cyclical aspects or peculiarities of the data. The remainder of this paper assumes always the same rather small list of statistical aggregate functions, but there are no technical reasons why other aggregate functions could not be involved also.

Algorithm 4 CalculateAF

```
Input: time series $\mathbf{s}$, aggregate function $a \in A$, window size $w$, ratio $q = f_{\mathbf{s}}/f_{\mathbf{r}}$.
  for $i' \in \{1, \ldots, |\mathbf{r}|\}$ do
    $i = i' \cdot q$
    if $i < w$ then
      $\mathbf{af}_{\mathbf{s},a,w}[i'] \leftarrow a(\mathbf{s}[1, \ldots, i])$
    else
      $\mathbf{af}_{\mathbf{s},a,w}[i'] \leftarrow a(\mathbf{s}[i - w + 1, \ldots, i])$
    end if
  end for
  return an aggregated feature: $\mathbf{af}_{\mathbf{s},a,w}$
```

An aggregate feature results from the use of an aggregate function a, applied to a *predictor* $\mathbf{s}$ using a sliding window with length w, as formally described in Algorithm 4. To construct an aggregate feature, Algorithm 4 slides a window over the predictor using the reference indices $i \in \{q, 2q, \ldots, |\mathbf{r}| \cdot q\}$. For each reference index i, the algorithm checks for boundary limitations.

If i is smaller than the window size w, the aggregate feature's i'th instance takes into account only the available data from the predictor, $\mathbf{s}[1, \ldots, i]$. Otherwise, the window is $\mathbf{s}[i - w + 1, \ldots, i]$. The aggregate function is then applied to the data in the window.

4.3.2 Feature Construction

This section describes the construction process of multiple aggregate features, as a search problem for the optimal combinations of aggregate function a, high frequency predictor $\mathbf{s}$, and window size w. The objective of the feature construction process is both to slow down the sampling frequency of one or more predictors, and to transform these into good aggregate features. A good aggregate feature should properly describe a target variable ($\mathbf{r}$) at its low sampling rate, $f_{\mathbf{r}}$.

In order to avoid brute force feature construction, and direct the search towards an optimal choice of $(\mathbf{s}, a, w)$, a ranking measure ranking the feature candidates is required. The ranking measures are obtained by the use of scoring functions, $SC(\mathbf{r}, \mathbf{af}_{\mathbf{s},a,w})$. To deal with classification problems, we considered the well-known entropy-based scoring function *information gain* (see for example [80]). For regression problems, the *cross-correlation* [63] scoring function is selected:

$$\gamma_{\mathbf{r},\mathbf{f}}(\tau) = E[(\mathbf{r}_i - \mu_{\mathbf{r}}) \cdot (\mathbf{f}_{i-\tau} - \mu_{\mathbf{f}})],$$

where τ is the time lag between an aggregate feature $\mathbf{f}$ and a target variable $\mathbf{r}$. In the presence of a delayed relation between action ($\mathbf{f}$) and reaction ($\mathbf{r}$), cross-correlation allows the identification and construction lag regression models [31].

The process of feature construction is detailed in Algorithm 5. This algorithm performs a grid search over the available predictor time series in $\mathbf{S}$ and the aggregate functions in A (the two outer loops). The number of different values for both these parameters of an aggregate feature is generally limited, so all combinations will be considered exhaustively.

For each choice of $\mathbf{s}$ and a, the algorithm returns the best window size w_{best}. In essence, this is a task of linear optimization of an unknown function for a given parameter w. In order to avoid a simple exhaustive search for the optimal window size, we sample this function iteratively, and zoom in on a promising interval $[w_l, w_h]$ at each iteration. This heuristic optimization algorithm works as follows.

Algorithm 5 ConstructCandidates

```
Input: set of predictor time series $\mathbf{S}$, target variable $\mathbf{t}$, scoring function $SC$,
  decision $threshold$, maximum window growth $\omega$, number of steps $m$.
  $\mathcal{C} \leftarrow \varnothing$
  $q \leftarrow |\mathbf{S}|/|\mathbf{r}|$
  for all $\mathbf{s} \in \mathbf{S}$ do
    for all $a \in A$ do
      $\lambda \leftarrow 1$
      $w_{best} \leftarrow 0, score_{best} \leftarrow 0$
      $w_l \leftarrow q, w_h \leftarrow q \cdot \omega$
      $stop \leftarrow$ false
      repeat
        $\delta \leftarrow \frac{w_h - w_l}{m}$
        $w \leftarrow 0, score \leftarrow 0$
        for all $i \in \{1, \ldots, m\}$ do
          $\mathbf{f} \leftarrow$ CalculateAF$(\mathbf{s}, a, w_l + i \cdot \delta, q)$
          if $SC(\mathbf{f}, \mathbf{t}) \geq score$ then
            $w \leftarrow w_l + i \cdot \delta$
            $score \leftarrow SC(\mathbf{f}, \mathbf{t})$
          end if
        end for
        if $score_{best} > score$ then
          $stop \leftarrow$ true
        else
          $score_{best} \leftarrow score, w_{best} \leftarrow w$
        end if
        $w_l \leftarrow w_{best} - q/\lambda, w_h \leftarrow w_{best} + q/\lambda$
        $\lambda \leftarrow \lambda + 1$
      until $stop$
      if $score_{best} > threshold$ then
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{af}_{\mathbf{s},a,w_{best}} \leftarrow$CalculateAF$(\mathbf{s}, a, w_{best}, q)\}$
      end if
    end for
  end for
  return $\mathcal{C}$
```

The interval is initialized based on a parameter ω which indicates the largest acceptable window growth relative to q (the natural window size). At each iteration of the repeat-until loop, a uniform sample is made of the current interval, in m steps (specified by the user, but typically low). Each candidate window size is used to compute a candidate aggregate function $f =$ CalculateAF$(\mathbf{s}, a, w_l + i \cdot \delta, q)$ and its associated *score*. By the end of each iteration, the current best window size w_{best} is known, and the interval of inspected sizes is reduced to $[w_{best} - q/\lambda, w_{best} + q/\lambda]$, narrowing constantly the original interval through the iterations, around the current w_{best}. This repeated zooming in on the best window size is continued until a more detailed inspection does not yield a better result.

Both Algorithm 4 and 5 are built under the assumption that both target and predictors are sampled at constant sampling rates, but this assumption can be broken in multiple ways. First consider the situation of a target measured at an unstable sampling rate. To overcome this limitation we would need to recalculate q by replacing $|\mathbf{S}|$ with the median sampling rate of $\mathbf{S}$, and recalculate the reference indexes i to synchronize both target and predictors. Secondly the predictors could be measured at an unstable sampling rate. This would require changing the algorithms from index-based windows into time-based windows. The indexes i' and reference indexes i in Algorithm 4 need to be referenced using timestamps, and the window size w needs to be expressed in time units.

Although exhaustive search over window sizes guarantees finding the absolute maximum, the computational costs of this approach would be unacceptable. For this reason, our algorithm employs heuristic optimization to find the optimal window size efficiently. As the score of aggregate functions is generally well-behaved, this heuristic algorithm will typically find the global optimum, rather than a local one. As an example, Figure 4.3 shows an exhaustive scoring landscape for different window sizes, given a low frequency (days) numeric target, a high frequency predictor (1 Hz) and an aggregate function (RMS). The vertical line represents the best window size w_{best}, as determined by Algorithm 5 in a fraction of the time taken by the exhaustive search. Note how this graph also provides a good example of how the optimal window size may differ substantially from the naive choice represented here by 1 on the horizontal scale.

Each combination of $\mathbf{s}$ and a results in a single w_{best}. The associated aggregate feature is then added to the result set of candidate features $\mathcal{C}$, under the condition that its score is higher than a certain *threshold*.

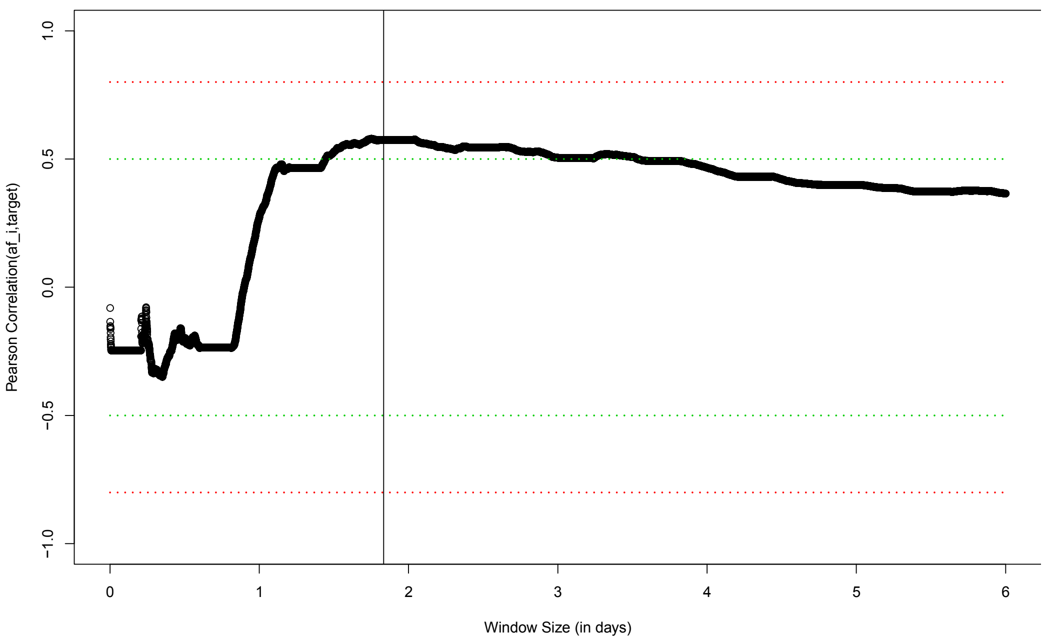


Figure 4.3: Landscape of scoring measure for one of the regression datasets. The horizontal scale indicates multiples of q (in this case 24 hours), the vertical scale the correlation with the target. Note how the optimal window size is almost twice (44 hours) that of the naive choice of $w = q$.

Throughout our experiments, the *threshold* value was set to 0.5 for information gain, and 0.2 for cross-correlation. For ease of implementation, our algorithms will always assume the number of steps m to be equal to the maximum window growth divided into q steps, ω/q, thus effectively removing one parameter.

An important characteristic of our algorithm is that it is memory-conscious. This sets it apart from other feature construction methods [6, 25], because it does not simultaneously materialize most of the inspected features. Keeping only the scores $SC(\mathbf{t}, \mathbf{af}_{\mathbf{s},a,w})$ gives us concise information about how good an aggregate feature could perform, relatively to a target variable $\mathbf{t}$. Every time Algorithm 5 is called, the end result is a collection of at most $p \cdot |A|$ candidate features, which is an acceptable number in most cases.

4.3.3 Feature Selection: Embedded approach

In order to increase the chances of finding dependencies between the aggregate features and the response, we developed an embedded feature selection

Algorithm 6 Aggregate Features Selection: Regression

```
Input: set of predictor time series $\mathbf{S}$, response variable $\mathbf{r}$, scoring function
  $SC$, maximum window growth $\omega$, number of steps $m$.
  $\mathcal{F} \leftarrow \varnothing$
  $\mathbf{t} \leftarrow \mathbf{r}$
  while $\neg whiteNoise(\mathbf{t})$ do
    $\mathcal{C} \leftarrow$ConstructCandidates$(\mathbf{S}, \mathbf{t}, SC, \omega, m)$
    if $\mathcal{C} = \varnothing$ then
      return $\mathcal{F}$
    end if
    $\mathbf{f}_{best} \leftarrow \arg\max_{\mathbf{f} \in \mathcal{C}} SC(f, \mathbf{t})$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{f}_{best}\}$
    $l \leftarrow fitLM(\mathcal{F}, \mathbf{r})$
    $\mathbf{t} \leftarrow \mathbf{r} - l(\mathcal{F})$
  end while
  return $\mathcal{F}$
```

method, such that at each iteration of the final modeling algorithm, we do not work with a static set of features. At each iteration, Accordion performs a new feature construction step, and searches for the best aggregate feature.

Regression problems

As the search space of candidate aggregate features can grow too large to explore exhaustively, we resort to a heuristic search that only constructs a subset of promising candidate features, $\mathcal{C}$. When the set $\mathcal{C}$ is large (> 40), a wrapper-based search for the optimal subset becomes impractical [33]. As for backward-stepwise selection in linear models, if the number of candidate features is larger than the number of instances, the use of the least squares method for coefficient estimation becomes impossible [84]. To overcome these potential problems, we employed a *forward-stepwise selection* process [39]. As described in Algorithm 6, at each iteration, we add a new aggregate feature creating a nested sequence of models, until one of the following stopping criteria is satisfied:

- The set of candidate aggregate features $\mathcal{C}$ returned by Algorithm 5 is empty.
- The decomposed target variable is considered white noise.

Algorithm 7 BuildAFTree

```
Input: set of predictor time series $\mathbf{S}$, nominal response variable $\mathbf{r}$, maximum
  window growth $\omega$, number of steps $m$, minimum leaf size $minsup$.
  $\mathcal{C} \leftarrow$ ConstructCandidates($\mathbf{S}, \mathbf{r}, IG, \omega, m$)
  if $\mathcal{C} = \varnothing$ or $|\mathbf{r}| < minsup$ then
    return $\varnothing$
  end if
  $\mathbf{f}_{best} \leftarrow \arg\max_{\mathbf{f} \in \mathcal{C}} IG(\mathbf{f}, \mathbf{r})$
  $c \leftarrow findSplit(\mathbf{f}_{best}, \mathbf{r})$
  $\mathbf{r}_l, \mathbf{r}_r \leftarrow \{r \in \mathbf{r} | c(r)\}, \{r \in \mathbf{r} | \neg c(r)\}$
  $\mathcal{F}_l \leftarrow$ BuildAFTree($\mathbf{S}, \mathbf{r}_l, \omega, m, minsup$)
  $\mathcal{F}_r \leftarrow$ BuildAFTree($\mathbf{S}, \mathbf{r}_r, \omega, m, minsup$)
  return $\mathcal{F}_l \cup \mathcal{F}_r \cup \{\mathbf{f}_{best}\}$
```

At each iteration of feature selection, new candidate aggregate features are generated according to the current approximation of the target. In the first iteration, Algorithm 6 uses the response $\mathbf{r}$ as a target to build a set of candidate aggregate features. From this set, it chooses the one with the highest score to add to the set of proposed features $\mathcal{F}$. A linear model is then fitted ($fitLM$) to the response variable $\mathbf{r}$, using the set of proposed features $\mathcal{F}$. In the following iterations, the residual (the part of the signal that cannot be predicted by the current model) becomes the new target variable, $\mathbf{t}$. At the end of the process, only a small subset of the constructed aggregate features is returned.

Classification problems

Decision trees are among the most popular classification models in machine learning, and one of its best-known characteristics is the ability to deal with multiple types of data [80], including trend-less time series. Growing a decision tree involves a divide-and-conquer strategy where each node splits the data into subsets according to conditions on the predictors, until splitting no longer increases the separation between classes. To explore time series with mixed sampling rates, we designed an embedded feature construction and selection method for decision trees, where at each split new features are constructed such that the scoring function information gain (IG) is maximized.

In our method, trees are built recursively. Algorithm 7 shows that at each iteration, a new set $\mathcal{C}$ of candidate aggregate features is constructed, through a search process looking for the best combinations of aggregate features and response variable $\mathbf{r}$. From $\mathcal{C}$, only the aggregate feature that maximizes IG will be used to produce a split: $findSplit(\mathbf{f}_{best}, \mathbf{r})$. The split is then used to create two branches, corresponding to the decomposition of the response variable $\mathbf{r}$, into two subsets $\mathbf{r}_l$ and $\mathbf{r}_r$. The subsets are then used recursively to create more splits until one of the following stopping criteria is satisfied:

- The set of candidate aggregate features $\mathcal{C}$, is returned empty from Algorithm 5.
- The target subset ($\mathbf{r}_l$ or $\mathbf{r}_r$) is smaller than a minimum support, $minsup \in \mathbb{N}^+$.

Note that we specifically do not use pruning techniques during feature construction, to avoid getting too small a feature set. Decisions about pruning strategies can be applied during the final stage of model building by the tree induction method of choice.

4.4 Experiments

In this section, we test our method experimentally, presenting results on the raw data of three datasets collected using multiple sensor systems. The first dataset features a classification problem, involving snowboarding in the Alps. The second and third dataset involve several regression problems, one related to the running speed estimation of an athlete as captured by a GPS sensor, and one describing the amount of sleep of different kinds, as a function of a person's daily routines. The algorithms described in the previous section and further data mining techniques described in this section were implemented in R [81].

In each experiment, we not only compute the results for our embedded method, but also consider traditional two-step alternatives: construction and then selection. Feature construction alternatives include a baseline aggregation method, and grid search feature construction. The baseline aggregates over a non-overlapping window of size $f_\mathbf{S}/f_\mathbf{r}$ by simple averaging. For the grid search approach, we materialize a rather large amount of aggregate features. The grid search is bounded, such that it generates an aggregate feature matrix of approximately 5 million cells, allowing an absolute comparison across datasets.

Dataset	Target	# Input samples	# Output samples	Method	# Features constructed	# Features selected	time (s)	Accuracy/ R^2
Snowboard	Activity	21 180	353	Accordion	30 180	15	305	**84.7%**
				Baseline	25	10	0.5	67.1%
				Grid search	17 150	15	526	83.1%
Speed Estimation	Speed (m/s)	951 200	2088	Accordion	11 580	2	2 654	**0.986**
				Baseline	6	3	1.99	0.388
				Grid search	2 394	25	1 141	0.508
Daily Routines	Light (h)	1 296 575	15	Accordion	19 260	4	2 334	**0.8407**
				Baseline	34	6	28.7	0.2089
				Grid search	323 400	36	62 409	0.8391
	REM (h)	1 296 575	15	Accordion	23 610	5	3 105	**0.9184**
				Baseline	34	4	28.7	0.1837
				Grid search	323 400	11	62 409	0.8838
	Deep (h)	1 296 575	15	Accordion	11 710	2	1 451	0.6719
				Baseline	34	4	28.7	0.0842
				Grid search	323 400	15	62 409	**0.8457**

Table 4.1: Experimental setup and results.

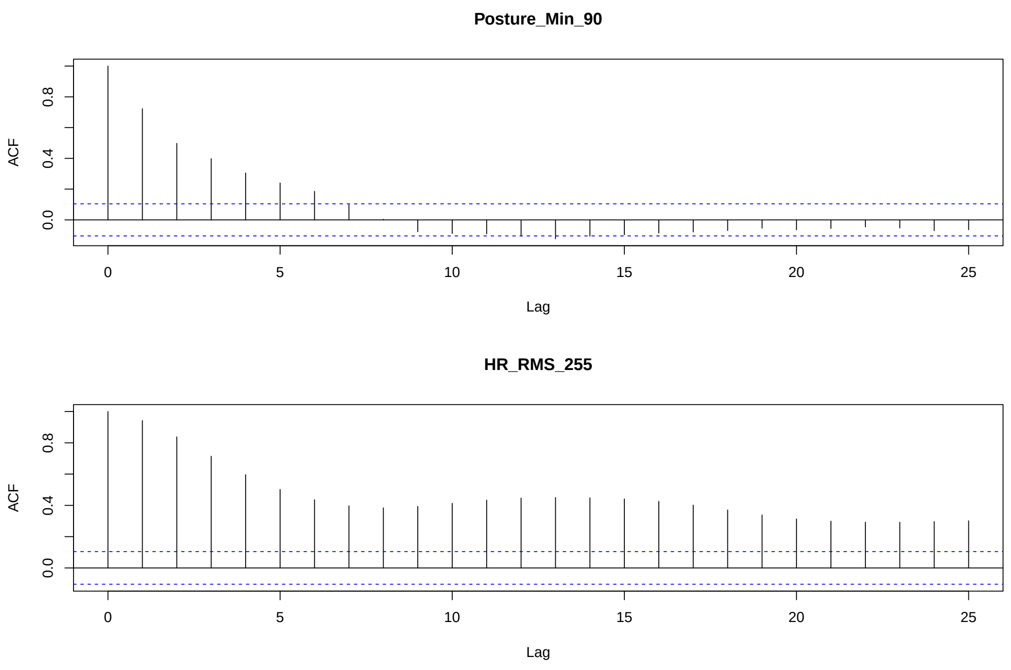


Figure 4.4: The plot shows the autocorrelation behaviour for two randomly sampled aggregate features, from the Routines dataset.

The alternative feature construction methods are followed by well-known feature selection methods, specifically Lasso for regression [89] and C4.5 for classification. For Lasso, we used the default options proposed in the glmnet [26] package in R. To choose the penalty parameter, we employed cross-validation on the training set, and chose the penalty parameter that minimizes the mean squared error.

In most cases, time series have a natural temporal structure. This prohibits us from assuming that subsequent aggregate feature instances are *iid* (independent and identically distributed). This follows from the fact that closer observations have a stronger relation than those further apart. Figure 4.4 shows the autocorrelations of two aggregate features sampled randomly from Accordion's candidate features, $\mathcal{F}$. As expected, all of them have a clear temporal structure. Breaking the *iid* assumption restricts the model evaluation methods that can be applied. For example, cross-validation should only be applied when features can be assumed to be *iid*. Consequently, we split the data into 66% of data before a selected point in time (the training set), and 34% for testing.

4.4.1 Snowboard Data

This experiment involves sensors collecting physiological signals from a subject while doing winter sports in the Alps. To collect this physiological data, we used a Zephyr BioHarness 3[2] sensor system, that is worn on the subject's chest. The BioHarness incorporates multiple sensors (ECG, chest expansion, temperature and tri-axial acceleration), that are embedded in a monitoring module and a lightweight strap. The system samples at multiple sampling rates for the different sensors, and derives from them a total of 25 physiological parameters (heart rate, breath rate, posture, peak acceleration, ...), logged at 1 Hz. The dataset used in this experiment was collected during 353 minutes of snowboarding. During the collection period, the subject used the BioHarness and a GoPro Hero3 HD camera. Afterwards, the video data was used to label the activities for each minute, from the following available labels: *lift, lying, sitting, snowboarding, standing* and *towlift.*

The baseline consists of 25 averaged predictors, with a sliding window of size 60, at 1/60 Hz (once per minute), matching the frequency of the target labels. Table 4.1 presents information for each decision tree built, where the baseline achieved a predictive accuracy of 67.1%. We used an implementation of C4.5 from the package rWeka[3], which allows R users to use Weka's[4] machine learning algorithms. Reduced error pruning was used as post-pruning strategy.

We employed the classification version of Accordion (Algorithm 7). The algorithm used information gain as scoring function, and was allowed to grow windows up to 5 minutes in length. After considering 30 180 candidate aggregate features, only 15 were selected to compose the set of aggregate features ($\mathcal{F}$). Figure 4.5 presents the resulting decision tree, with a prediction accuracy of 84.7% on the test set. At the root of the tree, active activities are separated from passive ones based on the minimum heart rate over the last minute. The right side of the tree distinguishes between active or recently active activities using heart rate, acceleration and breathing as input predictors. The left side of the tree predominantly uses acceleration variables to classify between the different passive activities. Since the predictors were logged at 1 Hz, the window sizes can be interpreted as the number of seconds aggregated. The variety of window sizes and aggregate functions (see Figure 4.5) reveal features with multiple degrees of integration over time.

[2]http://www.zephyranywhere.com/products/bioharness-3/
[3]http://cran.r-project.org/web/packages/RWeka/index.html
[4]http://www.cs.waikato.ac.nz/ml/weka/

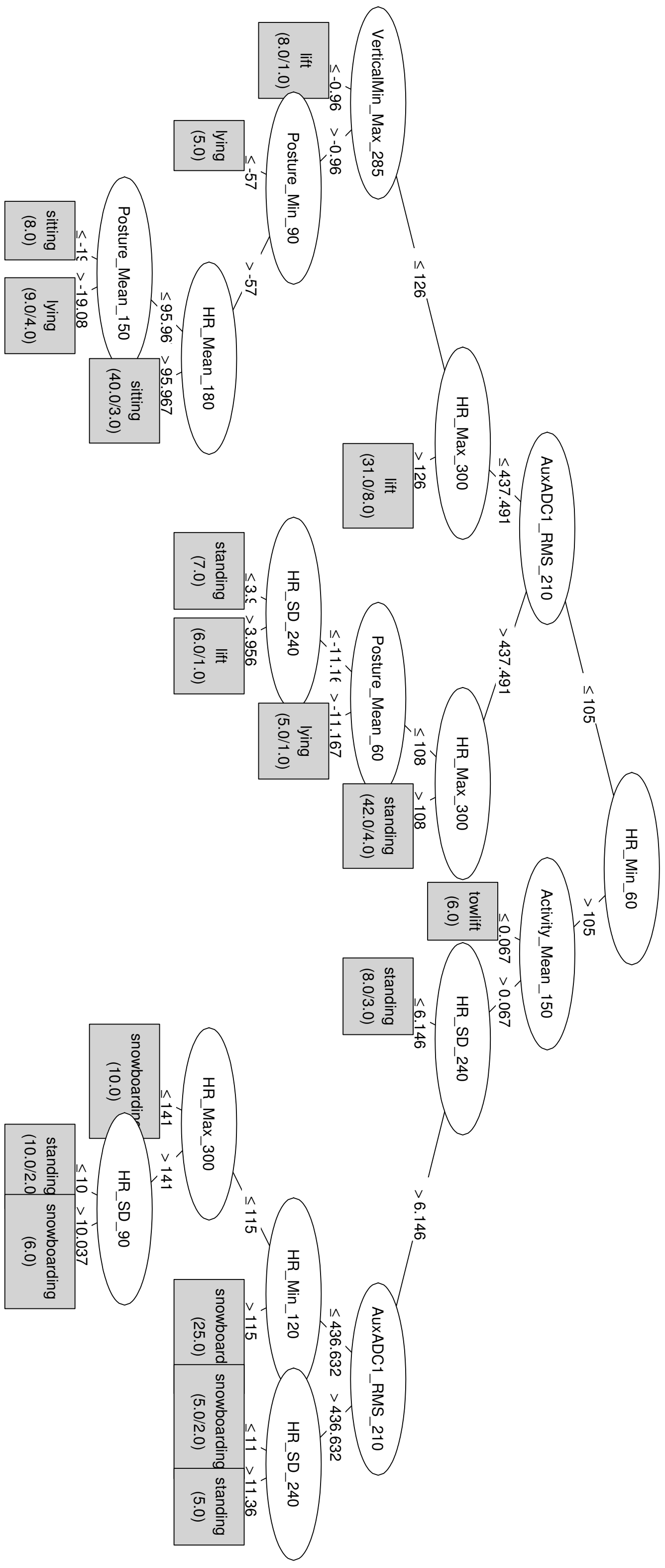


Figure 4.5: Decision Tree C4.5 implemented in Weka, built with the features proposed by Accordion.

The grid search approach materialized 17 150 aggregate features and fed it to C4.5 to build a model from a subset of these. The number of features constructed corresponds to a matrix of 5 million cells, as described before. This involved trying 80 different window sizes for each pair of aggregate function and predictor. Table 4.1 shows that Accordion outperforms grid search both in computation time and model accuracy.

4.4.2 Speed Estimation

This dataset was collected in the context of an athlete training for the Amsterdam marathon. In this context, two accelerometers[5] were worn by the athlete during four training sessions, one strapped to the right wrist and the other to the right ankle. A Garmin Forerunner[6] device was used to measure distance and speed.

The dataset considered here has as input about 2 hours and 40 minutes of running measurements from 2 triaxial accelerometers ($2 \times 3 \times 951\,200$ data points), at a constant sampling rate of 100 Hz. For the same measurement period, the target speed values were extracted from the Garmin Forerunner GPS, of which the median sampling rate is 0.2 Hz. The speed (in m/s) turned out to be captured at an unstable rate, with time lapses between measurements ranging from 1 to 10 seconds. Having an unstable target sampling rate is a specific challenge of this dataset, but one that can fairly easily be handled by our algorithms.

The design of our algorithms assumes that all the measurements are done at constant sampling rates. Note that, as long as the predictors are collected at a constant sampling rate, having an unstable target sampling rate (as is the case here) is not a problem. We used this specific challenge to show how our algorithms can be made to work on a broader set of tasks. In fact only two changes are needed. First, in Algorithm 5, recalculate the relation between predictors and target sampling rates to $q = 100/0.2 = 500$, to reflect the median sampling rate of the target. Second, as a minor modification of Algorithm 4, we calculate beforehand the reference indices i, such that predictors and target variable can be synchronized properly.

As baseline experiment, we aggregated 6 variables (2 accelerometers $\times$ 3 axes) over non-overlapping windows of variable sizes. With the predictors

[5]http://www.geneactiv.co.uk/

[6]https://buy.garmin.com/en-US/US/into-sports/running/cIntoSports-cRunning-p1.html

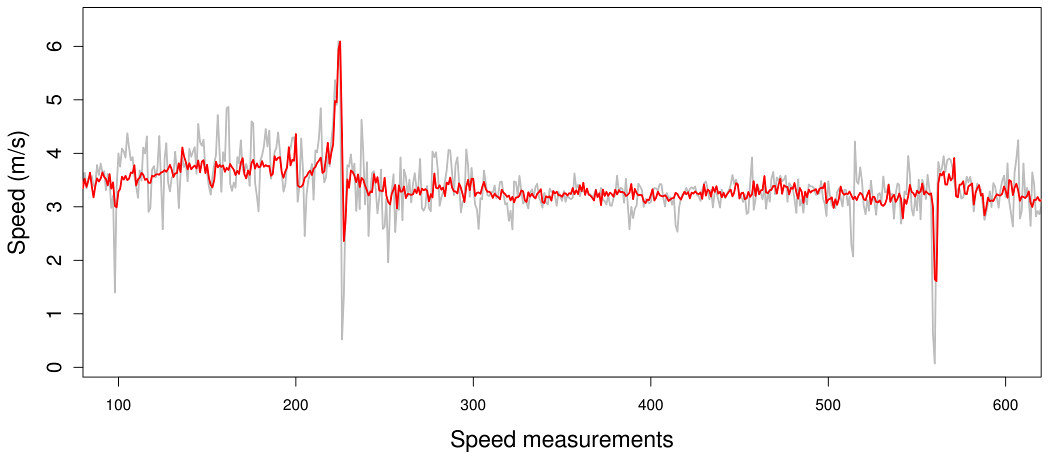


Figure 4.6: Predicted value (black) vs. real speed (gray).

and target variable synchronized, the Lasso regression selected only 3 features, with a fairly low coefficient of determination ($R^2 = 0.388$). As for grid search, using the limit of creating a feature dataset bounded to 5 million cells, we materialized 2 394 aggregate features in about 1 141 seconds, and subsequently submitted them also to Lasso. The achieved $R^2 = 0.508$, although higher than baseline, could not outperform Accordion ($R^2 = 0.986$).

The scoring function chosen in this experiment was cross-correlation, enabling so-called *lag regression*, and the maximum window size was set to 60 seconds. During the iterative process of construction and selection, 11 580 aggregate features were constructed, from which only two were selected (thus, two iterations). The resulting predictions on hold-out data are shown in Figure 4.6. The final lag regression model is as follow:

$$\begin{aligned} speed[i'] &= 1.198 \cdot af_{\textbf{ankleY},RMS,200}[i'] \\ &\quad +0.495 \cdot af_{\textbf{ankleZ},stdv,5886}[i'-4] + e[i'] \end{aligned}$$

Interestingly, the accelerometer strapped to the wrist was never selected to model the speed of the athlete. Also, the selected features use both short term (window size of 200 equals 2 seconds of accelerometer data) and long term information (window size of 5886 aggregate approximately 1 minute). Both these observations can help domain experts redefining future data collections and understanding multiple temporal phenomena. All this insight can be leveraged by the usage of aggregation functions well known to specific domains.

4.4.3 Daily Routines Data

Subsequently, we tested our method on another dataset involving three regression targets, related to the amount of time spent in *light sleep, REM* and *deep sleep*. During the course of 15 days, a subject produced data in the context of a self-tracking experiment, collected using various sensoring systems: a) the Zephyr BioHarness (see above), used during the day (except during bathing), b) OpenBeacon, an RFID wireless sensor system to monitor the time spent at different locations of the home, and c) a Beddit[7] sleep monitoring system to monitor the nights. This last system is used both for recording the breath and heart rate during the night, as well as determining the different sleep stages at night (a computation that is part of the black-box service of Beddit).

The dataset used for this experiment consisted of 34 input attributes, sampled at 1 Hz, of which 24 are physiological variables from both the BioHarness, and Beddit. The remaining 10 variables are binary. They refer to the subject's location and were extracted both from the OpenBeacon and Beddit sensor systems. Table 4.1 summarizes the experimental setup for all targets, as well as the information about results.

As baseline experiment, we used the same idea of feature construction for the previous baselines. Although the baseline is quite fast in terms of computation, the R^2 results (on the test dataset) show that it is a naive solution. As for the grid search solution, we materialized 323 400 aggregate features after about 17 hours of computation, which is considerably slower than what Accordion took to construct and select its aggregate features. After performing Lasso to all the targets, the R^2 results show that this alternative was outperformed by Accordion in two of the three targets.

For Accordion, we used cross-correlation as a scoring function, and the maximum window size allowed was 3 days. With our method, the target that took the longest to compute (about 50 minutes) was the REM stage of sleep. REM sleep is considered the lightest stage of sleep [36]. Figure 4.7 shows the stack of nested models created by forward selection. For this target, the algorithm constructed 23 610 aggregated features, from which only 5 (5 iterations) are proposed to explain REM linearly.

[7]Available from http://beddit.com

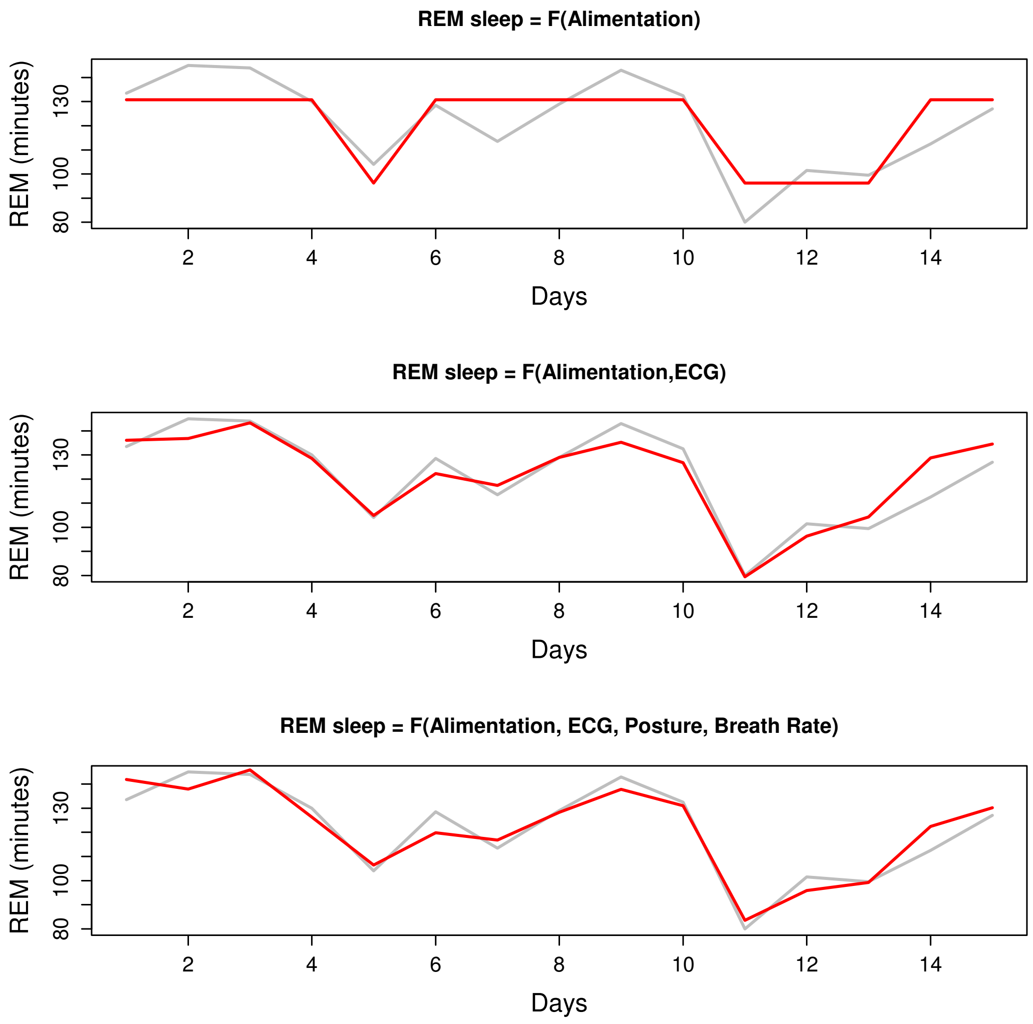


Figure 4.7: Predicted value (black) vs. real amount of REM sleep (gray), for models based on the first single, two and five (all) features, respectively.

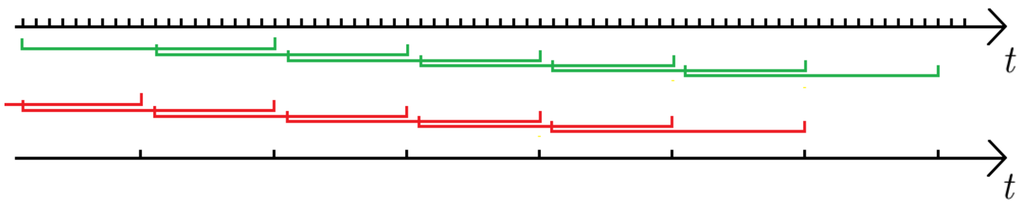


Figure 4.8: High frequency time series can be transformed into aggregated features, resulting in a linear lag regression between these and the target variable.

From the three targets addressed in this experiment, time spent in deep sleep gives a good example of both failure and success of our method. Accordion model scored lower than the one produced Lasso with grid search in terms of R^2, where Lasso wins over our *forward-stepwise selection* process in terms of model accuracy. On the other hand, Accordion is faster and produced an interpretable model (two selected aggregate features), whereas Lasso with grid search fails to deliver an interpretable model (check Table 4.1: R^2 and #Features selected). Our method produced 11 710 candidate features, from which only two were selected to explain the amount of deep sleep, resulting in the following linear model:

$$\begin{aligned} deepSleep[i'] &= -120.8 + 0.99 \cdot af_{\mathbf{Posture},RMS,158400}[i'] \\ &\quad +0.11 \cdot af_{\mathbf{HR},avg,155200}[i'-1] + e[i'] \end{aligned}$$

Figure 4.8 helps us to interpret the *deepSleep* model. Both aggregate features have window sizes of about two days, with almost 50% of overlapping.
For each moment i', deep sleep can be explained with posture over the last two days $af_{Posture,\ RMS,158400}(i')$, and the heart rate of almost two days with a delay of one day $af_{HR,\ avg,155200}(i'-1)$.

4.5 Related Work

The problem of activity recognition is commonly tackled with a two-stage process [6, 69, 58]: first, manually construct aggregate features and then apply a machine learning technique to discriminate between different activities. The task of feature construction is so central that surveys of feature construction techniques became necessary [25]. As the choices in feature construction influence all the experiments, this often leads to solutions that are overly specific to the experimental setup (sensors used, data collection, application). As a result, most methods are not generic [12]. Our algorithm embeds feature construction into the learning process, which increases the feature search space, reduces the time spent pre-processing data and avoids overly specific solutions, which makes it widely applicable.

From a data mining perspective, the use of time series as a data source has received considerable attention, and has developed into different areas of research [48, 102, 60], e.g. classification, summarization, subsequence clustering, motif discovery and anomaly detection. As for the challenge of mining time series with mixed sampling rates, this still remains underappreciated.

To the best of our knowledge, this paper is the first attempt to develop a data-driven generic solution to this problem, and to focus on the importance of optimizing the *automatic and dymanic construction and selection* of aggregate features with respect to a target variable, as opposed to static feature construction.

In econometrics, regression models are commonly used to relate variables at the same sampling frequency, even when the data sources are being collected at different rates. When dealing with mixed sampling rates, the most common technique is still to downsample the predictors [4], or upsample the target variable [3]. Recent work proposes solutions to forecast directly from variables with mixed sampling rates, both for univariate [28] and multivariate time series [57]. These proposed methods still rely mostly on the expertise and creativity of the economists (domain knowledge-driven), leaving no room for data-driven knowledge discovery, which our algorithm is capable of doing.

4.6 Conclusions and Future Work

When modeling time series for activity recognition, a drawback is the considerable amount of time required to pre-process them into good features, a process that often calls on domain knowledge about the underlying problem. Accordion shortens the pre-processing time, generating candidate aggregate features automatically by optimization of its components, $(\mathbf{s}, a, w)$. We still believe that domain knowledge can play a big role when modeling, but this effort could be redirected to higher level questions, such as which set of aggregate functions (A) to use. One of the directions for future work is naturally to extend the set of aggregate functions, especially to the frequency domain.

We also motivate the idea of embedding automatic feature construction into the machine learning process. The idea here is to stop relying on static sets of features, and at each iteration of feature selection direct the search for good candidate features. Making use of scoring functions, Accordion is able to test many candidate features, and return only a small set of selected features. Especially when quick but reliable results are required, or large datasets dictate a memory-conscious method, our algorithm is clearly a good choice. As future work, we would like to mimic the learning process of other supervized methods, both in regression and classification, keeping the idea of learning algorithms that do not rely on a static set of features.

One of the achievements of our approach is that it outputs interpretable aggregate features. The ability to interpret our aggregate features follows from the combinations of input variables, well-known aggregate functions and different window sizes. Interpretation of different window sizes come from the fact that our method searches for different phenomena by expanding or contracting the window size for each feature. In contrast, the standard approach in activity recognition is to take a more or less arbitrary choice about a window sizes [6, 69, 58, 37]. In the future, we would like to deal with unstable sampling rates, both of predictors and target, multiple sampling rates for the predictors, and targets at a *higher* sampling rate than the predictors.

Chapter 5

ClaRe: Classification and Regression Tool for Multivariate Time Series

Ricardo Cachucho, Stelios Paraschiakos, Kaihua Liu, Benjamin van der Burgh, Arno Knobbe

in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2018

Abstract

As sensing and monitoring technology becomes more and more common, multiple scientific domains have to deal with big multivariate time series data. Whether one is in the field of finance, life science and health, engineering, sports or child psychology, being able to analyze and model multivariate time series has become of high importance. As a result, there is an increased interest in multivariate time series data methodologies, to which the data mining and machine learning communities respond with a vast literature on new time series methods.
However, there is a major challenge that is commonly overlooked; most of the broad audience of end users lack the knowledge on how to implement and use such methods. To bridge the gap between users and multivariate time series methods, we introduce the ClaRe dashboard. This open source web-based tool, provides to a broad audience a new intuitive data mining methodology for regression and classification tasks over time series.

5.1 Introduction

Over the past few years, there is an increased interest in the analysis of multivariate time series data. A great deal of this interest is motivated by advances in sensor technology. In many application areas, deploying sensors for continuous monitoring has become a common strategy. Over the last 10 years, sensors are becoming more accurate, with better data communication protocols, smaller and last but not least, cheaper.

As a motivating example of the great developments in sensing technology, consider the UvA BiTS sensor system [11]. This bird tracking system currently weighs around 10 grams, is powered by a solar panel, and integrates a GPS and a tri-axial accelerometer. This bio-logger, designed for birds with at least 300 grams is capable of collecting, saving and transmitting the animal movements and overall migration. This sensor system ends up weighing less than $< 3\%$ of the bird body mass. This is an example of a system that is less invasive than traditional sampling methods in ecology, which normally involve multiple captures and releases. Therefore, both the possibilities for ecology studies and the amount of available data has expanded: more species, exact migrations, flight strategies, bird activities and foraging strategies. But how to efficiently and intuitively explore sensor data, still remains a data science challenge.

From the data science perspective, sensor systems will produce time series data. In the case of sensor networks, multiple variables are collected simultaneously, producing multivariate time series. Adding to that, when collected continuously, these datasets lead to big data challenges. This raized challenges to the data mining community, on how to deal with large multivariate time series. These challenges have attracted the attention of many researcher and lead to a vast literature on time series mining [24, 14]. With the exception of a few good examples [32, 10], there is still a gap between most of these methods and the potential end users, who may lack a technical background to implement them.

Most of the sciences based on empirical observations have the potential to benefit from technological advances in sensor systems:

- children can be monitored continuously to study their social competence [99, 100];
- environmental sciences can benefit from continuous sensing [11];

- civil engineering can develop predictive maintenance of infrastructures using sensor networks [68, 98, 67];
- life sciences and health are already heavily supported by machinery that uses sensors to measure all sort of phenomena [40, 106, 95].

A common link between all the above-cited publications is that they rely on sensor monitoring systems for their continuous sampling methodologies. The continuous nature of the measurements, lead to large multivariate time series datasets. As a consequence, the traditional data analysis tools based on classical statistics are commonly not applicable to this kind of data. This leads to an opportunity to shorten the gap between the data science community and empirical sciences, if we are able to create the appropriate tools.

One could argue that the data mining community is already encouraging the publication of source code and data associated with publications. However, without a deep knowledge on the published method and the language used to implement the code, such released source code targets only a limited audience. Another very significant effort to make machine learning methods more accessible is the release of packages with collections of algorithms, such as Scikit-learn [77] for Phyton or Caret [56] for R. The downside of such packages is the need to be proficient both in the programming language that implements the package of methods and the need to know how to build a data science methodology around the chosen method. At last, there are tools for a broad audience such as Weka [32], MOA [10], Knime [8], JMulTi [55] and SPSS [38], which are intuitive and provide graphical user interfaces. The problem with such tools is that upon the development of a new method, these tools are not flexible enough to easily incorporate them. Furthermore, focusing on multivariate time series, most of them are not designed to analyze this kind of data.

Our proposal to bridge the gap between new methods and a broad audience, is to build easily accessible web-based tools, with a user interface. Such tools require no installation, are platform-independent and can be highly intuitive. Intuitive, because most people already have been exposed to hundreds of web pages and know how to read and navigate them. With an accessible GUI, these tools will broaden the potential audience to non-experts in data science. Additionally, using the web interface allows us to present such tools as Software as a Service (SaaS).

As a motivation, consider the example of a healthy aging study, developed by a team with a biomedical background [95]. The study used multiple sensor platforms in order to predict the participant's activities and energy expenditure. This resulted in a dataset where a team of life science researcher is left to deal with multivariate time series. Therefore, it is essential to have at their disposal a tool that implements a data mining methodology that allows them to run experiments and model such multivariate time series data.

In this paper, we propose *ClaRe*[1], a *Cla*ssification and *Re*gression tool to model supervized multivariate time series. This SaaS tool adopts the *Accordion* algorithm from the previous chapter, to learn informative features and allows users to learn regression and classification models from multivariate time series with mixed sampling rates. Its intuitive web-based interface provides options of importing, pre-processing, modeling and evaluating multivariate time series data. In every step, plotting and saving data or results are allowed. Adding to the aforementioned, both source code and experimental data[2] are made openly available.

5.2 Tool Overview

ClaRe is a web-based tool that incorporates all the necessary steps for modeling time series with mixed sampling rates. Such time series are often collected from a network of sensors that measures complex phenomena. The output of such sensors are often multiple files that have variables measured at different rates and thus have special needs:

- pre-processing needs to include synchronization and merging;

[1]`http://fr.liacs.nl:7500`
[2]`https://github.com/parastelios/Accordion-Dashboard`

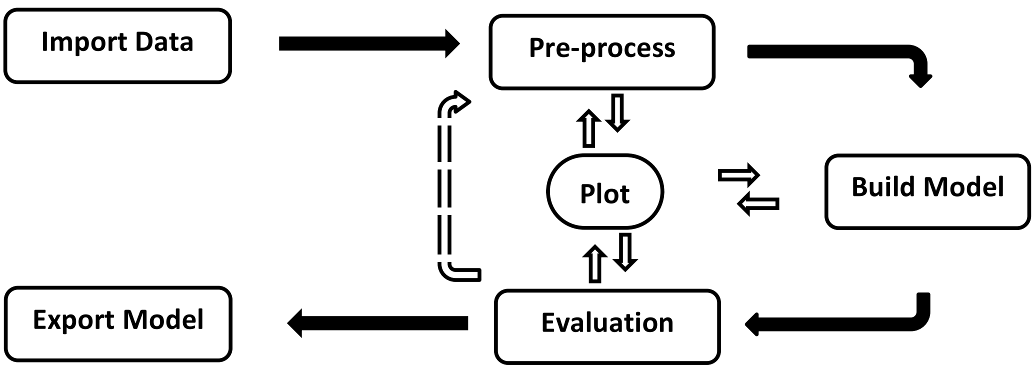


Figure 5.1: An overview of *ClaRe* tool architecture.

- plotting needs to be done using sampling techniques due to the size of such time series;
- learning strategies that take into account the temporal nature of the data;
- evaluation alternatives to cross-validation that reflect the true accuracy of the time series models.

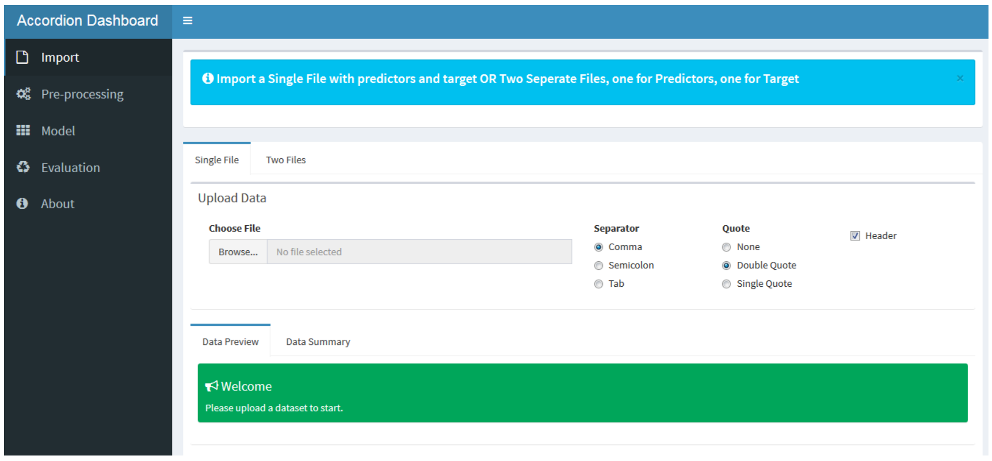
Our tool is novel due to the capacity to deal with the challenges above, which are recurrent when dealing with sensor data.

From a technical perpective, ClaRe also presents benefits in terms of development and deployment. Both front end and server are developed with R, using the *R Shiny* package [18]. This package provides a framework to interact between client and server side through R-scripts. As a result, the tool was easy to implement since only one programming language is used to manage both server and front end. From the deployment perspective, ClaRe's main advantage is its compatibility with all modern web browsers.

ClaRe's design presents an experimental methodology as shown in Figure 5.1. One can import and pre-process time series data, build regression or classification models, evaluate them, and export the results. The user can follow the proposed methodology intuitively, using web components that adjust to the user choices and guides the user troughout the data mining methodology. Each panel will be enumerated and explained below, following the CRISP-DM standards of the data mining methodology [107].

Import: When the user accesses the tool online, they are welcomed to the tool by the *Import* panel. To start, the user can upload predictors and target in a single or separate files (see Figure 5.2). In this panel the user can get a preview of the data available and descriptive statistics for all the variables available.

Pre-processing: Having imported the data, the user will be intuitively guided to the following panel: *Pre-processing*. Here, the user can choose from multiple pre-processing tasks, both generic for all sorts of datasets and specific to sensor-based multivariate time series. The generic tasks include selecting the variable the user wants to consider as a target, normalizing datasets, removing outliers. As for tasks that are more specific to multivariate time series datasets, one can merge multiple files into one dataset, synchronize data from different sensors and manage missing values.



Import a Single File with predictors and target OR Two Seperate Files, one for Predictors, one for Target

Upload Data

Choose File: Browse... No file selected

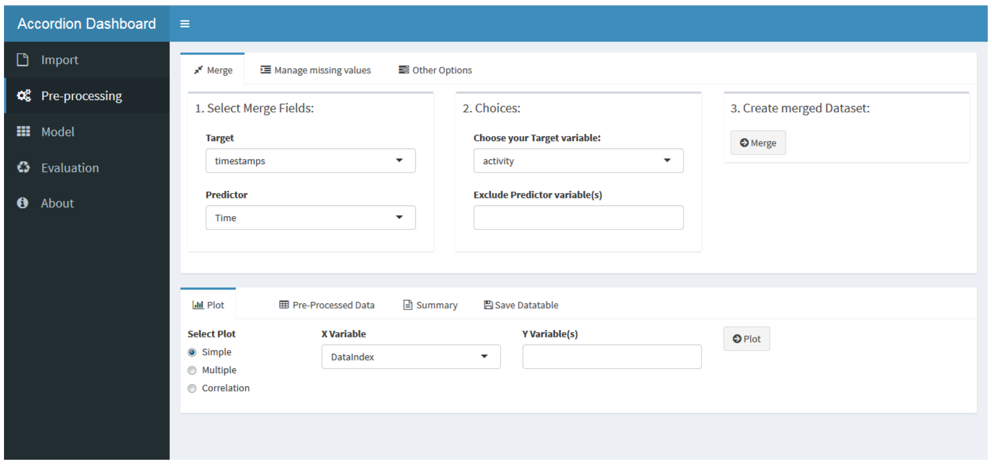
Separator: [✓ Comma] Semicolon Tab

Quote: None [✓ Double Quote] Single Quote

[✓ Header]

Welcome
Please upload a dataset to start.

Figure 5.2: *ClaRe* dashboard user interface: Import tab.



1. Select Merge Fields:

Target: timestamps

Predictor: Time

2. Choices:

Choose your Target variable: activity

Exclude Predictor variable(s):

3. Create merged Dataset:

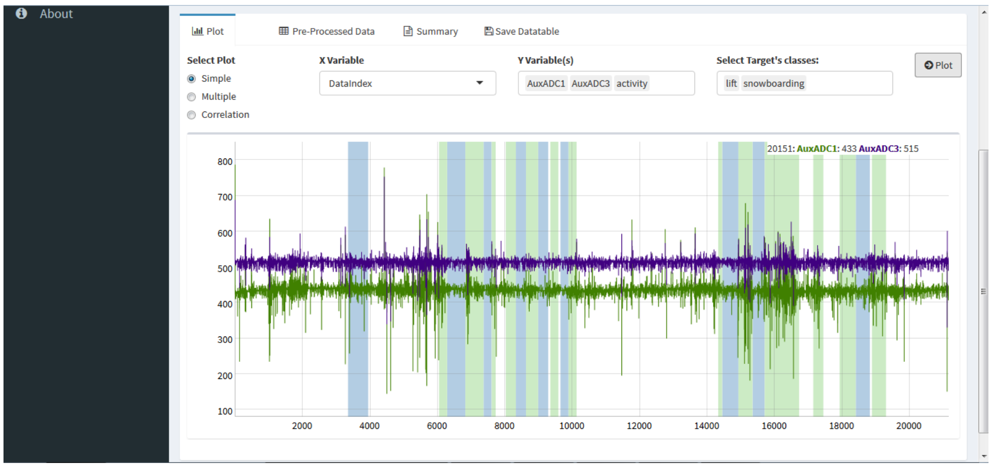
Select Plot: [✓ Simple] Multiple Correlation

X Variable: DataIndex

Y Variable(s):

Figure 5.3: *ClaRe* dashboard user interface: Pre-processing tab.

As an example of a common pre-processing task in multivariate time series, consider a sensor network where the predictors (e.g. accelerometers) are collected and saved into one file and the target (e.g. a persons activities) into a different file. The user would want to select the target and merge both files into one synchronized dataset. For that purpose, with ClaRe the user can choose the relevant variables and *merge* them with ease (see Figure 5.3). Please note that when the user selects the target, there is a selection of which model will be used: if the target is nominal the following panel turns into a *Classification* panel (see Figure 5.5); if the target is numeric the following panel turns into a *Regression* panel (see Figure 5.7).



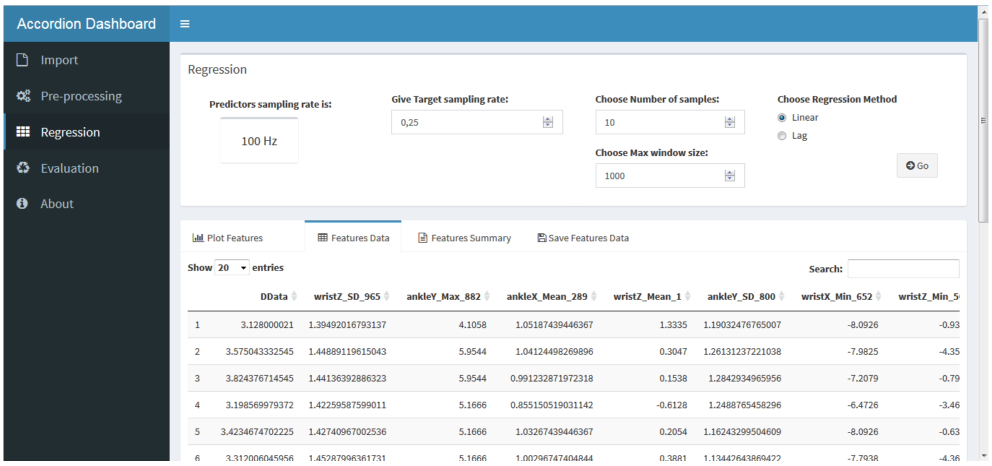
Select Plot: [✓ Simple] Multiple Correlation

X Variable: DataIndex

Y Variable(s): AuxADC1 AuxADC3 activity

Select Target's classes: lift snowboarding

Figure 5.4: *ClaRe* dashboard user interface: Plotting tab.



Predictors sampling rate is: 100 Hz

Give Target sampling rate: 0,25

Choose Number of samples: 10

Choose Max window size: 1000

Choose Regression Method: [✓ Linear] Lag

Show 20 entries

	DData	wristZ_SD_965	ankleY_Max_882	ankleX_Mean_289	wristZ_Mean_1	ankleY_SD_800	wristX_Min_652	wristZ_Min_5
1	3.128000021	1.39492016793137	4.1058	1.05187439446367	1.3335	1.19032476765007	-8.0926	-0.93
2	3.575043332545	1.44889119615043	5.9544	1.04124498269896	0.3047	1.26131237221038	-7.9825	-4.35
3	3.824376714545	1.44136392886323	5.9544	0.991232871972318	0.1538	1.2842934965956	-7.2079	-0.79
4	3.198569979372	1.42259587599011	5.1666	0.855150519031142	-0.6128	1.2488765458296	-6.4726	-3.46
5	3.4234674702225	1.42740967002536	5.1666	1.03267439446367	0.2054	1.16243299504609	-8.0926	-0.63
6	3.312006045956	1.45287996361731	5.1666	1.00296747404844	0.3881	1.13442643869422	-7.7938	-4.36

Figure 5.5: *ClaRe* dashboard user interface: Model tab.

Furthermore, there are multiple options to deal with missing values for time series, such as interpolation and repeating values. The panel of pre-processing is also allows more common tasks such as outlier removal, data normalization and conditional selection of variables.

In every step of the pre-processing, there are two useful inspection panels worth mentioning; plotting and saving. The plotting panel has multiple interactive functionalities, such as real time zoom in/out and target shading (see Figure 5.4). These plots are implemented using R's *dygraphs* package [97], which is capable of dealing with large datasets. At any time, the pre-processed dataset can be previewed and saved for further usage, allowing the experiments to continue over multiple user sessions.

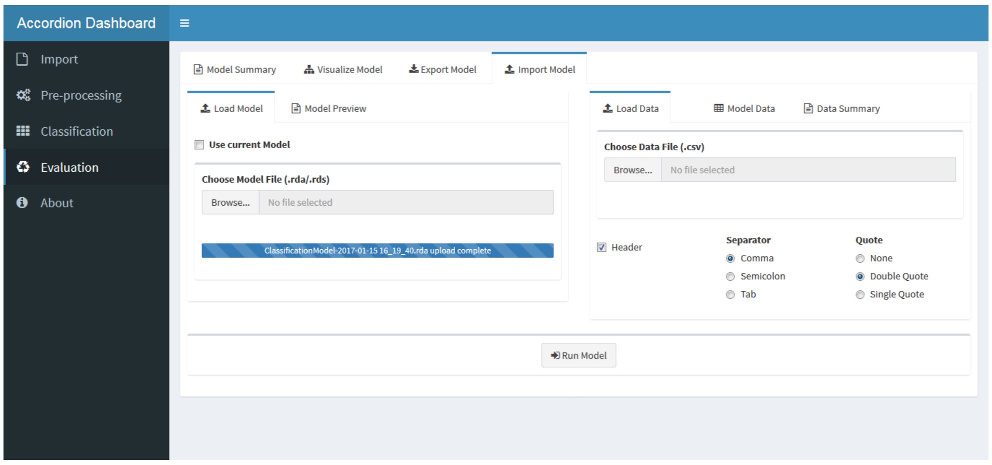
Model: As mentioned before, after choosing a numeric or nominal target, this panel changes into a regression or classification setup, respectively. The available regression models are a linear regression model and a lag regression model. As for the classification task, the available model is a decision tree. Both classification and regression models construct and select aggregate features as described in the previous chapter.

Accordion can be tuned with multiple parameters, which are available in the *Regression* or *Classification* panels (see Figure 5.5). For both classification and regression, one can tune the target's sampling rate, the maximum window size and the number of samples used to perform a greedy search for aggregate features. Additionally, in regression there is an option for which regression method to use (linear or lag). To ease the users mining process, the parameter defaults are computed automatically, according to the sampling rates in the original files.

After having the parameters tuned, the users can start the learning procedure of each model by clicking a Go-button. Then, *ClaRe* starts running the Accordion algorithm on the server side of the tool and in the front end a loading-cycle is displayed until the model is constructed. After the model is built, the user can have a first inspection of the selected features by visualizing them, inspecting descriptive statistics or checking which are the variables, aggregation function and window sizes that have been selected.

Evaluation: This panel allows the users to obtain detailed cross-validated evaluations of the constructed model. An added feature is cross-validation with Leave One Participant Out (LOPO) for models using multiple participants. Such type of scenarios can be found in applications such as activity recognition for multiple people [12, 14] or birds in the examples given in the introduction. With LOPO, the model is built multiple times, leaving each time one participant out of the learning process to validate. This evaluation method is especially important to assess the real accuracy of models, when the dataset instances are not independent and identically distributed.

The evaluation of results is different for regression and classification models. For regression, it provides a full summary of the computed coefficients and model errors. For classification, it returns the confusion matrix and the associated accuracies. Furthermore, a visualization panel is available, as presented in Figure 5.7. For both regression and classification, the user can plot the dataset with the aggregated features and the predicted target. Please note that the resulting dataset with aggregate features, has the same sampling rate as the target, which can be specified as a parameter in the *Regression* and *Classification* panel.



Use current Model

Choose Model File (.rda/.rds)
No file selected
ClassificationModel-2017-01-15 16_19_40.rda upload complete

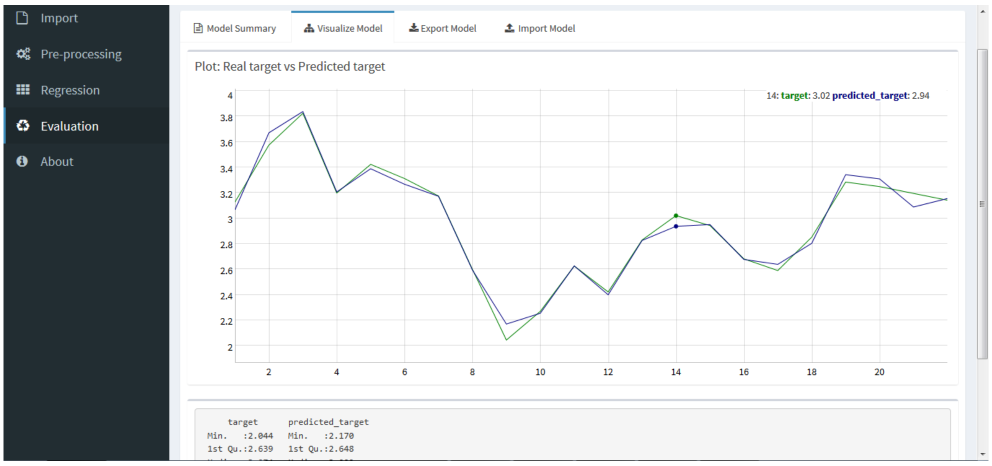
Choose Data File (.csv)
No file selected

[✓ Header]

Separator: [✓ Comma] Semicolon Tab

Quote: None [✓ Double Quote] Single Quote

Figure 5.6: *ClaRe* dashboard user interface: Evaluation tab



```
    target        predicted_target
Min.   :2.044   Min.   :2.170
1st Qu.:2.639   1st Qu.:2.648
```

Figure 5.7: *ClaRe* dashboard user interface: Visualization of predictions

There are other functionalities that expand the evaluation possibilities. For a start, one can export the constructed model and the associated dataset. Additionally, an existing model can be evaluated using new datasets, creating a train and test evaluation scenario (see Figure 5.6). Finally, the user has the option to directly evaluate both the existing model and new datasets. All these options give the flexibility to re-visit the evaluation over multiple user sessions.

5.3 Conclusion

This paper presents an easily accessible web-tool designated as *ClaRe*. *ClaRe* is a Software as a service (SaaS), which provides any user interested in mining multivariate time series, a methodology for supervized learning. More specifically, it allows users to deal with cases when the multivariate time series data have mixed sampling rates. Making use of intuitive menus, one can easily load one of multiple files, pre-process properly sensor systems data, learn time series models and evaluate the results. Additionally, *ClaRe* is a freely distributed and open source software, that allows reproducible research in a SaaS environment.

At any stage of the mining process, interactive plotting and saving options (for models and data) are available. Being built with such options, the tool allows the mining process to be revisited over multiple user sessions, giving additional flexibility to the users. The plotting facilities of the tool are built to deal with large datasets and to give further insights at each step to the users.

Chapter 6

Sports Analytics for Professional Speed Skating

**Arno Knobbe, Jac Orie, Nico Hofman,
Benjamin van der Burgh, Ricardo Cachucho**

in journal of the Data Mining and Knowledge Discovery, Volume 31, Issue 6, pp 1872–1902, Springer, November 2017

Abstract

In elite sports, training schedules are becoming increasingly complex, and a large number of parameters of such schedules need to be tuned to the specific physique of a given athlete. In this paper, we describe how extensive analysis of historical data can help optimize these parameters, and how possible pitfalls of under- and overtraining in the past can be avoided in future schedules. We treat the series of exercises an athlete undergoes as a discrete sequence of attributed events, that can be aggregated in various ways, to capture the many ways in which an athlete can prepare for an important test event. We report on a cooperation with the elite speed skating team LottoNL-Jumbo, who have recorded detailed training data over the last 15 years. In this project, we analyse this data, and extract actionable and interpretable patterns that can provide input to future improvements in training. We present two alternative techniques to aggregate sequences of exercises into a combined, long-term training effect, one of which based on a sliding window, and one based on a physiological model of how the body responds to exercise. Next, we use both linear modeling and Subgroup Discovery to extract meaningful models.

6.1 Introduction

This paper describes research challenges related to a recent Sports Analytics project between a leading Dutch professional speed skating team and data scientists from Universiteit Leiden and Hogeschool van Amsterdam. During its history, the athletic team, currently called LottoNL-Jumbo, has included numerous Dutch skaters that competed at the European, world, and Olympic level, and presently includes a world record holder and two Olympic medalists. The project involved 15 years of detailed training data kept by the coach of the team (second author of this paper) with the aim of improving the training program and further optimising the performance of current and future skaters of the team. In this paper, we report on the data science techniques required to analyse this non-trivial data, and showcase findings for specific athletes. A number of novel techniques are introduced to deal with the specifics of the recorded data, and to produce interpretable and actionable results that can help fine-tune the training programs.

Speed skating is a winter sport where athletes compete on skates to cover a given distance on an oval (indoor) ice rink. In this paper we focus here on long-track speed skating events, which involves a 400 meter oval track with two lanes. Events over multiple distances exist, ranging from 500 meters to 10 000 meters, with each skater typically specialising in one or two distances, depending on their physiology and training. Although each race in an event involves two skaters, the final standing is determined by the overall ranking of times of all participants. This effectively makes each race a time trial, where the outcome of a given skater is only determined by their own performance. From a data mining point of view, this is attractive since all results can be assumed independent, and one can simply collect all race results of an athlete without having to consider the influence of the ‘opponent’.

The available data, painstakingly collected by the coach, involves primarily descriptions of the daily training activities, partly structured and partly free text. The structured part of the description is very consistent, and captures a classification of the nature of the training (e.g. “*cycling extensive endurance*”), as well as numeric values indicating the duration (in minutes) and intensity (on a subjective scale of 1 to 10) of the session. Training data is specific to individual athletes, and the intensity of the training was obtained from the athlete, post hoc. With six training days per week, and potentially two sessions per day, this amounts to roughly 450 sessions per season, making for a substantial data collection per athlete/season. Next, race results are available that capture test events which will stand as our target. These were scraped from the internet[1]. As a result, the problem is essentially a *regression task*, since target and predictors are both numeric variables.

While evidently being in the regression domain, it is not immediately clear what the independent variables of our task might be. Clearly, the independent variables should capture aspects of the training program prior to the events in question. However, the available data is a long sequence of events with a small number of characteristics (type, duration, intensity, ...), so some form of transformation is necessary to arrive at an attribute-value representation that is amenable to main-stream regression analysis. In this paper, we take an *aggregation-based feature construction* approach, inspired by earlier work in [14], in order to derive a fairly extensive set of features that capture the preparation (training, but also absence thereof) from various angles, for example focussing on specific periods prior to the test event, or on specific intensity zones. In its basic form, the aggregation will take place over windows of varying lengths (ranging from one day to several weeks) using different aggregate functions and variables, with specifiers such as training type and intensity zones. In a more elaborate approach, developed for this specific purpose, the aggregation will take the form of convolution with a physiology-inspired kernel consisting of several exponential decay functions of varying half times. This kernel is inspired by the so-called *Fitness-Fatigue model* [16], that tries to capture how the human body responds to a specific training impulse over the course of time.

After having obtained a suitable attribute-value representation with potentially predictive features, the next challenge is to produce meaningful models from this dataset. The overall aim of this project is to provide the coaching staff with easy-to-understand, actionable pointers as to how to fine-tune the training routines, and avoid pitfalls of under and over-training. Therefore, we specifically intend to discover interpretable patterns, that are relatively easy to understand by the domain experts, and ideally do not involve a great many variables. We will be working with two types of regression techniques. Assuming mostly linear dependencies between the aggregated features and the target variable, *regularized linear regression* methods such as LASSO [27, 90] are attractive since they select features and produce relatively concise models of the data. However, with the physiological domain at hand, it is likely that non-linear dependencies will also exist, and rather, one expects thresholds to exist on the features, where too large or too low a value (e.g. training load) will produce sub-optimal results. For such phenomena, we expect *Subgroup Discovery* [50, 30, 5, 78] to produce more useful results.

[1]`http://www.osta.nl`

This Sports Analytics paper has two sides. On the Sports side, we present some interesting findings that are of practical relevance to the team, with the following key contributions:

- The application of the Fitness-Fatigue model and the fitting of this model to individual skaters, where the parameters of the model convey key properties of each skater.
- Various demonstrative, interpretable patterns concerning improved training practices.
- The presentation of results relating to competitions.
- The capability to produce detailed findings for other skaters.

On the Analytics side, we introduce a number of new ways to exploit detailed training data, of relevance not just in the speed skating discipline, and in some cases applicable to other analytics domains also. In this domain the key contributions are:

- Introduction of (conditional) aggregation as a way of aggregating discrete sequences of events, and producing a range of features that capture various aspects of those sequences.
- Aggregation by means of two options: one that is easy to compute and interpret (uniform window), one that is more physiologically plausible, and at the same time harder to compute (the Fitness-Fatigue model).
- Application of linear modeling and Subgroup Discovery in order to select key features and produce interpretable models. 5) Evaluation of models in terms of R^2 and p-values, that makes linear models and subgroups immediately comparable.

6.2 Speed Skating and Sports Analytics

The (long-track) speed skating takes place on an oval track 400 meters in length. Races are typically held with two participants at each time (skating in separate lanes), but each participant is ranked on their individual time. Both men and women compete, in separate competitions. Races come in various distances, but the most common distances at major events are 500 m, 1000 m, 1500 m, 3 000 m, 5 000 m and 10 000 m, of which the last distance only applies to men, who in turn do not skate the 3 000 m. These disciplines are usually divided into sprint, medium, and long distance, and skaters typically specialize in one of these, and compete in only a few of these disciplines, although participation is facultative. Each category requires a specific type of physiology, which explains the specialization of athletes. Furthermore, each distance requires a specific type of training, and exercises for one distance may actually harm the performance on other distances [64, 45]. This implies that our analysis will often be specific to a small number of similar distances, or even be specific to individual athletes. Since the training programs are well-developed, and the senior athletes will have several years of experience working with the coach, the produced findings may be subtle, which will often call for an athlete-specific approach.

Even though races for a specific event can be considered time trials, there will be a level of variance in the race results that cannot be explained by differences in race preparation and training. It is a well-know fact within speed skating that times are determined to a reasonable degree by the ice rink. First of all, the ice properties will differ from one venue to the next, and top venues will have better ice maintenance techniques and experience. Another factor that influences the times, besides ice quality, is the altitude of the venue, with higher ice rinks tending to be faster due to the lower air resistance. In order to compensate for the difference in rink speeds, we have opted to work with *relative times* rather than recorded times.

Definition 9 (Relative time)**.** *For a race of distance d, by a skater of gender g, at ice rink r, finishing in time t (in seconds), the* relative time *t_{rel} is defined as*

$$t_{rel} = t/t_{rec}(d, g, r)$$

where $t_{rec}(d, g, r)$ is the record for a specific discipline and ice rink.

Relative time will produce race results slightly above 1.0 or exactly 1.0 if a race either produced or replicated a rink record. The use of relative time not only allows comparisons between results at different venues, but theoretically also comparisons between results in different disciplines or even between men and women, although one might be comparing apples and oranges here on other aspects of the data. The rink records were scraped from the Dutch Wikipedia page[2] that collects local records. Note that the use of records to estimate the speed of a rink is not flawless. First of all, records are continuously subject to improvement, such that the definition of relative time is sometimes problematic. Next, some venues rarely host international events, such that their records do not fairly represent the theoretical speed of the rink, and actually produce under-estimates of the relative times athletes set (meaning they appear faster than they are). In order to avoid constantly

[2]`https://nl.wikipedia.org/wiki/Lijst_van_snelste_ijsbanen_ter_wereld`

having to update the list of records and subsequent analyses, which is rather time-consuming, we choose to fix the records at a particular point in time. A minor side effect of this is that some newer results may actually have a relative time below 1.0. Our collected list of international rink records will be made available[3] as part of this publication.

6.3 Feature Construction by Aggregation

As explained in the introduction, the training data takes the form of a sequence of annotated events, corresponding to the individual exercises an athlete performs. While being valuable information, this sequential representation will require certain transformations in order to elicit general characteristics of an athlete's preparation for a race. Individual exercises will generally not play a deciding role in the outcome (unless of very extreme nature), and it is the combined nature of exercises that determines the effect of the training program. Therefore, some form of aggregation is required to draw out the various aspects of training that potentially play a role. Although there is a large body of knowledge about the effect of certain types of exercise in the sports physiology literature, it is still uncertain what aspects of training and preparation determine the variance in race results that still remains, as for example exemplified in Figure 6.1. For this reason, our feature construction approach will include a rather large collection of features, with the aim of including many angles and leaving room for discovery in later stages of the analysis. Furthermore, it is not quite clear how long the effect of specific exercises lasts for individual athletes, and thus what period prior to the test event should actually be included in the aggregation. Our set of constructed features will therefore involve time windows of various lengths, ranging from one day to several weeks.

In this paper, we will consider two general aggregation approaches, the first of which involves uniform aggregation over the various windows. The second based on convolution using a kernel that is based on the exponential decay function.

[3]`http://datamining.liacs.nl/rink-records.txt`

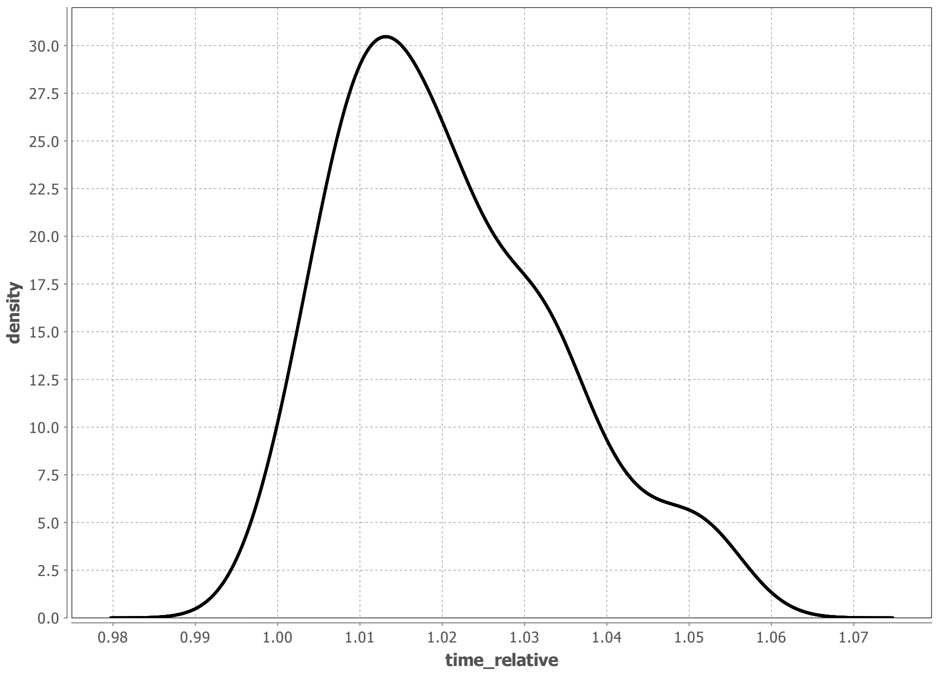


Figure 6.1: Distribution of relative times over 1000 m as estimated by kernel density estimation (based on 75 results, with a Gaussian kernel of $\sigma = 0.0051$). The probability density estimate continuing below 1.0 is an artefact of the KDE. The best time in the list is actually a rink record in the Hague (the Netherlands) at 1.0.

6.3.1 Uniform Aggregation

Before defining the notion of a (time) window, we first formalize the events to be aggregated, as they appear in the data of our application.

Definition 10 (Exercise)**.** *An* exercise *is defined as a tuple* $e = (t, ampm, dur, int, load)$*, where* t *is the date of the exercise,* $ampm$ *is a binary variable indicating the morning or afternoon session. Numeric values* dur*,* int*, and* $load$*, indicate the duration (in minutes), the subjective intensity (on a scale of 1 to 10), and the load (in intensity-minutes) of the exercise, respectively.*

The three crucial numeric attributes of an exercise specify the following:

- The *duration* simply specifies the length of the exercise. Durations tend to be rounded to quarters of hours (especially for longer exercises), but this is deliberate, and athletes generally adhere to the required duration.
- The *intensity* indicates how heavy the exercise was, as perceived by the

athlete, with 10 being the intensity of a race itself. During training, values of 9 or 10 are rare. Although intensity is a subjective measure, the athletes are very used to it, and will rate specific trainings fairly consistently.

- The *load* is defined as the product of duration and intensity, with the intention of capturing the total energy expenditure during the exercise. Although load is actually a derived attribute (it does not appear in our normalized database, for that reason), we include it in the definition of an exercise because it plays a crucial role both in the modeling as in the reasoning behind the training program[4].

Note that the races themselves also appear as exercises in the database, since it is crucial to include the training load produced by such intense events, when considering the preparation for other races. In speed skating, several races often take place in a single weekend, such that later races are influenced by earlier ones.
Definition 11 (Time Window)**.** *A* (time) window $w_{t,m}$ *is a set of exercises* $e_1, \ldots, e_n$*, such that all dates* $e_i.t$ *are before* t*, and not more than* $2m - 1$ *days before* t*:* $t - 2m + 1 \leq e_i.t < t$*.*

Note that day t itself is not part of the window. For reasons that will become clear in later sections, we have opted to define the length of a window in terms of its middle m, essentially indicating the 'centre of mass' of the window. A window $w_{t,1}$ will thus include the one day prior to t, $w_{t,2}$ indicates the three days prior to t, and so on.

For a window w, the following primitive aggregates will be considered:

Count Simply the number of exercises in w: $|w|$.

Sum(duration) The sum of durations of the exercises in w: $\sum_i e_i.dur$.

Sum(intensity) The sum of intensities of the exercises in w: $\sum_i e_i.int$.

Sum(load) The sum of loads of the exercises in w: $\sum_i e_i.load$.

Avg(duration) The average duration of the exercises in w: $\sum_i e_i.dur/|w|$.

Avg(intensity) The average intensity of the exercises in w: $\sum_i e_i.int/|w|$.

[4]Note that the definition in terms of a product of duration and intensity might be too simplistic, since neither duration nor intensity may be a linear scale. Doubling the length of an exercise may have an exaggerated effect if the intensity is (too) high, and doubling the intensity makes the exercise entirely different in nature, addressing different metabolic systems.

Avg(load) The average load of the exercises in w: $\sum_i e_i.load/|w|$.

Max(duration) The maximum duration of the exercises in w: $\max_i e_i.dur$.

Max(intensity) The maximum intensity of the exercises in w: $\max_i e_i.int$.

Max(load) The maximum load of the exercises in w: $\max_i e_i.load$.

Aggregation using the minimum was deemed senseless, since a very light training has little effect, and one could interpret daily rest periods as very light exercises anyway.

Specifiers

Each primitive aggregate listed above can be applied to all the exercises in a given window, or just to subsets of exercises from specific categories. These subsets are specified by what we will refer to as specifiers. We apply the following specifiers:

Morning/afternoon sessions By specifying *am*, *pm* or no specifier at all, the aggregate can include only the morning sessions, only the afternoon sessions, or all sessions, respectively. Note that during the winter, the coach will plan exercises on the ice in the morning, and alternative training in the afternoon, so distinguishing between those may be fruitful.

Intensity intervals Exercises at different intensities will trigger different parts of the system, and hence will produce a different training stimulus. As specifier, we optionally select only exercises within specific intensity intervals $[l, u]$, where $l \in [1, 10]$ and $u \in [l, 10]$.

Note that each type of specifier will introduce multiple variants of the primitive aggregates. For *ampm*, adding specifiers will raise the number of aggregates (per window size) from 10 to 30. For the intensity-intervals, there will be $1/2 \cdot 10 \cdot (10 + 1) = 55$ variants of each primitive. However they are only applied to the 4 primitive aggregates that do not involve intensity and load, producing $55 \cdot 4 = 220$ aggregates. In order to avoid combinatorial explosion of the aggregate collection, we do not include combinations of specifiers (such as low intensities in morning sessions). In total, there will be 250 aggregates per window.

Aggregation and Convolution

Observe that such uniform aggregation over a window can be seen as a form of convolution with a rectangular kernel [85]. The convolution of a time series $x(t)$ (in this case any of the training parameters that are aggregated) applies a kernel to the series to obtain a new series $y(t)$ as follows:

$$y(t) = h * x(t) = \sum_{i=-\infty}^{\infty} h(i)x(t-i)$$

Here, h refers to the kernel, which is required to sum to 1 over its domain. In the case of a uniform window, the kernel is essentially a rectangular function (remember that $2m-1$ is the length of the window):

$$h(t) = \begin{cases} 1/(2m-1) & \text{if } 0 \leq t \leq 2m-1 \\ 0 & \text{otherwise} \end{cases}$$

Since the rectangular kernel is zero over a large part of its domain, the convoluted function $y(t)$ can be computed much faster. In the next section, we will introduce a kernel that is both more natural and more expensive to compute.

6.3.2 The Fitness-Fatigue Model

Although uniform aggregation is intuitive and straightforward to implement (and as we will see, provides fairly good models), it is somewhat unnatural. First of all, it is unlikely that all exercises over a period of, say, four weeks will have the same influence on the level of fitness at a race. Rather, one would expect that exercises several weeks ago have a much smaller influence than more recent ones. Second, the hard distinction between an exercise 28 days ago, and one 29 days ago seems unnatural, and may introduce minor artefacts in the constructed features. Finally, there is a general pattern where the initial effect of an exercise is negative, while after a short period of rest and recuperation, the effect is positive. Ideally, the aggregated features should exhibit such behaviour.

In this section, we introduce a type of aggregation based on convolution with a more natural gradually progressing kernel. We will use a multi-component kernel that is taken from the physiology literature [16] and aims to model the complex way in which a human body responds to exercise by initial fatigue, followed by a slight improvement in performance, the effects of which die out gradually over the course of several week, returning to a state of fitness comparable to that prior to the exercise.

The core of this kernel is based on the *exponential decay* function, as follows:

$$h_e(m) = e^{-\lambda m}, \quad m \geq 0$$

The parameter λ here determines the speed with which this kernel decays towards zero, in other words, the speed with which the effects of exercise diminish over time. Although the exponential decay is defined in terms of λ (with unit s^{-1}), we will primarily define a specific kernel in terms of the parameter τ (in units s, or more conveniently, in days), which corresponds to the 'mean lifetime' of the kernel, and as such can be interpreted as the centre of mass of the kernel. This makes this parameter immediately comparable to parameter m of a uniform window, which is also the centre of mass of the kernel. The simple relationship between τ and λ is as follows:

$$\tau = 1/\lambda$$

The exponential decay function effectively models the diminishing positive effect of an exercise as time passes. However, it does not include the tiring effects of exercise in the few days after training, which may outweigh the positive influence of training. For this reason, [16] introduced the so-called *Fitness-Fatigue* model, which models these two effects as two components of a kernel with different weights and different decay factors, as follows:

$$h_2(m) = e^{-\lambda_{fit} m} - K e^{-\lambda_{fat} m}, \quad m \geq 0$$

λ_{fit} determines the speed with which the positive effects of training (the *fitness*) diminish, and typically corresponds to a τ_{fit} in the order of months, while λ_{fat} determines the shape of the fatigue curve immediately after the exercise. The associated τ_{fat} is typically in the range of two weeks. Initially, the influence of fatigue is about twice as big as that of the improved fitness (determined by the value of K).

The fitness in the above two-component model is assumed to be immediately improved after the exercise, which in practice is not the case. The desired adaptation in various metabolic systems will not take effect until several days after the exercise, such that the fitness will need to be modeled with an additional component [16], producing the following three-component kernel:

$$h_{ff}(m) = (e^{-\lambda_{fit} m} - e^{-\lambda_{del} m}) - K e^{-\lambda_{fat} m}, \quad m \geq 0$$

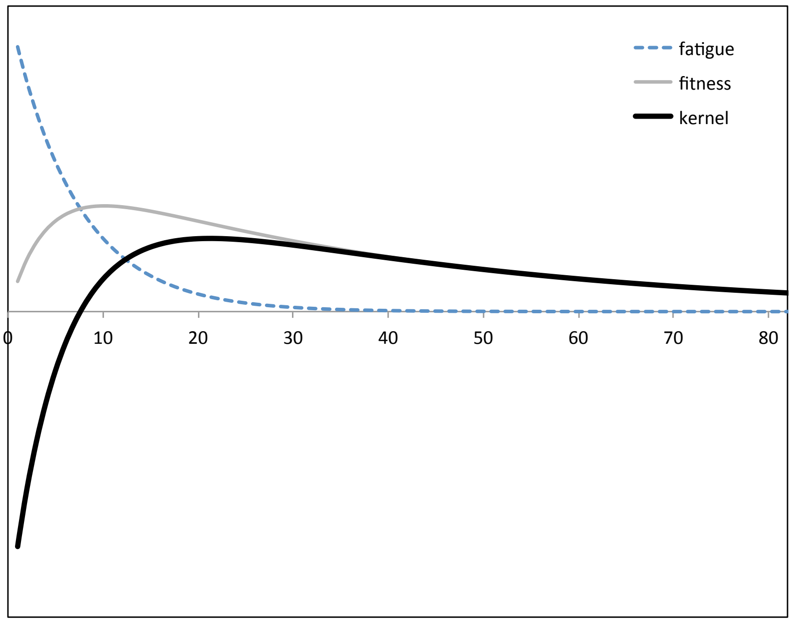


Figure 6.2: The three-component Fitness-Fatigue kernel (in solid black) as a function of time after the exercise (in days). The *fitness* and *fatigue* parts are also shown, in solid grey and dashed, respectively.

where λ_{del} affects the exponential function that reduces the initial fitness. In Figure 6.2, the combination of fitness and fatigue into this kernel is demonstrated. In [16], values are given for the associated parameters, obtained by fitting the convolved function to athletic data, producing the values below. Although these values seem reasonable, they will be athlete- and specialism-specific, such that we will fit these values to specific datasets collected, in the experimental section. The published values for the parameters are as follows:

$$\tau_{fit} = 50 \text{ days}, \quad \tau_{del} = 5 \text{ days}, \quad \tau_{fat} = 15 \text{ days}, \quad K = 2.0$$

6.4 Modeling Approaches

6.4.1 Regularized Linear Regression

In the previous section, we explained the procedures to build large sets of interpretable features about the training, that might be able to explain the target variables of performance. These target variables might be a linear function of a subset of the aggregate features, but we do not know which ones beforehand. In order to find a good subset of aggregate features for each target variable, we use LASSO [90], a method for estimating generalized linear models using convex penalties (l_1) to achieve sparse solutions [27].

Assume $\bar{t}$ is the mean of the target variable:

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

The coefficient of determination R^2 is now defined as:

$$R^2(t,f) = 1 - \frac{\sum_i (t_i - f_i)^2}{\sum_i (t_i - \bar{t})^2} \tag{6.1}$$

where $\sum_i (t_i - f_i)^2$ is the sum of squared differences between the actual and predicted target value, and $\sum_i (t_i - \bar{t})^2$ is the sum of squared differences between the target value and the constant function $t = \bar{t}$. Note that R^2 is between 0 and 1 whenever the model f is produced using the Ordinary Least Squares method, but may be lower than 0 for functions derived differently. R^2 is often interpreted as the *explained variance*, where a value of 0 means that no variance in the dependent variable can be explained by variance in the independent variable(s), and a value of 1 means that all variance can thus be explained (a perfect fit of the data).

6.4.2 Subgroup Discovery

The previous section focussed on linear models, assuming that the dependencies we hope to discover are indeed linear in nature. Unfortunately, in the domain we are focussing on, it is quite likely that the relationship between (extent of) training and performance is non-linear. When doing a certain training routine, it can be expected that the relationship is in fact curved, with peak performance being achieved at a certain optimal load on the human body. Doing too little will not achieve the right effect, but doing too much of the training also produces sub-optimal performance. Specifically, one can expect thresholds in the training load above (or below) which performance will rapidly diminish. Therefore, we will also experiment with modeling techniques that are more local in nature, and find subsets of the data where performance was surprisingly low, as well as finding variables and thresholds that will identify such sub-optimal subsets.

Our paradigm of choice for such (potentially) non-linear data is that of *Subgroup Discovery* [50, 54]. It is a data mining framework that aims to find interesting *subgroups* satisfying certain user-specified constraints. In this process, we explore a large search space to find subsets of the data that have a relatively high value for a user-defined quality measure. We consider constraints on attributes, and determine which records satisfy these constraints. These records then form a subgroup. The constraints on the attributes (the *description*) form an intensional specification of a part of our dataset, and the subgroup forms its extension (that is, an exhaustive enumeration of the members of the subgroup).

A number of papers in the literature discuss SD variants for regression tasks, which to some extent are applicable to our case. One group of techniques focusses on finding subgroups where the target shows a surprisingly high (or conversely, low) average value [30, 5, 78, 59]. Typical proposed quality measures use statistical tests to capture the level of deviation within the subgroup, often weighted by the size of the subgroup, for example the *mean test* or *z-score* [78],

$$\varphi_z(S) = \sqrt{|S|}\frac{\mu_S - \mu_0}{\sigma_S}$$

where μ_S and μ_0 stand for the subgroup and database means of the target, respectively, and σ_S denotes the standard deviation within the subgroup S. Other works consider the distribution of target values within the subgroup [41], and use statistical measures for assessing distributional differences.

In the majority of these quality measures, the interestingness is computed from the distribution of the subgroup alone, or when compared to that of the entire dataset. Here, we take a slightly different approach, and consider the subgroup description a dichotomy of the data, where both the distribution of the subgroup as well as of the *complement* play a role in determining the quality of the dichotomy. Therefore, we introduce a new quality measure for numeric targets in SD. This quality measure uses the well-known notion of R^2 to capture how well a subgroup and its corresponding complement describe the data, in comparison to the distribution of the entire dataset, so ignoring the dichotomy. Hence, we treat the subgroup as a model of the data, to be more specific a step function of two parts. The following two averages over the target t provide the constant prediction for, respectively, the subgroup and its complement:

$$\bar{t}_{subg} = \frac{1}{|S|}\sum_{i \in S} t_i$$

$$\bar{t}_{comp} = \frac{1}{n - |S|} \sum_{i \notin S} t_i$$

These two average values now lead to the following step function:

$$f_S(i) = \begin{cases} \bar{t}_{subg} & \text{if } i \in S \\ \bar{t}_{comp} & \text{if } i \notin S \end{cases}$$

The quality measure *Explained Variance* is now simply defined as follows:

$$\varphi_{EV}(S) = R^2(t, f_S)$$

This quality measure uses the definition of R^2 given in the previous section, which was independent of the nature of the model f. Note that by using this quality measure, we have a way of directly comparing the discovered subgroups (with corresponding step functions) to the linear models, which is a clear benefit over the traditional quality measures. We furthermore observe that the step functions, despite representing dichotomies, can be based on subgroups of multiple conditions. Therefore, the resulting step functions will be multi-dimensional (involving potentially multiple features). The quality measure introduced here was implemented in the Cortana Subgroup Discovery tool [66].

6.5 Experimental Results

In order to demonstrate the kinds of analyses and results of the proposed methods on actual data, and to test the benefits of individual techniques proposed above, we experiment with data from four athletes of the LottoNL-Jumbo team, two male and two female. All experiments were performed using three software components:

1. A relational database that organizes all the different datasets and meta-data concerning exercise and competition: the *Performance Sports Repository* (PSR).
2. A dashboard accessible over the Internet, that provides various views and visualizations of the data, and allows online aggregation and linear modeling of the data.
3. The generic Subgroup Discovery tool Cortana[5], which was extended

[5]Sources in Java and an executable of this tool can be downloaded from `http://datamining.liacs.nl/cortana.html`.

Table 6.1: Dataset details for competition results of four skaters.

	Gender	Distance	Competitions	Sessions
M1	Male	1000 m	75	2 758
M2	Male	500 m	142	2 930
F1	Female	1500 m	60	2 230
F2	Female	500 m	22	1 095

for this purpose with a direct database connection to the PSR, and the Explained Variance quality measure [66].

We will demonstrate the results on the datasets listed in Table 6.1, collected from four athletes. Next to competition data, we also have physical test data, for which we provide results for one of the athletes (M1 in the table below), for which we have 146 records.

We will generally describe three types of modeling of the data: 1) univariate models, either using a linear or a step function, where we rank all features by R^2, 2) multivariate models using LASSO, and 3) Subgroup Discovery using Cortana. Note that univariate step models can also be interpreted as subgroups with a single condition, such that results between settings 1 and 3 overlap to some extent. When mining for subgroups, we use beam search to a fairly shallow depth, typically to a maximum depth of three or less, depending on the experiment. When not further specified, the subgroups (or their step functions) presented involve a single feature ($d = 1$). The width of the beam is set to a default of $w = 100$ (candidates that proceed to the next level). For the numeric attributes, the Cortana setting 'best' is applied, which means that for each attribute, all numeric threshold are considered and the optimal split point is selected. The resulting locally optimal subgroup is added to the result list if of sufficiently quality, and conditionally added to the beam for further refinement.

Before considering more systematic experiments, for example, testing the merits of the Fitness-Fatigue model, we first present results for a single skater, and demonstrate the kinds of input given to the coach concerning possible changes to the training schedule.

6.5.1 Demonstration of results

This section discusses results for female skater F1 who specializes in the medium to long distances. We first focus on 1500 m races, for which we have 60 examples over a period of five years. The average relative time for this skater was 1.0391, so 3.91% above the track record. We have training details for the entire five years, such that we can aggregate the preparation for each of these races easily.

Uniform features We start with univariate results in the simplest setting: uniform aggregates without specifiers and simple linear models. The best-fitting aggregate that was found was `max_load_1` with the following model:

$$y = -0.000014x + 1.042$$

The explained variance of this model is a mere $R^2 = 0.0563$. The model starts with a reasonable intercept, and encourages a high load (the product of duration and intensity) on the day prior to the race. Although the effect is minor, this suggests that a peak right before a race (possibly due to another race in the same weekend) is beneficial. The step function associated with this aggregate function, with a threshold around 360, has a more pronounced $R^2 = 0.1233$. The two levels are $t_{rel} = 1.043$ for low maximum loads vs. 1.031.

The second-best aggregate is `avg_intensity_19`, with model

$$y = -0.003952x + 1.052$$

which suggests that the intensity should be kept high over a period of almost three weeks to improve race times. This aggregate actually scores highest on explained variance of the step function, with a minimum average intensity of 3.93 (low to moderate intensity). Lower intensities suggest an expected relative time of $t_{rel} = 1.044$, whereas higher values on average lead to relative times of $t_{rel} = 1.028$ ($R^2 = 0.2231$). For clarity, details of these three features are listed in Table 6.2.

Specifiers The addition of specifiers does a great deal to the quality of the univariate models. The following aggregate scores the best on R^2:

- `max_duration_int5_17` (the largest period spent at intensity 5 for the last 17 days prior to the race)

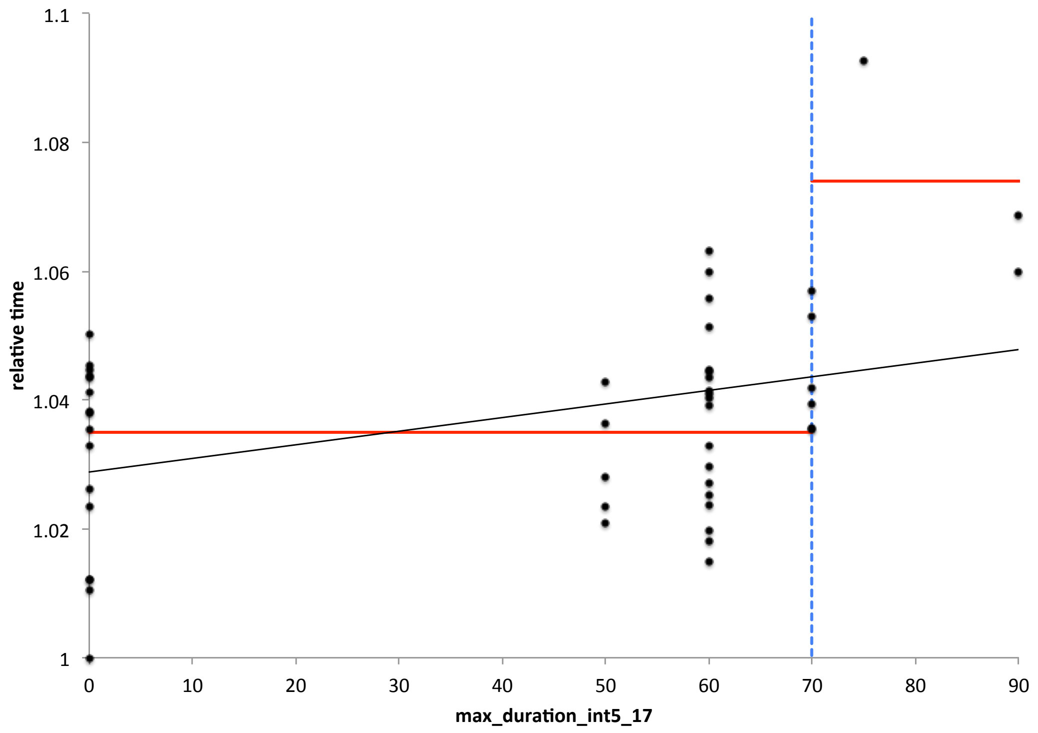


Figure 6.3: Graph showing the relation between `max_duration_int5_17` and the relative time of the subsequent race. The vertical dashed line indicates the threshold (inclusive to the left) and the solid horizontal lines the two average times for subgroup and complement. The black line with a slope indicates the best-fitting simple linear equation.

Exaggerating this type of training has a detrimental effect on the race outcome: longer durations of this specific training type lead to higher times. The step function ($R^2 = 0.3080$) caps this value at 70 minutes. With longer durations of intensity 5, relative times of $t_{rel} = 1.074$ are expected, compared to $t_{rel} = 1.035$ below this threshold (Fig. 6.3).

Fitness-Fatigue model Switching from uniform to FF features, we note that the following parameters provide the best (linear) aggregate, based on the sum of duration: $\tau_{fit} = 39, \tau_{del} = 4.0, \tau_{fat} = 7.0, K = 2.0$. The corresponding kernel is the one featured in Fig. 6.2 earlier in this paper. The associated explained variance is $R^2 = 0.1002$.

Multi-variate model The individual features presented so far do not lead to very well-fitting models, despite their role in informing the coach of ways to

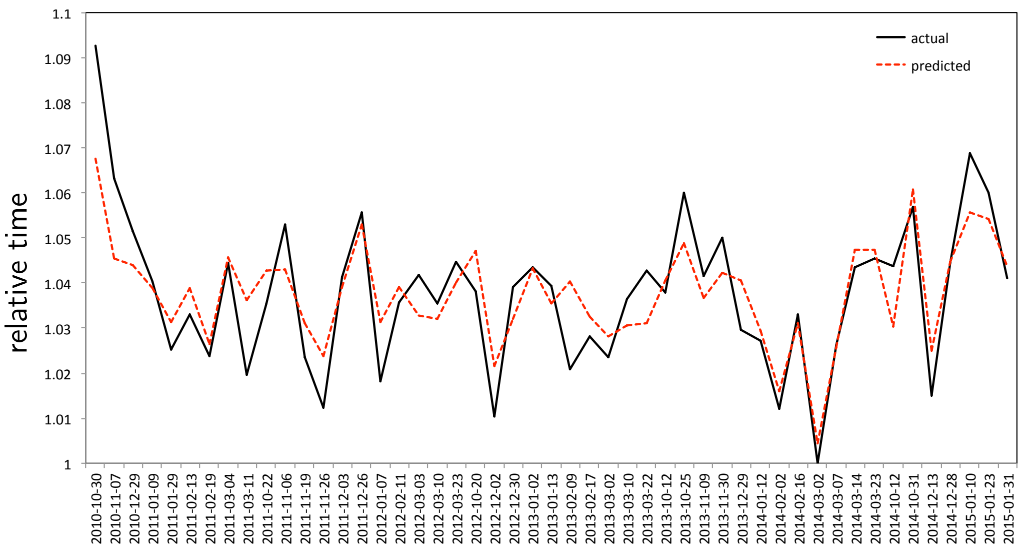


Figure 6.4: Obtained results (actual) over 60 historical races, compared to the predicted results of a relatively simple LASSO model based on the training sessions prior to each race.

Table 6.2: Overview of selected uniform features.

feature	linear model	R^2 linear	R^2 step	low	high
`max_load_1`	$-0.000014x + 1.042$	0.0563	0.1233	1.043	1.031
`avg_intensity_19`	$-0.003952x + 1.052$	0.0557	0.2231	1.044	1.028
`max_duration_int5_17`	$0.000211x + 1.029$	0.1491	0.3080	1.035	1.074

optimize the training and avoiding some pitfalls of under- and overtraining. More precise models can of course be obtained by involving multiple features. The graph in Fig. 6.4 presents the 60 results achieved by the skater, as well as the predicted times, by a multi-variate linear model. The model, induced by the LASSO procedure, involves 18 features, selected from the larger pool of uniform aggregates (ignoring the specifiers). The quality of the model is $R^2 = 0.721$, obtained on the training set[6]. Although such models (and more accurate models involving more complex aggregates) can be used to predict the outcome of an upcoming race, this particular prediction is of limited value. Rather, the model is more valuable from a knowledge discovery point of view, pointing to the features that matter most. In this case, the 18 features mostly include duration over short windows (one to five days), and intensities over longer windows (approx. two weeks).

[6]No distinction between training and test set was made for these experiments.

Subgroups In the above text, we have already presented the following three subgroups[7]:

- $\texttt{max\_load\_1} \leq 360$ ($R^2 = 0.1233$)
- $\texttt{avg\_intensity\_19} > 3.93$ ($R^2 = 0.2231$)
- $\texttt{max\_duration\_int5\_17} \leq 70$ ($R^2 = 0.3080$)

The first two subgroups relate to the experiment with just uniform aggregates, where the second subgroup is the optimal step function found at depth 1. The third subgroup relates to the experiment involving specifiers. While subgroups at depth 1 are interesting since they point to individual predictive features, they capture only shallow effects. We now present subgroups at greater depth, that indeed describe more complex concepts. The best subgroup found on the uniform windows (without specifiers) by Cortana at depth $d \leq 2$ is

$$\texttt{avg\_intensity\_20} > 3.94 \vee \texttt{sum\_duration\_2} > 170 \quad (R^2 = 0.4232)$$

Although Cortana produces subgroups as conjunctions of conditions, for reasons of presentation this was logically inverted[8] in the above subgroup. The subgroup, covering 17 cases, describes races with an average of 1.0299, compared to 1.0526 for the remainder. It specifies that whenever the average intensity over the last 20 days is too low, and the total duration of exercises over the last 2 days is also low, this has a negative effect on the race result. Note how the explained variance has almost doubled at $d \leq 2$. Adding a third feature to the subgroups only produced a marginal improvement, which is not uncommon in SD.

The addition of specifiers in combination with deeper subgroups produced slightly better results, with the top subgroup being as follows:

$$\texttt{avg\_duration\_int5\_17} < 60 \vee \texttt{sum\_duration\_int6789\_10} > 115 \quad (R^2 = 0.446)$$

Note that compared to the $d = 1$ result of $R^2 = 0.3080$, this is a reasonable improvement. The subgroup specifies that a lower duration of intensity 5 exercises over 17 day, or a higher duration of high-intensity exercises over 10 days, is good.

[7]Although subgroups are interpreted here as dichotomies, we present them as either lower or upper thresholds, such that cases meeting the condition(s) specified correspond to the faster races.

[8]The original subgroup discovered, $\texttt{avg\_intensity\_20} \leq 3.94 \wedge \texttt{sum\_duration\_2} \leq 170$, covers the complement.

Validation of results For the results above, one could wonder to what extent each result is statistically significant. For individual models, be it linear or step function, it is possible to compute a p-value that indicates to what extent the model might be due to chance. Such p-values will be reported in the detailed experiments in the next section. However, we should note that we are generating a substantial number of features, such that we are in fact testing multiple hypotheses. The best ranked result may thus appear to be significant, even though this is just a consequence of the many models considered. [23] presents a method for validating the results of an SD algorithm, by means of a *distribution of false discoveries*. This distribution is obtained by running the algorithm repeatedly on the data after swap-randomising the target attribute, thus capturing what maximum qualities can be obtained from random data (that resembles the original data). Using the distribution, it is possible to set a lower bound on the quality (in this case explained variance) as a function of the desired significance level α. Assuming a significance level $\alpha = 0.05$, this validation method produces a lower bound of $R^2_{min} = 0.2907$ for the uniform data without specifiers, searching for subgroups at depth $d = 1$. This means that our optimal subgroup

$$\texttt{avg\_intensity\_19} > 3.93 \ \ (R^2 = 0.2231)$$

is not actually significant at $\alpha = 0.05$. It is good to note that the lower bound produced by the swap randomization depends on the specific settings of the SD run. Specifically, if the extent of the search is bigger, more hypotheses will be tested, such that the lower bound will increase in order to account for the higher probability of finding a seemingly interesting subgroup by chance. When increasing the search depth to $d \leq 2$, the procedure produces a lower bound of $R^2_{min} = 0.4212$. This makes the earlier depth 2 result (without specifiers)

$$\texttt{avg\_intensity\_20} > 3.94 \lor \texttt{sum\_duration\_2} > 170 \ \ (R^2 = 0.4232)$$

just barely significant.

6.5.2 Fitness-Fatigue model

In this section, we analyse the specific merits of the Fitness-Fatigue model introduced in Sec. 6.3.2. We start by considering the four parameters the model involves, in order to fit the kernel to the specific physiological properties of the individual athlete. Rough values for the optimal setting were determined by informal experimentation, after which an extensive grid search was used

Table 6.3: Optimal parameters for the Fitness-Fatigue model for four speed skaters.

	Input context				Optimal parameters			
	Gender	Distance	Time	n	τ_{fit}	τ_{del}	τ_{fat}	K
M1	Male	1000 m	1.0207	75	13	2.0	2.0	2.00
M2	Male	500 m	1.0212	142	21	2.0	2.0	3.75
F1	Female	1500 m	1.0391	60	39	4.0	7.0	2.00
F2	Female	500 m	1.0691	22	29	4.0	4.0	2.00

to determine the optimal values for each athlete involved. These results are demonstrated in Table 6.3, for four athletes and their respective distance of speciality. The left columns indicate the gender of the athlete, the preferred distance, the average relative time, and the number of races n available. The remaining columns indicate the optimal values for $\tau_{fit}, \tau_{del}, \tau_{fat}$ and K. The three decay parameters are in days. As an optimization criterion, we select the R^2 of the (univariate) linear model of the best feature found.

In order to analyse the stability of these parameters, we selected the third athlete (the one with the most available data), and varied each parameter individually, fixing the remaining the parameters to the optimum found earlier. The R^2 of the best feature was recorded for each setting of the parameters. Fig. 6.5 demonstrates for each parameter how sensitive it is to change, in terms of quality of fit of the FF model. We note that all functions are very well-behaved and smooth over the domains considered, with the selected optimum clearly being undisputed. Furthermore, observe that the functions appear to be convex, making them fairly straightforward to optimize. Hence, the relatively simple grid search used in the pragmatic setting can be easily replaced by a more efficient hill-climber.

Let's consider the table of FF parameters in more detail. First of all, the rough numbers are very plausible from a physiological point of view. Clearly, the fatigue and (delayed) gain in fitness should be in the order of a few days, while the prolonged benefit of the exercise remains for a longer period in the order of several weeks. Also, the optimal values clearly differ per athlete, as a function of the different physiology and type of training the athlete is subjected to generally. Table 6.3 also suggests a difference between men and women, with men having a shorter time scale than women, both for the recuperation and how long the benefit lasts, although such conclusions are hard to draw from only four cases. Note also that the values reported

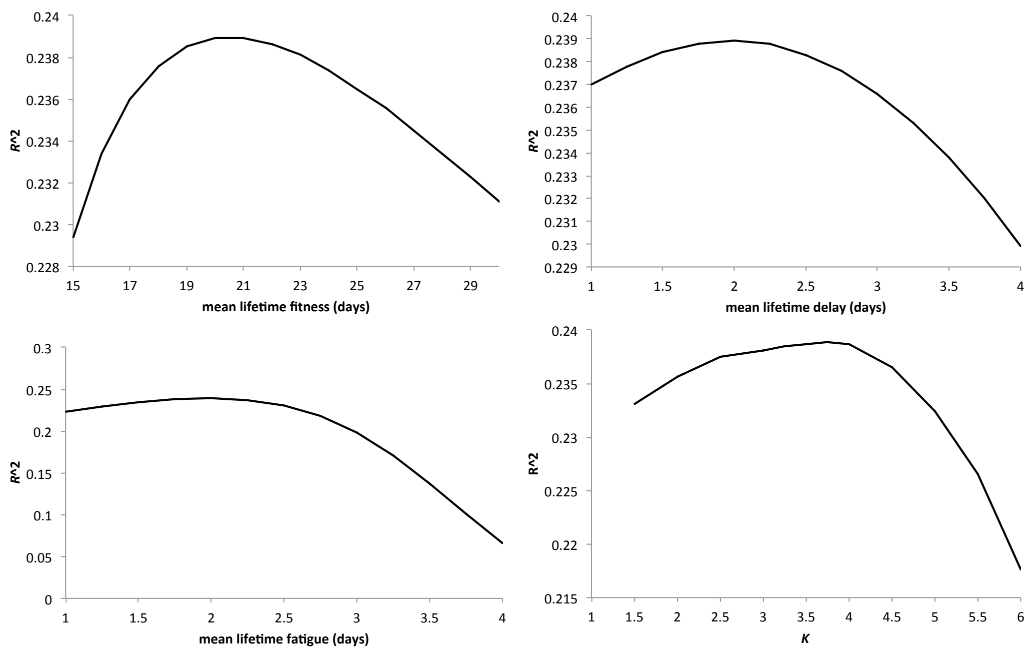


Figure 6.5: Analysis of sensitivity of the features to varying the four parameters $\tau_{fit}, \tau_{del}, \tau_{fat}$ and K (from top left to bottom right). Clearly, these functions are smooth and well-behaved.

here are somewhat different from the ones reported in [16], which are: $\tau_{fit} = 50, \tau_{del} = 5, \tau_{fat} = 15, K = 2.0$. As a last observation, we note that τ_{del} and τ_{fat} tend to assume very similar values. What impact this has from a physiological perspective (the fatigue and beneficial adaptation of the body go hand in hand?) is hard to say, but at least from a modeling perspective, it is a good opportunity to dispense with one parameter, and make the fitting of models more efficient.

Having a stable and physiologically plausible FF model of the training response, it is now time to turn to the question whether the model indeed produces a better fit, compared to our baseline of uniform aggregates. To this end, we again consider the explained variance R^2 of the linear model on the best feature found, first for uniform features, then for the exponentially decaying features. Since above, the FF model was optimized without specifiers (intensity zones and morning/afternoon distinction), we compare the results with a similar setting for the uniform features. Table 6.4 presents these results. The columns marked "R^2 linear" indicate the explained variance of the simple linear model. The indicated p-values for each result refer to the statistical significance of a linear regression t-test: the significance

Table 6.4: Comparison of goodness of fit of best univariate linear model for (middle) the uniform aggregates and (right) the Fitness-Fatigue model.

			Uniform		Fitness-Fatigue	
	Gender	Distance	R^2 linear	p-value	R^2 linear	p-value
M1	Male	1000 m	**0.41**	4.16×10^{-10}	0.30	3.85×10^{-7}
M2	Male	500 m	0.17	2.46×10^{-7}	**0.23**	7.02×10^{-10}
F1	Female	1500 m	0.06	9.95×10^{-3}	**0.11**	1.69×10^{-3}
F2	Female	500 m	**0.54**	9.53×10^{-5}	0.50	2.58×10^{-5}

of the best model, testing the hypothesis that the coefficient of the model is not 0 (in other words, testing whether the dependent variable is indeed influenced by the independent variable).

Based on these numbers, there is clearly not a consistent benefit of the FF model, over the less natural uniform features. Especially in the case of the first athlete, the uniform features are in fact more accurate. The feature in question (although there are multiple variants of similar score) concerns the sum of the duration over 9 days, which is an indication of too intense training and hence too high levels of fatigue as a result. Also for the fourth athlete, the uniform features come out on top. Still, for individual athletes the FF model may show a considerable improvement in fit over the unnatural uniform features.

Alternative aggregate functions The presentation of the FF model in terms of convolution translates into SQL as the `SUM` aggregate function. If features based on `SUM` have a potential benefit, so might the alternative functions `AVG` and `MAX`. We present an additional experiment in Table 6.5 that investigates the added value of these two aggregates to the plain implementation of the FF model used so far. The last two columns of the table show that in two cases, `AVG` or `MAX` do outperform the standard convolution. This is interesting, since there is clearly the potential to improve the models in this way. However, the features are less intuitive to understand since they are not based on the standard definition of convolution in terms of summation.

Table 6.5: Analysis of the benefit of adding `AVG` and `MAX` as aggregates to the Fitness-Fatigue model.

			SUM		SUM, AVG, MAX	
	Gender	Distance	R^2 linear	p-value	R^2 linear	p-value
M1	Male	1000 m	0.30	3.85×10^{-7}	0.30	3.85×10^{-7}
M2	Male	500 m	0.24	7.02×10^{-10}	**0.28**	1.60×10^{-11}
F1	Female	1500 m	0.11	1.69×10^{-3}	**0.15**	5.22×10^{-4}
F2	Female	500 m	0.50	2.58×10^{-5}	0.50	2.58×10^{-5}

6.6 Conclusion

We have presented a general approach to the modeling of training data in elite sports, with a specific application to speed skating in the LottoNL-Jumbo team. Our approach computes the combined effect of a training schedule by aggregating details of the individual training sessions, and thus capturing a considerable number of aspects of training and how one prepares for important test moments, such as physical tests and races. Since it is not entirely clear from the literature what aspects of training contribute the most, and what parameters individual athletes need to tweak in order to optimize the training to their specific physique and specialization, we produce a reasonably large collection of promising features. The most relevant features are then selected by a number of techniques, specifically univariate linear regression, the LASSO regression process and Subgroup Discovery. The linear modeling methods assume that the dependencies of interest are indeed monotonic and linear, that is, adding more load to the exercise will increase the (long-term) fitness of the athlete's body. Clearly, this is not generally the case, and one would expect there to be certain thresholds, above which training is no longer beneficial. For each aspect of training, there is an optimal volume, above or below which training is ineffective. This suggests that non-linear models, or models that are able to represent thresholds (such as subgroups) will outperform linear models.

As mentioned in our introduction, we aim to discover interpretable and actionable patterns in the data, such that the coach can immediately incorporate the most significant findings in the preparation for upcoming events, as well as in future training schedules. We believe that our presented approach, that deliberately presents simple results, and gives clear guidelines and boundaries on training load, makes this possible. In fact, individual findings on the athletes of the team have led to (subtle) modifications in training regimens, most notably where sprinters were sometimes subjected to too much aerobic exercise. It is good to stress again the athlete-specific nature of our analyses. Luckily, for a reasonable number of skaters, we have a long enough history to have a substantial database of training-response examples, where natural variation in preparation has produced a productive dataset. Athlete-specific data leads to athlete-specific findings, and one should therefore not interpret any discovered pattern as a general rule of exercise physiology, but rather as an opportunity to optimize training for that athlete.

We have presented a number of anecdotal results for a specific skater, demonstrating that interpretable and actionable results can be found. The best-fitting subgroup suggests that for a good result, this skater should avoid longer exercises at intensity 5 (over a longer window), as well as (slightly) increase the exposure to intensities 6 to 9. Although separate results appear very significant, a more thorough analysis using swap-randomization is necessary to account for the many features and models being considered. For this specific skater, statistically significant results could be obtained, despite there only being 60 available races. With up to 142 race results, other skaters will allow for much more significant findings.

The Fitness-Fatigue model, introduced as a more natural way to aggregate training impulses over time, produced reasonable results. After experimenting with four different skaters, two male and two female, very consistent and realistic values were found for the four parameters of this model. Although slight variations did occur, most notably between the male and female skaters, the general picture did match that of the coach. Knowing these (athlete-specific) parameters in detail allows the coach to mix exercise and recuperation in a more precise manner. From a modeling point of view, we also demonstrated that the optimal values of these parameters can be found efficiently, due to their well-behaved nature.

In a number of detailed experiments, we compared different choices in our modeling approach. A first experiment compared the uniform window to the Fitness-Fatigue kernel. Given the unnatural nature of a rectangular kernel, one would expect the FF model to be superior. Somewhat surprisingly, our data did not support this hypothesis. The FF model did indeed produce superior models for two athletes, but the uniform window performed equally superior on the remaining two, leaving this comparison unresolved.

Finally, we considered an experiment whether using the aggregate functions `MAX` and `AVG`, alongside the more obvious `SUM`, would be beneficial. This was indeed sometimes the case, although not by a large margin. Whether such slightly more accurate models are in fact attractive is questionable, since the combination with the non-trivial FF kernel does not lead to very interpretable patterns.

Chapter 7

Conclusions

In this thesis, we addressed data mining methods, tools and applications for multivariate time series. The sequential nature of time series imposes specific algorithmic solutions to address this type of data. Additionally, this thesis narrows the focus to *multivariate* time series. The multivariate aspect of the data reflects the complexity and multi-faceted nature of the system under investigation. Whether one is measuring infrastructures, athletic performance, monitoring human activities or analysing life style, there are numerous aspects that can be measured. This increasing growth of data frequency and sources stimulate this data centric era. But with new opportunities also come several challenges from the perspective of the methods and tools used.

In this chapter, we reflect on our main contributions to meet some of the knowledge discovery challenges surrounding multivariate time series. We can separate our contributions into two sides: unsupervised and supervised learning. Next to these two paradigms, we focus on a group of three aspects of data science: methods, tools and applications, which for lack of a better term, we refer to as the data science *triad*. Aside from these contributions, we take the opportunity to reflect on two research directions in this era of data science. Firstly, machine learning as an optimization process in two directions: better data representation and better model learning algorithms. Secondly, data science as a paradigm shift of scientific process: scarcity of data versus big data.

Unsupervised Multivariate Time Series In Chapter 2, we introduced a data mining method to find biclusters in multivariate time series, which addresses research questions **Q1** and **Q2**. This task addresses the situation of a system that has some recurring patterns over time in a subset of variables. In contrast with traditional biclustering algorithms, our method is able to find significant periods of time (larger sequences) where multiple variables deviate in a coherent manner. By coherent in this context, we mean that for each selected variable all segments have similar probability distributions.

We argue that framing this pattern recognition challenge as a biclustering task offers considerable benefits. Firstly, pattern recognition tasks in time series have been focusing primarily on the univariate scenario and this neglects relationships between variables. Take as an example the monetary exchange market. We know that the exchange rate of different currencies are connected, but which currencies are strongly related to the Brazilian real and which are related to the Singapore dollar? Furthermore, are these relations always present, or are they triggered by specific events? How long do these relationships last? The same analogy holds for almost any system measured over time. We claim that biclustering multivariate time series can playing an important role in finding such patterns.

Supervised Multivariate Time Series In order to tackle research questions **Q3** and **Q4**, in Chapter 4 we introduced Accordion, a greedy search algorithm that produces good aggregate features, both for regression and classification. With such features, one has a better data representation and this leads to better models. In the supervised learning setting, Accordion is a wrapper algorithm that integrates the feature construction and selection into the learning process. Our method differs from the common practice of considering feature construction as an isolated pre-processing step. We demonstrated the positive effects of searching for good aggregate features automatically by optimizing the selection of three components: time series variable, aggregate function and window size.

Our method automates a feature construction and selection processes, combining multivariate time series with mixed sampling rates. Normally such optimization processes come at the expense of producing features which are not interpretable. We decided to focus on the automatic construction and selection of interpretable aggregate features. By interpretable, we mean the combinations of input variables, aggregate functions and different window sizes. Our optimization process, for each combination of variable and aggregation function, expands and contracts the window size, capturing phenomena that work on different time scales. We believe this method to be highly promising for further development and implementation in efficient computation architectures, and better exploration of functional properties of

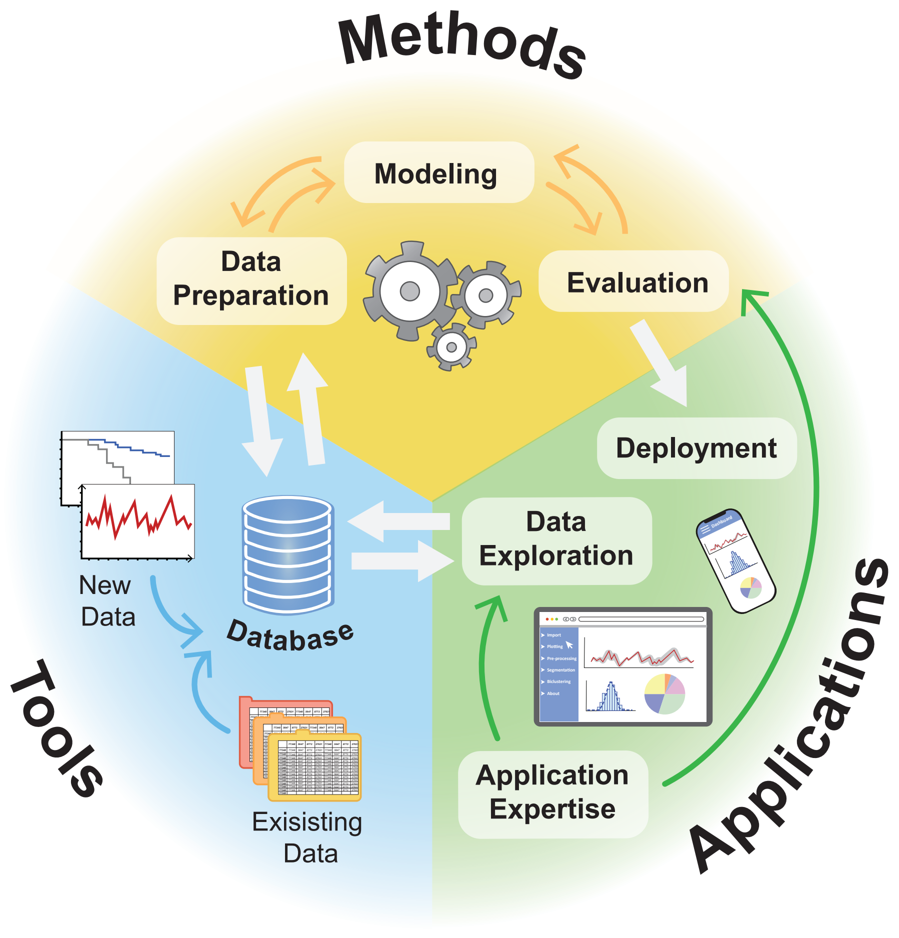


Figure 7.1: The triad of data science projects.

different aggregations. This will allow us to get faster and better results in larger search spaces.

Data Science Triad In addition to the machine learning methods introduced in Chapters 2 and 4, we observe that such methods stand at the intersection between tools and applications. In fact, data science can be seen as the integration of methods, tools and applications. Thus, relevant data science is the implementation of relevant methods, tailored to a specific application, while using the appropriate tools.

We illustrate the data science triad in Figure 7.1, by representing a generic scheme of what a data science project normally entails. Unlike other fields such as data mining, data science is finding an equilibrium between designing generalized machine learning methods and focusing on specific applications with efficient tools. Examples of such relationships can also be found in this thesis. For each of the methods (Chapters 2 and 4) there are corresponding easy to access and intuitive tools in Chapters 3 and 5, respectively. Such tools address research question **Q6**. Additionally, Chapter 6 focuses on a specific application of analyzing and improving the performance of elite speed skating athletes (research question **Q5**). This is done by using tailored features and models for each athlete and a relational database designed for elite sports performance monitoring.

Machine Learning: Optimizing Two Complementary Directions
The machine learning process is typically characterized as by being a process of exploration and exploitation of the data. Thus, names for the field such as data mining become intuitive to understand. Maybe the best explanation for this tendency has to do with the uncountable data sources and numerous data structures that exist. Here one could make an analogy with the proverb of Muhammad and the mountain, the machine learning algorithms being Muhammad and the mountain being the data (big data). It makes sense to see the challenge of creating an algorithm that is able to deal with different mountains, an algorithm that is adaptable to different datasets, reliable in its findings and fast in the construction of a new model.

An alternative to the view above is to have an algorithm that is able to describe properly the data. The process of improving a model outcome can be solved with good representations of the decision space. This space is described with the input data or transformations of it. These variable transformation we normally refer to as feature construction, extraction and selection. In the light of Muhammad and the mountain, instead of the mountain itself, maybe the model can be improved by knowing the height of the mountain, the soil properties, a vast photo gallery of the mountain from different perspectives and exposure. If only one could define an algorithm that searches for such descriptions automatically, maybe linear models could solve the majority of modeling challenges.

The experimental results in Chapter 4 indicate that there are significant gains to be had from focusing on the data representation. In the case of this thesis, the focus was on improving decision trees and linear regression models. Probably many more well-known algorithms can benefit from a layer of data transformation to reach the appropriate representation of the input space.

In fact, some years back, it would have been difficult to motivate such an algorithm for a task that is commonly seen as a pre-processing task. Presently, with developments in fields such as convolutional neural networks, motivating automatic data transformation has become more acceptable. As it seems with the advances in convolutional neural networks, these two optimization directions are actually perfectly complementary.

Data Science as a Scientific Paradigm Shift Data science being a discipline at the intersection of multiple other disciplines, it brings together multiple empirical sciences, which depend on observations to draw founded conclusions. In order to avoid drawing incorrect conclusions from observations, traditionally *hypothesis testing* is put forward as an essential, if not the only, scientific paradigm. With the current wealth of data, the field of data science is put at the center of a great scientific discussion. Is hypothesis testing presently still a sufficient scientific paradigm for research?

Hypothesis testing has been at the core of empirical sciences. This importance is due to two main reasons: data was scarce and costly to gather, and starting from a fixed, prior hypothesis is traditionally how to determine causality. As a start, in this period of data abundance, we are often analyzing the whole population instead of only a tiny sample. Concepts such as big data could trigger a deeper discussion about our present capacity for inference. Secondly, hypothesis testing is still considered the golden standard to determine causality. But many new developments in data science, for example *causal inference* [76] are demonstrating that there are in fact ways to avoid the *post hoc ergo propter hoc* fallacies. This makes the analysis of observational data, the predominant setting in data science, a very acceptable approach for scientific discovery.

So how could a research project be designed according to this new scientific paradigm? Take the speed skating example of Chapter 6. The central question there is how to increase the performance of elite athletes and win more medals. Following the data science rationale, we need to decide on what all can be monitored that can conceivably influence the performance of the athlete. One could measure training sessions, daily state of mind, eating logs, sleeping habits, metabolic variables, you name it. What and how to measure is no longer governed by preconceived ideas about what is expected to be the principle driver of athletic performance (the 'hypothesis'), but rather by technological, ethical and pragmatic considerations. If it can be practically, ethically and realistically measured, let's just include it in the analysis. After data collection, machine learning will consider numerous promising hypotheses in an exploratory manner, followed by a host of robust evaluation methods and experts to validate any scientific discoveries.

Bibliography

[1] https://www.universiteitleiden.nl/nieuws/2016/10/leidse-onderzoekers-winnen-publieksprijs-slimste-project-van-nederland.

[2] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering — a decade review. *Information Systems*, 53:16–38, 2015.

[3] E. Angelini, J. Henry, and M. Marcellino. Interpolation and backdating with a large information set. *Journal of Economic Dynamics and Control*, 30(12):2693–2724, Dec. 2006.

[4] M. Armesto. Forecasting with mixed frequencies. *Federal Reserve Bank of St. Louis*, pages 521–536, 2010.

[5] M. Atzmüller and F. Lemmerich. Fast subgroup discovery for continuous target concepts. In *ISMIS, the International Symposium on Methodologies for Intelligent Systems*, pages 35–44, 2009.

[6] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. *Pervasive Computing*, pages 1–17, 2004.

[7] S. Barkow, S. Bleuler, A. Prelić, P. Zimmermann, and E. Zitzler. Bicat: A biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, May 2006.

[8] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter*, 11(1):26–31, Nov. 2009.

[9] A. Bifet. *Adaptive Learning and Mining for Data Streams and Frequent Patterns*. PhD thesis, Universitat Politècnica de Catalunya, 2009.

[10] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[11] W. Bouten, E. W. Baaij, J. Shamoun-Baranes, and K. C. Camphuysen. A flexible gps tracking system for studying bird behaviour at multiple scales. *Journal of Ornithology*, 154(2):571–580, 2013.

[12] A. Brush, J. Krumm, J. Scott, and T. Saponas. Recognizing Activities from Mobile Sensor Data: Challenges and Opportunities. In *Proceedings Ubicomp' 11*, 2011.

[13] R. Cachucho, K. Liu, S. Nijssen, and A. Knobbe. Bipeline: A web-based visualization tool for biclustering of multivariate time series. In B. Berendt, B. Bringmann, É. Fromont, G. Garriga, P. Miettinen, N. Tatti, and V. Tresp, editors, *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III*, pages 12–16, Cham, 2016. Springer International Publishing.

[14] R. Cachucho, M. Meeng, U. Vespier, S. Nijssen, and A. Knobbe. Mining multivariate time series with mixed sampling rates. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 413–423, New York, NY, USA, 2014. ACM.

[15] R. Cachucho, S. Nijssen, and A. Knobbe. Biclustering multivariate time series. In *Intelligent Data Analysis*, 2017.

[16] T. Calvert, E. Banister, M. Savage, and T. Bach. A systems model of the effects of training on physical performance. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(2):94–102, 1976.

[17] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2016. R package version 0.13.1.

[18] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. shiny: Web application framework for r [computer software]. *URL http://CRAN. R-project. org/package= shiny (R package version 1.0. 0)*, 2017.

[19] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[20] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM.

[21] G. Creasey, P. Jarvis, and L. Berk. Play and social competence. In O. N. Saracho and B. Spodek, editors, *Early Childhood Education: Inquiries and Insights. Multiple Perspectives on Play in Early Childhood Education*. State University of New York Press, New York, 1998.

[22] M. de Winter. *A Missing Value Ignoring Approach for Whole Time Series Clustering of Longevity Corebody Temperature Data*. Master's thesis, Leiden University, Leiden Institute of Advance Computer Science, 2015.

[23] W. Duivesteijn and A. Knobbe. Exploiting false discoveries – statistical validation of patterns and quality measures in subgroup discovery. In *ICDM, the International Conference on Data Mining*, pages 151–160, 2011.

[24] P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, Dec. 2012.

[25] D. Figo, P. C. Diniz, D. R. Ferreira, and J. a. M. P. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, Mar. 2010.

[26] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 2010.

[27] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[28] E. Ghysels, P. Santa-Clara, and R. Valkanov. Predicting volatility: getting the most out of return data sampled at different frequencies. *Journal of Econometrics*, 2006.

[29] J. P. Gonçalves, S. C. Madeira, and A. L. Oliveira. Biggests: integrated environment for biclustering analysis of time series gene expression data. *BMC Research Notes*, 2(1):124, 2009.

[30] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data Mining and Knowledge Discovery*, 19(2):210–226, 2009.

[31] D. Gujarati. *Basic econometrics*. McGraw Hill, fourth edition, 2003.

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, Nov. 2009.

[33] T. Hastie, R. Tibshirany, and J. Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction.* Springer, second edition, 2009.

[34] J. Heinrich, R. Seifert, M. Burch, and D. Weiskopf. Bicluster viewer: a visualization tool for analyzing gene expression data. In *International Symposium on Visual Computing*, pages 641–652. Springer, 2011.

[35] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. Toivonen. Time series segmentation for context recognition in mobile devices. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 203–210. IEEE, 2001.

[36] J. A. Hobson. Rem sleep and dreaming: towards a theory of protoconsciousness. *Nature Reviews Neuroscience*, 10(11):803–813, 2009.

[37] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 159–163. ACM, 2005.

[38] IBM Corporation. *IBM SPSS Statistics.*

[39] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning*, pages 121–129, 1994.

[40] S. Jongstra, L. W. Wijsman, R. Cachucho, M. P. Hoevenaar-Blom, S. P. Mooijaart, and E. Richard. Cognitive testing in people at increased risk of dementia using a smartphone app: The ivitality proof-of-principle study. *JMIR mHealth and uHealth*, 5(5):e68, 2017.

[41] A. Jorge, P. Azevedo, and F. Pereira. Distribution rules with numeric attributes of interest. In *PKDD, the European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 247–258, 2006.

[42] S. Kaiser and F. Leisch. A toolbox for bicluster analysis in r. In *UseR*, 2008.

[43] S. Kaiser and F. Leisch. A generalized motif bicluster algorithm. In *UseR*, 2009.

[44] S. Kaiser, R. Santamaria, T. Khamiakova, M. Sill, R. Theron, L. Quintales, F. Leisch, and E. D. Troyer. *biclust: BiCluster Algorithms*, 2015. R package version 1.2.0.

[45] W. Kenney, J. Wilmore, and D. Costill. *Physiology of Sport and Exercise*. Human Kinetics, 2015.

[46] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[47] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[48] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, pages 349–371, 2002.

[49] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[50] W. Klösgen. *Subgroup Discovery, Handbook of Data Mining and Knowledge Discovery*, chapter 16.3. Oxford University Press, 2002.

[51] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray cancer data: Co-clustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[52] A. Knobbe, H. Blockeel, and A. Koopman. InfraWatch: Data management of large systems for monitoring infrastructural performance. In *Intelligent Data Analysis*, pages 91–102, 2010.

[53] A. Knobbe, J. Orie, N. Hofman, B. van der Burgh, and R. Cachucho. Sports analytics for professional speed skating. *Data Mining and Knowledge Discovery*, 31(6):1872–1902, Nov 2017.

[54] P. Kralj Novak, N. Lavrač, and G. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.

[55] M. Krätzig. The software jmulti. *Applied Time Series Econometrics, Cambridge University Press, Cambridge*, pages 289–299, 2004.

[56] M. Kuhn. Caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[57] V. Kuzin, M. Marcellino, and C. Schumacher. MIDAS vs. mixed-

frequency VAR: Nowcasting GDP in the euro area. *International Journal of Forecasting*, 27(2):529–542, Apr. 2011.

[58] J. Kwapisz, G. Weiss, and S. Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 2011.

[59] F. Lemmerich, M. Atzmueller, and F. Puppe. Fast exhaustive subgroup discovery with numerical target concepts. *Data Mining and Knowledge Discovery*, pages 1–52, 2015.

[60] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03*, page 2, 2003.

[61] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, Jan. 2004.

[62] S. C. Madeira and A. L. Oliveira. A linear time biclustering algorithm for time series gene expression data. In R. Casadio and G. Myers, editors, *Algorithms in Bioinformatics: 5th International Workshop, WABI 2005, Mallorca, Spain, October 3-6, 2005. Proceedings*, pages 39–52, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[63] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting methods and applications.* John Wiley & Sons, 1998.

[64] W. McArdle, F. Katch, and V. Katch. *Exercise Physiology: nutrition, energy, and human performance.* Lippincott, Williams & Wilkins, 2014.

[65] M. Meeng, R. Cachucho, and A. Knobbe. Quickie: Quick user interface for convolution kernel-involving experiments. `http://liacs.leidenuniv.nl/~meengm/doc/index.html`. Accessed: 2017-12-07.

[66] M. Meeng and A. Knobbe. Flexible enrichment with Cortana – software demo. In *BeneLearn, the Annual Belgian-Dutch Conference on Machine Learning*, pages 117–119, 2011.

[67] S. Miao, U. Vespier, R. Cachucho, M. Meeng, and A. Knobbe. Predefined pattern detection in large time series. *Information Sciences*, 329:950–964, 2016.

[68] S. Miao, U. Vespier, J. Vanschoren, A. Knobbe, and R. Cachucho. Modeling sensor dependencies between multiple sensor types. In *Proceedings of BeneLearn*, 2013.

[69] E. Miluzzo, N. Lane, and K. Fodor. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In *Proceedings of Embedded network sensor systems*, pages 337–350, 2008.

[70] D. Minnen, C. Isbell, I. Essa, and T. Starner. Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 601–606, Washington, DC, USA, 2007. IEEE Computer Society.

[71] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 473–484, 2009.

[72] T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pac. Symp. Biocomput*, pages 77–88, 2003.

[73] J. Orie, N. Hofman, J. de Koning, and C. Foster. Thirty-eight years of training distribution in olympic speed skaters. *Int J Sports Physiol Perform*, 9:93–9, 2014.

[74] S. Paraschiakos. *Comparing Sensor Networks for Activity Recognition.* Master's thesis, Leiden University, Leiden Institute of Advance Computer Science, 2017.

[75] J.-g. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie. Online pose classification and walking speed estimation using handheld devices. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, page 10, 2012.

[76] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2nd edition edition, 2009.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[78] B. Pieters, A. Knobbe, and S. Džeroski. Subgroup discovery in ranked data, with an application to gene set enrichment. In *Preference Learning 2010 at ECML PKDD, the European Conference on Ma-*

chine Learning and Principles and Practice of Knowledge Discovery in Databases, 2010.

[79] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006.

[80] J. Quinlan. Induction of decision trees. *Machine learning*, pages 81–106, 1986.

[81] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[82] R. Santamaría, R. Therón, and L. Quintales. Bicoverlapper: A tool for bicluster visualization. *Bioinformatics*, 24(9):1212, 2008.

[83] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[84] V. Stodden. *Model selection when the number of variables exceeds the number of observations.* Phd thesis, Standford University, 2006.

[85] D. Stranneby and W. Walker. *Digital Signal Processing and Applications.* Elsevier, 2004.

[86] M. Sugiyama, T. Kanamori, T. Suzuki, M. C. d. Plessis, S. Liu, and I. Takeuchi. Density-difference estimation. *Neural Comput.*, 25(10):2734–2775, Oct. 2013.

[87] J. Sun, S. Papadimitriou, and P. S. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 1076–1080, Washington, DC, USA, 2006. IEEE Computer Society.

[88] Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of time-series motif from multi-dimensional data based on mdl principle. *Mach. Learn.*, 58(2-3):269–300, Feb. 2005.

[89] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58, 1994.

[90] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[91] H. Turner, T. Bailey, and W. Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics & Data Analysis*, 48(2):235–254, 2005.

[92] D. United Nations. World population ageing: 1950-2050. Technical report, New York, NY, USA, 2002.

[93] A. Vahdatpour, N. Amini, and M. Sarrafzadeh. Toward unsupervised activity discovery using multi-dimensional motif detection in time series. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1261–1266, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[94] O. van de Rest, B. A. Schutte, J. Deelen, S. A. Stassen, E. B. van den Akker, D. van Heemst, P. Dibbets-Schneider, et al. Metabolic effects of a 13-weeks lifestyle intervention in older adults: The growing old together study. *Aging (Albany NY)*, 8(1):111, 2016.

[95] O. van de Rest, B. A. Schutte, J. Deelen, S. A. Stassen, E. B. van den Akker, D. van Heemst, P. Dibbets-Schneider, et al. Metabolic effects of a 13-weeks lifestyle intervention in older adults: The growing old together study. *Aging (Albany NY)*, 8(1):111, 2016.

[96] J. Van Hoof, H. Kort, P. Rutten, and M. Duijnstee. Ageing-in-place with the use of ambient intelligence technology: Perspectives of older users. *International journal of medical informatics*, 80(5):310–331, 2011.

[97] D. Vanderkam, J. Allaire, J. Owen, D. Gromer, P. Shevtsov, and B. Thieurmel. *dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library*, 2016. R package version 0.8.

[98] J. Vanschoren, U. Vespier, S. Miao, M. Meeng, R. Cachucho, and A. Knobbe. Large-scale sensor network analysis: applications in structural health monitoring. In *Big Data Management, Technologies, and Applications*, pages 314–347. IGI Global, 2014.

[99] G. Veiga, L. Ketelaar, W. De Leng, R. Cachucho, J. N. Kok, A. Knobbe, C. Neto, and C. Rieffe. Alone at the playground. *European Journal of Developmental Psychology*, 14(1):44–61, 2017.

[100] G. Veiga, W. Leng, R. Cachucho, L. Ketelaar, J. N. Kok, A. Knobbe, C. Neto, and C. Rieffe. Social competence at the playground: Preschoolers during recess. *Infant and Child Development*, 26(1), 2017.

[101] U. Vespier. *Mining sensor data from complex systems.* PhD thesis, Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University, 2015.

[102] U. Vespier, A. Knobbe, S. Nijssen, and J. Vanschoren. MDL-Based Analysis of Time Series at Multiple Time-Scales. In *Proceedings of ECML PKDD '12*, 2012.

[103] U. Vespier, S. Nijssen, and A. Knobbe. Mining characteristic multi-scale motifs in sensor-based time series. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2393–2398, New York, NY, USA, 2013. ACM.

[104] R. Westendorp and et. al. Nonagenarian siblings and their offspring display lower risk of mortality and morbidity than sporadic nonagenarians: The leiden longevity study. *Journal of the American Geriatrics Society*, 57(9):1634–1637, 2009.

[105] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 7. MIT press Cambridge, MA, 1949.

[106] L. W. Wijsman, E. Richard, R. Cachucho, A. J. de Craen, S. Jongstra, and S. P. Mooijaart. Evaluation of the use of home blood pressure measurement using mobile phone-assisted technology: the ivitality proof-of-principle study. *JMIR mHealth and uHealth*, 4(2), 2016.

[107] R. Wirth and J. Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pages 29–39, 2000.

[108] Y. Zhang, H. Zha, and C.-H. Chu. A time-series biclustering algorithm for revealing co-regulated genes. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01*, ITCC '05, pages 32–37, Washington, DC, USA, 2005. IEEE Computer Society.

Nederlandse Samenvatting

Time is a core dimension to understand the evolution of many phenomena. The economic situation of a given country, the current behavior of a customer, the diagnosis of a disease, are just a few examples that can be better understood by looking into time-ordered data.

With the consolidation of the digital development era, the collection and storage of time series data became ubiquitous. Making sense of such time series data can provide insights that help us understand better the present phenomena, extract trends and, in stationary cases, predict future events. Now, in a world increasingly connected by sensors and tractable behaviors, extracting meaningful explanations from time series can be a difficult task. This is why machine learning methods (or more generically, artificial intelligence) became so central nowadays.

Mining time series is a machine learning subfield that focuses on a particular data structure, where variables are measured over (short or long) periods of time. In this thesis we focus on multivariate time series, with multiple variables measured over the same period of time. In most cases, such variables are collected at different sampling rates. As a practical example, consider studying the health of an individual. Variables can be activity behaviors (number of steps, number of floors climbed), biometrics (weight, heart rate), medical records (historical of diseases, metabolites) and economical situation (house owner, contractual situation, income level). When combined, these variables can be explored with machine learning methods for multiple purposes.

In this Ph.D. thesis, we analyze and propose both supervised and unsupervised machine learning methods, which deal with multivariate time series. Firstly, we consider the possibility of unsupervised learning. In this case, we propose a pattern recognition method that discovers subsets of variables that show consistent behavior in a number of shared time segments. Furthermore, when in a supervised setting, given a dependent variable (target), we propose a method that aggregates independent variables into meaningful features. This method wraps both preprocessing and model learning tasks and generalized to work both for regression and classification problems. Experimental results show the potential of both supervised and unsupervised approaches mentioned.

Additionally to the methods above, we provide two tools in the form of Software as a Service, where users without programming background can intuitively follow the learning and testing methodologies for both methods. The intuition behind these tools is to explore the present role of a data scientist. The machine learning methods we implement are just one of the components of these tools, while the focus is on the user of the tool. The users of such tools are guided trough the whole process of importing data, preprocessing, learning and evaluating results.

Finally, we present an applied study of machine learning to improve speed skating athletes performance. More concretely, we report on a cooperation with the LottoNL-Jumbo team, who have an extensive asset of detailed training data. Here, we make a deep analysis of historical data, in order to help optimize performance results. We combine training and competition results data and make a production ready implementation of a machine learning system to optimize training schedules. In other perspective, we help avoiding under- and overtraining before a given competition.

English Summary

Time is a core dimension to understand the evolution of many phenomena. The economic situation of a given country, the current behavior of a customer, the diagnosis of a disease, are just a few examples that can be better understood by looking into time-ordered data.

With the consolidation of the digital development era, the collection and storage of time series data became ubiquitous. Making sense of such time series data can provide insights that help us understand better the present phenomena, extract trends and, in stationary cases, predict future events. Now, in a world increasingly connected by sensors and tractable behaviors, extracting meaningful explanations from time series can be a difficult task. This is why machine learning methods (or more generically, artificial intelligence) became so central nowadays.

Mining time series is a machine learning subfield that focuses on a particular data structure, where variables are measured over (short or long) periods of time. In this thesis we focus on multivariate time series, with multiple variables measured over the same period of time. In most cases, such variables are collected at different sampling rates. As a practical example, consider studying the health of an individual. Variables can be activity behaviors (number of steps, number of floors climbed), biometrics (weight, heart rate), medical records (historical of diseases, metabolites) and economical situation (house owner, contractual situation, income level). When combined, these variables can be explored with machine learning methods for multiple purposes.

In this Ph.D. thesis, we analyze and propose both supervised and unsupervised machine learning methods, which deal with multivariate time series. Firstly, we consider the possibility of unsupervised learning. In this case, we propose a pattern recognition method that discovers subsets of variables that show consistent behavior in a number of shared time segments. Furthermore, when in a supervised setting, given a dependent variable (target), we propose a method that aggregates independent variables into meaningful features. This method wraps both preprocessing and model learning tasks and generalized to work both for regression and classification problems. Experimental results show the potential of both supervised and unsupervised approaches mentioned.

Additionally to the methods above, we provide two tools in the form of Software as a Service, where users without programming background can intuitively follow the learning and testing methodologies for both methods. The intuition behind these tools is to explore the present role of a data scientist. The machine learning methods we implement are just one of the components of these tools, while the focus is on the user of the tool. The users of such tools are guided trough the whole process of importing data, preprocessing, learning and evaluating results.

Finally, we present an applied study of machine learning to improve speed skating athletes performance. More concretely, we report on a cooperation with the LottoNL-Jumbo team, who have an extensive asset of detailed training data. Here, we make a deep analysis of historical data, in order to help optimize performance results. We combine training and competition results data and make a production ready implementation of a machine learning system to optimize training schedules. In other perspective, we help avoiding under- and overtraining before a given competition.

Resumo

O tempo é uma dimensão fundamental para a explicação de muitos fenômenos. A situação económica de um determinado país, o comportamento expectável de um cliente, ou o diagnóstico das causas de uma doença são só alguns exemplos de fenómenos que podem ser melhor entendidos com dados ordenados temporalmente (séries temporais).

Com a consolidação da era digital, a recolha e armazenamento de series temporais tornou-se omnipresente. Um correcta interpretação de series temporais possibilita uma melhor compreensão de fenómenos actuais, a extração de tendências e a previsão de eventos futuros. Agora, num mundo cada vez mais conectado através de sensores e monitorização de comportamentos digitais, a extração de conhecimento de séries temporais poderá tornar-se uma grande desafio. Foi este desafio que tornaram os métodos de aprendizagem automática (normalmente denominados de inteligência artificial) em ferramentas essenciais nos dias que correm.

A extração de conhecimento de séries temporais é um campo especializado num tipo de estrutura de dados onde, para cada sistema ou sujeito, as variáveis são medidas várias vezes ao longo de um determinado período de tempo. Nesta tese focamos nesta estrutura de dados e consideramos o caso de séries temporais multivariadas. Na maioria dos casos, consideramos também que as variáveis são medidas a diferentes frequências (uma vez por cada dia, hora, minuto, segundo...). Por exemplo, variáveis poderão ser indicadores de actividade física (número de passos, número de patamares subidos a pé, ...), registos médicos (historial médico, variáveis metabólicas, ...) e indicadores económicos (proprietário de imobiliário, salário bruto, ...). Quando combinadas, estas variáveis podem ser exploradas com múltiplos fins, através de métodos de inteligência artificial.

Nesta tese de doutoramento, propomos métodos supervisionados e não supervisionados de aprendizagem automática (algoritmos), capazes de lidar com séries temporais. Primeiro, propomos um método de análise de clusters (biclustering). Este, foi capaz de encontrar conjuntos de variáveis que mostram um comportamento similar e consistente num determinado número de segmentos temporais. Alternativamente, numa situacão supervisionada, dada uma variável dependente e um conjunto de variáveis independentes, propomos um outro método que transforma e agrega as variáveis independentes de forma inteligente. Os resultados experimentais apresentam o potencial de ambos os métodos acima referidos.

Para além dos métodos de aprendizagem automática, nesta tese introduzimos duas ferramentas de Software enquanto serviço (*Software as a Service*). Nestas ferramentas, utilizadores sem conhecimentos de programacão podem seguir de forma intuitiva, o processo de aprendizagem automática e teste dos métodos mencionados no parágrafo anterior. Estas ferramentas permitem efectuar todo o processo de um cientista de dados, comecando pelo processo de tratamento de dados, aprendizagem, optimização e availiacao de modelos. A principal contribuição destas ferramentas não são os métodos de inteligência artificial, mas sim o apoio aos utilizadores destes métodos.

Por fim, apresentamos um método de inteligência artificial aplicado a um desporto de alta competição. O objectivo deste estudo é a melhoria dos resultados nas competições de atletas de patinagem de velocidade, um desporto com uma grande dimensão económica e social na Holanda. Mais concretamente, utilizando os dados de treino de uma equipa de alta competição, construimos modelos que optimizam os resultados competitivos através de ajustamento nos treinos. O nosso estudo mostra que, através de modelos personalizados, podemos evitar o treino excessivo ou defecitário.

List of publications

1. S. Miao, U. Vespier, J. Vanschoren, A. J. Knobbe, and R. Cachucho, Modeling sensor dependencies between multiple sensor types. In *Proceedings of the 2013 Machine Learning Conference of Belgium and the Netherlands (BeneLearn 2013)*, 2013.

2. J. Vanschoren, U. Vespier, S. Miao, M. Meeng, R. Cachucho, and A. J. Knobbe, Large-scale sensor network analysis: applications in structural health monitoring. In *Chapter of Big Data Management, Technologies, and Applications, IGI Global*, 2014.

3. R. Cachucho, M. Meeng, U. Vespier, S. Nijssen, A. J. Knobbe, Mining multivariate time series with mixed sampling rates. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, (Ubicomp '14)*, 2014.

4. M. He, R. Cachucho, and A. J. Knobbe, Football player's performance and market value. In *Proceedings of the 2nd Workshop of Sports Analytics, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2015)*, 2015.

5. S. Miao, U. Vespier, R. Cachucho, M. Meeng, and A. J. Knobbe, Predefined pattern detection in large time series. In *Information Sciences, Volume 329, Elsevier*, 2016.

6. L. W. Wijsman, E. Richard, R. Cachucho, A. J.M. de Craen, S. Jongstra, and S. P. Mooijaart, Evaluation of the use of home blood pressure measurement using mobile phone-assisted technology: the iVitality proof-of-principle study. In *JMIR mHealth and uHealth, Volume 4, Number 2, JMIR Publications Inc.*, 2016.

7. G. Veiga, W. de Leng, R. Cachucho, L. Ketelaar, J. Kok, A. J. Knobbe, C. Neto, and C. Rieffe, Social competence at the playground: Preschoolers during recess. In *Infant and Child Development, Volume 26, Number 1, Wiley Online Library*, 2017.

8. G. Veiga, L. Ketelaar, W. De Leng, R. Cachucho, J. Kok, A. J. Knobbe, C. Neto, and C. Rieffe, Alone at the playground. In *European Journal of Developmental Psychology, Volume 14, Number 1, Taylor & Francis*, 2017.

9. R. Cachucho, K. Liu, S. Nijssen, and A. J. Knobbe, Bipeline: a web-based visualization tool for biclustering of multivariate time series. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2016)*, 2016.

10. S. Jongstra, L. W. Wijsman, R. Cachucho, M. P. Hoevenaar-Blom, S. P. Mooijaart, and E. Richard, Cognitive testing in people at increased risk of dementia using a smartphone app: the iVitality proof-of-principle study. In *JMIR mHealth and uHealth, Volume 5, Number 5, JMIR Publications Inc.*, 2017.

11. A. J. Knobbe, J. Orie, N. Hofman, B. van der Burgh, and R. Cachucho, Sports analytics for professional speed skating. In *Data Mining and Knowledge Discovery, Volume 31, Number 6, Springer*, 2017.

12. R. Cachucho, S. Nijssen, and A. J. Knobbe, Biclustering multivariate time series. In *Proceedings of the 2017 Springer International Symposium on Intelligent Data Analysis (IDA 2017)*, 2017.

13. R. Cachucho, S. Paraschiakos, K. Liu, B. van der Burgh, and A. J. Knobbe, ClaRe: classification and regression tool for multivariate time series. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2018)*, 2018.

14. E. van der Spoel, N. Oei, R. Cachucho, F. Roelfsema, J. F. P. Berbée, G. J. Blauw, H. Pijl, N. M. Appelman-Dijkstra, and D. van Heemst, The 24-hour serum profiles of bone markers in healthy older men and women. In *Bone, Volume 120, Elvesier*, 2019.

Curriculum Vitae

Ricardo Emanuel de Gouveia da Costa Cachucho was born on the island of Madeira, Portugal, on the 26th of December in 1986. From 2001 until 2004 he was a student at Associação Promotora de Ensino Livre (APEL) in Funchal. He then moved to Porto to study Economics at Faculdade de Economia do Porto, at Porto University. He obtained his B.Sc. degree in 2009. Afterwards, he enrolled in a data mining Master programme at Porto University, graduating with a M.Sc. degree in October 2009. Towards the end of this Master programme, he developed his thesis with João Gama and João Mendes Moreira, both members of the Laboratory of Artificial Intelligence (LIAAD). During this period, he also moved to Leiden in the Netherlands, as part of an Erasmus exchange program.

Since February 2011, Ricardo has been living in the Netherlands. As of October 2011, Ricardo started working at Leiden Institute of Advanced Computer Science (LIACS) as a junior researcher. In January 2014, he started his Ph.D. in Computer Science in Leiden University. From August 2016, for a period of one year, he was a data scientist working for the football club A.Z. Alkmaar. This was followed by a one year Postdoc contract divided between LIACS and LUMC, where Ricardo managed two projects related to machine learning applied to wearable technologies. Since September 2018, he works at Pegasystems as Senior Solutions Consultant, with a focus on delivering actionable machine learning across industries.

This Ph.D. trajectory was performed under the guidance of prof. dr. Joost N. Kok, and with direct supervision of dr. Arno J. Knobbe.