SAPIENZA
UNIVERSITÀ DI ROMA

Prediction of Batch Processes Runtime Applying Dynamic Time Warping and Survival Analysis

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Data Science

Candidate

Paolo Graniero
ID number 1733715

Thesis Advisors

Prof. Chatzigiannakis Ioannis
Dr. Gärtler Marco

Academic Year 2018/2019

Thesis not yet defended

Prediction of Batch Processes Runtime Applying Dynamic Time Warping and Survival Analysis

Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: paolograniero8@gmail.com

Abstract

Batch monitoring is the sub-branch of statistical process monitoring that deals with the statistical analysis of variations occurring in the data describing industrial batch processes. Batch runs corresponding to the same recipe usually have different duration. The data collected by the sensors that equip batch production lines reflects this fact: time series with different lengths and unsynchronized events. Dynamic Time Warping (DTW) is an algorithm successfully used in batch monitoring, too, to synchronize and map to a standard time axis two series, an action called alignment. The on-line alignment of running batches, although interesting by itself, gives no information on the remaining time frame of the batch, such as its total runtime, or time-to-end. We notice that this problem is similar to the one addressed by Survival Analysis (SA), a statistical technique of regular use in clinical studies to model time-to-event data. Machine learning (ML) algorithms adapted to survival data exist, with increased predictive performance compared to classical formulations. We apply a SA-ML-based system to the problem of predicting the time-to-end of a running batch. We show a new application of DTW: the information returned by open-ended DTW can be used to select relevant data samples for the SA-ML system. We will see that there will be no adverse effect on predictive performance, while there will be a decrease in computational cost compared to the same SA-ML system that uses all the data available. We tested the system on a real-world dataset coming from a chemical plant.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Acknowledgements

I take this space to thank all the teachers I have ever had the opportunity to meet during my studies until now. Some of them gave me much, some of them gave me less, but all of them have been a guide towards reaching this goal.

The biggest thank, my full gratitude goes to my mother and my father, Antonietta and Gerardo, for their lifetime support, their patience. They have been my teachers and mentors even before I could speak. Every accomplishment I have ever achieved, and everything I will achieve in the future is thanks to them. I hope I made them proud of me as much as I am proud of them.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

If two people say a particular word, the chances are that it takes them different time to pronounce it. A situation as typical as this brings out an attribute common to many processes, both natural and artificial: different realizations of the same underlying process can have different duration — which is usually the rule, not the exception. From a human perspective, the example just stated hardly poses any issues: saying *now* or *nooow* makes no difference in a conversation; we easily recognize the word for what it is, *now*. A healthy human brain can extract relevant information and match the two words effortlessly.

The thing gets a little trickier from the machine perspective: referring to the example above, a difference in duration translates to different lengths of the recorded audio signals. This difference makes the direct comparison of the two samples not possible through standard matching procedures. For example, computing the Euclidean distance between the vector representations of the audio signals is not possible if the vectors have different length; an additional step is required to make the two samples of the same length, and thus directly comparable.

The objective of having data samples of the same length is just one of the aspects to take into account in such situations. For example, even in the presence of two series of data with the same length, significant events in the series could happen at different relative time positions, making it still difficult to compare them. Taking a step further in these considerations, we could also examine the case in which the process under examination is still occurring and ask ourselves: how long it will take to complete?

These are some ordinary circumstances faced when dealing with series of data with different length and unsynchronized events. Among others, these are some of the same problems addressed in statistical process monitoring, the general topic in which we can frame this work. In particular, we will look at the case of monitoring a batch process.

Batch production is a common manufacturing technique for chemical, pharmaceutical, and food industries where a given product is realized in a stage-wise manner following a given formula or recipe. Such a recipe defines rigorously the sequence of steps that each batch run has to follow. Nonetheless, it is quite common that distinct batch runs for the same recipe differ in several aspects, some of them time-related. For example, the total duration or duration of the individual sub-phases can vary

and, even in synchronized phases, significant events or alarms can happen at different relative positions in time. The magnitude of such variations requires advanced process monitoring techniques to ensure the consistency and the quality of the product, to improve the safety levels of the plant, and to better understand and control the process (H. Ramaker and Sprang 2004). This monitoring is, in turn, beneficial to the process operations and planning, and can lead to overall improvements.

A particular branch of process monitoring is the aforementioned statistical process monitoring: a set of techniques relying on mathematical tools that help to identify and control variations in the production process, analyzing data coming from the reactors. These data, collected from a large number of sensors, come with some peculiar characteristics as a consequence of the temporal variability stated above, affecting the analysis performed on them.

Many established statistical techniques for process monitoring, such as Multiway Principal Component Analysis, require that the input data have equal length, that for a batch process is its duration. As previously stated, even in case of the same duration, significant events could be asynchronous, leading to problems during the analysis, such as comparing different but synchronous events. These issues are not unique to batch data: Dynamic Time Warping (DTW), which originated in signal processing (Sakoe and Chiba 1978), is a widely successful and often adopted technique to homogenize data with a different time frame to a standard duration and synchronization of characteristics. An adoption to batch data is given, for example, in (Kassidas, Taylor, and MacGregor 1998).

There exist different versions of DTW; two of them are the standard DTW and the open-ended DTW. In batch monitoring, standard DTW is used to align two completed batches, mainly during off-line analysis, when completed batches are available. The aim of applying standard DTW, in this case, is to have data samples with the same length and synchronized events. The open-ended version of the algorithm is useful in on-line scenarios, where a running batch, therefore not yet completed, is aligned to a prefix of a completed batch. As such, a comparison between the running batch and a historical one is straightforward. Unfortunately, such DTW alignment does not give any information on the remaining time frame; for example, the time left before completion of the batch, the focus of this work.

We can look at the specific issue of establishing the time left until the completion of a batch as a time-to-event modeling problem, usually addressed by techniques such as Survival Analysis (SA). This statistical tool is a standard adopted technique in clinical studies to model the time until the occurrence of an event of interest, usually linked to the course of an illness. Many machine learning (ML) algorithms, like the ones described in (Hothorn et al. 2005), have been adapted to handle survival data, resulting in improved predictive performance over more standard techniques.

In this work, we address the problem of predicting the time-to-end of a running batch, articulating the approach in two phases. We first check the feasibility of using SA in the context of batch monitoring. After assessing this, we investigate if, by applying DTW, it is possible to obtain any improvements regarding the predictive performance or the computational cost of the analysis.

The final product of this work is a system combining DTW and SA that aims at predicting the time-to-end of a running batch given historical information on the process in a computationally efficient way. We apply DTW as a data-selection tool,

not as a time normalization technique, or to compute a distance measure, like it is usually used. We use the mapping information contained in the warping path, one of DTW algorithm's output, to select only a fraction of the data available. This data is used to train a SA-ML-based algorithm that returns an estimate of the time-to-end of the running batch. The proposed system results in a significant improvement of the computational cost of the analysis compared to using all the data available to train the same SA-ML-based model.

We have organized this work in three parts: part I (chapters 2 and 3) describes dynamic time warping with some of its applications to batch process monitoring, and survival analysis. This information will be useful to understand the content of part II (chapters 4 and 5), that describes the system proposed in this work and the data used to test it. Part III (chapters 6 and 7) discusses the results obtained and describes the conclusions we drew about the applicability of and possible improvements to the proposed approach.

Chapter 2

Dynamic Time Warping and Batch Process Monitoring

We briefly describe the Dynamic Time Warping (DTW) algorithm int his chapter. After explaining the general idea behind it, we give more detailed information, introducing relevant concepts and nomenclature. Our scope here is to give enough information to know what DTW is, what is the idea behind it, and to describe its main characteristics. We also give a brief review of current practice involving the use of DTW in batch process monitoring at the end of this chapter. These descriptions will help us in understanding better the role of DTW in the system proposed in this work.

2.1 Introduction

Dynamic Time Warping (DTW) is the name of a class of algorithms usually used for time series alignment. With the term *alignment*, we refer mostly to the homogenization of two characteristics of the time series under examination: total length and relative position of the events in the series. These phenomena, different duration and presence of unsynchronised events, usually occur when there are differences in local speed of the processes described (Spooner, Kold, and Kulahci 2017).

DTW originates from the field of spoken word recognition with the work of Sakoe and Chiba (Sakoe and Chiba 1978): they proposed a dynamic programming approach to the problem of removing non-linear fluctuations in the time axis of two recorded audio signals representing two words to be matched. The fluctuations are removed applying a non-linear function, the warping path: this is one of the outputs of the algorithm that we will describe later in this chapter.

Generally speaking, DTW is a pattern-matching algorithm which finds an optimal alignment between two time series that minimizes a suitably defined distance measure, invariant under local stretching and compression of the time dimension (Giorgino et al. 2009): the minimum distance computed between two warped time series is the so-called DTW distance, the other output of the algorithm that we will describe later.

The rest of this chapter is organized as follows: first, we describe the main characteristics of DTW, introducing useful concepts and nomenclature. Then, we

present the dynamic programming solution to DTW and the open-ended version of the algorithm. Finally, we describe some of the applications of DTW to the field of batch process monitoring.

2.2 DTW fundamentals

A general approach in comparing two or more entities is computing some dissimilarity measure between them, like when we compute the euclidean distance between pairs of vectors to establish what are the vector closer to a given one.

For two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ the Euclidean distance d is computed as

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{k=1}^{K} (u_k - v_k)^2} \tag{2.1}$$

An implicit assumption in (2.1) is that the number of components of the vectors is the same, namely, K.

Suppose now that we want to compare two sequences of values, let them be time series objects:

$$\begin{aligned} Y &= (y_1, y_2, \ldots, y_N) = \{y_t\}_{t \in \{1,\ldots,N\}} \\ X &= (x_1, x_2, \ldots, x_M) = \{x_t\}_{t \in \{1,\ldots,M\}} \end{aligned} \tag{2.2}$$

of length $N, M \in \mathbb{N}$ respectively. The assumption stated above about Equation (2.1) is usually not fulfilled when dealing with time series data: N and M in (2.2) could differ, making the computation of the distance in (2.1) impossible. A graphical example of such situation is presented in Figure 2.1

Simple ways to overcome this problem include brute force cutting of the more extended series, or linear transformations of the time axis in order to stretch (compress) the shorter (longer) series to match the length of the longer (shorter) one (Spooner, Kold, and Kulahci 2017). The first approach, cutting the more extended series, does not take into account variation in the local time of the series. The second approach, linear transformation of the time axis, assumes that such variations in the local time are present, but occur uniformly throughout the series. Both approaches have been shown to perform poorly in pattern recognition scenarios (Sakoe and Chiba 1978): we loose too much information with one approach, or we introduce an unrealistic assumption with the other one.

Unlike the approaches discussed above, DTW realizes a non-linear warping of the two series: the two time axis are locally stretched and compressed in order to make one series resemble the other as much as possible (Giorgino et al. 2009). Figure 2.2 graphically shows the mapping between the two series presented in Figure 2.1, aligned via DTW.

The final result of applying DTW is time normalization of the two series under examination: the warped series have the same length, and events are synchronized as much as possible. In practice, this warping is realized replicating every point of a time series as many times as the number of points in the other series to which DTW maps it. Figure 2.3 makes this action more clear.

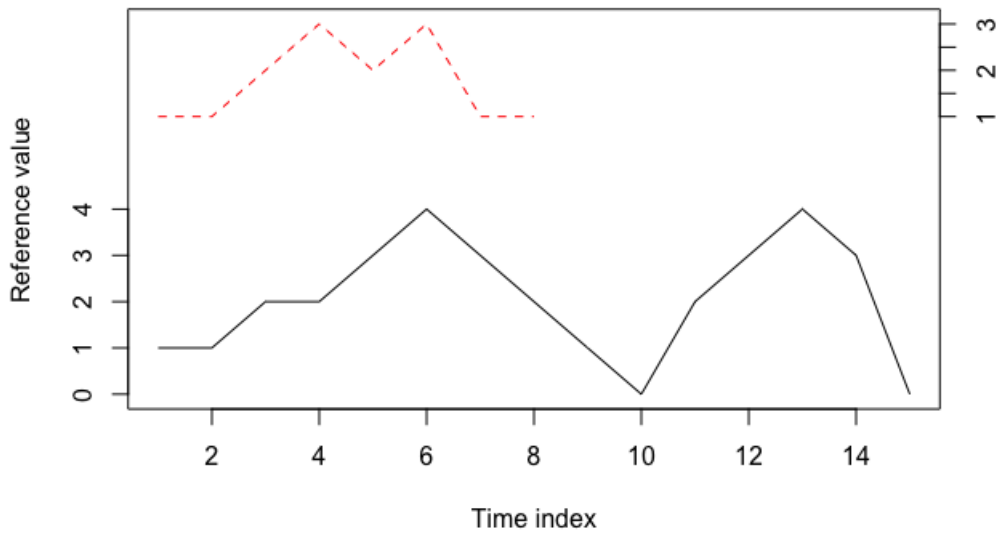


Figure 2.1. Example of two time series relative to the same phenomenon, but with different length: $N = 15$, $M = 8$.

Once the two time series have the same length and the events are matched, i.e., they are aligned, it is possible to compute the euclidean distance between them.

The minimum distance over every possible alignment is the so-called *DTW distance*, one of the outcomes of the algorithm together with the mapping between the series corresponding to the optimal alignment, the warping path.

We can interpret the DTW distance as the stretch-insensitive measure of the inherent difference between the two time series (Giorgino et al. 2009) while the warping path contains information on the local distortion of the time axes. It also acts as a mapping between the two series, relating similar regions with each other.

The following section presents a more formal description of the details of the algorithm.

2.3 Formal description of DTW

This section formally introduces the standard symmetric DTW algorithm, describing how it is possible to compute the DTW distance and to determine the optimal alignment between two time series.

Let's consider again the time series in 2.2:

$$Y = (y_1, y_2, \ldots, y_N)$$
$$X = (x_1, x_2, \ldots, x_M)$$

with length $N, M \in \mathbb{N}$ respectively. The two sequences are supposed to be sampled

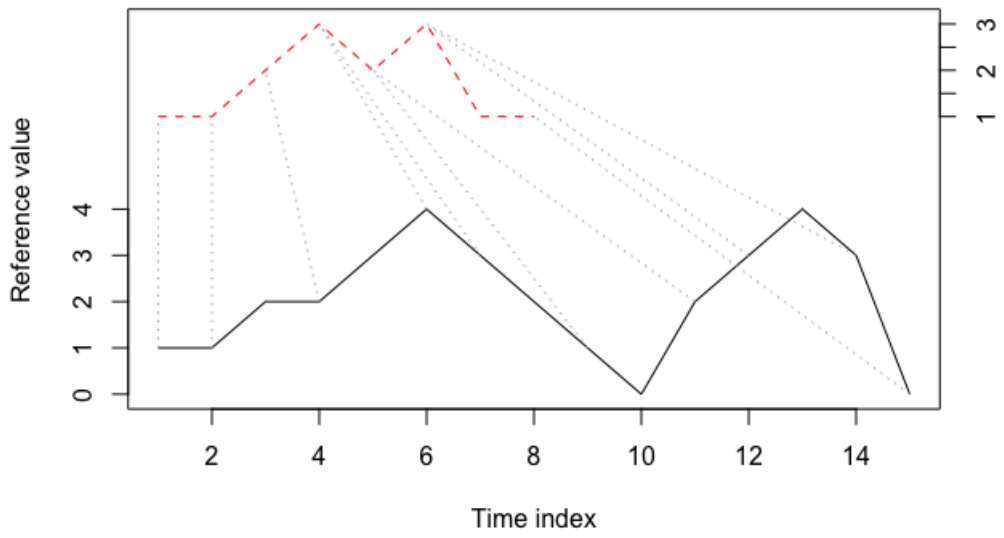


Figure 2.2. DTW mapping.

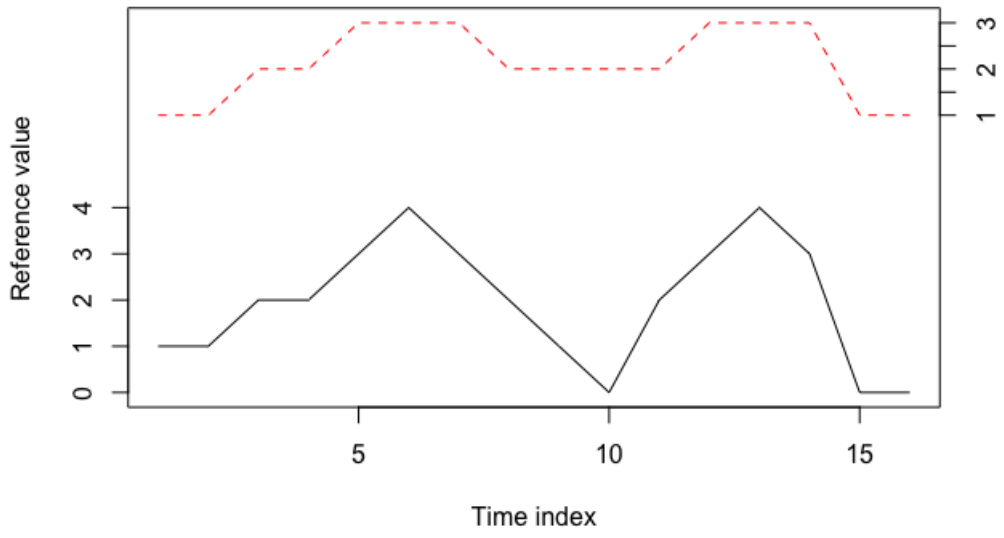


Figure 2.3. Visualization of the warped time series. The common axis has length $K = 16$

at the same rate, but they do not need to be simple discrete signals: any feature can be used to represent them. Let's denote the sampling space of these features as $\mathcal{F}$. Therefore $x_m, y_n \in \mathcal{F}$ for $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N\}$.

In order to compare the two sequences we need a *local dissimilarity measure* defined for every $u, v \in \mathcal{F}$, that is, a function of the form

$$d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_0^+ \tag{2.3}$$

Being a dissimilarity measure, $d(\cdot, \cdot)$ is small for similar features and big for different ones. A usual choice is Euclidean distance that in matrix form we write as:

$$d(\boldsymbol{u}, \boldsymbol{v}) = |\boldsymbol{u} - \boldsymbol{v}|\boldsymbol{W}|\boldsymbol{u} - \boldsymbol{v}|^T \tag{2.4}$$

where $\boldsymbol{W}$ is the weight matrix: a diagonal matrix with element w_{ii} being the weight assigned to the i-th component of $\boldsymbol{u} \in \mathcal{F}$.

Computing the pairwise dissimilarity between every point in the two series, one obtain the *local dissimilarity matrix* $\hat{d} \in \mathbb{R}^{N \times M}$, i.e.:

$$\hat{d}_{nm} \equiv d(n, m) \equiv d(y_n, x_m) \tag{2.5}$$

In Equation (2.5) we introduce the two equivalent notations of using as arguments of the local dissimilarity function both a pair of indexes referring to the elements of the two series and two actual elements of the sample space $\mathcal{F}$.

The objective of DTW is to find an alignment that minimizes the overall dissimilarity, also called *total cost*. The alignment is described by the so-called *warping path*, that we formally define now:

Definition 2.3.1 *Given two sequences* $Y = (y_1, y_2, \ldots, y_N)$ *and* $X = (x_1, x_2, \ldots, x_M)$ *of length* $N, M \in \mathbb{N}$ *respectively, a warping path is a sequence* $w = (w_1, \ldots, w_K)$ *with* $w_k = (n_k, m_k) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$ *for* $k \in \{1, \ldots, K\}$ *that satisfies the following conditions:*

1. *Boundary condition:* $w_1 = (1, 1)$ *and* $w_K = (N, M)$.
2. *Monotonicity condition:* $n_1 \leq n_2 \leq \ldots \leq n_K$ *and* $m_1 \leq m_2 \leq \ldots \leq m_K$.
3. *Step size condition:* $w_{k+1} - w_k \in \{(1, 0), (0, 1), (1, 1)\}$ *for* $k \in \{1, \ldots, K - 1\}$.

A warping path $w = (w_1, \ldots, w_K)$ defines an alignment between the two sequences $Y = (y_1, y_2, \ldots, y_N)$ and $X = (x_1, x_2, \ldots, x_M)$ pairing the points y_{m_k} and x_{n_k} of Y and X. The boundary condition requires that the first and last points of the two series match each other, that is $w_1 = (1, 1)$ and $w_K = (N, M)$. Removing the boundary condition on the last points returns the *open-ended* version of DTW, which deals with prefix matching. The monotonicity condition ensures that the time ordering of the elements of the series before the alignment is preserved in the aligned series. The step size condition is a kind of continuity constraint: no element in the two series is skipped during the alignment, and no replication of the pair of indices is allowed: all the w_k are pairwise distinct.

Figure 2.4 shows the effect of respecting the three conditions in Definition 2.3.1 and their violation in a classical representation of the warping path. In this representation, the x and y axis show the time index of the two series, and the dots represent the matched points. As an example of a warping path, we can write the one derived from the representation in Figure 2.4 (a):

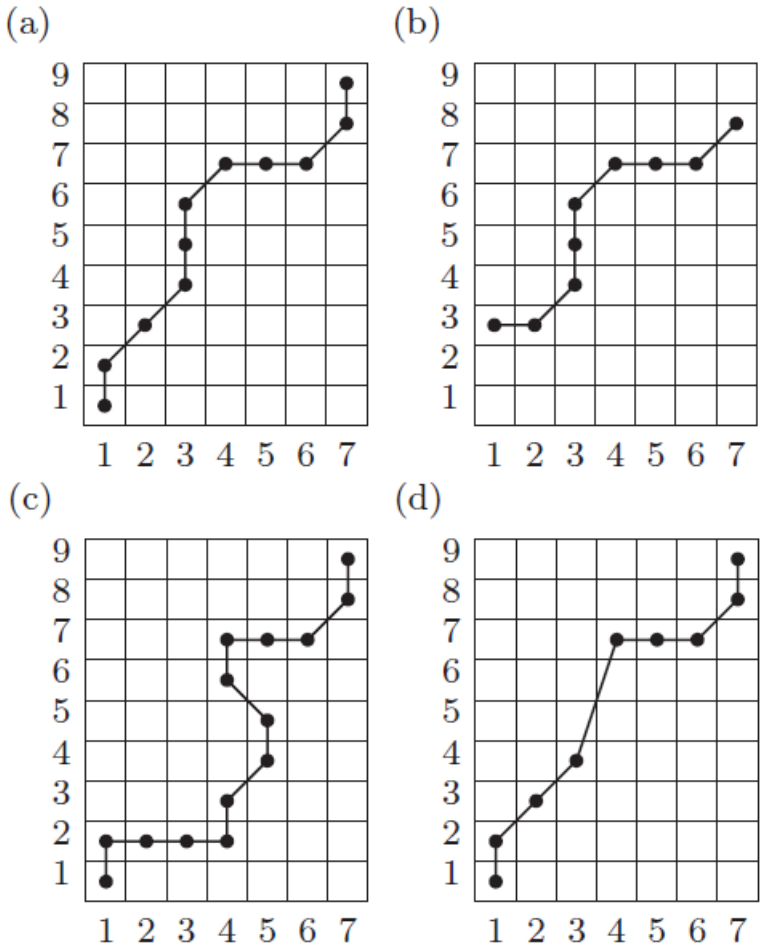


Figure 2.4. (a) Warping path satisfying all conditions in Definition 2.3.1 . (b) Violation of boundary condition. (c) Violation of monotonicity condition. (d) violation of step size condition. Picture from (Müller 2007).

$$w = [(1,1),(1,2),(2,3),(3,4),(3,5),(3,6),(4,7),(5,7),(6,7),(7,8),(7,9)] \quad (2.6)$$

where, the lengths of the two series are 7 and 9 and the length of the warping path is 11.

Given the warping path $w = (w_1, \ldots, w_K)$ that aligns Y and X, there are different ways to assign a total cost $D_w(Y, X)$ to it. The easiest one is simply summing the cost of each step, considered to be the dissimilarity between the two points paired by the warping step:

$$D_w(Y, X) = \sum_{k=1}^{K} d(y_{n_k}, x_{m_k}) \quad (2.7)$$

There are some aspects to point out about this definition that regard the

minimization of the total cost:

- Diagonal steps are favored compared to horizontal and vertical ones
- The total cost depends on the length of the warping path

A definition of the cost of a warping path that takes into account these issues is the following (Giorgino et al. 2009):

$$D_w(Y, X) = \frac{\sum_{k=1}^{K} d(y_{n_k}, x_{m_k}) m_w(k)}{M_w} \tag{2.8}$$

where $m_w(k)$ is a per-step weighting coefficient and $M_w = \sum_{k=1}^{K} m_k$ is the corresponding normalization constant. This constant is not always well defined: whether it is independent of the warping path w or not depends on the choice of the $m_w(k)$: various choices of the set of $m_w(k)$ defines different *step patterns*, something we will talk about later on in Section 2.4.

Given the total cost of an alignment as defined in (2.8), the objective of DTW is to find the warping path w^* that minimizes it over all possible alignments. One option to do this is enumerating all the possible warping paths satisfying the imposed constraints and then choosing the one, not necessarily unique, corresponding to the minimum distance. This procedure is not computationally feasible since the number of paths to consider grows exponentially with the size of the problem (i.e., length of the series). The dynamic programming approach allows us to find the optimal path in $O(MN)$; different approximate solutions exist to improve the computational complexity up to $O(N + M)$ (Salvador and Chan 2007).

The next section describes the dynamic programming approach to DTW.

2.4 Dynamic programming solution to DTW

This section describes how to find an optimal warping path in $O(NM)$ via a dynamic programming approach. We introduce the notions of step pattern and accumulated distance matrix.

Dynamic programming is an optimization method in which splits the given problem into many sub-problems that are solved recursively to find the optimal solution to the original problem. To apply this paradigm to the DTW problem, we consider sub-sequences of X and Y of length $n \leq N$ and $m \leq M$ respectively

$$X_m = (x_1, \ldots, x_m)$$
$$Y_n = (y_1, \ldots, y_n)$$

and define

$$D(n, m) = \min_w (D_w(Y_n, X_m)) \tag{2.9}$$

In other words, $D(n, m)$ is the DTW distance of the sub-problem defined by the sub-sequences Y_n, X_m. The values $D(n, m)$ define a matrix $\hat{D} \in \mathbb{R}^{N \times M}$, called the *accumulated distance matrix*. It is clear that $D(N, M)$ is the DTW distance of

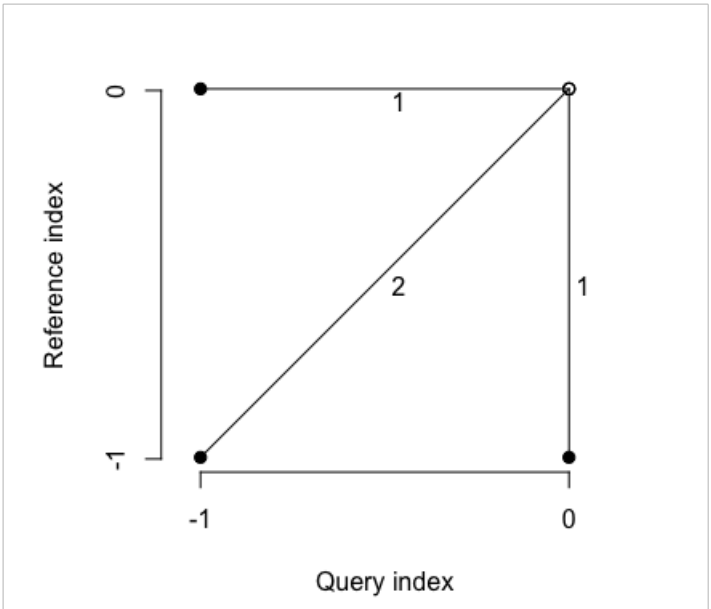


Figure 2.5. Representation of the *symmetric2* step pattern in relative coordinates

the original problem, and the corresponding warping path is the optimal one for these sub-sequences. Equation (2.9) gives a way to compute the elements of the accumulated distance matrix iteratively. The only remaining task is defining how to fill the cells of this matrix. The concept of *step pattern* serves this purpose.

A step pattern defines two things: what are the possible values a cell of the accumulated distance matrix can take and what are the steps of the optimal warping path preceding a given step. This is done specifying the weighting coefficients m_k appearing in Equation (2.8). Through the selection of a specific step pattern, it is possible to force a particular behavior on the warping path. Two examples considering standard step patterns found in most literature about DTW can show what this means.

Figure 2.5 shows how the *symmetric2* step pattern is represented using *relative coordinates*. It is a graphical representation of the iterative formula used to fill the accumulated distance matrix. The cell of interest (the one to be filled in the accumulated distance matrix) in this representation is the one in $(0, 0)$. For a cell (i, j) of the accumulated distance matrix, the *symmetric2* step pattern gives the following iteration formula:

$$D(i,j) = min \begin{cases} D(i-1,j) + d(i,j) \\ D(i-1,j-1) + 2d(i,j) \\ D(i,j-1) + d(i,j) \end{cases} \tag{2.10}$$

where $d(i, j)$ represents the local dissimilarity defined in Equation (**??**).

Having Figure 2.5 as a reference, we see that full dots refer to elements of the accumulated distance matrix, while empty dots refer to elements of the local distance

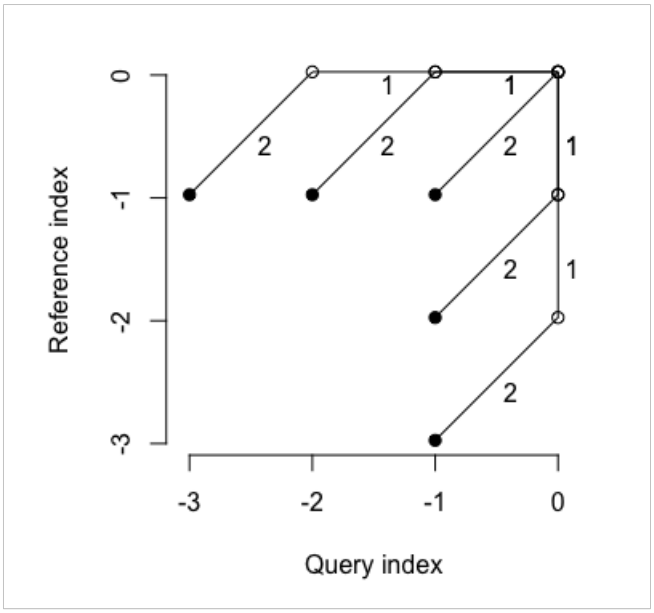


Figure 2.6. Representation of the *symmetricP05* step pattern in relative coordinates

matrix. The numbers on the edges represent the multiplicative factor m_k of the element of the local distance matrix. Once assigned $D(1,1) = d(1,1)$, every other element of the accumulated distance matrix can be computed iteratively. If an entry falls outside of the matrix, i.e. $i - 1 < 1$ or $j - 1 < 1$, that element is set to $+\infty$ by convention.

Another, more complex, example is the *symmetricP05* step pattern represented in Figure 2.6. The iteration formula in this case reads:

$$D(i,j) = min \begin{cases} D(i-1,j-3) + 2d(i,j-2) + d(i,j-1) + d(i,j) \\ D(i-1,j-2) + 2d(i,j-1) + d(i,j) \\ D(i-1,j-1) + 2d(i,j) \\ D(i-2,j-1) + 2d(i-2,j-1) + d(i,j) \\ D(i-3,j-2) + 2d(i-2,j) + d(i-1,j) + d(i,j) \end{cases} \tag{2.11}$$

Comparing the two step patterns, we notice that *symmetric2* imposes no local constraint on the warping path: any number of vertical or horizontal steps is allowed without any need for diagonal steps. On the other hand, *symmetricP05* introduces a constraint on the local slope of the warping path: it is possible to take at most two consecutive vertical or horizontal steps, after which a diagonal step is required as specified by the step pattern. This constraint limits the strength of the warping and is usually a strategy adopted to avoid unreasonable warpings: a strong warping could take a long section of a series and map it to a single point of the other one. The local constraint also affects the global behavior of the warping path, resulting in a restriction of the search space for the optimal warping path. Such constraint

affects the warping path as a whole and is thus known as *global constraint*. We give more information about this later.

The two step patterns shown above were introduced in (Sakoe and Chiba 1978) together with the *symmetric1*, *symmetricP1* and *symmetricP2*. For *symmetricP1*, *symmetricP2* and *symmetricP05* the *Px* notation serves to indicate the ratio between the number of diagonal steps required after one horizontal/vertical step in the warping path. For *symmetricP05* this ratio is 1:2, in *symmetricP1* and *symmetricP2* it is respectively 1:1 and 2:1. Figure 2.7 shows *symmetric1*, *symmetricP1* and *symmetricP2* step patterns explicitly.

Once we have filled the whole accumulated distance matrix, we traverse it backwards from cell (N, M) to cell $(1, 1)$ following the path of minimum cost, that is, for a cell (n_k, m_k), the predecessor step in the warping path will be:

$$(n_{k-1}, m_{k-1}) = argmin \begin{cases} D(n_k - 1, m_k) \\ D(n_k - 1, m_k - 1) \\ D(n_k, m_k - 1) \end{cases} \tag{2.12}$$

The minimum in Equation (2.12) could be not unique, so the optimal warping path is not unique in general, unlike the DTW distance that is always well defined.

Global constraints

Step patterns are also called *local constraints* since they limit the behavior of the warping path on a local scale. On the other hand, *global constraints* act by restricting the search space for the warping path to a specific region of the accumulated distance matrix. The main reason to apply a global constraint is to improve the computational cost of filling the accumulated distance matrix, even though the computational class of the problem is not changed. Two of the most known global constraint are the Sakoe-Chiba band and the Itakura parallelogram.

- The **Sakoe-Chiba band** (Figure 2.8) arises when we limit the maximum possible time distance between to aligned points. A Sakoe-Chiba band of width q corresponds to imposing that

$$|n_k - m_k| \leq q \ \forall k \in \{1, \ldots, K\} \tag{2.13}$$

- The **Itakura parallelogram** (Figure 2.9) forces a smoother alignment of the end sections, pushing the warping path toward the main diagonal of the accumulated distance matrix. A *symmetricPp* local constraint results in an Itakura parallelogram with slopes $\frac{p}{1+p}$ and $\frac{p+1}{p}$; p represents the shape parameter of the Itakura parallelogram.

2.5 Open-ended DTW

If in Definition 2.3.1, we relax the boundary condition of the warping path at the endpoint, we obtain the open-ended DTW. This version of DTW does not look for a global alignment: instead, it aligns one of the series to a prefix of the other one.

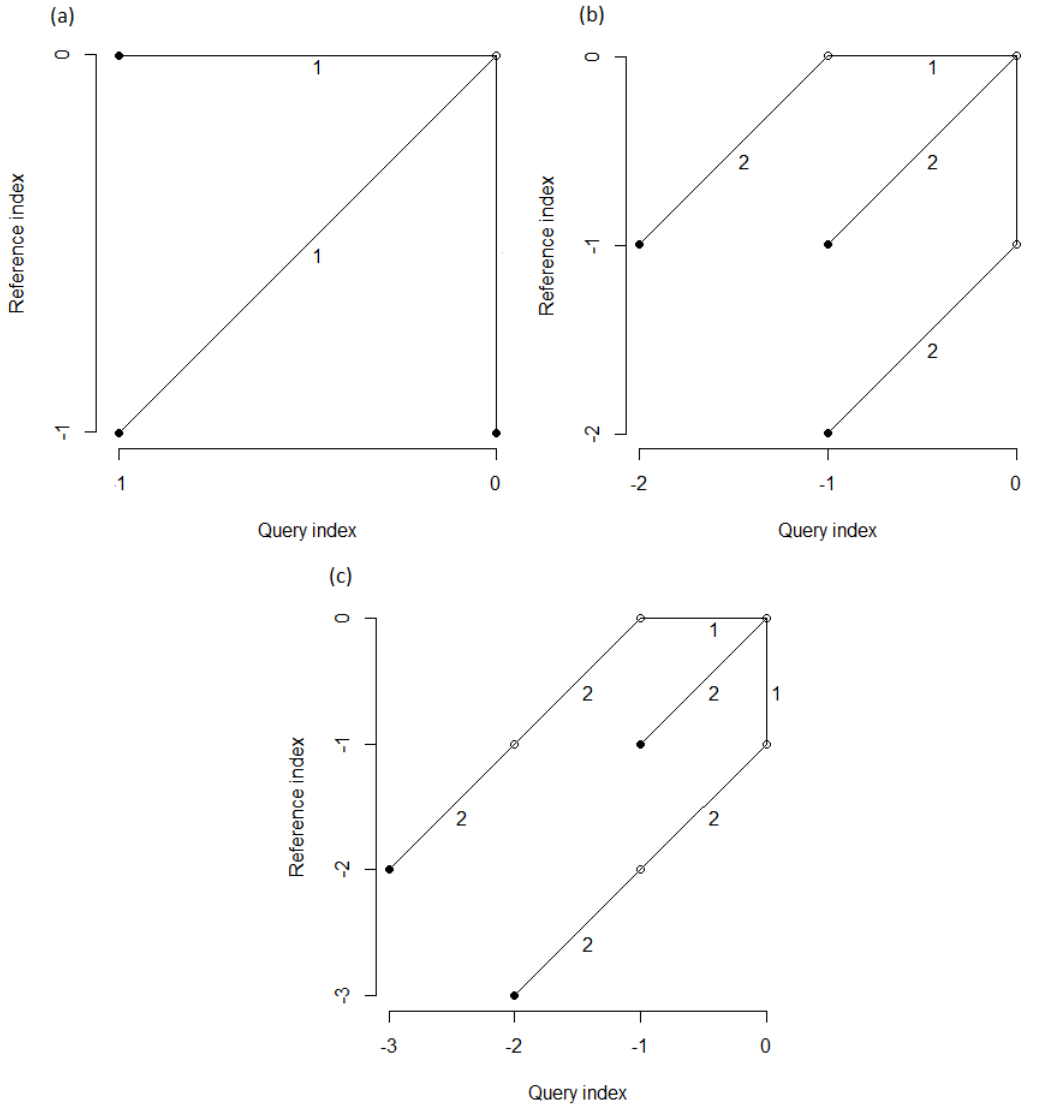


Figure 2.7. Standard step patterns: (a) *symmetric1*. (b) *symmetricP1*. (c) *symmetricP2*

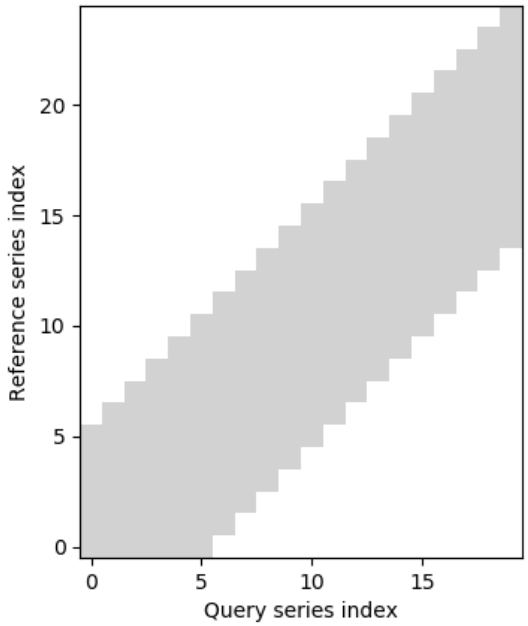


Figure 2.8. Sakoe-Chiba band with width $q = 5$. The warping path is confined to the grey region

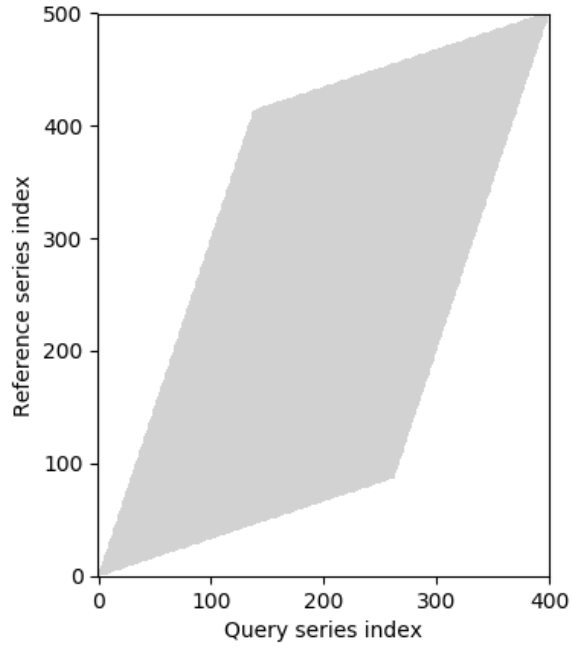


Figure 2.9. Itakura Parallelogram with shape parameter $p = 0.5$. The warping path is confined to the grey region

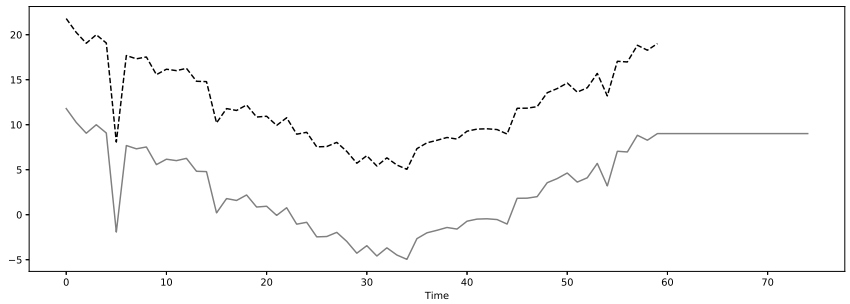


Figure 2.10. Example of situation in which open-ended DTW is useful. This version of the algorithm can discard the constant portion of the signal, clearly an artifact

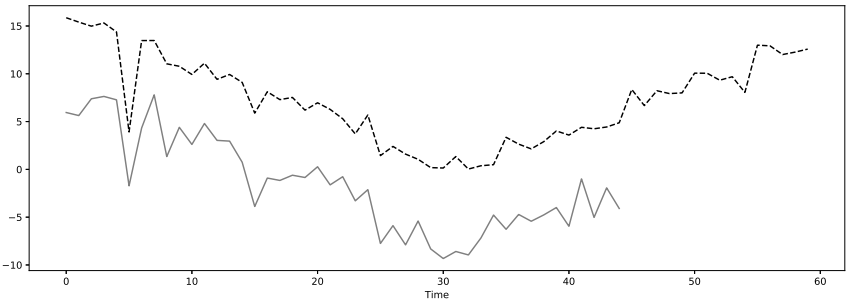


Figure 2.11. Example of ongoing series, to be aligned via open-ended DTW to a complete one.

This possibility allows for more flexibility of the alignment and is useful in many situations: for example, it can be used to discard irrelevant end regions, like the trivial example in Figure 2.10. Here the standard version of DTW would cause an unnecessary distortion of the series, or it would simply wrap the whole constant section to the last point of the short one. Open-ended DTW instead maps the short series to the prefix of the long one exactly matching it, discarding the rest.

A situation in which open-ended DTW is the only way to apply DTW is in on-line applications: we align an incomplete series to a prefix of a complete one, in order to get information about its current development. Figure 2.11 illustrates an example of such a situation.

The problem we found with this version of DTW is that it gives no information on the remaining time frame of the ongoing series. Given the alignment so far, what is the probable final duration of the ongoing series?

This work represents an effort to answer this question, applied to the specific case where the series under examination represent data from a batch production line. We briefly describe this context in the next section, also noting how DTW has been used so far to make it clear how the approach in this work is different and how it can complement current practices.

2.6 DTW in the process industry

As reported in the introductory chapter, this work aims to propose a system capable of predicting the total runtime of a running batch in a computationally efficient way. The proposed system is an example of batch process monitoring, that is, statistical process monitoring (SPM) applied to batch processes.

This section introduces the batch production paradigm and the data produced in such processes, in addition to a description of how DTW is usually used when monitoring batch processes.

2.6.1 Batch production

Batch production is a common manufacturing technique in many industries, such as chemical, pharmaceutical, and food industries. It is especially useful for high-quality, low-volume products.

A typical batch run looks like this: the ingredients are poured in a batch reactor where the reaction is initiated through the administration of heat or chemical initiators. After a certain amount of time and a specified sequence of actions, the reactants have been converted to the final product, and the reactor is ready for another batch (H. Ramaker and Sprang 2004). At the opposite side of the spectrum of manufacturing techniques, we find continuous production, where we have uninterrupted flows of ingredients entering the production line and final products coming out of it.

In the brief description given above of batch processes, we can identify a loading phase and what is essential for us, a transformation phase where the ingredients are converted to the desired final product. The amounts of ingredients to load, the sequence of steps to perform during the transformation phase, and every other possible operational instruction needed to obtain a given product are part of the recipe of the process under consideration. We say that batch processes are recipe-driven. Such a recipe is one of the advantages of batch production: for example, using the same equipment, it is possible to obtain a different product changing the recipe. Other advantages include the possibility to test and certify batch-by-batch the products, more comfortable to deal with from a regulatory point of view (H. Ramaker and Sprang 2004).

Batch processes come with other peculiar characteristics, some of them problematic for the operations since bringing variability to the process: the most common example of this variability is that even following the same recipe, the quality of the final product can vary from batch to batch due to unobserved phenomena. In this work, we focus our attention on the variability in the batch time, i.e., the time it takes to convert the ingredients to the final product. Different runs of the same recipe usually have different duration; sometimes, a batch can take a multiple of the time it took for another batch. This fact poses challenges in planning the operations of the production line, like when to start the next batch, or it could also be a matter of safety: abnormal duration could be linked to unexpected and potentially dangerous phenomena taking place in the line.

All the variability present in the processes requires advanced monitoring techniques. Some of these techniques rely on statistical principles to analyze the data

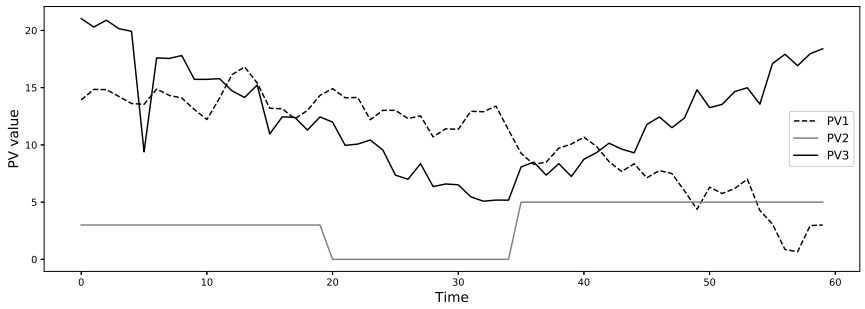


Figure 2.12. Example of batch data

coming in real-time from the reactors and derive useful information about the process. These techniques take on the collective name of statistical process monitoring. We now describe the data usually collected in batch production lines.

Batch production data

The reactors for batch production are always equipped with several sensors that measure different variables over time. Most of them are engineering variables, such as temperature, pressure, flows of materials or cooling fluids, the power consumed by steering engines; some other variables represent the state of specific equipment, like the opening of a valve. All of them are generally named Process Variables (PVs).

The number of sensors that equip a production line can easily reach hundreds; so does the number of PVs measured and collected. The sampling rate at which the PVs are measured varies from seconds to minutes, depending on the rate at which we expect the variable to change; for example, a binary variable like the opening of a valve could be measured only at the time the valve changes state.

For each batch run, the data collected represent a multivariate time series: each dimension represents a PV, each measured with the same sampling rate (when the PVs are measured at different sampling rates, aggregation or interpolation are needed). Meta-data comprise the recipe used, starting and ending time and date, nature, and units of measure of each PV.

When considering many batches, the resulting data set consists of several multivariate time series, ideally all with the same dimensionality and possibly with different lengths. In practice, it is possible to have different dimensions for series corresponding to different batch runs since some sensors could be turned off or added in a second moment, or the data could be missing for some unknown reason.

An example of such data is given in Figure 2.12

2.6.2 Statistical process monitoring

Every industrial process is subject to variations. Some of them are inherent to the process itself, while some others are external (H. Ramaker and Sprang 2004). Process monitoring is performed in every industry to prevent process disturbances from causing environmental stress and economic losses. Process monitoring has the potential to lead to time and cost reduction.

The aspiration of who operates a production line is to have a stable process, consistent and under control. In order to reach this optimum condition, statistical process monitoring studies the variations occurring in the process variables, trying to understand the source of variability or at least predicting the effect of the variation, with a double aim: establishing if an ongoing process will give an on-spec final product and improving the formula being used in future runs.

Batch process monitoring is the application of SPM to batch processes. The analysis differs from other types of processes, mainly due to the time variability stated above in this work. The high non-linearity of the behaviors of the PVs needs specific adaptations of standard techniques.

DTW in batch process monitoring

This section aims at showing how DTW is currently used in the monitoring and analysis of industrial processes. This investigation of current practice serves as one of the motivations for the attempt made in this thesis work to better exploit the information returned by the algorithm. A new analysis is then possible, focusing on different but not less critical tasks than the usual ones.

As we saw in the previous sections, the information returned by DTW is quite rich:

- A stretch-invariant distance measure, the DTW distance.
- A mapping between the two series being compared, the warping path.

The first of this information, the DTW distance, has resulted particularly useful in classification tasks. For example, a significant concern in industrial process monitoring is fault detection and isolation (FDI), therefore classifying batches as faulty or non-faulty. The term fault in this context indicates anything that causes a deviation of the operations from a desired operating point (Kassidas, Taylor, and MacGregor 1998).

Some of the modern methods used for fault detection rely on supervised classification approaches leveraging historical data on previous faults. An industrial plant is usually equipped with a multitude of sensors that acquire measurements about engineering variables and state of the machines. When a faulty situation arises, like a change in the quality or quantity of the final product, the related data are used as an instance of a faulty pattern. A diagnostic study on the causes of the fault, like checking the quality of the raw materials or the state of a sensor, is used to assign a label to the given faulty pattern. Once several such patterns are collected, building up a database of faults, the next step is classifying the state of the plant comparing its current situation with the fault database. There are some requirements for a fault diagnosis system, some of which have been met applying DTW (Kassidas, Taylor, and MacGregor 1998):

- The diagnosis should be independent of the duration of a fault.
- The diagnosis should be independent of the operating point of the plant.
- The diagnosis should take into account the uncertainty related to the onset of a fault

First, the time normalization effect of DTW helps with the different duration problem. Moreover, different speeds in the dynamic of the plant are also naturally taken into account, satisfying the second requirement in the list, and making the DTW distance suitable for classification use. Regarding the last requirement, it is possible to include points supposed to belong to the faulty pattern, and then let DTW decide whether to keep them or not in the alignment. This decision is possible using a step pattern like *symmetric2*, that allows for any number of warping steps, thus making the algorithm capable of cutting out the end sections of the pattern that are not useful.

Many statistical procedures for monitoring batch processes rely on the use of Multiway Principal Component Analysis (MPCA). This technique is used to monitor the behavior of the PVs as and identifying deviation from normal behavior, possibly identifying the PVs that influence or are influenced by the fault. Techniques like MPCA require that the data being used are of the same length, and usually, DTW is used to homogenize the time axis of the data used. The application of MPCA is both off-line, for example, when identifying types of faults to create a database of such instances (Kassidas, Taylor, and MacGregor 1998), or on-line, checking in real-time the behavior of the system to predict the failure of a batch (Gao et al. 2001).

This chapter introduced the first principal component of the system proposed in this work. We now have a standard dictionary that will be useful when describing the system proposed in Chapter 5 and the necessary preprocessing steps described in Chapter 4.

Chapter 3

Survival Analysis Fundamentals

This chapter briefly introduces Survival Analysis, a statistical tool aimed at analyzing and modeling time-to-event data. Usually, the challenge with such data is the presence of censored data points for which we do not have an actual time-to-event, but only a measure of a time where the event did not happen. This censoring is not the case in our data set, but this statistical technique is still valuable since it makes the approach proposed in this model able to generalize to situations in which we have incomplete batches.

3.1 Introduction

Survival Analysis (SA) is a statistical technique used to analyze and model time-to-event data. The main reason behind the development of such a sub-field of statistics is the problem of dealing with censored data. This type of data occurs because of either the time limitation of the study period or losing track during the observation period (P. Wang, Li, and Reddy 2017). A classic example is found in the health-care domain, where the event of interest could be death, hospital readmission, or discharge from hospitalization (P. Wang, Li, and Reddy 2017). The censoring in such examples can be due to many factors: death caused by something different from the disease or condition object of the study, readmission to a different hospital, or patients moved to other hospitals. In all these cases, the time to the event of interest is not directly observable; we know a time value up to which the event of interest has not occurred yet.

A data point in a survival analysis data set will usually be represented by a triplet (X_i, y_i, δ_i): X_i is the feature vector, δ_i is the indicator variable regarding censoring ($\delta_i = 0$ for uncensored data and $\delta_i = 1$ for censored data), and y_i denotes the observed time that is equal to the survival time T_i for uncensored instances and to the censoring time C_i for censored instances.

In this work, the data set analyzed presents no censoring, but the survival analysis-machine learning approach, in combination with a standard regression model on the output of the model, has resulted in a better performance against the sole standard regression model for the prediction of the time-to-event considered here.

We first briefly describe the standard approach to SA, and then we focus on the

ML approach. The latter, used in this work, has a better predictive performance compared to standard approaches. The predictive performance is not directly related to the estimation of the quantity of interest, the time-to-event. We give more details in Section 3.3.

3.2 Standard Survival Analysis

One of the primary goals in SA is to estimate the so-called survival function, which is used to represent the probability that the time to the event of interest T is not earlier than a specified time t (Lee and J. Wang 2003), that is

$$S(t) = Pr(T \geq t) \tag{3.1}$$

From $S(t)$ we derive the cumulative death distribution function $F(t) = 1S(t)$ (the probability that the event of interest occurs earlier than t) and the death density function $f(t) = \frac{d}{dt}F(t)$.

These functions relate to the commonly used hazard function $h(t)$, also called the conditional failure rate, as

$$h(t) = \frac{f(t)}{S(t)} \tag{3.2}$$

All the functions are essential ingredients of Survival Analysis. We do not need to go into details about them since it would not allow us to gain deeper insights into the system we propose in this work. Standard statistical approaches aim at making predictions of the survival time and estimate the survival probability at the estimated survival time. However, the focus is more on characterizing both the distributions of the event times and the statistical properties of the parameter estimation by estimating the survival functions (P. Wang, Li, and Reddy 2017).

Statistical approaches can be grouped in three main categories: parametric, non-parametric and semi-parametric. The base for some of them is Cox's proportional hazard assumption:

$$h(t, X_i) = h_0(t)exp(X_i\beta) \tag{3.3}$$

where X_i is a feature vector, $h_0(t)$ is the baseline hazard function, β is the coefficient vector. This assumption is interesting to us since, for machine learning approaches that do not rely on such an assumption, it is not possible to obtain a time-to-event estimate.

3.3 Machine learning approach to survival analysis

Machine learning approaches to SA focus more on the prediction of event occurrence at a given time point by incorporating the traditional SA methods with various ML techniques. Such methods are suited for high-dimensional problems and take advantage of the recent developments in machine learning and optimization to learn the dependencies between features and survival times in different ways (P. Wang, Li, and Reddy 2017).

Almost all ML algorithms have been adapted to incorporate survival concepts: we have survival trees, support vector machines adapted to SA, and also ensemble methods. In this work, we use as SA-ML model, a Gradient Boosting Survival Analysis (Hothorn et al. 2005). This model does not satisfy Cox's proportional hazard model, and because of this, it is not possible to obtain a direct estimate of the time-to-event.

The output of such a model is a risk score on an arbitrary scale. It can be used to rank data samples: indicating with $R(i)$ the predicted risk score for a sample i, if $R(i) > R(j)$ then we expect sample i to experience the event of interest before sample j. If we orde the samples according to their predicted risk score (in ascending order), one obtains the sequence of events, as predicted by the model (*Understanding Predictions in Survival Anlysis* n.d.). In this work, we try to overcome this limitation by applying a standard regression model to the output of the SA-ML model: the regression model, Random Forest, learns to convert the predicted risk score to an actual time-to-event estimate given the risk score of the training samples. It then predicts the time to event of the test sample given its predicted risk score.

In this chapter, we have very briefly introduced the main concepts of survival analysis. This technique is suited to deal with time-to-event data, the same as we have here: time until the completion of a batch. Recent developments in machine learning methods have been used in the survival context to obtain algorithms specific for survival data with a focus on the predictive performance of the time-to-event.

Many machine learning approaches do not return a direct time estimate, but only a risk score. We have described how we try to overcome this problem in the present work, applying a standard regression model to the output of the survival model. The implementation in the system proposed in this work will be described in Chapter 5.

Chapter 4

Data

This chapter describes the data available to test the proposed system and how it has to be processed to be used by it. Then we say a few words about the optimal configuration of DTW.

4.1 Raw data

The data used in this work come from an industrial plant for the batch production of a chemical product. Due to non-disclosure agreements, it is not possible to give more details about the process generating the data, the features' names presented are anonymized, and every graphical representation of any PV trend is not possible.

The data set contains multivariate time-series representing the sensor data gathered during the execution of an individual operation of the production process. Nevertheless, for clarity reasons, we will refer to each data sample as a batch, even though they do not represent entire batches.

The raw data contains batch runs performed from November 2014 to December 2017, for a total of 425 batches. Figure 4.1 represents the duration in minutes of every batch run against its starting date; different colored regions refer to different years. From this figure, it is possible to observe an increasing trend in the duration of the batch runs; since we had no information to explain this phenomenon, we supposed that it is due to small changes and improvements made to the recipe. To mitigate the influence of this trend and rely solely on the information contained in the PVs, we split the dataset year by year for the three years 2015, 2016, 2017, at the moment of testing the proposed system, as explained in Chapter 6. The data from 2014 has been merged with the 2015's ones.

4.2 Preprocessing

The raw data described in the previous section need to be preprocessed to be ready for analysis. The actions performed on the raw data, in a year-wise manner, are:

- Outlier detection and removal
- Reference batch selection

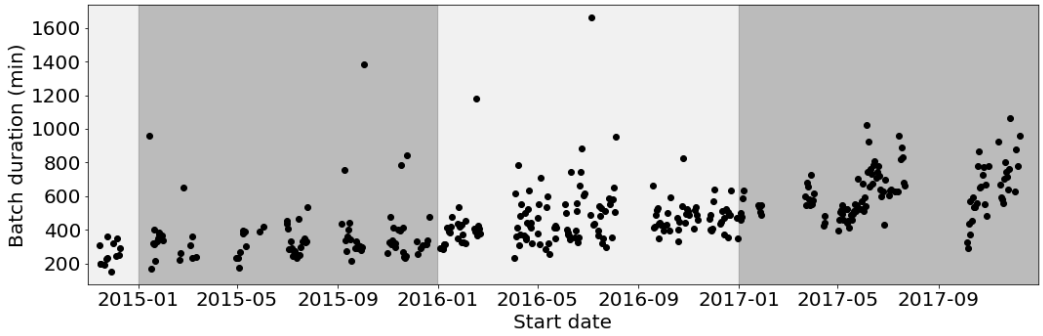


Figure 4.1. Duration of a batch against its starting date. The vertical bands identifies the different years

- PVs Selection
- PVs Rescaling

The next few sections will describe each of this steps.

4.2.1 Outliers detection and removal

We consider outlier batches according to their duration. We chose to identify the outliers by the box-plot method: if the duration of a batch is above 1.5 times the interquartile range above the upper quartile, we consider it an outlier. For example, figure 4.2 represents the outlier identification for the 2016 data: we removed from the dataset the seven batches with duration over 708 minutes.

We need to remove outlier batches for two reasons. First, outliers decrease the performance of the model. Second, if we do not do it, we cannot configure DTW properly: as mentioned in Section 2.4, one of the parameters to set up in DTW is the step pattern. The choice of the step pattern depends on the difference in length of the query batches and the reference one: if this difference is too big, the only allowed step patterns are the ones that do not restrict extreme warpings (like

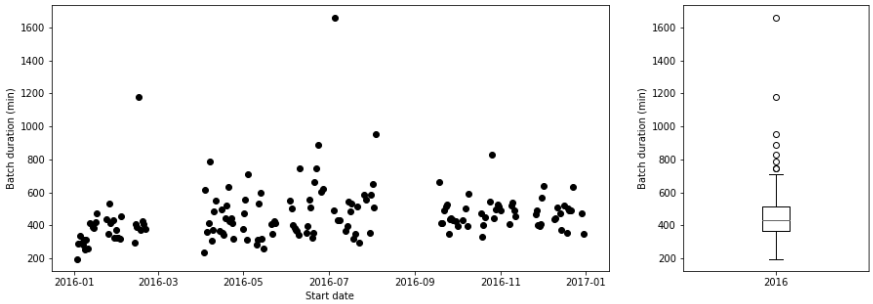


Figure 4.2. Identification of outlier batches. The scatter plot on the left represents the 2016 data from Figure 4.1; the box-plot on the right identifies 7 batches (with duration over 708 minutes) as outliers.

symmetric2). This situation is not a desirable one since extreme warpings are rarely realistic. Therefore, we need to be able to choose also among step patterns, like *symmetricP05*, that are more realistic for these applications; the drawback is that such step patterns can align sequences with a difference in length not too big. More precisely, the maximum p-parameter, allowed for the alignment of two series with length N and M respectively, is:

$$p_{max} = \frac{min(N, M)}{|N - M|} \tag{4.1}$$

The global maximal p-parameter for a given data set to be aligned is the minimum of the p_{max} computed over all the batches aligned against the reference one, which we introduce in the next section.

4.2.2 Reference batch selection

When aligning a time series data set via DTW, we need to select a single reference batch: this batch should represent a typical realization of the process under examination, both in terms of duration and evolution of the PVs. After reviewing the relevant literature (Spooner, Kold, and Kulahci 2017) and discussing it with domain experts, we decided to select as reference batch the one with median duration.

The reference batch is specific to a given data set, and since we will evaluate the proposed system on three different data sets, one for each year considered, we will have a different reference batch for each year.

The other batches in the data set are usually called query batches.

4.2.3 PVs selection

In Section 4.1, we said that it is possible to have different PVs collected for different batches. This situation is the first possible issue to take into account after selecting the reference batch. Another possible source of problems is that, from the DTW point of view, a constant PV gives no warping information: actually, it can be detrimental to the goodness of the alignment. We deal with these two issues in the following way: first of all, we remove from the reference batch all the constant PVs. This way, the reference batch will contain only PVs with warping information. Then we consider the other batches in the dataset and retain only the PVs left in the reference batch. If a batch does not have all the PVs present in the reference batch, it is considered not useful and removed from the dataset.

These actions can cause discarding some batches: we will give details on the exact number of batches remaining and the number of PV used in Chapter 6 when evaluating the proposed system.

4.2.4 PVs rescaling

DTW requires the choice of a distance measure to compute the distance matrix on which the whole algorithm rests. Many distance measures, such as the Euclidean distance used in this work, are sensitive to the range of values of the features. We, therefore, need to rescale the features in such a way that no one of them has an

unwanted high relevance in the computation of the distance between two points. Since the system proposed in this work aims at on-line application, the selection of the rescaling method is non-trivial. Since we use DTW also off-line, we could think of using standard methods to cope with this problem in this stage, like the standard normalization (subtracting the mean and dividing by the standard deviation). This choice is not feasible for two reasons: the data are not normally distributed, and the mean and the standard deviation are quantities that depend on the number of sampled data points (Kassidas, Taylor, and MacGregor 1998). Two batches that differ only for the duration of a flat portion of a PV will have a very different mean and standard deviation, even though from the DTW perspective, they are essentially the same. This fact implies that a better choice of normalization is scaling the PVs to the $[0, 1]$ interval, subtracting the minimum value, and dividing by the range $\max - \min$. Minimum and maximum value could refer to the batch under consideration, but this choice would be problematic in the on-line scenario. The choice made in this work is to scale every PV in the data set, considering its range in the reference batch. In this way, on-line normalization requires no additional consideration. The negative effect of such choice, if present, will be mitigated when optimizing the weights of the variables.

4.3 DTW configuration

The high degree of flexibility of DTW makes it suitable for a variety of problems, at the expense that it needs to be configured appropriately to perform well. The two main features of the algorithm to configure, after selecting the general set up (standard or open-ended version, feature space) are the variables weights and the step pattern to use for alignment.

Variables weights should reflect the importance of each feature for the alignment process. The selected step pattern should not result in extreme warpings, as mentioned in Section 2.4, but should nevertheless minimize as much as possible the DTW distance, to ensure proper alignment.

To perform such configuration, we follow closely two procedures found in the literature. For variables weight optimization, we follow (H.-J. Ramaker et al. 2003), apart from some minor changes that we will point out later. For what concerning step pattern selection, we follow the procedure suggested in (Spooner, Kold, and Kulahci 2017).

We describe the complete optimization procedure in the Appendix. Here, we limit ourselves to describe the guiding concept of such optimization.

For what concerning the weights of the variables, we select them giving high weights to features containing more warping information; that is, the warping of such features alone has a strong influence on the warping path resulting from the alignment.

The selection of the step pattern emerges from minimizing the combination of two measures: DTW distance and time distortion. We select a step pattern that compromises better the goodness of the alignment (DTW distance), but that does not deform excessively the series (time distortion).

For more information about these procedures, we refer to the paper cited above

and the Appendix.

This chapter described the data that enters the system described in the next chapter and the configuration of the DTW algorithm it uses.

The next chapter will describe in detail every component of the system.

Chapter 5

Proposed System

This chapter describes two systems called the SA-system and the SA+DTW-system. The SA-system represents the direct application of survival analysis to the problem of predicting the time-to-end of a running batch using all the available historical information. The SA+DTW-system adds a data-selection step to the SA-system, performed via DTW: the information contained in the warping path is used to select only a small fraction of the original historical data that is used to train the SA part of the system.

5.1 SA-System

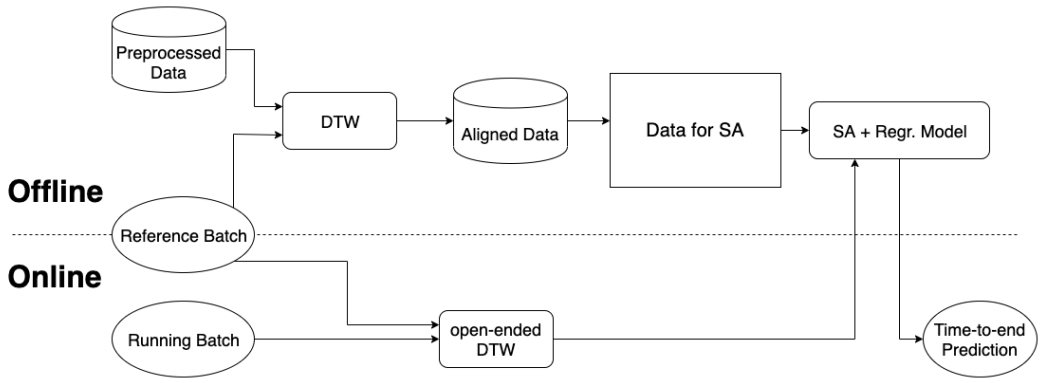


Figure 5.1. Architecture of the SA-system. The historical data about previous batches is aligned and then used to train a SA-ML model, that is then used online to give a time-to-end prediction based on the current state of the running batch

We call SA-system the architecture presented in Figure 5.1. It mainly consists of an off-line phase where the preprocessed data is prepared via DTW alignment, and the ML part of the system is trained. The ML part consists of a SA-ML-based model followed by a regression model that converts the output of the SA-ML-based model, a risk score, to an actual time-to-end estimate. All the data available about previous batches, once aligned, are used for this training phase. The result is a trained model that is then used in the on-line phase to obtain a time-to-end prediction for the

running batch based on its current status.

In this section, we describe each component appearing in the architecture of Figure 5.1. Section 5.2 describes the composition of the other architecture proposed, the SA+DTW-system, that has some components in common with the one discussed here. Some choices somewhat arbitrary in this system were made keeping in mind the same choices made for the SA+DTW-system, where coherence between on-line and off-line methods required certain choices.

Preprocessed data and reference batch

The preprocessed data is what we described in Chapter 4. Every batch instance is a multi-variate time series represented as a matrix, with columns denoting the different PVs and rows representing samples at each time point. The reference one is the one selected according to the principles discussed in Section 4.2.2.

The preprocessed data and reference batch are components in common between the two architectures described in this chapter.

DTW

This off-line DTW component represents the DTW alignment performed between the preprocessed data and the reference batch. The effect of the alignment is to add two features to the samples in data: the DTW distance and the time index of the mapped point on the reference batch.

In principle, these two features would not be required to apply the SA-ML model, but in order to make the comparison with the SA+DTW-system coherent, we add them in this case too.

The DTW version to be used, standard or open-ended, is a matter of choice. We use the open-ended version on each prefix of the old batches to be coherent with the on-line choice.

Aligned data

This block refers to the data aligned via DTW. As said before, it contains both the DTW distance and the mapped index on the reference batch.

Data for SA

This component of the system represents the data to be used to train the SA-ML model. It consists of the aligned data, thus containing both the information about the PVs values and the one returned by the DTW application (DTW distance and mapped index on the reference batch). These data represent the independent variables, while the response variable consists of the time until the end of the batch.

SA + Regression model

This block encloses the predictive components of the system. We first have the trained SA-ML model that, for the reasons discussed in Section 3.3, cannot directly output a time to end prediction. Instead, the output of the SA model is a risk score

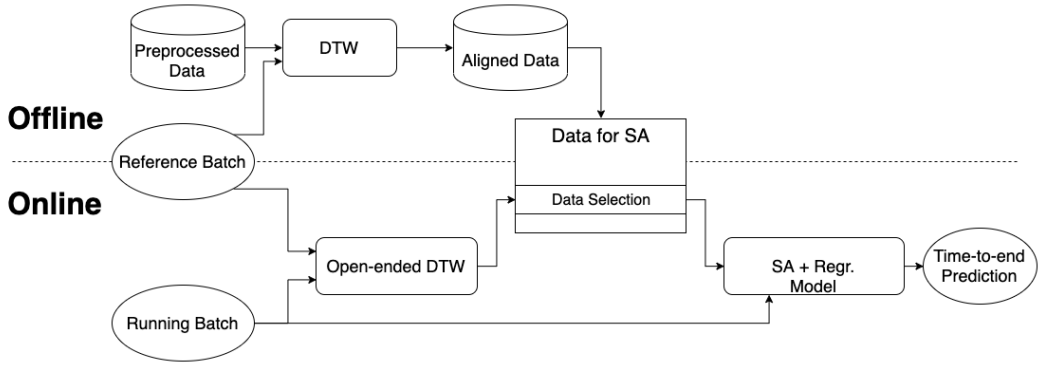


Figure 5.2. Overview of the SA+DTW system

on an arbitrary scale that depends on the data used to train the model. The strategy adopted here is to train the SA model on all the available data and then create a new data set with the predicted risk score for all the data available. This risk score will represent the independent variable used to train the following regression model, in our case, a Random Forest model, with the time until the end of the batch as the response variable. The output of this block is, therefore, a time-to-end prediction.

Running batch and Open-ended DTW

The running batch is virtually the only quantity handled on-line in the SA-system. The running batch is first aligned to the reference batch via open-ended DTW, in order to get the DTW distance and mapped index on the reference batch information needed for coherence with the SA model trained off-line. After this, the data that enters the SA+Regr. Model block is only the current state of the running batch, namely the PVs values at the time considered, together with the information from the DTW alignment. The SA+Regr. Model block processes this single data point and returns a time-to-end prediction.

Time-to-end prediction

This block represents the output of the system: an estimate of the time-to-end of the running batch.

5.2 SA+DTW-System

We call SA+DTW-system the architecture schematically represented in Figure 5.2. Unlike the SA-system described in the previous section, the system presented here has an on-line phase much relevant and articulated. This section describes each element of both phases, referring to the previous section for blocks in common between SA+DTW-system and SA-system. When necessary, we highlight some choices that we made for general coherence of the system, but that could be somewhat arbitrary in the SA-system.

Preprocessed data and reference batch

These two blocks are the same as the ones described in the previous section, representing respectively the information about previous realizations of the formula and the batch, considered the standard one, to which we compare the others.

DTW

This block is again a common one with the SA-system, and is used to align the historical data to the reference batch and add the DTW alignment information, namely DTW distance and mapped index on the reference batch. This mapped index is of crucial importance for the SA+DTW system, as we will see when describing the *Data-selection* block.

Aligned data

This block refers to the data aligned via DTW, as for the SA-system.

Data for SA

As for the SA-system, these data consist of the DTW aligned data to which we add the response variable representing the time until the end of the batch. This block is put between the off-line and on-line sections of the diagram because the whole data set is realized off-line, but the data that is used by the SA-ML model is selected on-line; The selection depends on the state of the running batch. The description of the on-line phase that follows this block, and in particular, the *Data-selection* block, will make this point more clear.

Running batch

Unlike what happens in the SA-system, this block is only one of the various quantities that enter into play in the on-line phase of the SA+DTW system. Its role is fundamentally different in this system, even though the operations performed on it are the same. The running batch is aligned to the reference one via open-ended DTW to obtain the DTW distance and the mapped index on the reference batch, a fundamental quantity in this system. Then, when it comes to the prediction phase, the single data point representing the current state of the running batch enters the predictive block, described below.

Open-ended DTW

This block is the same as the one in the SA-system and serves to align the running batch to the reference one, actually to a prefix of it, returning as information the DTW distance and the mapped index on the reference one. This mapped index is the crucial information that enters the next block, the *Data-selection* one.

Data-selection

This block represents the main difference between the SA-system and the SA+DTW one. Once the running batch has been aligned via open-ended DTW to the reference batch, the information about the mapped index on the reference batch is used to select, from the *Data for SA*, only the fraction that, off-line, was mapped to the same index on the reference batch. This use of DTW represents the new contribution of this work: the use of DTW not as a time normalization tool or as a distance measure between time series, but as a means to filter relevant data for time-related analysis.

SA+Regression Model

In the SA-system, put this block in the off-line portion of the diagram. For its content and general functioning, we refer to what written in the previous section about the SA-system. Here we limit ourselves to highlight the fundamental difference that the data used to train the SA-ML model, in this case, is not represented by the whole dataset *Data for SA*, but only by the fraction selected by the block *Data-selection*. Therefore, it is clear that the training of both the SA model and the following regression model, the same random forest model as in the SA-system, is performed every time a new data point from the running batch is received.

Time-to-end prediction

This block is the output of the whole system, an estimate of the time-to-end of the running batch.

5.3 Rationale behind the SA+DTW-system

Why should we not use all the available data to predict the time-to-end of the batch? This question could arise after understanding the different functioning of the two systems. We can think about the SA-system as of a standard ML predictive system: a model trained on a historical data set that is used to predict a certain quantity of interest. While experimenting with the SA-system, we realized that there were some issues we could try to remove with the help of DTW.

First of all, the training of the model on the whole data set available is computationally expensive: as we will see in the results chapter, for the data set at hand the training of the model can take up to hours just for the training, and it results in a model that also takes some time to return a prediction given a new data point. This slowness could be a problem at the moment we have new data available; for example, after completion of a batch, it automatically should become part of the historical data. This addition would be a problem if every time we complete a batch, we have to train a whole new and computationally heavy model.

In addition to the above computational issue, we hypothesized the following about the information contained in the data: considering an old batch, the information about the time until the completion of the running batch is unlikely to be evenly spread across the whole old batch. DTW comes to help at this point: its ability to map time fluctuations helps us identifying the points on the old batches more relevant

to the current state of the running batch. The identification is made possible by the warping path resulting from the alignment of the running batch with the reference one. We use the last point of the warping path as a mapping from the running batch to the reference batch, and from this one to the whole set of old batches. The filtered data points are the ones that DTW relates to as optimally aligned to the current state of the running batch. We expect that such data contain enough information to return a reliable prediction of the time-to-end of the running batch. With reliable, we mean that the predictive performance of the SA+DTW-system is not worse than the performance of the SA-system, which we could say, uses noisy data.

On the computational side, the training of the *SA + Regr. model* for each new data point from the running batch could be seen as too much overhead for the system doing the computation. As we will see in the results section, the opposite is true: the filtered data set is so small that the training and prediction steps take just seconds, perfectly acceptable for a system with *proper time* (the rate at which new data points arrive) of one minute.

In this chapter, we described the two systems proposed in this work. The SA-system is simply an application of survival analysis to the problem of predicting the time until the completion of a batch. It has very few differences when compared to a standard machine learning predictive system. The SA+DTW-system is more sophisticated and incorporates a data-selection stage that exploits DTW to select data points that are more related from a time-wise perspective to the current state of the running batch. Finally, we have explained how the idea of the SA+DTW system was born, and the metric we will use to test its usefulness.

The following chapter evaluates and compares the performance of both systems in terms of predictive accuracy and computational cost.

Chapter 6

Results

This chapter describes both the experimental setting in which the systems proposed were tested and the results obtained, through which we compare them. The data we refer to in this chapter were described in Chapter 4. We describe the results separately in terms of predictive and computational performance. The former describes how accurately the systems predict the time-to-end of a running batch. The latter describes the computational effort of the two approaches.

6.1 Experimental setting

As we illustrated in Chapter 4, the data we use to test and compare the systems come from a chemical batch line production and span three years of production (from 2015 to 2017).

To have a more variegate testing scenario, we consider each year separately to test the two systems. A choice like this one allows not only to have a general idea of the performance of the systems but also to compare it among different groups of data, since the data from different years show different characteristics, as described in Chapter 4 as well.

For each year, we consider approximately the first two-thirds of the batches as the historical data and test the systems on the remaining third of batches. We show the resulting situation for each year in Table 6.1 below.

The experiments have been carried out on the same desktop machine (Quad-core, 2.90 GHz, 16 GB of memory).

To measure the predictive performance of the systems, we consider the mean absolute deviation of the prediction from the actual time-to-end of the batch. On the other hand, computational performance is not straightforward to measure and compare since the structure of the computations for the two systems, both off-line and on-line, is different. We give the results obtained about these two performance metrics

Table 6.1. Number of train and test batch per year

Year	2015	2016	2017
Number of train batches	80	101	75
Number of test batches	40	50	37

in the next two sections. The last section of the chapter gives some considerations that will be useful for the discussion carried out in the last chapter of this work regarding conclusions drawn from the results and possible future research directions.

6.2 Predictive performance

This performance measure refers to the ability of a system to estimate the quantity of interest accurately. There exist many possible metrics to quantify such accuracy: we chose the absolute error, which is the absolute value of the deviation of the predicted value from the actual time-to-end of the batch. We average the errors over all the batches in the test set.

Simple average and SA-system

Before comparing the performance of the two systems proposed in this work, we have first to point out the base level chosen as reference. In the literature, we found no use of SA for the specific problem of determining the time-to-end of a batch process. Therefore, in the first place, we have to determine the feasibility of using such an approach. As a baseline for the predictive performance, we chose a simple average method: the estimated time-to-end of a batch in the test set is given by the average time-to-end of the train batches longer than the current length of the test batch. For example, the prediction at time $t = 200$ for a test batch will be the average of the time-to-end of the train batches that last more than 200 minutes.

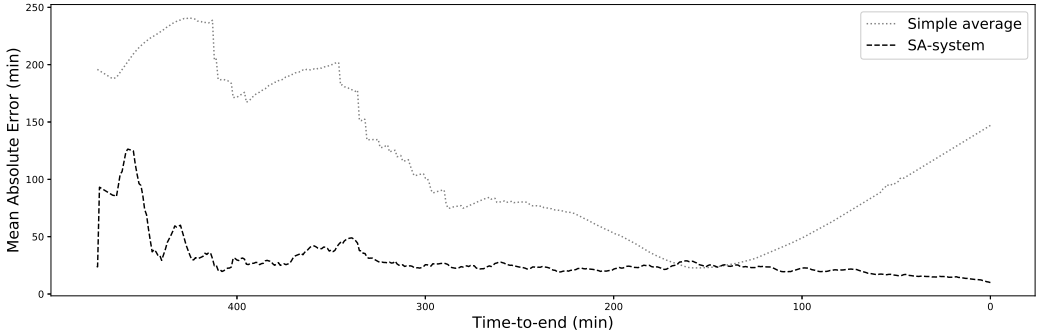
We consider this simple method the base performance any system has to meet at least to be considered useful.

Therefore, the first comparison to do is between this simple average method and the SA-system. Figure 6.1 shows this comparison, with the absolute error averaged over all the batches in the test set. We note here that, both for the SA-system and the SA+DTW-system, the actual prediction is the average of the predictions from the last 10 minutes: this averaging reduces the fluctuations due to the instability of the open-ended DTW mapping; we discuss this issue in the next chapter.

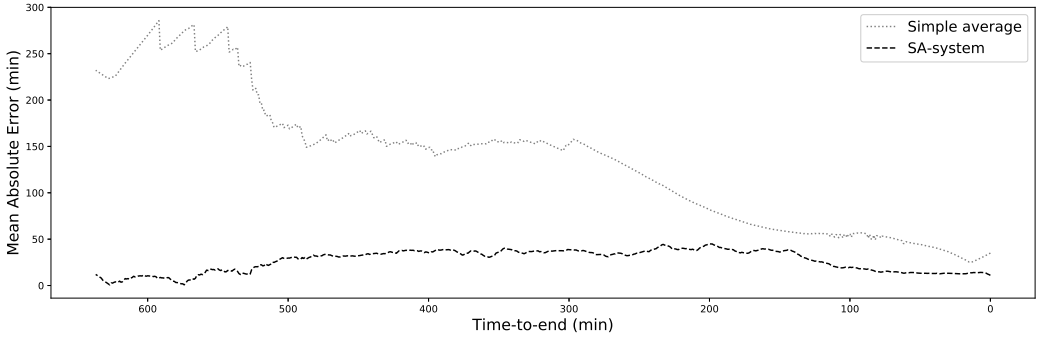
Some remarks are necessary for the plots in Figure 6.1. First of all, we note that the x-axis refers to the time-to-end of the test batches, therefore decreasing from left to right. Since we have batches with different duration, the number of points over which we compute the mean absolute error varies. Explicitly, it increases from left to right: we start from the single point representing the starting point of the most extended batch, arriving at the last point of all the batches. This difference in numerosity of data explains the high variability of the left-most part of the plots, where fewer points are available. Second, we note how the simple average method error increases towards the end of the batches. This effect is due to the overestimation of the duration of short batches: by construction, this method considers all the batches longer than the one considered. For the shortest batches, we have a high number of longer batches that will contribute more to the prediction, with high estimates.

Now, we can comment on the quality of such results.

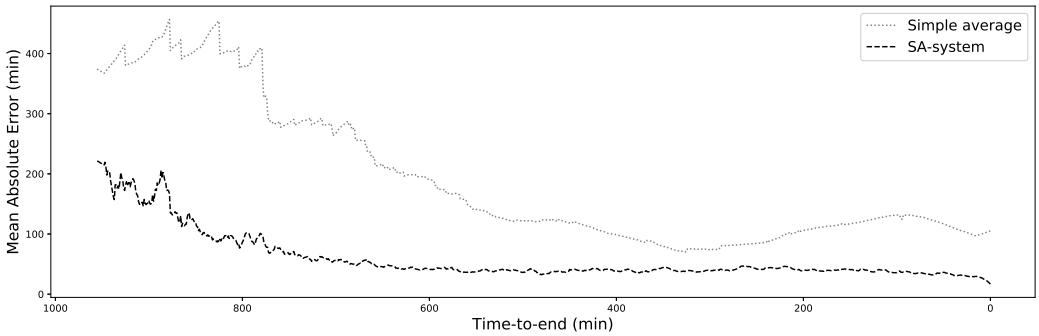
- Except for a small portion of 2015, the SA-system systematically outperforms



(a) 2015



(b) 2016



(c) 2017

Figure 6.1. Predictive performance of the SA-system over the 3 years examined. The systems passes the test of the comparison with the simple average predictive method.

the simple average method. This performance reassures us that SA captures at least some of the information contained in the data related to the time-to-end of the batches.

- The results from 2016 show an unexpected initial trend, especially compared to the other two years. This trend is probably just a fortunate case, where the predictions are excellent even though we are still quite far from the end of the batch.

- We expect the high variability in the left-most portion of the plots from what said above about the number of points taken into account at each time instant. All the trends stabilize proceeding to the right, towards the end of the batches.

Overall, the performance is satisfactory: for all the years, the average error is significantly smaller than the actual time-to-end.

SA+DTW-system

The above description of the predictive performance of the SA-system told us that it is feasible to use SA for the problem of predicting the time-to-end of a running batch. Now we investigate the performance of the SA+DTW-system from the same predictive performance point of view to see whether a loss of accuracy occurs, selecting only a fraction of the data via DTW.

Figure 6.2 adds the results from the SA+DTW-system to what previously shown in Figure 6.1.

It is possible to appreciate a durable consistency of performance, especially for the year 2017, where the performance of the SA+DTW-system is systematically better than the SA-system. For the year 2016, there is a relevant loss of performance only for large values of time-to-end, while for the year 2015, the performance stays consistent for the whole time.

These results show that, at least from the predictive point of view, there is no downside in applying the DTW data-selection step. The data selected via DTW in the SA+DTW-system contain enough information to perform as good as the SA-system, which uses all the data available.

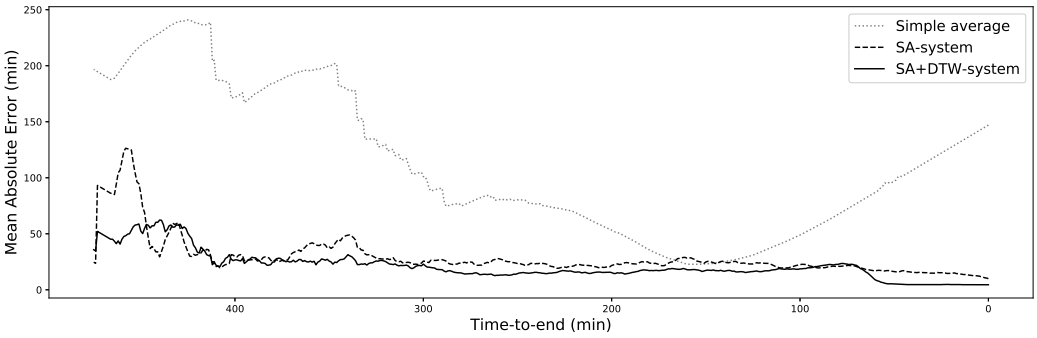
What remains to asses is the computational performance of the two systems, which we describe in the next section.

6.3 Computational performance

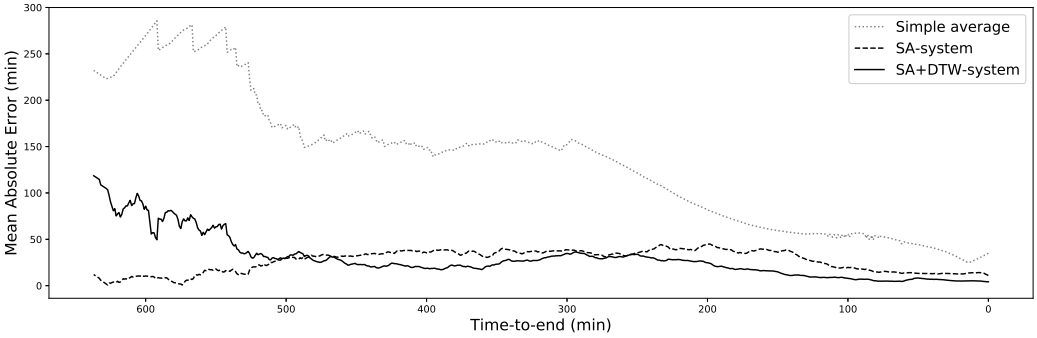
We measure the computational performance of the two systems by the time taken to perform the computations of interest. We run all the experiments on the same machine in the same conditions, so a direct comparison is possible.

The main difference between the two systems, SA and SA+DTW, respectively, is that in the former one, the training of the predictive model happens only once, off-line, while in the latter, the predictive model has to be trained for every new prediction performed.

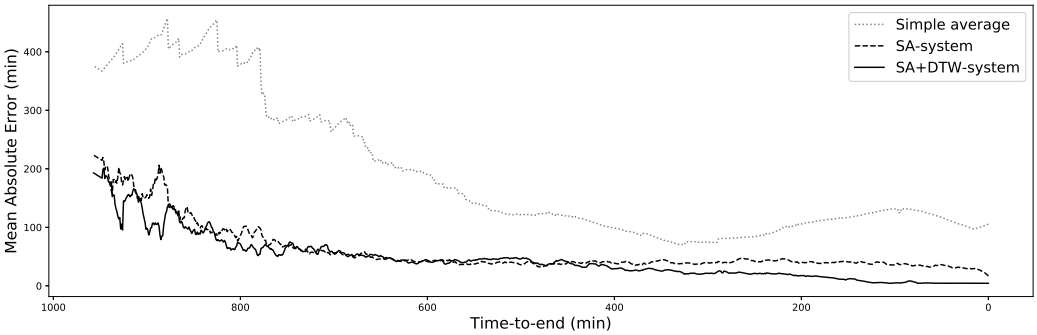
Table 6.2 shows the quantities of interest for the SA-system, divided by year: the size of the data set used, the time taken for training the predictive model, and the



(a) 2015



(b) 2016



(c) 2017

Figure 6.2. Comparison of the predictive performance of the two systems together with the simple average method performance. The SA+DTW-system shows performance as good as the one of the SA-system most of the time, with the only relevant exception being for the initial section of long batches in 2016. Overall, the results are satisfactory in every period.

Table 6.2. SA-system: data set size and computational performance

Year	2015	2016	2017
Training dataset size [rows × columns]	39431×28	77989×34	79833×34
Training time [minutes]	34	126	134
Single prediction time [seconds]	~ 0.6	~ 1.1	~ 1.1

Table 6.3. SA+DTW-system: average data set size and computational performance

Year	2015	2016	2017
Average dataset size [rows × columns]	149×28	180×34	131×34
Single prediction time [seconds]	< 1	< 1	< 1

time needed for predicting a single new data point. The most relevant information here regards the training time; it ranges from 34 minutes for the smallest dataset to more than 2 hours for the others. This training time could be acceptable if this operation would be required only once, but it needs to be performed every time we want to add the information obtained from the completion of new batches.

Table 6.3 shows the computational performance for the SA+DTW-system. The quantities of interest are different here, we have no off-line training, and the size of the data set used by the predictive model for the on-line training changes at every new prediction, so we only give its average size. The single prediction-time value includes the time for the on-line training of the model and the time needed to return a prediction.

First of all, we note that the time needed for a single prediction is constant and comparable to what obtained with the SA-system. The average data set size indicates that, on average, we use less than 1 percent of the data available, nevertheless the predictive performance is virtually unaltered, as we saw in the previous section.

From the computational point of view, the SA+DTW-system shows its strength: the on-line performance is unaltered, while there is no need to spend hours on off-line training. In addition to this, we can add new information without any significant issue, without any need for re-training: the system will choose the new data points, if necessary, and use them for the on-line training.

In this chapter, we presented the results obtained by the two systems on the data described in Chapter 4. The results of the predictive performance show that there is no relevant difference between the two systems. On the other hand, the results on the computational performance show that the SA+DTW-system should be preferred, avoiding the expensive off-line training.

In the next chapter, we draw some conclusions about these results, motivating our belief that the SA+DTW-system can be improved with further research and experimentation, tuning some aspects of the data-selection step and the alignment of the historical data.

Chapter 7

Conclusions

We discuss the implications of the results obtained in the previous chapter and how they support the hypothesis presented in this work. In particular, we explain how both the SA approach and the DTW data-selection point of view could benefit the batch process monitoring practice. We then point to some possible future research directions.

7.1 Survival analysis in batch process monitoring

As we saw in the brief review of current practice in batch process monitoring of Section 2.6, the main focus of process monitoring is on fault detection and identification. In this scenario, the introduction of survival analysis represents a novelty for the field in which we apply it, batch process monitoring.

The results obtained in the previous chapter allow us to at least propose its use in such a scenario. The performance obtained was considered acceptable by the domain experts consulted, but this depends on the specific application and its requirements and costs.

As we pointed out in Chapter 3, the use of SA is of interest over other regression models, given its ability to use censored data. The focus of batch process monitoring on fault detection is not by accident: faults that lead to the loss of a batch are not uncommon. The use of SA allows using data about incomplete batches too.

7.2 DTW as data-selection tool

The application of DTW as a data-selection tool represents a new use of such technique. The main problem DTW solves is time normalization and event synchronization; we expanded its applicability, exploiting the information contained in the warping path. Such a mapping function turns out to be able to select the relevant data to the time-related problem of estimating the time-to-end of the running batch. Such information is completely hidden when applying DTW directly, but it is a useful quantity that, as we have seen, is not entirely unrelated from the alignment performed by DTW.

We note that the DTW configuration used in this work is a basic one, apart from the optimal configuration of weights and step-pattern. In the next section, we

point out some possible improvements.

7.3 Further research

As pointed out in the previous section, the DTW alignment proposed in this work is quite basic. There are some issues with such an essential version, especially regarding the open-ended application: firstly, the warping path obtained for the same running batch at two subsequent time instant can significantly differ since the algorithm is sensitive to fluctuations in the values of the PVs. Secondly, an alignment based only on the values of the PVs could be sub-optimal.

The first issue pointed out could be handled in several ways: for example, a low-pass filter could be applied to the data to remove fast fluctuations. A more elegant approach could be using one of the more recent developments of DTW, like shape-DTW (Zhao and Itti 2018) and derivative-DTW (Keogh and Pazzani 2001). These forms of the algorithm do not align the time series based only on the values of the PVs: the former matches local patterns of the series, while the latter matches the points considering the value of the discrete derivative computed at that point too. Such versions of the algorithm could solve simultaneously the two issues above, returning a more stable open-ended mapping and improving the data-selection step.

The high flexibility of DTW allows for a wide range of possible optimizations. In this work, we focused on the fundamental idea, giving enough evidence to stimulate the research of batch process monitoring in this sense, expanding the possible applications of an already widely used algorithm.

Chapter 8

Appendix

8.1 Software used

We used the Python (Rossum 1995) programming language to develop the whole system. The major Python packages we used are:

- Numpy (T. E. Oliphant 2006)
- Scikit-learn (Pedregosa et al. 2011)
- SciPy (Jones, T. Oliphant, Peterson, et al. 2001–)
- Matplotlib (Hunter 2007)
- Pandas (McKinney et al. 2010)
- Scikit-survival (Pölsterl, Navab, and Katouzian 2015; Pölsterl, Navab, and Katouzian 2016; Pölsterl, Gupta, et al. 2016)

8.2 DTW configuration

The high flexibility in the configuration of DTW is a great advantage in terms of adaptability to different domains and problems, but it also has some drawbacks: standard choices could be non-optimal for the problem under consideration, and in order to guide a rational choice of the various parameters it is usually required an amount of process knowledge not always available or difficult to acquire (H.-J. Ramaker et al. 2003).

In this section, we introduce a two-step procedure to configure two aspects of the algorithm optimally:

- Optimization of variable weights that leads to an estimate of the relative importance of the different PVs for what concerning the alignment of the data set.
- constraint selection used to tune the strength of the possible warping induced by the algorithm.

8.2.1 Variable weights

The standard choice for the distance measure $d(\cdot, \cdot)$ introduced in Equation (2.3) when dealing with real valued time series data is the euclidean distance.

Let $\boldsymbol{y_i}$ and $\boldsymbol{x_j}$ be respectively two data points (considered as column vectors) of the n-variate time series $\boldsymbol{Y}$ and $\boldsymbol{X}$; the euclidean distance between these two points can be written in matrix form as:

$$d(i, j) = [\boldsymbol{y_i} - \boldsymbol{x_j}]\boldsymbol{W}[\boldsymbol{y_i} - \boldsymbol{x_j}]^T \tag{8.1}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with W_{mm} being the weight for the m-th feature. Features with a relatively large weight will contribute more to the distance and will, therefore, have a greater influence on the resulting alignment (H.-J. Ramaker et al. 2003).

The act of assigning weights to the single features could be guided, in principle, by process knowledge. The problem is that usually, the number of features is so high (tens or even hundreds) to make it impossible to have enough prior information to make optimal, or even coherent, choices.

Other options based on iterative procedures not requiring prior information but using only information available in the data set have been introduced in the literature. Kassidas et al. (Kassidas, Taylor, and MacGregor 1998) suggested an iterative algorithm focused on the consistency of a variable from process to process. The algorithm can be summarized as follows:

Step 1 Initialize all variable weights to 1, warp, and synchronize every query process with the reference one.

Step 2 From the synchronized trajectories, calculate the average trajectory.

Step 3 For each process variable of each process (queries and reference), compute the euclidean distance with the corresponding variable of the average trajectory. Sum them to obtain a single value for each process variable.

Step 4 Update the variable weights with the reciprocal of the number computed in the previous step and then normalize the set of weights such that they sum to the number of features.

Step 5 Repeat all the steps above with these new weights.

The algorithm stops after a predefined number of iterations (the number suggested by Kassidas et al. is 10) or after a (not specified in the cited article) convergence criterion.

It is clear from the explanation above that high weights will be assigned to variables that are close, in terms of euclidean distance, to the average trajectory, regardless of the shape of the process variable at hand. Variables with a flat or close to flat trend, with a constant value throughout the processes that clearly contains little information about the process and the alignment, will receive high weights, and that is something to avoid. It is safer and more natural to assume that the most warping information is contained in those variables with a peculiar shape, not in the flat ones. As an extreme case to make this point clear is the alignment of

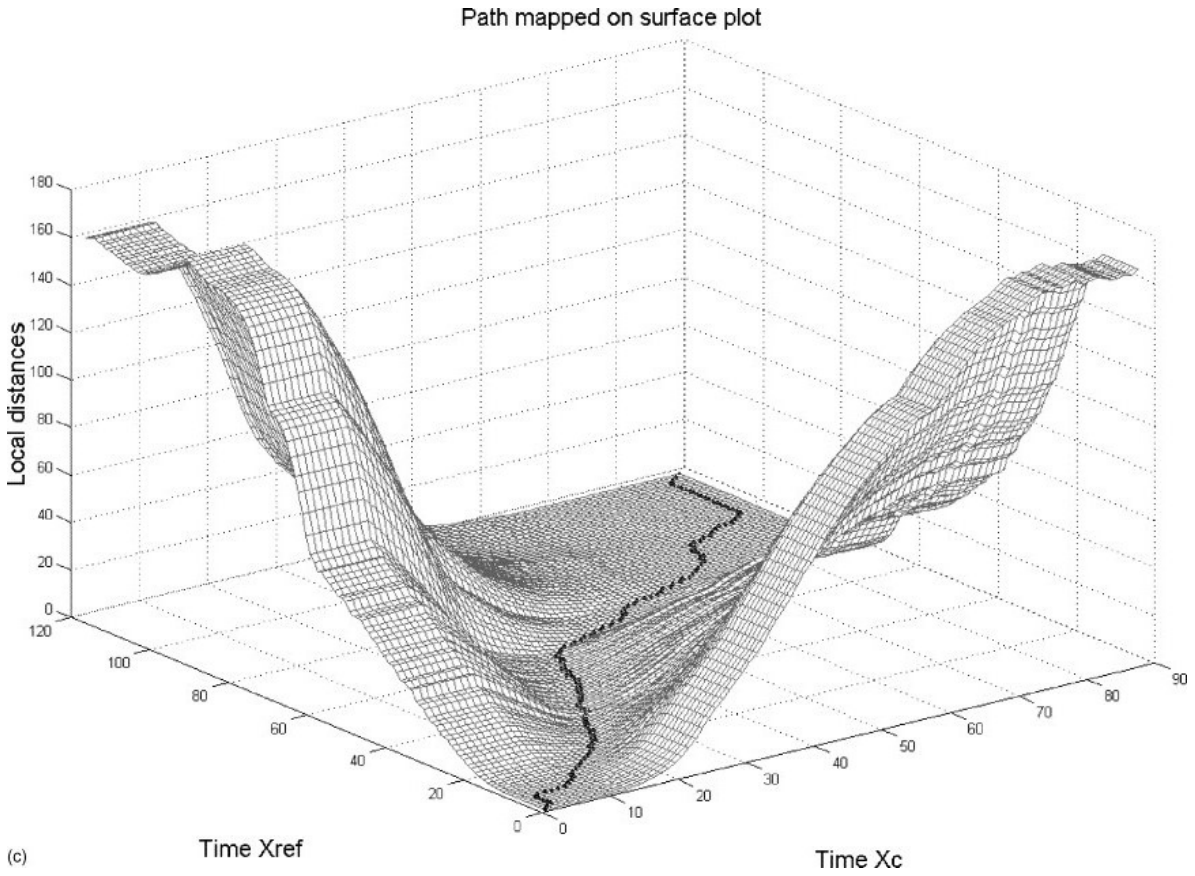


Figure 8.1. Warping path mapped on surface plot of local distances (H.-J. Ramaker et al. 2003)

two constant variables. Any warping would align them, regardless of length and magnitude.

In order to avoid this problem, Ramaker et al. in (H.-J. Ramaker et al. 2003) suggest giving high weights to variables that present a steep valley in the surface plot of the local distance matrix in correspondence of the warping path. This concept is visualized in Figure 8.1, taken from the article cited.

The iterative algorithm proposed by Ramaker et al. in (H.-J. Ramaker et al. 2003) involves the following steps:

Step 1 Initialize all variable weights to 1, warp, and synchronize the first query process with the reference one, collecting the warping path.

Step 2 For each process variable, fill the local distance matrix.

Step 3 Map the warping path collected in the first step onto each local distance matrix and compute the updated weight corresponding to each variable as follows:

$$w_{ii} = \frac{MLD_i(off\ path)}{MLD_i(on\ path)}$$

where MLD_i stands for Mean Local Distance (i-th variable), and in particular:

- $MLD_i(off\ path)$ is the mean local distance computed on all the points of the local distance matrix, not belonging to the warping path.
- $MLD_i(on\ path)$ is the mean local distance computed on the points of the local distance matrix belonging to the warping path.

If the warping path follows a valley of the local distance matrix (seen as a surface plot), then the $MLD(off\ path)$ for that variable will sensibly exceed the $MLD(on\ path)$, resulting in greater weight.

Step 4 Repeat all preceding steps on all the other processes contained in the data set with the updated weights (normalized to sum to the number of variables). Going through every process in the data set counts as a single iteration of the algorithm.

The algorithm stops after a predefined number of iterations or after reaching convergence. In the thesis project, the chosen criterion is

$$\frac{|\boldsymbol{w_j} - \boldsymbol{w_{j-1}}|_2}{|\boldsymbol{w_{j-1}}|_2} < 0.01$$

where $\boldsymbol{w_j}$ is the vector whose components are the variables weights at iteration j and $|\cdot|_2$ is the L_2-norm of the vector.

The application of the described article for variables weights optimization can serve multiple purposes:

- As preprocessing step, weight optimization allows reducing the dimensionality of the process. It is possible to set a threshold for the weights under which a variable is discarded in the alignment process, or alternatively, it is possible to select the top k variables with the highest weights. In both cases, the result is a data set with a lower dimensionality that, without any doubt, decreases the computational complexity of the alignment.
- To gain insights on the process and get feedback on the previous hypothesis: a domain expert could try to manually assign an importance score to the different variables based on his established expertise. It is then possible to compare his hypothesis with the weights returned by the algorithm. It could be the case that a variable considered an essential indicator of the overall state of the process, or its progress, is not optimal when used for alignment.

8.2.2 Step pattern

Selecting an optimal step pattern for the data set at hand can be of critical importance for the quality of the alignment, since we could end up with extreme warpings or no warping at all, depending on the strength of the constraint imposed to the algorithm. An optimally chosen step pattern would capture the time fluctuations in the process, resulting in a more meaningful alignment.

The procedure we used to select the optimal step pattern is thee suggested in (Spooner, Kold, and Kulahci 2017). In this work, the author considers two quantities describing a given alignment:

- The DTW distance resulting from the alignment, that is a direct measure of how close are the warped series.
- The time distortion of the alignment, a measure of how many times the alignment includes a warping step. In other words, for how many k's we have that $w_k - w_{k-1} \in \{(0,1),(1,0)\}$. Such a situation occurs when there is a stretch or compression in the warped time series.

When aligning two or more series, we want a small DTW distance to ensure proper alignment in terms of matched points, but we do not want this to be at the expense of unrealistic compressions and stretches in the series. To include both aspects into consideration, we select the step pattern that minimizes the alignment score. Considering a data set aligned to a given reference series, the alignment score is defined as follows

$$\text{Alignment Score} = \sqrt{\bar{N}^2_{Warps(scaled)} + \bar{D}^2_{(scaled)}} \tag{8.2}$$

Where:

- $\bar{N}_{Warps(scaled)}$ represents the scaled average time distortion of the aligned data set: we first compute the time distortion N^i_{Warps} for each series aligned series, then we scale the values to the $[0,1]$ interval obtaining the values $N^i_{Warps(scaled)}$, and then we compute their mean, $\bar{N}_{Warps(scaled)}$.
- $\bar{D}_{(scaled)}$ is the average scaled DTW distance, obtained similarly to the time distortion score: we compute the DTW distance for each aligned series, D^i; then we normalize them to the interval $[0,1]$ to obtain the values D^i_{scaled}, of which we compute the average value $\bar{D}_{(scaled)}$.

Given a set of possible step patterns, we compute the alignment score for each one of them and then select the one that minimizes it. Usually the set of possible step pattern comprises the non-constraining *symmetric1* and *symmetric2*, and the set of all allowed *symmetricPx*, for each x that allows to align every series in the data set to the reference one.

Bibliography

Gao, Xiang et al. (2001). "Multivariate Statistical Process Monitoring Based on Synchronization of Batch Trajectories Using DTW". In: *IFAC Proceedings Volumes* 34.27, pp. 333–337.

Giorgino, Toni et al. (2009). "Computing and visualizing dynamic time warping alignments in R: the dtw package". In: *Journal of statistical Software* 31.7, pp. 1–24.

Hothorn, Torsten et al. (2005). "Survival ensembles". In: *Biostatistics* 7.3, pp. 355–373.

Hunter, John D (2007). "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3, pp. 90–95.

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001–). *SciPy: Open source scientific tools for Python*. [Online; accessed: 2019-06-14]. URL: http://www.scipy.org/.

Kassidas, Athanassios, Paul A. Taylor, and John F. MacGregor (1998). "Off-line diagnosis of deterministic faults in continuous dynamic multivariable processes using speech recognition methods". In: *Journal of Process Control* 8.5. ADCHEM '97 IFAC Symposium: Advanced Control of Chemical Processes, pp. 381–393. ISSN: 0959-1524. DOI: https://doi.org/10.1016/S0959-1524(98)00025-0. URL: http://www.sciencedirect.com/science/article/pii/S0959152498000250.

Keogh, Eamonn J and Michael J Pazzani (2001). "Derivative dynamic time warping". In: *Proceedings of the 2001 SIAM international conference on data mining*. SIAM, pp. 1–11.

Lee, Elisa T and John Wang (2003). *Statistical methods for survival data analysis*. Vol. 476. John Wiley & Sons.

McKinney, Wes et al. (2010). "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX, pp. 51–56.

Müller, M (2007). *Information Retrieval for Music and Motion*. Springer.

Oliphant, Travis E (2006). *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct, pp. 2825–2830.

Pölsterl, Sebastian, Pankaj Gupta, et al. (2016). "Heterogeneous ensembles for predicting survival of metastatic, castrate-resistant prostate cancer patients". In: *F1000Research* 5.

Pölsterl, Sebastian, Nassir Navab, and Amin Katouzian (2015). "Fast Training of Support Vector Machines for Survival Analysis". In: *Machine Learning and*

Knowledge Discovery in Databases. Ed. by Annalisa Appice et al. Cham: Springer International Publishing, pp. 243–259. ISBN: 978-3-319-23525-7.

Pölsterl, Sebastian, Nassir Navab, and Amin Katouzian (2016). "An Efficient Training Algorithm for Kernel Survival Support Vector Machines". In: *arXiv preprint arXiv:1611.07054*.

Ramaker, H and Eric van Sprang (2004). "Statistical batch process monitoring". In:

Ramaker, Henk-Jan et al. (2003). "Dynamic time warping of spectroscopic BATCH data". In: *Analytica Chimica Acta* 498.1, pp. 133–153. ISSN: 0003-2670. DOI: https://doi.org/10.1016/j.aca.2003.08.045. URL: http://www.sciencedirect.com/science/article/pii/S0003267003011309.

Rossum, G. van (1995). *Python tutorial*. Tech. rep. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI).

Sakoe, Hiroaki and Seibi Chiba (1978). "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE transactions on acoustics, speech, and signal processing* 26.1, pp. 43–49.

Salvador, Stan and Philip Chan (2007). "Toward accurate dynamic time warping in linear time and space". In: *Intelligent Data Analysis* 11.5, pp. 561–580.

Spooner, Max, David Kold, and Murat Kulahci (2017). "Selecting local constraint for alignment of batch process data with dynamic time warping". In: *Chemometrics and Intelligent Laboratory Systems* 167, pp. 161–170.

Understanding Predictions in Survival Anlysis (n.d.). https://scikit-survival.readthedocs.io/en/latest/understanding_predictions.html. Accessed: 2019-06-14.

Wang, Ping, Yan Li, and Chandan K Reddy (2017). "Machine learning for survival analysis: A survey". In: *arXiv preprint arXiv:1708.04649*.

Zhao, Jiaping and Laurent Itti (2018). "Shapedtw: shape dynamic time warping". In: *Pattern Recognition* 74, pp. 171–184.