4

Dynamic Time Warping

Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions (Fig. 4.1). Intuitively, the sequences are warped in a nonlinear fashion to match each other. Originally, DTW has been used to compare different speech patterns in automatic speech recognition, see [170]. In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data.

In this chapter, we introduce and discuss the main ideas of classical DTW (Sect. 4.1) and summarize several modifications concerning local as well as global parameters (Sect. 4.2). To speed up classical DTW, we describe in Sect. 4.3 a general multiscale DTW approach. In Sect. 4.4, we show how DTW can be employed to identify all subsequence within a long data stream that are similar to a given query sequence (Sect. 4.4). A discussion of related alignment techniques and references to the literature can be found in Sect. 4.5.

4.1 Classical DTW

The objective of DTW is to compare two (time-dependent) sequences $X := (x_1, x_2, \ldots, x_N)$ of length $N \in \mathbb{N}$ and $Y := (y_1, y_2, \ldots, y_M)$ of length $M \in \mathbb{N}$. These sequences may be discrete signals (time-series) or, more generally, feature sequences sampled at equidistant points in time. In the following, we fix a *feature space* denoted by $\mathcal{F}$. Then $x_n, y_m \in \mathcal{F}$ for $n \in [1 : N]$ and $m \in [1 : M]$. To compare two different features $x, y \in \mathcal{F}$, one needs a *local cost measure*, sometimes also referred to as *local distance measure*, which is defined to be a function

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}. \tag{4.1}$$

Typically, $c(x, y)$ is small (low cost) if x and y are similar to each other, and otherwise $c(x, y)$ is large (high cost). Evaluating the local cost measure for

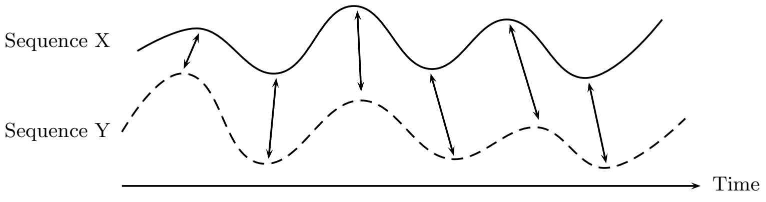

Fig. 4.1. Time alignment of two time-dependent sequences. Aligned points are indicated by the *arrows*

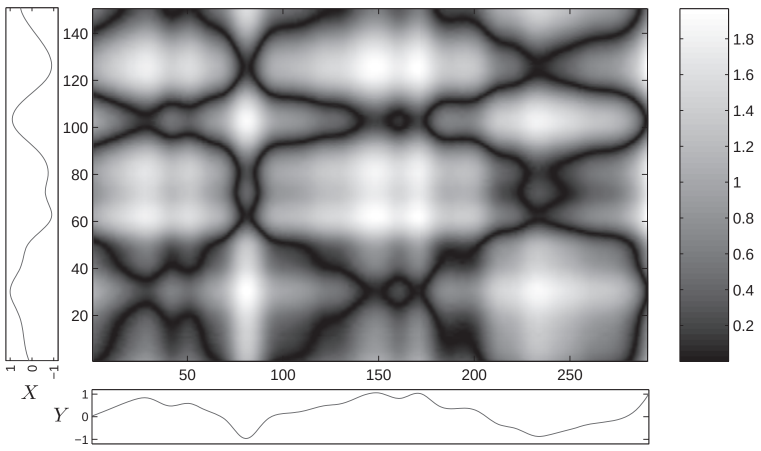

Fig. 4.2. Cost matrix of the two real-valued sequences X (*vertical axis*) and Y (*horizontal axis*) using the Manhattan distance (absolute value of the difference) as local cost measure c. Regions of low cost are indicated by *dark colors* and regions of high cost are indicated by *light colors*

each pair of elements of the sequences X and Y, one obtains the *cost matrix* $C \in \mathbb{R}^{N \times M}$ defined by $C(n, m) := c(x_n, y_m)$, see Fig. 4.2. Then the goal is to find an alignment between X and Y having minimal overall cost. Intuitively, such an optimal alignment runs along a "valley" of low cost within the cost matrix C, see Fig. 4.4 for an illustration. The next definition formalizes the notion of an alignment.

Definition 4.1. *An* (N, M)-warping path *(or simply referred to as* warping path *if* N *and* M *are clear from the context) is a sequence* $p = (p_1, \ldots, p_L)$ *with* $p_\ell = (n_\ell, m_\ell) \in [1 : N] \times [1 : M]$ *for* $\ell \in [1 : L]$ *satisfying the following three conditions.*

(i) Boundary condition: $p_1 = (1, 1)$ *and* $p_L = (N, M)$.
(ii) Monotonicity condition: $n_1 \leq n_2 \leq \ldots \leq n_L$ *and* $m_1 \leq m_2 \leq \ldots \leq m_L$.
(iii) Step size condition: $p_{\ell+1} - p_\ell \in \{(1, 0), (0, 1), (1, 1)\}$ *for* $\ell \in [1 : L - 1]$.

Note that the step size condition (iii) implies the monotonicity condition (ii), which nevertheless has been quoted explicitly for the sake of clarity. An (N, M)-warping path $p = (p_1, \ldots, p_L)$ defines an alignment between two sequences $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ by assigning the element x_{n_ℓ} of X to the element y_{m_ℓ} of Y. The boundary condition enforces that the first elements of X and Y as well as the last elements of X and Y are aligned to each other. In other words, the alignment refers to the entire sequences X and Y. The monotonicity condition reflects the requirement of faithful timing: if an element in X precedes a second one this should also hold for the corresponding elements in Y, and vice versa. Finally, the step size condition expresses a kind of continuity condition: no element in X and Y can be omitted and there are no replications in the alignment (in the sense that all index pairs contained in a warping path p are pairwise distinct). Figure 4.3 illustrates the three conditions.

The *total cost* $c_p(X, Y)$ of a warping path p between X and Y with respect to the local cost measure c is defined as

$$c_p(X, Y) := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}). \tag{4.2}$$

Furthermore, an *optimal warping path* between X and Y is a warping path p^* having minimal total cost among all possible warping paths. The *DTW distance* $\mathrm{DTW}(X, Y)$ between X and Y is then defined as the total cost of p^*:

$$\begin{aligned} \mathrm{DTW}(X, Y) &:= c_{p^*}(X, Y) \qquad (4.3) \\ &= \min\{c_p(X, Y) \mid p \text{ is an } (N, M)\text{-warping path}\} \end{aligned}$$

We continue with several remarks about the DTW distance. First, note that the DTW distance is well-defined even though there may be several warping paths of minimal total cost. Second, it is easy to see that the DTW distance is symmetric in case that the local cost measure c is symmetric. However,

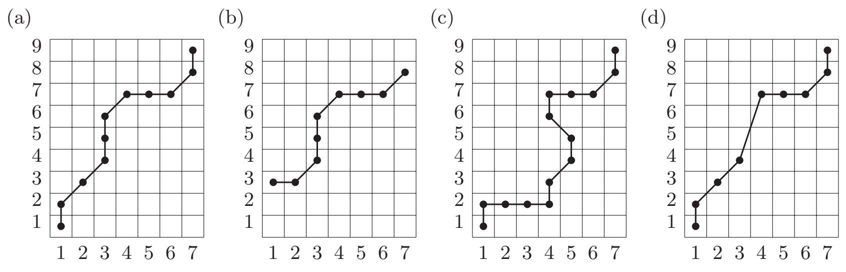

Fig. 4.3. Illustration of paths of index pairs for some sequence X of length $N = 9$ and some sequence Y of length $M = 7$. **(a)** Admissible warping path satisfying the conditions (i), (ii), and (iii) of Definition 4.1. **(b)** Boundary condition (i) is violated. **(c)** Monotonicity condition (ii) is violated. **(d)** Step size condition (iii) is violated

the DTW distance is in general not positive definite even if this holds for c. For example, one obtains $\mathrm{DTW}(X,Y)=0$ for the sequences $X:=(x_1,x_2)$ and $Y:=(x_1,x_1,x_2,x_2,x_2)$ in case $c(x_1,x_1)=c(x_2,x_2)=0$. Furthermore, the DTW distance generally does not satisfy the triangle inequality even in case c is a metric. This fact is illustrated by the following example.

Example 4.2. Let $\mathcal{F}:=\{\alpha,\beta,\gamma\}$ be a feature space consisting of three features. We define a cost measure $c:\mathcal{F}\times\mathcal{F}\to\{0,1\}$ by setting $c(x,y):=0$ if $x=y$ and $c(x,y):=1$ if $x\neq y$ for $x,y\in\mathcal{F}$. Note that c defines a metric on $\mathcal{F}$ and particularly satisfies the triangle inequality. Now consider $X:=(\alpha,\beta,\gamma)$, $Y:=(\alpha,\beta,\beta,\gamma)$, and $Z:=(\alpha,\gamma,\gamma)$. Then one easily checks that $\mathrm{DTW}(X,Y)=0$, $\mathrm{DTW}(X,Z)=1$, and $\mathrm{DTW}(Y,Z)=2$. Therefore, $\mathrm{DTW}(Y,Z)>\mathrm{DTW}(X,Y)+\mathrm{DTW}(X,Z)$, i. e., the DTW distance does not satisfy the triangle inequality. Finally, note that the paths $p^1=((1,1),(2,2),(3,2),(4,3))$, $p^2=((1,1),(2,1),(3,2),(4,3))$, and $p^3=((1,1),(2,2),(3,3),(4,3))$ are different optimal warping paths between Y and Z of total cost two. This shows that an optimal warping path is generally not unique.

To determine an optimal path p^*, one could test every possible warping path between X and Y. Such a procedure, however, would lead to a computational complexity that is exponential in the lengths N and M. We will now introduce an $O(NM)$ algorithm that is based on *dynamic programming*. To this end, we define the prefix sequences $X(1\!:\!n):=(x_1,\dots x_n)$ for $n\in[1:N]$ and $Y(1\!:\!m):=(y_1,\dots y_m)$ for $m\in[1:M]$ and set

$$D(n,m):=\mathrm{DTW}(X(1\!:\!n),Y(1\!:\!m)). \tag{4.4}$$

The values $D(n,m)$ define an $N\times M$ matrix D, which is also referred to as the *accumulated cost matrix*. Obviously, one has $D(N,M)=\mathrm{DTW}(X,Y)$. In the following, a tuple (n,m) representing a matrix entry of the cost matrix C or of the accumulated cost matrix D will be referred to as a *cell*. The next theorem shows how D can be computed efficiently.

Theorem 4.3. *The accumulated cost matrix D satisfies the following identities: $D(n,1)=\sum_{k=1}^{n}c(x_k,y_1)$ for $n\in[1:N]$, $D(1,m)=\sum_{k=1}^{m}c(x_1,y_k)$ for $m\in[1:M]$, and*

$$D(n,m)=\min\{D(n-1,m-1),D(n-1,m),D(n,m-1)\}+c(x_n,y_m) \tag{4.5}$$

for $1<n\leq N$ and $1<m\leq M$. In particular, $\mathrm{DTW}(X,Y)=D(N,M)$ can be computed with $O(NM)$ operations.

Proof. Let $m=1$ and $n\in[1:N]$. Then there is only one possible warping path between $Y(1:1)$ and $X(1:n)$ having a total cost of $\sum_{k=1}^{n}c(x_k,y_1)$. This proves the formula for $D(n,1)$. Similarly, one obtains the formula for $D(1,m)$. Now, let $n>1$ and $m>1$ and let $q=(q_1,\dots,q_{L-1},q_L)$ be an optimal warping path for $X(1\!:\!n)$ and $Y(1\!:\!m)$. Then the boundary condition implies $q_L = (n, m)$. Setting $(a, b) := q_{L-1}$, the step size condition implies $(a, b) \in \{(n-1, m-1), (n-1, m), (n, m-1)\}$. Furthermore, it follows that $(q_1, \ldots, q_{L-1})$ must be an optimal warping path for $X(1:a)$ and $Y(1:b)$ (otherwise, q would not be optimal for $X(1:n)$ and $Y(1:m)$). Since $D(n, m) = c_{(q_1,\ldots,q_{L-1})}(X(1:a), Y(1:b)) + c(x_n, y_m)$, the optimality of q implies the assertion of (4.5). □

Theorem 4.3 facilitates a recursive computation of the matrix D. The initialization can be simplified by extending the matrix D with an additional row and column and formally setting $D(n, 0) := \infty$ for $n \in [1:N]$, $D(0, m) := \infty$ for $m \in [1:M]$, and $D(0, 0) := 0$. Then the recursion of (4.5) holds for $n \in [1:N]$ and $m \in [1:M]$. Furthermore, note that D can be computed in a column-wise fashion, where the computation of the m-th column only requires the values of the $(m-1)$-th column. This implies that if one is only interested in the value $\mathrm{DTW}(X, Y) = D(N, M)$, the storage requirement is $O(N)$. Similarly, one can proceed in a row-wise fashion, leading to $O(M)$. However, note that the running time is $O(NM)$ in either case. Furthermore, to compute an optimal warping path p^*, the entire $(N \times M)$-matrix D is needed. It is left as an exercise to show that the following algorithm OptimalWarpingPath fulfills this task.

Algorithm: OptimalWarpingPath

Input: Accumulated cost matrix D.
Output: Optimal warping path p^*.

Procedure: The optimal path $p^* = (p_1, \ldots, p_L)$ is computed in reverse order of the indices starting with $p_L = (N, M)$. Suppose $p_\ell = (n, m)$ has been computed. In case $(n, m) = (1, 1)$, one must have $\ell = 1$ and we are finished. Otherwise,

$$p_{\ell-1} := \begin{cases} (1, m-1), & \text{if } n = 1 \\ (n-1, 1), & \text{if } m = 1 \\ \operatorname{argmin}\{D(n-1, m-1), & \\ \qquad D(n-1, m), D(n, m-1)\}, & \text{otherwise,} \end{cases} \tag{4.6}$$

where we take the lexicographically smallest pair in case "argmin" is not unique.

Figure 4.4 shows the optimal warping path p^* (white line) for the sequences of Fig. 4.2. Note that p^* covers only cells of C that exhibit low costs (cf. Fig. 4.4a). The resulting accumulated cost matrix D is shown in Fig. 4.4b.

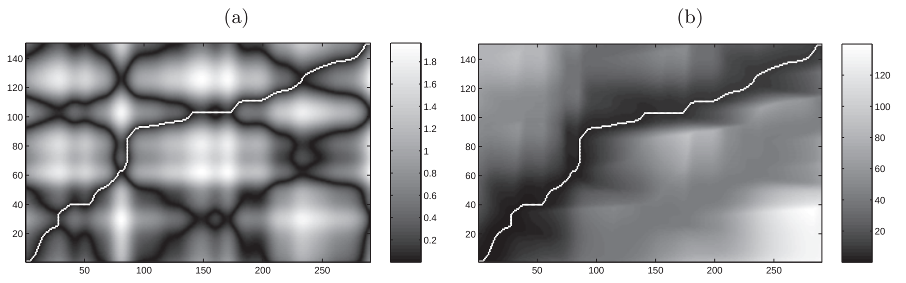

Fig. 4.4. **(a)** Cost matrix C as in Fig. 4.2 and **(b)** accumulated cost matrix D with optimal warping path p^* (*white line*)

4.2 Variations of DTW

Various modifications have been proposed to speed up DTW computations as well as to better control the possible routes of the warping paths. In this section, we discuss some of these variations and refer to [170] for further details.

4.2.1 Step Size Condition

Recall that the step size condition (iii) of Definition 4.1 represents a kind of local continuity condition, which ensures that each element of the sequence $X = (x_1, x_2, \ldots, x_N)$ is assigned to an element of $Y = (y_1, y_2, \ldots, y_M)$ and vice versa. However, one drawback of this condition is that a single element of one sequence may be assigned to many consecutive elements of the other sequence, leading to vertical and horizontal segments in the warping path, see Fig. 4.6a. Intuitively, the warping path can get stuck at some position with respect to one sequence, corresponding to a local deceleration by a large factor (or, conversely, to a local acceleration by a large factor regarding the second sequence).

To avoid such degenerations, one can modify the step size condition to constrain the slope of the admissible warping paths. As a first example, we replace the step size condition (iii) of Definition 4.1 by the condition $p_{\ell+1} - p_\ell \in \{(2,1), (1,2), (1,1)\}$ for $\ell \in [1 : L]$, see Fig. 4.5b. This leads to warping paths having a local slope within the bounds $\frac{1}{2}$ and 2. The resulting accumulated cost matrix D can then be computed by the recursion

$$D(n,m) = \min\{D(n-1,m-1), D(n-2,m-1), D(n-1,m-2)\} + c(x_n, y_m) \tag{4.7}$$

for $n \in [2 : N]$ and $m \in [2 : N]$. As initial values, we set $D(0,0) := 0$, $D(1,1) := c(x_1, y_1)$, $D(n,0) := \infty$ for $n \in [1 : N]$, $D(n,1) := \infty$ for $n \in [2 : N]$, $D(0,m) := \infty$ for $m \in [1 : M]$, and $D(1,m) := \infty$ for $m \in [2 : M]$. Note that, with respect to the modified step size condition, there is a warping

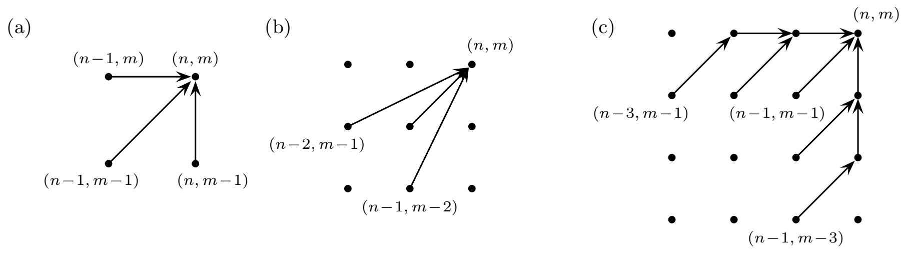

Fig. 4.5. Illustration of three different step size conditions, which express different local constraints on the admissible warping paths. (**a**) corresponds to the step size condition (iii) of Definition 4.1

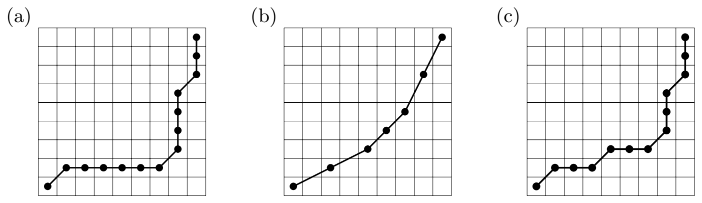

Fig. 4.6. Three warping paths with respect to the different step size conditions indicated by Fig. 4.5. **(a)** Step size condition of Fig. 4.5a may result in degenerations of the warping path. **(b)** Step size condition of Fig. 4.5b may result in the omission of elements in the alignment of X and Y. **(c)** Warping path with respect to the step size condition of Fig. 4.5c

path between two sequences X and Y if and only if the lengths N and M differ at most by a factor of two. Furthermore, note that not all elements of X need to be assigned to some element of Y and vice versa. This is illustrated by Fig. 4.6b: here, x_1 is assigned to y_1, x_3 is assigned to y_2, but x_2 is not assigned to any element of Y (i.e., x_2 is omitted and does not cause any cost at all).

Figure 4.5c gives a second example for a step size condition, which avoids such omission while imposing constraints on the slope of the warping path. The recursion of the resulting accumulated cost matrix D is given by

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + c(x_n,y_m) \\ D(n-2,m-1) + c(x_{n-1},y_m) + c(x_n,y_m) \\ D(n-1,m-2) + c(x_n,y_{m-1}) + c(x_n,y_m) \\ D(n-3,m-1) + c(x_{n-2},y_m) + c(x_{n-1},y_m) + c(x_n,y_m) \\ D(n-1,m-3) + c(x_n,y_{m-2}) + c(x_n,y_{m-1}) + c(x_n,y_m) \end{cases} \tag{4.8}$$

for $(n,m) \in [1:N] \times [1:M] \setminus \{(1,1)\}$. Here, the initial values are set to $D(1,1) := c(x_1,y_1)$, $D(n,-2) := D(n,-1) := D(n,0) := \infty$ for $n \in [-2:N]$, and $D(-2,m) := D(-1,m) := D(0,m) := \infty$ for $m \in [-2:M]$. The slopes of the resulting warping paths lie between the values $\frac{1}{3}$ and 3. Note that this step size conditions enforce that all elements of X are aligned to some element of Y and vice versa. In other words, in the recursion (4.8) all elements of X and Y generate some cost in the accumulated cost matrix D – opposed to the recursion (4.7). Figure 4.6 illustrates the differences of the resulting optimal warping paths computed with respect to different step size conditions.

4.2.2 Local Weights

To favor the vertical, horizontal, or diagonal direction in the alignment, one can introduce an additional weight vector $(w_d, w_h, w_v) \in \mathbb{R}^3$, yielding the recursion

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases} \tag{4.9}$$

for $n \in [2:N]$ and $m \in [2:M]$. Furthermore, $D(n,1) := \sum_{k=1}^{n} w_h \cdot c(x_k, y_1)$ for $n > 1$, $D(1,m) = \sum_{k=1}^{m} w_v \cdot c(x_1, y_k)$ for $m > 1$, and $D(1,1) := c(x_1, y_1)$. The equally weighted case $(w_d, w_h, w_v) = (1,1,1)$ reduces to classical DTW, see (4.5). Note that for $(w_d, w_h, w_v) = (1,1,1)$ one has a preference of the diagonal alignment direction, since one diagonal step (cost of one cell) corresponds to the combination of one horizontal and one vertical step (cost of two cells). To counterbalance this preference, one often chooses $(w_d, w_h, w_v) = (2,1,1)$. Similarly, one can introduce weights for other step size conditions.

4.2.3 Global Constraints

One common DTW variant is to impose global constraint conditions on the admissible warping paths. Such constraints do not only speed up DTW computations but also prevent pathological alignments by globally controlling the route of a warping path. More precisely, let $R \subseteq [1:N] \times [1:M]$ be a subset referred to as global *constraint region*. Then a *warping path relative to R* is a warping path that entirely runs within the region R. The *optimal warping path relative to R*, denoted by p^*_R, is the cost-minimizing warping path among all warping paths relative to R.

Two well-know global constraint regions are the *Sakoe-Chiba band* and the *Itakura parallelogram*, as indicated by Fig. 4.7. Alignments of cells can be selected only from the respective shaded region. The *Sakoe-Chiba band* runs along the main diagonal and has a fixed (horizontal and vertical) width $T \in \mathbb{N}$. This constraint implies that an element x_n can be aligned only to one of the elements y_m with $m \in \left[\frac{M-T}{N-T} \cdot (n-T), \frac{M-T}{N-T} \cdot n + T\right] \cap [1:M]$, see Fig. 4.7a. The *Itakura parallelogram* describes a region that constrains the slope of a warping path. More precisely, for a fixed $S \in \mathbb{R}_{>1}$, the Itakura parallelogram consists of all cells that are traversed by some warping path having a slope

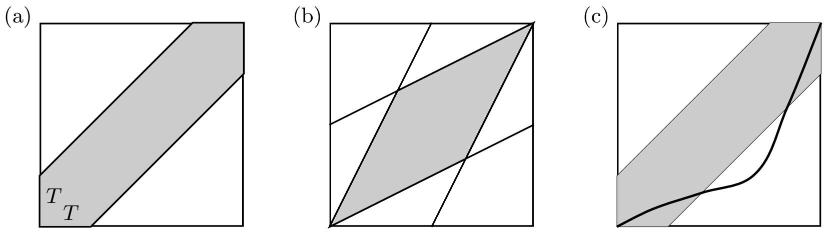

Fig. 4.7. (a) Sakoe-Chiba band of (horizontal and vertical) width T. **(b)** Itakura parallelogram with $S = 2$. **(c)** Optimal warping path p^* (*black line*) which does not run within the constraint region R

between the values $1/S$ and S, see Fig. 4.7b. Note that the local step size condition introduced in Sect. 4.2.1 may also imply some global constraint. For example, the step size condition $p_{\ell+1} - p_\ell \in \{(2,1),(1,2),(1,1)\}$ for $\ell \in [1:L]$, as indicated by Fig. 4.5b, implies a global constraint region in form of an Itakura parallelogram with $S = 2$.

For a general constraint region R, the path p^*_R can be computed similar to the unconstrained case by formally setting $c(x_n, y_m) := \infty$ for all $(n,m) \in [1:N] \times [1:M] \setminus R$. Therefore, in the computation of p^*_R only the cells that lie in R need to be evaluated. This may significantly speed up the DTW computation. For example, in case of a Sakoe-Chiba band of a fixed width T, only $O(T \cdot \max(N,M))$ computations need to be performed instead of $O(NM)$ as required in classical DTW. Here, note that one typically has $T \ll M$ and $T \ll N$.

However, the usage of global constraint regions is also problematic, since the optimal warping path may traverse cells outside the specified constraint region. In other words, the resulting optimal (constrained) warping path p^*_R generally does not coincide with the optimal (unconstrained) warping path p^*, see Fig. 4.7c. This fact may lead to undesirable or even completely useless alignment results.

4.2.4 Approximations

An effective strategy to speed up DTW computations is based on the idea to perform the alignment on coarsened versions of the sequences X and Y, thus reducing the lengths N and M of the two sequences. Such a strategy is also known as dimensionality reduction or data abstraction. For example, to reduce the data rate, one could process the sequences by some suitable low-pass filter followed by downsampling. Another strategy is to approximate the sequences by some piecewise linear (or any other kind of) function and then to perform the warping at the approximation level. For further strategies and an overview, we refer to [102].

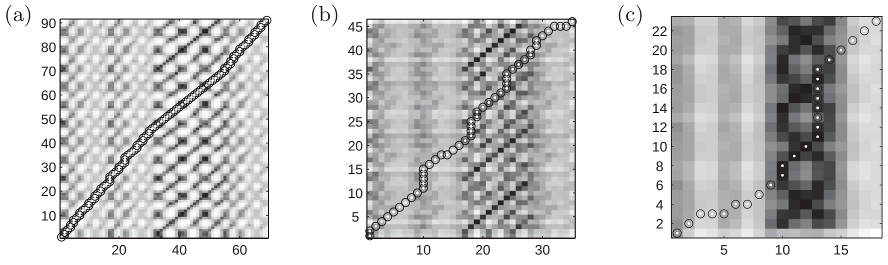

Fig. 4.8. **(a)** Cost matrix and optimal warping path (*dotted path*) between two feature sequences X of length $N = 91$ and Y of length $M = 68$. **(b)** Cost matrix obtained after low-pass filtering and downsampling (by a factor of two) the feature sequences. The resulting optimal warping path at the lower resolution level does not accord well to the optimal warping path of (a). **(c)** Further decreasing the feature resolution destroys the structure of the cost matrix and leads to a completely useless alignment

One important limitation of this approach, however, is that the user must carefully specify the approximation levels used in the alignment. If the user chooses too fine of an approximation, the gains in speed are negligible. Conversely, if the user chooses too coarse of an approximation, e. g., by decreasing the sampling rate of the feature sequences X and Y, the resulting optimal warping path may become inaccurate or even completely useless, see [102]. This fact is also illustrated by Fig. 4.8.

4.3 Multiscale DTW

To obtain an efficient as well as robust algorithm to compute DTW-based alignments, one can combine the strategies described in Sects. 4.2.3 and 4.2.4 in some iterative fashion to generate data-dependent constraint regions. The general strategy is to recursively project an optimal warping path computed at a coarse resolution level to the next higher level and then to refine the projected path. In this section, we summarize the main ideas of this approach, which will be referred to as multiscale DTW (MsDTW). For details we refer to [184]. A similar approach has been applied, e. g., to melody alignment [1] and to audio alignment [142].

Let $X_1 := X$ and $Y_1 := Y$ be the sequences to be synchronized, having lengths $N_1 := N$ and $M_1 := M$, respectively. It is the objective to compute an optimal warping path p^* between X_1 and Y_1. The highest resolution level will also be referred to as Level 1. By reducing the feature sampling rate by a factor of $f_2 \in \mathbb{N}$, one obtains sequences X_2 of length $N_2 := N_1/f_2$ and Y_2 of length $M_2 := M_1/f_2$. (Here, we assume that f_2 divides N_1 and M_1, which can be achieved by suitably padding X_1 and Y_1.) Next, one computes an optimal warping path p_2^* between X_2 and Y_2 on the resulting resolution

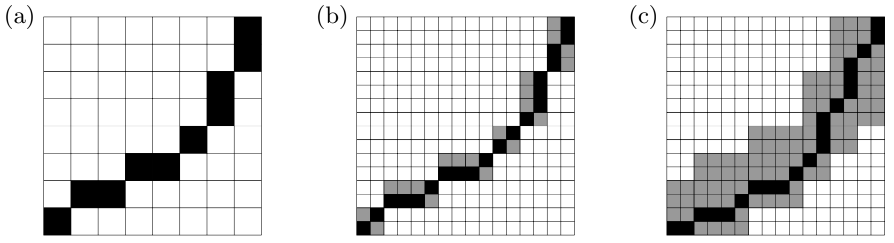

Fig. 4.9. (a) Optimal warping path p_2^* on Level 2. **(b)** Optimal warping path p_R^* with respect to the constraint region R obtained by projecting path p_2^* to Level 1. (Here, p_R^* does not coincide with the (unconstrained) optimal warping path p^*.) **(c)** Optimal warping path $p_{R^\delta}^*$ using an increased constraint region $R^\delta \supset R$ with $\delta = 2$. Here, $p_{R^\delta}^* = p^*$

level (Level 2). This path is projected onto Level 1 and there defines a constraint region R. Note that R consists of $L_2 \times f_2^2$ cells, where L_2 denotes the length of p_2^*. Finally, an optimal warping path p_R^* relative to R is computed. We say that this procedure is *successful* if $p^* = p_R^*$. The overall number of cells to be computed in this procedure is $N_2M_2 + L_2f_2^2$, which is generally much smaller than the total number N_1M_1 of cells on Level 1. In an obvious fashion, this procedure can be recursively applied by introducing further levels of decreasing resolution. For a complexity analysis, we refer to [184].

The constrained path p_R^* may not coincide with the optimal path p^*. To alleviate this problem, one can increase the constraint region R – at the expense of efficiency – by adding δ cells to the left, right, top, and bottom of every cell in R for some parameter $\delta \in \mathbb{N}$. The resulting region R^δ will be referred to as δ-neighborhood of R, see Fig. 4.9.

4.4 Subsequence DTW

In many applications, the sequences to be compared exhibit a significant difference in length. Instead of aligning these sequences globally, one often has the objective to find a subsequence within the longer sequence that optimally fits the shorter sequence, see Fig. 4.10. For example, assuming that the longer sequence represents a given database and the shorter sequence a query, a typical goal is to identify the fragment within the database that is most similar to the query. The problem of finding optimal subsequences can be solved by a variant of dynamic time warping, as will be described in this section.

Let $X := (x_1, x_2, \ldots, x_N)$ and $Y := (y_1, y_2, \ldots, y_M)$ be feature sequences, where we assume that the length M is much larger than the length N. In the following, we fix a local cost function c. It is the goal to find a subsequence $Y(a^* : b^*) := (y_{a^*}, y_{a^*+1}, \ldots, y_{b^*})$ with $1 \leq a^* \leq b^* \leq M$ that

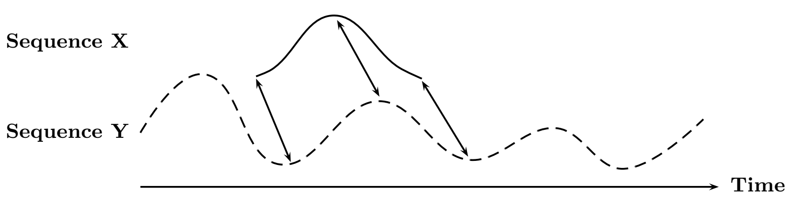

Fig. 4.10. Optimal time alignment of the sequence X with a subsequence of Y. Aligned points are indicated by the *arrows*

minimizes the DTW distance to X over all possible subsequences of Y. In other words,

$$(a^*, b^*) := \underset{(a,b)\,:\,1\leq a\leq b\leq M}{\operatorname{argmin}} \Big(\mathrm{DTW}\big(X\,,\,Y(a\!:\!b)\big)\Big). \tag{4.10}$$

The indices a^* and b^* as well as an optimal alignment between X and the subsequence $Y(a^*:b^*)$ can be computed by a small modification in the initialization of the DTW algorithm as described in Theorem 4.3. The basic idea is not to penalize the omissions in the alignment between X and Y that appear at the beginning and at the end of Y. More precisely, we modify the definition of the accumulated cost matrix D by setting $D(n,1) := \sum_{k=1}^{n} c(x_k, y_1)$ for $n \in [1:N]$ and $D(1,m) := c(x_1, y_m)$ (opposed to $D(1,m) := \sum_{k=1}^{m} c(x_1, y_k)$) for $m \in [1:M]$). The remaining values of D are defined recursively as in (4.5) for $n \in [2:N]$ and $m \in [2:N]$. Similar to Sect. 4.1, one can also define an extended accumulated cost matrix by setting $D(n,0) := \infty$ for $n \in [0:N]$ and $D(0,m) := 0$ (opposed to $D(0,m) := \infty$) for $m \in [0:M]$. The index b^* can be determined from D by

$$b^* := \underset{b\in[1:M]}{\operatorname{argmin}}\, D(N,b). \tag{4.11}$$

To determine a^* as well as the optimal warping path between X and the subsequence $Y(a^*:b^*)$, we apply Algorithm OptimalWarpingPath from Sect. 4.1, this time, however, starting with $p_L = (N, b^*)$. Let $p^* = (p_1, \ldots, p_L)$ denote the resulting path. Then $a^* \in [1:M]$ is the maximal index such that $p_\ell = (a^*, 1)$ for some $\ell \in [1:L]$. In other words, all elements of Y to the left of y_{a^*} and to the right of y_{b^*} are left unconsidered in the alignment and do not cause any additional costs. It is left as an exercise to show that the subsequence $Y(a^*:b^*)$ indeed has minimal total cost in the alignment with X among all possible subsequences of Y. The optimal warping path between X and $Y(a^*:b^*)$ is given by $(p_\ell, \ldots, p_L)$. Obviously, the computational complexity of the subsequence DTW algorithm is $O(NM)$. Furthermore, note that the optimal subsequence $Y(a^*:b^*)$ is in general not uniquely defined – there may be several choices for b^* in (4.11), and in the construction of a^* we have used a maximality criterion.

We finally describe how the accumulated cost matrix D can be used to derive an entire list of subsequences of Y that are close to X with respect to the DTW distance. To this end, we define a distance function

$$\Delta : [1:M] \to \mathbb{R}, \qquad \Delta(b) := D(N,b), \tag{4.12}$$

which assigns to each index $b \in [1:M]$ the minimal DTW distance $\Delta(b)$ that can be achieved between X and a subsequence $Y(a\!:\!b)$ of Y ending in y_b. For each $b \in [1:M]$, the DTW-minimizing $a \in [1:M]$ can be computed analogously to a^* as described above using Algorithm OptimalWarpingPath and starting with $p_L = (N,b)$. Note that if $\Delta(b)$ is small for some $b \in [1:M]$ and if $a \in [1:M]$ denotes the corresponding DTW-minimizing index, then the subsequence $Y(a\!:\!b)$ is close to X. This observation suggests the following algorithm to compute all (up to some specified overlap) subsequences of Y similar to X:

```
Algorithm: ComputeSimilarSubsequences

Input:     $X = (x_1,\dots,x_N)$   query sequence
           $Y = (y_1,\dots,y_M)$   database sequence
           $\tau \in \mathbb{R}$              cost threshold
Output:    Ranked list of all (essential distinct) subsequences of $Y$
           that have a DTW distance to $X$ below the threshold $\tau$.

(0)  Initialize the ranked list to be the empty list.
(1)  Compute the accumulated cost matrix $D$ w.r.t. $X$ and $Y$,
(2)  Determine the distance function $\Delta$ as in (4.12).
(3)  Determine the minimum $b^* \in [1:M]$ of $\Delta$.
(4)  If $\Delta(b^*) > \tau$ then terminate the procedure.
(5)  Compute the corresponding DTW-minimizing index $a^* \in [1:M]$.
(6)  Extend the ranked list by the subsequence $Y(a^*\!:\!b^*)$.
(7)  Set $\Delta(b) := \infty$ for all $b$ within a suitable neighborhood of $b^*$.
(8)  Continue with Step (3).
```

Note that Step (7) is introduced to exclude an entire neighborhood of b^* from further consideration, thus avoiding that the ranked output list contains many subsequences that only differ by a slight shift. (For example, if $Y(a:b)$ is in the list, one can prevent that $Y(a:b+1)$ is in the list as well.) Depending on the application, one may choose a fixed size of the neighborhood around b^* or one may adaptively adjust the size according to the local property of Δ around b^*. For example, one may require that b^* is a local minimum of Δ and then determines the neighborhood confined by the closest local maxima to the left and to the right of b^*.

We illustrate the procedure of Algorithm COMPUTESIMILARSUBSEQUENCES by the example shown in Fig. 4.11a. The input consists of the query sequences X and the database sequence Y as well as the cost threshold $\tau = 5$. We iteratively look for all indices $b \in [1 : M]$ that are local minima of the distance function Δ with $\Delta(b) \leq \tau$, see Fig. 4.11b. In the first iteration, we obtain the local minimum $b^* = 291$ with $\Delta(b^*) = 2.13$, which also constitutes the global minimum of Δ. Based on Algorithm OPTIMALWARPINGPATH, one obtains $a^* = 163$ as well as the optimal warping path that aligns X with the subsequence $Y(a^* : b^*)$, see Figs. 4.11c, d. We next determine the closest local maximum $b^*_\ell = 265$ to the left and closest local maximum $b^*_r = 313$ to the right of b^* and exclude the neighborhood $[b^*_\ell : b^*_r]$ for further consideration by setting $\Delta(b) := \infty$ for $b \in [b^*_\ell : b^*_r]$. We then proceed with the modified Δ in the same fashion, obtaining the local minima $b^*_2 = 454$ with $\Delta(b^*_2) = 2.66$ and $a^*_2 = 367$. In a third iteration, one obtains $b^*_3 = 137$ with $\Delta(b^*_3) = 3.50$ and $a^*_2 = 44$. The three resulting "matches," i. e., subsequences of Y close to X, are shown in Fig. 4.11d.

4.5 Further Notes

Originating from speech processing, dynamic time warping (DTW) has become a well-established method to account for temporal variations in the comparison of related time series. A classical and comprehensive account on DTW and related pattern recognition techniques is given by Rabiner and Juang [170] in the context of speech recognition. In their book, one also finds a detailed introduction to Hidden Markov Models (HMMs) – a statistical method that extends the concept of DTW-based pattern recognition.

In addition to speech recognition, dynamic time warping has found numerous applications in a wide range of fields including data mining, information retrieval, bioinformatics, chemical engineering, signal processing, robotics, or computer graphics, see, e. g., [100] and the references therein. Basically any data that can be transformed into a (linear) sequence of features can be analyzed with DTW, which includes data types such as text, video, audio, or general time series. In the field of music information retrieval, DTW plays in important role for synchronizing music data streams [57,94,141,142,196,202]. DTW has also been used in the field of computer animation to analyze and align motion data [22, 75, 93, 106, 143, 217, 220]. In Chaps. 5 and 10, some of these reference will be discussed in more detail.

Extensive research has been performed on how to accelerate DTW computations, in particular for one-dimensional (or low-dimensional) real-valued sequences, often referred to as time series, see [102, 184] and the references therein. The problem of indexing large time series databases has also attracted great interest in the database community, see Last et al. [117] for an overview. Keogh [100] describes an indexing method based on lower bounding techniques that makes retrieval of time-warped time series feasible even for large

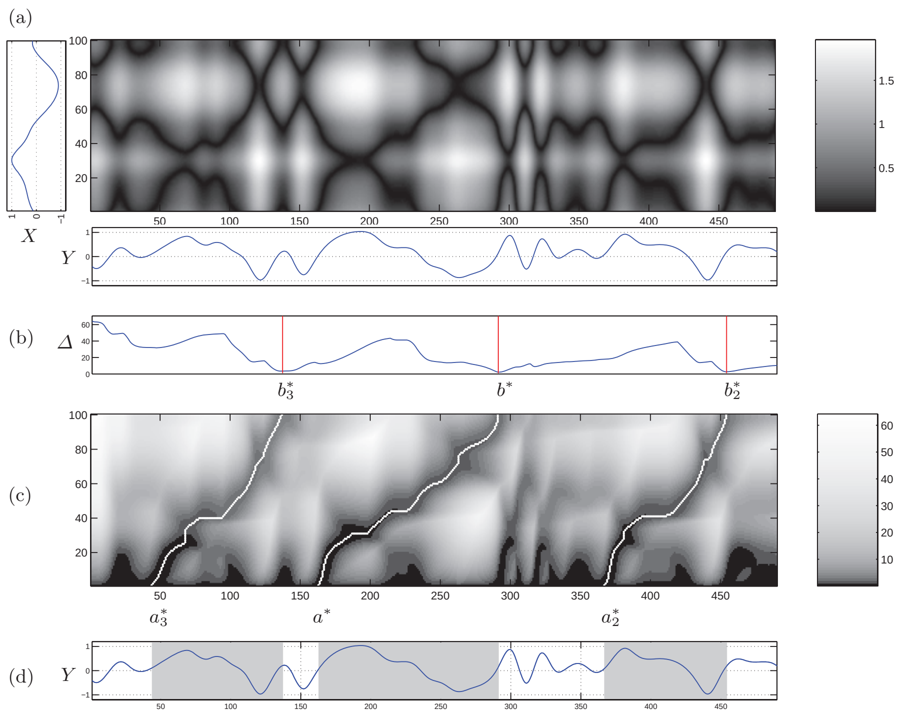

Fig. 4.11. (**a**) Cost matrix between the query sequences X (vertical axis) and the database sequence Y (horizontal axis) using the absolute value as local cost measure c. (**b**) Distance function Δ corresponding to the top row of the matrix D. The *red vertical lines* indicate the three local minima b^*, b_2^*, and b_3^* having a Δ-value below the cost threshold $\tau = 5$. (**c**) Accumulated cost matrix D for subsequence DTW and the three optimal warping paths (*white lines*) corresponding to the minima of (b). (**d**) Resulting subsequences $Y(a_3^*:b_3^*)$, $Y(a^*:b^*)$, and $Y(a_2^*:b_2^*)$ of Y (indicated by *grey regions*)

datasets. An early account on efficient indexing for subsequence matching of one-dimensional time series is given by Faloutsos et al. [62]. Most of these approaches follow the general strategy to first extract a coarse approximation from each time series of the database. Such approximations may be based on taking the first few coefficients of a Fourier [2] or wavelet transform [39], or the average values of adjacent analysis windows [101]. Next, based on a suitable distance measure on the approximations, one computes lower bounds for the distances between the corresponding time series. To view of efficient retrieval, the approximations can then be stored by means of multidimensional indexing methods such as an R-tree [90].

Closely related to DTW is the *edit distance*, which is sometimes also referred to as *Levenshtein distance* [120]. The edit distance is used to compute a distance between strings, i. e., one-dimensional sequences consisting of discrete symbols (rather than sequences consisting of continuous features). It is defined as the minimum number of operations needed to transform one string into the other, where an operation is an insertion, a deletion, or a substitution of a single symbol. The edit distance is used in fields such as text retrieval (spell checkers, plagiarism detection) or molecular biology (to compute a distance between DNA sequences, i. e., strings over $\{A,C,G,T\}$). For a detailed account on the edit distance with its applications to bioinformatics, we refer to [18].

Vlachos et al. [209] introduced similarity measures for multidimensional time series (having values in $\mathbb{R}^d$) based on the concept of longest common subsequences (LCS) – a variant of the edit distance that is more robust to noise and outliers. For low dimensions ($d \leq 3$), they also describe some efficient approximation algorithms that compute these similarity measures.

The computation of DTW, edit, as well as LCS distances can be done efficiently by means of *dynamic programming*, which is a general method for reducing the running time of algorithms exhibiting the properties of overlapping subproblems and optimal substructure. For a general introduction to dynamic programming, we refer the reader to the standard text book by Cormen et al. [47].