Appendix D

Pattern detection algorithm

In this appendix, we describe the algorithm used to match a complex pattern with a time series. A complex pattern is composed of a sequence of subsignals of fixed shape. The matching consists of determining the optimal positions of the subsignals in such way that they do not overlap and that the Euclidean distance between the complex pattern and the original time series is minimized. This is essentially a combinatorial problem, but the algorithm described below is efficient.

D.1 Problem statement

In this section, we view a (complex) pattern as a compact representation of a set of time series and the pattern matching problem as the computation of the distance between a particular time series and this set. Although these ideas could be applied to any kind of signals, we particularize the presentation to the case of discrete time, finite duration, and real-valued signals (i.e. elements of $\mathbb{R}^N$).

We define a complex pattern $\mathcal{P}$ as an ordered sequence of K elementary patterns

$$\mathcal{P} \triangleq (p_1(.), p_2(.), \ldots, p_K(.)),$$

where each elementary pattern $p_k(.)$ $(k = 1, \ldots, K)$ is defined as a time series of P_k values, $(p_{k,0}, \ldots, p_{k,P_k-1})$. Such a complex pattern is essentially a compact representation of a set of candidate signals. More precisely, given a total length N such that

$$N \geq \sum_{k=1}^{K} P_k,$$

we identify the complex pattern with the following set of signals[1]

$$S_{\mathcal{P}}^N \triangleq \left\{ s(\cdot) \in \mathbb{R}^N | s(\cdot) = \left([*]^{l_0} p_1(\cdot) [*]^{l_1} p_2(\cdot) \cdots [*]^{l_{K-1}} p_K(\cdot) [*]^{l_K} \right) \right\}, \tag{D.1}$$

where $[*]^{l_k}$ denotes a (dummy) sequence of length l_k of arbitrary real numbers $(l_k \geq 0)$. Since the total length of any element of $S_{\mathcal{P}}^N$ is equal to N we have the following additional constraint among the lengths of the dummy signals $[*]^{l_k}$:

$$\sum_{k=0}^{K} l_k = N - P_{\mathcal{P}}, \tag{D.2}$$

[1]In what follows, we will identify elements of $\mathbb{R}^N$, sequences of real numbers of length N, and temporal signals defined on the bounded integer time axis $0, \ldots, N-1$; moreover, we will (non-orthodoxly) mix sequence, vector, and signal notation where we find it appropriate.

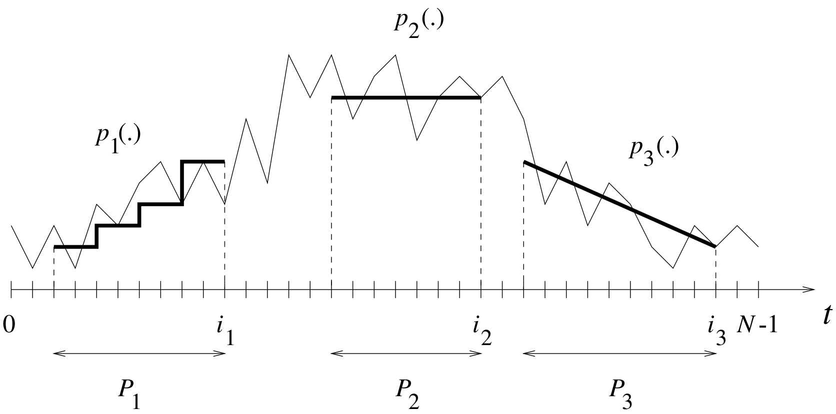

Figure D.1: One possible configuration of three patterns in a time series

where $P_{\mathcal{P}} = \sum_{k=1}^{K} P_k$ denotes the total length of the elementary patterns defining $\mathcal{P}$.

Given two time series $a(\cdot) = (a_0, \ldots, a_{N-1})$ and $s(\cdot) = (s_0, \ldots, s_{N-1})$ defined on N, their Euclidean distance is defined by

$$d(a(\cdot), s(\cdot)) \triangleq \sum_{i=0}^{N-1} (a(i) - s(i))^2. \tag{D.3}$$

Given a pattern $\mathcal{P}$ and a time series $a(\cdot) = (a_0, \ldots, a_{N-1})$, the Euclidean distance between these two objects is thus defined by

$$D(a(\cdot), \mathcal{P}) \triangleq \inf_{s(\cdot) \in S_{\mathcal{P}}^N} \{d(a(\cdot), s(\cdot))\}. \tag{D.4}$$

The objective of the pattern matching algorithm described in the next section is essentially to compute this distance efficiently.

Notice that, once the lengths l_k of the dummy portions in (D.1) are fixed, the minimization of the Euclidean distance is obvious. Indeed, for fixed values of l_k the element of $S_{\mathcal{P}}^N$ which minimizes (D.4) is simply obtained by setting the values in each dummy portion of $s(\cdot)$ to the values of $a(\cdot)$ in the same position. Consequently, the lower bound of (D.4) can be determined by searching the optimal lengths l_k of the $K+1$ dummy signals under the constraints

$$\sum_{k=0}^{K} l_k = N - P_{\mathcal{P}} \quad \text{and} \quad l_k \geq 0. \tag{D.5}$$

This is essentially a combinatorial problem. Notice that the number of combinations of $K+1$ integer values satisfying the constraints (D.5) is lower bounded by $\mathcal{C}_{N-P_{\mathcal{P}}}^{K}$ and upper bounded by $(N - P_{\mathcal{P}})^K$. In many practical situations (long-time series, large number of short patterns), this number is far too high to allow for a complete enumeration. The next section describes an iterative algorithm to solve efficiently this problem.

D.2 Optimal matching algorithm

In order to formulate our algorithm, let us consider a complex pattern $\mathcal{P}$ as defined in the preceding section, and define, for a particular signal of $s(\cdot) \in S_{\mathcal{P}}^N$, by i_k the end-position of the k-th elementary pattern in $s(\cdot)$. This is illustrated in Figure D.1 in the case of a complex pattern composed of three elementary patterns. Clearly, the above considered set of sequences $(l_0, l_1, \ldots, l_K)$ characterizing the signals $S_{\mathcal{P}}^N$ is in one-to-one relationship with the sequences of end-positions $(i_1, \ldots, i_K)$ satisfying the following constraints:

$$\begin{aligned} i_1 &\geq P_1 - 1, && \text{(D.6)} \\ i_k &\geq i_{k-1} + P_k, \forall k = 2, \ldots, K, && \text{(D.7)} \\ i_K &< N. && \text{(D.8)} \end{aligned}$$

Thus, in terms of the i_k values, the determination $D(a(\cdot), \mathcal{P})$ is equivalent to computing the following quantity:

$$\min_{i_1,\ldots,i_K | C_{\mathcal{P}}(i_1,\ldots,i_K;N)} \sum_{k=1}^{K} d(p_k(.), a(.), i_k) \tag{D.9}$$

where $c_P(i_1, i_2, \ldots, i_K; N)$ corresponds to conditions (D.7), and the distance $d(p_k(.), a(.), i)$ (for $k = 1, \ldots, K$ and $i = P_k - 1, \ldots, N-1$) is given by:

$$d(p_k(.), a(.), i) = \sum_{j=i-P_k+1}^{i} (p_{k,j-i+P_k+1} - a_j)^2. \tag{D.10}$$

Notice that in the distance computation (D.9) the parts of $s(\cdot)$ and $a(\cdot)$ corresponding to the dummy portions have been dropped since they can match perfectly for each choice of locations and thus lead to a zero contribution in the minimum distance.

D.2.1 Backwards recursion

The algorithmic solution of the above problem is based on a recursion relation holding for the function $g(i, k)$ defined by

$$g(i,k) = \min_{i_1,\ldots,i_k | C_{\mathcal{P}}(i_1,\ldots,i_k;i+1)} \sum_{j=1}^{k} d(p_j(.), a(.), i_j), \tag{D.11}$$

for all values of $k = 1, \ldots, K$ and $i = \sum_{j=1}^{k} P_j - 1, \ldots, N-1$. Notice that $g(i, k)$ is by definition the minimum distance between the time series $a(\cdot)$ truncated to the first $i+1$ time instants and the complex pattern truncated to the first k elementary patterns. Thus, obviously,

$$D(a(\cdot), \mathcal{P}) = g(N-1, K). \tag{D.12}$$

Notice also that the values $g(i, k)$ are by definition infinite for those values of i, k for which there is no admissible way to match the sub-signal and the sub-pattern, namely when $i + 1 < \sum_{j=1}^{k} P_j$.

The function $g(\cdot, \cdot)$ satisfies the following recursive relationship

$$g(i,k) = \min\{d(p_k(.), a(.), i) + g(i - P_k, k-1), g(i-1, k)\}. \tag{D.13}$$

Indeed, to optimally match the first k patterns in the first $i+1$ values of $a(\cdot)$,

- either the k^{th} elementary pattern ends at position i: in this case the positions of the $k-1$ first patterns must be optimal over the $i - P_k + 1$ first instants, and this indeed corresponds to the value $g(i - P_k, k-1)$;

- or the k^{th} elementary pattern ends at a strictly lower position in the time series than i: in this case the minimum distance is provided by $g(i-1, k)$.

Table D.1: Forward sequential pattern matching algorithm

```
Set α=−1
Set β=N − P_𝒫 − 1
For i=α to β
   Set g(i,0)=0
End for
For k=1 to K
   Set α=α + P_k                          ; The smallest candidate value of i_k.
   Set β=β + P_k                          ; The largest candidate value of i_k.
   Set g(α − 1,k) = +∞
   For i=α to β
      Set d=d(p_k(.),a(.),i)
      If d + g(i − P_k,k − 1) < g(i − 1,k) then
         Set p(i,k)=i
         Set g(i,k)=d + g(i − P_k,k − 1)
      Else
         Set p(i,k)=p(i − 1,k)
         Set g(i,k)=g(i − 1,k)
   End for
End for
Return g(N − 1,K)
```

Table D.2: Algorithm which computes the positions corresponding to the minimal distance

```
Set i_K = p(N − 1,K)
For k=K − 1 to 1
   Set i_k = p(i_{k+1} − P_{k+1},k)
```

D.2.2 Forward sequential algorithm

This recursive formulation of $g(i,k)$ is the basis for the forward sequential algorithm of Table D.1, which properly initializes the table representing the values of $g(i,k)$ and, for each value of k, computes only those values of $g(i,k)$ required to determine $g(N-1,K)$. Since it is also interesting to find the values of i_k, $k=1,\ldots,K$, which realize the minimum in (D.9), the algorithm stores this information in another table denoted $p(i,k)$. This latter table is then used a posteriori to retrieve the positions i_k which realize the minimum, according to the algorithm of Table D.2.

D.2.3 Computational complexity

The algorithm computes $K(N+1-P_{\mathcal{P}})$ values of $g(i,k)$ and requires P_k operations at each step k to compute the distances $d(p_k,a(\cdot),i)$ for each one of the $N+1-P_{\mathcal{P}}$ candidate values of i. Hence, the number of operations of this algorithm is in general $O((N-P_{\mathcal{P}})P_{\mathcal{P}})$.

In the particular case where the elementary patterns $p_k(.)$ are constant over the P_k

time points, the distances $d(p_k, a(\cdot), i)$ for growing values of i may be computed in constant time using the following update formula:

$$d(p(.), a(.), i) = d(p(.), a(.), i-1) + (p - a_i)^2 - (p - a_{i-P+1})^2, i = P, \ldots, N-1$$

with

$$d(p(.), a(.), P-1) = \sum_{j=0}^{P-1} (p - a_j)^2.$$

Hence, in this particular case, the computation of the distance for all values of i requires order N operations. Following the same idea, when the elementary pattern is represented by a piecewise constant signal in s segments, these distances may be computed in $O(N.s)$.

Finally, when the complex patterns are extracted from piecewise constant signals represented with S segments (and hence contain at most S elementary patterns, which all together contain at most S segments), the computation of the minimal distance using to the algorithm of Table D.1 requires at most $O(N.S)$ operations, whatever the number of elementary patterns chosen and whatever the lengths P_k of these patterns.