CSE515 Advanced Algorithms
Lecture 3: Dynamic Programming I

Antoine Vigneron
antoine@unist.ac.kr

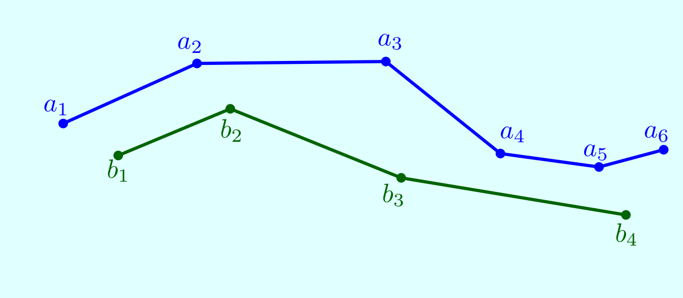
Ulsan National Institute of Science and Technology

March 9, 2021

Introduction

- I posted Exercise Sets 1 and 2, as well as notes on Lecture 2
- *Dynamic programming* (DP) is an important algorithms design technique, that often yields polynomial-time algorithms.
- It is one of the first techniques to try when you face a nontrivial algorithmic problem.
- You must have studied it in the undergraduate algorithmcourse.
- This lecture is a review of DP through an example: *Dynamic Time Warping* (DTW), which is a similarity measure for time series.
- Next lecture will give another example: A histogram construction problem.

Time Series



- We are given two sequences of points $A = (a_1, a_2, \ldots, a_m)$ and $B = (b_1, b_2, \ldots, b_n)$.
- Example: two sequences of points in $\mathbb{R}^2$.
- Point sequences are called *time series* in statistics.

Metric Spaces

- Sequences $A = (a_1, a_2, \ldots, a_m)$ and $B = (b_1, b_2, \ldots, b_n)$ come from a *metric space* (M, d), and we know the distance $d(a_i, b_j)$ for all i, j.

Metric Spaces

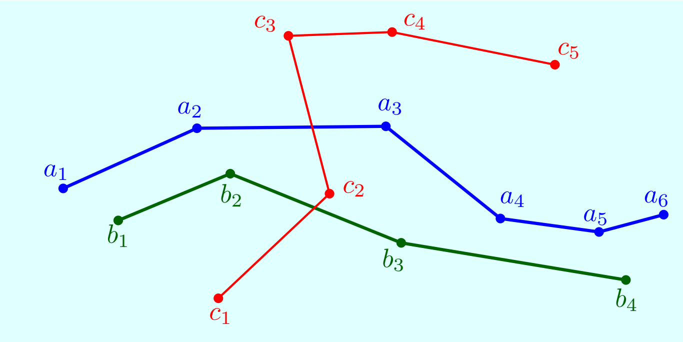
Let M be a set and $d : M \times M \to \mathbb{R}$. We say that (M, d) is a *metric space* if $\forall p, q, r \in M$

- $d(p, q) \geqslant 0$
- $d(p, q) = 0 \;\Leftrightarrow\; p = q$
- $d(p, q) = d(q, p)$ (symmetry)
- $d(p, r) \leqslant d(p, q) + d(q, r)$ (triangle inequality)

- Example of metric space: $\mathbb{R}^d$ with the Euclidean distance.

Similarity Measures for Point Sequences

- Problem: How can we measure similarity between A and B?
- Idea: Find a *similarity measure* $S(\cdot,\cdot) \geqslant 0$ such that $S(A,B)$ is small whenever A and B are similar.



- In the example above, we would like to have $S(A,B) \leqslant S(A,C)$.

The Euclidean Distance

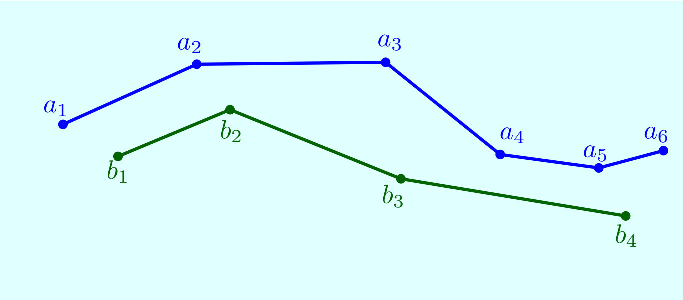
- First approach: The *Euclidean distance*

$$E(A, B) = \sqrt{\sum_{i=1}^{n} d(a_i, b_i)^2}$$

  - This is a metric.
  - Problem: Requires $m = n$.

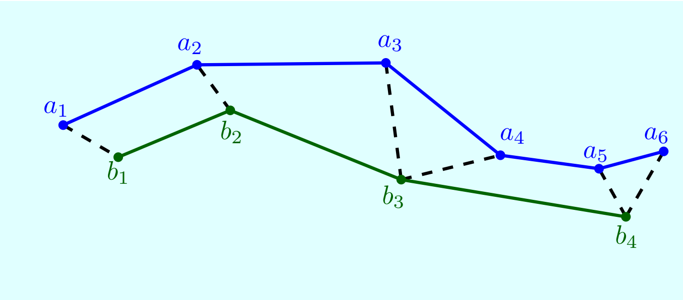
Dynamic Time Warping

- *Dynamic Time Warping (DTW)* allows us to measure similarity between two sequences A and B when their lengths m and n differ.



Dynamic Time Warping

- Idea: We find a *coupling* of A and B, and sum up the distances between the pairs in this coupling.



- $d(a_1, b_1) + d(a_2, b_2) + d(a_3, b_3) + d(a_4, b_3) + d(a_5, b_4) + d(a_6, b_4)$.

Sequence Coupling

Definition (Coupling)

An (m, n)-coupling is a sequence (α_k, β_k), $k = 1, \dots, \ell$ such that:

- $\alpha_1 = \beta_1 = 1$.
- $\alpha_\ell = m$
- $\beta_\ell = n$
- For all $k = 2, \dots, \ell$,

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_{k-1} + 1, \beta_{k-1} + 1), \\ (\alpha_{k-1}, \beta_{k-1} + 1), \text{ or} \\ (\alpha_{k-1} + 1, \beta_{k-1}). \end{cases}$$

Dynamic Time Warping

Definition (Dynamic Time Warping)

For any two point sequences $A = (a_1, \dots, a_m)$ and $B = (b_1, \dots, b_n)$, $\mathrm{DTW}(A, B)$ is the minimum over all (m, n)-couplings (α, β) of $\sum_{k=1}^{\ell} d(a_{\alpha_k}, b_{\beta_k})$.

- It satisfies the recurrence relation

$$\mathrm{DTW}(A_i, B_j) = d(a_i, b_j) + \min \begin{pmatrix} \mathrm{DTW}(A_{i-1}, B_{j-1}), \\ \mathrm{DTW}(A_{i-1}, B_j), \\ \mathrm{DTW}(A_i, B_{j-1}) \end{pmatrix}$$

where $A_i = (a_1, \dots, a_i)$, $B_j = (b_1, \dots, b_j)$ and $i, j > 1$.

First Algorithm

Algorithm 1: Recursive computation of DTW

```
function DTW(A_i, B_j)
    if i = 1 and j = 1 then
        return d(a_1, b_1)                                  ▷ base case
    if j = 1 then
        return d(a_i, b_j) + DTW(A_{i−1}, B_j)
    if i = 1 then
        return d(a_i, b_j) + DTW(A_i, B_{j−1})
    return d(a_i, b_j)+
            min(DTW(A_{i−1}, B_{j−1}), DTW(A_{i−1}, B_j), DTW(A_i, B_{j−1}))
```

Analysis

- Problem: Algorithm 1 runs in *exponential time*.
- More precisely:

Proposition

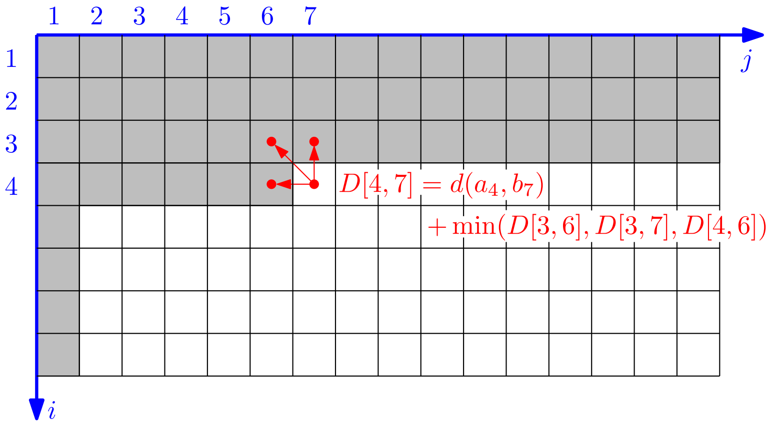
Let $T(m, n)$ be the running time of Algorithm 1 on sequences of lengths m and n. Then $T(n, n) = \Omega(3^n)$.

Proof.

Done in class. $\square$

Better Algorithm

- Idea: Fill a table $D[i,j] = \mathrm{DTW}(A_i, B_j)$ in a bottom-up manner.



Better Algorithm

Algorithm 2

function $\mathrm{DTW}((a_1, \ldots, a_m), (b_1, \ldots, b_n))$
  $D \leftarrow$ new $m \times n$ array
  $D[1,1] \leftarrow d(a_1, b_1)$
  for $i \leftarrow 2, m$ **do**
    $D[i,1] \leftarrow D[i-1,1] + d(a_i, b_1)$
  for $j \leftarrow 2, n$ **do**
    $D[1,j] \leftarrow D[1,j-1] + d(a_1, b_j)$
  for $i \leftarrow 2, m$ **do**
    for $j \leftarrow 2, n$ **do**
      $D[i,j] \leftarrow d(a_i, b_j) + \min(D[i-1,j], D[i-1,j-1], D[i,j-1])$
  return $D[m,n]$

Analysis

Algorithm 2

```
1: function DTW((a_1, ..., a_m), (b_1, ..., b_n))
2:     D ← new m × n array
3:     D[1,1] ← d(a_1, b_1)
4:     for i ← 2, m do
5:         D[i,1] ← D[i−1,1] + d(a_i, b_1)
6:     for j ← 2, n do
7:         D[1,j] ← D[1,j−1] + d(a_1, b_j)
8:     for i ← 2, m do
9:         for j ← 2, n do
10:            D[i,j] ← d(a_i, b_j) + min(D[i−1,j],
                        D[i−1,j−1], D[i,j−1])
11:    return D[m,n]
```

l.	cost	times
2	$O(mn)$	1
3	$\Theta(1)$	1
4	$\Theta(1)$	m
5	$\Theta(1)$	$m-1$
6	$\Theta(1)$	n
7	$\Theta(1)$	$n-1$
8	$\Theta(1)$	m
9	$\Theta(1)$	$(m-1)n$
10	$\Theta(1)$	$(m-1)(n-1)$
11	$\Theta(1)$	1

Proposition

Algorithm 2 runs in $\Theta(mn)$ time.

Dynamic Programming

- This is an example of *dynamic programming*.

Dynamic Programming

Dynamic programming consists in:

- Breaking the problem into subproblems.
- Solving each subproblem just *once*, and *storing* the result.

- Algorithm 1, on the other hand, solved some subproblems an exponential number of times.

Dynamic Time Warping: Concluding Remarks

- Often used in practice.
- Example: similarity of GPS traces.
- DTW is *not* a metric: does not satisfy triangle inequality
- Complexity:
  - We presented an $O(mn)$ time algorithm.
  - No better algorithm is known.
  - Recent work suggests that it may not be possible to do much better.