Time-series data analysis
Why deal with sequential data?

• Because all data is sequential 😊

• All data items arrive in the data store in some order

• Examples
  – transaction data
  – documents and words

• In some (or many) cases the order does not matter

• In many cases the order is of interest
Time-series data

- Financial time series, process monitoring...
Questions

• What is the **structure** of sequential data?

• Can we represent this structure
Sequence segmentation

• Gives an accurate representation of the structure of sequential data

• How?
  – By trying to find homogeneous segments

• Segmentation question:

• Can a sequence \( T = \{t_1, t_2, ..., t_n\} \) be described as a concatenation of subsequences \( S_1, S_2, ..., S_k \) such that each \( S_i \) is in some sense homogeneous?

• The corresponding notion of segmentation in unordered data is clustering
Dynamic-programming algorithm

- Sequence $T$, length $n$, $k$ segments, cost function $E()$, table $M$
- For $i=1$ to $n$
  - Set $M[1,i] = E(T[1...i])$ //Everything in one cluster
- For $j=1$ to $k$
  - Set $M[j,j] = 0$ //each point in its own cluster
- For $j=2$ to $k$
  - For $i=j+1$ to $n$
    - Set $M[j,i] = \min_{i' < i} \{ M[j-1,i] + E(T[i'+1...i]) \}$
- To recover the actual segmentation (not just the optimal cost) store also the minimizing values $i'$
- Takes time $O(n^2k)$, space $O(kn)$
Example
Basic definitions

• Sequence $T = \{ t_1, t_2, \ldots, t_n \}$: an ordered set of $n$ $d$-dimensional real points $t_i \in \mathbb{R}^d$

• A $k$–segmentation $S$: a partition of $T$ into $k$ contiguous segments $\{ s_1, s_2, \ldots, s_k \}$
  
  – Each segment $s \in S$ is represented by a single value $\mu_s \in \mathbb{R}^d$ (the representative of the segment)

• Error $E_p(S)$: The error of replacing individual points with representatives
  \[
  E_p(S) = \left( \sum_{s \in S} \sum_{t \in s} |t - \mu_s|^p \right)^{\frac{1}{p}}
  \]
The k–segmentation problem

Given a sequence $T$ of length $n$ and a value $k$, find a $k$–segmentation $S = \{s_1, s_2, \ldots, s_k\}$ of $T$ such that the

- Common cases for the error function $E_p$: $p = 1$ and $p = 2$.

  - When $p = 1$, the best $\mu_s$ corresponds to the median of the points in segment $s$.

  - When $p = 2$, the best $\mu_s$ corresponds to the mean of the points in segment $s$. 
Optimal solution for the $k$-segmentation problem

- **Bellman’61** The $k$-segmentation problem can be solved optimally using a standard dynamic-programming algorithm

\[
E_p(S_{\text{opt}}(T[1 \ldots n], k)) = \\
\min_{j<n} \{ E_p(S_{\text{opt}}(T[1 \ldots j], k-1)) \\
+ E_p(S_{\text{opt}}(T[j+1, \ldots, n], 1)) \} 
\]

- Running time $O(n^2k)$
  - Too expensive for large datasets!
Heuristics

• Bottom–up greedy (BU): $O(n \log n)$
  – [Keogh and Smyth’97, Keogh and Pazzani’98]

• Top–down greedy (TD): $O(n \log n)$
  – [Douglas and Peucker’73, Shatkay and Zdonik’96, Lavrenko et. al’00]

• Global Iterative Replacement (GiR): $O(nI)$
  – [Himberg et. al ’01]

• Local Iterative Replacement (LiR): $O(nI)$
  – [Himberg et. al ’01]
Approximation algorithm

- **Theorem** The segmentation problem can be approximated within a constant factor of 3 for both $E_1$ and $E_2$ error measures. That is,

$$E_p(S_{DnS}) \leq 3E_p(S_{OPT}) \quad p = 1, 2$$

- The running time of the approximation algorithm is:

$$O(n^{4/3}k^{5/3})$$
Divide ’n Segment (DnS) algorithm

• **Main idea**
  
  – Split the sequence arbitrarily into subsequences
  – Solve the $k$–segmentation problem in each subsequence
  – Combine the results

• **Advantages**
  
  – Extremely simple
  – High quality results
  – Can be applied to other segmentation problems [Gionis’03, Haiminen’04, Bingham’06]
## DnS algorithm – Details

**Input:** Sequence $T$, integer $k$  
**Output:** a $k$-segmentation of $T$

1. Partition sequence $T$ arbitrarily into $m$ disjoint intervals $T_1, T_2, \ldots, T_m$
2. For each interval $T_i$ solve optimally the $k$-segmentation problem using DP algorithm
3. Let $T'$ be the concatenation of $mk$ representatives produced in Step 2. Each representative is weighted with the length of the segment it represents
4. Solve optimally the $k$-segmentation problem for $T'$ using the DP algorithm and output this segmentation as the final segmentation
The DnS algorithm

Input sequence $T$ consisting of $n=20$ points ($k=2$)
The DnS algorithm – Step 1

Partition the sequence into $m=3$ disjoint intervals
The DnS algorithm – Step 2

Solve optimally the $k$-segmentation problem into each partition ($k=2$)
The DnS algorithm – Step 2

Solve optimally the $k$-segmentation problem into each partition ($k=2$)
The DnS algorithm – Step 3

Sequence $T'$ consisting of $mk=6$ representatives
The DnS algorithm – Step 4

Solve $k$-segmentation on $T$ ($k=2$)
Running time

• In the case of equipartition in Step 1, the running time of the algorithm as a function of \( m \) is:

\[
R(m) = m \left( \frac{n}{m} \right)^2 k + (mk)^2 k
\]

• The function \( R(m) \) is minimized for

\[
m_0 = \left( \frac{n}{k} \right)^{\frac{2}{3}}
\]

• Running time \( R(m_0) = 2n^{4/3}k^{5/3} \)
The segmentation error

- **[Theorem]** The segmentation error of the DnS algorithms is at most three times the error of the optimal (DP) algorithm for both $E_1$ and $E_2$ error measures.

$$E_p(S_{DnS}) \leq 3E_p(S_{OPT}) \quad p = 1,2$$
Proof for $E_1$

- $\lambda_t$: the representative of point $t$ in the optimal segmentation
- $\tau$: the representative of point $t$ in the segmentation of Step 2

**Lemma:** $\sum_{t \in T} d_1(t, \tau) \leq \sum_{t \in T} d_1(t, \lambda_t)$
Proof:

- \( \lambda_t \): the representative of point \( t \) in the optimal segmentation
- \( \tau \): the representative of point \( t \) in the segmentation of Step 2
- \( \mu_t \): the representative of point \( t \) in the final segmentation in Step 4

**Lemma**: \[
\sum_{t \in T} d_1(t, \tau) \leq \sum_{t \in T} d_1(t, \lambda_t)
\]

\[E_1(S_{DnS}) = \sum_{t \in T} d_1(t, \mu_t) \]

\[\leq \sum_{t \in T} (d_1(t, \tau) + d_1(\tau, \mu_t)) \quad \text{(triangle inequality)}\]

\[\leq \sum_{t \in T} (d_1(t, \tau) + d_1(\tau, \lambda_t)) \quad \text{(optimality of DP)}\]

\[\leq \sum_{t \in T} (d_1(t, \tau) + d_1(\tau, t) + d_1(t, \lambda_t)) \quad \text{(triangle inequality)}\]

\[\leq 2 \sum_{t \in T} d_1(t, \lambda_t) + \sum_{t \in T} d_1(t, \lambda_t) \quad \text{(Lemma)}\]

\[= 3E(S_{OPT})\]
Trading speed for accuracy

- Recursively divide (into \( m \) pieces) and segment

- If \( \chi=(n_i)^{1/2} \), where \( n_i \) the length of the sequence in the \( i \)-th recursive level (\( n_1=n \)) then
  - running time of the algorithm is \( O(n \log \log n) \)
  - the segmentation error is at most \( O(\log n) \) worse than the optimal

- If \( \chi=\text{const} \), the running time of the algorithm is \( O(n) \), but there are no guarantees for the segmentation error
Real datasets – DnS algorithm
Real datasets – DnS algorithm
Speed vs. accuracy in practice