Time series are everywhere…

- Time series, that is, sequences of observations made through time, are present in everyday’s life:
  - Temperature, rainfalls, seismic traces
  - Weblogs
  - Stock prices
  - EEG, ECG, blood pressure
  - Enrolled students at the Engineering Faculty
  - …

- This as well as many of the following figures/examples are taken from the tutorial given by Eamonn Keogh at SBBD 2002 (XVII Brazilian Symposium on Databases)

www.cs.ucr.edu/~eamonn/
Why is similarity search in t.s.’s important?

- Consider a large time series DB:
  - 1 hour of ECG data: 1 GByte
  - Typical Weblog: 5 GBytes per week
  - Space Shuttle DB: 158 GBytes
  - MACHO Astronomical DB: 2 TBytes, updated with 3 GBytes a day
    (20 million stars recorded nightly for 4 years)
    http://wwwmacho.anu.edu.au/

- Similarity search can help you in:
  - Looking for the occurrence of known patterns
  - Discovering unknown patterns
  - Putting “things together” (clustering)
  - Classifying new data
  - Predicting/extrapolating future behaviors
  - ...

How to measure similarity

- Given two time series of equal length D, the commonest way to measure their (dis-)similarity is based on Euclidean distance
- However, with Euclidean distance we have to face two basic problems
  1. High-dimensionality: (very) large D values
  2. Sensitivity to “alignment of values”

- For problem 1, we need to define effective lower-bounding techniques that work in a (much) lower dimensional space
- For problem 2, we will introduce a new similarity criterion

\[ L_2(s,q) = \sqrt{\sum_{t=0}^{D-1} (s_t - q_t)^2} \]
Dimensionality reduction: DFT (1)

- The first approach to reducing the dimensionality of time series, proposed in [AFS93], was based on Discrete Fourier Transform (DFT)
- **Remind:** given a time series \( s \), the Fourier coefficients are complex numbers (amplitude, phase), defined as:
  \[
  S_f = \frac{1}{\sqrt{D}} \sum_{t=0}^{D-1} s_t \exp(-j2\pi ft/D) \quad f = 0, \ldots, D-1
  \]
- From **Parseval theorem** we know that DFT preserves the energy of the signal:
  \[
  \sum_{t=0}^{D-1} s_t^2 = \sum_{f=0}^{D-1} |S_f|^2
  \]
- Since DFT is a **linear transformation** we have:
  \[
  \sum_{t=0}^{D-1} (s_t - q_t)^2 = \sum_{f=0}^{D-1} |S_f - Q_f|^2 = \sum_{t=0}^{D-1} |S_f|^2 - \sum_{f=0}^{D-1} |Q_f|^2
  \]
  thus, DFT preserves the Euclidean distance
- And? What can we gain from such transformation??

Dimensionality reduction: DFT (2)

- The key observation is that, by keeping only a small set of Fourier coefficients, we can obtain a good approximation of the original signal
- **Why:** because most of the energy of many real-world signals concentrates in the low frequencies ([AFS+93]):
  - More precisely, the energy spectrum (\(|S_f|^2 \text{ vs. } f|\) behaves as \(O(f^b)\), \(b > 0\):
    - \(b = 2\) (random walk or brown noise): used to model the behavior of stock movements and currency exchange rates
    - \(b > 2\) (black noise): suitable to model slowly varying natural phenomena (e.g., water levels of rivers)
    - \(b = 1\) (pink noise): according to Birkhoff’s theory, musical scores follow this energy pattern
  - Thus, if we only keep the first few coefficients (\(D' << D\)) we can achieve an effective dimensionality reduction
- **Note:** this is the basic idea used by well-known compression standards, such as JPEG (which is based on Discrete Cosine Transform)
- For what we have seen, this “projection” technique satisfies the L-B lemma
An example: EEG data

- Sampling rate: 128 Hz

Time series (4 secs, 512 points)  
Energy spectrum

Another example

- 128 points

s' = approximation of s with 4 Fourier coefficients

<table>
<thead>
<tr>
<th>data values</th>
<th>Fourier coefficients</th>
<th>First 4 Fourier coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4995</td>
<td>1.5698</td>
<td>1.5698</td>
</tr>
<tr>
<td>0.5084</td>
<td>1.0485</td>
<td>1.0485</td>
</tr>
<tr>
<td>0.5023</td>
<td>0.7160</td>
<td>0.7160</td>
</tr>
<tr>
<td>0.5761</td>
<td>0.8406</td>
<td>0.8406</td>
</tr>
<tr>
<td>0.5973</td>
<td>0.3705</td>
<td>0.3705</td>
</tr>
<tr>
<td>0.6153</td>
<td>0.4670</td>
<td>0.4670</td>
</tr>
<tr>
<td>0.6301</td>
<td>0.2667</td>
<td>0.2667</td>
</tr>
<tr>
<td>0.6540</td>
<td>0.1928</td>
<td>0.1928</td>
</tr>
<tr>
<td>0.6515</td>
<td>0.1635</td>
<td></td>
</tr>
<tr>
<td>0.6590</td>
<td>0.1622</td>
<td></td>
</tr>
<tr>
<td>0.6972</td>
<td>0.9922</td>
<td></td>
</tr>
<tr>
<td>0.6751</td>
<td>0.1282</td>
<td></td>
</tr>
<tr>
<td>0.6843</td>
<td>0.7433</td>
<td></td>
</tr>
<tr>
<td>0.6954</td>
<td>0.1416</td>
<td></td>
</tr>
<tr>
<td>0.7086</td>
<td>0.1400</td>
<td></td>
</tr>
<tr>
<td>0.7200</td>
<td>0.1412</td>
<td></td>
</tr>
<tr>
<td>0.7412</td>
<td>0.1530</td>
<td></td>
</tr>
<tr>
<td>0.7595</td>
<td>0.1705</td>
<td></td>
</tr>
<tr>
<td>0.7799</td>
<td>0.1013</td>
<td></td>
</tr>
<tr>
<td>0.7956</td>
<td>0.1150</td>
<td></td>
</tr>
<tr>
<td>0.8115</td>
<td>0.1801</td>
<td></td>
</tr>
<tr>
<td>0.8247</td>
<td>0.1082</td>
<td></td>
</tr>
<tr>
<td>0.8345</td>
<td>0.0812</td>
<td></td>
</tr>
<tr>
<td>0.8407</td>
<td>0.0347</td>
<td></td>
</tr>
<tr>
<td>0.8431</td>
<td>0.0052</td>
<td></td>
</tr>
<tr>
<td>0.8423</td>
<td>0.0117</td>
<td></td>
</tr>
<tr>
<td>0.8387</td>
<td>0.0002</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Comments on DFT

- Can be computed in $O(D \log D)$ time using FFT (provided $D$ is a power of 2)
- Difficult to use if one wants to deal with sequences of different length
- Not really amenable to deal with “signals with spots” (time-varying energy)
- An alternative to DFT is to use wavelets, which takes a different perspective:
  - A signal can be represented as a sum of contributions, each at a different resolution level
  - Discrete Wavelet Transform (DWT) can be computed in $O(D)$ time
- Experimental results however show that the superiority of DWT w.r.t. DFT is dependent on the specific dataset

Dimensionality reduction: PAA

- PAA (Piecewise Aggregate Approximation) [KCP+00, YF00] is a very simple, intuitive and fast ($O(D)$) method to approximate time series
  - Its performance is comparable to that of DFT and DWT
- We take a window of size $W$ and segment our time series into $D' = D/W$ “pieces” (sub-sequences), each of size $W$
- For each piece, we compute the average of values, i.e.
- Our approximation is therefore $s' = (s'_1, \ldots, s'_{D'})$
- We have $\sqrt{Wx} L_2(s', q') \leq L_2(s, q)$
  - (arguments generalize those used for the “global average” example)
  - The same can be generalized to work with arbitrary $L_p$-norms [YF00]
The “alignment” problem

- Euclidean distance, as well as other Lp-norms, are not robust w.r.t., even small, contractions/expansions of the signal along the time axis
  - E.g., speech signals
- Intuitively, we would need a distance measure that is able to “match” a point of time series \( s \) even with “surrounding” points of time series \( q \)
  - Alternatively, we may view the time axis as a “stretchable” one
- A distance like this exists, and is called “Dynamic Time Warping” (DTW)!

![Fixed Time Axis](image1)

Sequences are aligned “one to one”

![Warped Time Axis](image2)

Non-linear alignments are possible

How to compute the DTW (1)

- Assume that the two time series \( s \) and \( q \) have the same length \( D \)
  - Note that with DTW this is not necessary anymore!
- Construct a \( D \times D \) matrix \( d \), whose element \( d_{ij} \) is the distance between \( s_i \) and \( q_j \)
  - We take \( d_{ij} = (s_i - q_j)^2 \), but other possibilities exist (e.g., \( |s_i - q_j| \))

<table>
<thead>
<tr>
<th>( D=6 )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>( q )</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

\[ L_2(s,q) = \sqrt{29} \]

- The “rules of the game”:
  - Start from \((0,0)\) and end in \((D-1,D-1)\)
  - Take one step at a time
  - At each step, move only by increasing \( i, j \), or both
    - I.e., never go back!
  - “Jumps” are not allowed!
  - Sum all distances you have found in the “warping path”

\[
\begin{array} {cccccccc}
&  &  &  &  &  &  & \\
&  &  &  &  &  &  & \\
&  &  &  &  &  &  & \\
&  &  &  &  &  &  & \\
&  &  &  &  &  &  & \\
&  &  &  &  &  &  & \\
7 & 25 & 16 & 25 & 36 & 16 & 0 & \\
3 & 1 & 0 & 1 & 4 & 0 & 1 & \\
4 & 4 & 1 & 4 & 9 & 1 & 0 & \\
5 & 9 & 4 & 9 & 16 & 4 & 1 & \\
2 & 0 & 1 & 0 & 1 & 4 & & \\
1 & 1 & 4 & 1 & 0 & 4 & 9 & \\
\end{array}
\]
How to compute the DTW (2)

- The figure shows a possible warping path $w$, whose “cost” is 21
  - The “Euclidean path” moves only along the main diagonal, and costs 29

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>25</th>
<th>16</th>
<th>25</th>
<th>36</th>
<th>16</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The DTW is the minimum cost among all the warping paths

- But the number of path is exponential in $D$
- Ok, but we can use dynamic programming, with complexity $O(D^2)$

How to compute the DTW (3)

- From the $d$ matrix, incrementally build a new matrix $WP$, whose elements $wp_{ij}$ are recursively defined as:
  \[
  wp_{ij} = d_{ij} + \min\{wp_{i-1,j}, wp_{i,j-1}, wp_{i-1,j-1}\}
  \]

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>25</th>
<th>16</th>
<th>25</th>
<th>36</th>
<th>16</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Then set
  \[
  d_{DTW}(s,q) = \sqrt{wp_{0,0}}
  \]
A real-world graphical example

**Power-Demand time series**
Each sequence corresponds to a week’s demand for power in a Dutch research facility in 1997

- Monday was a holiday
- Wednesday was a holiday

---

**Fast searching with DTW**

- We have now 2 problems to face, if we want to use DTW for searching:
  1. Computing the DTW is very time-consuming
  2. How to index it?

- Both problems can be solved:
  1. Use a lower-resolution approximation of the time series
     - However the method can introduce false dismissals

<table>
<thead>
<tr>
<th>Time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3 sec</td>
<td></td>
</tr>
<tr>
<td>22.7 sec</td>
<td></td>
</tr>
</tbody>
</table>
Indexing the DTW (sketch) (1)

- An effective indexing technique for DTW has been proposed in [Keo02].
- The method applies only if we have some "global constraint" on the allowed warping paths.

The Sakoe-Chiba band of width $h=4$

Indexing the DTW (sketch) (2)

- The idea is to create an "envelope", whose size depends on $h$, of the query $q$...

\[ U_i = \max\{q_{ih}, \ldots, q_{i+h}\} \]
\[ L_i = \min\{q_{ih}, \ldots, q_{i+h}\} \]

- ...then each time series $s$ in the DB is approximated using PAA...
- ...a distance function between an approximation of the query envelope (which depends on the window size $W$ of PAA) and the PAA representation $s'$ of $s$ is then defined...
- ...it is proved that this distance function lower bounds $d_{DTW}$...
- ...and that PAA approximations can be indexed by an R-tree.
Some experimental results (from [Keo02])

- Two datasets (left: synthetic; right: a mix of real data)

![Graph showing normalized CPU cost vs DB size for Random Walk II and Mixed Bag datasets.]

Final considerations

- We have just seen some basic techniques to deal with (large) time series databases

- Other relevant problems exist and have attracted interest, among which:
  - Searching for similar sub-sequences
  - Searching for multi-dimensional time series (i.e., trajectories)