Measuring distance/ similarity of data objects

Multiple data types

- Records of users
- Graphs
- Images
- Videos
- Text (webpages, books)
- Strings (DNA sequences)
- Timeseries
- **How do we compare them?**

Feature space representation

- Usually data objects consist of a set of attributes (also known as **dimensions**)

- J. Smith, 20, 200K

- If all d dimensions are **real-valued** then we can **visualize** each data point as points in a d**-dimensional space**

- If all d dimensions are **binary** then we can think of each data point as a **binary vector**

Distance functions

- The distance $d(x, y)$ between two objects $x$and y is a metric if
  - $d(i, j) \geq 0$ (non-negativity)
  - $d(i, i) = 0$ (isolation)
  - $d(i, j) = d(j, i)$ (symmetry)
  - $d(i, j) \leq d(i, h) + d(h, j)$ (triangular inequality) [**Why do we need it?**]
- The definitions of distance functions are usually different for real, boolean, categorical, and ordinal variables.
- Weights may be associated with different variables based on applications and data semantics.

Data Structures

- data matrix

$$\begin{array}{cc} & \text{attributes/dimensions} \\ \text{tuples/objects} & \begin{bmatrix} x_{11} & \cdots & x_{1l} & \cdots & x_{1d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{il} & \cdots & x_{id} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nl} & \cdots & x_{nd} \end{bmatrix} \end{array}$$

- Distance matrix

$$\begin{array}{cc} & \text{objects} \\ \text{objects} & \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix} \end{array}$$

Distance functions for real-valued vectors

- L_p norms or Minkowski distance:

$$L_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- p = 1, L_1, Manhattan (or city block) or Hamming distance:

$$L_1(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i| \right)$$

Distance functions for real-valued vectors

- **L_p** norms or Minkowski distance:

$$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- **p = 2, L_2,** Euclidean distance:

$$L_2(x,y) = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2}$$

Distance functions for real-valued vectors

- Dot product or cosine similarity

$$\cos(x, y) = \frac{x \cdot y}{||x||\,||y||}$$

- Can we construct a distance function out of this?
- When use the one and when the other?

Hamming distance for 0-1 vectors

x	0	1	0	0	1	0	0	1	0
y	1	0	0	0	0	1	0	1	1

$$L_1(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i| \right)$$

How good is Hamming distance for 0-1 vectors?

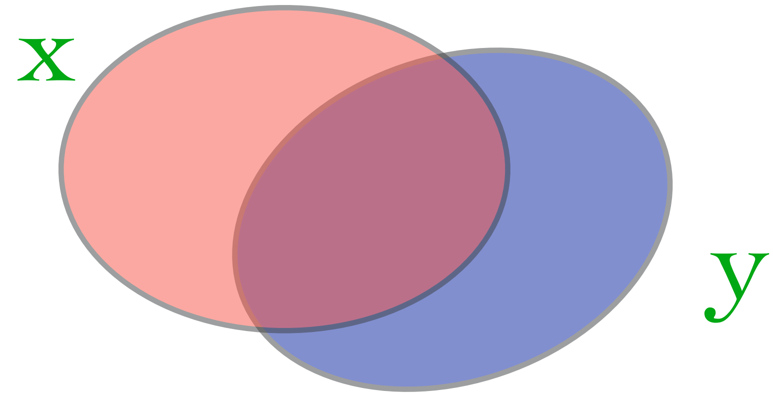
- **Drawback**

- Documents represented as sets (of words)
- Two cases
  - Two **very large** documents -- almost identical -- but for 5 terms
  - Two **very small** documents, with 5 terms each, disjoint

Distance functions for binary vectors or **sets**

- **Jaccard** similarity between binary vectors x and y (Range?)

$$\mathrm{JSim}(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

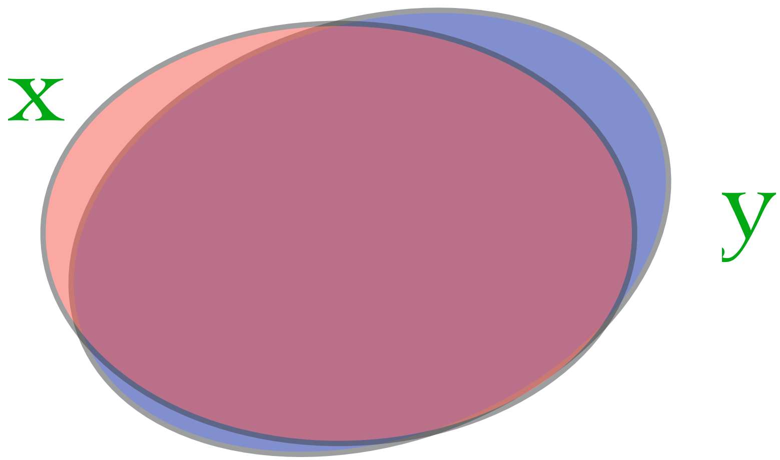


- **Jaccard** distance (Range?):

$$\mathrm{JDist}(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|}$$

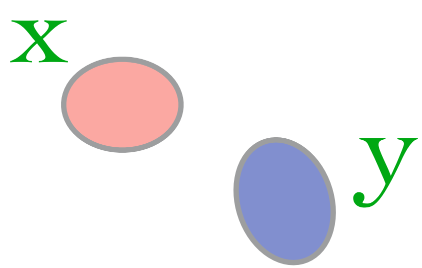
The previous example

- Case 1 (very large almost identical documents)



$J(x, y)$ almost 1

- Case 2 (small disjoint documents)



$J(x, y) = 0$

Jaccard similarity/distance

- Example:
  - JSim = 1/6
  - Jdist = 5/6

	Q1	Q2	Q3	Q4	Q5	Q6
X	1	0	0	1	1	1
Y	0	1	1	0	1	0

Distance functions for strings

- **Edit distance** between two strings x and y is the **min** number of operations required to transform one string to another

- Operations: replace, delete, insert, transpose etc.

Distance functions between strings

- Strings x and y have **equal length**
- Modification of **Hamming** distance
- Add 1 for all positions that are different

x	c	g	t	a	a	c	g
y	g	a	t	t	a	c	a

- Hamming distance = 4
- **Drawbacks?**

Hamming distance between strings -- drawbacks

- Strings should have equal length
- What about

x	a	g	a	t	t	a	c
y	g	a	t	t	a	c	a

- String Hamming distance = 6

Edit Distance

- **Edit distance** between two strings x and y of length n and m resp. is the **minimum** number of single-character edits (insertion, deletion, substitution) required to change one word to the other

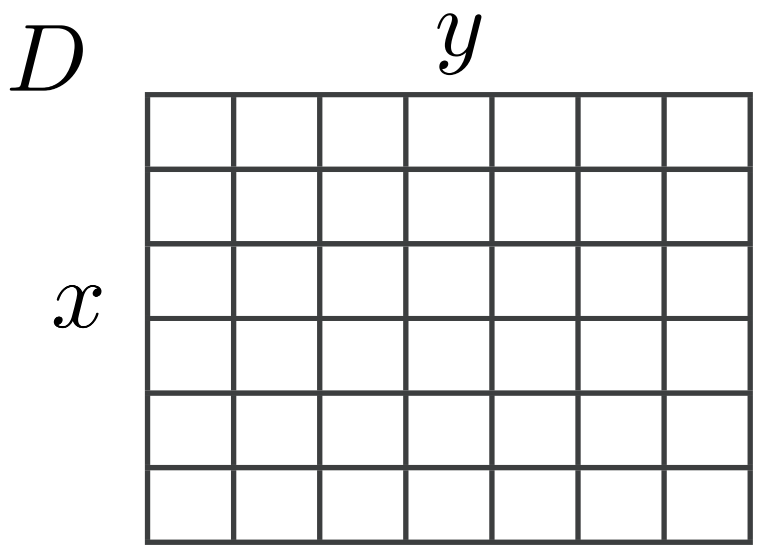
Example

- I N T E N T I O N
- E X E C U T I O N

- I N T E * N T I O N
- * E X E C U T I O N
- d s s   i s

Computing the edit distance

- Dynamic programming
- Form nxm distance matrix D (x of length n, y of length m)



- D(i,j) is the optimal distance between strings x[1..i] and y[1..j]

Computing the edit distance

- How to compute D(i,j)?
- Either
  - match the last two characters (substitution)
  - match by deleting the last char in one string
  - match by deleting the last character in the other string

Computing edit distance

$$
\begin{aligned}
D(i,j) = \min\{&D(i-1,j) + \mathrm{del}(X[i]),\\
&D(i,j-1) + \mathrm{ins}(Y[j]),\\
&D(i-1,j-1) + \mathrm{sub}(X[i],Y[j])\}
\end{aligned}
$$

- Running time? Metric?

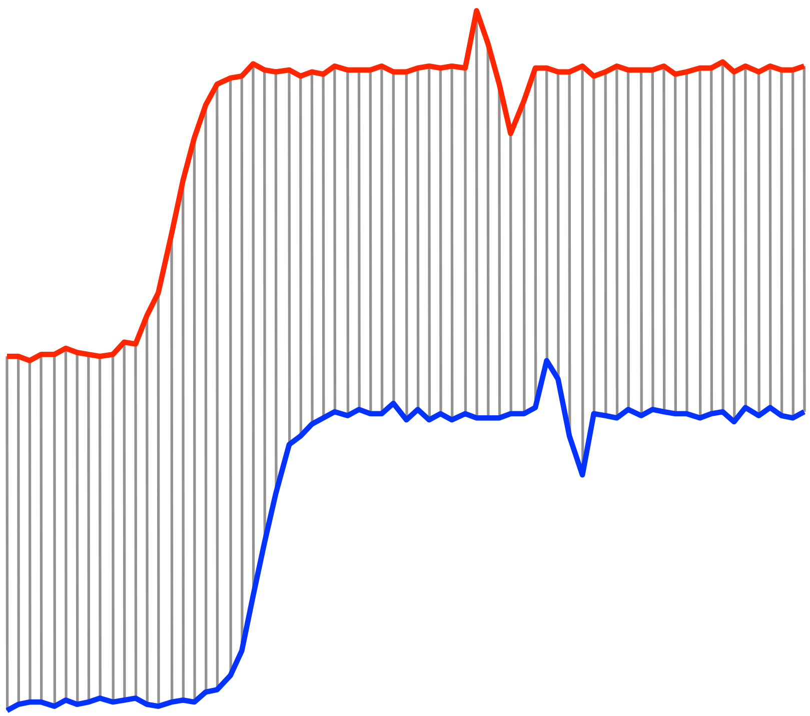
Distance function between time series

- time series can be seen as vectors
- apply existing distance metrics
- L-norms

- what can go wrong?

Distance functions between time series

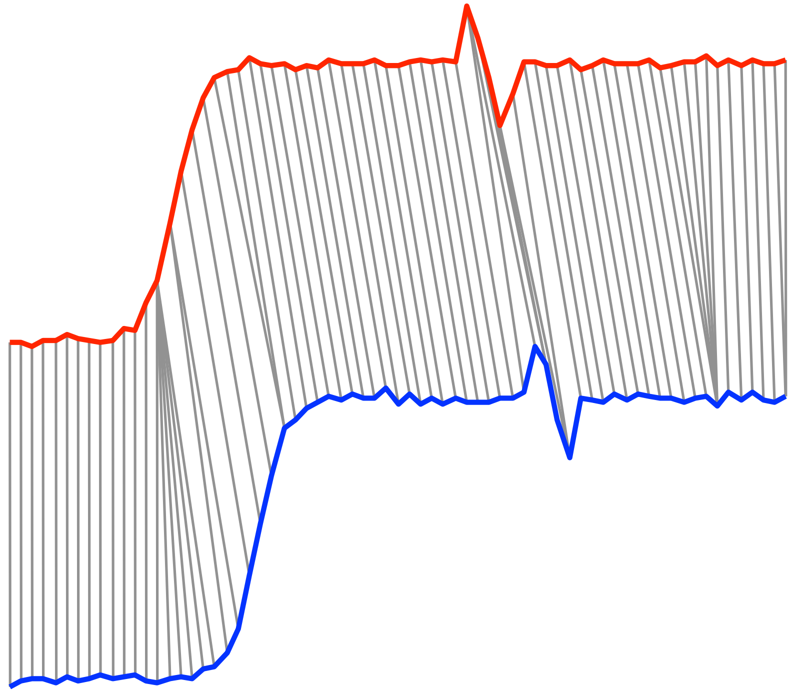
- Euclidean distance between time series



figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

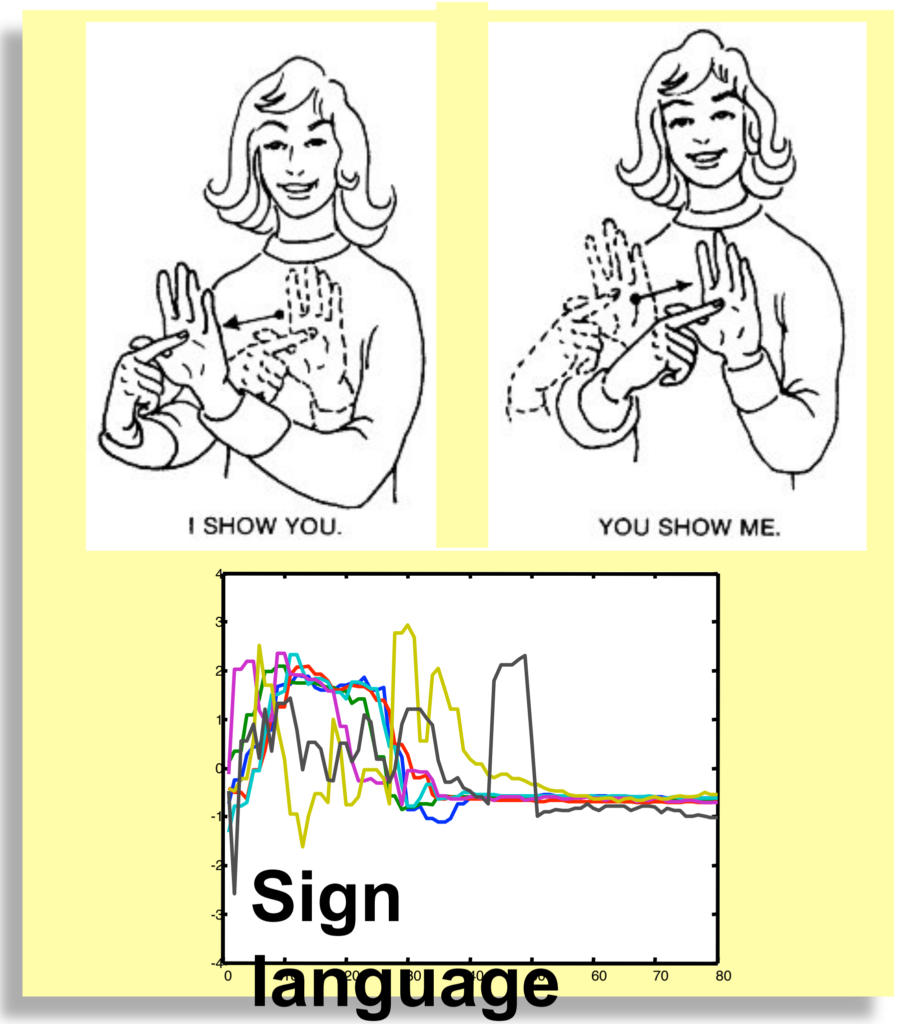
Dynamic time warping

- Alleviate the problems with Euclidean distance



figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

Dynamic time warping

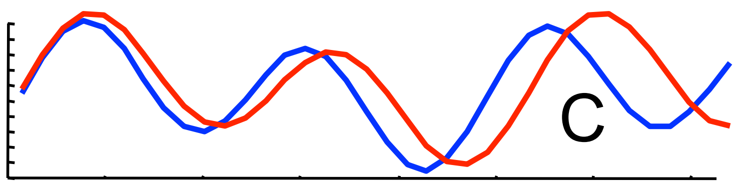


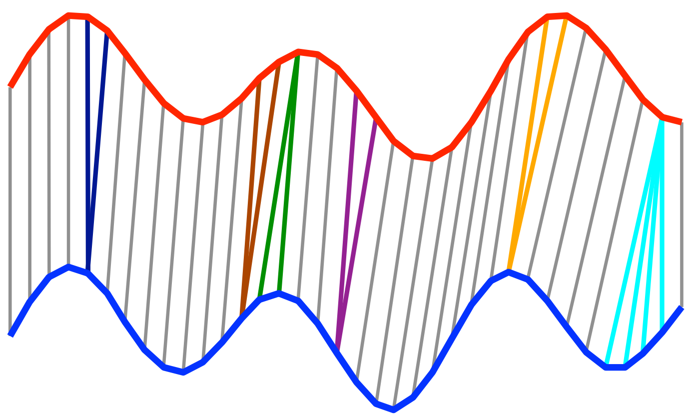
- Quite useful in practice

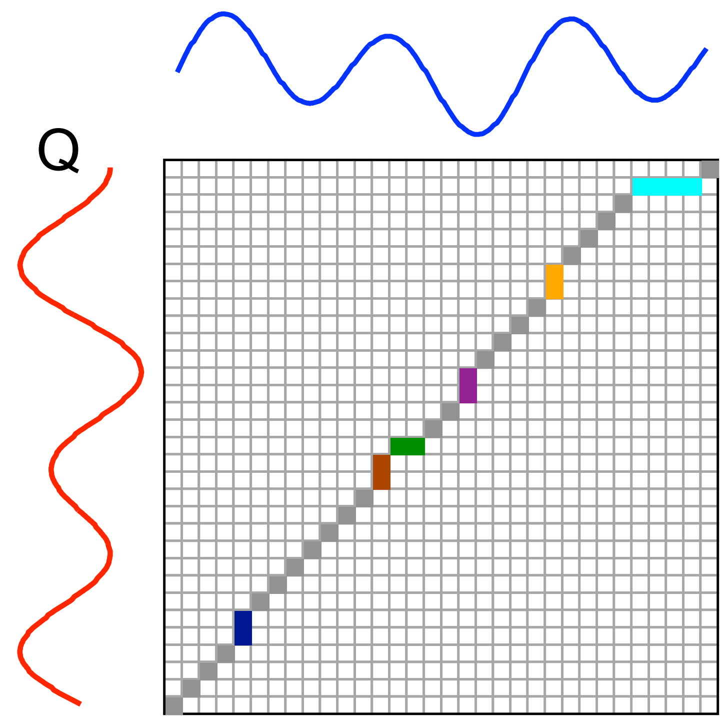
figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

Dynamic time warping

- how to compute it?
- Dynamic programming



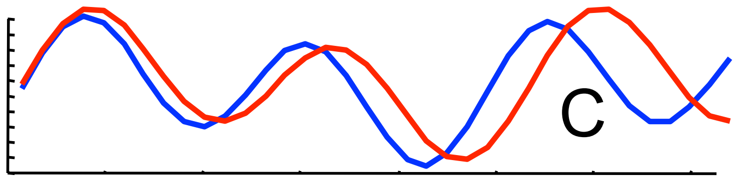


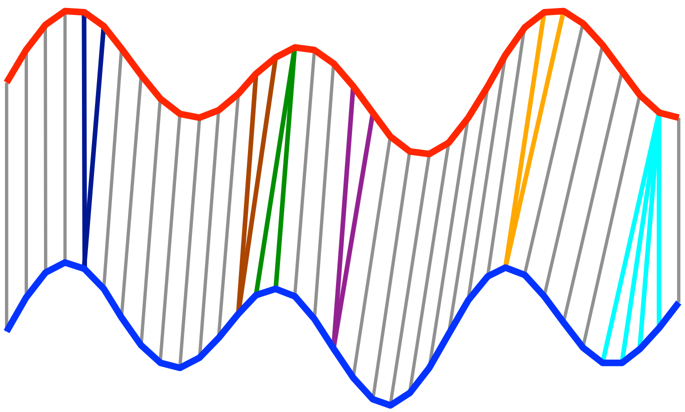


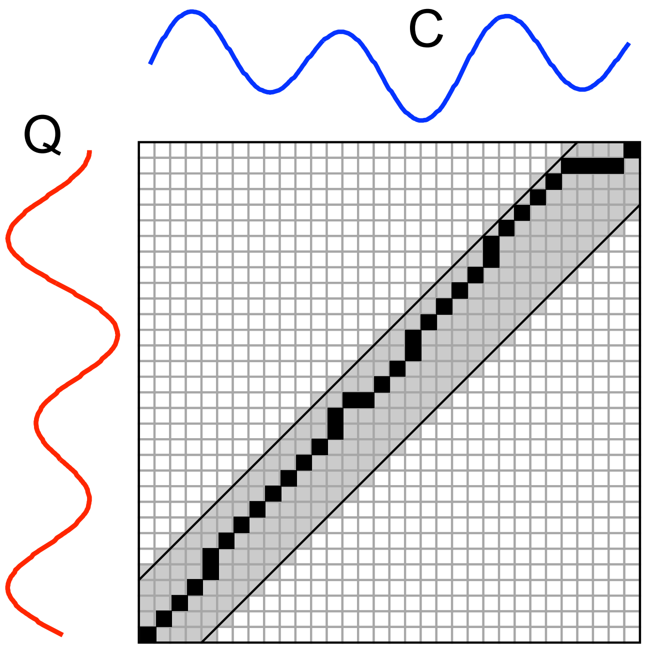
figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

Dynamic time warping

- constraints for more efficient computation







figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt