CS-E4600
Algorithmic methods of data mining

Aristides Gionis
Dept of Computer Science

Slide set 3: Distance functions

reading assignment

RLU book: chapter 3.5

many different data

documents

records of users

graphs

images

videos

strings

time series

distance functions

need to compare objects in the data

are two given users similar?

find the most similar image to a given one..

how likely is for two genomic sequences to be mutation of each other?

distance functions

dataset X as a collection of objects

write x, y, z, ... for objects in X

at this point no assumption about the representation of objects in X

x can be

- real-valued vectors
- binary vectors
- sets
- time series
- images

distance functions

want to define function

$$d : X \times X \rightarrow \mathbb{R}$$

what properties should d have?

distance functions

$d(x, y) \geq 0$ non negativity

$d(x, y) = 0 \text{ iff } x = y$ isolation

$d(x, y) = d(y, x)$ symmetry

$d(x, y) \leq d(x, z) + d(z, y)$ triangle inequality

metric distance functions and metric spaces

a distance function that satisfies all properties

non-negativity,

isolation,

symmetry, and

triangle inequality

is called a metric

a data space equipped with a metric function is called metric space

similarity functions

distance function $d : X \times X \to \mathbb{R}$

large for dissimilar objects

similarity function $s : X \times X \to \mathbb{R}$

large for similar objects

often similarity **s** is between 0 and 1

$$s(x, y) = 1 - d(x, y)$$

$$s(x, y) \propto e^{-d(x,y)}$$

distance functions between real-valued data

dataset $X \subseteq \mathbb{R}^m$

data points $x = [x_1 \ldots x_m], \;\; y = [y_1 \ldots y_m]$

L_p norm or Minkowski distance

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

for p=1, Manhattan (or city-block) distance

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

distance functions between real-valued data

L_p norm or Minkowski distance

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

for p=2, Euclidean distance

$$L_2(x, y) = \sqrt{\sum_{i=1}^{m} |x_i - y_i|^2}$$

distance functions between real-valued data

L_p norm or Minkowski distance

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

for $p \to \infty$, L_∞ distance

$$L_\infty(x,y) = \max_i |x_i - y_i|$$

Minkowski distances

for all p, the Minkowksi distance is a metric

can you show it?

data structures

data matrix

$$\begin{pmatrix} x_{11} & \dots & x_{1m} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nm} \end{pmatrix}$$

distance matrix

$$\begin{pmatrix} 0 & \dots & & & \\ d(2,1) & 0 & \dots & & \\ \vdots & & & & \\ d(n,1) & d(n,2) & \dots & d(n,n-1) & 0 \end{pmatrix}$$

to think: which are the advantages and disadvantages of each representation?

similarity function between real-valued data

dataset $X \subseteq \mathbb{R}^m$

data points $x = [x_1 \dots x_m]$, $y = [y_1 \dots y_m]$

cosine similarity

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

commonly used when vectors represent documents

dot product: without the normalization

when to use cosine and when dot product?

it depends on the application

distance functions between 0/1 data

dataset $X \subseteq \{0,1\}^m$

data points $x = [x_1 \ldots x_m], \quad y = [y_1 \ldots y_m]$

L_p norm or Minkowski distance

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

p=1 (Hamming distance) is used, as all differences are 0 or 1

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

Hamming distance: example

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

Hamming distance = 5

distance functions between sets

ground set U of m elements

dataset $X \subseteq 2^U$

each set $x \in X$ can be seen as a binary vector

use the Hamming distance

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

drawbacks?

Hamming distance for measuring set similarity

drawback

binary vectors are often "symmetric" with respect to 0s and 1s

e.g., they tend to have a comparable number of 0s and 1s

Hamming distance treats 0s and 1s equally

on the other hand, sets (transaction data) tend to be very sparse

this makes the use of Hamming distance problematic

Hamming distance for measuring set similarity

drawback

consider documents represented as sets

consider the following two cases:

1. two very large documents, thousands of terms each, almost identical, except 5 terms
2. two very small documents, with 5 terms each, disjoint

the two cases are conceptually very different,

but the Hamming distance is the same in both case

measuring set similarity using the Jaccard coefficient

consider sets x and y

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

a similarity function between 0 and 1

value 1: sets identical

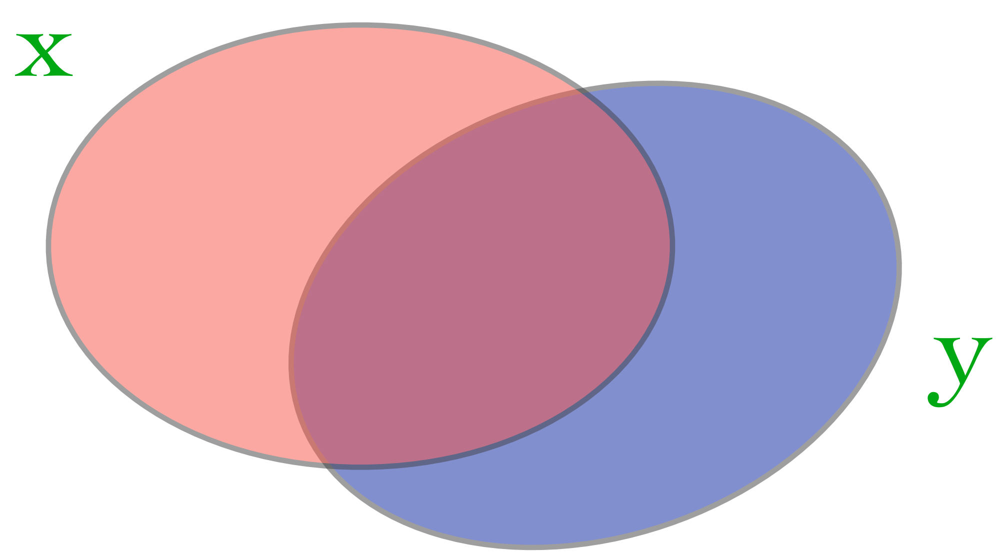
value 0: sets disjoint

the Jaccard coefficient similarity treats 0s and 1s differently

measuring set similarity using the Jaccard coefficient

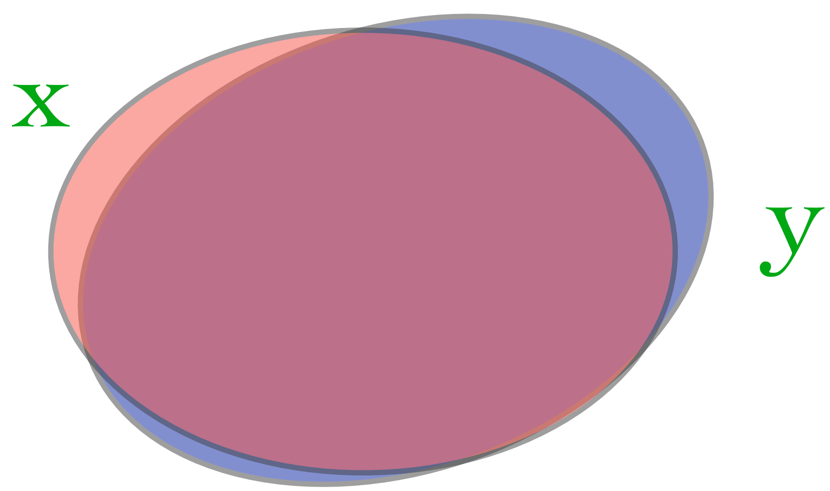
set similarity $$J(x,y) = \frac{|x \cap y|}{|x \cup y|}$$

in Venn diagram:



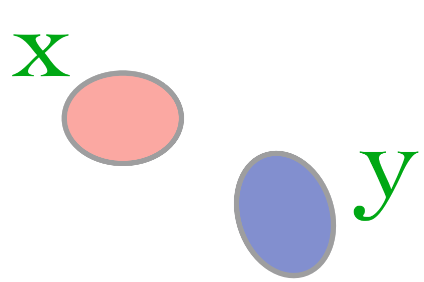
the previous example

case 1



$J(x, y)$ almost 1

case 2



$$J(x, y) = 0$$

distance functions between strings

strings x and y of equal length

modification of the 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

string Hamming distance = 4

drawbacks?

distance functions between strings

1. strings must have equal length
2. what about

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

string Hamming distance = 6

string edit distance

consider two strings x and y

try to change one to another

only single-character edits are allowed

- insert character
- delete character
- substitute character

edit distance is the minimum number of such operations

not necessary to have equal length!

string edit distance

example

x	a	g	a	t	t	a	c	
	g	a	t	t	a	c		remove a
								add a
y	g	a	t	t	a	c	a	

string edit distance = 2

string edit distance

consider two strings x and y of lengths n and m, respectively

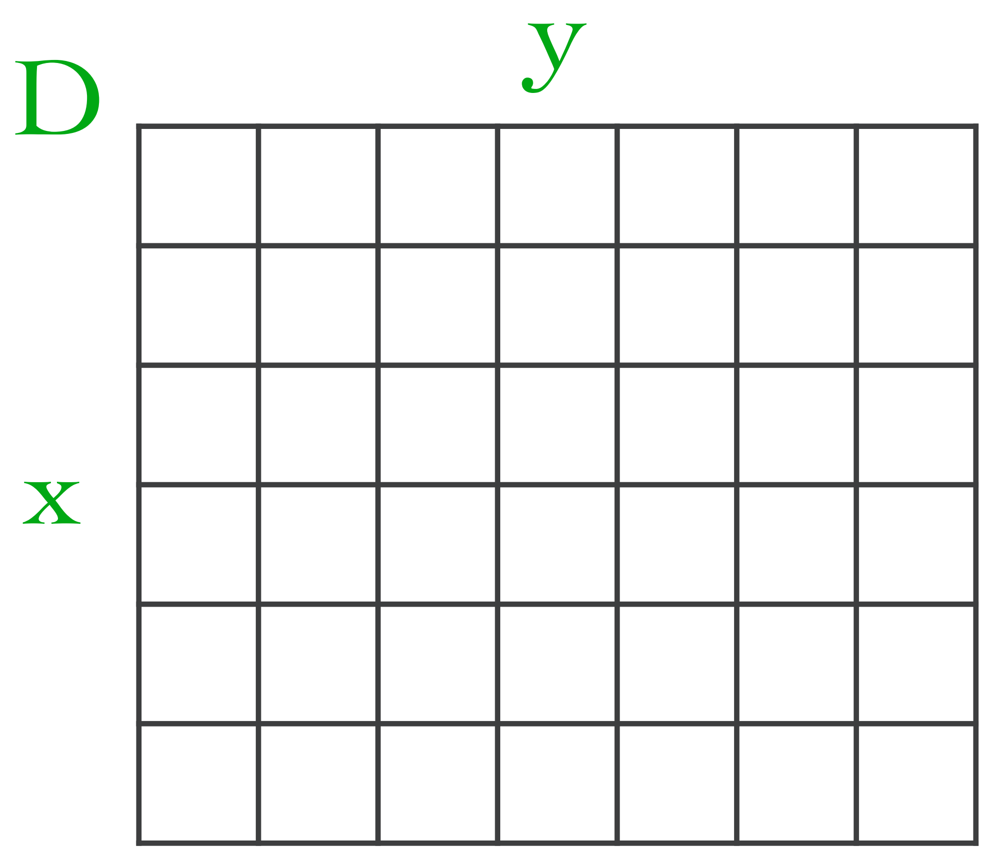
how can I compute the string edit distance between x and y?

how expensive is this computation?

string edit distance

dynamic programming

form n x m distance table D



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

string edit distance

how to compute D(i,j), the distance strings x[1..i] and y[1..j]?

either:

match the last two characters

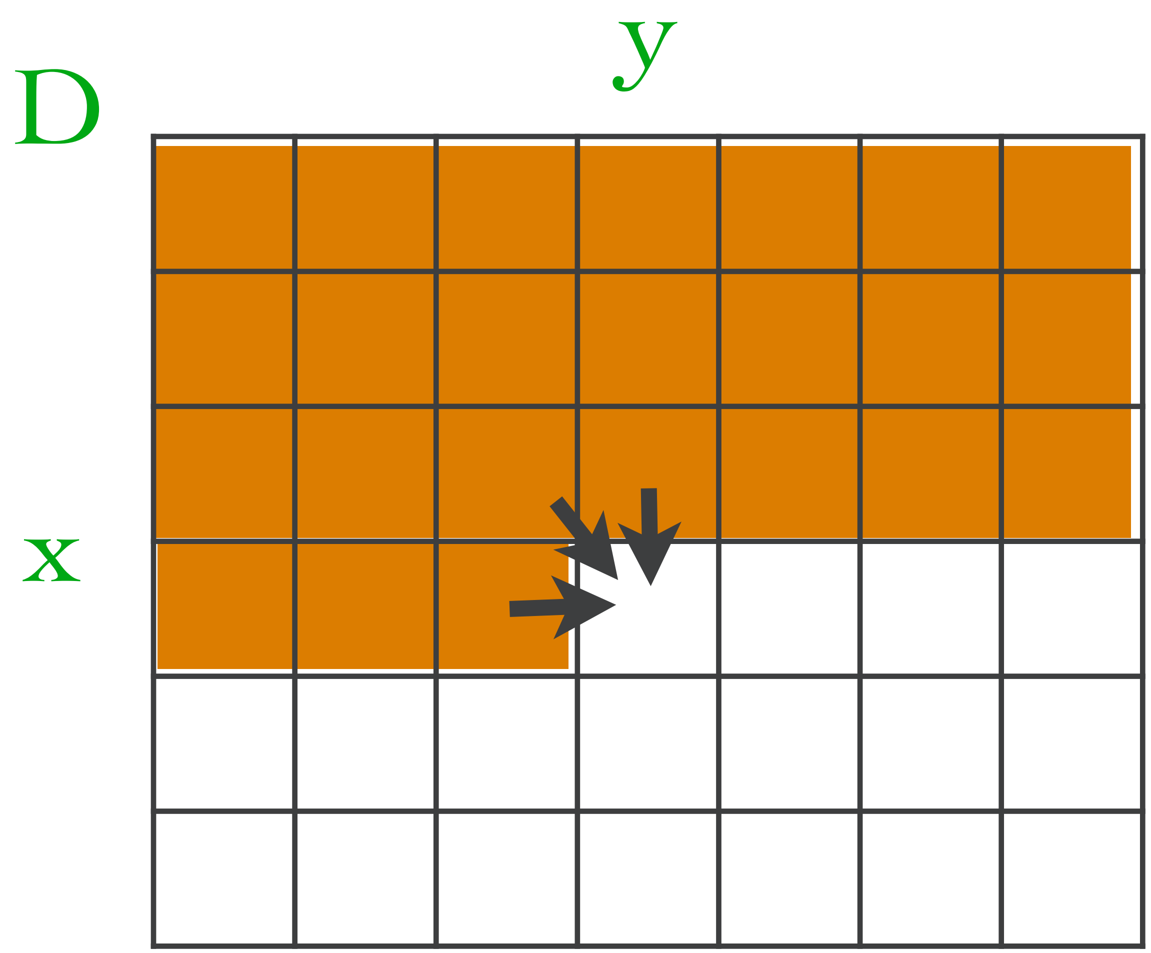
match by deleting the last character in one string

match by deleting the last character in the other string

dynamic programming equation

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

computing string edit distance



how fast can be computed?

O(mn) where m and n are the lengths of the two strings

distance functions between time series

time series can be seen as vectors

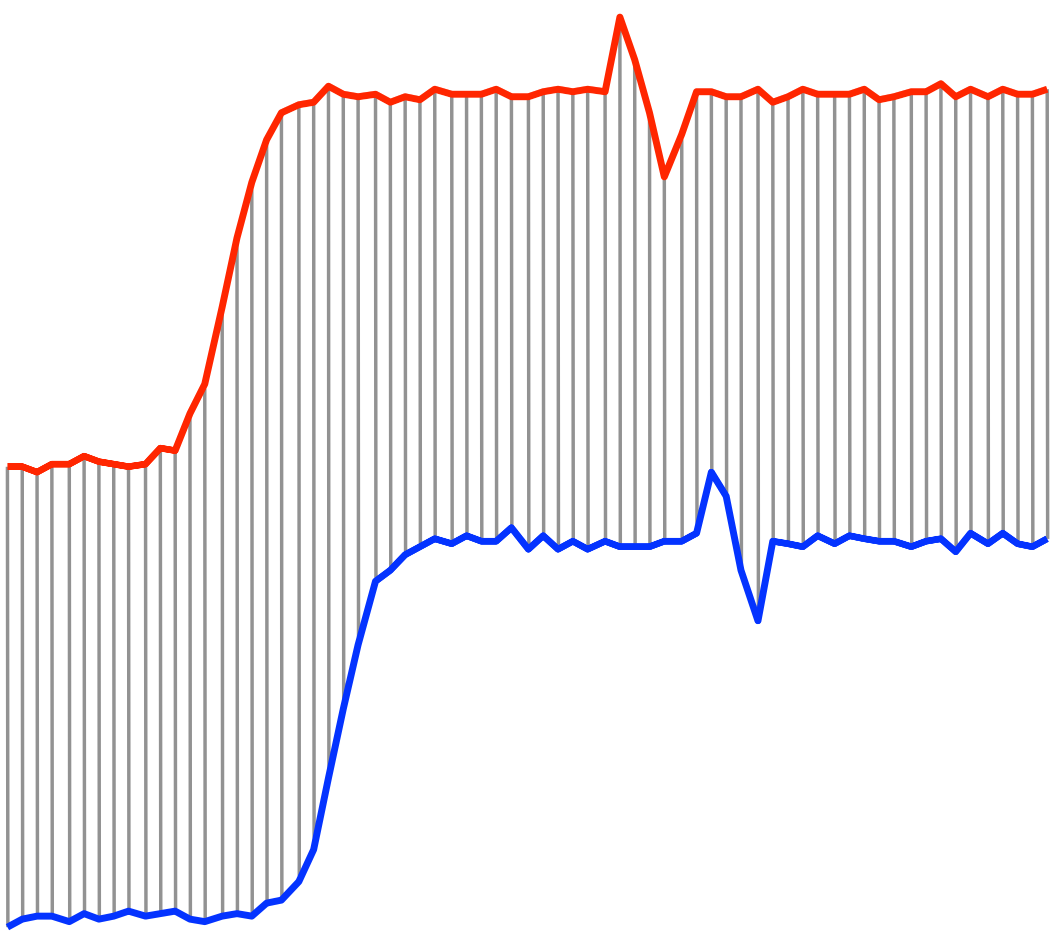
apply existing distance measures

L_2 (Euclidean), L_1, $L_{infinity}$ (max)

what can go wrong?

distance functions between time series

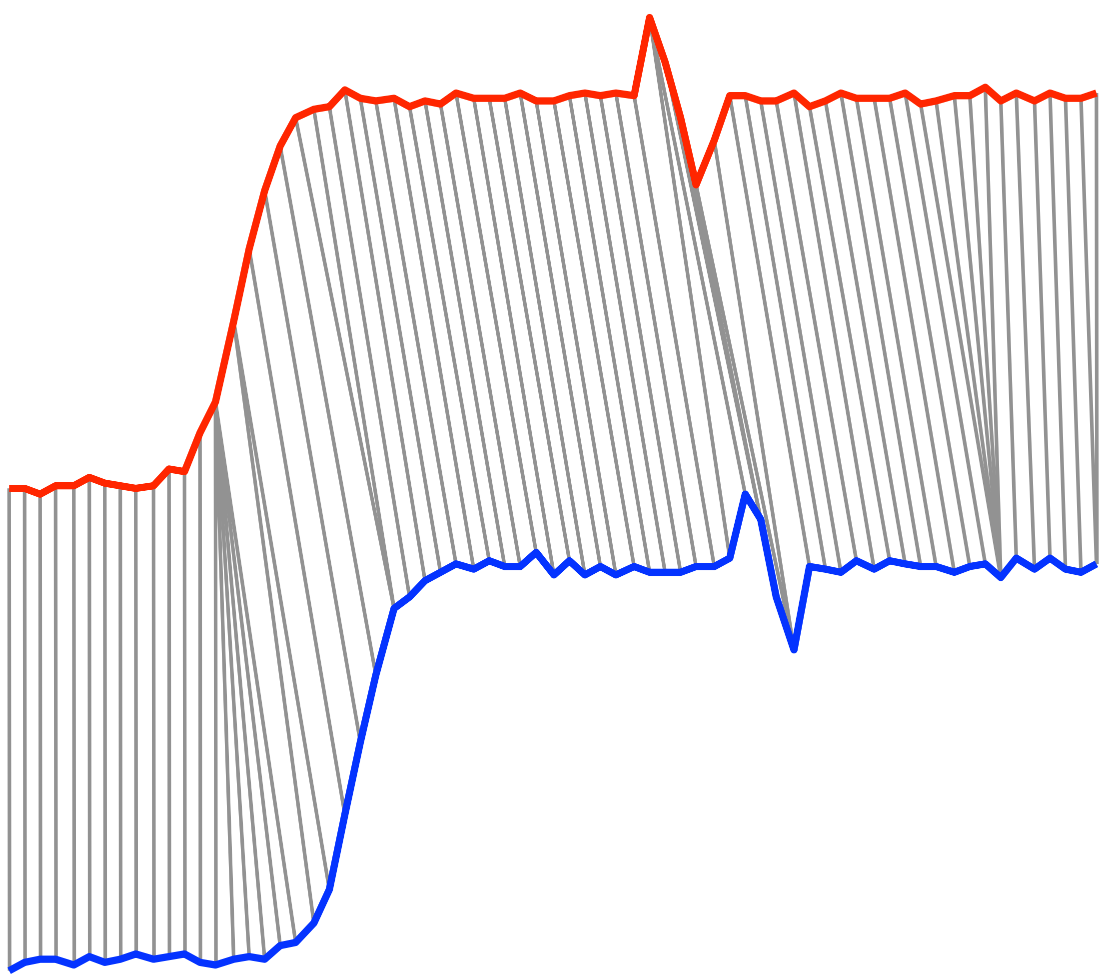
Euclidean distance between time series



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

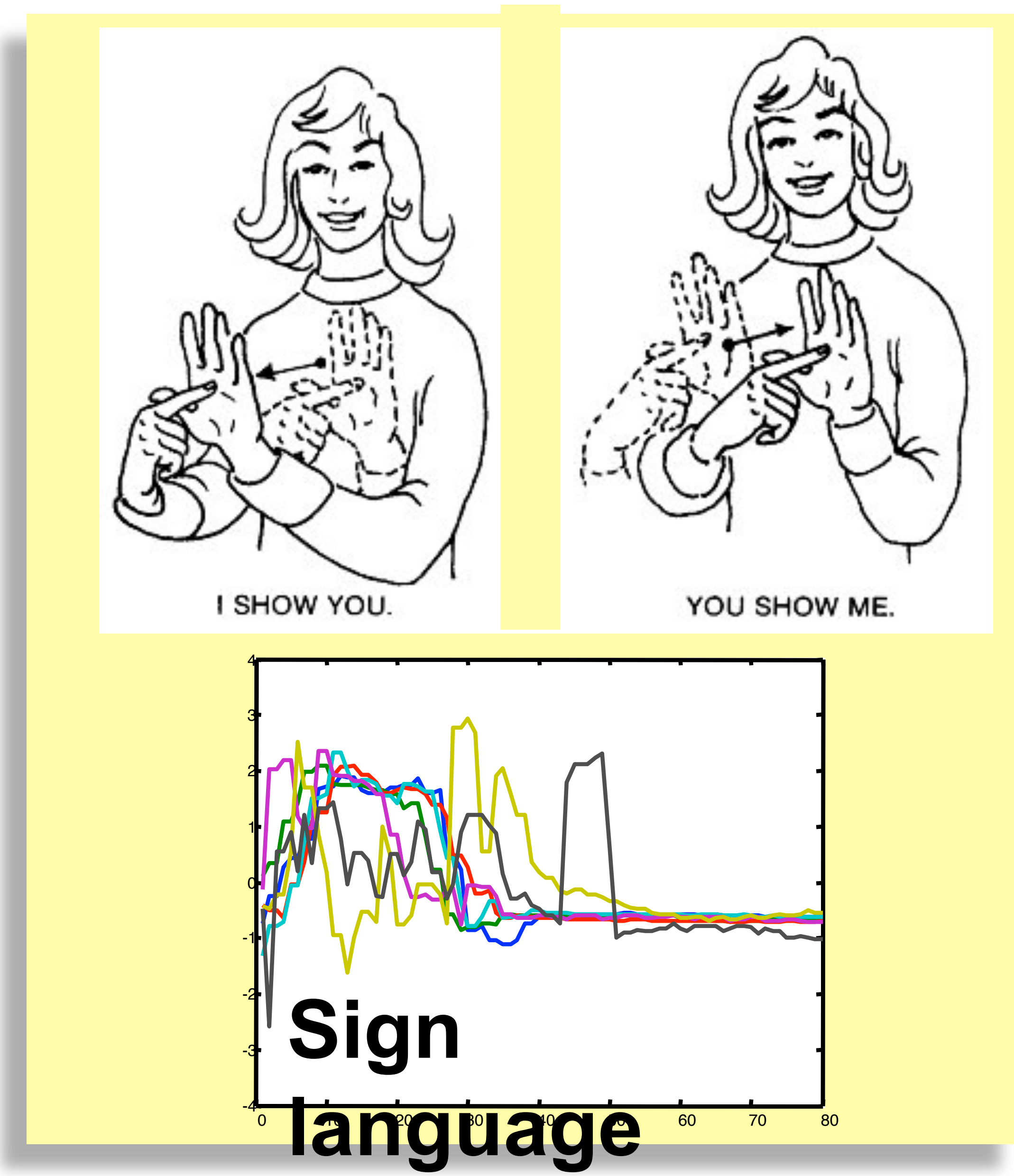
dynamic time warping

alleviate the problems of Euclidean distance



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

dynamic time warping

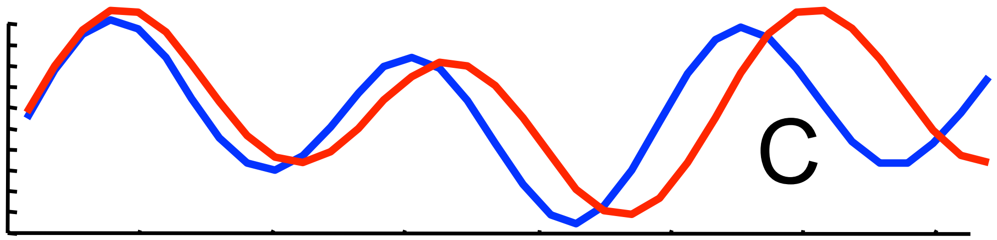


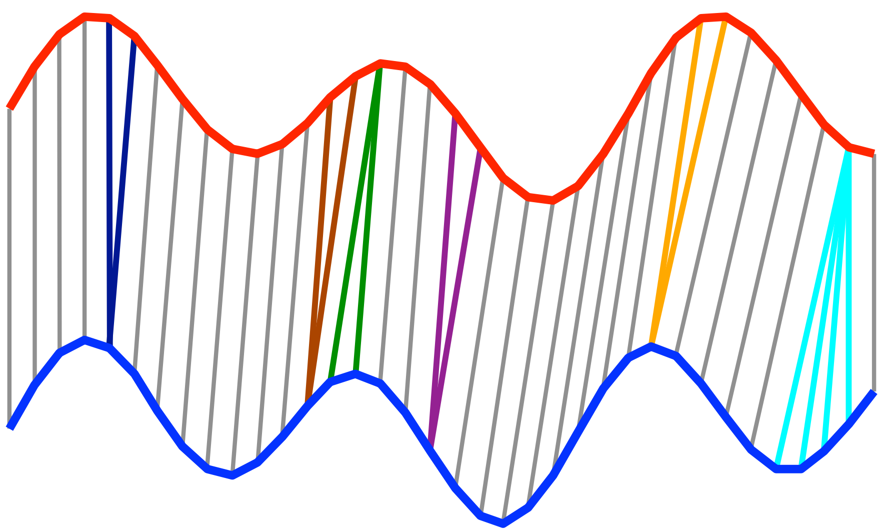
quite useful in practice

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

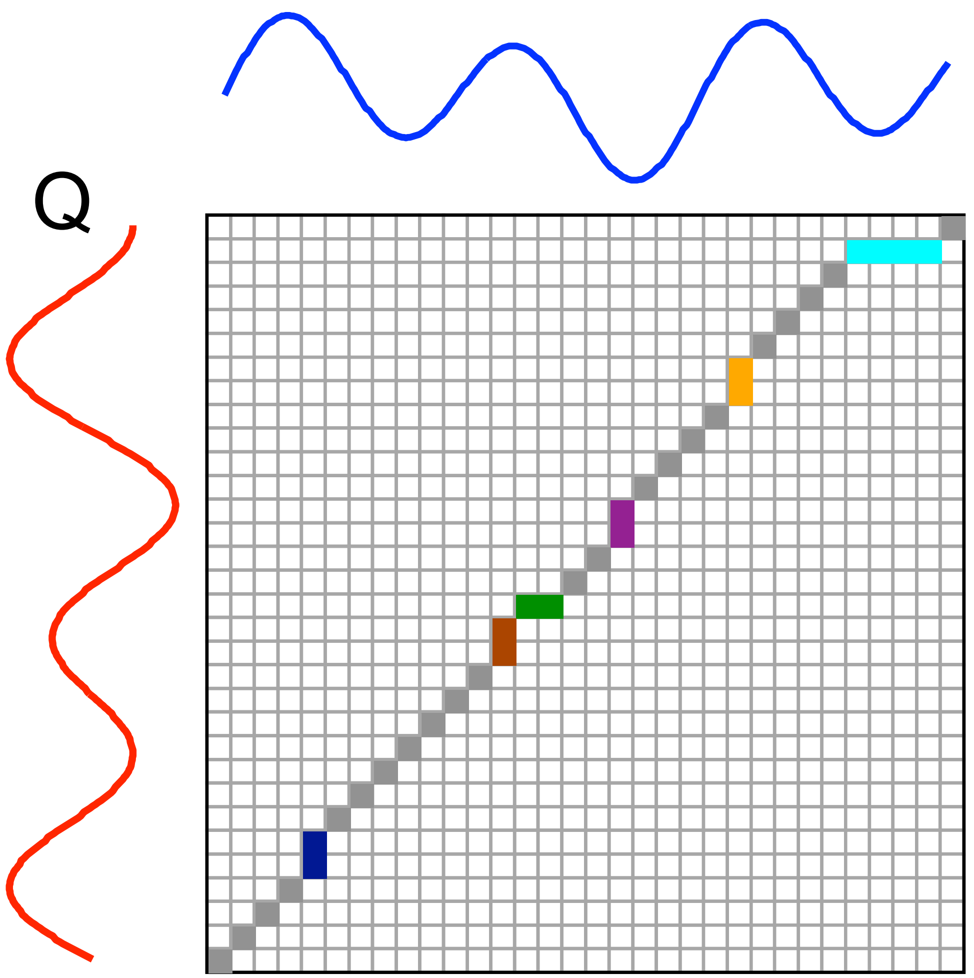
dynamic time warping

how to compute it?





dynamic programming!

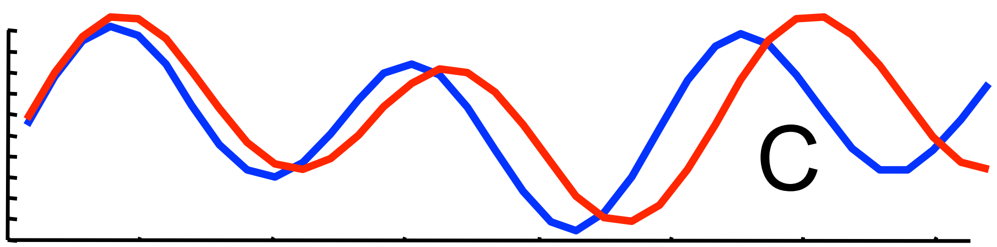


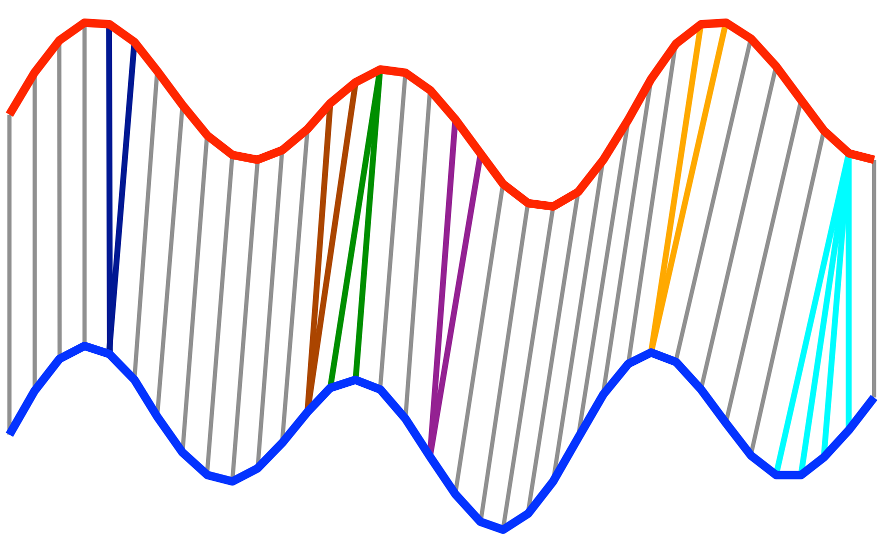
figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

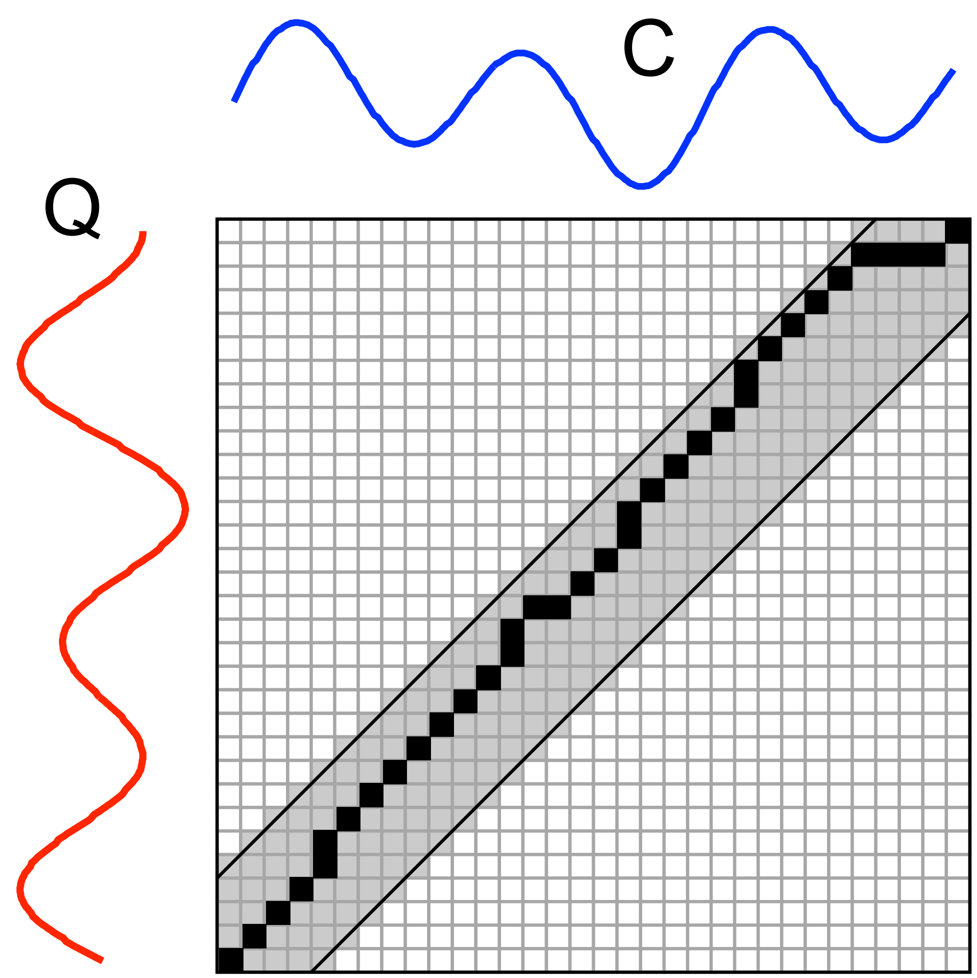
Aalto University

dynamic time warping

constraints for more efficient computation

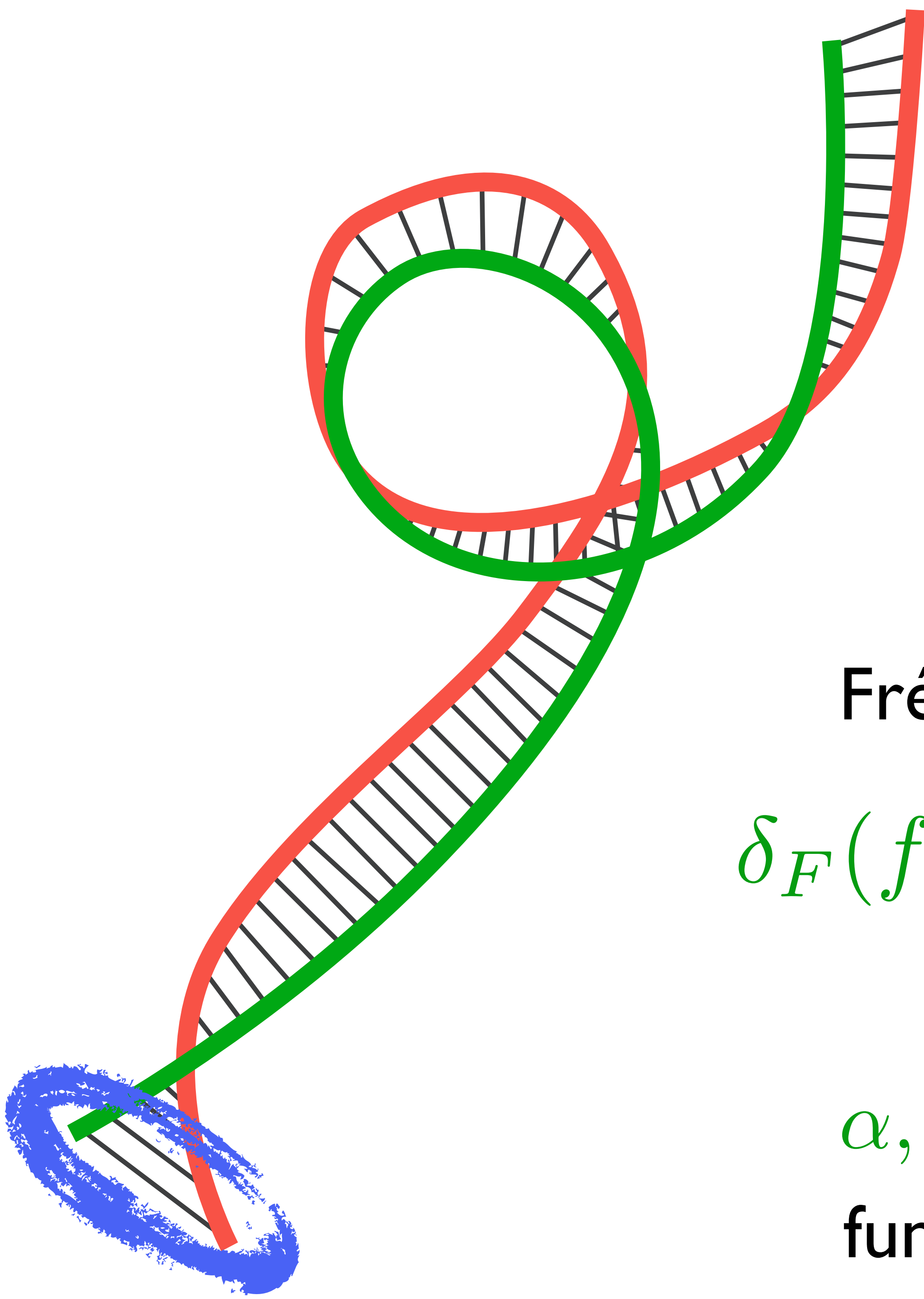






figures from Eamonn Keogh www.cs.ucr.edu/~eamonn/DTW_myths.ppt

comparing curves



metric (V, d)

curves

$$f : [a, b] \to V$$

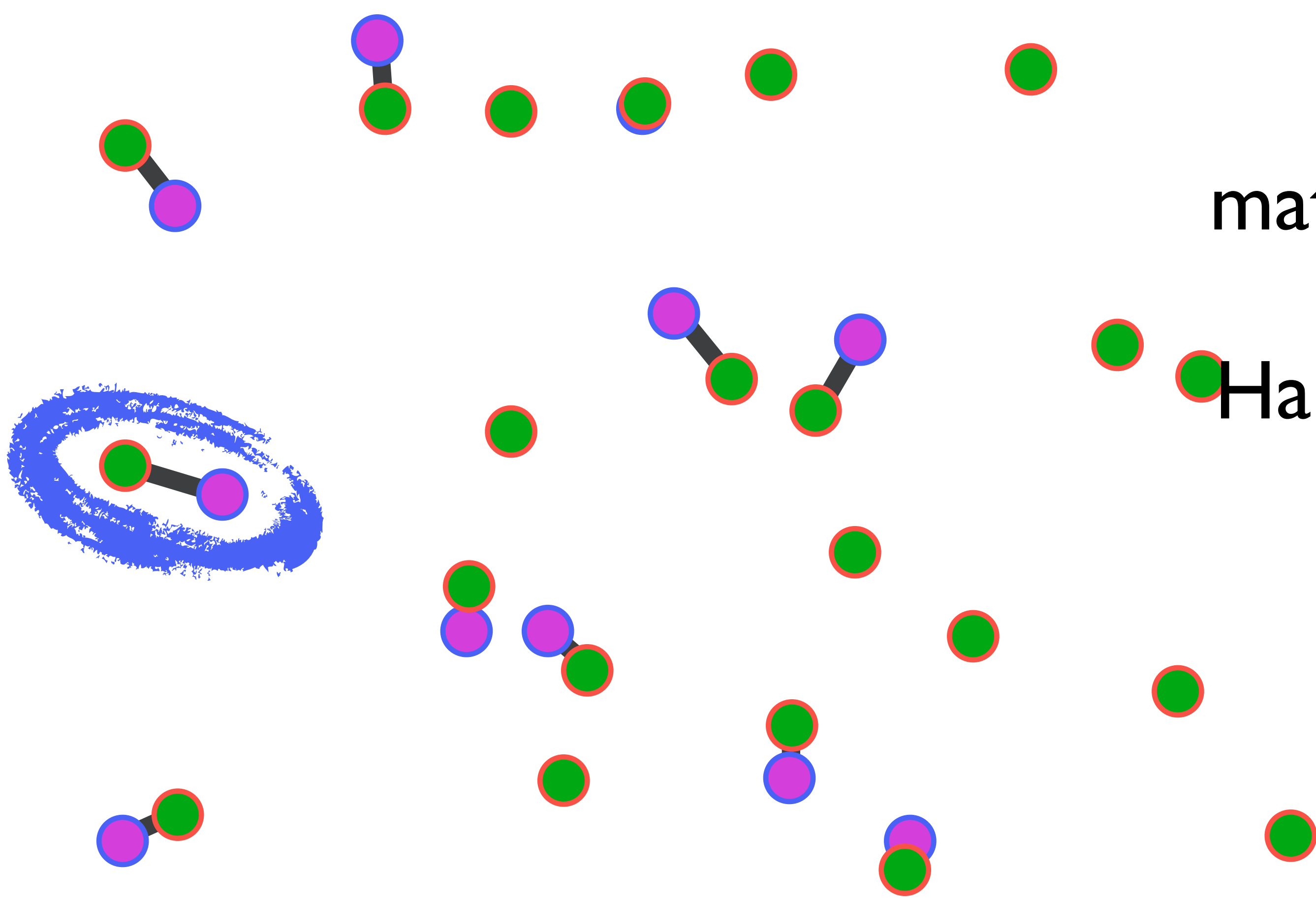
$$g : [a', b'] \to V$$

Fréchet distance:

$$\delta_F(f, g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} d(f(\alpha(t)), g(\beta(t)))$$

α, β arbitrary continuous nondecreasing functions from $[0, 1]$ to $[a, b]$ and $[a', b']$

comparing point sets



matching distance

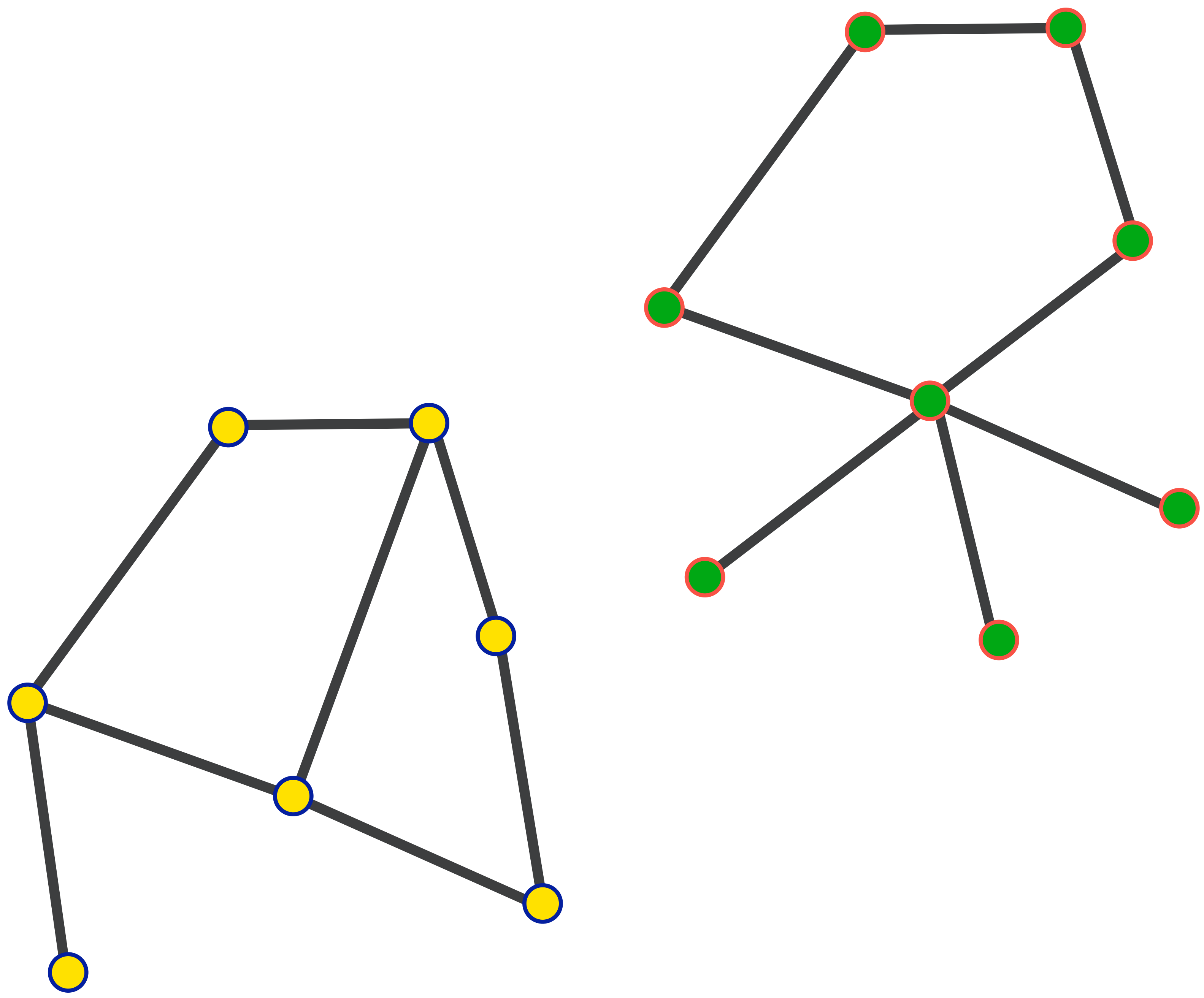
Hausdorff distance

comparing graphs

graph isomorphism

graph edit distance

motifs frequency



summary

defined generic distance functions between objects

considered metric properties of distance functions

defined specific distance functions for

- vectors
- 0/1 vectors
- sets
- strings
- timeseries
- trajectories
- sets of points

embeddings

embeddings

metric spaces (X, d) and (X', d')

mapping $f : X \to X'$

isometric or distance preserving if

$$d(x, y) = d'(f(x), f(y))$$

for all $x, y \in X$

embeddings

why embeddings?

why do I care?

do you know any example of an embedding?

embeddings distortion

contraction

$$\max_{x,y \in X} \frac{d(x,y)}{d'(f(x),f(y))}$$

expansion or stretch

$$\max_{x,y \in X} \frac{d'(f(x),f(y))}{d(x,y)}$$

distortion = contraction x expansion

embeddings distortion

(equivalent definition)

distortion: smallest α such that

$$r\, d(x,y) \leq d'(f(x), f(y)) \leq \alpha\, r\, d(x,y)$$

what is the role of r ?

why embeddings?

some problems can be solved more easily in one metric than another

if can embed X to X' with small distortion

and know how to solve problem on X'

then can solve problem on X approximately

quality of approximation depends on the distortion

L_1 to L_∞

for all vectors $\mathbf{x} \in \mathbb{R}^k$

$$||\mathbf{x}||_1 = \sum_i |x_i| = \sum_i \mathrm{sgn}(x_i) \cdot x_i \geq \sum_i y_i \cdot x_i$$

for all vectors $\mathbf{y} \in \{-1, +1\}^k$

so

$$||\mathbf{x}||_1 = \max\{\mathbf{x} \cdot \mathbf{y} \mid \mathbf{y} \in \{-1, +1\}^k\}$$

L_1 to L_∞

for each vector $\mathbf{x} \in \mathbb{R}^k$

consider all possible vectors $\mathbf{y} \in \{-1, +1\}^k$

2^k in total

define $f(\mathbf{x}) = \langle \mathbf{x} \cdot \mathbf{y}_1, \ldots, \mathbf{x} \cdot \mathbf{y}_{2^k} \rangle$

$$||f(u) - f(v)||_\infty = ||f(u-v)||_\infty =$$

$$= \max_{\mathbf{y}} \{(\mathbf{u} - \mathbf{v}) \cdot \mathbf{y}\} = ||u - v||_1$$

distance preserving!

L_1 to L_∞ application

find the furthest pair of points in L_1

running time? $\mathcal{O}(n^2 k)$

embed points in L_∞

dimension? 2^k

running time? $\mathcal{O}(n 2^k)$

worth if $k << n$ in particular if $k = o(\log n)$

other interesting embeddings

1. consider a metric space (X,d) where X has n objects

   then: (X,d) can be embedded isometrically to $(\mathbb{R}^{n-1}, L_\infty)$

2. consider a tree metric (X,d) with X has n objects

   tree metric means that the objects of X can be represented as nodes of a tree, and distances between pairs of objects are equal to the tree path between the corresponding nodes (so, obviously it is a metric space)

   then: (X,d) can be embedded isometrically to $(\mathbb{R}^{\log n}, L_\infty)$

proof of these claims is left as exercise high dimensional data

optional reading

Johnson-Lindestrauss lemma

An Elementary Proof of a Theorem of Johnson and Lindenstrauss, Dasgupta and Gupta

Database-friendly random projections: Johnson-Lindenstrauss with binary coins, Achlioptas

properties of high-dimensional data

Foundations of Data Science, Avrim Blum, John Hopcroft, and Ravindran Kannan, chapter 2

book available online

real-world data are high dimensional

a document represented as a set of terms

(dimensionality = number of distinct terms)

a user represented as a set of movies she has watched

(dimensionality = number of movies)

a movie represented as a set of users who have watched it

(dimensionality = number of users)

i.e., dimensionality is hundred thousands or millions

how do high-dimensional data look like?

fact : the volume of a high-dimensional object is concentrated on its surface

proof sketch

consider object A in R^d, of volume $vol(A)$

shrink A along all dimensions, and obtain object $(1-e)A$

volume of new object is

$vol((1-e)A) = (1-e)^d \, vol(A)$

as d grows, the ratio $vol((1-e)A) / vol(A)$ goes to 0

so almost all of the volume is out of $(1-e)A$

geometry of a high-dimensional unit ball

unit d-dimensional ball : vectors in R^d having norm ≤ 1

properties of the unit ball :

most vectors lie on the surface of the ball

(most vectors have norm almost = 1)

most vectors lie on the equator

(they are orthogonal to the vector that shows north pole)

most pairs of vectors are almost orthogonal

most pairs of vectors have almost the same distance

in previous statements we can replace most with random (why?)

how to draw a random vector in the d-dimensional unit ball

consider $d=1$

pick a point uniformly in $[-1,1]$

what about large d ?

attempt 1 : pick each coordinate uniformly in $[-1,1]$

attempt 1 fails ; vectors are concentrated on the corners

attempt 2 : pick each coordinate from a Gaussian with mean 0 and std.dev 1, and then normalize

attempt 2 succeeds

how do I prove the previous claims?

proving properties of high dimensional data using probabilistic statements

i.e., properties of unit ball

apply concentration bounds

(tail inequalities, Chernoff bound, law of large numbers)

see next lectures

what do the previous properties imply for real-world data?

one possible implication:

real-world high-dimensional data are meaningless / boring!

all data points are near-orthogonal and equal-distant

well, not quite so !

real-world data are not random

hidden structure and dependencies make data interesting

data often lie in lower-dimensional manifolds within a high dimensional space

real-world high-dimensional data

yet, some part of the data may be random, or close to random

some behavior expected for random data may happen in real-world data

the curse of dimensionality

the curse of dimensionality

1. spurious patterns may appear
2. exponential increase of volume with number of dimensions

   need a lot more data to "fill up" the space

3. the efficiency of many algorithms degrades rapidly as the dimensionality increases

   distance distributions become more concentrated and pruning strategies fail

   index structures fail as the dimensionality of the data increases

4. data in large dimensions is difficult to visualize

dimensionality reduction

(embedding to lower dimensionality)

the curse of dimensionality

the efficiency of many algorithms depends on the number of dimensions d

distance / similarity computations are at least linear to the number of dimensions

index structures fail as the dimensionality of the data increases

data in large dimensions is difficult to visualize

what if we were able to...

...reduce the dimensionality of the data,

while maintaining the meaningfulness of the data ?

dimensionality reduction

consider dataset X consisting of n points in a d-dimensional space

data point x in X is a vector in R^d

data can be seen as an n x d matrix

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{nd} \end{pmatrix}$$

dimensionality-reduction methods :

- dimension selection: choose a subset of the existing dimensions
- dimension composition: create new dimensions by combining existing ones

dimensionality reduction

dimensionality-reduction methods :

dimension selection: choose a subset of the existing dimensions

dimension composition: create new dimensions by combining existing ones

both methodologies map each vector x in R^d to a vector y in R^k

mapping:

$A : R^d \longrightarrow R^k$

for the idea to be useful we want: $k \ll d$

linear dimensionality reduction

dimensionality-reduction mapping:

$A : R^d \longrightarrow R^k$

assume that A is a linear mapping

it can be seen as a matrix ($d \times k$)

$y = x A$

so

$Y = X A$

objective : Y should be as close as possible to X

closeness: pairwise distances

Johnson-Lindenstrauss lemma:

consider dataset X of n points in R^d, and $\varepsilon>0$

then there exists $k=O(\varepsilon^{-2}\log n)$ and a linear mapping $A : R^d \rightarrow R^k$, such that for all x and z in X

$$(1-\varepsilon)\ ||x-z||^2 \leq (d/k)\ ||xA-zA||^2 \leq (1+\varepsilon)\ ||x-z||^2$$

what is the intuitive interpretation of this statement?

Johnson-Lindenstrauss lemma

each vector x in X is projected onto a k-dimensional vector

$y = xA$

dimension of the projected space is $k=O(\varepsilon^{-2}\log n)$

sq. distance $||x-z||^2$ is approximated by $(d/k)||xA-zA||^2$

idea behind the proof:

expected sq. norm of a projection of a unit vector onto a random subspace is k/d

the probability that it deviates from its expectation is very small

then, show that the statement holds for all pairs of distances

the random projections

random projections are represented by a linear transformation matrix A

each vector x in X is projected onto a k-dimensional vector

$$y = x A$$

what is the matrix A?

the random projections

JL mapping: the elements A(i,j) of A can be drawn from the normal distribution N(0,1)

resulting rows of A define random directions in R^d

another way to define A is ([Achlioptas 2003])

$$A(i,j) = \begin{cases} 1 & \text{with prob. } 1/6 \\ 0 & \text{with prob. } 2/3 \\ -1 & \text{with prob. } 1/6 \end{cases}$$

why is this useful?

all zero-mean, unit-variance distributions for A(i,j) would give a mapping that satisfies the Johnson-Lindenstrauss lemma

do you know of any other method to embed the data in low dimensions?

comparison with principal component analysis (PCA)

	random projections	PCA
aimed to	preserve all pairwise distances	explain the variance of the data
dimensionality of projected space	O(logn) (does not work for d=2 or 3)	typically used with few dimensions, e.g., 2 or 3
typically used for	preprocess the data so as to use for other computations (e.g., similarity search)	visualization
computation	linear, very efficient	requires singular-value decomposition, expensive