Introduction to Data Science

Lecture 3: Similarity and distance measures

Data Science and Engineering Department
Faculty of Informatics
ELTE University

Zakarya Farou zakaryafarou@inf.elte.hu

What is distance?

Let $\mathcal{S}$ be a space of data objects. A distance function has the type:

$$d : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^{+} \cup \{0\}$$

Intuitively: Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{S}$ be objects.

- If $d(\mathbf{x}, \mathbf{y})$ small, $\mathbf{x}$ and $\mathbf{y}$ are close or similar.
- If $d(\mathbf{x}, \mathbf{y}) < d(\mathbf{x}, \mathbf{z})$, $\mathbf{x}$ is closer/more similar to $\mathbf{y}$ than $\mathbf{z}$

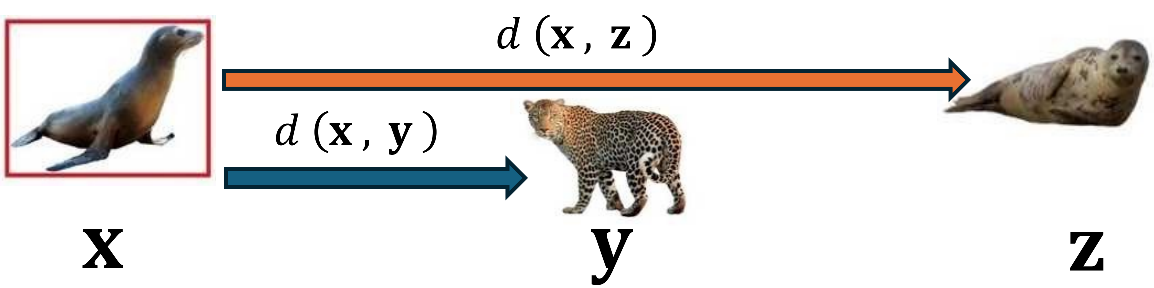


Image source Lin 2018. https://www.linkedin.com/pulse/ cosine-similarity-classification-michael-lin

Similarity vs Distance

Similarity function $s : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$

- $s(\mathbf{x}, \mathbf{y})$ large when $\mathbf{x}$ and $\mathbf{y}$ similar $\Rightarrow$ small $d(\mathbf{x}, \mathbf{y})$
- often $s : \mathcal{S} \times \mathcal{S} \to [0, 1]$
- $\Rightarrow$ possible to induce distance $d_s = 1 - s$
- if $s : \mathcal{S} \times \mathcal{S} \to [0, 1]$, possible to induce similarity

$$s_d = 1 - d$$

- if not, then e.g.,

$$s_d = 1 - {}^{d}/_{d_{max}} \quad \text{OR} \quad s_d = 1 - {}^{1}/_{1+d}$$

Metric: distance d that satisfies 4 properties

1. $d(x, y) \geq 0$ (non-negativity or separation)
2. $d(x, y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

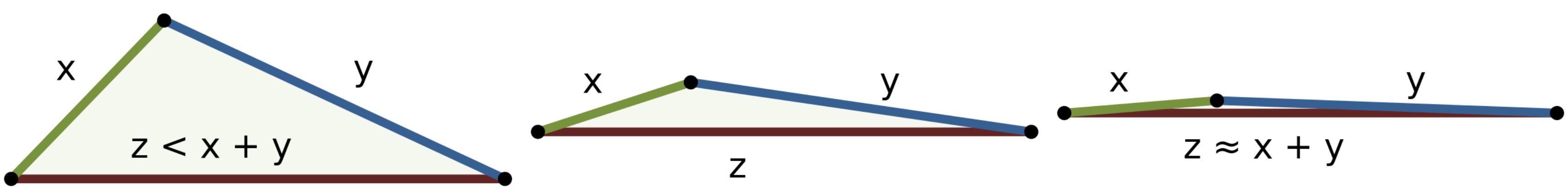


Image source
https://en.wikipedia.org/wiki/Triangle_inequality

Metric space

Metric space $(\mathcal{S}, d)$ = data space equipped with a metric

- 3-D Euclidean space or any normed vector space
- no need to be a vector space! (e.g., space of strings + suitable metric)

Why they are so nice?

- Many tasks can be performed more efficiently!
- Especially similarity search (find nearest neighbors, closest cluster centers, similar documents, etc.)

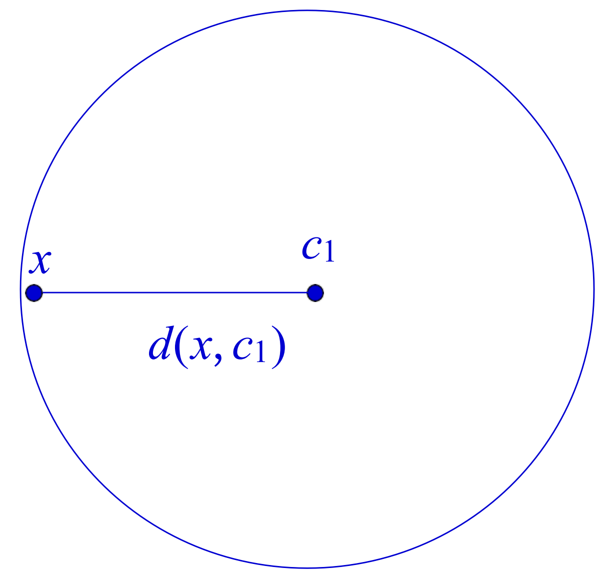
Example how $\triangle$ inequality can speed up things

Problem: given cluster centroids $c_1, \ldots, c_k$, find the closest c_i for all data points x. (d is a metric)

1. Naive solution: calculate all $d(x, c_i)$. (nk calculations)
2. **Pruning trick:** given $d(c_i, c_j)$ for all i, j and $d(x, c_1)$ to the currently closest c_1.

   Test: If $d(c_1, c_2) > 2d(x, c_1)$, then c_2 cannot be closer to x!

   If c_2 was closest to c_1, then c_1 is closest to x.



c_2 cannot be inside the circle

since $d(c_1, c_2) > 2d(x, c_1)$

Example how $\triangle$ inequality can speed up things

More pruning by utilizing upper and lower bounds of distances!

Further reading:

- Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (pp. 147-153).

- Hamerly, G. (2010, April). Making k-means even faster. In Proceedings of the 2010 *SIAM international conference on data mining* (pp. 130-140). Society for Industrial and Applied Mathematics.

Do you know distance or similarity measures for these data types?

- Numerical
- Categorical
- Mixed
- Binary
- Strings
- Text
- Graphs
- Time series

Multidimensional numerical: L_{p}-norm

Objects are x $= (x_1, \ldots, x_k)$ and y $= (y_1, \ldots, y_k)$, $x_i, y_i \in \mathbb{R}$

Most common measure L_p-norm or Minkowski distance:

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

- different variants by setting p
- e.g., Euclidean distance: $L_2(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{k} |x_i - y_i|^2 \right)^{\frac{1}{2}}$
- metric, if $p \geq 1$

Manhattan (city block) distance L_1

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

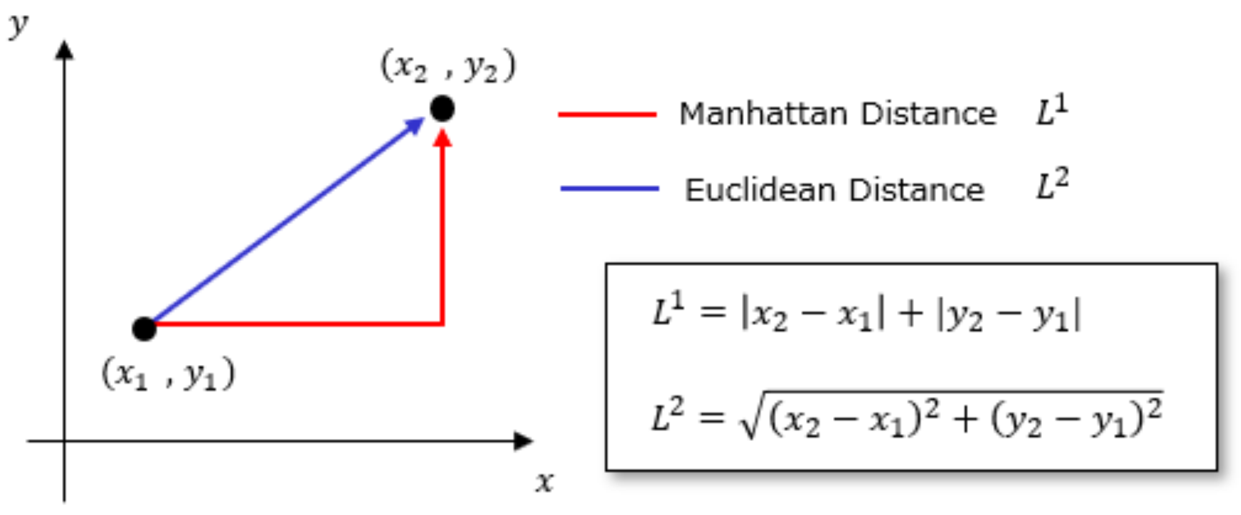
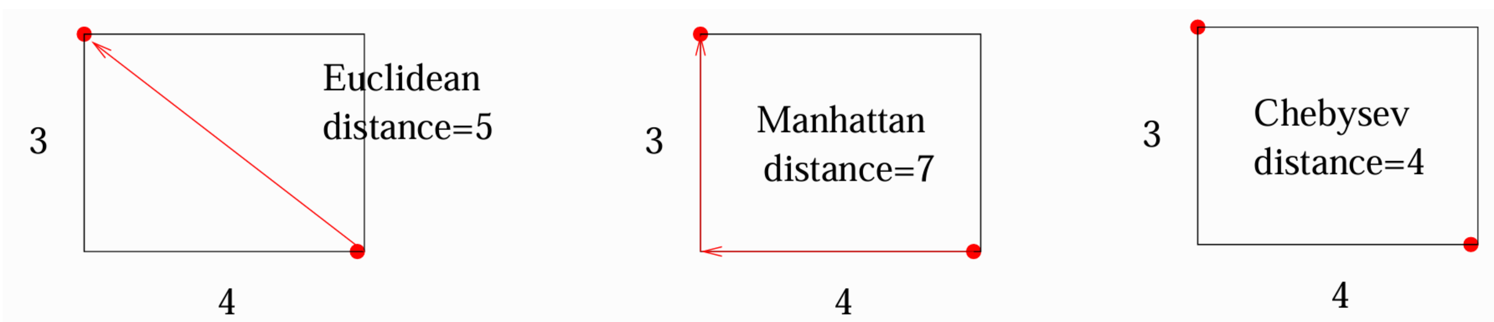


image source

https://taketake2.com/N102_en.html

L_p-norms

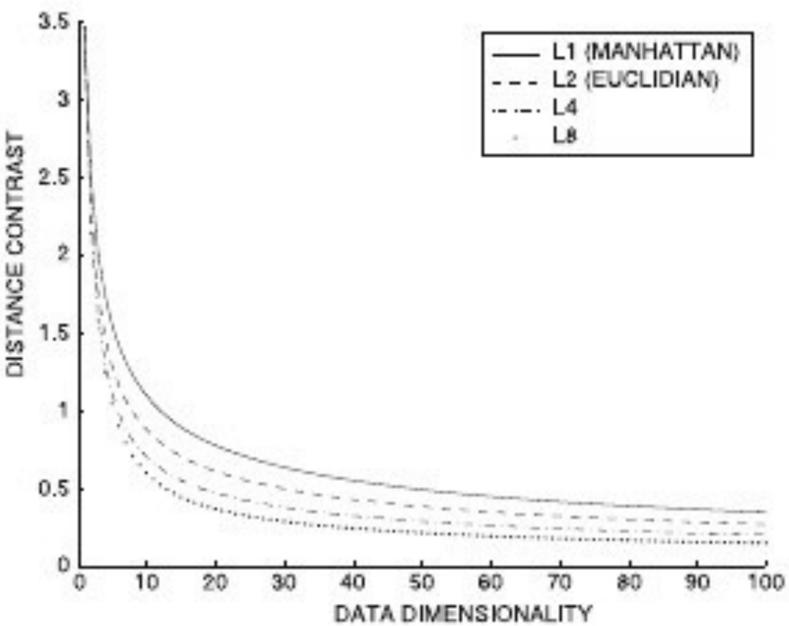
- $p = 1$: Manhattan distance $L_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{k} |x_i - y_i|$
- $p = 2$: Euclidean distance $L_2(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{k} |x_i - y_i|^2\right)^{1/2}$
- $p \to \infty$: Chebyshev distance $L_\infty(\mathrm{x}, \mathrm{y}) = \max_i |x_i - y_i|$



More details:
https://www.youtube.com/watch?v=NKuLYRui-NU&ab_channel=Dr.WillWood

L_p-norms do not work well in high dimensions

Curse of dimensionality: Contrasts $\frac{d_{max}-d_{min}}{d_{avg}}$ between largest and the smallest distances disappear.
Behavior in random data:



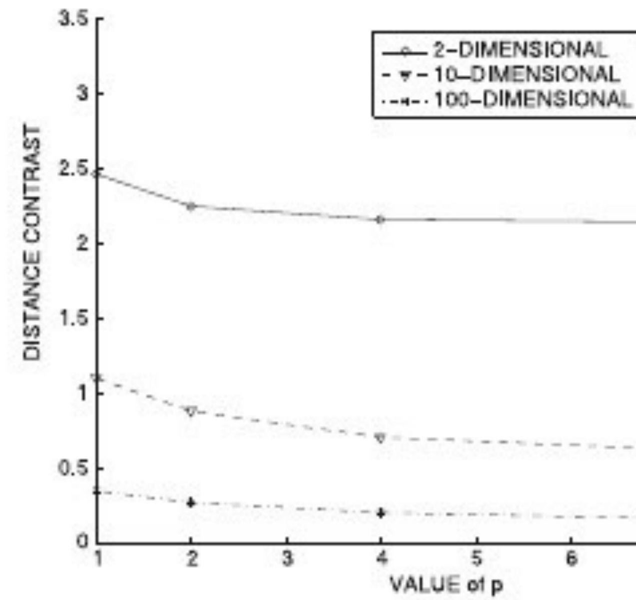


Image source: Aggarwal 2015

L_p-norms do not work well in high dimensions

- irrelevant features tend to dominate $L_2, \ldots, L_\infty$
- Consider $L_\infty(\mathsf{x}, \mathsf{y})$ when x and y have similar values in 999 dimensions but dissimilar in 1 irrelevant attribute!

$$\Rightarrow$$

- generalized Minkowski distance give weights a_i reflecting importance: $L_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{k} a_i |x_i - y_i|^p\right)^{\frac{1}{p}}$

- fractional L_p quasinorms set $p \in]0, 1[$ (not metrics)
- match-based similarity

Multidimensional numerical: Match-based similarity

Observations:

1. Features may be only **locally relevant** (e.g., blood glucose for diabetic patients but not for epileptic).
2. In large dimensions, two objects are unlikely to have similar values, unless the feature is relevant.

$\Rightarrow$ **emphasize dimensions where objects are close/similar!**

(Euclidean and pals do the opposite)

Cosine similarity and distance

Cosine similarity:

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

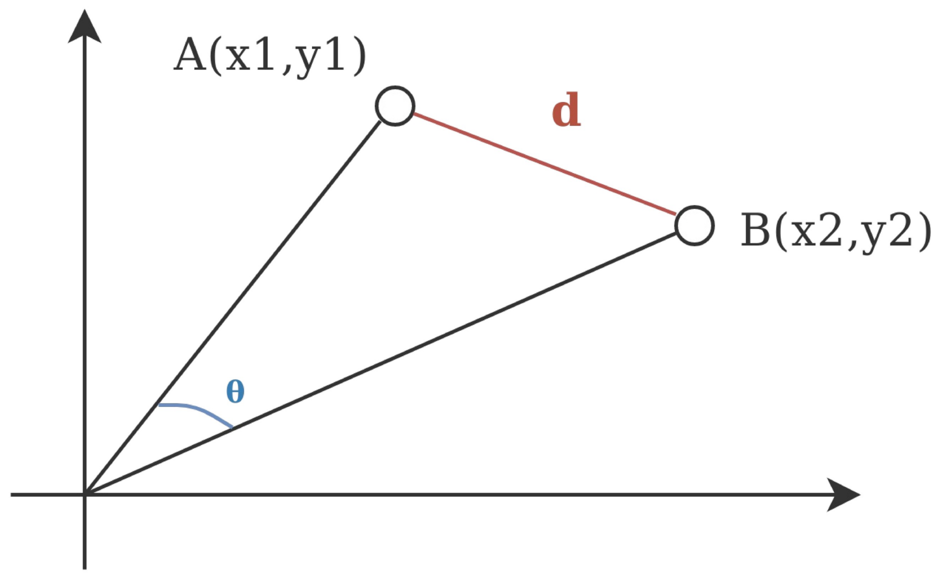
- Suitable for numerical (continuous or integers) and binary data.
- in $[-1, 1]$, most similar if $cos(\mathrm{x}, \mathrm{y}) = 1$
- popular for text documents (their numerical presentation)

Cosine distance: $1 - cos(\mathrm{x}, \mathrm{y})$

- $[0, 1]$, if all vector elements non-negative ($x_i \geq 0$)

Cosine similarity and distance

Relationship to Euclidean distance L_2:
if vectors are normalized (length 1),
$L_2{}^2(\mathsf{x}, \mathsf{y}) = 2(1 - cos(\mathsf{x}, \mathsf{y}))$

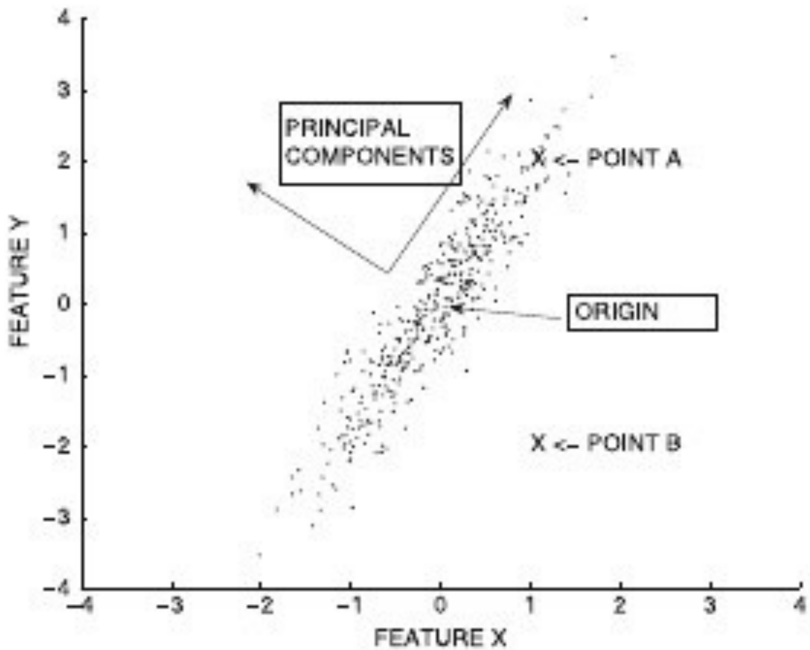


When to use?
When you do **not** want to consider vector magnitudes, only their directions (for example, when using word frequencies to represent a document).

image source
https://cmry.github.io/notes/euclidean-v-cosine

Should the distance reflect data distribution?

Should A and B be equally distant from the origin?



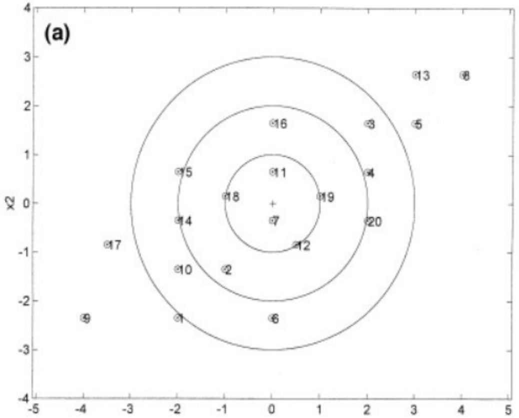
high variance direction $\Rightarrow$ more likely to be distant $\Rightarrow$
could consider A closer than B $\Rightarrow$
Mahalanobis distance

$$Maha(\mathrm{x},\mathrm{y}) = \sqrt{(\mathrm{x}-\mathrm{y})\Sigma^{-1}(\mathrm{x}-\mathrm{y})^T}$$

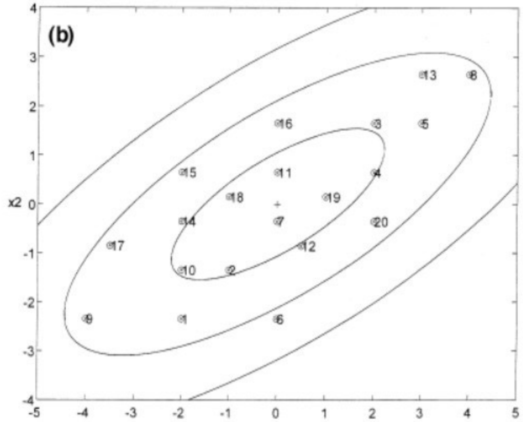
(Σ = covariance matrix)

Mahalanobis distance $\text{Maha}(x, y) = \sqrt{(x - y)\Sigma^{-1}(x - y)^T}$

The Mahalanobis distance considers how spread apart points are in the dataset (i.e. the variance of the dataset) to weigh the absolute distance from one point to another.



Points on each circle have the same Euclidean distance to the origin.



Points on each ellipsis have the same Mahalanobis distance to the origin.

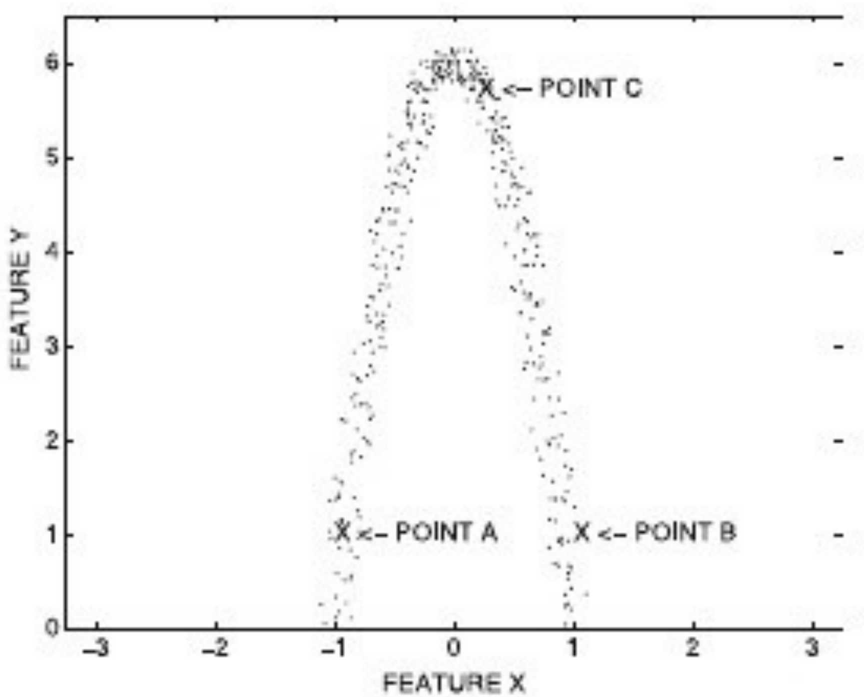
When to use?

When your points are correlated (Equiv to Euclidean distance when the points are uncorrelated)
When the covariance of the points is very far from spherical.

image source

https://queirozf.com/entries/similarity-measures-and-distances-basic-reference-for-data-science-practitioners

Should the distance reflect data distribution?



Which pair of points are closest to one another?

Categorical data: similarity

Generic function:

$$sim(x, y) = \sum_{i=1}^{k} w_i \, s(x_i, y_i)$$

- ✦ Typically weight $w_i = \frac{1}{k} (k = number\ of\ features)$
- ✦ Many choices for $\boldsymbol{s}$, e.g., in overlap similarity $\boldsymbol{s}$ is:

$$s(x_i, y_i) = \begin{cases} 1, if \ \ x_i = y_i \\ 0, otherwise \end{cases}$$

Overlap similarity= fraction of dimensions where $\boldsymbol{x}$ and $\boldsymbol{y}$ have equal value.

Categorical data: similarity

Or take into account frequency of value:

$$p_i(x_i) = \frac{fr(A_i = x_i)}{n} = \text{fraction of records having } A_i = x_i$$

Goodall measure (its one variant):

$$s(x_i, y_i) = \begin{cases} 1 - p_i^2(x_i) & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases}$$

Further reading Boriah et al. (2008): Similarity measures for categorical data: A comparative evaluation.

Similarity in mixed data (without transformations)

Give weights to numerical and categorical components:

$$sim(\mathrm{x}, \mathrm{y}) = \lambda \cdot NumSim + (1 - \lambda) \cdot CatSim$$

- How do you choose λ?
- e.g., a fraction of numerical features in data
- $NumSim$ and $CatSim$ often in different scales ⇒
  - calculate standard deviations (σ_N and σ_C) in similarity values.

$$sim(\mathrm{x}, \mathrm{y}) = \lambda \cdot NumSim/\sigma_N + (1 - \lambda) \cdot CatSim/\sigma_C$$

Binary data: distance and similarity

Data points x and y are bit strings (length k)

Hamming distance = L_1 norm for binary data

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

=number of positions where bits differ

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

Hamming distance = 5

Image source: CS-E4600 fall 2019 slides

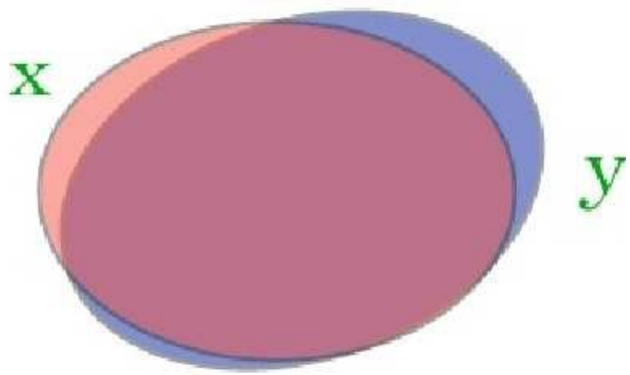
Set data can be presented as binary

	low fat milk	apple juice	white bread	edam cheese		
basket1	0	0	1	1	0	
basket2	1	1	0	0	0	
basket3	0	1	0	1	0	...
basket4	1	0	1	0	1	
basket5			.			

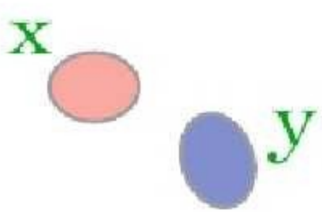
- transactions (like market baskets)
- occurrence of words in documents
- over-expressed or under- expressed genes in sam- ples

Set data often very sparse (= most values are 0s) ⇒ number of common elements more important

Hamming distance for transaction data?



1. Two sets with 1000 items and 995 common



2. Two sets with 5 items, but none common

Both have Hamming distance = 10

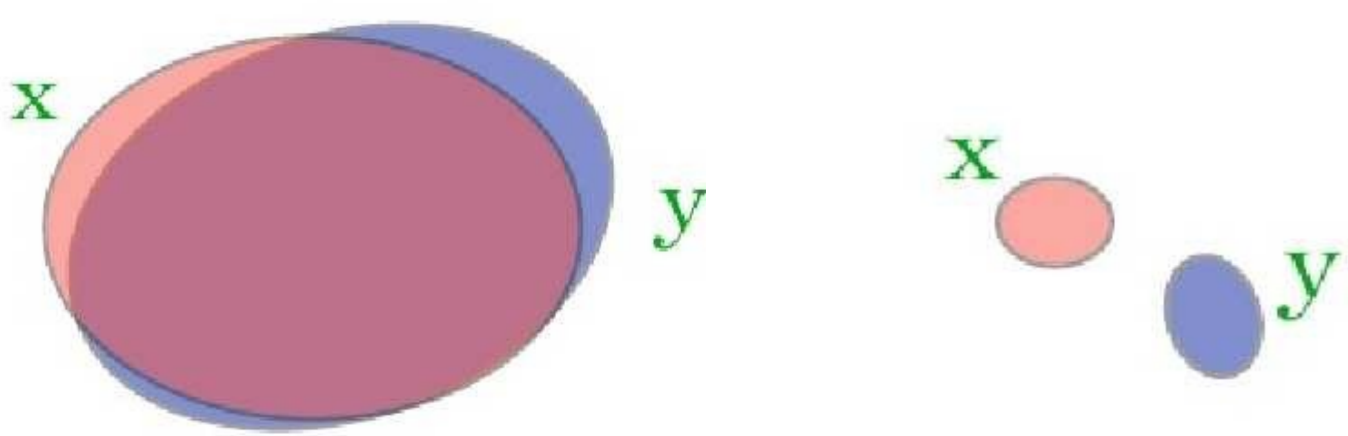
image source: CS-E4600 fall 2019 slides

Jaccard coefficient for set similarity

Given sets x and y

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

- treats 0s and 1s differently
- Previous example: in case 1 $J = 0.99$, in case 2 $J = 0$



String data: distance

Given strings x and y of the same length.
Modification of the Hamming distance

- add 1 for all positions that are different

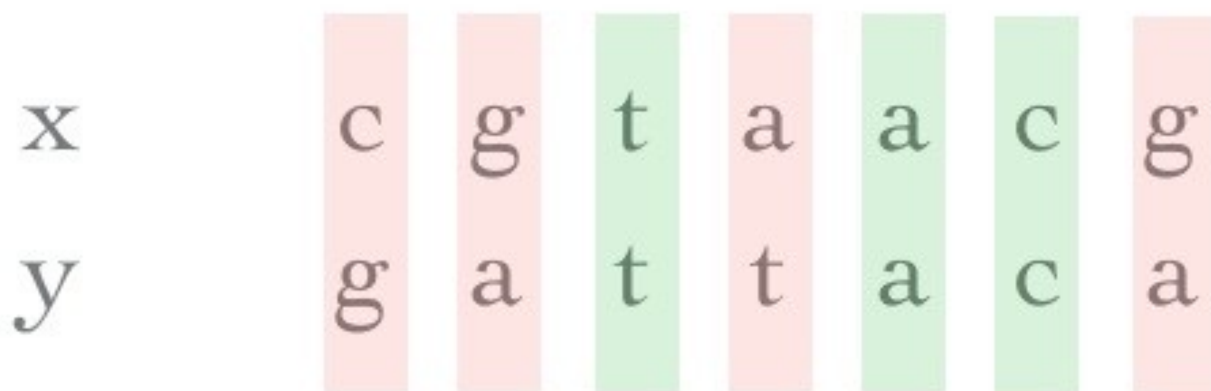
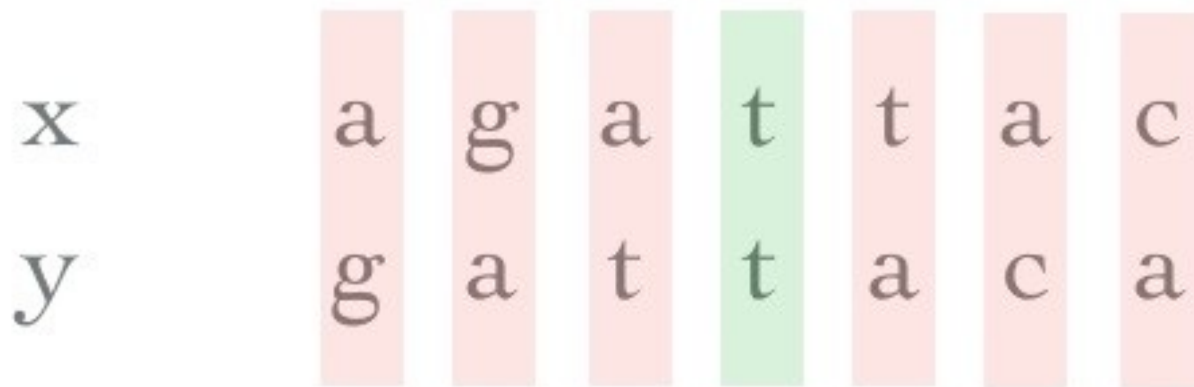


image source: CS-E4600 fall 2019 slides

Is Hamming distance good for strings?

- Strings must have equal length
- Punishes a lot for small typos:



String Hamming distance = 6

image source: CS-E4600 fall 2019 slides

String edit distance

Given 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=minimum cost of such operations
- Levensthein distance=minimum number of such operations (unit costs)
- edit operations can have different costs w_{ins}, w_{del}, w_{sub}
- metric, if positive costs and each operation has an inverse operation with the same cost

String edit distance examples

Levensteihn(kitten, sitting)=3:

1. kitten → sitten (substitute "s" for "k")
2. sitten → sittin (substitute "i" for "e")
3. sittin → sitting (insert "g" at the end)

---

image source: CS-E4600 fall 2019 slides

Text data: similarity between documents

Let's present text documents as document-term matrices.

- x and y are m -dimensional vectors, where m = lexicon size
- x_i = frequency of term i in the document x
- then take cosine similarity:

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

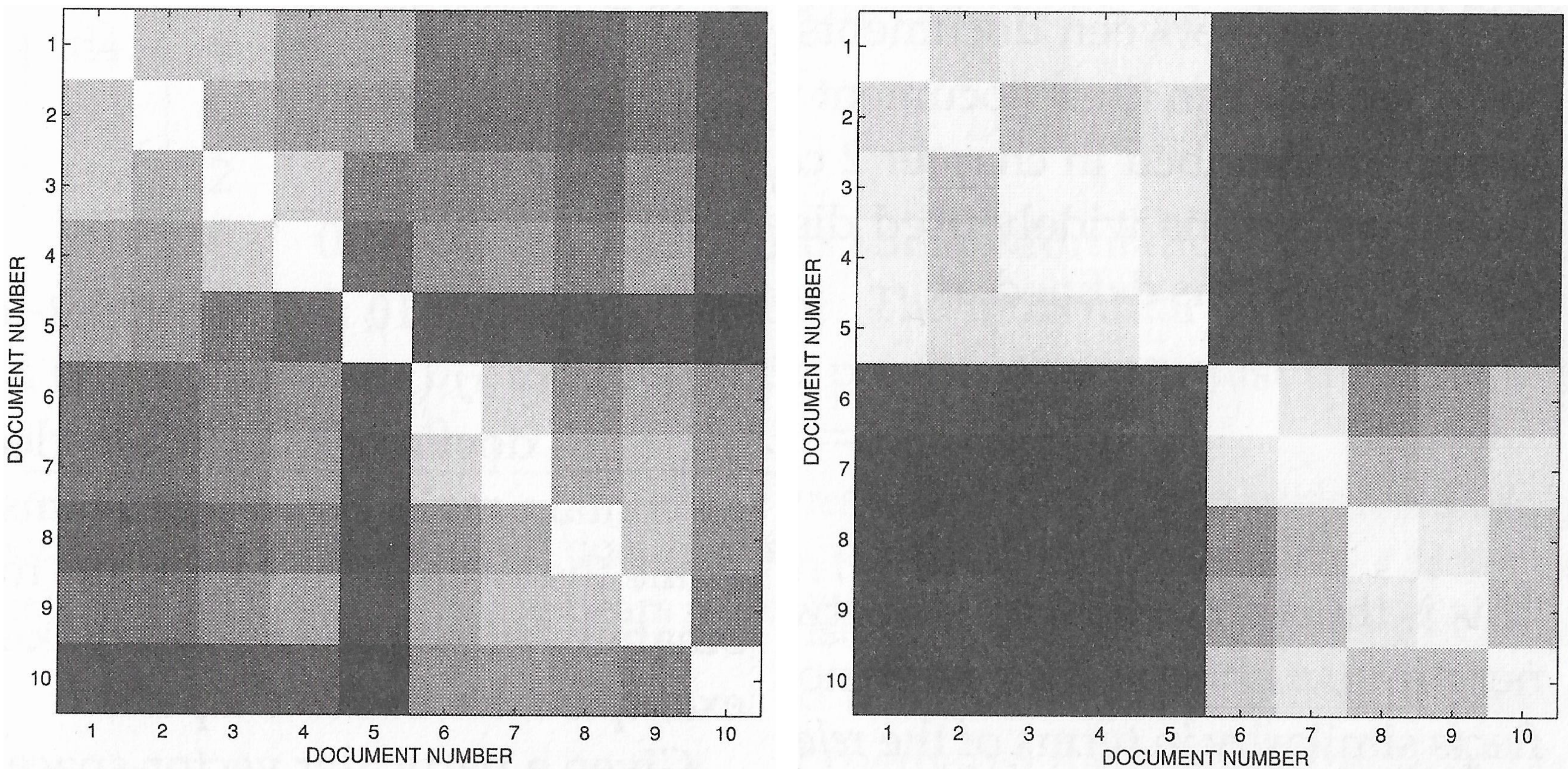
- Note: Boolean model also possible, where $x_i = 1$, if ith term occurs in the document
  - Jaccard coefficient or cosine similarity

Text data: Example (Hand et al. 2001)

	t1	t2	t3	t4	t5	t6
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	31	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

source: Hand, Mannila, Smyth: Principles of data mining, 2001

Text data: Example (Hand et al. 2001)



Left: Euclidean distance (bright=small distance), right: cosine similarity (bright=large similarity)

Text data: similarity between documents

- How to lessen the effects of overly common words and give more weight to rarer words? $\Rightarrow$ tf-idf presentation
- inverse document frequency $idf_i = \log \frac{n}{n_i}$ where n_i = number of documents where ith word occurs

- idf_i gives more weight to rarer words
- calculate tf-idf-values for all x_i and y_i:

$$tf\text{-}idf(x_i) = x_i \cdot \log \frac{n}{n_i}$$

- $\rightarrow$ cos similarity between transformed x and y
- Warning: many variants of tf-idf! Check always the equation

Text data: Example (Hand et al. 2001)

	t1	t2	t3	t4	t5	t6
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	31	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

2.53	14.56	4.60	0	0	2.07
3.37	6.93	2.55	0	1.07	0
1.26	11.09	2.55	0	0	0
0.63	4.85	1.02	0	0	0
4.53	21.48	10.21	0	1.07	0
0.21	0	0	12.47	2.50	11.09
0	0	0.51	22.18	4.28	0
0.31	0	0	15.24	1.42	1.38
0.10	0	0	23.56	9.63	17.33
0.63	0	0	11.78	1.42	15.94

Original document-frequency matrix and corresponding *tf-idf* matrix.

Other data types

See the textbook!

- time series: Ch 3.4
- graphs: Ch 3.5

Textbook: Aggarwal 2015

Summary

- Choose distance and similarity measures carefully!
- Curse of dimensionality → for multidimensional data consider L_p with small p, cosine or match-based similarity
- If the distribution is very heterogenous, it is beneficial to adjust to local variations in distances (but costs!)
- Metric distances can speed similarity search, but non-metrics may perform better in high dimensions