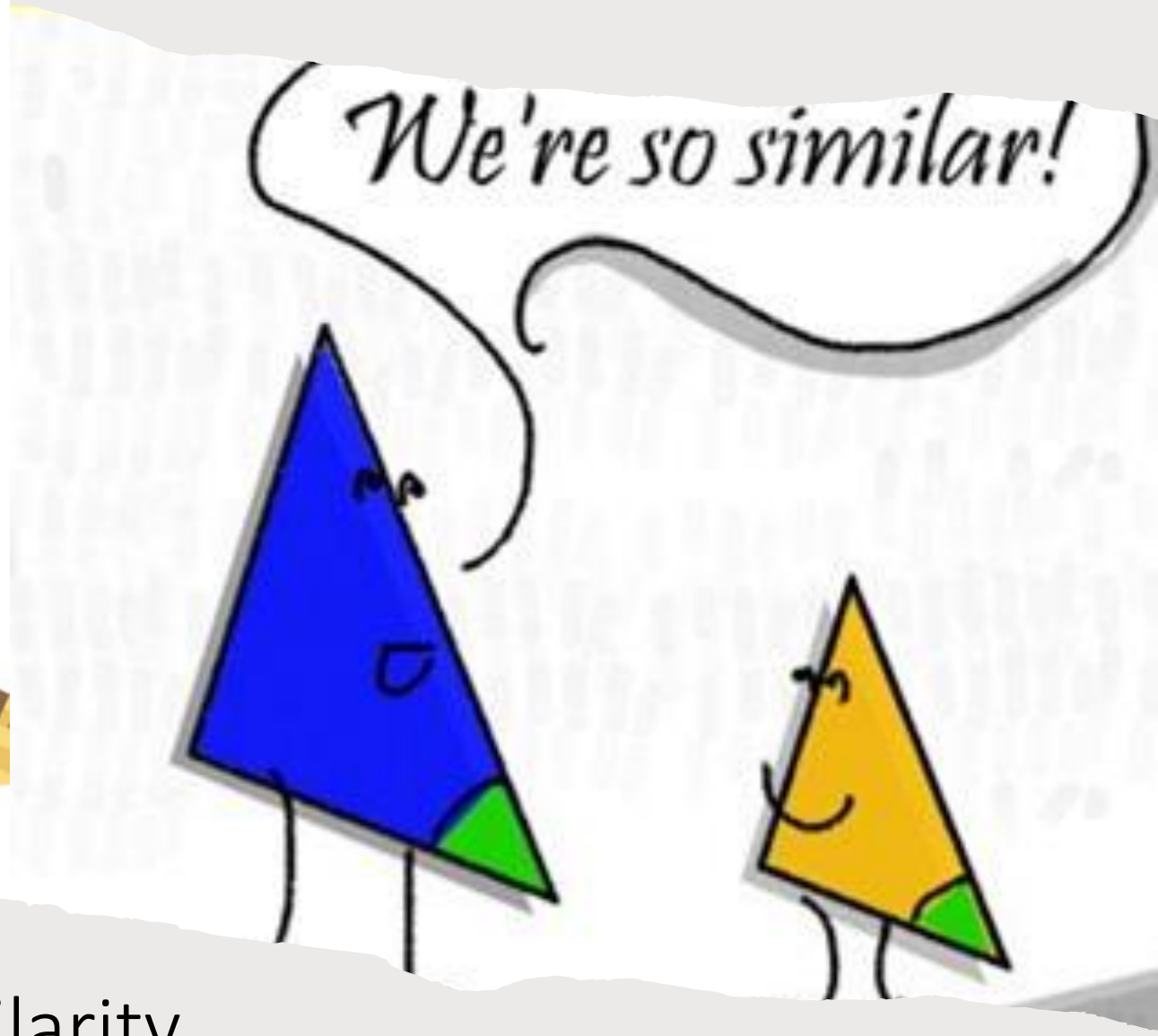


Data Similarity and Dissimilarity

Prepared By

-Anooja Joy

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.
Webster's Dictionary

Similarity is hard to define, but…
"*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

Similarity and Dissimilarity

- **Similarity is the quantity that reflects strength of relationship between 2 objects or features.**
- **Similarity is difficult to measure.**
- Dissimilarity measures the discrepancy between 2 objects based on several features. It is a measure of dissimilarity.
- Distance measures dissimilarity.
- When similarity is 1 dissimilarity is 0 and similarity is 0 dissimilarity is 1.
- Sim= 1-Disim

Distance Met ric Properties

- A **metric** is a **distance function ƒ** defined that have the following properties.

1. **nonnegativity:** ƒ (**x, y**) ≥ 0; Distance is non-negative

2. **reflexivity:** ƒ (**x, y**) = 0 ⇔ **x** = **y**;Distance an object to itself is 0.

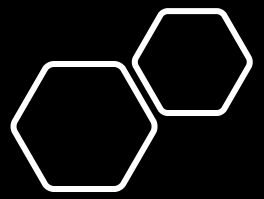
3. **commutativity/ symmetry:** ƒ (**x, y**) = ƒ (**y, x**); Distance is a symmetric function.

4. **triangle inequality:** ƒ (**x, y**) ≤ ƒ(**x, z**) + ƒ (**y, z**), where **x**, **y**, and **z** are arbitrary data points.Going directly fro object x to y in space is no more than taking detour over any object z.

A distance that satisfies these properties is a **metric**

Similarity measures for numeric data

- **Similarity measure** is the numerical measure of the degree to which of how alike two data objects are.
- A similarity coefficient indicates the strength of the relationship between two data points. The more the two data points resemble one another, the larger the similarity coefficient will be.
- It often fall between **0** (**no similarity**) and **1** (**complete similarity**)
- Similarity might be used to identify
  - duplicate data that may have differences due to typos.
  - equivalent instances from different data sets. E.g. names and/or addresses that are the same but have misspellings.
  - groups of data that are very close (clusters)



Similarity Measures Applications

- Clustering
- Outlier Analysis
- Nearest Neighbor Classification
- Recommendation engines
- Text related preprocessing techniques
- Different classification problems
- Email spam or ham classification problems

Dissimilarity Measure

- **Dissimilarity measure** is the numerical measure of how different two data objects are. It range from 0 (objects are alike) to ∞ (objects are different). Distance is used as a synonym for dissimilarity
- Dissimilarity might be used to identify
  - outliers
  - interesting exceptions, e.g. credit card fraud
  - boundaries to clusters
  - **Proximity Measures**
- It can be either similarity or dissimilarity.

Data matrix & Dissimilarity Matrix

Data matrix: A data matrix of n data points with ldimensions. This structure stores the n data objects in the form of a relational table, or n-by-l matrix (n objects×l attributes).Each row corresponds to an object

$$D = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1l} \\ x_{21} & x_{22} & \cdots & x_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nl} \end{pmatrix}$$

- **Dissimilarity matrix** is a triangular matrix of n data points that registers only the distance of dissimilarity. It stores a collection of proximities for a pair of n objects. d(i, j) is themeasured dissimilarity or “difference” between objects i and j.

$$\begin{bmatrix} d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

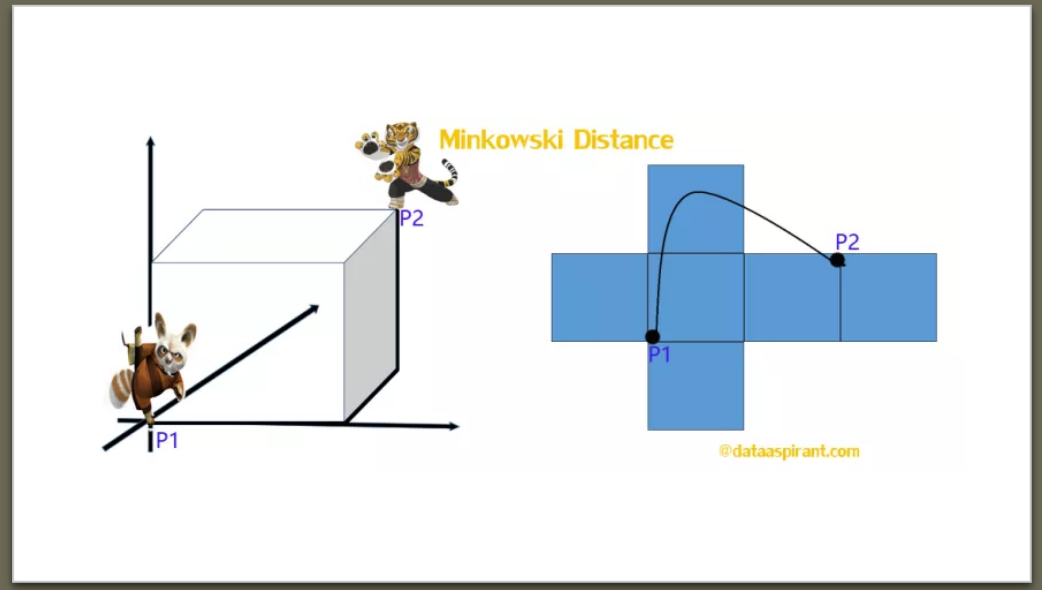
A data matrix is made up of two entities or “things”, namely rows (for objects) and columns (for attributes). Therefore, the data matrix is often called a two-mode matrix. The dissimilarity matrix contains one kind of entity (dissimilarities) and so is called a one-mode matrix.

Dissimilarity of Numeric Data

1. Minkowski distance
2. Euclidean distance
3. Manhattan distance
4. Supremum distance
5. Mahalanobis distance
6. Bhattacharyya distance

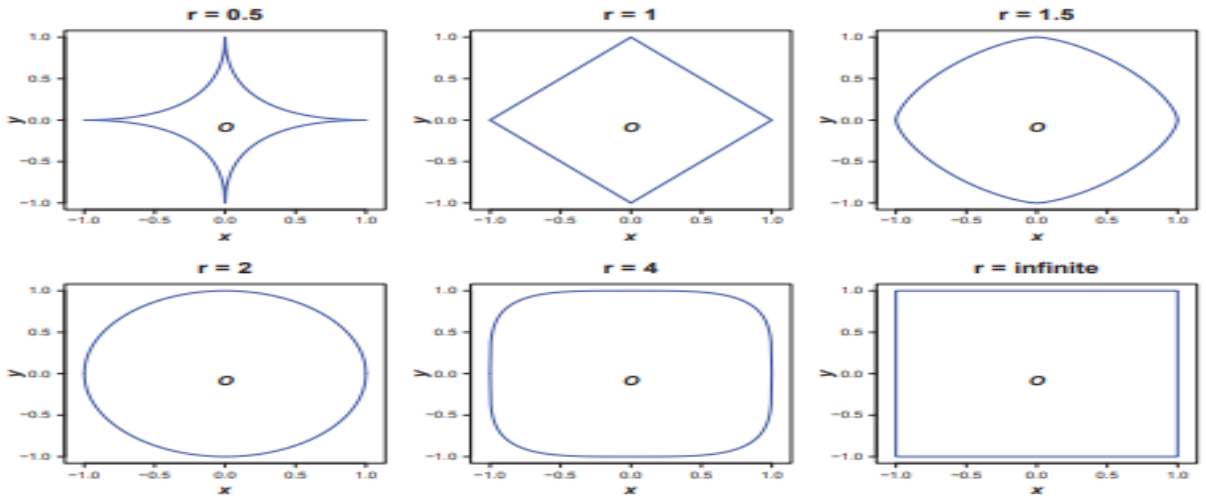
1. Minkowski Distance

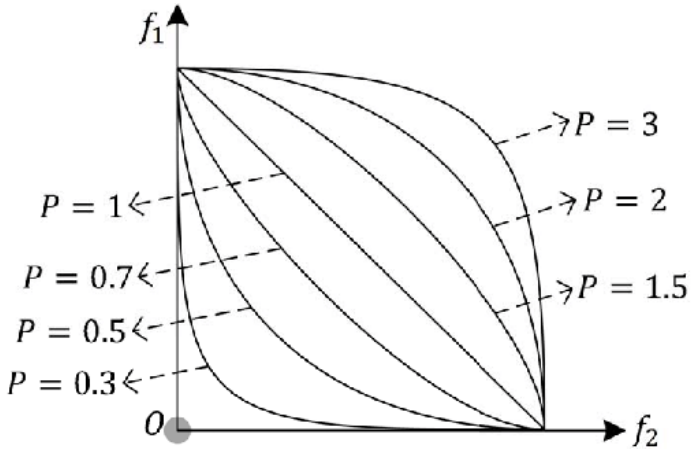
- Minkowski Distance is a generalization of Euclidean Distance and Manhattan distance in a normed vector space.
- Normed vector space. What is Normed vector space? A Normed vector space is a vector space on which a norm is defined. Suppose X is a vector space then a norm on X is a real valued function $||x||$ which satisfies below conditions -
  - **Zero Vector-** Zero vector will have zero length.
  - **Scalar Factor-** The direction of vector doesn't change when you multiply it with a positive number though its length will be changed.
  - **Triangle Inequality-** If distance is a norm then the calculated distance between two points will always be a straight line.
- Although *p* can be any real value, it is typically set to a value between 1 and 2. For values of *p* less than 1, the formula above does not define a valid distance metric since the triange inequality is not satisfied.



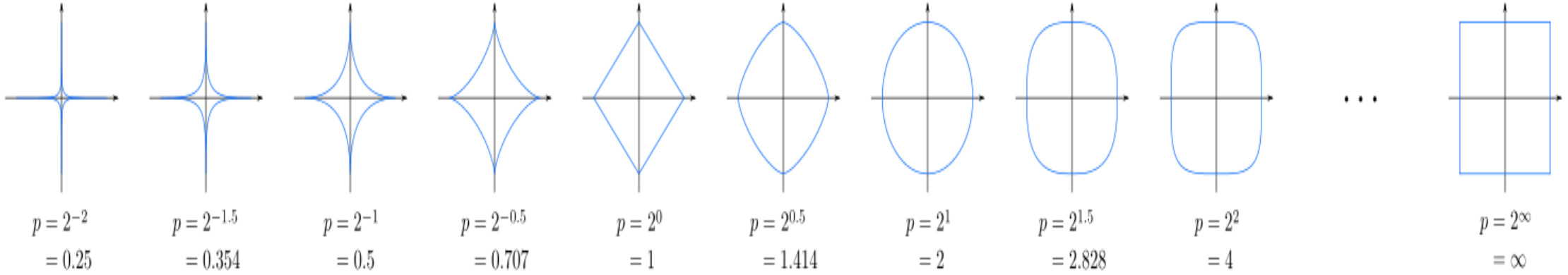
$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Minkowski distance





Example of different Minkowski distance contour curves when the value of P alters



- The following figure shows unit circles (the set of all points that are at the unit distance from the centre) with various values of p. All the points that are at a distance of 1 from the center, which is the definition of a circle in Euclidean distance (r = 2). Notice how that circle grows progressively until reaching the square form in the infinity (r → +∞). This is because when r increases, the influence of the highest component |xi − yi| r in equation increases notably compared to the other components in the distance computation

Synonyms of Minkowski

- Different names for the Minkowski distance or Minkowski metric arise from the order:
- **P= 1** is the **Manhattan distance**. Synonyms are **L1-Norm, Taxicab,** or **City-Block distance**. For two vectors of ranked ordinal variables, the Manhattan distance is sometimes called F**oot-ruler distance.**
- **P= 2** is the **Euclidean distance**. Synonyms are **L2-Norm or Ruler distance.** For two vectors of ranked ordinal variables, the Euclidean distance is sometimes called **Spear-man distance.**
- **P= ∞** is the **Chebyshev distance**. Synonyms are **Lmax-Norm or Chessboard distance.**

Example Minkowski Distance

Features	Coord1	Coord2	Coord3	Coord4	Coord5	Coord6
Object A	0	3	4	5		
Object B	7	6	3	-1		

Minkowski distance for order 3 is

$$d_{BA} = \left(|0-7|^3 + |3-6|^3 + |4-3|^3 + |5+1|^3\right)^{\frac{1}{3}}$$

$$= \sqrt[3]{343+27+1+216} = \sqrt[3]{587} = 8.373$$

Example Minkowski Distance

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

The Manhattan distance is obtained setting p=1 in the Minkowski distance

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

The Euclidean distance is obtained setting p=2 in the Minkowski distance

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

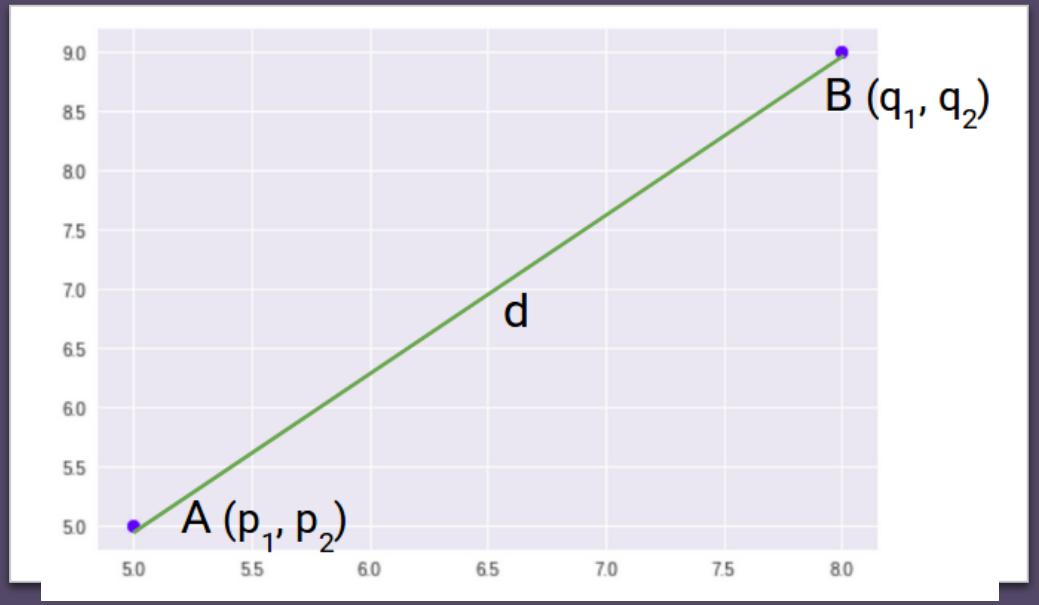
The Cheybshev distance is obtained setting p=inf in the Minkowski distance

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

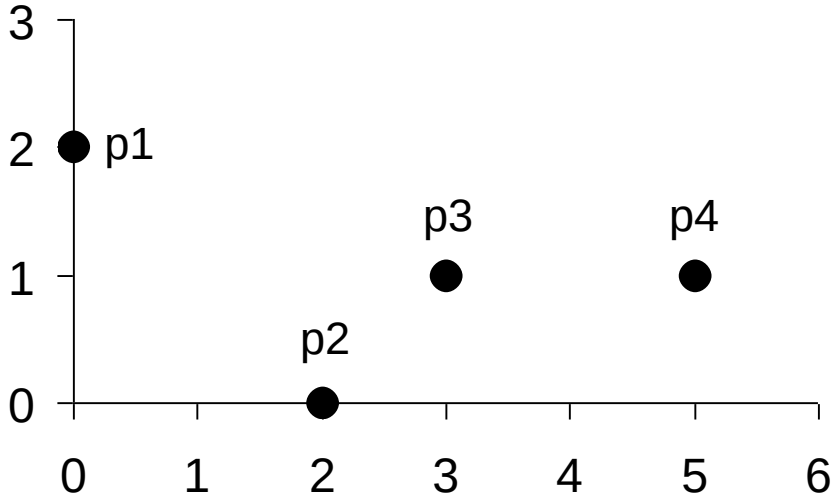
2. Euclidean distance

- The Euclidean distance between two points(x,y) in any dimension of space is the length of the path connecting them. The **Pythagorean theorem** gives this distance between two points.
- ***Euclidean Distance*** *represents the shortest distance between two points*
- When data is **dense or continuous**, Euclidean distance is the best proximity measure.
- where *n* is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) or data objects **x** and **y**.
- Eg: Calculate Euclidean distance between points [0,3,4,5],[7,6,3,-1]



$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

Euclidean Distance Examples



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$\sqrt{(2-0)^2+ (0-2)^2}= \sqrt{8}$

$\sqrt{(3-0)^2+ (1-2)^2}= \sqrt{10}$

$\sqrt{(5-0)^2+ (1-2)^2}= \sqrt{26}$

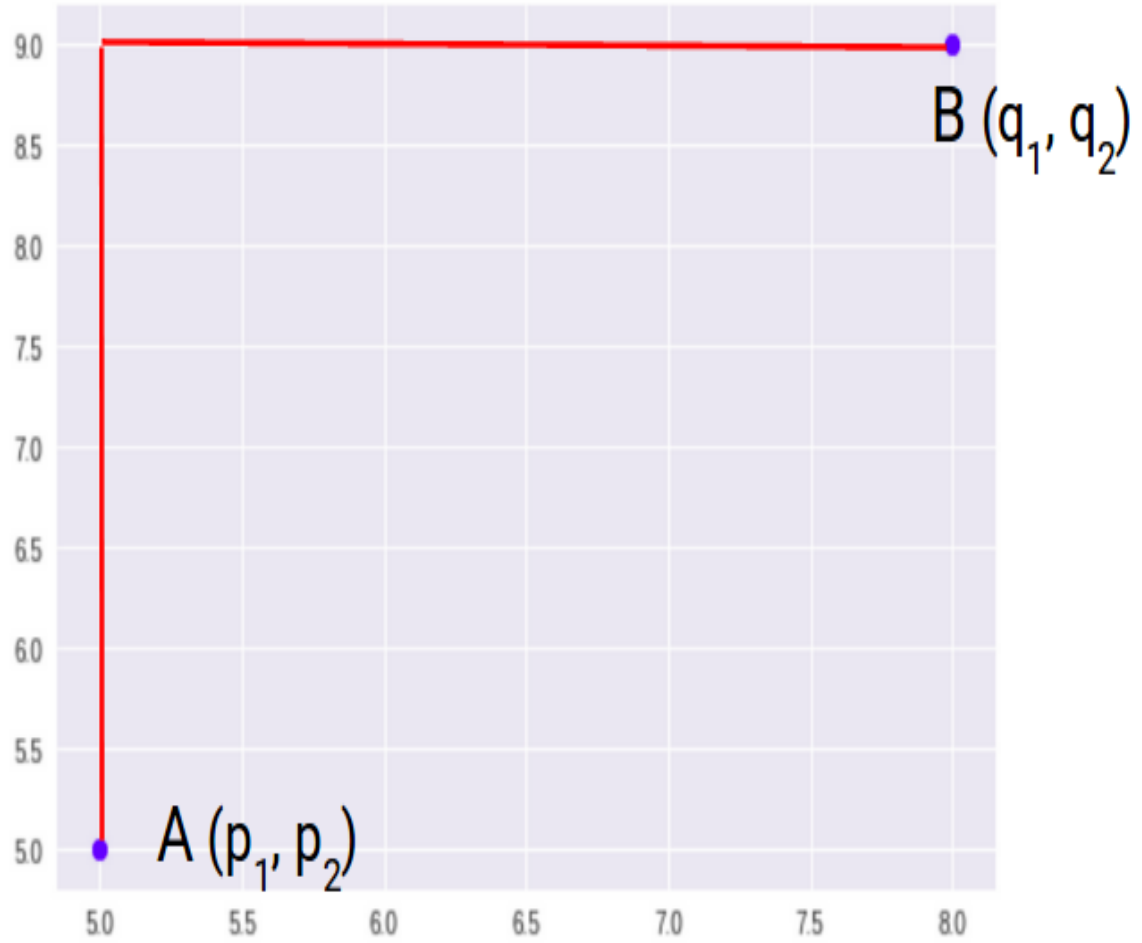
	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

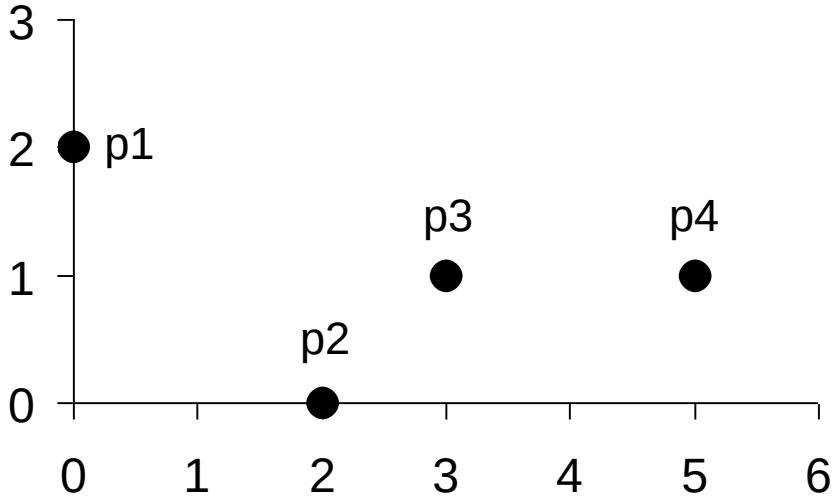
3. Manhattan distance

- Manhattan distance is a metric in which the distance between two points is calculated as the **sum of the absolute differences of their Cartesian coordinates.**
- Manhattan distance also known as **city block distance**, is **the distance in blocks between any 2 points.**
- Manhattan distance is usually preferred over the more common Euclidean distance when there is **high dimensionality in the data.**
- In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates.
- Manhattan distance = $|x1 - x2| + |y1 - y2|$

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$



Manhattan Distance Examples



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

|2-0|+ |0-2|= 4

|3-0|+ |1-0|= 4

|5-0|+ |1-2|= 6

	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

Distance Matrix

4. Supremum Distance

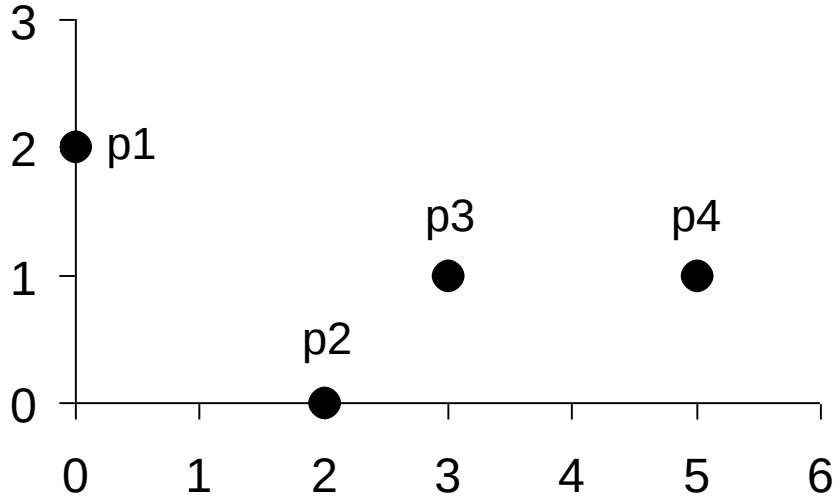
- Supremum Distance is **the maximum difference between any component** (attribute) of the vectors. It is a metric defined on a vector space **where distance between two vectors is the greatest of their difference along any coordinate dimension.** CD(x, y) = max i |xi − yi |
- Synonyms are **Lmax-Norm** or **Chessboard distance**. P= ∞ on Minkowski distance is the Chebyshev distance also known as supremum distance.
- Chebyshev distance is appropriate in cases when two objects are to be defined as different **if they are different in any one dimension.**

$$d(i,j) = \lim_{h\to\infty}\left(\sum_{f=1}^{p}|x_{if}-x_{jf}|^{h}\right)^{\frac{1}{h}} = \max_{f}^{p}|x_{if}-x_{jf}|$$

$$d_{BA} = \max\{|0-7|,|3-6|,|4-3|,|5+1|\}$$
$$= \max\{7,3,1,6\} = 7$$

Features	Coord1	Coord2	Coord3	Coord4	Coord5	Coord6
Object A	0	3	4	5		
Object B	7	6	3	-1		

Chebyshev Distance Examples



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

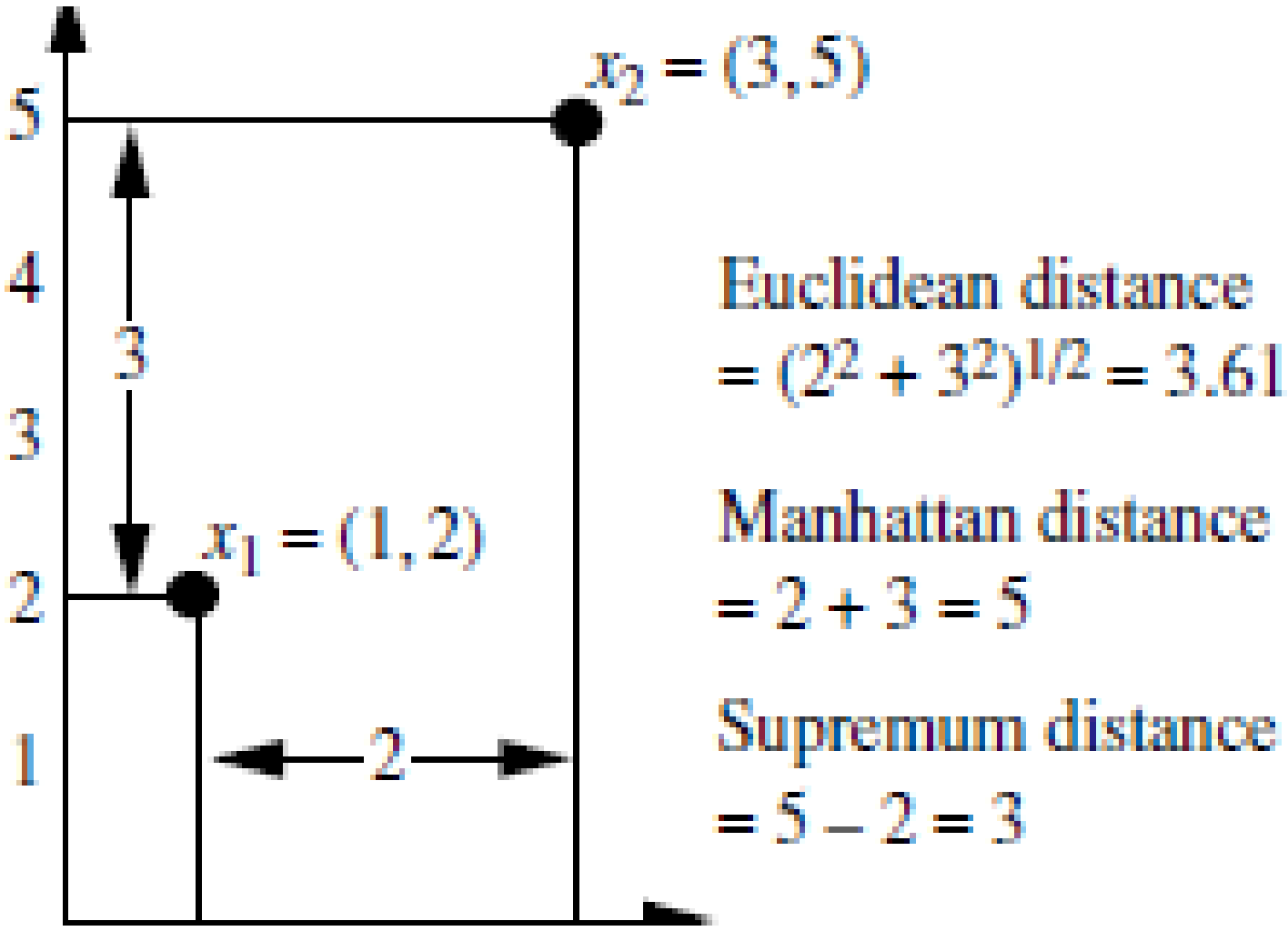
Max(|2-0|, |0-2|)= 2

Max(|3-0|, |1-0|)= 3

Max(|5-0|, |1-2|)= 5

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

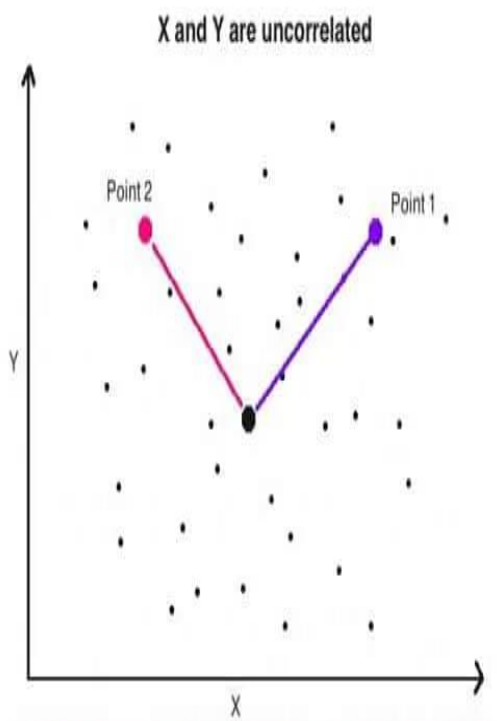
Distance Matrix

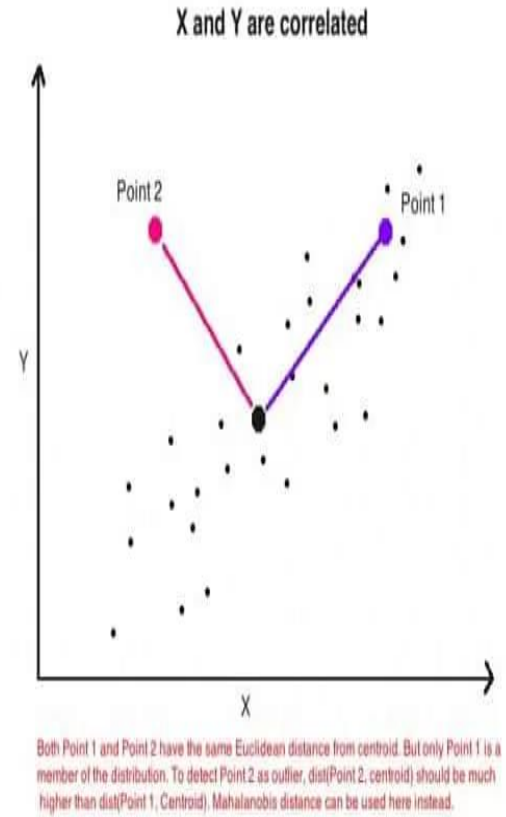


Manhattan, Euclidean & Supremum Distance between two objects

Euclidean distance Issues

- The two points above are equally distant (Euclidean) from the center. But only one of them (blue) is actually more close to the cluster, even though, technically the Euclidean distance between the two points are equal.
- This is because, Euclidean distance is a distance between two points only. It does not consider how the rest of the points in the dataset vary. So, it cannot be used to really judge how close a point actually is to a distribution of points.





Euclidean distance Issues

- The two tables above show the ‘area’ and ‘price’ of the same objects. Only the units of the variables change.
- Since both tables represent the same entities, the distance between any two rows, point A and point B should be the same. But Euclidean distance gives a different value even though the distances are technically the same in physical space.

Area (sq.ft)	Price ($ 1000's)	Area (acre)	Price ($M)
2400	156000	0.0550944	156
1950	126750	0.0447642	126.75
2100	105000	0.0482076	105
1200	78000	0.0275472	78
2000	130000	0.045912	130
900	54000	0.0206604	54

5. Mahalanobis Distance

- Mahalanobis Distance is a measure of distance between **a data vector and a set of data**, or a variation that measures the **distance between two vectors from the same dataset**.
- Mahalanobis Distance is used for calculating the distance between two data points in a **multivariate space**
- Also known as **quadratic distance,** measures separation of 2 groups of objects.

$$\mathbf{mahalanobis}(\mathbf{x},\mathbf{y}) = ((\mathbf{x}-\mathbf{y})^T \Sigma^{-1}(\mathbf{x}-\mathbf{y}))^{-0.5}$$

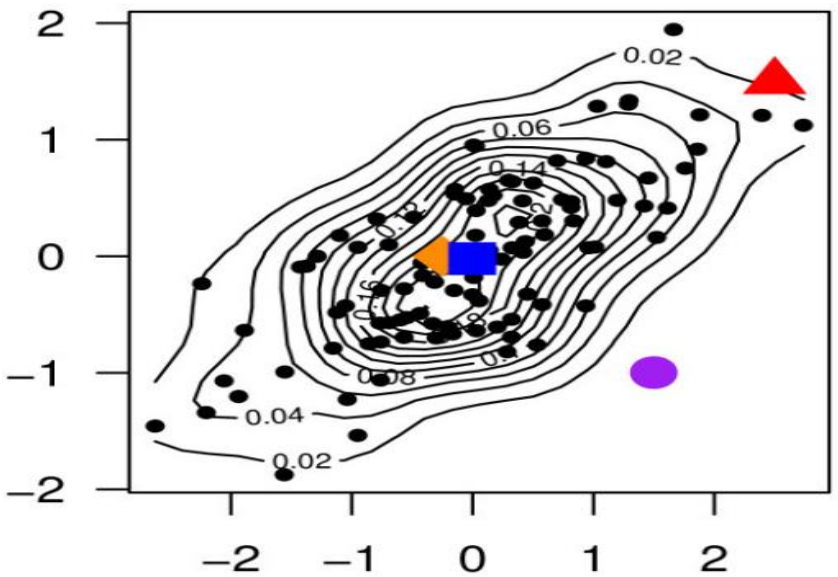
Σ is the covariance matrix

- The data of both group should have same number of variables(no. Of columns) but not same number of data(rows).
- *The **Mahalanobis distance** is a measure of the distance between a point P and a distribution D. The idea of measuring is, how many standard deviations away P is from the mean of D.*
- The benefit of using mahalanobis distance is, it takes **covariance** in account which helps in measuring the strength/similarity between two different data objects.
- **When the covariance matrix is identity Matrix, the mahalanobis distance is the same as the Euclidean distance.**
- **Useful for detecting outliers(multivariate), multivariate anomaly detection, classification** on highly imbalanced datasets and one-class classification

Formulas

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}.$$



$$d_M(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

$$= \sqrt{\begin{bmatrix} x_1 - y_1 & x_2 - y_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}}$$

$$= \sqrt{\begin{bmatrix} \frac{x_1 - y_1}{\sigma_1^2} & \frac{x_2 - y_2}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}}$$

$$= \sqrt{\frac{(x_1 - y_1)^2}{\sigma_1^2} + \frac{(x_2 - y_2)^2}{\sigma_2^2}}$$

Example

X	Y	Z
Height	Score	Age
64.0	580.0	29.0
66.0	570.0	33.0
68.0	590.0	37.0
69.0	660.0	46.0
73.0	600.0	55.0
m =68.0	600.0	40.0

n=5 . you want to know how far another person, v = (66, 640, 44), is from this data

COVARINACE MATRIX

$$COV(X,Y,Z)=\begin{vmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) \end{vmatrix}$$

SOLUTION

	X	Y	Z		X-Xbar	Y-Ybar	Z-Zbar		(X-Xbar)^2	(Y-Ybar)^2	(Z-Zbar)^2
	64	580	29		-4	-20	-11		16	400	121
	66	570	33		-2	-30	-7		4	900	49
	68	590	37		0	-10	-3		0	100	9
	69	660	46		1	60	6		1	3600	36
	73	600	55		5	0	15		25	0	225
Sum	340	3000	200	n=5				SUM	46	5000	440
Mean	68	600	40	n-1=4				VARIANCE	11.5	1250	110

									(X-Xbar)*(Y-Ybar)	(X-Xbar)*(Z-Zbar)	(Y-Ybar)*(Z-Zbar)
			11.5	50	34.75				80	44	220
		COV(X,Y,Z)=	50	1250	205				60	14	210
			34.75	205	110				0	0	30
									60	6	360
		INV(COV(X,Y,Z)=	3.68852	0.06273	-1.2821				0	75	0
			0.06273	0.00222	-0.024			SUM	200	139	820
			-1.2821	-0.024	0.45877			COVARIAN	50	34.75	205

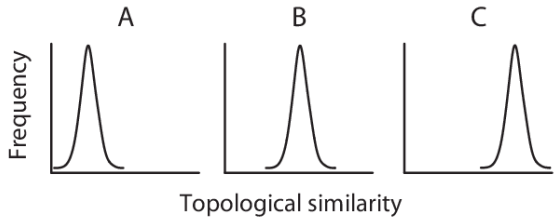
								Given v =	(66, 640, 44)			
								V-Mean	[-2,40,4]			
	-2	40	4	•	3.68852	0.06273	-1.2821	•		-2	=	
					0.06273	0.00222	-0.024			40		
					-1.2821	-0.024	0.45877			4		
	-9.9964	-0.1325	3.44127	•	-2	28.45732	5.33					
					40							
					4							

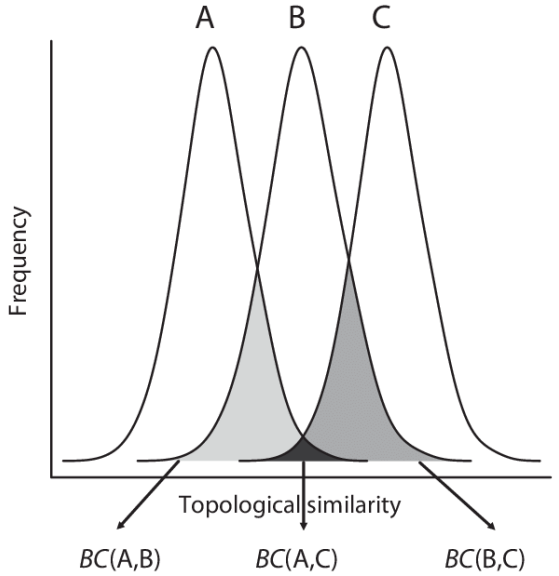
6. Bhattacharyya Distance

- **Bhattacharyya distance** measures the similarity of two probability distributions.
- It is closely related to the **Bhattacharyya coefficient** which is a measure of the amount of overlap between two statistical samples or populations.
- Both measures are named after Anil Kumar Bhattacharya, a statistician who worked in the 1930s at the Indian Statistical Institute
- For discrete probability distributions p and q over the same domain X, it is defined as: $D_B(p,q) = -\ln(BC(p,q))$
- where: $BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)}$
- is the Bhattacharyya coefficient. **Bhattacharyya coefficient** is an approximate measurement of the amount of overlap between two statistical samples. The coefficient can be used to determine the relative closeness of the two samples being considered. For continuous distributions, the Bhattacharyya coefficient is defined as: $BC(p,q) = \int \sqrt{p(x)q(x)}\,dx$

Bhattacharyya coefficient

The Bhattacharyya Coefficient (BC) is the overlap of the distribution of three similarity metrics





Bhattacharyya distance in the Multivariate Normal Case

$$b = \frac{1}{8}(\mu_2 - \mu_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\mu_2 - \mu_1)$$

$$+ \frac{1}{2} \ln \frac{|(\Sigma_1 + \Sigma_2)/2|}{|\Sigma_1|^{1/2}|\Sigma_1|^{1/2}},$$

- where μi and ∑i refer to mean and and covariance of ith cluster.Σ = Σ1 + Σ2 /2

Things to be taken care for numeric attribute distance calculation

- In some cases data are normalized before applying distance calculations.
- Transforming data to fall within a smaller or common range such as [-1,1] or [0.0,1.0]
- E.g values of height attributes
- Smaller then unit of measurement larger will be the range of values (weight effect)
- Normalization attempts to assign equal weight.

Similarity measures for symmetric and asymmetric binary data

1. Simple matching coefficient
2. Jaccard coefficient
3. Hamming distance

Proximity Measure for binary attribute

- **Nominal attribut**e with only 2 states (0 and 1) are known as **Binary attributes.**
- Treating binary variables as if they are interval-scaled can lead to misleading clustering results. Therefore, methods specific to binary data are necessary for computing dissimilarities.
- There are **2 types** of binary attributes
  1. **Symmetric Binary:** A binary attribute that has only 2 outcomes and both outcomes equally important **E**g: Male and Female
  2. **Asymmetric Binary:** outcomes of the states not equally important. E.g., medical test (positive vs. negative) Convention: assign 1 to most important outcome (e.g., HIV positive)
- Similarity that is based on symmetric binary variables is called **invariant similarity.**

Contingency Table

- Prepare contingency table for objects i and j, If aboth binary variables are thought of as having the same weight,where
  - **q** is the number of attributes that equal 1 for both objects i and j
  - **r** is the number of attributes that equal 1 for object i but that are 0 for object j
  - **s** is the number of attributes that equal 0 for object i but equal 1 for object j,
  - **t** is the number of attributes that equal 0 for both objects i and j.
  - **p** is the total number of attributes where **p = q + r + s + t.**

Object i	Object j		
	1	**0**	**Sum**
1	**q**=$f11$	**r**= $f10$	**q+r**
0	**s**=$f01$	**t**=$f00$	**s+t**
Sum	**q+s**	**r+t**	**p**=q+r+s+t

Symmetric Binary Dissimilarity

- Dissimilarity that is based on symmetric binary variables is called symmetric binary dissimilarity.
- Distance measure for symmetric binary variables ie, the dissimilarity between objects i and j:

$$d(i, j) = \frac{r + s}{q + r + s + t}$$

=(*f*10 + *f*01) / (*f*11 + *f*10 + *f*01 + *f*00)

Asymmetric Binary Dissimilarity

- Given two asymmetric binary variables, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match). Therefore, such binary variables are often considered "monary" (as if having one state).
- The dissimilarity based on such variables is called asymmetric binary dissimilarity. In asymmetric binary dissimilarity the number of negative matches, t, is considered unimportant and thus is ignored in the computation.

$$d(i, j) = \frac{r + s}{q + r + s}$$

= (*f*10 + *f*01) / (*f*11 + *f*10 + *f*01)

EXAMPLE 1

- Suppose that a patient record table contains the **attributes** **name, gender, fever, cough, test-1, test2, test-3,** and **test-4**, where **name** is an **object identifier**, **gender** is a **symmetric attribute**, and the remaining attributes are **asymmetric binary**. For asymmetric attribute values, let the values Y (yes) and P (positive) be set to 1, and the value N (no or negative) be set to 0. Estimate who all can have similiar disease based on disimilarity measure.

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

EXAMPLE 1 SOLUTION

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

		Mary		
		1	0	$\sum_{row}$
Jack	1	2	0	2
	0	1	3	4
	$\sum_{col}$	3	3	6

		Jim		
		1	0	$\sum_{row}$
Jack	1	1	1	2
	0	1	3	4
	$\sum_{col}$	2	4	6

		Mary		
		1	0	$\sum_{row}$
Jim	1	1	1	2
	0	2	2	4
	$\sum_{col}$	3	3	6

- The distance between each pair of the three patients, Jack, Mary, and Jim, is :
- These measurements suggest that Jim and Mary are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease

Symmetric Binary similarity: Simple matching coefficient

- **Simple matching coeffecient** is a similarity coeffecient defined for symmetric binary attributes. It is defined as:

S =1-D =1- $\frac{r+s}{q+r+s+t}$

= q+t/q+r+s+t

= $(f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00})$= number of matches / number of attributes

- This measure counts both presences and absences equally.
- **Application:** To find students who had answered similarlt in a test that consist of only true or false answers.

Asymmetric Binary Similarity: **Jaccard coefficient**

- Complementarily, we can measure the difference between two binary attributes based on the notion of similarity instead of dissimilarity.
- The asymmetric binary similarity between the objects i and j can be computed as,

$$sim(i, j) = \frac{q}{q + r + s} = 1 - d(i, j)$$

= number of 11 matches / number of non-zero attributes

= $(f_{11}) / (f_{01} + f_{10} + f_{11})$

- The coefficient sim(i, j) is called the **Jaccard coefficient**

EXAMPLE 1

- **Given: x** = 1 0 0 0 0 0 0 0 0 0 *and* **y** = 0 0 0 0 0 0 1 0 0 1 .
  Calculate SMC and Jaccard's coeffiecient.

SOLUTION

- $f01 = 2$ (the number of attributes where **x** was 0 and **y** was 1)
- $f10 = 1$ (the number of attributes where **x** was 1 and **y** was 0)
- $f00 = 7$ (the number of attributes where **x** was 0 and **y** was 0)
- $f11 = 0$ (the number of attributes where **x** was 1 and **y** was 1)
- SMC = $(f11 + f00) / (f01 + f10 + f11 + f00)$ = (0+7) / (2+1+0+7) = 0.7
- J = $(f11) / (f01 + f10 + f11)$ = 0 / (2 + 1 + 0) = 0

EXAMPLE 2

- Consider the following dataset, where objects are defined with binary attributes. Gender = {M, F}, ,Hobby = {T, C}, Job = {Y, N} Food = {V, N}, Caste = {H, M}, Education = {L, I} How you can calculate similarity between these 2 people based on similarity measure if Gender, Hobby and Job are symmetric binary attributes and Food, Caste, Education are asymmetric binary attributes?V-1,M-1,L-1

Object	Gender	Food	Caste	Education	Hobby	Job
Hari	M	V	M	L	C	N
Ram	M	N	M	I	T	N
Tomi	F	N	H	L	C	Y

Solution

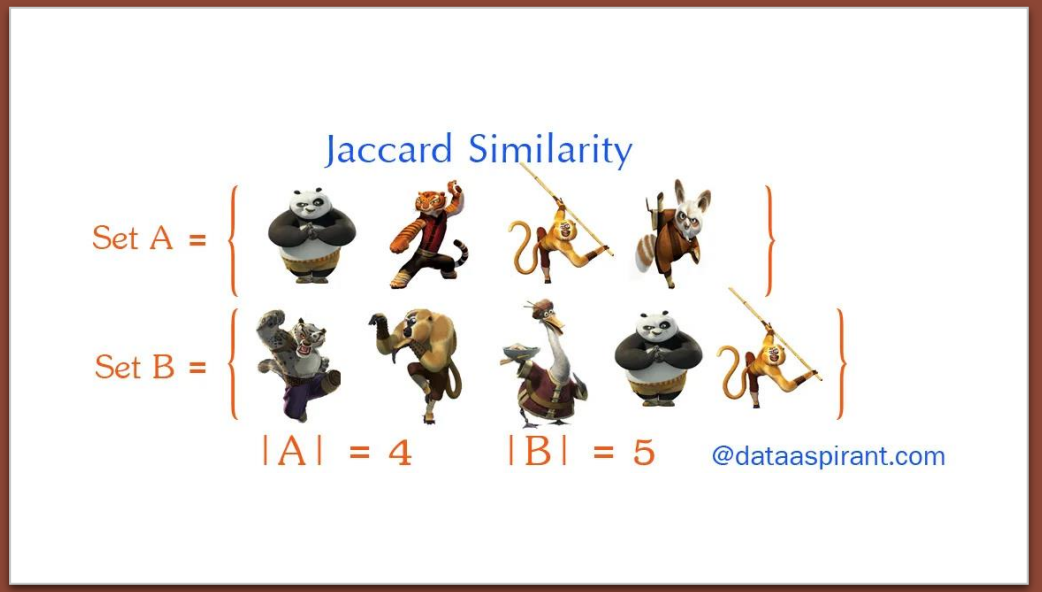
- Obtain the similarity matrix with Jaccard coefficient of objects

$$\mathcal{J} = \begin{array}{c} \\ H \\ R \\ T \end{array} \begin{array}{c} \begin{array}{ccc} H & R & T \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ \mathcal{J}(R,H) & 0 & 0 \\ \mathcal{J}(T,H) & \mathcal{J}(T,R) & 0 \end{bmatrix} \end{array}$$

- J(Hari, Ram)=1/(1+2+0)=0.33
- J(Ram, Tomi) = 0/0+1+1 =0
- J(Tom,Hari)=1/1+2+0=0.33

Jaccard coefficient

- So far discussed some metrics to find the similarity between objects. where the objects are points or vectors. When we consider Jaccard similarity these objects will be sets.



Jaccard Similarity J (A,B) = | Intersection (A,B) | / | Union (A,B) |

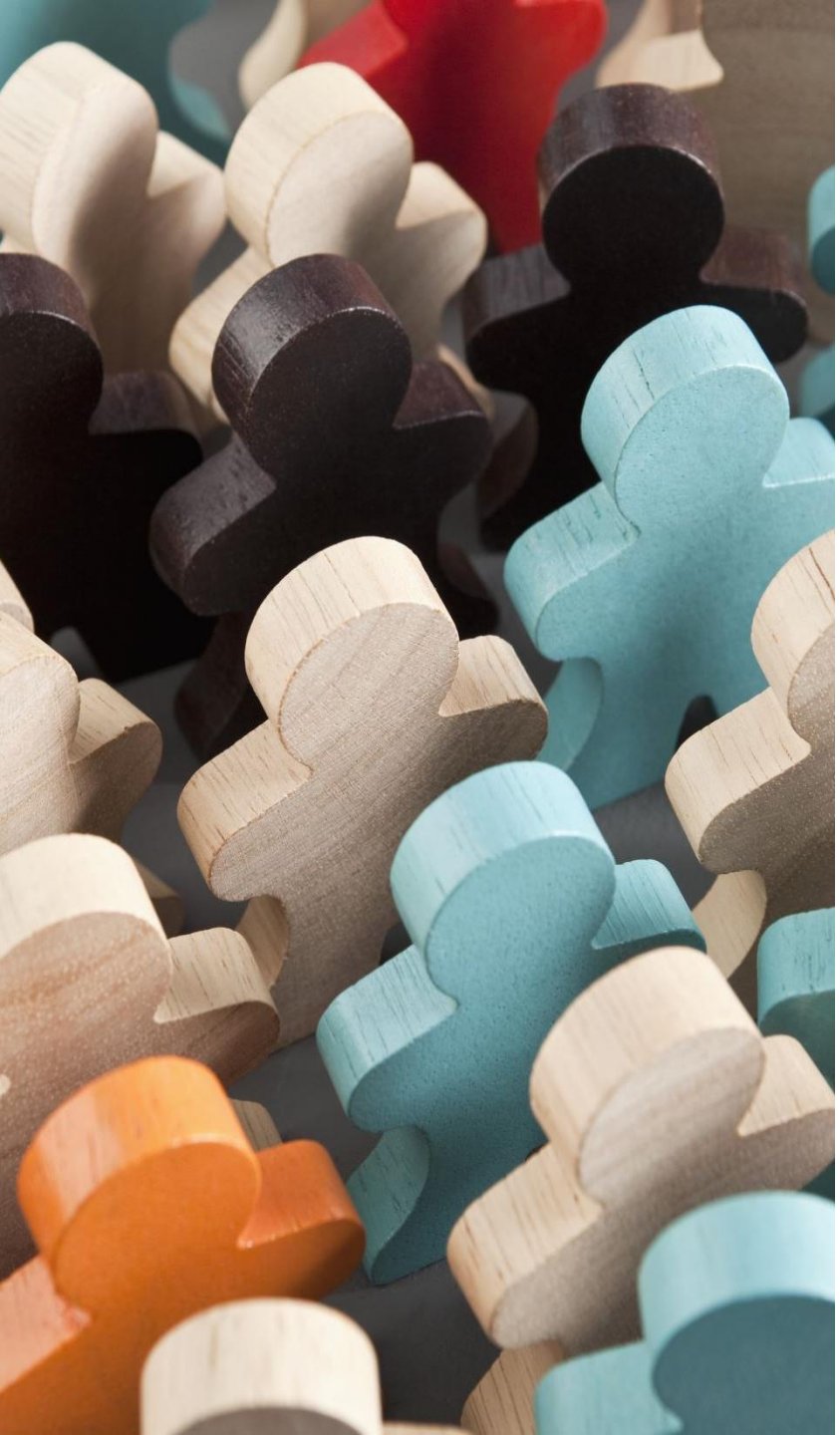
= 2 / 7

= 0.286

Hamming distance

- *Hamming Distance measures the distance between two strings of the same length. Hamming distance d=q+r.*
- *The Hamming Distance between two strings of the same length is the number of bit positions at which the corresponding characters or bits are different.*
- Example: x = 010101001 and y = 010011000
- Hamming distance = 3; there are 3 binary numbers different between the x and y.
  x = 010**1**01**0**00**1**
  y = 010**0**01**1**00**0**
- Hamming distance can be treated as a special instance of Manhattan distance, when attribute values ∈[0, 1] is called Hamming distance.
- Hamming distance is used to measure the distance between categorical variables
- *Hamming Distance=7*

euclidean and manhattan



Proximity Measure of Nominal Attributes/ Categorical Attributes

- The values of a Nominal attribute are categories, states, or "names of things". It is referred as categorical attributes or enumerations. The values do not have any meaningful order(rank, position) . A **nominal attribute** can take **2 or more states.**
- Example: Color (red, yellow, blue, green), profession.

"How is dissimilarity computed between objects described by nominal attributes?"

- **Simple Matching Method:** The **dissimilarity between two objects i and j** can be computed based on the ratio of mismatches:

  d(i,j)=(p-m)/p *Where **m***: #number of matches(i.e., the number of attributes for which i and j are in the same state), ***p***: total #number of variables/attributes **describing the objects**

- **similarity** can be computed as $$sim(i, j) = 1 - d(i, j) = \frac{m}{p}.$$

EXAMPLE

- Consider the nominal data given here.

object identifier	*test-1 (nominal)*	*test-2 (ordinal)*	*test-3 (numeric)*
1	code-A	excellent	45
2	code-B	fair	22
3	code-C	good	64
4	code-A	excellent	28

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}$$

here we have one nominal attribute, test-1, we set p = 1 so that d(i, j) evaluates to 0 if objects i and j match, and 1 if the objects differ.

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Proximity Measure of Nominal Attributes: Alternative approach

- Nominal attributes can be encoded by asymmetric binary attributes by creating a new binary attribute for each of the M states. For an object with a given state value, the binary attribute representing that state is set to 1, while the remaining binary attributes are set to 0.
- **Example:** To encode the nominal attribute **map color**, a binary attribute can be created for each of the **five colors** red, yellow, blue, green and white. For an object having the color yellow, the yellow attribute is set to 1, while the remaining four attributes are set to 0.

Example: Nominal attribute *educationlevel*: has three values,

- *Primary school, High school, university*
- We create three binary attributes.
- If a particular data instance in the original data has *university* as the value for *educationlevel*,
- then in the transformed data, we set the value of the attribute *university* to 1, and
- the values of attributes *primaryschool* and *highschool* to 0

Proximity Measures for Ordinal Attributes

- The values of an ordinal attribute have a meaningful order or ranking about them, yet the magnitude between successive values is unknown.
- An ordinal variable can be discrete or continuous.

1. Replace an ordinal variable valueby its rank: $r_{if} \in \{1, \ldots, M_f\}$
2. Normalize the rank by mapping the range of each variable onto [0, 1] by replacing i-th object in the f-th variable by using

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

3. Dissimilarity can then be computed using any of the distance measures described prior for numeric attributes, using zif to represent the f value for the ith object

- The value of ***f*** for the ith object is $\boldsymbol{x_{if}}$, and ***f*** has ***Mf*** ordered states, representing the ranking 1, . . . , Mf . Replace each xif by its corresponding rank, rif 2 {1, . . . , Mf}.

EXAMPLE

- Consider the nominal data given in test-2.

object identifier	*test-1 (nominal)*	*test-2 (ordinal)*	*test-3 (numeric)*
1	code-A	excellent	45
2	code-B	fair	22
3	code-C	good	64
4	code-A	excellent	28

There are three states for test -2, namely **fair, good,** and **excellent**, that is Mf = 3.

1. Replace each value for **test -2 by its rank**, the four objects are assigned the **ranks 3, 1, 2,** and **3,** respectively.
2. Normalizes the ranking by mapping **rank 1** to **0.0[1-1/3-1]**, **rank 2** to **0.5[2-1/3-1]**, and **rank 3** to **1.0[3-1/3-1]**.
3. Use Euclidean distance for dissimilarity calculation

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

EXAMPLE

- Given an ordinal data: freshman, sophomore, junior, senior. Compute dissimilarity matrix using Manhattan Distance.

SOLUTION

1. Replace an ordinal variable valueby its rank: freshman: 0; sophomore: 1; junior: 2; senior 3
2. Normalize rank: freshman: **0**((1-1)/(4-1)); sophomore: **1/3**((2-1)/(4-1); junior: **2/3**((3-1)/(4-1)); senior **1**((4-1)/(4-1)
3. Dissimilarity matrix using Manhattan Distance

$$\begin{bmatrix} 0 & & & \\ 1/3 & 0 & & \\ 2/3 & 1/3 & 0 & \\ 1 & 2/3 & 1/3 & 0 \end{bmatrix}$$

Similarity and dissimilarity measure for single attribute

- The following table shows the similarity and dissimilarity between two objects, *x* and *y,* with respect to a single, simple attribute.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = \lvert x - y \rvert / (n-1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = \lvert x - y \rvert$	$s = -d$, $s = \frac{1}{1+d}$, $s = e^{-d}$, $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Dissimilarity Between Attributes of mixed type

Dissimilarity Between Attributes of mixed type

- A database may contain different types of variables **interval-scaled, symmetric binary, asymmetric binary, nominal, and ordinal**

Approach 1

- compute the similarity between each attribute separately and then combine these attribute using a method that results in a similarity between 0 and 1. Combine the different variables into a single dissimilarity matrix, bringing all of the meaningful variables onto a common scale of the interval [0.0, 1.0].
- Suppose that the data set contains p variables of mixed type. The dissimilarity d(i, j) between objects i and j is defined as

$$d(i, j) = \frac{\sum_{f=1}^{P} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{P} \delta_{ij}^{(f)}}$$

- $\delta_{ij}^{(f)} = 0$
  - if either (1) x_{if} or x_{jf} is missing (i.e., there is no measurement of variable f for object i or object j),
  - or (2) $x_{if} = x_{jf} = 0$ and variable f is asymmetric binary;
- otherwise $\delta_{ij}^{(f)} = 1$

Dissimilarity Between Attributes of mixed type

- The contribution of variable f to the dissimilarity between i and j, that is, dij (f) is computed dependent on its type:
  - If f is **interval-based/numeric:**
    - use the normalized distance so that the values map to the interval [0.0,1.0].

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$$

  - If f is **binary or categorical**:
    - dij (f) = 0 if xif = xjf , or dij (f) = 1 otherwise
  - If f is **ordinal**:
    - compute ranks rif and and zif = rif−1 Mf−1 , and treat zif as numeric

EXAMPLE

Table 2.2: A sample data table containing attributes of mixed type.

object identifier	*test-1 (nominal)*	*test-2 (ordinal)*	*test-3 (numeric)*
1	code-A	excellent	45
2	code-B	fair	22
3	code-C	good	64
4	code-A	excellent	28

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Dissimilarity matrix for test-1

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

Dissimilarity matrix for test-2

- use the dissimilarity matrices obtained for test -1 and test –2
- Compute the dissimilarity matrix for the third attribute, test-3 (which is numeric)

SOLUTION

- compute the dissimilarity matrix for the third attribute, test-3 (which is numeric)
- maxhxh = 64 and minhxh = 22. The difference between the two is used to normalize the values of the dissimilarity matrix(eg: d21=(45-22)/(64-22)=0.55. Find all the dij values for attribute test-3
- The resulting dissimilarity matrix for test -3 is:

$$\begin{bmatrix} 0 & & & \\ 0.55 & 0 & & \\ 0.45 & 1.00 & 0 & \\ 0.40 & 0.14 & 0.86 & 0 \end{bmatrix}$$

SOLUTION

- Now compute the dissimilarity matrices for the three attributes: for each of the three attributes, the indicator dau(f) ij = 1
- Now compute each d(i,j): e.g.

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

$$d(3,1) = \frac{1(1)+1(0.50)+1(0.45)}{3} = 0.65.$$

- The resulting dissimilarity matrix is

$$\begin{bmatrix} 0 & & & \\ 0.85 & 0 & & \\ 0.65 & 0.83 & 0 & \\ 0.13 & 0.71 & 0.79 & 0 \end{bmatrix}$$

- As a result:
  - objects 1 and 4 are the most similar
  - objects 1 and 2 are the least similar

Proximity Measures for Interval-scaled Attributes

- Same as that of Numeric attributes

Proximity Measures for ratio-Scaled Attributes

Ratio-scaled attributes

- Numeric attributes, but unlike interval-scaled attributes, their scales are exponential,
- There are three methods :
  - treat them like interval-scaled variables — *not a good choice!(scale may be distorted)*
  - apply logarithmic transformation ($y_{if} = log(x_{if})$) and then treat it as an interval-scaled attribute
  - treat them as continuous ordinal data treat their ranks as interval-scaled.

Similarity features of Text

- **Matching** – How many words or characters match between two inputs. Ex: “abc” and “abz” have 2 characters matching but 1 different.
- **Sequence** – Are matching characters in same order or sequence? Ex: “abc” and “abz” have matching characters ‘ab’ in same sequence. “abc” and “zba” have matching ‘ab’ characters but not in same sequence.
- **Sound** – Some spellings might be different, but when we pronounce them they might sound same. That might be a criteria to categorize them as similar.

Similarity Measures for Textual Data

1. Edit distance
2. Cosine distance
3. Jaro distance
4. n-Gram distance
5. Longest Common Subsequence

EDIT DISTANCE APPLICATION

- Spell correction
  - The user typed "graffe"

  Which is closest?
    - graf
    - graft
    - grail
    - giraffe

- Computational Biology
  - Align two sequences of nucleotides

    AGGCTATCACCTGACCTCCAGGCCGATGCCC
    TAGCTATCACGACCGCGGTCGATTTGCCCGAC

  - Resulting alignment:

    −AGGCTATCACCTGACCTCCAGGCCGA−−TGCCC−−−
    TAG−CTATCAC−−GACCGC−−GGTCGATTTGCCCGAC

- Also for Machine Translation, Information Extraction, Speech Recognition

EDIT DISTANCE APPLICATION

- Evaluating Machine Translation and speech recognition

```
R Spokesman confirms    senior government adviser was shot
H Spokesman said    the senior          adviser was shot dead
              S     I             D                       I
```

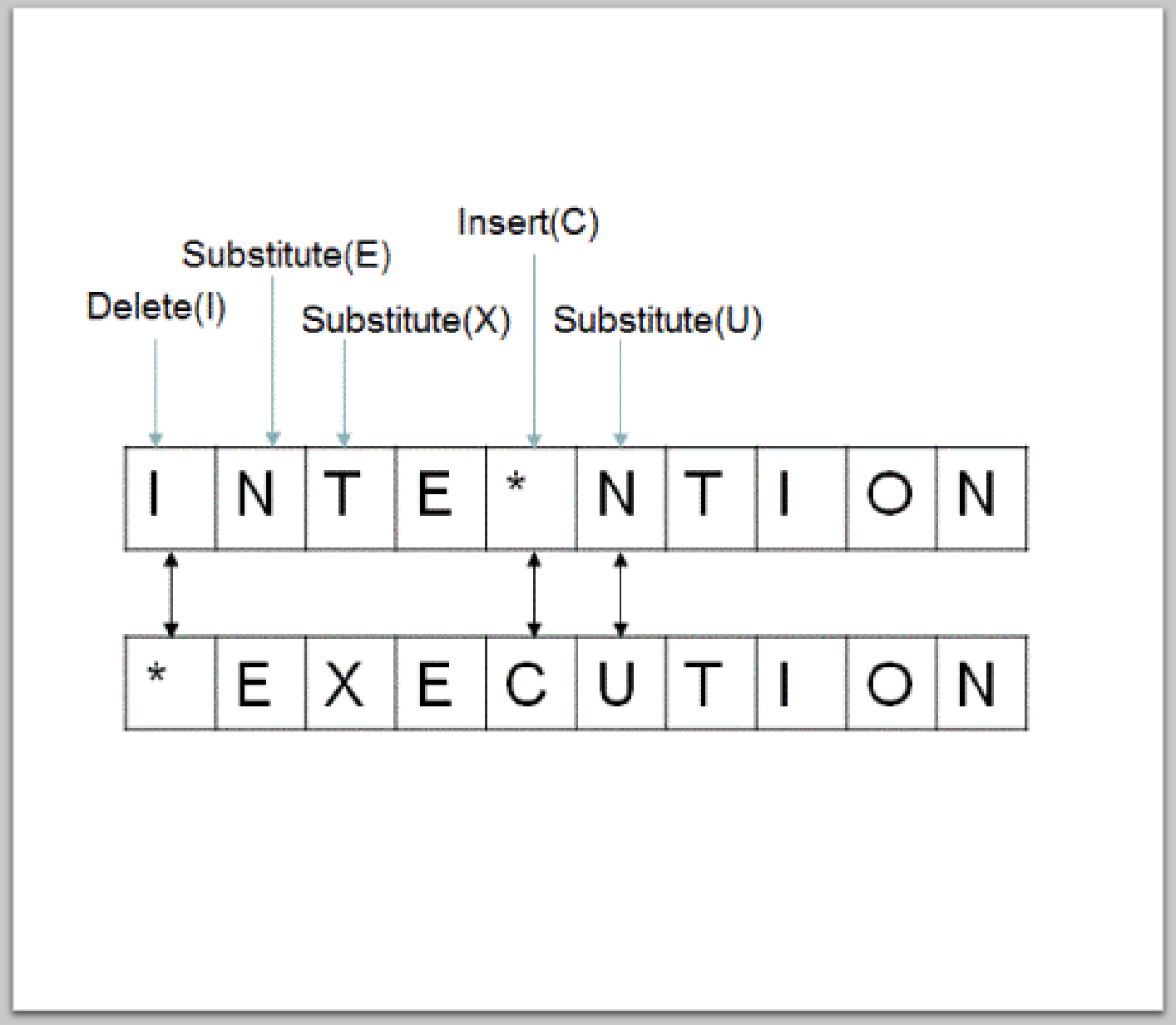
- Named Entity Extraction and Entity Coreference
  - IBM Inc. announced today
  - IBM profits
  - Stanford President John Hennessy announced yesterday
  - for Stanford University President John Hennessy

Edit distance

- **Minimum Edit distance** between two strings str1 and str2 is defined as the **minimum number of editing operations** like **insert, delete, substitute** required to transform **str1 into str2**.
- All the operations are of equal cost.
- **Edit distance** is a way of quantifying how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other.
- For example if str1 = "ab", str2 = "abc" then making an insert operation of character 'c' on str1 transforms str1 into str2. Therefore, edit distance between str1 and str2 is **1.**

EXAMPLE

- if str1 = "INTENTION" and str2 = "EXECUTION", then the minimum edit distance between str1 and str2 turns out to be 5 .
- If substitution cost is 2 then it becomes **Levenshtein Distance** which is 8.



EDIT DISTANCE PROBLEM AS AN OPTIMAL SUBSTRUCTURE using Dynamic Programming

- The Edit distance problem has an optimal substructure(problem can be broken down into smaller, simple , which can be broken down into yet simpler subproblems, and so on, until, finally, the solution becomes trivial).

$$
\textbf{Dist[i][j]} = \begin{cases} \textbf{when X[i-1] == Y[j-1]} \\ \quad \text{dist[i - 1][j- 1]} \\ \textbf{when X[i-1] != Y[j-1]} \\ \quad \textbf{1 +} \ \text{Min} \begin{cases} \text{dist[i-1][j],} \\ \text{dist[i][j - 1],} \\ \text{dist[i -1][j-1]} \end{cases} \end{cases}
$$

Why Backtracking is needed?

- Edit distance isn't sufficient
  - We often need to align each character of the two strings to each other
- We do this by keeping a "backtrace"
- Every time
  we enter a cell, remember where we came from
- When we reach the end,
  - Trace back
    the Path from The Upper Right Corner To Read Off The alignment

	If(r==c) just copy the digonal elements

substitute	delete
insert	Min(substitute,delete,insert)+1

	0/NULL	k	i	t	t	e	n
0/NULL	0	1	2	3	4	5	6
s	1						
i	2						
t	3						
t	4						
e	5						
n	6						
g	7						

	If(r==c) just copy the digonal elements

substitute ↘	delete ↓
insert →	Min(substitute,delete,insert)+1

	0/NULL	k	i	t	t	e	n
0/NULL	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

CHALLENGE

- Compute Edit distance table for **STR1:** Tamming test and **STR 2:** Taming text

		t	a	m	m	i	n	g		t	e	s	t
	0	1	2	3	4	5	6	7	8	9	10	11	12
t	1	0	1	2	3	4	5	6	7	8	9	10	11
a	2	1	0	1	2	3	4	5	6	7	8	9	10
m	3	2	1	0	1	2	3	4	5	6	7	8	9
i	4	3	2	1	1	1	2	3	4	5	6	7	8
n	5	4	3	2	2	2	1	2	3	4	5	6	7
g	6	5	4	3	3	3	2	1	2	3	4	5	6
	7	6	5	4	4	4	3	2	1	2	3	4	5
t	8	7	6	5	5	5	4	3	2	1	2	3	4
e	9	8	7	6	6	6	5	4	3	2	1	2	3
x	10	8	8	7	7	7	6	5	4	3	2	2	3
t	11	8	9	8	8	8	7	6	5	3	3	3	2

Dynamic Programming for Levenshtein Distance

Initialization

```
D(i,0) = i
D(0,j) = j
```

$$D[i,j] = \min \begin{cases} D[i-1,j] + \text{del-cost}(source[i]) \\ D[i,j-1] + \text{ins-cost}(target[j]) \\ D[i-1,j-1] + \text{sub-cost}(source[i], target[j]) \end{cases}$$

Recurrence Relation:

```
For each  i = 1…M
      For each  j = 1…N
                          D(i-1,j) + 1
            D(i,j)= min   D(i,j-1) + 1
                          D(i-1,j-1) +   2; if X(i) ≠ Y(j)
                                         0; if X(i) = Y(j)
```

Termination:

```
D(N,M) is distance
```

EXAMPLE

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

EXAMPLE

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

EXAMPLE

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Adding Backtrack

Base conditions:

$D(i,0) = i \qquad D(0,j) = j$

Termination:

$D(N,M)$ is distance

Recurrence Relation:

For each $i = 1 \ldots M$

For each $j = 1 \ldots N$

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{deletion} \\ D(i,j-1) + 1 & \text{insertion} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \quad \text{substitution} \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

$$ptr(i,j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

BACKTRACKING LEVENSHTEIN

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**	
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9	
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10	
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11	
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10	
e	4	↙ 3	← 4	↙← **5**	← **6**	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	
t	3	↙←↓ 4	↙←↓ **5**	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8	
n	2	↙←↓ **3**	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7	
i	**1**	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8	
#	**0**	1	2	3	4	5	6	7	8	9	
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**	

Types of Edit Distance

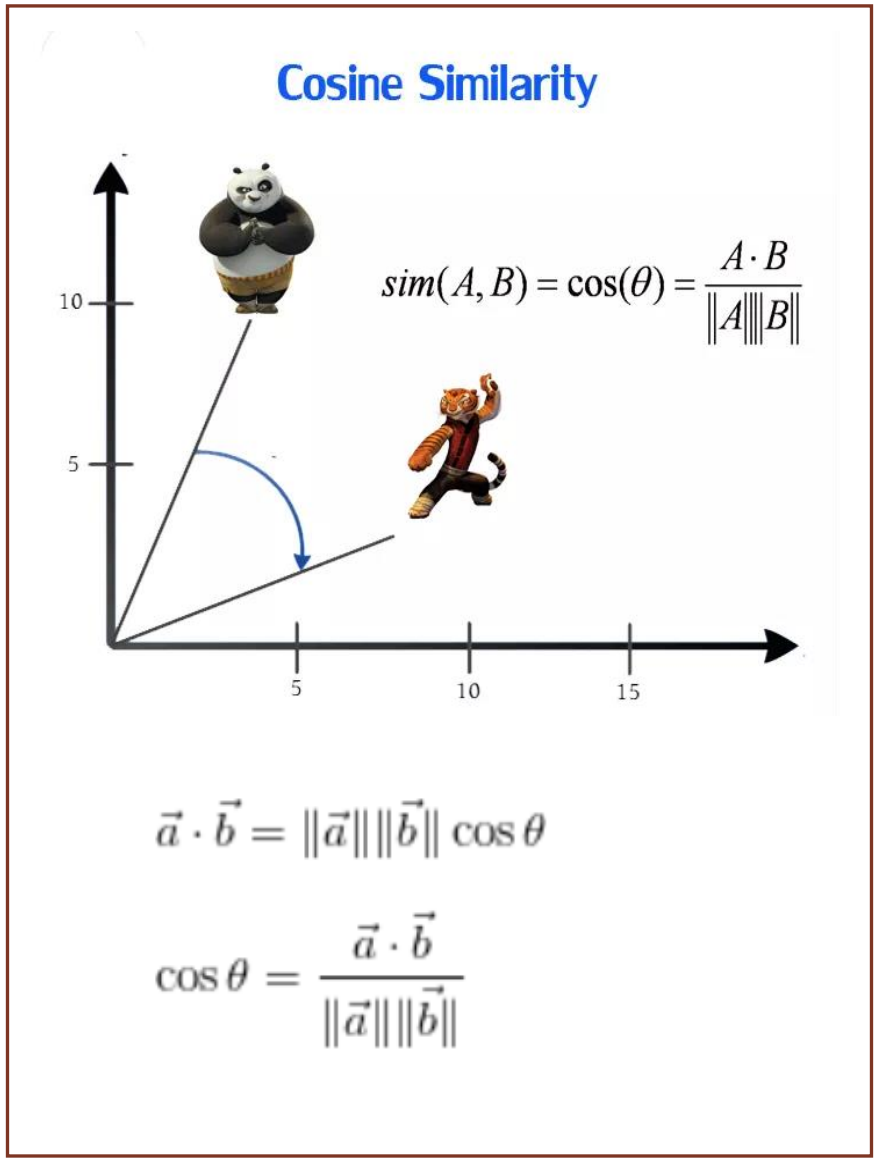
- Different types of edit distance allow different sets of string operations. For instance:
  - **Levenshtein distance** allows **deletion, insertion** and **substitution**.
  - **Longest common subsequence (LCS)** distance allows only **insertion** and **deletion**, not substitution.
  - **Hamming distance** allows only **substitution**, hence, it only applies to strings of the same length.
  - **Jaro distance** allows only **transposition**.

Cosine distance

- Cosine distance metric is used to find similarities between **different documents.**
- In cosine metric we measure the **degree of angle between two documents/vectors**. It is thus a **judgment of orientation** and **not magnitude.**
- Two vectors with the **same orientation/similarity** have a **cosine similarity of 1**, two vectors at **90°** have a **similarity of 0**. Whereas **two vectors diametrically opposed** having a **similarity of -1**, independent of their magnitude.
- By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of **0° is 1**, and it is **less than 1** for any other angle.
- The cosine similarity metric finds the **normalized dot product** of the two attributes.
- If A and B are two document vectors, then

cos(A, *B*) = $similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$

- where · indicates **vector dot product** and **|| A ||** is the length of vector A.
- dotproduct holds information about the direction of the vectors



Interpretation of Cosine Similarity Matrix

- Unlike other similarity measures, a cosine similarity is a measure of the **direction-length resemblance between vectors.**
- An **angle of** 0° means that **cos θ = 1** and that the **vectors are oriented in identical directions**; i.e., that the **corresponding data sets are completely similar to one another**.
- An angle of **90°** means that **cos θ = 0** and that the corresponding **variables are perpendicular**, **but not necessarily that are uncorrelated** unless these are also mean-centered.

Example Cosine similarity

d_1 = **3 2 0 5 0 0 0 2 0 0**

d_2 = **1 0 0 0 0 0 0 1 0 2**

$\boldsymbol{d_1 \cdot d_2}$= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$||\boldsymbol{d_1}||$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)$^{0.5}$ = (42) $^{0.5}$ = 6.481

$||\boldsymbol{d_2}||$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2) $^{0.5}$ = (6) $^{0.5}$ = 2.245

cos($\boldsymbol{d_1}$, $\boldsymbol{d_2}$) = .3150,

Cosine Distance=1-cos(d1,d2)=1-.3150

EXAMPLE 2

- Compute distance between Document1 and Document2 using Cosine Metric

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

SOLUTION

$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$

$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$

$d_1 \bullet d_2 = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25$

$||d_1|| = (5*5+0*0+3*3+0*0+2*2+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

$||d_2|| = (3*3+0*0+2*2+0*0+1*1+1*1+0*0+1*1+0*0+1*1)^{0.5} = (17)^{0.5} = 4.12$

$\cos(d_1, d_2) = 0.94$

Example

- Given 2 statements
  - Julie loves me more than Linda loves me
  - Jane likes me more than Julie loves me
- Find how similar these texts are, purely in terms of word counts (and ignoring word order)

SOLUTION

me	2	2
Jane	0	1
Julie	1	1
Linda	1	0
likes	0	1
loves	2	1
more	1	1
than	1	1

- The two vectors are, again:
- a: [2, 0, 1, 1, 0, 2, 1, 1]

  b: [2, 1, 1, 0, 1, 1, 1, 1]

- The cosine of the angle between them is about 0.822.
- These vectors are 8-dimensional. A virtue of using cosine similarity is clearly that it converts a question that is beyond human ability to visualise to one that can be. In this case you can think of this as the angle of about 35 degrees which is some 'distance' from zero or perfect agreement.

Jaro distance

- **Jaro Similarity** is the measure of similarity between two strings. The value of Jaro distance ranges from 0 to 1. where **1 means the strings are equal** and **0 means no similarity between two strings. It checks sequence similarity**
- Jaro distance measure is based on the **number or order of the characters between two strings which are common;**

$$Jaro\ similarity = \begin{cases} 0, \text{ if m=0} \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right), \text{ for m!=0} \end{cases}$$

where:

- **m** is the **number of matching characters**
- **t** is **half the number of transpositions**
- **|s1|** and **|s2|** is the **length of string s1** and **s2** respectively

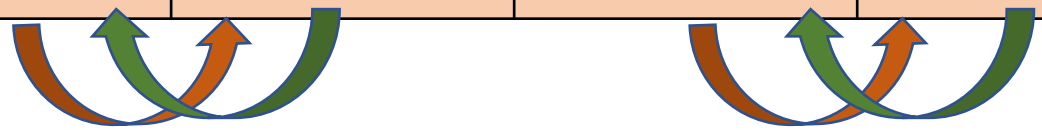
- **Condition for matching chracters and getting number of transpositions:** Two characters from s1 and s2 are **considered matching** only if they are the same and not farther than $\left\lfloor \frac{max(|s1|, |s2|)}{2} \right\rfloor - 1$ characters apart.

- For example, in comparing **CRATE** with **TRACE**, only '**R' 'A' 'E'** are the **matching characters**, i.e. **m=3.** Although 'C', 'T' appear in both strings, they are farther apart than 1 (the result of (5/2)floor -1. Therefore, t=0 .
- Each character of s_1 is compared with all its matching characters in s_2.The number of matching (but different sequence order) characters divided by 2 defines the number of transpositions.

EXAMPLE 1

- Let **s1=”arnab”**, **s2=”raanb”**, Calculate Jaro distance.
- Number of matching characters= **5** . Since Not farther than (5/2)floor -1.

a	r	n	a	b
r	a	a	n	b



- But the order is not the same, so the number of characters which are **not in order is 4,** so the **number of transpositions/2 is 4/2=2**.
- Therefore Jaro similarity can be calculated as follows:
  Jaro Similariy = (1/3) * {(5/5) + (5/5) + (5-2)/5 } = **0.86667**
- **Jaro Distance =1-0.86667=0.13333**

EXAMPLE 2

- Given S1=WINKLER and s2=WELFARE

W	I	N	K	L	E	R
W	E	L	F	A	R	E

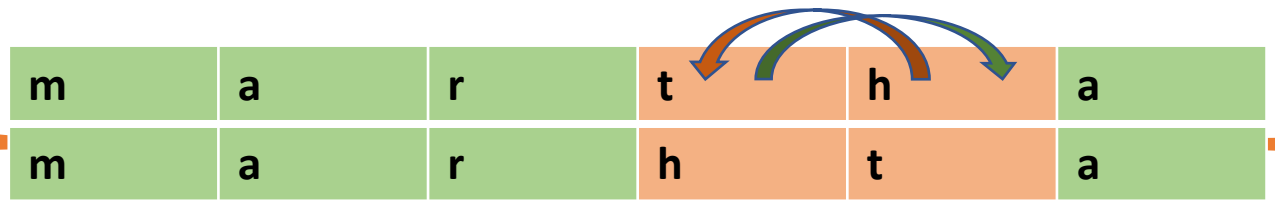
- Matching characters=WLER and WLRE
- M=4,S1=7 S2=7
- Transposition=2/2=1
- Jaro Similarity= (1/3) * {(4/7) + (4/7) + (4-1)/4}=53/(3*28)=53/84=0.63095

W	L	E	R
W	L	R	E

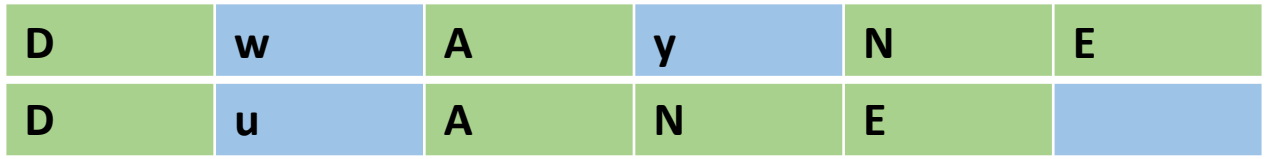
EXAMPLE 2

1. Given the strings *s1 = "**martha**"* and *s2 ="**marhta**". Calculate Jaro Similarity.*
2. Given the strings *s1 = **DWAYNE*** and *s2 =**DUANE.** Calculate Jaro Similarity.*
3. Given the strings *s1 = "**CRATE**"* and *s2 = "**TRACE**". Calculate Jaro Similarity.*

Solution

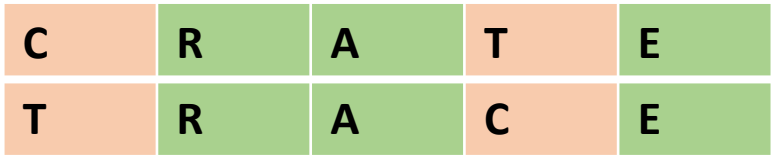


1. ***m*** = 6
   t = 2/2 =1 (2 couples of non matching characters, the 4-th and 5-th) { t/h ; h/t }
   |s1| = 6 and **|s2| = 6**
   - **Jaro Similarity =** (⅓) (6/6 + 6/6 + (6–1)/6) = 17/18 **= 0.944**



2. **m=4**
   t=0. (In DwAyNE versus DuANE the matching letters are already in the same order D-A-N-E, so no transpositions are needed.)
   |s_1|=6 and **|s_2|=5**
   - **Jaro Similarity =**(⅓) *(4/6 + 4/5 + (4–0)/4) **= 37/45=0.822**

3. ***m*** **= 3**
   t = 0 (only **'R' 'A' 'E'** are the **matching characters**, i.e. **m=3.** Although 'C', 'T' appear in both strings, they are farther apart than 1 (the result of (5/2)floor -1. Therefore, t=0 . }
   |s1| = 5 and **|s2| = 5**
   - **Jaro Similarity** = (⅓) *(3/5 + 3/5 + (3–0)/3) **=11/15**=0.733333



Jaro-Winkler Similarity

- Jaro – Winkler Similarity uses a prefix scale ‘p’ which **gives a more accurate answer** when **the strings have a common prefix up to a defined maximum length l**.
- Jaro Winkler similarity is defined as follows

$$Sw = Sj + P * L * (1 – Sj)$$

  - where, **Sj,** is jaro similarity
  - **Sw,** is jaro- winkler similarity
  - **P** is the scaling factor (0.1 by default)
  - **L** is the length of the matching prefix upto a maximum 4 characters
- The lower the Jaro–Winkler distance for two strings is, the more similar the strings are. The score is normalized such that 1 means an exact match and 0 means there is no similarity. The **Jaro–Winkler similarity** is the inversion
- Although often referred to as a *distance metric*, the Jaro–Winkler distance is not a metric in the mathematical sense of that term because it does not obey the triangle inequality.

EXAMPLE

- Let s1=”arnab”, s2=”aranb”. The Jaro similarity of two strings is 0.933333 (From the above calculation.)
- The length of matching prefix is 2(ar) and we take scaling factor as 0.1
- Substituting in the formula;
  Jaro-Winkler Similarity= 0.9333333 + 0.1 * 2 * (1-0.9333333) = 0.946667

EXAMPLE

- *“martha”/ “marhta”*
- prefix lenght of ***l = 3*** (which refers to *“mar”*). We get to:
- ***dw*** = 0,944 + ((0,1*3)(1–0,944)) = 0,944 + 0,3*0,056 = 0,961Jaro-Winkler distance = 96,1%

n-Gram edit distance

- An ***n*-gram** is a **contiguous sequence of *n* items** from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.
- An n-gram can be obtained by splitting a string in sequences with the length n, ie substrings of length N are named "N-grams".
- **Example:** string “abcde”
  - **bigrams** are: ab, bc, cd, and de
  - **trigrams** will be: abc, bcd, and cde
  - **4-grams** will be abcd, and bcde.
- **There are different variants of N-gram distance:**
  - **Dice N-gram Matching Method**
  - **Generalized N-gram Matching for string matching**
  - **Bigram method**
  - **Trigram Method**

Dice N-gram Matching

- **Dice similarity** is the most popular N-Gram method.
- The measures are defined as the **ratio of the number of n-grams that are shared by two strings** and **the total number of n-grams in both strings**.
- The coefficient value varies between zero and one. If two terms have no characters in common then the coefficient value is zero. On the other hand, if they are identical, the coefficient value will be one.
- For two string X and Y, the Dice coefficient is measured as

$$d(X,Y) = \frac{2(n-gram(X \frown Y))}{(n-gram(X)) + (n-gram(Y))}$$

- where n-grams(X) is a multi-set of letter n-grams in X.
- Dice coefficient with bigrams (DICE) is a particularly popular word similarity measure

Dice N-gram Matching

- DICE(Zantac, Contac)
- Zantac: za, an, nt, ta, ac
- Contac: co, on, nt, ta, ac
- $$d(X,Y) = \frac{2(n-gram(X \frown Y))}{(n-gram(X)) + (n-gram(Y))}$$

  = (2 · 3)/(5 + 5) = 6/10=0.6

- **Problems of Dice N-gram**
  - **Low resolution:** it often fails to detect any similarity between strings that are very much alike. Example: Verelan/Virilon have no n-grams in common.
  - It can return the **maximum similarity** value of 1 for **strings that are non-identical. E**xample, both Xanex and Nexan are composed of the same set of bigrams: {an,ex,ne,xa}.
  - It often associates n-grams that occur in radically different word positions, as in the pair Voltaren/Tramadol.

N-gram Distance

- **N-gram distance:**

$$\text{Sim}(s1,s2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} (h(i))$$

- Where **N = max(N(s1), N(s2))**
- **n=2 for bigram, 3 for trigram,4 for 4-grams**
- h(i) = 1 if n-element subsequence beginning from position i in s1 appears in s2 h(i) = 0 otherwise

Bigram and Trigram

- Bigram

$$sim(s_1, s_2) = \frac{1}{N-2+1} \sum_{i=0}^{N-2+1} h(i) = \frac{1}{N-1} \sum_{i=0}^{N-1} h(i)$$

- Trigram

$$sim(s_1, s_2) = \frac{1}{N-3+1} \sum_{i=0}^{N-3+1} h(i) = \frac{1}{N-2} \sum_{i=0}^{N-2} h(i)$$

Generalized N-gram Matching

- Generalized n-gram matching was introduced by Niewiadomski

$$sim(s_1, s_2) = f(n_1, n_2) \sum_{i=n_1}^{n_2} \sum_{j=1}^{N-n+1} h(i, j)$$

- $$\text{where } f(n_1, n_2) = \frac{2}{(N-n_1+1)(N-n_2+2)-(N-n_2+1)(N-n_1)}$$ $=2/(N^2+N)$
- denotes the number of possible substrings not shorter than n1 and not longer than n2 in s1

EXAMPLE

1. Let s1 = ELOQUENTLY, s2 = INELOQUENT. N(s1) =10 and N(s2) =10. Calculate the 4 n-gram distances.

SOLUTION:

s2 occurs in the substring of s1 as follows:

1-element E, L, O, Q, U, E, N, T = 8

2-element EL, LO, OQ, QU, UE, EN, NT, = 7

3-element ELO, LOQ, OQU, QUE, UEN, ENT = 6

4-element ELOQ, LOQU, OQUE, QUEN, UENT = 5

5 -element ELOQU, LOQUE, OQUEN, QUENT = 4

6-element ELOQUE, LOQUEN, OQUENT = 3

7-element ELOQUEN, LOQUENT = 2

8 -element ELOQUENT = 1

EXAMPLE

1. Generalized n-gram matching

$$sim(s_1, s_2) = \frac{2}{N^2+N} \sum_{i=1}^{N} \sum_{j=1}^{N-i+1} h(i, j) = \frac{2}{10^2+10} \times \frac{8+7+6+5+4+3+2+1}{1} = \frac{2*36}{110} = 0.65$$

$$\text{Bigram} = sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{10-1} \times \frac{7}{1} = \frac{7}{9} = 0.77$$

$$\text{Trigram} = sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{10-2} \times 6 = \frac{6}{8} = 0.75$$

$$\text{Dice's Coefficient} = d(X, Y) = \frac{2(n-gram(X \frown Y))}{n-gram(X)+(n-gram(Y))} = \frac{2(7)}{9+9} = \frac{14}{18} = 0.77$$

EXAMPLE 2

- Let s1 = PROGRAMMER, s2 = PROGRAMMING. N(s1) =10 and N(s2) =11, max{N(s1), N(s2)} = 11. Calculate the 4 n-gram distances.

1. Generalized n-gram matching

$$sim(s_1, s_2) = \frac{2}{N^2+N} \sum_{i=1}^{N} \sum_{j=1}^{N-i+1} h(i,j) = \frac{2}{11^2+11} \times \frac{9+7+6+5+4+3+2+1}{1} = \frac{2*36}{132}$$

72/132=0.545

2. Dice's Coefficient $= d(X,Y) = \frac{2(n\text{-}gram(X \frown Y))}{n\text{-}gram(X)+(n\text{-}gram(Y))} = \frac{2(7)}{9+10} = \frac{14}{19} = 0.74$

3. Bigram $= sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{11-1} \times \frac{7}{1} = \frac{7}{10} = 0.70$

4. Trigram $= sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{11-2} \times 61 = \frac{6}{9} = 0.64$

SOLUTION

- s2 occurs in the substring of s1 as follows:

1-element P, R, O, G, R, A, M, M = 8

2-element PR, RO, OG, GR, RA, AM, MM = 7

3-element PRO, ROG, OGR GRA, RAM, AMM = 6

4-element PROG, ROGR, OGRA, GRAM, RAMM = 5

5 -element PROGR, ROGRA, OGRAM, GRAMM = 4

6-element PROGRA, ROGRAM, OGRAMM = 3

7-element PROOGRAM, ROGRAMM = 2

8 -element PROGRAMM = 1

Longest Common Subsequence Algorithm

- A subsequence of a string S, is a set of characters that appear in left - to-right order, but not necessarily consecutively.
- **Example**: Consider ACTTGCG
  - ACT , ATTC , T , ACTTGC are all subsequences. TTA is not a subsequence. There are 2n subsequences of string S of length n.
- A common subsequence of two strings is a subsequence that appears in both strings. A longest common subsequence is a common subsequence of maximal length.

Example

S1 = AAACCGTGAGTTATTCGTTCTAGAA

S2 = CACCCCTAAGGTACCTTTGGTTC

LCS is ACCTAGTACTTTG

RECURSIVE ALGORITHM

- If last characters of both sequences match (or X[m-1] == Y[n-1]) then L(X[0..m-1], Y[0..n-1]) = 1 + L(X[0..m-2], Y[0..n-2])
- If last characters of both sequences do not match (or X[m-1] != Y[n-1]) then L(X[0..m-1], Y[0..n-1]) = MAX (L(X[0..m-2], Y[0..n-1]), L(X[0..m-1], Y[0..n-2])

Examples:

1) Consider the input strings "AGGTAB" and "GXTXAYB". Last characters match for the strings. So length of LCS can be written as: L("AGGTAB", "GXTXAYB") = 1 + L("AGGTA", "GXTXAY")

2) Consider the input strings "ABCDGH" and "AEDFHR". Last characters do not match for the strings. So length of LCS can be written as: L("ABCDGH", "AEDFHR") =

MAX (L("ABCDG", "AEDFHR"), L("ABCDGH", "AEDFH"))

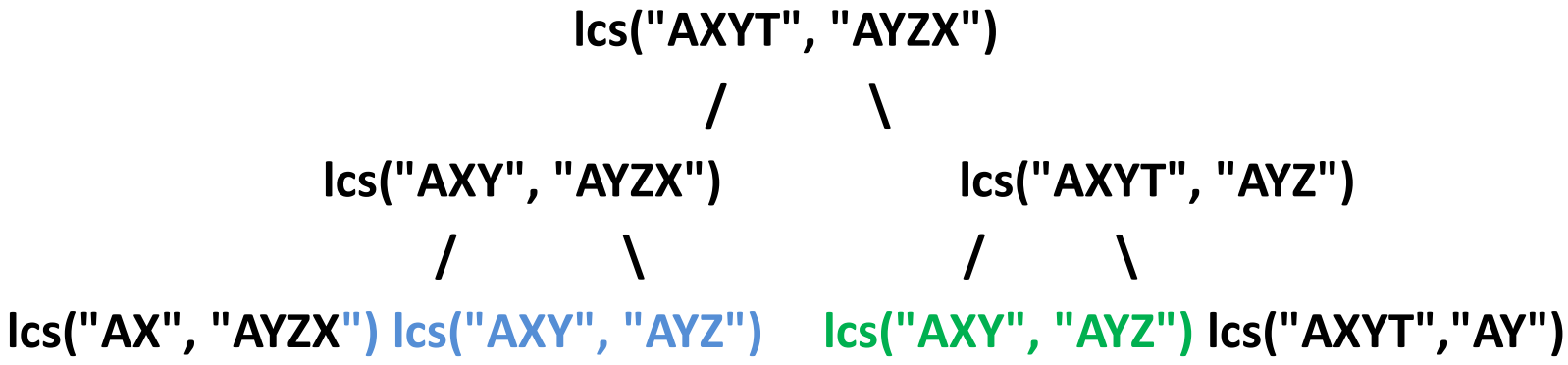
RECURSIVE ALGORITHM

```
LCS(i,j){
If(A[i] ≠ null ||B[j]≠null)
        return o;
Elseif(A[i]==B[j])
        return (1+ LCS(i+1,j+1))
Else
        return(max(LCS(i+1,j),LCS(I,j+1))

}
```

Overlapping Subproblem

- This is a correct solution but it's very time consuming. For example, if the two strings have no matching characters, so the last line always gets executed, the time bounds are binomial coefficients, which (if m=n) are close to 2^n.



- In the above partial recursion tree, lcs(“AXY”, “AYZ”) is being solved twice. So this problem has Overlapping Substructure property and re-computation of same sub problems can be avoided by either using Memoization or Tabulation by using "top down“ approach of dynamic programming. The concept is to cache the result of a function given its parameter so that the calculation will not be repeated; it is simply retrieved, or memo-ed. Most of the time a simple array is used for the cache table, but a hash table or map could also be employed.

DYNAMIC PROGRAMMING APPROACH

Theorem: Let X =< x1, x2, . . . , xm > and Y =< y1, y2, . . . , yn > be sequences, and let Z =< z1, z2, . . . , zk > be any LCS of X and Y .

1. If xm = yn, then zk = xm = yn and Zk−1 is an LCS of Xm−1 and Yn−1.
2. If xm ≠ yn, then zk ≠ xm implies that Z is an LCS of Xm−1 and Yn .
3. If xm ≠ yn, then zk ≠ yn implies that Z is an LCS of X and Yn−1.

- So dynamic way of solving LCS is

$$LCS[i, j] = \begin{cases} \textbf{if(A[i]==B[j])} \\ \qquad \textbf{return (1+ LCS(i-1,j-1)} \\ \textbf{Else} \\ \qquad \textbf{return max(LCS[i, j − 1], LCS[i − 1, j])} \end{cases}$$

ALGORITHM

```
LCS – Length(X, Y )
m= length[X]
 n= length[Y ]
 for i =1 to m
        c[i, 0] = 0
 for j =0 to n
        c[0, j] = 0
 for i = 1 to m
        for j = 1 to n
              if xi = =yj
                    c[i, j] = c[i – 1, j – 1] + 1
                    B[i, j] := ‘D’ or ↖
              else if (c[i – 1, j] ≥c[i, j – 1])
                    c[i, j] =c[i – 1, j]
                    B[i, j] := ‘U’ or ↑
              else
                    c[i, j] =c[i, j – 1]
                    B[i, j] := ‘L’ or ←
        return c and B
```

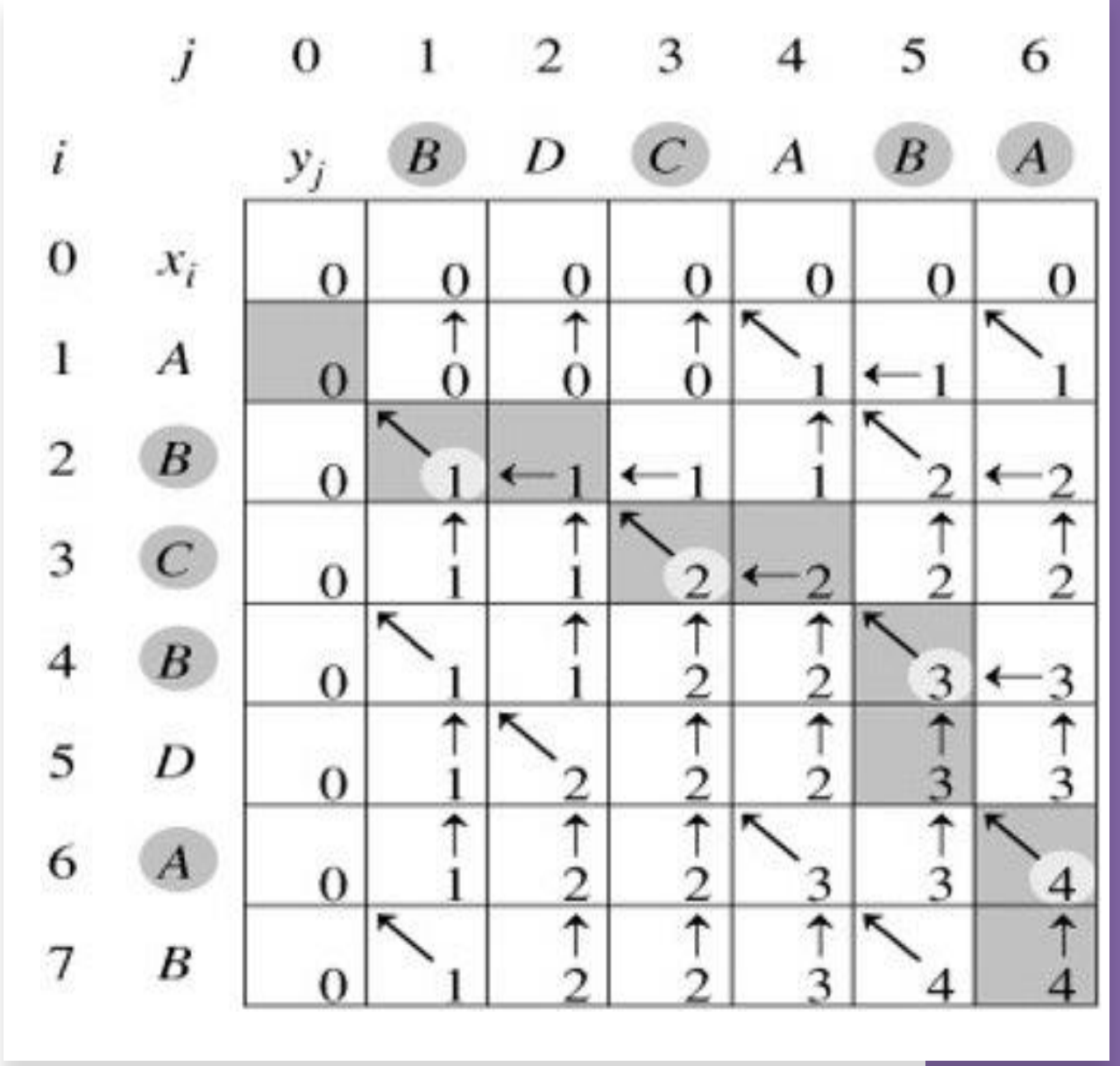
SEQUENCE RETRIEVAL

Algorithm: Print-LCS (B, X, i, j)

```
if i = 0 and j = 0
return
if B[i, j] = 'D' or ↖
        Print-LCS(B, X, i-1, j-1)
        Print(x_i)
else if B[i, j] = 'U' or ↑
        Print-LCS(B, X, i-1, j)
else
        Print-LCS(B, X, i, j-1)    //for 'L' or ←
```

EXAMPLE 1

- we have two strings ***X =*** ***ABCBDAB*** and ***Y = BDCABA*** to find the longest common subsequence. Following the algorithm LCS-Length-Table-Formulation



i		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B		0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C		0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B		0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D		0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A		0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B		0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

EXAMPLE 2

X= innovation and **Y= tionwagon LCS=inaon**

	Yj	T	I	O	N	W	A	G	O	N
Xi	0	0	0	0	0	0	0	0	0	0
I	0	0	1	1	1	1	1	1	1	1
N	0	0	1	1	2	2	2	2	2	2
N	0	0	1	1	2	2	2	2	2	3
O	0	0	1	2	2	2	2	2	3	3
V	0	0	1	2	2	2	2	2	3	3
A	0	0	1	2	2	2	3	3	3	3
T	0	1	1	2	2	2	3	3	3	3
I	0	1	2	2	2	2	3	3	3	3
O	0	1	2	3	3	3	3	3	4	4
N	0	1	2	3	4	4	4	4	4	5

Similarity features of Text

Algorithm	Comparison by	Case Sensitive	Sequence Matters?	Result
Cosine Similarity / Distance	word	✓	✗	Number (0 to 1)
Hamming Distance (Same length input strings only)	character	✓	✓	Count of substitutions
Jaccard Similarity / Distance	character	✓	✗	Number (0 to 1)
Jaro Winkler Similarity / distance	character	✓	✓	Number (0 to 1)
Levenshtein Distance	character	✓	✓	Count of edits
Longest Common Subsequence	character	✓	✓	Common String

Bregman Divergence

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^T \nabla \varphi(\mathbf{y})$$

- **Bregman distance** **is a measure of distance between two points, defined in terms of a strictly convex function.** When the points are interpreted as probability distributions – notably as either values of the parameter of a parametric model or as a data set of observed values – the resulting distance is calculated using Bregman divergence.
- Generalize squared Euclidean distance to a class of distances that all share similar properties
- It's a family of **proximity functions that have common properties**. It represents loss or distortion functions.
- **What is a loss function?**
- Let x and y be two points, where y is regarded as the original point and x is some distortion or approximation of it. x may be a point that was generated by adding random noise to y. The **goal** is to **measure the resulting distortion or loss** that results if **y is approximated by x.** Of course, the more similar x and y are, the smaller the loss or distortion.
- Bregman divergences can be used as dissimilarity functions. **Bregman divergence (loss function)** D(x, y) generated by that function is given by the following equation: **$D(x, y) = \phi(x) - \phi(y) - \langle \nabla(\phi(y)), (x - y) \rangle$**
  - **Φ**- a strictly convex function
  - **$\nabla\phi(y)$** is the gradient of ϕ evaluated at y
  - **x–y** is the vector difference between x and y
  - **$\langle \nabla\phi(y), (x - y) \rangle$** is the inner product between $\nabla\phi(x)$ and $(x - y)$. For points in Euclidean space, the inner product is just the dot product.
- $d^2(x,y) = \|x-y\|^2 = \langle x-y, x-y \rangle = \|x\|^2 - \|y\|^2 - \langle 2y, x-y \rangle$. Derived using the euclidean distance formula.

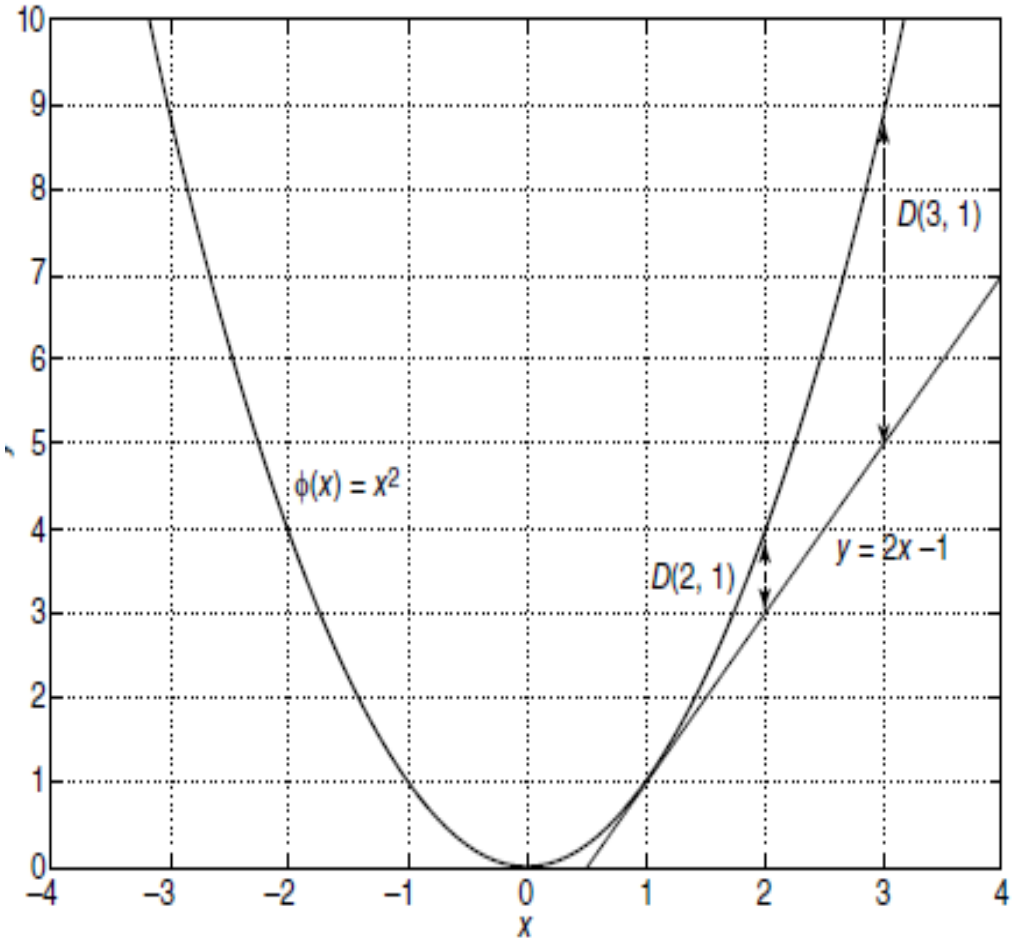


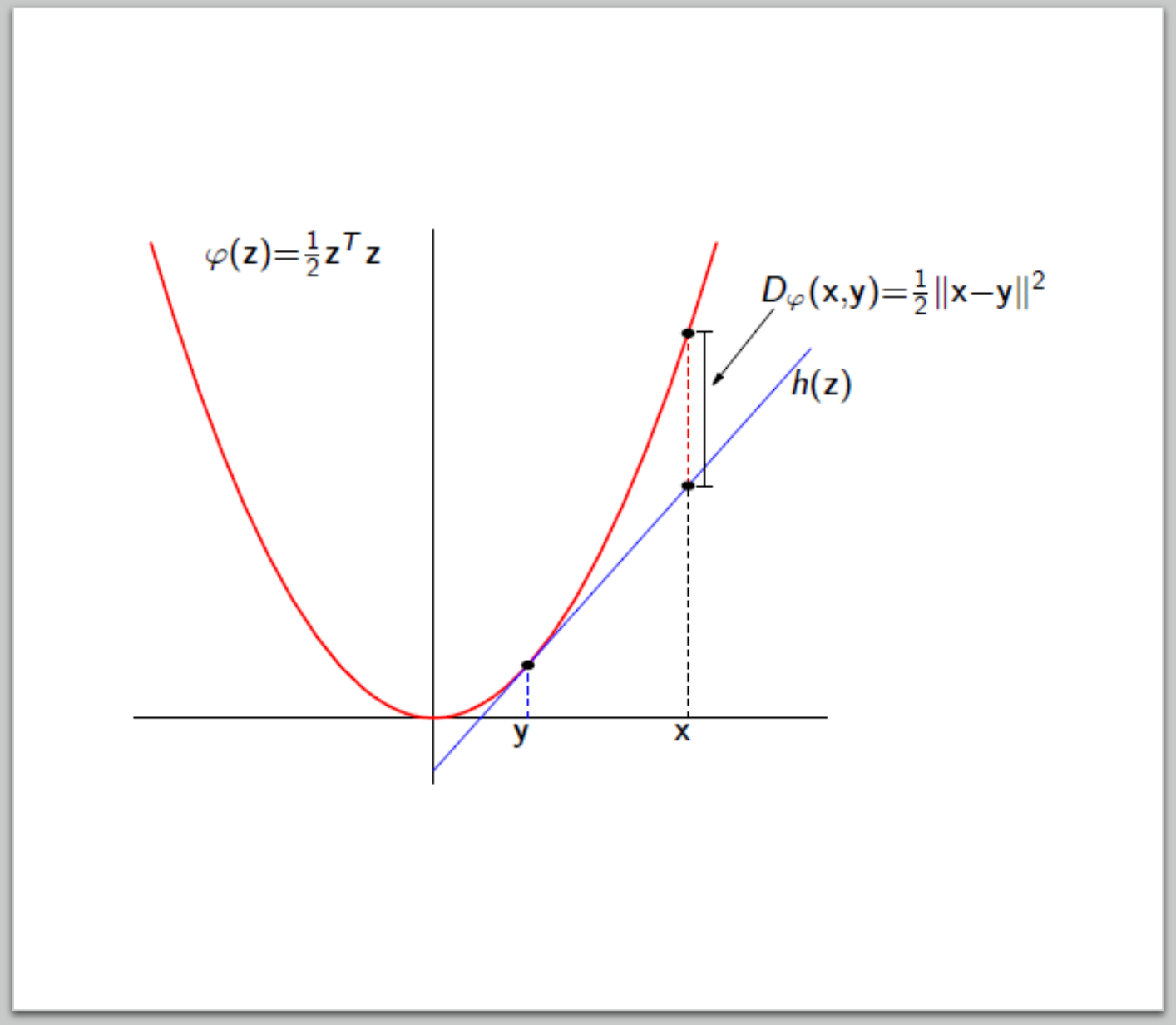
Figure 2.18. Illustration of Bregman divergence.

Bregman Divergence

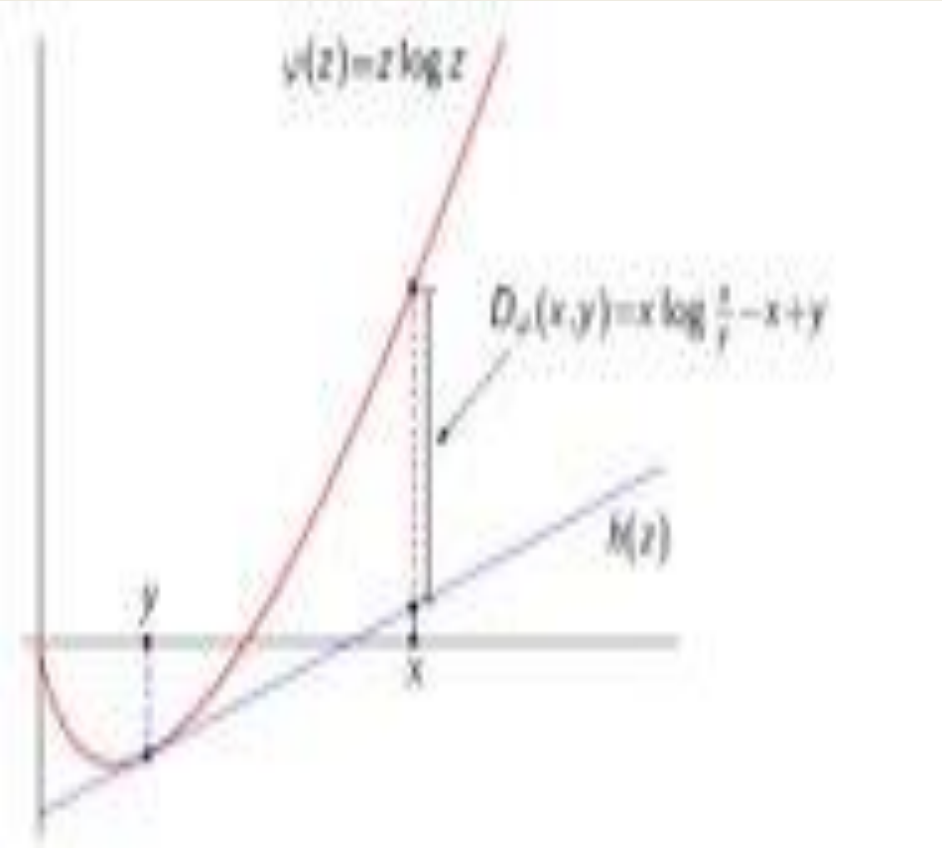
- Let x and y be real numbers and $\phi(t)$ be the real valued function, $\phi(t) = t^2$. The gradient reduces to the derivative and the dot product reduces to multiplication. $D(x, y) = x^2 - y^2 - 2y(x - y) = (x - y)^2$
- The graph for this example, with y = 1, is shown for two values of x: x = 2 and x = 3..

Types of Bregman Divergence

1. Squared Euclidean distance is a Bregman divergence



Kullback-Leibler (KL) divergence



- Relative Entropy (or KL-divergence) is another Bregman divergence. Bregman divergence using the convex function $fKL(p)=\sum n_i=1 p_i \log p_i$

$$D_\varphi(x,y)=x\log\frac{x}{y}-x+y$$

$$D_\varphi(\mathbf{x},\mathbf{y})=\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2$$

$$D_\varphi(x,y)=\frac{x}{y}-\log\frac{x}{y}-1$$

Properties Of Bregman Divergence

- **Non-negativity**:$D_F(p,q) \geq 0$ for all p, q. This is a consequence of the convexity of F.
- **Convexity**:$D_F(p,q)$ is convex in its first argument, but not necessarily in the second argument
- **Linearity**
- **Mean as minimizer**: A key result about Bregman divergences is that, given a random vector, the mean vector minimizes the expected Bregman divergence from the random vector

Thank You