Measuring Similarity/Dissimilarity

Similarity/Dissimilarity

Briefly outline how to compute the *dissimilarity* between objects described by the following types of variables:

- Numerical (interval-scaled) variables

Similarity/Dissimilarity

Briefly outline how to compute the *dissimilarity* between objects described by the following types of variables:

Numerical (interval-scaled) variables

Use Euclidean distance or Manhattan distance. Euclidean distance is defined as:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2}$$

The Manhattan distance is defined as:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|$$

A = (2, 4)
B = (1, 3)

Euclidean distance = math.sqrt(pow(2-1, 2) + pow(4-3, 2))

A = (2, 7, 5)
B = (4, 8, 9)

ed = math.sqrt(pow(2-4, 2) + pow(7-8, 2) + pow(5-9, 2))

Euclidean distance is the direct and straight line distance.

Whereas:

Manhattan distance is indirect and calculated along x and y axis.

Euclidean Distance

```
from scipy.spatial import distance
```

```
import math
A = (2, 4)
B = (1, 3)
print(math.dist(A, B))
```

```
1.4142135623730951
```

```
A = (2, 7, 5)
B = (4, 8, 9)
print(math.dist(A, B))
```

```
4.58257569495584
```

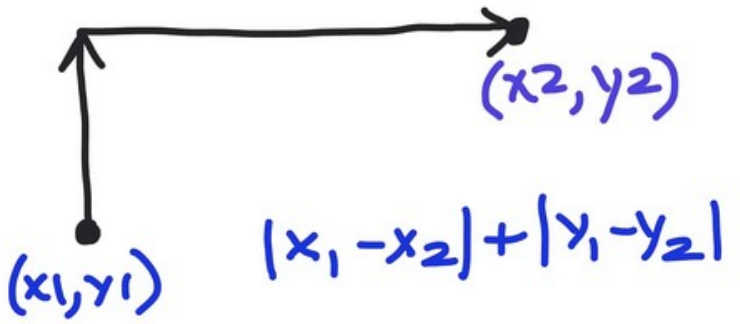
```
import numpy as np
p1 = np.array((1,2,3))
p2 = np.array((3,2,1))
sq = np.sum(np.square(p1 - p2))
print(np.sqrt(sq))
```

```
2.8284271247461903
```

```
distance.euclidean(p1, p2)
```

```
2.8284271247461903
```

Manhattan Distance




```
def manhattan(a, b):
    return sum(abs(val1-val2) for val1, val2 in zip(a,b))

#define vectors
A = [2, 4, 4, 6]
B = [5, 5, 7, 8]

#calculate Manhattan distance between vectors
manhattan(A, B)
```

```
9
```

```
A = (2, 4)
B = (1, 3)
A = np.array(A)
B = np.array(B)
print(np.sum(np.abs(A - B)))
```

```
2
```

```
A = (2, 7, 5)
B = (4, 8, 9)
A = np.array(A)
B = np.array(B)
print(np.sum(np.abs(A - B)))
```

```
7
```

```python
import numpy as np
p1 = np.array((1,2,3))
p2 = np.array((3,2,1))
print(np.sum(np.abs(p1 - p2)))
```

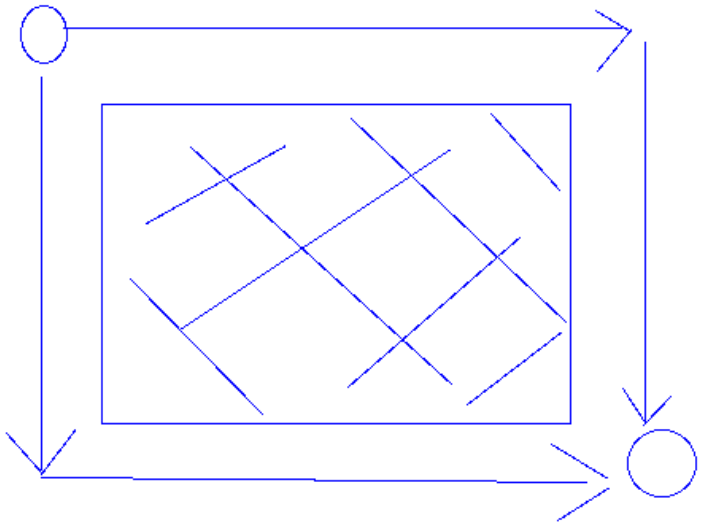
```
4
```

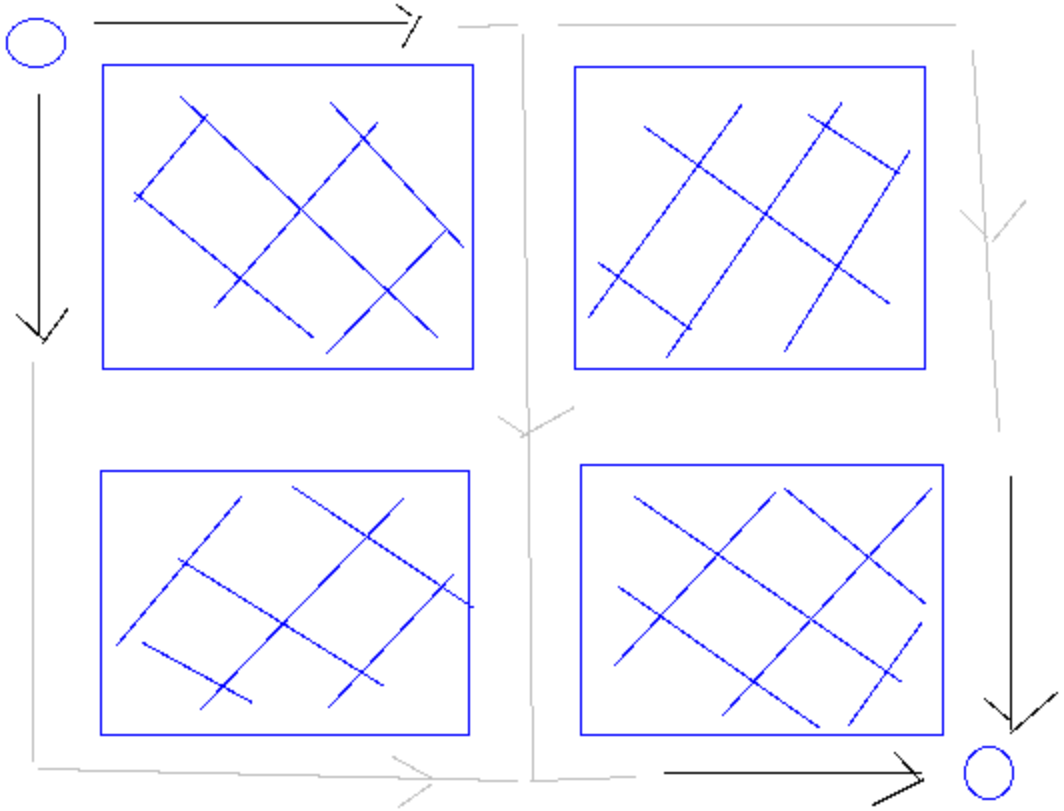
```python
from scipy.spatial import distance
print(distance.cityblock(p1, p2))
```

```
4
```

Manhattan Distance

Manhattan Distance is inspired from a traveler going from one point to another in the city of Manhattan.





Manhattan distance in real world

Euclidean distance: Straight line distance between two points.

Manhattan distance: is cityblock distance that is you would go from place to place in car.

Minkowski Distance

Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):
Compute the ***Minkowski*** *distance* between the two objects, using $p = 3$.

In plain English: kth root of sum of kth powers of absolute value of differences.

$$= \quad (|22-20|^3 + |1-0|^3 + |42-36|^3 + |10-8|^3)^{1/3} = 6.15$$

For manhattan distance, p = 1 in Minkowski dist.
For euclidean distance, p = 2 in Minkowski dist.
Manhattan and Euclidean distances are special cases of Minkowski.

```
from scipy.spatial import distance
import numpy as np
u = np.array((1,2,3))
v = np.array((3,2,1))
```

```
distance.minkowski(u, v, p=3)
```

```
2.5198420997897464
```

```
u = np.array((22, 1, 42, 10))
v = np.array((20, 0, 36, 8))
distance.minkowski(u, v, p=3)
```

```
6.153449493663682
```

Similarity/Dissimilarity

Briefly outline how to compute the *dissimilarity* between objects described by:

Asymmetric binary variables

If all binary attributes have the same weight then they are symmetric. Let's say we have the contingency table:

object i \ object j	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

If the binary attributes are asymmetric, Jaccard coefficient is often used:
For cell (i=1, j=1) representing #(object I = 1 and object J = 1):

J = q / (q + r + s)

J = # elements in intersection / # elements in the union

Jaccard similarity (aka coefficient or score) ranges from 0 to 1.
Jaccard dissimilarity (aka distance) = 1 – Jaccard Similarity

Cart ID	Apple	Tomato	Eggs	Milk	Coffee	Sugar
A	1	0	0	1	1	1
B	0	0	1	1	1	0
C	1	0	0	1	0	1
D	0	1	1	0	1	0

Cart ID	A	B	C	D
Apple	1	0	1	0
Tomato	0	0	0	1
Eggs	0	1	0	1
Milk	1	1	1	0
Coffee	1	1	0	1
Sugar	1	0	1	0

The reason why ‘t’ is not considered in Jaccard coeff. Is because, while taking an example of shopping cart, it would not make same sense to count the number of times that were missing from both the cart I and J. If we count the number of items that were missing in both carts, it would increase the similarity.
If it is given the attributes are binary and assymmetric, t is not counted. If it is symmetric, then formula for J may include ‘t’ in the numerator and denominator:
$J = (q + t) / (q + r + s + t)$

Suppose we have two vectors, **A** and **B**, each with n binary attributes.

In this case, the Jaccard similarity (index) can be calculated as:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

and Jaccard distance can be calculated as:

$$d_J = \frac{M_{01} + M_{10}}{M_{01} + M_{10} + M_{11}} = 1 - J$$

where:

- M_{11} is the total numbers of attributes, for which both **A** and **B** have 1
- M_{01} is the total numbers of attributes, for which **A** has 0 and **B** has 1
- M_{10} is the total numbers of attributes, for which **A** has 1 and **B** has 0
- M_{00} is the total numbers of attributes, for which both **A** and **B** have 0

and:

$$M_{11} + M_{01} + M_{10} + M_{00} = n$$

	Apple	Tomato	Eggs	Milk	Coffee	Sugar
A	1	0	0	1	1	1
B	0	0	1	1	1	0

Step 1:

We will first need to find the total number for attributes for each 𝕚

	Count	Explanation
M_{11}	2	Both customers bought coffee and milk
M_{01}	1	Customer A didn't buy eggs, whereas Customer B bought eggs
M_{10}	2	Customer B didn't buy apple and sugar, whereas Customer 1 bought apple and sugar
M_{00}	1	Neither of customers bought tomato

We can validate the groups by summing up the counts. it should be equal to 6 which is the 𝕟number of attributes (products):

$$M_{11} + M_{01} + M_{10} + M_{00} = 2 + 1 + 2 + 1 = 6$$

Step 2:

Since we have all the required inputs, we can now calculate the Jaccard similarity:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} = \frac{2}{1+2+2} = \frac{2}{5} = 0.4$$

And Jaccard distance:

$$d_J = \frac{M_{01} + M_{10}}{M_{01} + M_{10} + M_{11}} = \frac{1+2}{1+2+2} = \frac{3}{5} = 0.6$$

Similarity/Dissimilarity

Briefly outline how to compute the *dissimilarity* between objects described by the following types of variables:

Categorical variables

A categorical variable is a generalization of the binary variable in that it can take on more than two states.

The dissimilarity between two objects *i* and *j* can be computed as:

$$d(i,j) = \frac{p-m}{p}$$

where *m* is the number of *matches* (i.e., the number of variables for which *i* and *j* are in the same state), and *p* is the total number of variables.

Similarity/Dissimilarity

Briefly outline how to compute the *dissimilarity* between objects described by the following types of variables:

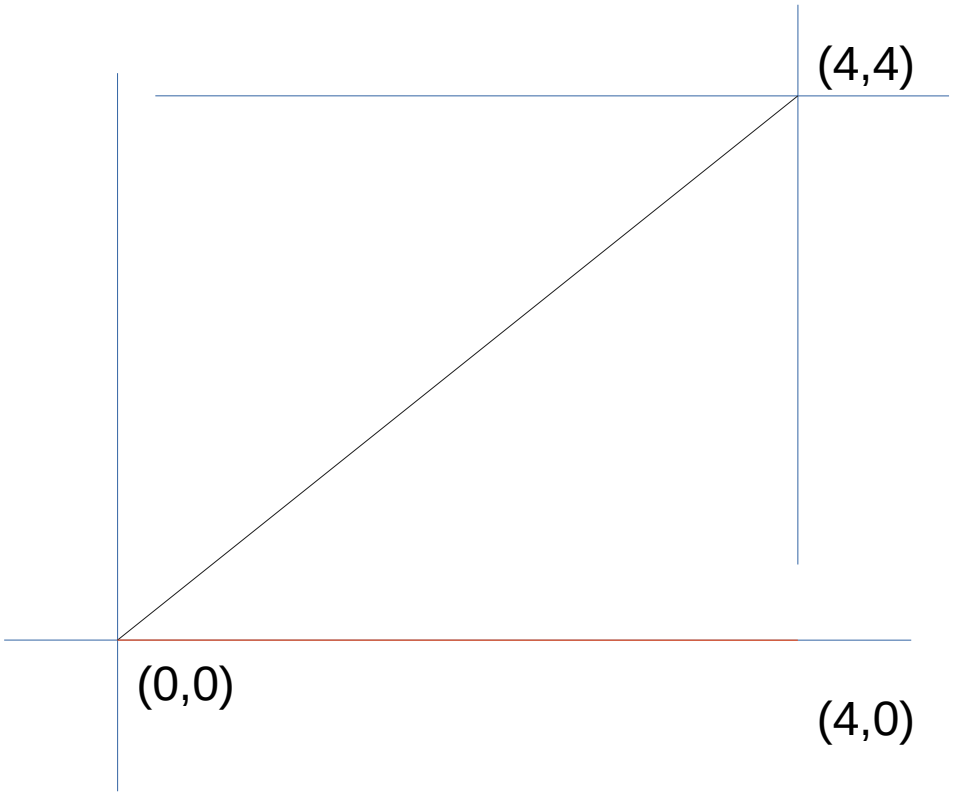
Text

$$s(x, y) = \frac{x^t \cdot y}{||x||||y||}$$

This is equal to dot-product of the corresponding features divided by the magnitude. Geometrically, it is equal to the cosine of angle between two lines.

Cosine Similarity

A = (4, 4)
B = (4, 0)



Cosine similarity between A and B is: cos(theta)

Note: It is very easy to visualize it in 2D.

Formula wise: Cosine similarity = A.B / |A| * |B|
This formula has come after rearranging the dot product formula:
A.B = |A| * |B| * cos(theta)

Cosine Similarity

Find cosine similarity between the following two sentences:

IIT Delhi course on data analytics IIT
IIT Delhi course on data analytics

Term: tf for d1, tf for d2

IIT	2	1
Delhi	1	1
Course	1	1
On	1	1
Data	1	1
Analytics	1	1

d1.d2 = 2.1 + 1.1 + 1.1 + 1.1 + 1.1 + 1.1 = 7
|d1| = sqrt(2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2) = sqrt(9)
|d2| = sqrt(1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2) = sqrt(6)

Cosine similarity = d1.d2 / |d1| * |d2| = 7 / sqrt(9*6) = 0.93

Cosine Similarity

Sent 1: I bought a new car today.
Sent 2: I already have a new car.
Step 1: Create vectors.

$$s(x, y) = \frac{x^t \cdot y}{||x||||y||}$$

	Vector 1	Vector 2	Product of components
I	1	1	1
bought	1	0	0
a	1	1	1
new	1	1	1
car	1	1	1
today	1	0	0
already	0	1	0
have	0	1	0

Cosine Similarity

V1 = 1,1,1,1,1,1,0,0
V2 = 1,0,1,1,1,0,1,1
v1.v2 = 1 + 0 + 1 + 1 + 1 + 0 + 0 + 0 = 4
|v1| = math.sqrt(1**2 + 1**2 + 1**2 + 1**2 + 1**2 + 1**2 + 0 + 0) = 2.45
|v2| = math.sqrt(1**2 + 0 + 1**2 + 1**2 + 1**2 + 0 + 1**2 + 1**2) = 2.45

Cosine sim = 4 / (2.45 * 2.45)
= 0.66

Cosine Similarity

Cosine Similarity: It is a bag-of-words based model i.e., order of words does not matter.

Sent1: I bought a new car today.
Sent2: I haven’t bought a new car today.

Step 1: Creating vectors

Cosine similarity is based on whether same words have been used in two sentences.

“How similar are two sentences in terms of the words (irrespective of their meaning) used in them?”

Cosine Similarity

Find cosine similarity between the following two sentences:
Brand new course on data analytics
Test-1 is scheduled later this month

Cosine similarity = 0.0

Thank You!