Lab 07 ¶

Similarity measures

1. Given $x = (15,20,9,14)$, and y = (14,18,7,10), find Manhattan, Euclidean, L_3-norm, L_∞-norm, distances between x and y.

   What do you observe as degree of norm increases in *Minkowski* distance?

   Figure out one application example for each of this measure.

In [1]:

```
import numpy as np
x=np.array([15,20,9,14])
y=np.array([14,18,7,10])
```

In [2]:

```
x
```

Out[2]:

```
array([15, 20,  9, 14])
```

In [3]:

```
y
```

Out[3]:

```
array([14, 18,  7, 10])
```

Manhattan Distance

In [4]:

```
# |x1-y1|+|x2-y2|+|x3-y3|+|x4-y4|
m=0
D=np.arange(len(x))
for i in range(len(x)):
    D[i]=abs(x[i]-y[i])
    m=m+D[i]
```

In [5]:

```
m # Manhattan Distance
```

Out[5]:

```
9
```

Minkowski Distance

$$distance(x,y) = \left(\sum_{i=1}^{n}(|x_i - y_i|)^r\right)^{\frac{1}{r}}$$

In [6]:

```
def Lnorm(X,Y,r):
    sum=0
    for i in range(len(X)):
        sum=sum+(abs(X[i]-Y[i]))**r
    return pow(sum,1/r)
```

In [7]:

```
Lnorm(x,y,1) # -> Manhattan
```

Out[7]:

```
9.0
```

In [8]:

```
Lnorm(x,y,2) #-> Eucledian
```

Out[8]:

```
5.0
```

In [9]:

```
Lnorm(x,y,3) # -> L3 norm
```

Out[9]:

```
4.3267487109222245
```

As Degree of Norm Increases-> Minkowski Distance Decreases

2. What is your interpretation of Hamming distance, Jaccard similarity, and Cosine similarity? Enumerate suitability and unsuitability of each of these measures.

   Compute these measure for following two binary vectors x, and y, find out Hamming distance, Jaccard similarity, and Cosine similarity.

   x: 0101010001
   y: 0100011000

1. In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

2. Jaccard Similarity is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

3. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

x: 0101010001
y: 0100011000

In [10]:

```
# Hamming Distance
d=0
x=(0,1,0,1,0,1,0,0,0,1)
y=(0,1,0,0,0,1,1,0,0,0)
for i in range(len(x)):
    if x[i]!=y[i]:
        d=d+1
print('Hamming Distance is:',d)
```

```
Hamming Distance is: 3
```

- **Jaccard Coefficient (JC)** is used for such cases, and expressed as-

  JC = number of 11 matches / number of not-both-zero attributes values

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

In [11]:

```
# Jaccard Index
m11=m01=m10=0
for i in range(len(x)):
    if x[i]==1 and y[i]==1:
        m11=m11+1
    elif x[i]==0 and y[i]==1:
        m01=m01+1
    elif x[i]==1 and y[i]==0:
        m10=m10+1

J=m11/(m11+m10+m01)
print('Jaccard Coefficient=',J)
```

```
Jaccard Coefficient= 0.4
```

In [12]:

```
# Cosine Similarity
x=np.asarray(x)
y=np.asarray(y)
```

In [13]:

```
x
```

Out[13]:

```
array([0, 1, 0, 1, 0, 1, 0, 0, 0, 1])
```

In [14]:

```
y
```

Out[14]:

```
array([0, 1, 0, 0, 0, 1, 1, 0, 0, 0])
```

In [15]:

```
modx=Lnorm(x,np.zeros(len(x)),2)
mody=Lnorm(y,np.zeros(len(y)),2)
print(modx,' ',mody)
```

```
2.0   1.7320508075688772
```

In [16]:

```
cs=(np.dot(x,y))/modx*mody
print('Cosine Similarity is:',cs)
```

```
Cosine Similarity is: 1.7320508075688772
```

a. Suppose x and y are two items. Each dimension is a feature. The values 1/0 indicate that an item has or does not have that feature. Which measure you think is suitable for measuring the distance between items x and y? Justify your answer.

b. Suppose x and y are two documents, and each dimension is a *word* (or term) count (vectors are no more binary in that case). Which measure you think is appropriate for measuring similarity between document x and y? Justify your answer.

1. For a, I think Jaccard Similarity is better, because it is the case of Recommendation where two items are compared based on their similarity.

Jaccard Coefficient ignores (0,0) pairs which are of no use.

2. For b, I think Cosine Similarity is better because when dot product is taken, the angle between 2 vectors is also taken into consideration . Whereas Eucledian distance would increase if we double the vectors, Cosine Similarity would still be same.

3. Compute the Jaccard similarities of each pair of the following three sets: {Health, Patient, Politics, Shah}, {Health, Gujarat, Tiger, Politics}, and {Hardik, Patel, Politics}.

   Explain three applications (discussed in lecture) of Jaccard Similarity of Documents.

In [17]:

```
A=['Health','Patient','Politics','Shah']
B=['Health','Gujarat','Tiger','Politics']
C=['Hardik','Patel','Politics']
print(A,B,C)
```

```
['Health', 'Patient', 'Politics', 'Shah'] ['Health', 'Gujarat', 'Tiger',
'Politics'] ['Hardik', 'Patel', 'Politics']
```

In [18]:

```
from functools import reduce
a=reduce(np.intersect1d, (A,B,C))
a
```

Out[18]:

```
array(['Politics'], dtype='<U8')
```

In [19]:

```
intersect_all=len(a)
b=reduce(np.union1d, (A,B,C))
b
```

Out[19]:

```
array(['Gujarat', 'Hardik', 'Health', 'Patel', 'Patient', 'Politics',
       'Shah', 'Tiger'], dtype='<U8')
```

In [20]:

```
union_all=len(b)
```

In [21]:

```
print('Jaccard Similarity among A,B,C is:',intersect_all/union_all)
```

```
Jaccard Similarity among A,B,C is: 0.125
```

Three Applications of Jaccard Similarity of Documents:

1. Plagiarism: suppose we want check a document for Plagiarism. The problem can be solved by seeing the document in question is “similar” to any of document(s) in the repository.

2. A web crawler must be able to detect if the document is mirror of other document.

3. A news aggregator detecting that news article is same, but coming from different sources.

4. Below are given two student objects. Compute the similarity between Pankaj and Mitesh.

	Pankaj	Mitesh
Member Programming Club	Yes	Yes
Sportsman	Yes	No
Watches average number of movies in a month	5	10
Early Riser	Yes	No
Average Attendance in Lectures(%)	40	60
CPI	7.8	6.9

In [35]:

```
import pandas as pd
data=pd.DataFrame(columns=['Pankaj','Mithesh'])
rows=['MPC','Sportsman','Watches','Earlyriser','Att','CPI']
```

Finding out Cosine Similarity

Denote yes->1 No->0

In [60]:

```
Pankaj=np.array([1,1,5,1,40,7.8])
Mithesh=np.array([1,0,10,0,60,6.9])
```

In [61]:

```
Pankaj
```

Out[61]:

```
array([ 1. ,  1. ,  5. ,  1. , 40. ,  7.8])
```

In [62]:

```
Mithesh
```

Out[62]:

```
array([ 1. ,  0. , 10. ,  0. , 60. ,  6.9])
```

In [63]:

```
modx=Lnorm(Pankaj,np.zeros(len(x)),2)
mody=Lnorm(Mithesh,np.zeros(len(y)),2)
```

In [64]:

```
modx
```

Out[64]:

```
41.095498536944405
```

In [65]:

```
mody
```

Out[65]:

```
61.22589321520757
```

In [66]:

```
np.dot(Pankaj,Mithesh)
```

Out[66]:

```
2504.82
```

In [67]:

```
cs=(np.dot(Pankaj,Mithesh))/(modx*mody)
print('Cosine Similarity between Pankaj and Mithesh is:',cs)
```

```
Cosine Similarity between Pankaj and Mithesh is: 0.9955134666977309
```

In [ ]:

```

```