Worksheet 04

Name: Muniruddin Ahmed Siddiqui
UID: U42483225

Topics

- Distance & Similarity

Distance & Similarity

Part 1

a) In the minkowski distance, describe what the parameters p and d are.

- p is an arbitrary parameter that must be >= 1
- d is the number of dimensions x and y have

b) In your own words describe the difference between the Euclidean distance and the Manhattan distance.

The main difference between these distance metrics is how they measure distances; when p=2 it is Euclidean distance and when p=1 it is Manhattan distance. Euclidean distance calculates the shortest straight-line distance, while Manhattan distance calculates the distance along a grid-like path.

Consider A = (0, 0) and B = (1, 1). When:

- p = 1, d(A, B) = 2
- p = 2, d(A, B) = $\sqrt{2} = 1.41$
- p = 3, d(A, B) = $2^{1/3} = 1.26$
- p = 4, d(A, B) = $2^{1/4} = 1.19$

c) Describe what you think distance would look like when p is very large.

I think the distance would approach 1 as p grows larger and larger.

d) Is the minkowski distance still a distance function when p < 1? Expain why / why not.

No. Counter-example:
Suppose A(0,0), B(1,0), C(0,1). Then D(B,A) = D(A,C) = 1 and D(B,C) = $2^{1/p}$. If p < 1, then 1/p > 1. So D(B,C) > D(B,A) + D (A,C). However, this violates the triangular inequality, so it would not be a distance function.

e) when would you use cosine similarity over the euclidan distance?

When direction matters more than magnitude, e.g: a dataset like a document and an abstract describing that document would yield a large Euclidean distance but a small cosine distance as they both talk about the same topic.

f) what does the jaccard distance account for that the manhattan distance doesn't?

The Jaccard distance takes into account the overlap between the datasets.

Part 2

Consider the following two sentences:

```
s1 = "hello my name is Alice"
s2 = "hello my name is Bob"
```

using the union of words from both sentences, we can represent each sentence as a vector. Each element of the vector represents the presence or absence of the word at that index.

In this example, the union of words is ("hello", "my", "name", "is", "Alice", "Bob") so we can represent the above sentences as such:

```
v1 = [1,    1, 1,   1, 1,    0]
#     hello my name is Alice
v2 = [1,    1, 1,   1, 0, 1]
#     hello my name is    Bob
```

Programmatically, we can do the following:

```
corpus = [s1, s2]
all_words = list(set([item for x in corpus for item in x.split()]))
print(all_words)
v1 = [1 if x in s1 else 0 for x in all_words]
print(v1)
```

```
['hello', 'Alice', 'name', 'Bob', 'my', 'is']
[1, 1, 1, 0, 1, 1]
```

Let's add a new sentence to our corpus:

```
s3 = "hi my name is Claude"
corpus.append(s3)
```

a) What is the new union of words used to represent s1, s2, and s3?

```
corpus = [s1, s2, s3]
all_words = list(set([item for x in corpus for item in x.split()]))
print(all_words)
```

```
['hello', 'Alice', 'name', 'Bob', 'my', 'is', 'hi', 'Claude']
```

b) Represent s1, s2, and s3 as vectors as above, using this new set of words.

```
print(all_words)
v1 = [1 if x in s1 else 0 for x in all_words]
print(v1)
v2 = [1 if x in s2 else 0 for x in all_words]
print(v2)
v3 = [1 if x in s3 else 0 for x in all_words]
print(v3)
```

```
['hello', 'Alice', 'name', 'Bob', 'my', 'is', 'hi', 'Claude']
[1, 1, 1, 0, 1, 1, 0, 0]
[1, 0, 1, 1, 1, 1, 0, 0]
[0, 0, 1, 0, 1, 1, 1, 1]
```

c) Write a function that computes the manhattan distance between two vectors. Which pair of vectors are the most similar under that distance function?

```
def minkowski_dist(x, y, p):
    if p < 1:
        raise ValueError("p must be greater than 1.")
    if len(x) != len(y):
        raise ValueError("x and y must be in the same dimensional
space.")
    res = 0
    for i in range(len(x)):
        res += abs(x[i] - y[i]) ** p
    return res ** (1 / p)

def manhattan_dist(x, y):
    return minkowski_dist(x, y, 1)

print(manhattan_dist(v1, v2))
```

```
2.0
```

d) Create a matrix of all these vectors (row major) and add the following sentences in vector form:

- "hi Alice"
- "hello Claude"
- "Bob my name is Claude"
- "hi Claude my name is Alice"
- "hello Bob"

```
corpus = ["hi Alice", "hello Claude", "Bob my name is Claude", "hi
Claude my name is Alice", "hello Bob"]
all_words = list(set([item for x in corpus for item in x.split()]))
print(all_words)

matrix = [[1 if word in sentence else 0 for word in all_words] for
```

```
sentence in corpus]

for vector in matrix:
    print(vector)
```

```
['hello', 'Alice', 'Bob', 'name', 'my', 'is', 'hi', 'Claude']
[0, 1, 0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 1, 1, 1, 1, 0, 1]
[0, 1, 0, 1, 1, 1, 1, 1]
[1, 0, 1, 0, 0, 0, 0, 0]
```

e) How many rows and columns does this matrix have?

```
rows = len(matrix)
cols = len(matrix[0])

print("rows:", rows)
print("columns", cols)
```

```
rows: 5
columns 8
```

f) When using the Manhattan distance, which two sentences are the most similar?

```
minimum = manhattan_dist(matrix[0], matrix[1])
vectors = (0, 1)
for i in range(1, len(matrix)):
    for j in range(i, len(matrix)):
        if i != j:
            new_min = manhattan_dist(matrix[i], matrix[j])
            if minimum > new_min:
                minimum = new_min
                vectors = (i, j)
print(vectors)
print(matrix[vectors[0]], matrix[vectors[1]])
```

```
(1, 4)
[1, 0, 0, 0, 0, 0, 0, 1] [1, 0, 1, 0, 0, 0, 0, 0]
```