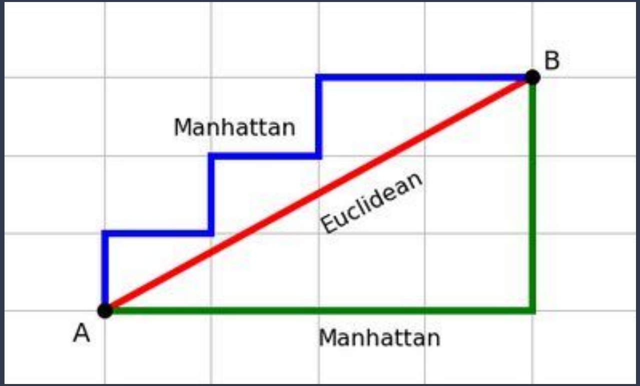
Distances in unsupervised learning

Group ARO - Andreas, Owais & Rasmus
AI, OLA 2 - 15/03/24



Manhattan Distance – Theory

"Manhattan Distance is the sum of absolute differences between points across all the dimensions."

It is the total sum of the difference between the x-coordinates and y-coordinates

In multidimensions (calculated by taking the sum of distances between (x, y)-coordinates):

$$Distance(x, y) = \sum_{i=1}^{n} |x_i - y_i| = |x_2 - x_1| + |y_2 - y_1| + ... + |x_n - x_1| + |y_n - y_1|$$

Where n = number of dimensions

It is computationally less expensive to compute compared to Euclidean distance because it doesn't involve taking square roots.

Manhattan Distance – Example with 2 dimensions

$$Distance(x, y) = \sum_{i=1}^{n} |x_i - y_i| = |x_2 - x_1| + |y_2 - y_1| + \ldots + |x_n - x_1| + |y_n - y_1|$$

```
# Example train and test datasets
train_data = np.array([[1, 2], [3, 5], [7, 8], [10, 12]])
test_data = np.array([[2, 4], [5, 7]])

# Calculate Manhattan distances between train and test datasets
manhattan_distances_train_test = manhattan_distances(train_data, test_data)

print("Manhattan distances between train and test datasets:")
print(manhattan_distances_train_test)
```

```
Manhattan distances between train and test datasets:
[[ 3.  9.]
 [ 2.  4.]
 [ 9.  3.]
 [16. 10.]]
```

For each of the data points in the *train_data* dataset the manhattan distance is calculated to the test_data points.

So the distance between the first data point in the train_data set **[1, 2]** and the first data point in test_data set **[2, 4]** is = **|x2-x1|+|y1-y1|=|2-1|+|4-2| = |1|+|2| = 3**

And the distance between the first data point in the train_data set **[1, 2]** and the second test_data set **[5, 7]** = **|5-1|+|7-2| = |4|+|5| = 9**

Thereby the result for the first data point in the train_data **[1, 2]** to the data points in the test_data is **[3. 9.]** .

Same is now calculated for the rest of the data points in train_data.

Manhattan Distance – Example with more than 2 dimensions

$$Distance(x,y) = \sum_{i=1}^{n} |x_i - y_i| = |x_2 - x_1| + |y_2 - y_1| + \ldots + |x_n - x_1| + |y_n - y_1|$$

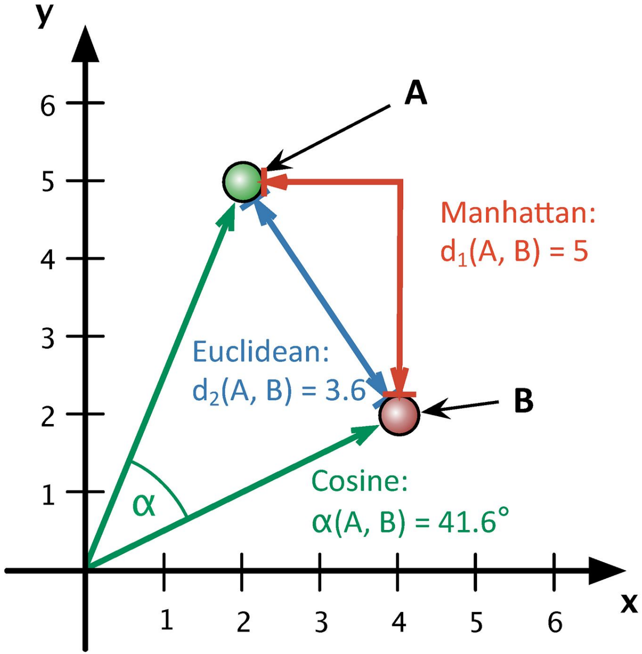
```python
# Example train and test datasets with more than 2 dimensions
train_data = np.array([[1, 2, 3], [3, 5, 2], [7, 8, 1], [10, 12, 4]])
test_data = np.array([[2, 4, 1], [5, 7, 3], [4, 5, 2]])

# Calculate Manhattan distances between train and test datasets
manhattan_distances_train_test = manhattan_distances(train_data, test_data)

print("Manhattan distances between train and test datasets:")
print(manhattan_distances_train_test)
```

```
Manhattan distances between train and test datasets:
[[ 5.  9.  7.]
 [ 3.  5.  1.]
 [ 9.  5.  7.]
 [19. 11. 15.]]
```

Manhattan Distance – Visualization



Manhattan Distance - Code

Check `ManhattanDistances.ipynb` in the repository

Euclidean

Theory:
The Euclidean distance gives the distance between two points, or the straight line distance. The distance can be calculated from coordinates using the Pythagorean theorem. It is occasionally called the Pythagorean distance.

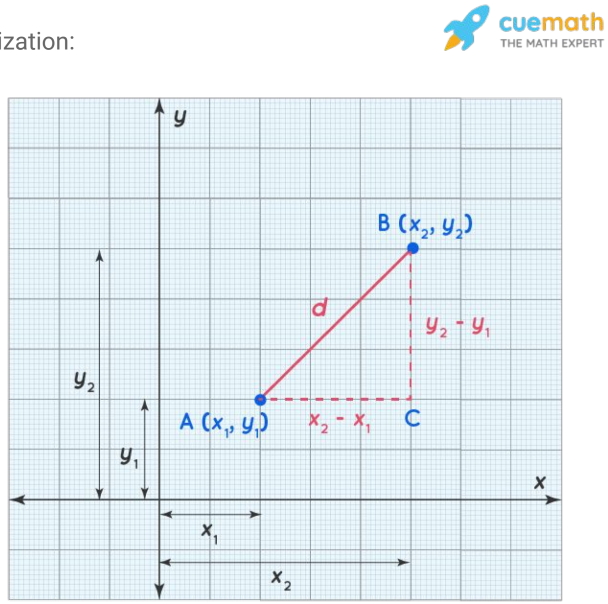
The formula is

$$AB^2 = AC^2 + BC^2$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

Visualization:



Euclidean Code Quick way

Using scipy's distance() or math.dist() to calculate distance in python:

```
point_1 = (1,2)
point_2 = (4,7)

from math import dist
dist_math = dist(point_1, point_2)

from scipy.spatial import distance
dist_scipy = distance.euclidian(point_1, point_2)

print(dist_scipy)
print(dist_math)

# Both will print around 5.83
```

Euclidean Code Own method using sum()

```
point_1 = (1,2)
point_2 = (4,7)

def euclidean_distance(pointA, pointB):
	differences = [pointA[x] - pointB[x] for x in range(len(pointA))]
	differences_squared = [difference ** 2 for difference in differences]
	sum_of_squares = sum(differences_squared)
	return sum_of_squares ** 0.5

print(euclidean_distance(point_1, point_2))

# Prints around 5.83
```

Hamming Distance

"The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different."

Waggener, Bill (1995). Pulse Code Modulation Techniques. Springer. p. 206. ISBN 978-0-442-01436-0.

Theory:

- metric for calculating the distance between two binary strings or vectors of the same length.
- the distance is the difference in characters between two strings.
- Richard Hamming. Made for error detections and error correction.

hund vs. fugl = 3

10110 vs. 10101 = 2

Code:

```
def hamming_distance_bytes(byte1, byte2):
    xor_result = byte1 ^ byte2
    distance = bin(xor_result).count('1')
    return distance

distance = hamming_distance_bytes(21, 22)
print(f"The Hamming distance is: {distance}")
```

```
The Hamming distance is: 2
```