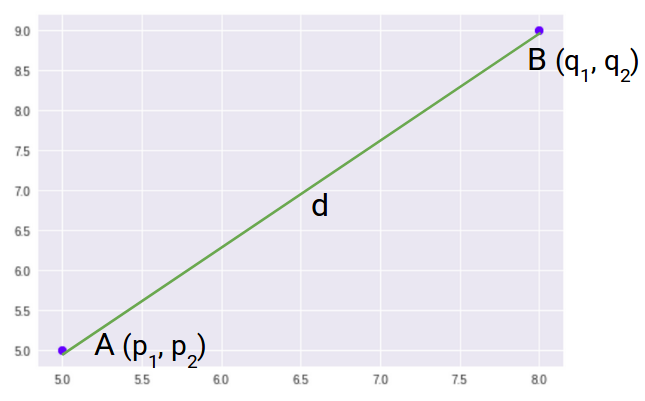
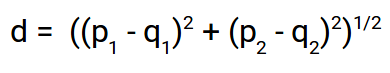
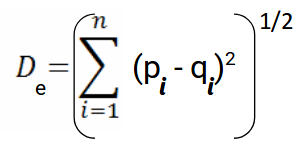
**EUCLIDEAN**

[Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance) calculates the distance between two real-valued vectors. Euclidean Distance represents the shortest distance between two points.





The generalized formula for an n-dimensional space is given as



n = number of dimensions

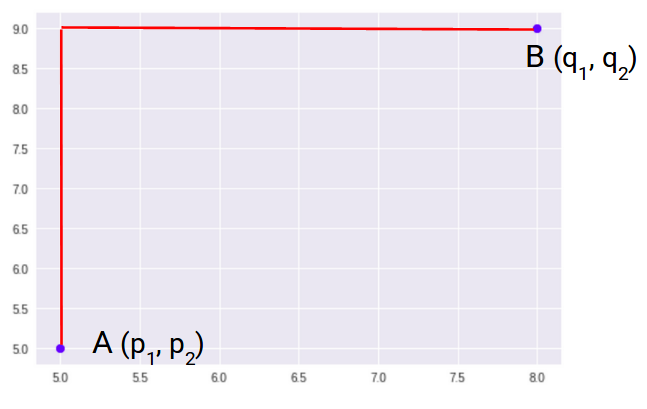
pi, qi = data points

euclidean distance in python

**MANHATTAN**

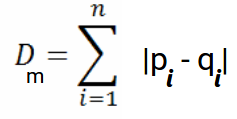
The [Manhattan distance](https://en.wikipedia.org/wiki/Taxicab_geometry), also called the Taxicab distance or the City Block distance calculates the distance between two real-valued vectors.

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



manhattan distance formula

The generalized formula for an n-dimensional space is given as



n = number of dimensions

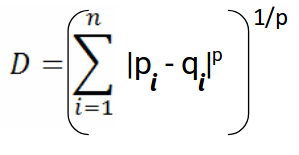
pi, qi = data points

manhattan distance in python

**MINKOWSKI**

[Minkowski distance](https://en.wikipedia.org/wiki/Minkowski_distance) calculates the distance between two real-valued vectors.

It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the order that allows different distance measures to be calculated.



minkowski distance in python

**HAMMING**

[Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance) calculates the distance between two binary vectors (binary strings or bit strings)

Hamming Distance measures the similarity between two strings of the same length. The Hamming Distance between two strings of the same length is the number of positions at which the corresponding characters are different.

For a one-hot encoded string, it might make more sense to summarize to the sum of the bit differences between the strings which will always be a 0 or 1.

red = [1, 0, 0]

green = [0, 1, 0]

blue = [0, 0, 1]

Hamming Distance = sum for i to N abs(v1[i] – v2[i])

For bit strings that may have many 1 bit, it is more common to calculate the average number of bit differences to give a hamming distance score between 0 (identical) and 1 (all different).

row1 = [0, 0, 0, 0, 0, 1]

row2 = [0, 0, 0, 0, 1, 0]

Hamming Distance = (sum for i to N abs(v1[i] – v2[i])) / N) = 1/3