Double-click (or enter) to edit

What Are Distance Metrics?

Distance metrics are a key part of several machine learning algorithms. These distance metrics are used in both supervised and unsupervised learning, generally to calculate the similarity between data points.

An effective distance metric improves the performance of our machine learning model, whether that's for classification tasks or clustering.

Types of Distance Metrics in Machine Learning

Euclidean Distance

Manhattan Distance

Minkowski Distance

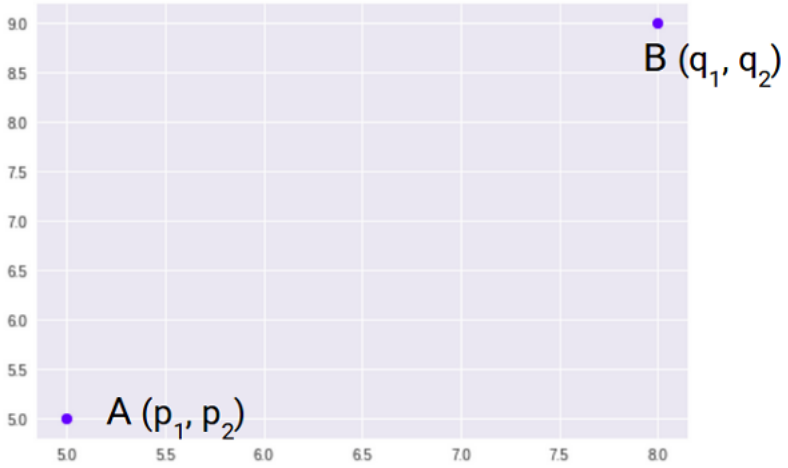
Hamming Distance

Euclidean distance is a widely used distance metric. It works on the principle of the Pythagoras theorem and signifies the shortest distance between two points. Euclidean distance is used in many machine learning algorithms as a default distance metric to measure the similarity between two recorded observations.

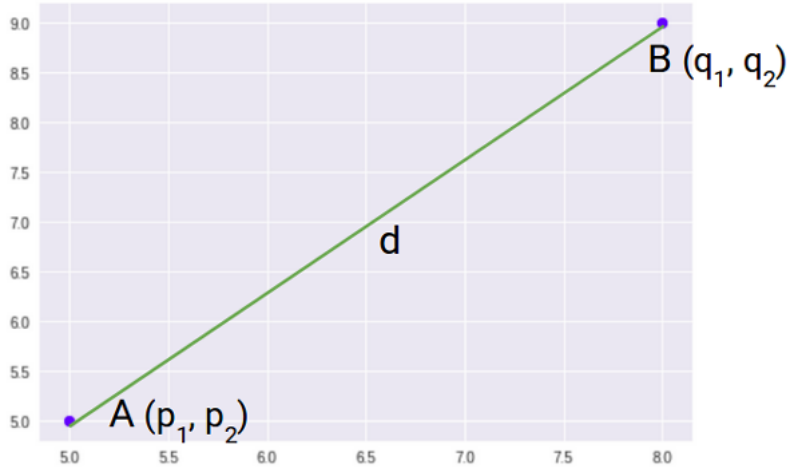
Let's start with the most commonly used distance metric – Euclidean Distance.

Euclidean Distance

This distance metric to measure the similarity between observations. Let's say we have two points, as shown below:



- So, the Euclidean Distance between these two points, A and B, will be:



- Formula for Euclidean Distance

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$

- We use this formula when we are dealing with 2 dimensions. We can generalize this for an n-dimensional space as:

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

Where,

n = number of dimensions

pi, qi = data points

Let’s code Euclidean Distance in Python.

This will give you a better understanding of how this distance metric works.

We will first import the required libraries. I will be using the SciPy library that contains pre-written codes for most of the distance functions used in Python:

```
# importing the library
from scipy.spatial import distance

# defining the points
point_1 = (1, 2, 3)
point_2 = (4, 5, 6)
point_1, point_2
```

```
((1, 2, 3), (4, 5, 6))
```

These are the two sample points that we will be using to calculate the different distance functions. Let’s now calculate the Euclidean Distance between these two points:

```
# computing the euclidean distance
euclidean_distance = distance.euclidean(point_1, point_2)
print('Euclidean Distance b/w', point_1, 'and', point_2, 'is: ', eucl
```

```
Euclidean Distance b/w (1, 2, 3) and (4, 5, 6) is:  5.196152422706632
```

✓ 0s completed at 7:33 AM