**Understanding Vector Similarity for Machine Learning**

Cosine Similarity, Dot Product, Manhattan Distance L1, Euclidian Distance L2.

[[Frederik vl](https://medium.com/@frederik.vl?source=post_page-----b9c10f7506de--------------------------------)](https://medium.com/@frederik.vl?source=post_page-----b9c10f7506de--------------------------------)

[[Advanced Deep Learning](https://medium.com/advanced-deep-learning?source=post_page-----b9c10f7506de--------------------------------)](https://medium.com/advanced-deep-learning?source=post_page-----b9c10f7506de--------------------------------)

[Frederik vl](https://medium.com/@frederik.vl?source=post_page-----b9c10f7506de--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F3b95e44a5f7c&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fadvanced-deep-learning%2Funderstanding-vector-similarity-b9c10f7506de&user=Frederik+vl&userId=3b95e44a5f7c&source=post_page-3b95e44a5f7c----b9c10f7506de---------------------post_header-----------)

Published in

[Advanced Deep Learning](https://medium.com/advanced-deep-learning?source=post_page-----b9c10f7506de--------------------------------)

·

5 min read

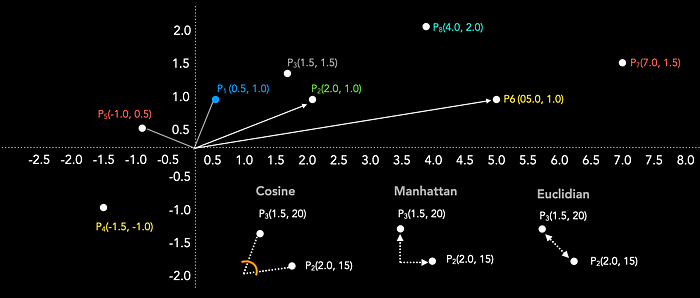
·

Oct 8

102

2

Similarity measures play a crucial role in machine learning. These measures quantify the similarity between objects, data points, or vectors in a mathematical manner. Understanding the concept of similarity in the vector space and employing appropriate measures is fundamental in solving a wide range of real-world problems. There are several similarity measures that can be used to calculate how close two vectors are in the embedding space.



**Cosine Similarity**

Cosine similarity (cos⁡(θ)) value ranges from -1 (not similar) to +1 (very similar). Figure 1 shows that point A(1.5, 1.5) and point B(2.0, 1.0) are close together in the 2-dimensional embedding space. When we calculate the cosine similarity, we obtain a value of 0.948, confirming that both vectors are quite similar. In contrast, when we compare the similarity of point A(1.5, 1.5) and point C(-1.0, -0.5), we observe that the cosine similarity is -0.948, indicating that both vectors are dissimilar. We can see that they are in opposite directions in the embedding space. A cos⁡(θ) value of 0 would indicate that both vectors are perpendicular to each other, showing neither similarity nor dissimilarity.

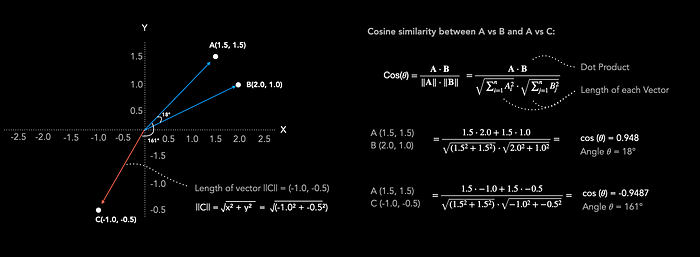


Figure 2: Cosine Similarity

To calculate the cosine similarity between two vectors, one can simply divide the dot product of both vectors by the product of their lengths. Cosine Similarity primarily considers the angle between two vectors to determine their similarity and disregards the length of the vectors.

Calculating cosine similarity is straightforward in Python. One can convert the similarity value, cos(θ), to the angle measured in degrees between the two vectors (θ) by taking the inverse cosine.”

import torch  
import torch.nn.functional as F  
import math  
  
#Create 3 Vectors  
A = torch.tensor([1.5,1.5])  
B = torch.tensor([2.0,1.0])  
C = torch.tensor([-1.0,-0.5])  
  
# Calculate cosine similarity cos(𝜃):  
cos = F.cosine\_similarity(A, B, dim=0)  
print("Cosine Similarity:", cos)  
  
# Calculate the angle 𝜃:  
# acos is the inverse of cos(x)  
theta = math.acos(cos)   
  
# Convert radians to degrees  
theta\_degrees = math.degrees(theta)  
  
print("Angle in radians:", theta)  
print("Angle in degrees:", theta\_degrees)

**Dot Product**

The Dot Product is a commonly used similarity metric. Both the Dot Product and Cosine Similarity are closely related concepts. However, Dot Product values can range from negative infinity to positive infinity, with negative values indicating opposite directions, positive values indicating the same direction, and a value of 0 when the vectors are perpendicular. Larger Dot Product values indicate greater similarity. Figure 3 illustrates the calculation of the Dot Product between Point P1 and the remaining Points P2 to P5.

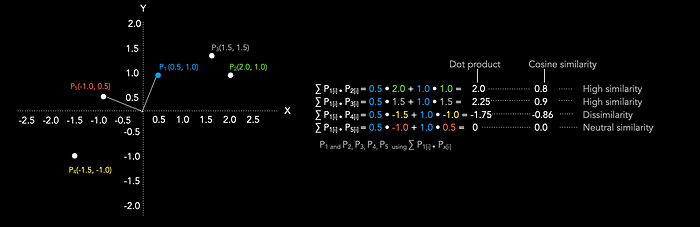


Figure 3: Comparison between Dot Product and Cosine Similarity

The Dot Product can be derived from the cosine equation: by multiplying the cosine of the angle between two vectors by the lengths of both vectors, we obtain the Dot Product, as depicted in Figure 4. Consequently, the Dot Product is influenced by the length of vector embeddings, which can be a crucial consideration when choosing similarity measures



Figure 4: Cosine & DotProduct

How does the Dot Product affect similarity measures? Imagine you are calculating similarity for a collection of scientific research papers. The length of the embedding vectors of research papers is proportional to the number of citations they have received. Initially, you use cosine similarity to calculate the similarity between research papers. However, you decide to switch to using the dot product instead. How does the similarity between research papers change?

Cosine similarity considers both the direction and magnitude of vectors, making it suitable for cases where the length of the vectors is not directly related to their similarity. However, when you use the dot product, only the magnitudes of the vectors matter, and the direction becomes less relevant.

1. Papers with high citation counts (longer vectors) will have higher dot product similarity scores with other high-citation papers because their magnitudes contribute more to the result.
2. Papers with low citation counts (shorter vectors) will have lower dot product similarity scores with high-citation papers because their magnitudes are smaller.
3. Papers that are not closely aligned in terms of their content (even if they have similar citation counts) may have lower dot product similarity scores compared to cosine similarity because the dot product does not consider the direction of the vectors.

In contrast, if one switches from Dot Product to Cosine Similarity, Papers with high citation counts become less similar than papers with less citation counts because the length of the longer embeddings for high citation papers is not taken into account anymore.

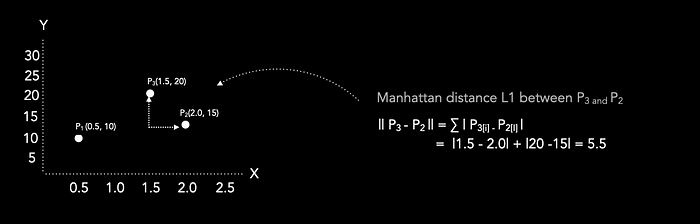
**Manhattan (L1) and Euclidian (L2) Distance**

Manhattan distance calculates distance by summing the absolute differences along each dimension, whereas Euclidean distance calculates the straight-line distance between points.

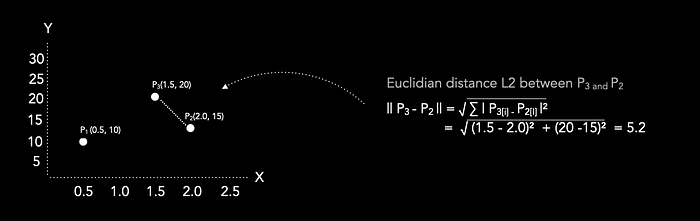
Manhattan distance is suitable for scenarios involving grid-like movement or when individual dimensions have varying importance. On the other hand, Euclidean distance is ideal when measuring the shortest path or when all dimensions contribute equally to the distance.

In most cases, Manhattan distance produces larger values than Euclidean distance for the same pair of points .As the data dimensionality increases, Manhattan Distance becomes the preferred choice compared to the Euclidean distance metric [1].

**Manhattan Distance L1**



**Euclidian Distance L2**



**Literature**

[1] C. Aggarwal, A. Hinneburg and D. Keim, “On the Surprising Behavior of Distance Metrics in High Dimensional Space”, 2001.<https://bib.dbvis.de/uploadedFiles/155.pdf>