**Introduction**

Unsupervised learning, particularly clustering, hinges on a fundamental question: *how similar or dissimilar are data points to each other?* This core concept is driven by **similarity measures**, mathematical tools that quantify closeness or distance between data items. As Chopra (2023) explains, clustering “Support Vector Machine (SVM) is one of the most popular supervised machine learning algorithms. It is used for performing classification and clustering tasks,” making similarity metrics pivotal for deciding how clusters form (p. 49). In this post, I will explore two popular similarity measures: **Euclidean distance** and **Cosine similarity**, explaining their mathematical foundations, strengths, weaknesses, and practical application in unsupervised learning. (Paul, n.d.)

**1. Euclidean Distance**

**a. Explanation and Mathematical Definition**

Euclidean distance is perhaps the most intuitive similarity measure. It defines the “straight-line” distance between two points in a multi-dimensional space.

Mathematically, given two points:

A group of math equations

AI-generated content may be incorrect.

the Euclidean distance d(X,Y) is:

A mathematical equation with numbers and symbols

AI-generated content may be incorrect.

In simpler terms, it squares the difference between each corresponding feature, sums those squares, and takes the square root.

As Castro (2020) mentions in his video lecture, Euclidean distance “if you have these two data sets and you can see here that the range of one variable goes from zero to one and the other variable from zero to six all the way around so in this case the<..>” (00:44), this works like the ruler you’d use in real life; it’s measuring the direct path from one point to another.

**b. Strengths and Weaknesses**

**Strengths:**

* **Interpretability:** Euclidean distance is easy to understand and visualize, especially in 2D or 3D space.
* **Widely Supported:** Many algorithms, including K-Means, default to Euclidean distance because it performs well with numeric, continuous data (Chopra, 2023; Simplilearn, 2018).
* **Efficiency:** Computationally straightforward, especially with vectorized operations in Python libraries like NumPy.

**Weaknesses:**

* **Sensitive to Scale:** Euclidean distance is highly sensitive to the scale of data. A single feature with a large range can dominate the distance calculation. Hence, standardization or normalization is often required (Chopra, 2023, p. 29).
* **Not Robust to Outliers:** Large differences in one feature can disproportionately affect the overall distance.
* **Limited for Sparse Data:** Performs poorly when dealing with sparse vectors, such as text data represented via TF-IDF.

Kerzner (2022) emphasizes the importance of selecting techniques suited for the specific data context, stating, “Success requires matching the technology to the data’s characteristics rather than forcing the data to fit the method” (p. 371).

**2. Cosine Similarity**

**a. Explanation and Mathematical Definition**

Cosine similarity measures the cosine of the angle between two vectors. Instead of considering the magnitude of vectors (like Euclidean distance), it considers only the *orientation*—how similar their directions are.

Given vectors X and Y:

A black and white text

AI-generated content may be incorrect.

where:

* X\*Y is the dot product of the vectors
* ∥X∥ and ∥Y∥ are the Euclidean norms (magnitudes) of the vectors

Values range between -1 and 1:

* **1:** vectors point in the same direction
* **0:** vectors are orthogonal (no similarity)
* **-1:** vectors point in opposite directions

Castro (2020) explains, “it's which one minimizes this expression so the only thing that matters is how many points you have on the right or on the left and this is why this is called the taxicab distance because when you're comparing these two points the only thing that matters is if this point is in this side of the of the city or in this side of the city<..> .“ (03:22) Cosine similarity ignores the absolute values of the features, focusing instead on whether two vectors share the same orientation in space, making it excellent for text data where absolute word frequencies might vary but the pattern matters. (Paul, n.d.).

**b. Strengths and Weaknesses**

**Strengths:**

* **Scale Invariance:** Unlike Euclidean distance, cosine similarity is insensitive to the magnitude of vectors. Two vectors with the same proportions but different lengths will still be considered identical in direction.
* **Excellent for Sparse Data:** Ideal for high-dimensional, sparse data such as document-term matrices in text analysis.
* **Fast Computation:** Especially efficient in sparse matrix calculations.

**Weaknesses:**

* **Not a True Metric:** Cosine similarity is technically not a distance metric because it does not satisfy the triangle inequality, limiting its use in some algorithms requiring metric space properties in Distance Metrics (Paul, n.d.).
* **Sensitive to Data Distribution:** For data where magnitude carries meaning (e.g., physical measurements), cosine similarity might fail to capture differences adequately.
* **Requires Non-Negative Data:** It can behave unexpectedly if the data includes negative values, common in some centered datasets.

**Practical Example: Clustering News Articles**

**Scenario**

Suppose we have a dataset of 500 news articles represented as TF-IDF vectors (terms weighted by their importance in the corpus). We want to group these articles into topics.

**Using Euclidean Distance**

Applying **Euclidean distance** on TF-IDF vectors can be misleading. Let’s consider:

* **Article A:** TF-IDF vector = [0.8, 0.2, 0.0, 0.0, 0.0]
* **Article B:** TF-IDF vector = [0.16, 0.04, 0.0, 0.0, 0.0]

Article B is merely a scaled-down version of Article A, possibly due to document length differences.

**Euclidean Distance:**



This non-zero distance suggests these articles are meaningfully different, even though they talk about precisely the same topics.

Chopra (2023) discusses this pitfall, noting that Euclidean distance can be KNN algorithm is similar to the Supervised Learning technique. KNN algorithm works on the similarity technique and allots the category to the new data which is similar to the available categories. KNN is said to be non-parametric algorithm.” (p. 78) And can incorrectly separate vectors that are directionally similar but differ in length.

**Using Cosine Similarity**

Let’s compute the cosine similarity instead:

Dot product A\*B:



Norms:

A math equations with numbers

AI-generated content may be incorrect.

Cosine similarity:

A black numbers and a black line

AI-generated content may be incorrect.

Despite Euclidean distance indicating a difference, cosine similarity correctly identifies the articles as **identical in orientation**, reflecting they share the same topic distribution.

This is why Paul (n.d.) emphasizes cosine similarity’s suitability for text clustering: In document clustering, cosine similarity often outperforms Euclidean distance because documents may have different lengths but similar patterns, in clustering (para. 4).

**Implication in K-Means Clustering**

Most implementations of K-Means clustering default to Euclidean distance because the algorithm minimizes intra-cluster variance calculated through Euclidean norms (Chopra, 2023; Simplilearn, 2018). However, for text data, practitioners often switch to **Spherical K-Means**, which replaces Euclidean distance with cosine similarity. The algorithm then maximizes cosine similarity between documents and cluster centroids.

In the news article, an example, using cosine similarity results in clusters reflecting semantic topics rather than absolute word counts. For instance, two documents of different lengths about “climate change” will cluster together because they use similar terms in similar proportions, regardless of length.

This aligns with insights from Datatab (2023), who note in their video, “dimensions and then calculate the distances from each cluster to every other cluster the distance between Allen and Lisa is given by the square root out of 5 minus squared plus 2 minus 3 squared which <..>” (04:10). The choice of similarity measures can drastically change clustering outcomes; picking one aligned with the data type and business objective is essential, this is part of distances from each cluster.

**Silhouette Score and Evaluation**

No discussion of similar measures in clustering is complete without mentioning **cluster evaluation**. Both Euclidean and cosine-based clustering solutions can be evaluated using the **Silhouette Score**:

A mathematical equation with black text

AI-generated content may be incorrect.

* a = average intra-cluster distance
* b = average nearest-cluster distance

Scores close to 1 suggest tight, well-separated clusters. Chopra (2023) emphasizes that using the correct similarity measure improves silhouette scores because clusters become more meaningful (p. 80-88).

**Conclusion**

Similarity measures form the mathematical heart of unsupervised learning. Euclidean distance excels in numeric, low-dimensional data but struggles with scale and sparse data. Cosine similarity, on the other hand, thrives in high-dimensional contexts like text mining, ignoring magnitude in favor of directional alignment. As Kerzner (2022) advises, “Most organizations have impediments that can affect all projects, not just innovation. The impediments that have the greatest impact are those related to how a firm funds and staffs projects. Companies have strict policies on how funds are allocated, especially for innovation activities. There is a great deal of competition for these funds.” (p. 384), this is selecting the appropriate tools for the data and objectives is critical for innovation success. By understanding these nuances, data scientists can better tailor clustering solutions to diverse datasets, ensuring that the clusters discovered reflect genuine patterns rather than artifacts of inappropriate distance metrics.

**References**

Castro, M. (2020, November 23). *Distance and similarity measures* [Video]. YouTube. https://www.youtube.com/watch?v=ySyuf0rMEdk

Chopra, D. (2023). *Introduction to machine learning with Python*. Bentham Science Publisher.

Datatab. (2023). *Hierarchical cluster analysis [Simply explained]* [Video]. YouTube. <https://www.youtube.com/watch?v=8QCBl-xdeZI>

Kerzner, H. (2022). *Innovation project management: Methods, case studies, and tools for managing innovation projects*. John Wiley & Sons Incorporated.

Paul, S. (n.d.). *Hierarchical clustering in Python*. GitHub. <https://github.com/sandipanpaul21/Clustering-in-Python/tree/master>

Quintana, A. M. Q., & Zambrano, A. X. B. (2024). Project-based learning. *International Research Journal of Management, IT & Social Sciences*, 11(1), 58–65. <https://www.researchgate.net/publication/377674767_Project-based_learning>

Simplilearn. (2018, March 19). *K means clustering algorithm | K means in Python | Machine learning algorithms* [Video]. YouTube. <https://www.youtube.com/watch?v=Xvwt7y2jf5E>