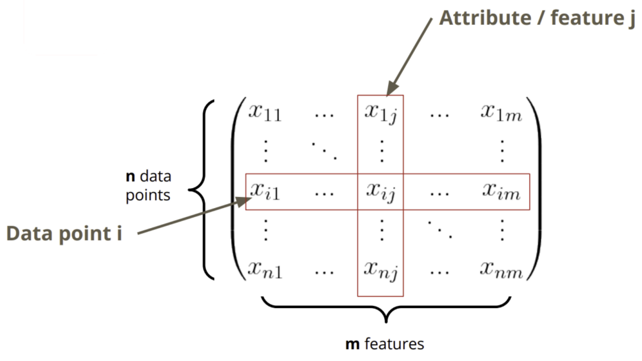
I. Data

We represent data as a grid (or matrix) because it organizes information in a structured, numerical way that computers and mathematical models can easily work with.

- Each row represents one observation (a data point).
- Each column represents one measurable property (a feature).


Attribute / feature j
n data points
Data point i
m features
x_{11} … x_{1j} … x_{1m}
x_{i1} … x_{ij} … x_{im}
x_{n1} … x_{nj} … x_{nm}

This structure lets us quickly access, compare, and compute relationships between data points -- for example, by calculating distances, averages, correlations, or performing matrix operations.

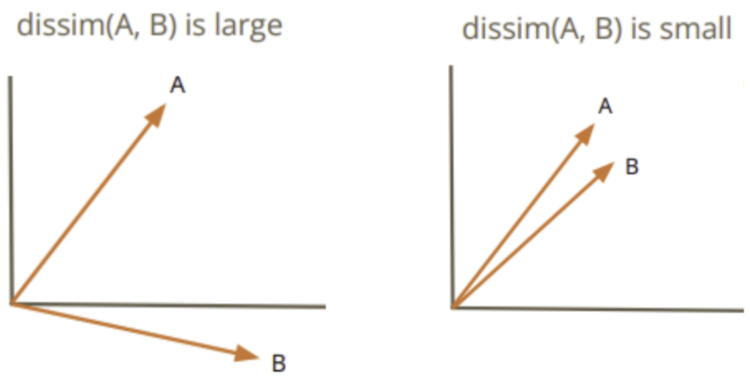
Feature space: a geometric representation of your data where each feature (column) becomes one axis, and each data point (row) becomes a point in that space.

II. Dissimilarity and Distance

Dissimilarity: a way to measure how different two data points are from each other. It's a function that takes two objects and returns a large value when they are very different and a small value when they are similar.

For example, if you compare two people by age and income:

- If their ages and incomes are close, the dissimilarity is small.
- If their ages and incomes are very different, the dissimilarity is large.



In data analysis, dissimilarity helps us compare, group, or separate data points.

Distance: a specific kind of dissimilarity that follows certain mathematical rules. It measures how far apart two points are in a feature space -- just like physical distance measures how far two locations are on a map.

A function is a distance if it satisfies these properties (regarding datapoints):

1. d(i, j) = 0 only when the two points are the same.
2. d(i, j) = d(j, i) (symmetry).
3. d(i, j) <= d(i, k) + d(k, j)

(Triangle inequality: going directly from A to C datapoints is never longer than going A→B→C.)

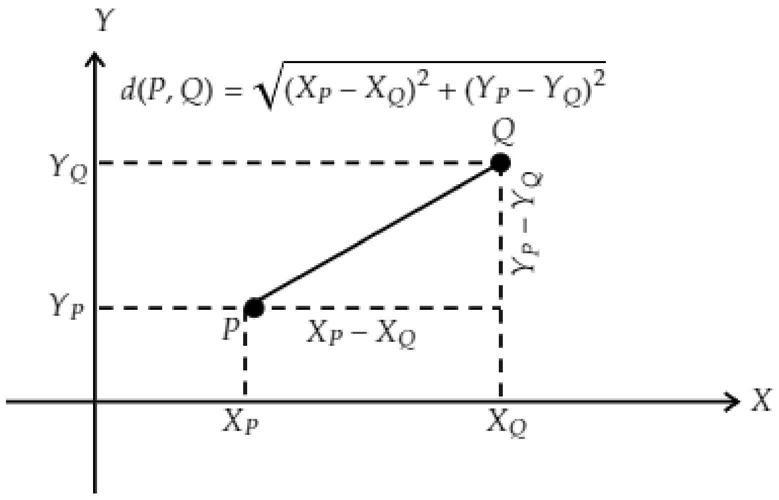
III. Distance Metrics

1. Euclidean Distance

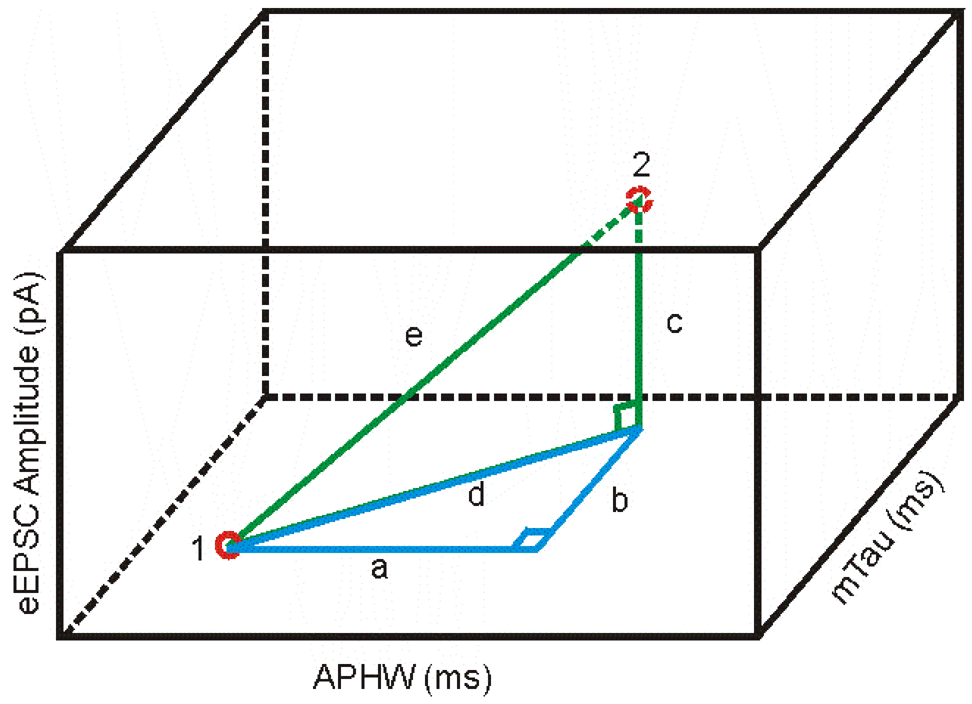
- Concept: Calculates the minimum straight-line distance between two points. It's an extension of the Pythagorean theorem.
- Formula (2D): $Distance = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$
- Formula (n dimensions): the square root of the sum of the squared differences across all dimensions

How the formula creates a straight line distance:

1. The distance in a 2D space is the hypotenuse of a right-angled triangle.
2. The terms (x2−x1) and (y2−y1) represent the **lengths of the two legs** of the triangle (the horizontal and vertical differences).
3. is the direct application of $a^2 + b^2 = c^2$, where D is the hypotenuse, which is the **shortest possible path** (the straight line) between the two points.

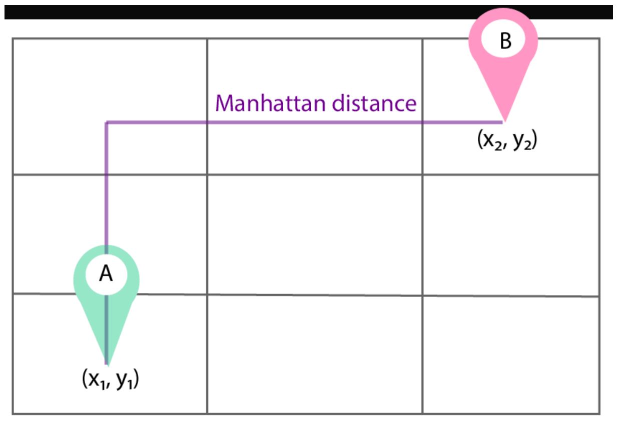


Computational Cost: It is computationally costly as the number of dimensions increases because it involves squaring values



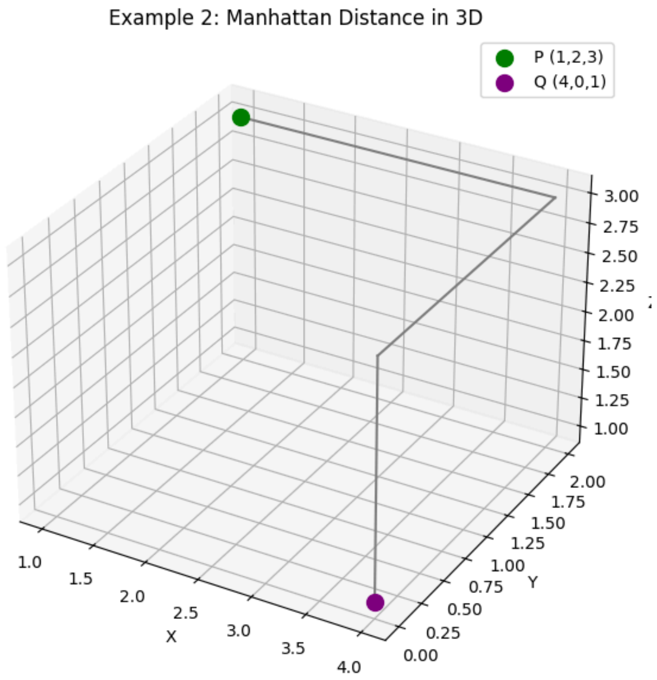
2. Manhattan Distance

- Concept: Calculates the distance by summing the absolute differences of the coordinates. It's like navigating a grid (like city blocks).
- Formula (2D): D = |x2−x1|+ |y2−y1|
- Formula (n dimensions): the sum of the absolute differences of the coordinates in each dimension.
- Comparison to Euclidean: Euclidean finds the minimum distance (diagonal), while Manhattan sums the side distances (L-shape)



How the formula creates distance: Instead of calculating the diagonal, straight-line path like Euclidean distance, Manhattan distance calculates the path as if you are moving in a grid. Imagine a taxi driving in a city grid—it must follow the streets and cannot cut diagonally through buildings.

Computational Cost: It is computationally cheaper than Euclidean distance. It's often preferred for high-dimensional data.



3. Minkowski Distance

- Concept: This is the generalization of both Euclidean and Manhattan distance.
- Formula: $Distance = (\sum_{i=1}^{n} |xi - yi|^{p})^{1/p}$ OR $\sqrt[p]{|x1 - x2|^{p} + \ldots\ldots + |z1 - z2|^{p}}$
- Relationship to Others:
  - If P = 1, it becomes the Manhattan Distance (L1 norm)
  - If P = 2, it becomes the Euclidean Distance (L2 norm)

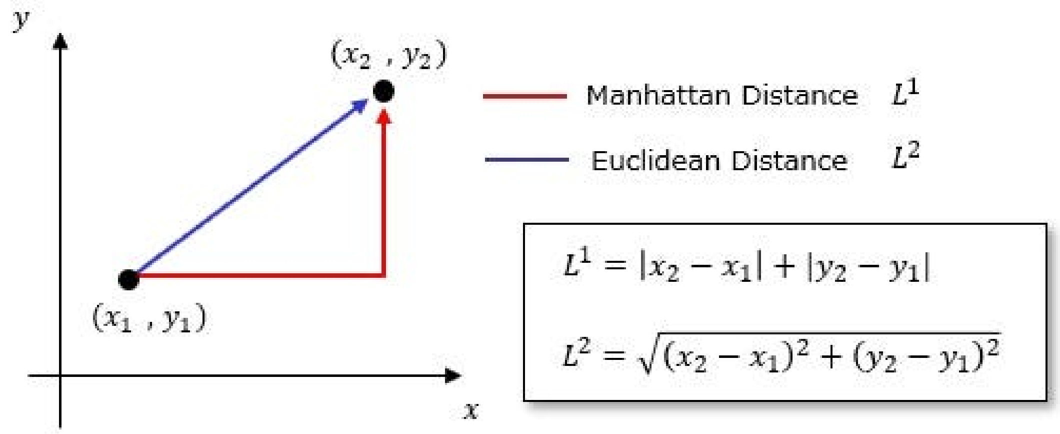
Understanding P:

In essence, P controls the "path" the distance is measured along. A higher value of P places greater emphasis on the larger coordinate differences, as raising a number greater than one to a higher power increases its value exponentially more than a smaller number.

P=1 (Manhattan)	Linear Weighting – differences are summed directly (raised to the power of 1). All dimensional differences contribute equally to the distance. Measures grid or "taxicab" movement.
P=2 (Euclidean)	Quadtrattic Weighting – Differences are squared. This means large differences are given significantly more weight than small differences. It gives the straight-line distance.
P=3 (L3 Norm)	Cubic Weighting – Differences are raised to the third power. This places even more emphasis on the largest coordinate differences than the Euclidean distance does. The metric becomes more dominated by the dimension with the biggest difference.
P→∞ (Chebyshev /L-infinity)	Extreme Weighting – In the extreme limit, the distance is determined only by the single largest difference across all dimensions.

Minkowski in general: Minkowski takes the absolute differences between the coordinates of two points and powers them to P in order to weight the contributions of each dimensional difference. Raising the differences to the power of P controls the impact of larger differences: a higher P gives disproportionately more weight to the largest gaps between the coordinates.

It then P roots the whole thing in order to return the final distance to a linear unit of measurement. Since the original differences were raised to the P power, taking the P-th root reverses this scaling, converting the metric back into a unit that can be intuitively understood as a "distance" or length. This root ensures the distance is measured in the same units as the input data.



IV. Similarity
A similarity function is a function that takes two objects (data points) and returns a large value if these objects are similar.

1. Jaccard Similarity
Jaccard similarity measures how similar two sets are by comparing how many elements they share. It's useful for binary or categorical data -- like words in documents, items in shopping lists, or features that are present/absent.

Formula: for two sets A and B: $J(A,\ B) = \frac{A \cap B}{A \cup B}$

|A∩B|: number of elements common to both sets
|A∪B|: total number of elements in both sets

J(A,B) = 1 → The sets are **identical** (perfectly similar).

J(A,B)= 0 → The sets share **no elements** (completely different).

Values between 0 and 1 show **partial similarity**.

For example:
A = {apple, banana, cherry}
B = {banana, cherry, date}
A∩B = {banana, cherry} = 2
A∪B = {apple, banana, cherry, date} = 4
J(A, B) = 2/4 = 0.5 …. Jaccard Similarity = 50% …….. they are 50% similar

Jaccard Distance: you can turn it into a dissimilarity (distance) measure:

D(A, B) = 1 - J(A, B)

So in the example above, D(A,B) = 1 − 0.5 = 0.5
That means they are also **50% dissimilar**.

---

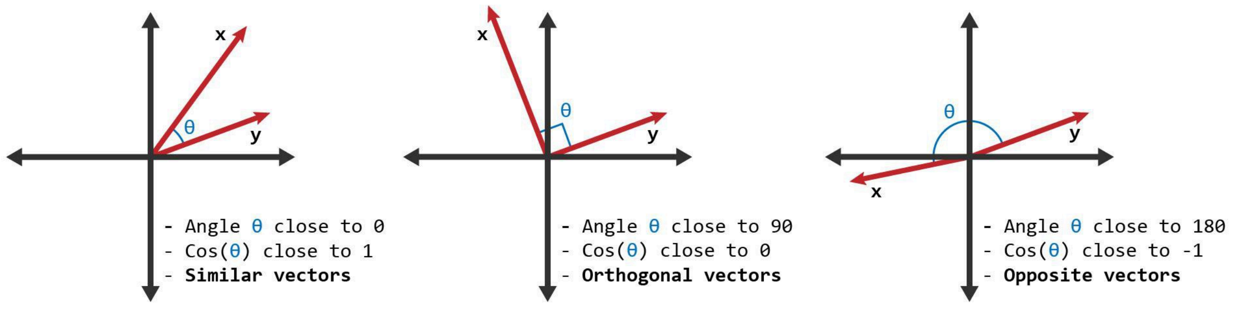
2. Cosine Similarity

Cosine similarity measures how similar two vectors are by looking at the angle between them -- not their length (magnitude).

similarity(x, y) = cos(θ) where θ = angle between those two vectors (e.g., datapoints)

Cosine Similarity	Meaning
1	Vectors point in exactly the same direction → perfectly similar
0	Vectors are orthogonal (unrelated)
-1	Vectors point in opposite directions → perfectly dissimilar

Cosine helps covert the angle (e.g. 80 degrees) to a similarity score [-1, 1]



Understanding θ (angle):

When we represent data as datapoints (vectors) like [1,2] or [3,4], we can imagine them as **arrows** starting from the origin and pointing somewhere in space.

- The direction of each arrow shows the pattern of the data.
- The length of the arrow shows its magnitude (size or intensity).

The angle (θ) between two vectors tells us how similar their directions are. If two arrows

- Point in exactly the same direction, the angle is 0° → cos(0) = 1 → very similar.

- Are perpendicular (90°), the angle is 90° → cos(90) = 0 → unrelated.
- Point in opposite directions, the angle is 180° → cos(180) = -1 → completely opposite.

The Point of Cosine Similarity:
**Cosine similarity uses this idea to measure how aligned two data points are, regardless of their size.

[1,1] and [100,100] point in the same direction (angle = 0°), so cosine similarity = 1 (they're identical in pattern). Even though their magnitudes differ, **their relationship between features is the same.** (whereas distance would consider the vectors far apart).

Cosine Dissimilarity (this is not distance):
- d(x, y) = 1 / s(x, y)
- d(x, y) = 1 - s(x, y)

Example Applications of Cosine Similarity/Dissimilarity:
- Comparing text documents represented by word frequencies -- longer documents have bigger numbers, but the relative proportions of words are what matter.

- Two people might listen to music at different volumes — one at level 3, another at level 10 — but if they both turn up the same songs and turn down the same ones, cosine similarity sees their listening patterns (shapes) as identical, even though their overall loudness (magnitude) differs.