How t-Distributed Stochastic Neighbor Embedding (t-SNE) works ?

Steps of t-Distributed Stochastic Neighbor Embedding (t-SNE) ?

Based on the steps you provided, here are the names for the five steps in the PCA algorithm :

1. **High-Dimensional Distance Calculation :** The algorithm first calculates the distance between every pair of data points in the high-dimensional space.

2. **High-Dimensional Similarity (Affinity) :** Using the calculated distances, the algorithm determines a probability or **affinity score** for each point with respect to every other point. A high score means they are likely neighbors.

3. **Low-Dimensional Initialization :** Before optimization begins, the data points are randomly placed in a lower-dimensional space, typically 2D or 3D.

4. **Iterative Optimization :** This is the core of the algorithm. It iteratively attracts points with high affinity and repels points with low affinity in the low-dimensional space to minimize the difference between the high-dimensional and low-dimensional probability distributions.

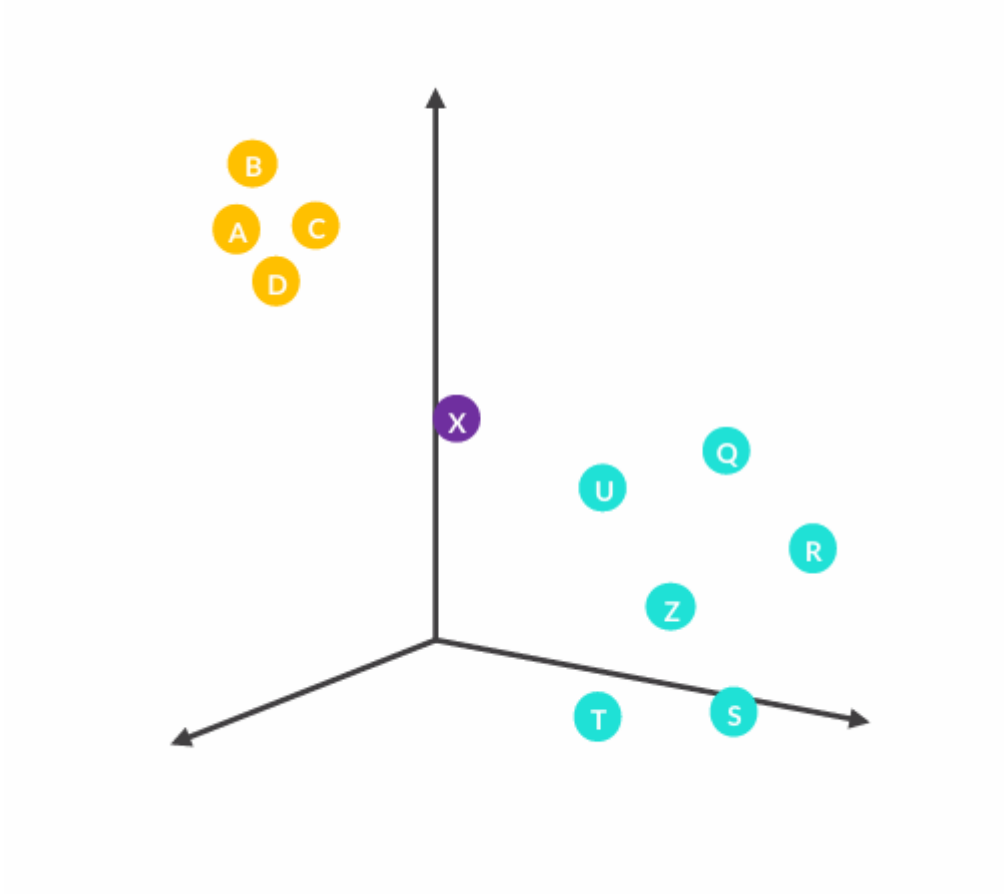
5. **Convergence :** The algorithm stops when the positions of the data points in the low-dimensional space have stabilized and no longer change significantly with each iteration.

Step 1 - High-Dimensional Distance Calculation

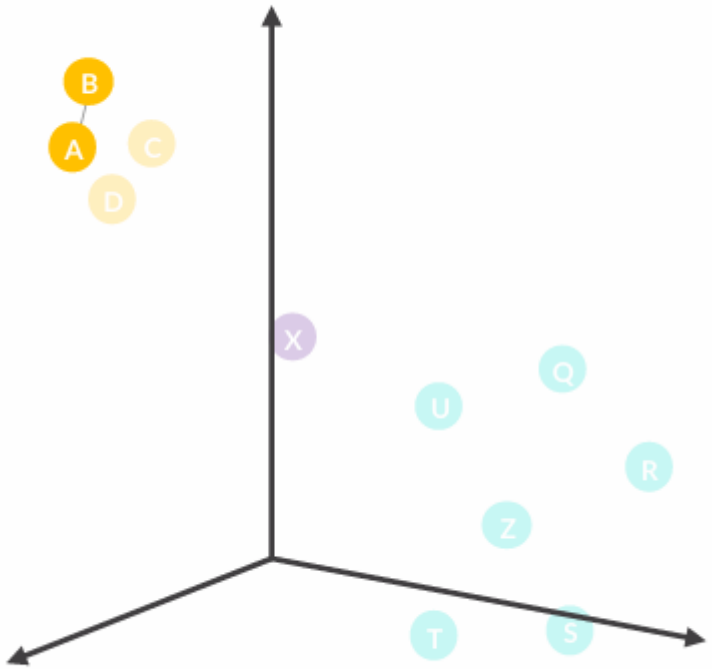
Step 1 - High-Dimensional Distance Calculation

The algorithm first calculates the distance between every pair of data points in the high-dimensional space.

Step 1.1 - Calculate the distance between points in a high-dimensional space

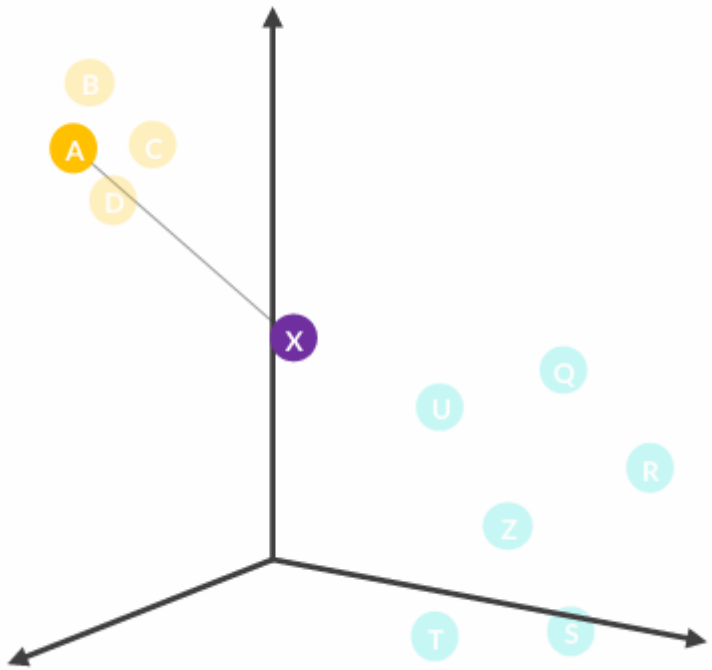


Step 1.2 - Calculate the distance between points in a high-dimensional space



$\text{distance}_{AB} = \mathbf{1.5}$

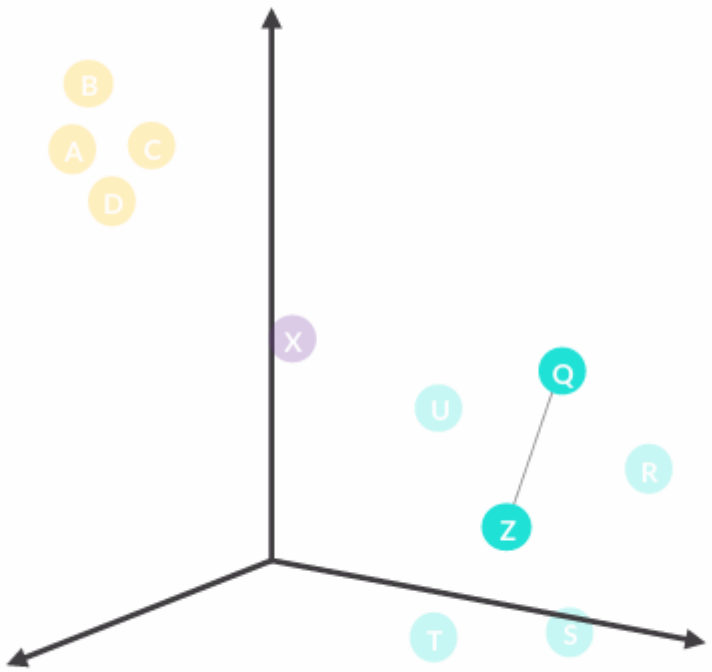
Step 1.3 - Calculate the distance between points in a high-dimensional space



$distance_{AB} = 1.5$

$distance_{AX} = 5.3$

Step 1.4 - Calculate the distance between points in a high-dimensional space

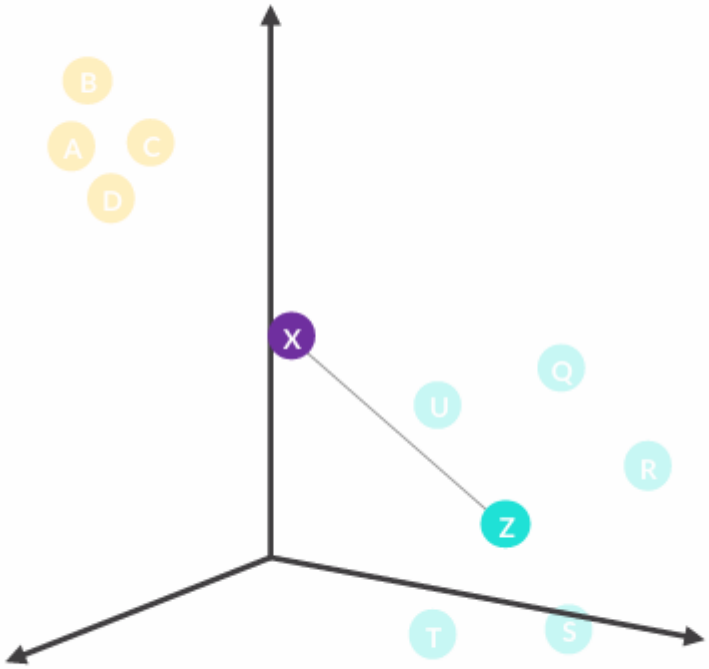


$distance_{AB} = 1.5$

$distance_{ZQ} = 3.2$

$distance_{AX} = 5.3$

Step 1.5 - Calculate the distance between points in a high-dimensional space



$distance_{AB} = 1.5$ $distance_{ZQ} = 3.2$

$distance_{AX} = 5.3$ $distance_{ZX} = 5.3$

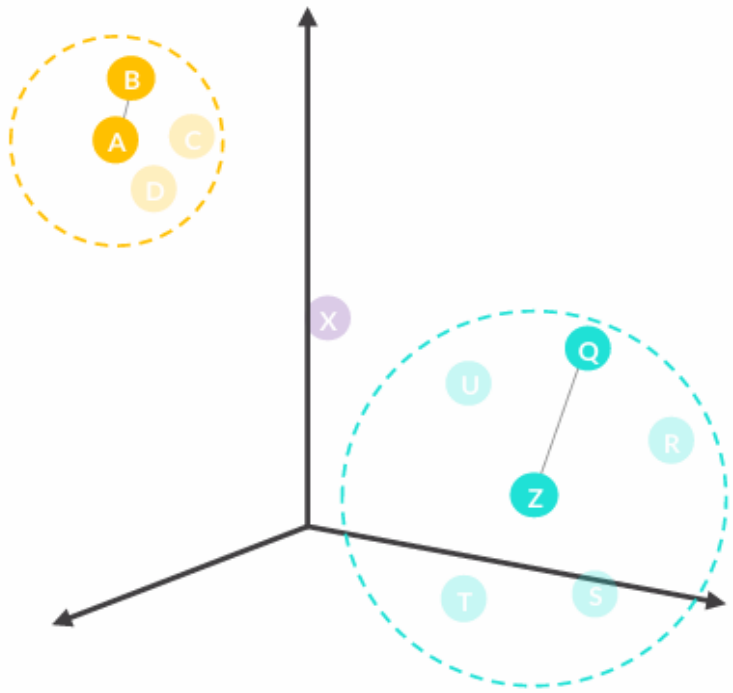
Point X is the same distance from points A and Z

Step 2 – High-Dimensional Similarity (Affinity)

Step 2 – High-Dimensional Similarity (Affinity)

Using the calculated distances, the algorithm determines a probability or **affinity score** for each point with respect to every other point. A high score means they are likely neighbors.

Step 2.1 - Use the distances to calculate the affinity score (0-1) for each point with the rest (high affinity score = high probability of being neighbors)



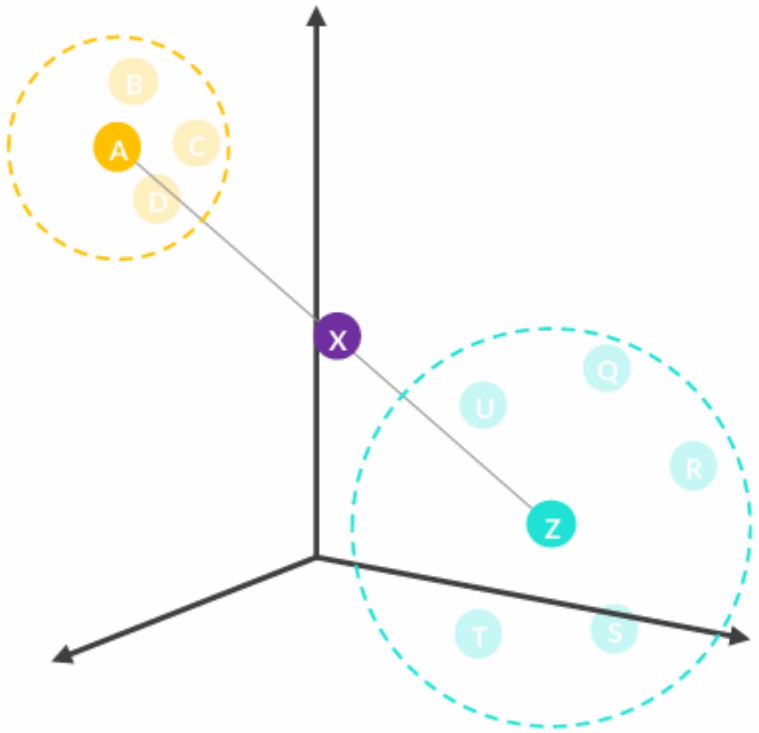
$distance_{AB} = 1.5$ $affinity_{AB} = high$

$distance_{ZQ} = 3.2$ $affinity_{ZQ} = high$

How does this work?

- Based on the density of points around A and the distance between A and B, it's highly likely that B is A's neighbor
- Even though Q is further from Z than B is from A, since Z is part of a less dense region there is still a high chance that Q is Z's neighbor

Step 2.2 - Use the distances to calculate the affinity score (0-1) for each point with the rest (high affinity score = high probability of being neighbors)



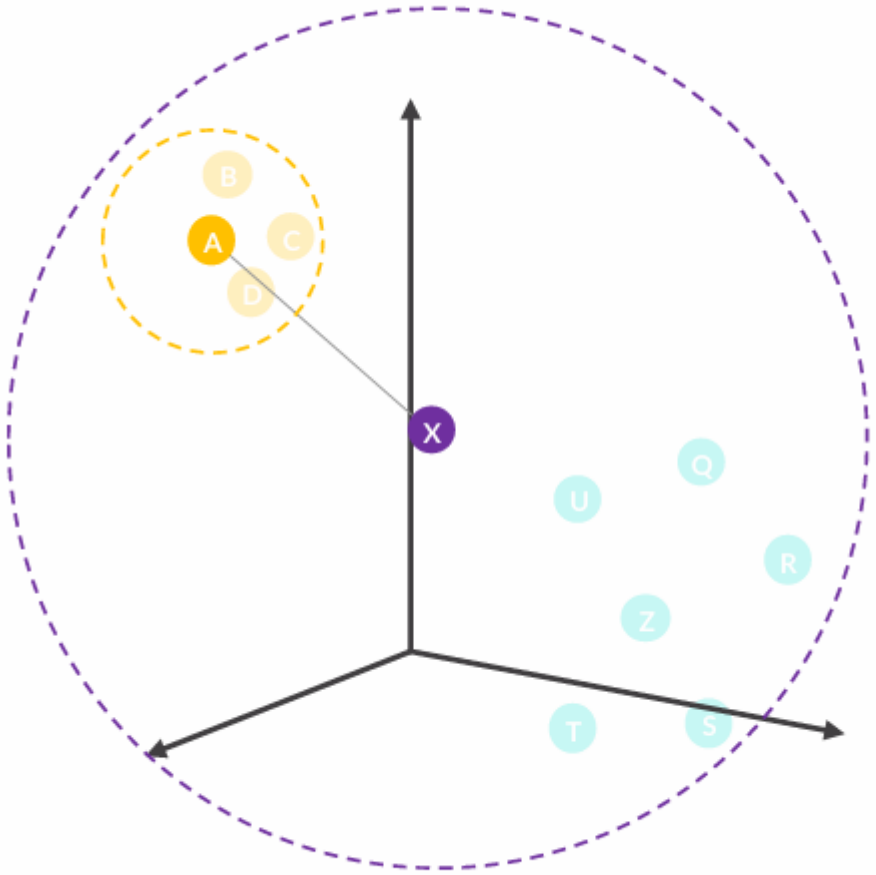
$distance_{AX}$ = **5.3** $affinity_{AX}$ = **low**

$distance_{ZX}$ = **5.3** $affinity_{ZX}$ = **medium**

How does this work?

- Even though X is the same distance from A and Z, it's more likely that X is a neighbor of Z due to the sparse density around Z

Step 2.3 - Use the distances to calculate the affinity score (0-1) for each point with the rest (high affinity score = high probability of being neighbors)



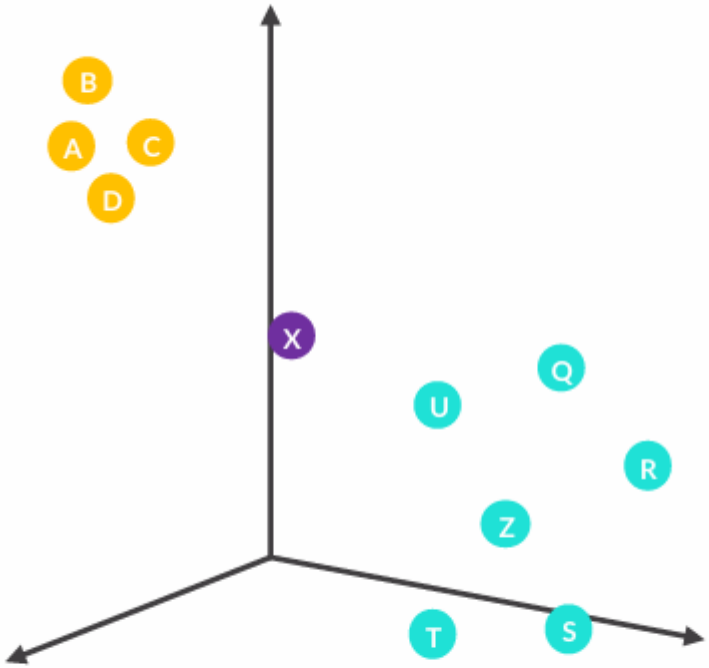
$distance_{AX} = 5.3$ $affinity_{AX} = \textbf{low}$

$distance_{XA} = 5.3$ $affinity_{XA} = \textbf{medium}$

How does this work?

- The density of points around X is much sparser, so it's much more likely that *A* is a neighbor of X than X being a neighbor of *A*

Step 2.4 - Use the distances to calculate the affinity score (0-1) for each point with the rest (high affinity score = high probability of being neighbors)



$affinity_{AB}$ = **high**

$affinity_{AX}$ = **low**

$affinity_{XA}$ = **medium**

$affinity_{ZQ}$ = **high**

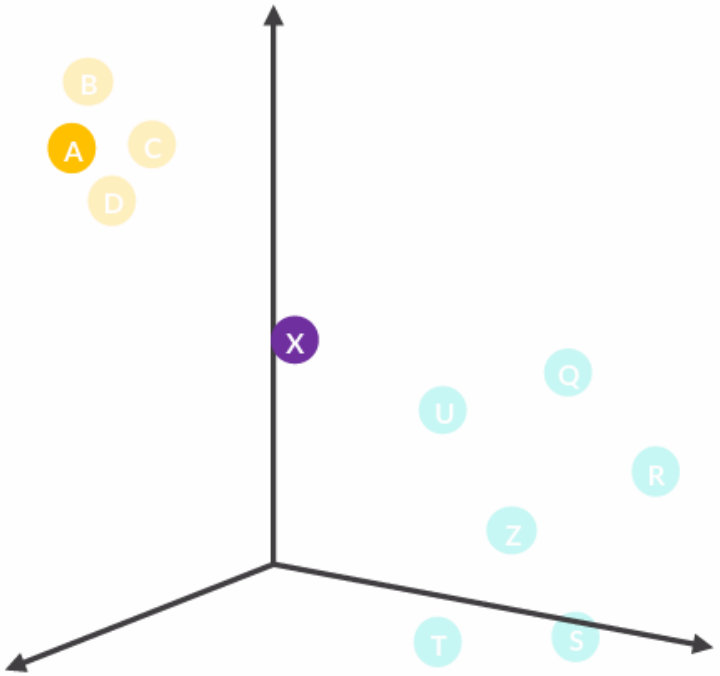
$affinity_{ZX}$ = **medium**

$affinity_{XZ}$ = **medium**

If you're curious, the affinity score is calculated by putting the distances through a Gaussian transformation and calculating probabilities – but it's not something you should be worried about!

Step 3 – Calculate the average affinity score for each pair of points

Step 3.1 - Calculate the average affinity score for each pair of points

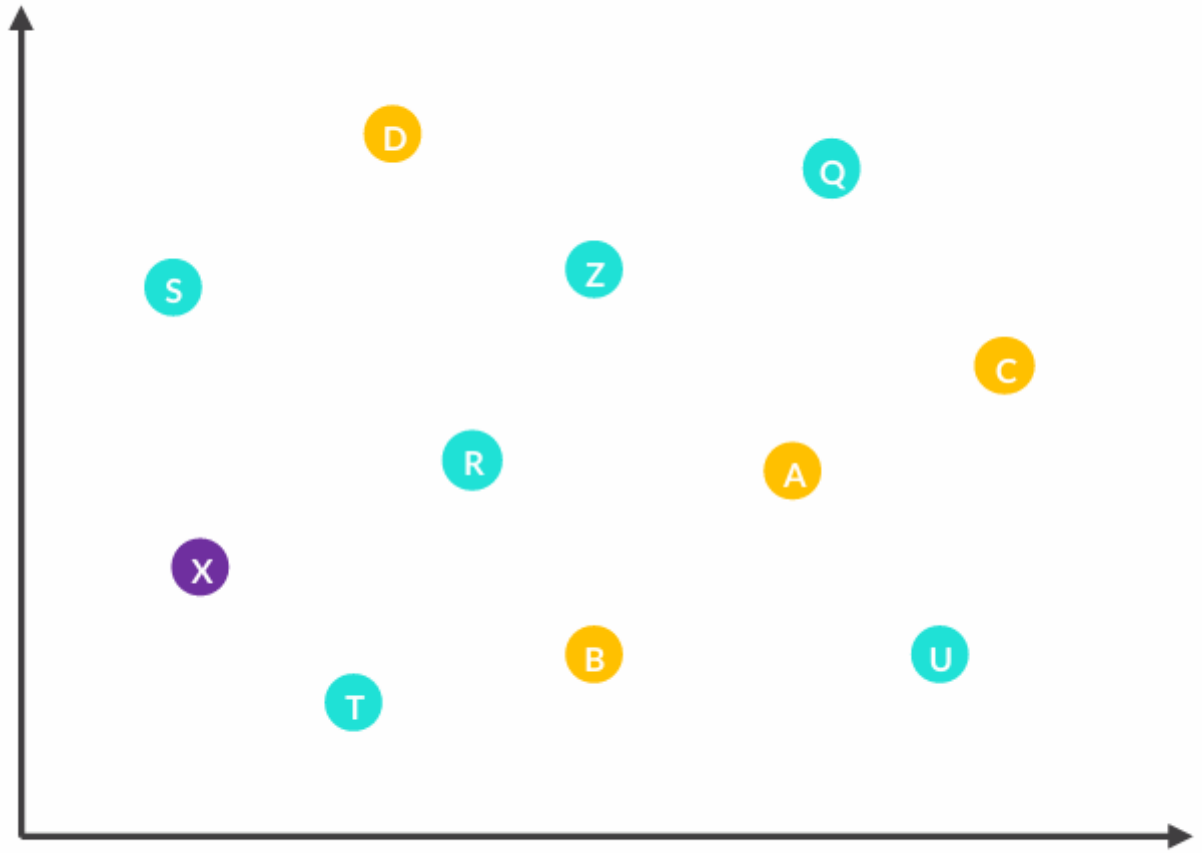


$affinity_{AX}$ = **low** $affinity_{XA}$ = **medium**

Affinity between A and X: **medium-low**

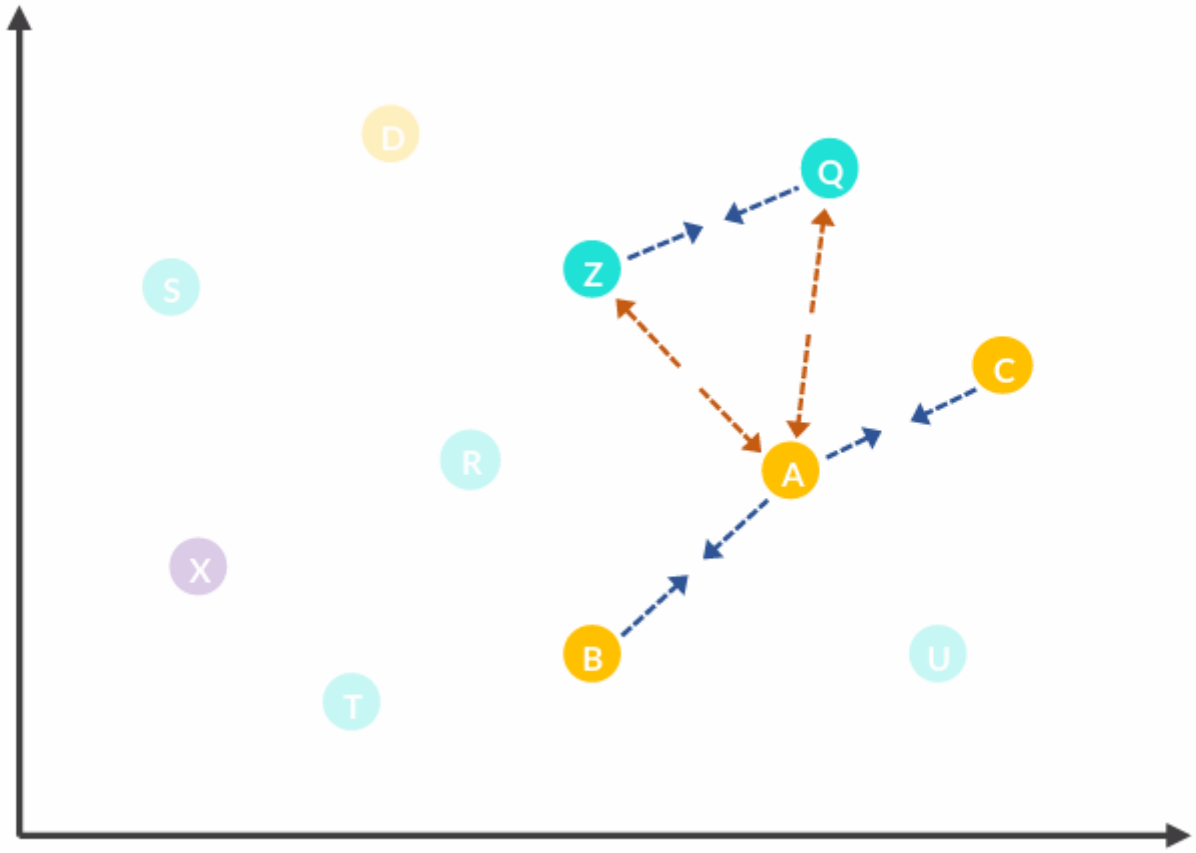
Step 4 - Randomly place the data points in a low dimensional space

Step 4.1 - Randomly place the data points in a low dimensional space

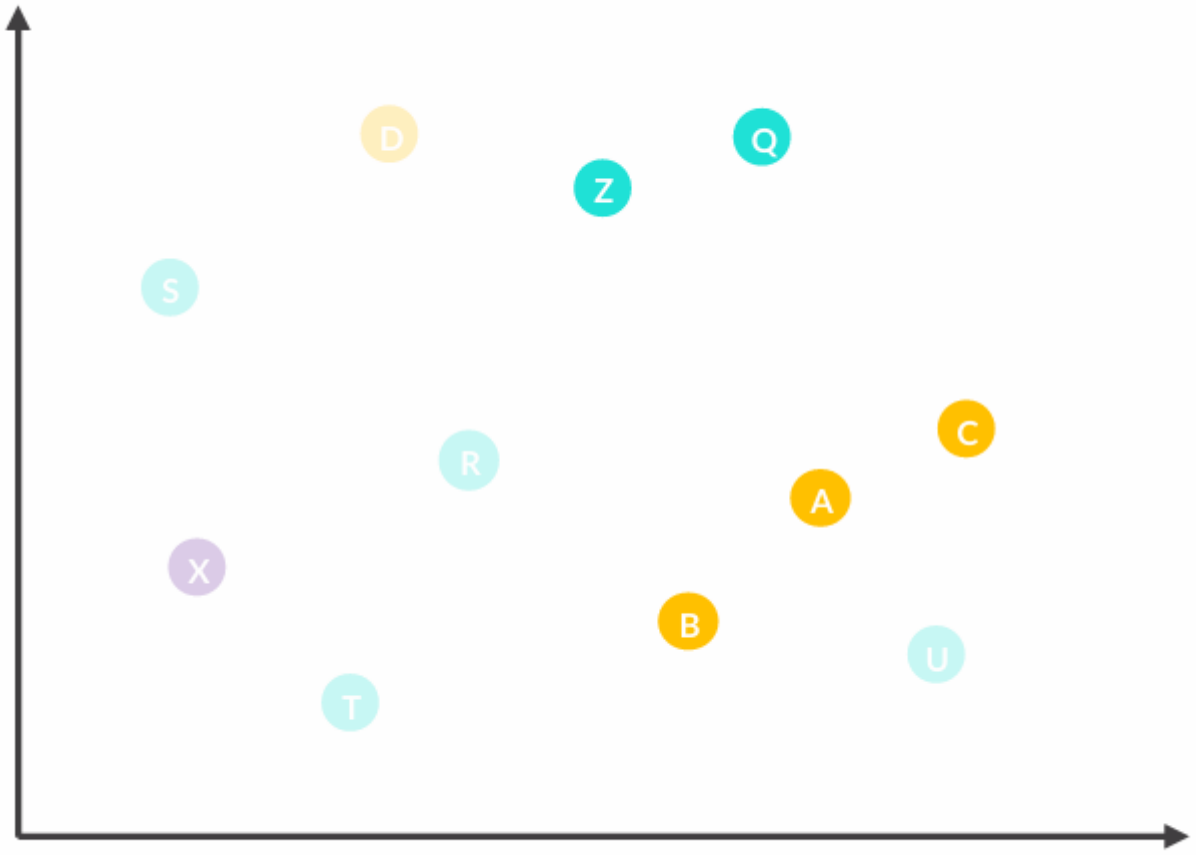


Step 5 - Dimensionality Reduction

Step 5.1 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position

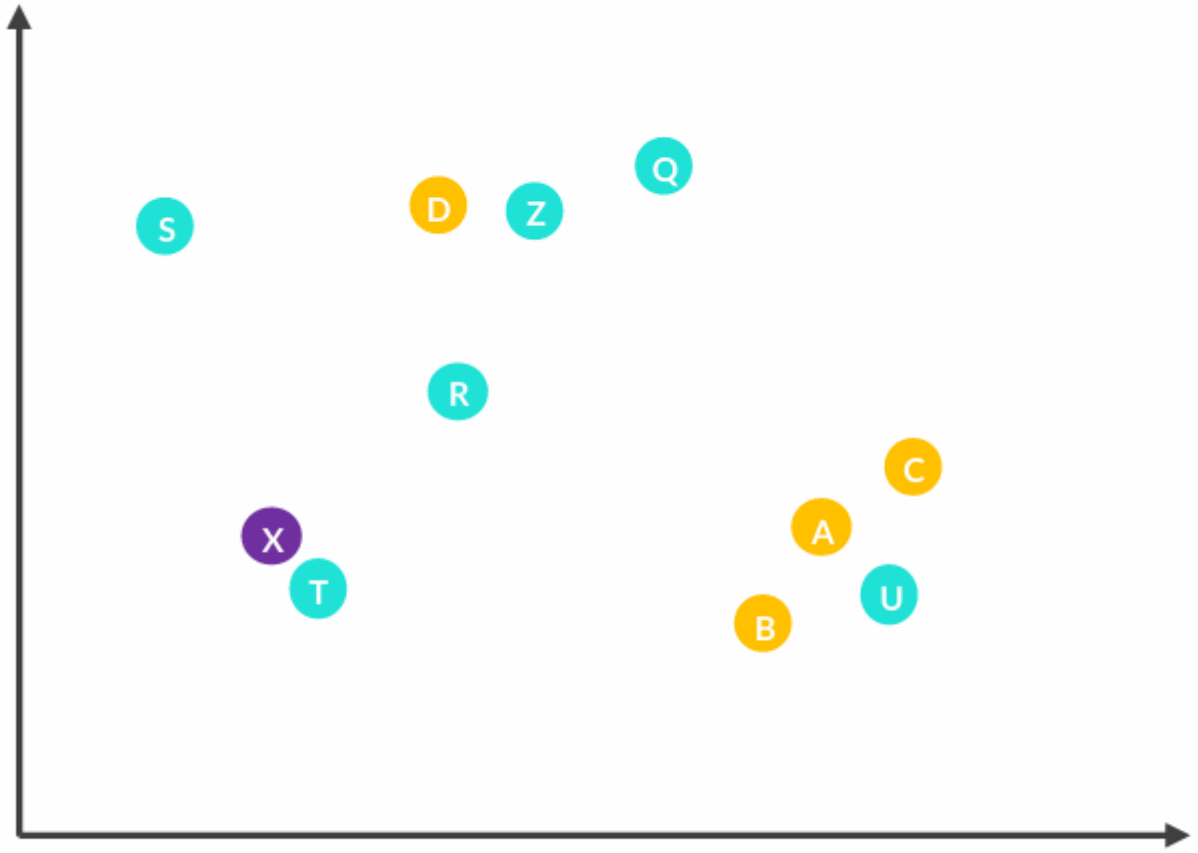


Step 5.2 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position

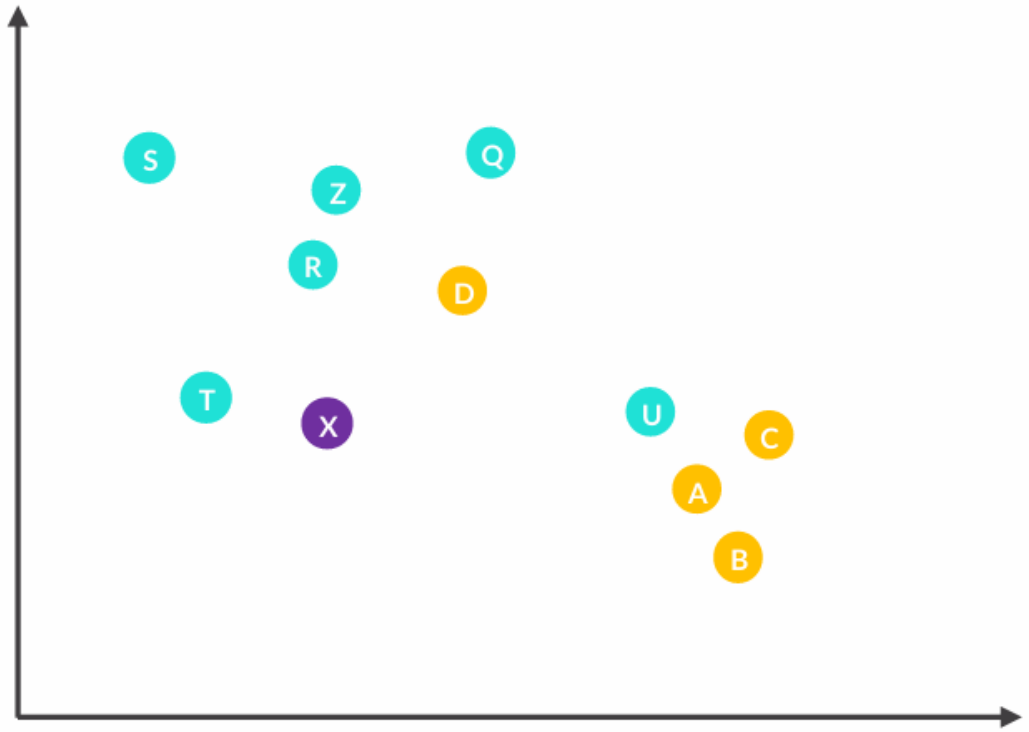


This attraction and repulsion is done by gradient descent on a KL loss function

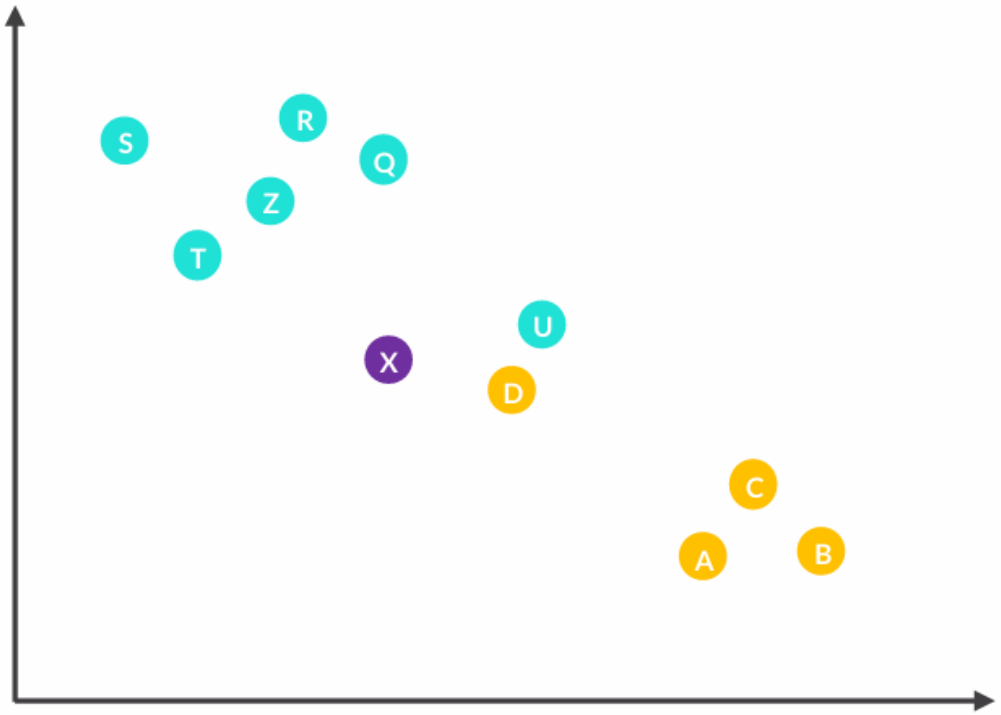
Step 5.3 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



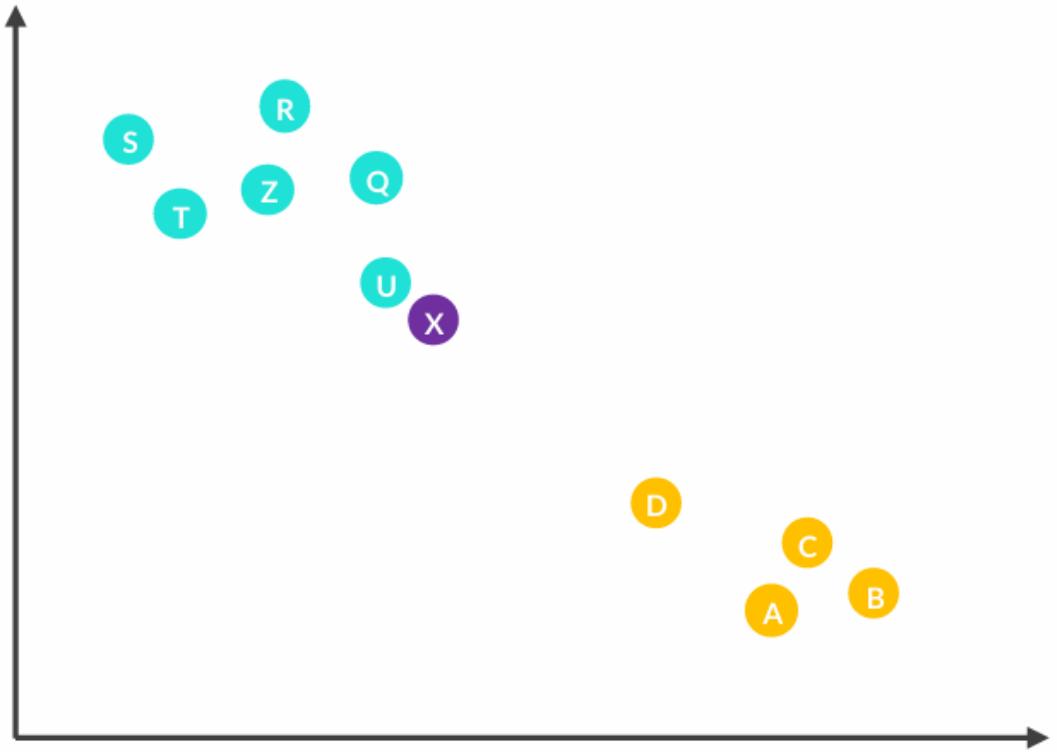
Step 5.4 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



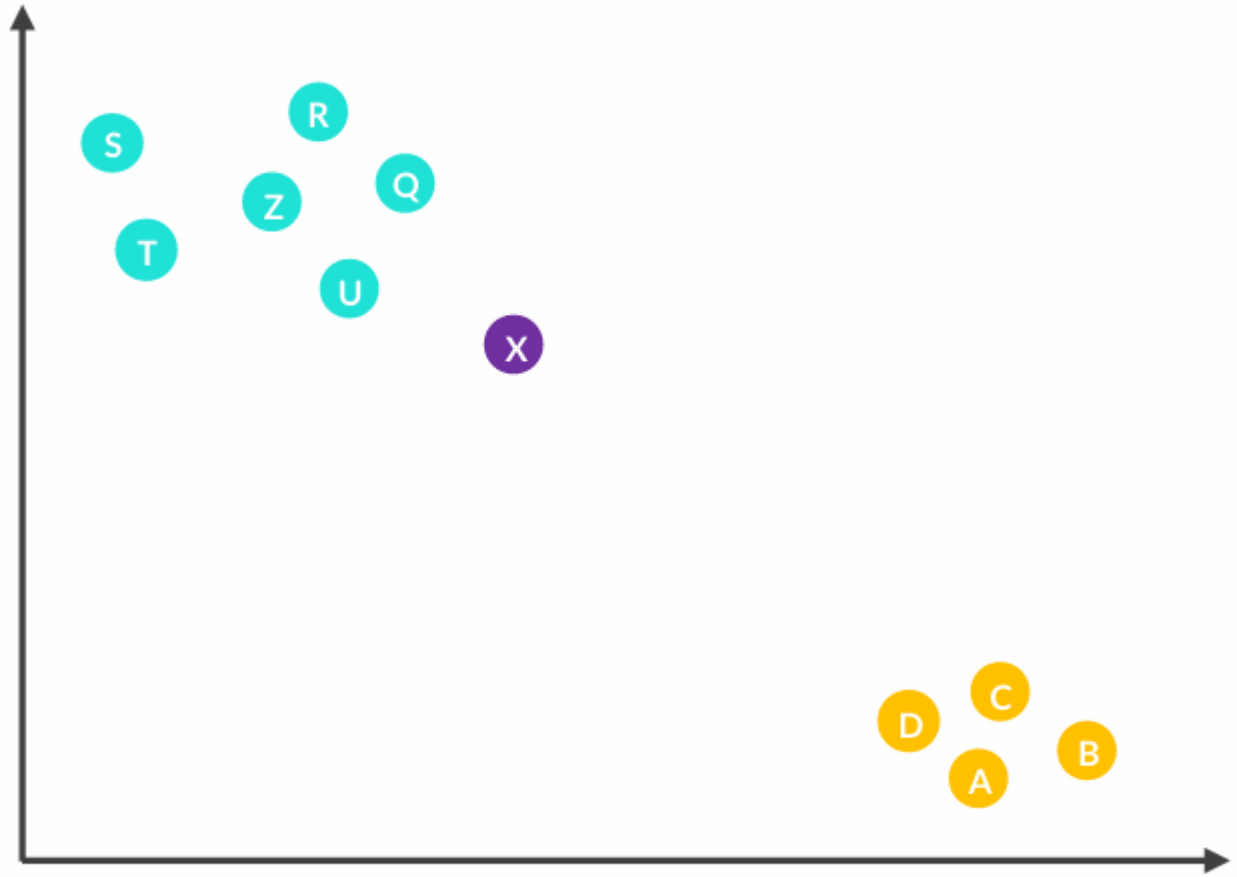
Step 5.5 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



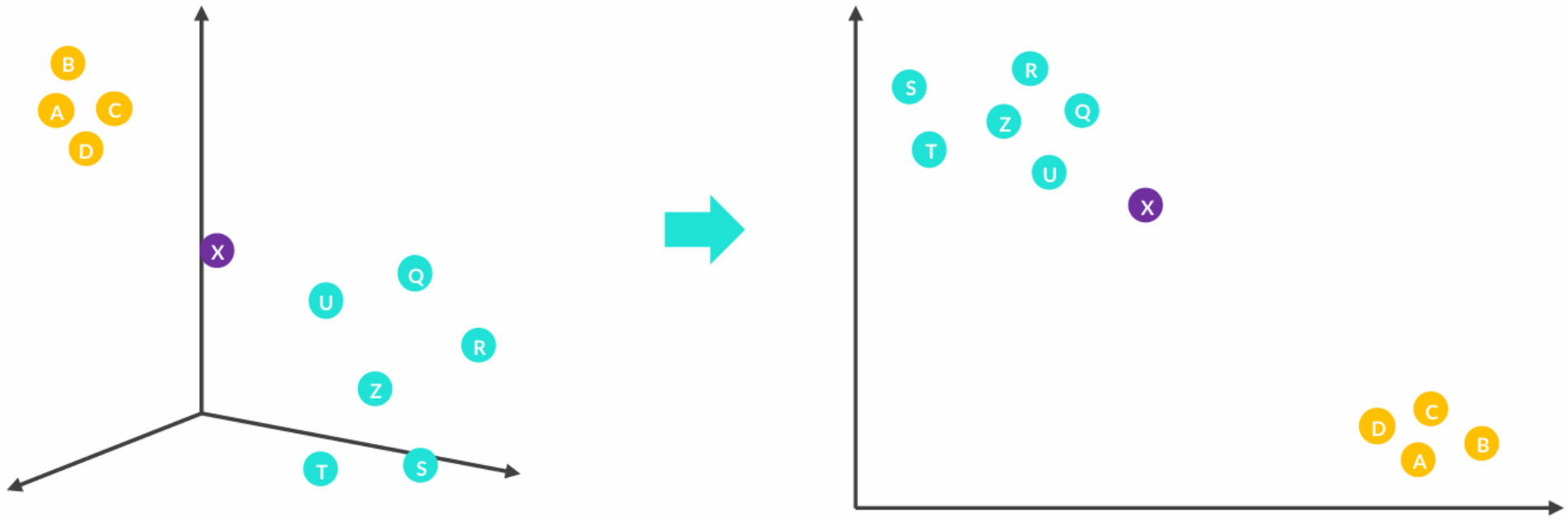
Step 5.6 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



Step 5.7 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



Step 5.8 - Use the affinity scores between pairs to attract (high affinity) or repulse (low affinity) the points until they have "stabilized" into their final position



t-SNE (t-distributed Stochastic Neighbor Embedding) summarization

t-SNE (**t**-distributed **S**tochastic **N**eighbor **E**mbedding) is a powerful **dimensionality reduction** technique used primarily for **data visualization**. Its main purpose is to map high-dimensional data (with many features) to a lower dimension (usually 2D or 3D) so that it can be easily plotted.

Unlike other methods like PCA, which focuses on preserving global variance, t-SNE excels at preserving the **local structure** of the data. This means that data points that are close together in the original high-dimensional space will remain close together in the t-SNE plot, allowing you to visually identify natural clusters or groups in your data.

How it Works ?

The algorithm works in two main stages:

- **Similarity Measurement:** It calculates the similarity between every pair of data points in the high-dimensional space, converting these similarities into probabilities. Points that are close together have a high probability of being neighbors.
- **Optimization:** It then creates a low-dimensional map and iteratively adjusts the positions of the points to make the similarity relationships match as closely as possible. It attracts similar points and repels dissimilar ones until the map stabilizes.
- The final t-SNE plot is a representation of the data's underlying structure, making it an excellent tool for exploratory data analysis to uncover hidden clusters and relationships.

LET’S GO