



1495  
UNIVERSITY OF  
ABERDEEN

CELEBRATING  
**525 YEARS**  
1495 – 2020

ABERDEEN 2040

# Data Dimensionality – 2

Data Mining & Visualisation  
Lecture 22

2025

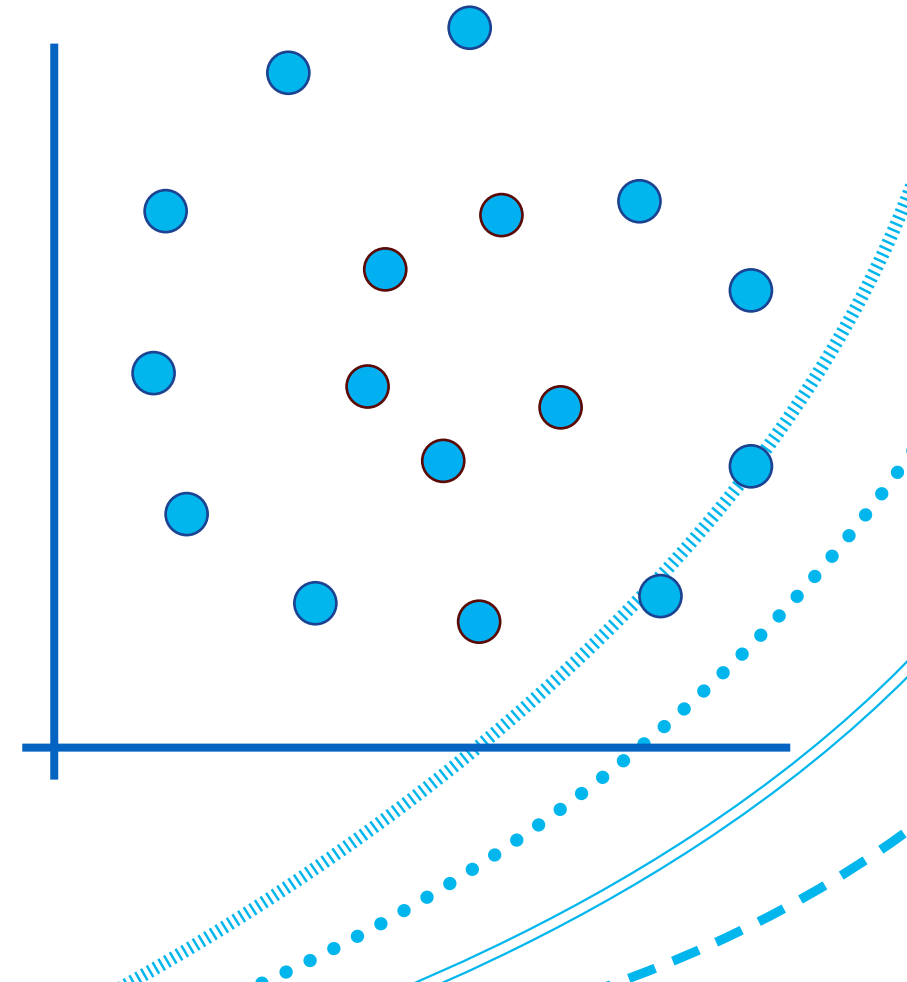
# Today...

- t-SNE
- UMAP
- Summary

# Dimensionality Reduction Techniques

So far, the two dimensionality reduction approaches that we have explored, PCA and LDA, are both *linear* methods.

That is, both methods rely on *linear* projections, and will not work as well when dealing with *non-linear data*.



# t-Distributed Stochastic Neighbour Embedding



# t-SNE

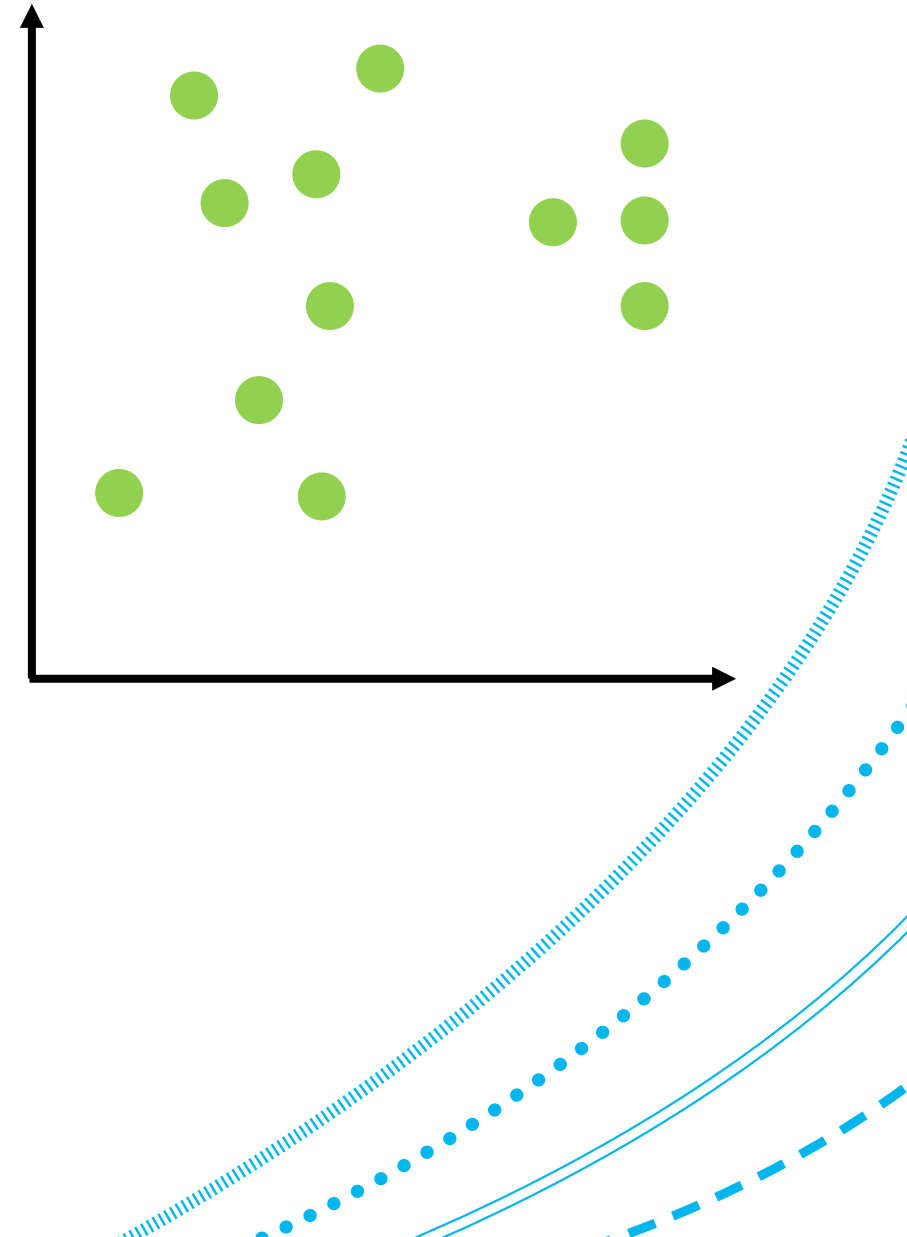
t-Distributed Stochastic Neighbour Embedding (or t-SNE) is a dimensionality reduction technique.

It is often used as an approach for *visualizing and exploring* high-dimensional, non-linear data.

# t-SNE: Step 1

The first thing that t-SNE does is to assess the pairwise similarities between each data point and every other point.

It does this by calculating the **probability** that each datapoint is likely to be a neighbour of that point

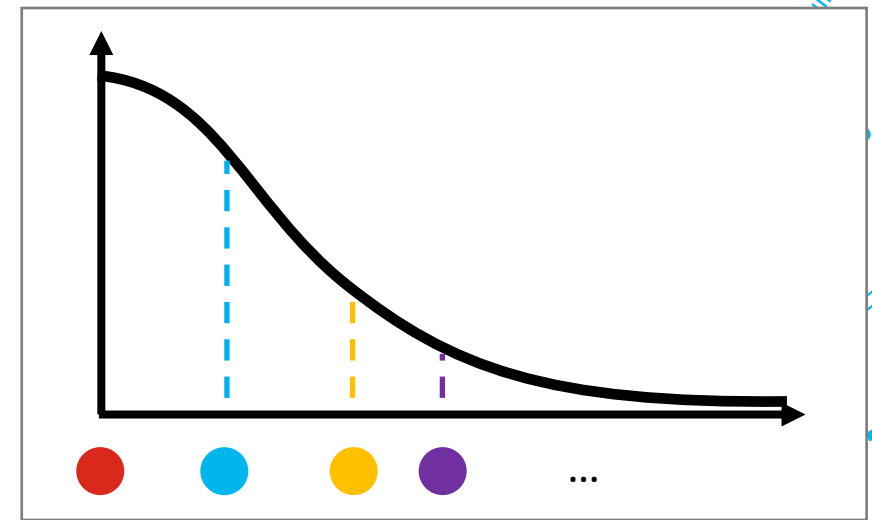
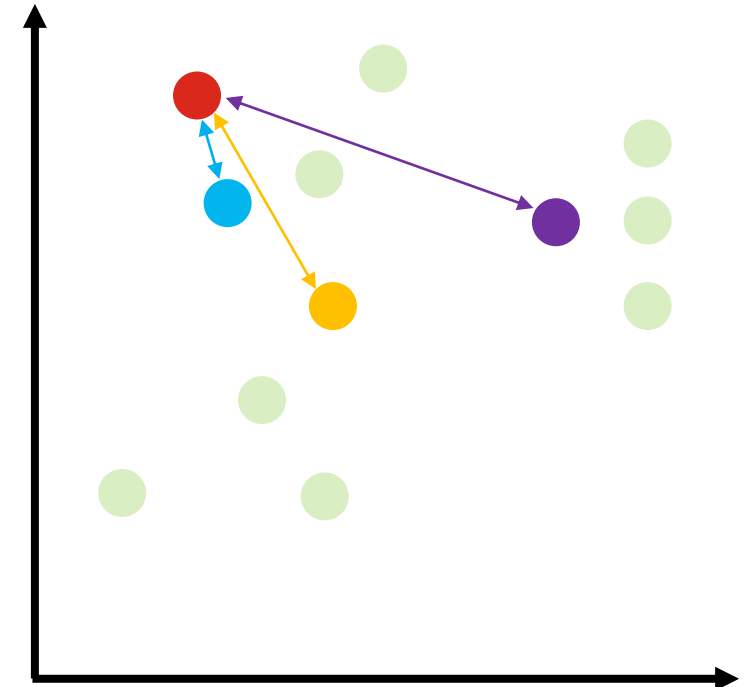


# t-SNE: Step 1

For each point, this process results in a probability distribution based on a Gaussian (normal) distribution, centered around that point.

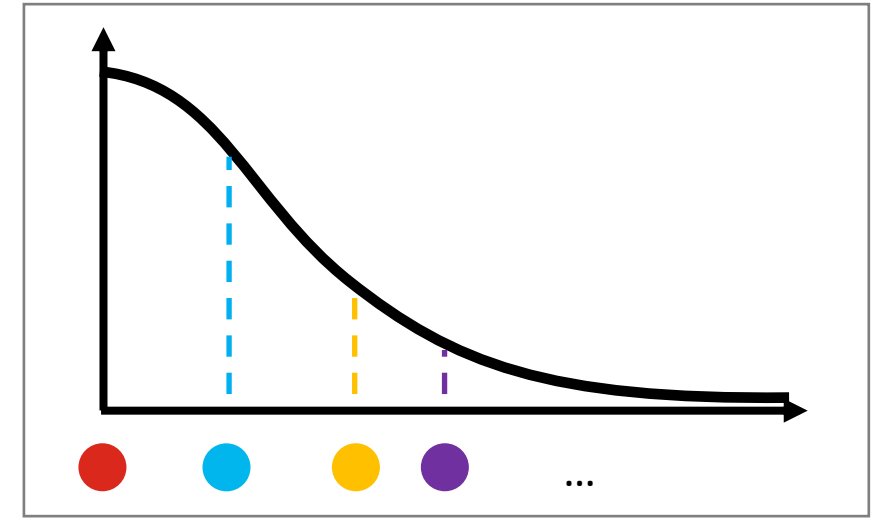
Closer points have a higher probability, whereas farther points have lower probabilities.

Let's just focus on a few points...



# t-SNE: Step 1

These probability values are then normalized, so that the probabilities for each point all add up to 1.



$$1 = P(\text{blue} | \text{red}) + P(\text{yellow} | \text{red}) + P(\text{purple} | \text{red}) + P(\dots)$$

By doing so, we control for the density of the neighbouring points to the point of interest.

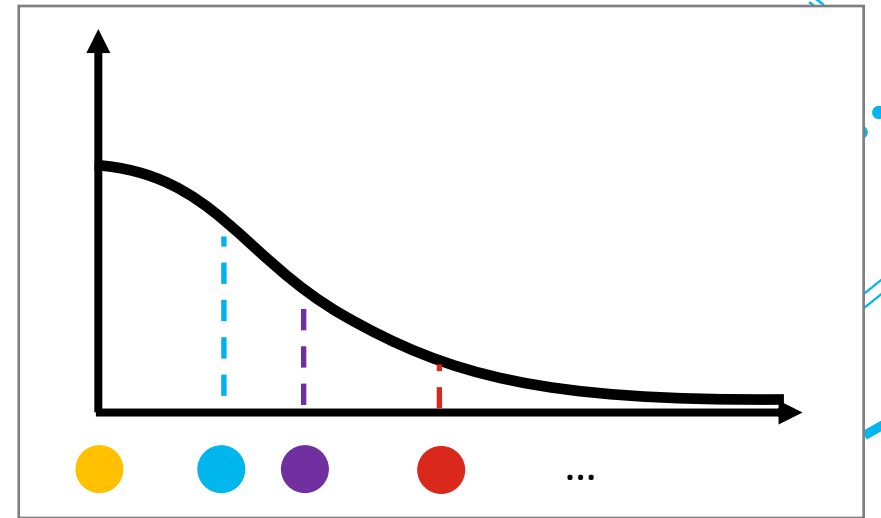
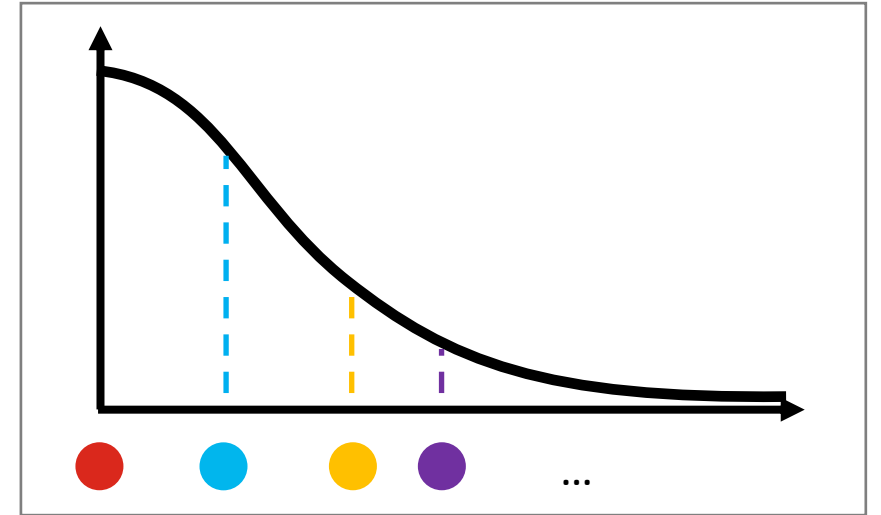


# t-SNE: Step 2

We have a distribution of conditional probabilities for each point.

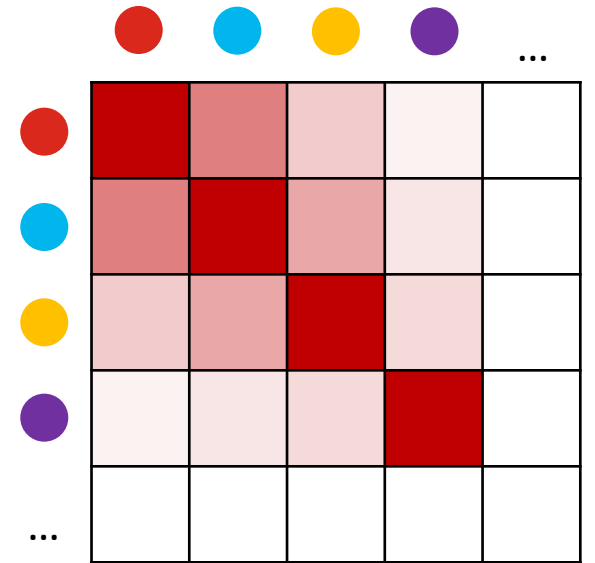
However, since we normalized these,  $p(\text{red}|\text{yellow})$  may not be the same as  $p(\text{yellow}|\text{red})$ .

So we therefore **average** the values between each pair of points to get a *joint probability* per pair.



# t-SNE: Step 2

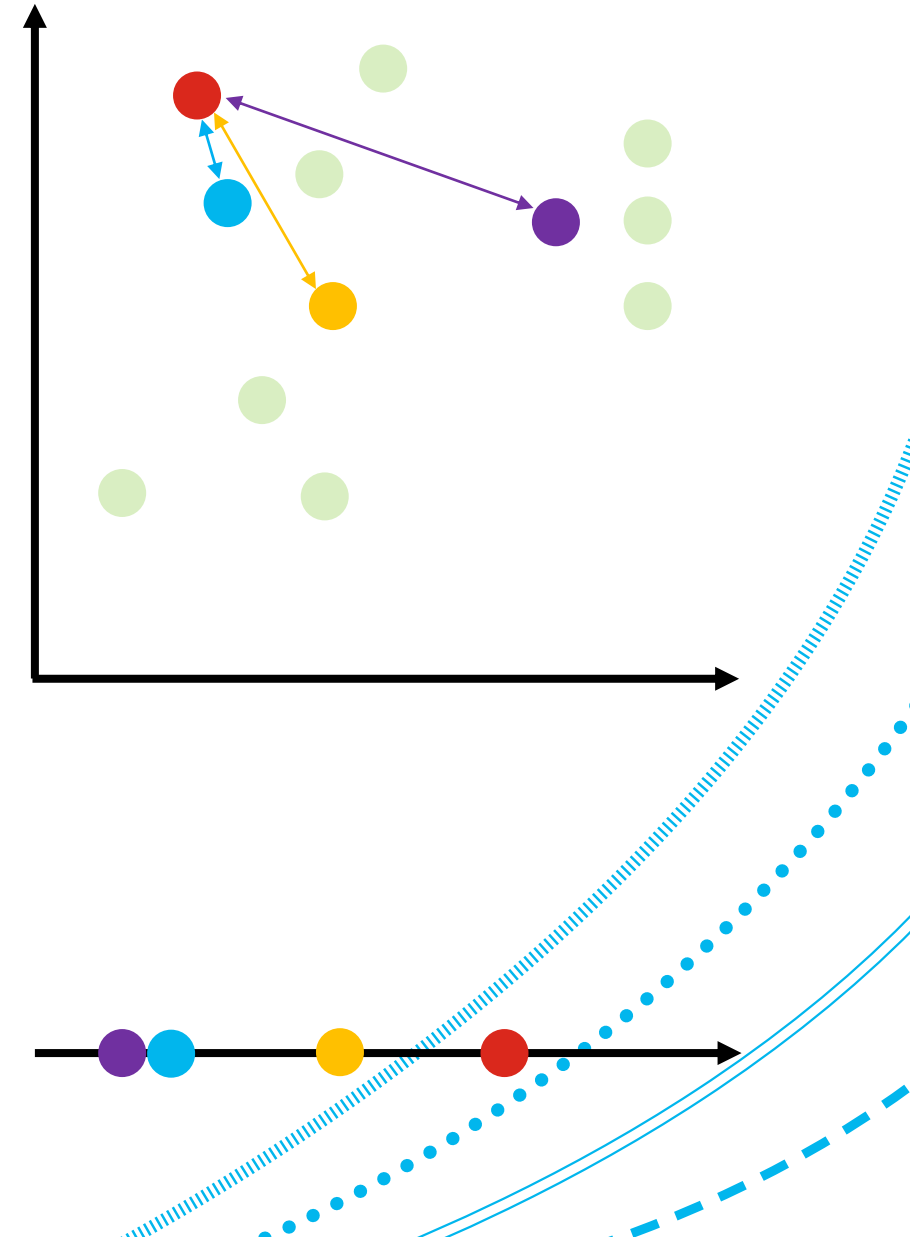
In effect, what this gives us is a proximity matrix, containing the normalized probabilities across all pairs of points.



# t-SNE: Step 3

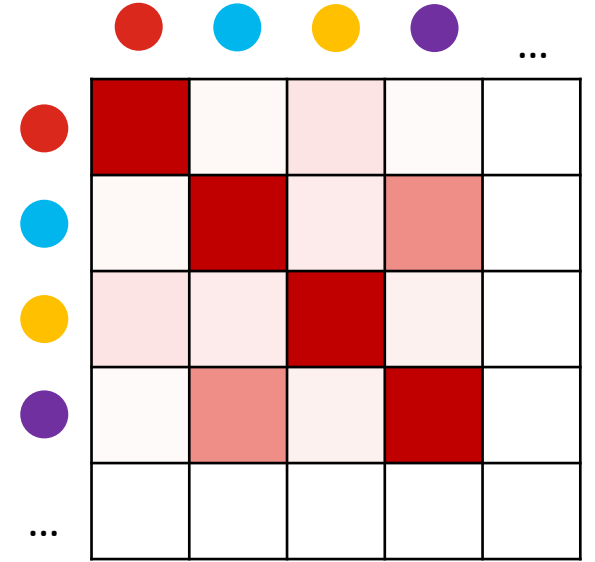
Next, we **randomly** project our datapoints onto our desired number of dimensions, and then compute the similarity scores again.

However, this time, instead of using a Gaussian (normal) distribution, we use a t-distribution (the 't' in t-SNE).



# t-SNE: Step 3

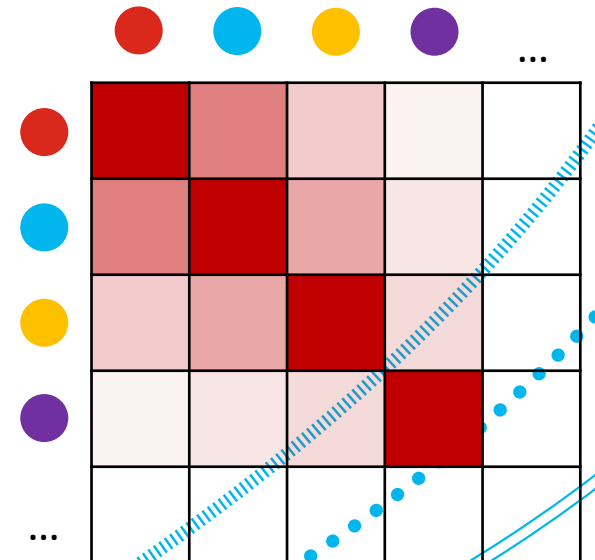
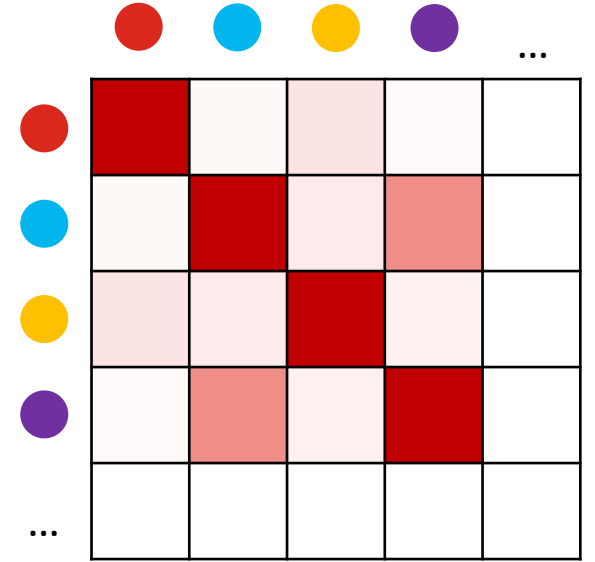
This gives us another proximity matrix, but this time for the points as they are projected onto the lower dimensions.



# t-SNE: Step 3

This gives us another proximity matrix, but this time for the points as they are projected onto the lower dimensions.

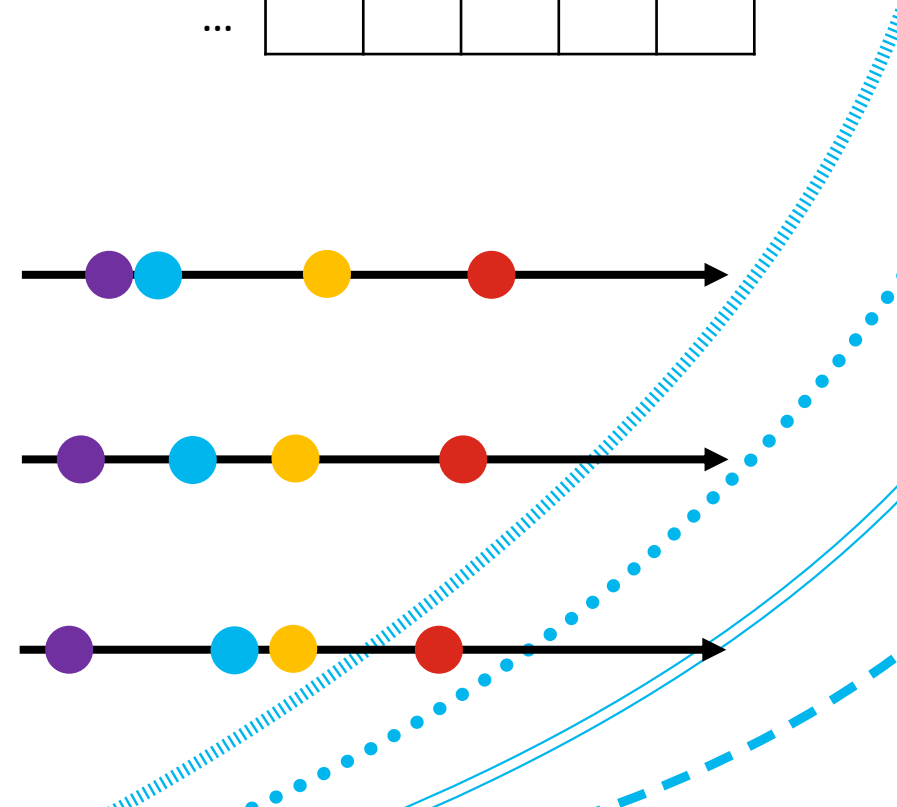
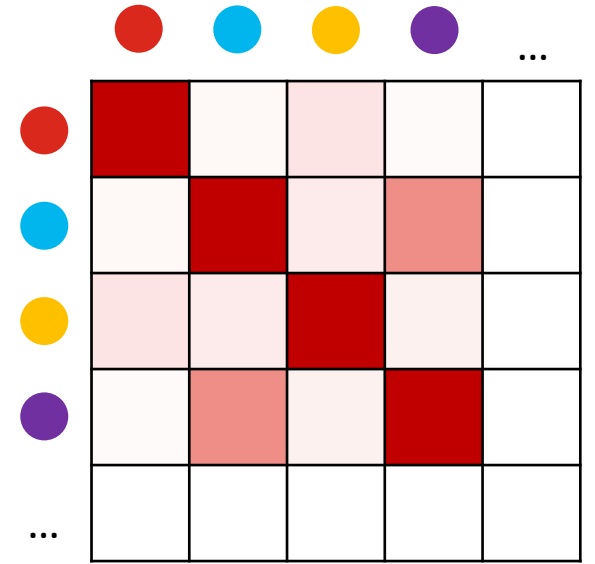
In effect, we want *this* proximity matrix to look more like our *original* proximity matrix.



# t-SNE: Step 4

Next, we slightly adjust each of the points on our lower-dimensional representation, so that it minimizes the difference to the proximity matrix of the higher dimension.

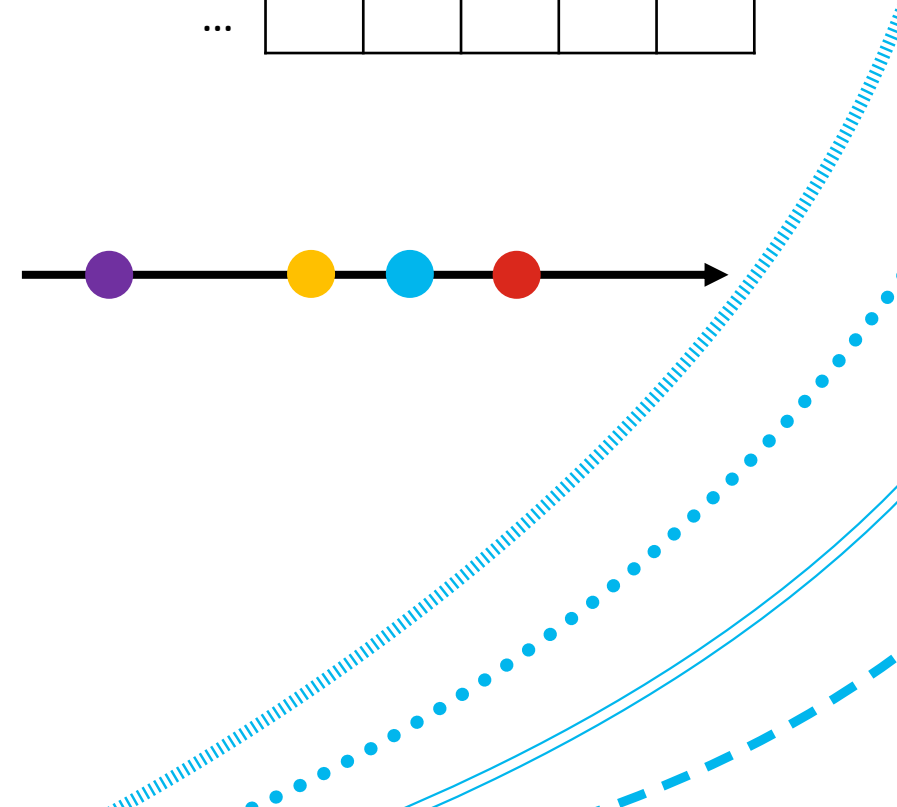
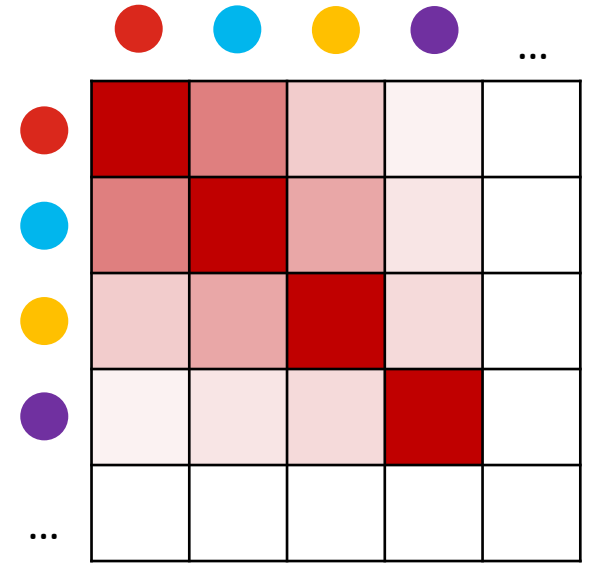
This process is repeated iteratively, using *gradient descent*, usually until we reach a pre-defined number of iterations.



# t-SNE: Step 4

Once the gradient descent process ends, the proximity matrix for our lower-dimensionality representation should more closely match that of the higher-dimensionality representation.

At this point, the process has completed, and we have our lower-dimension representation.



# t-SNE

t-SNE is a very effective process for data visualization, allowing us to map and plot out very high-dimensional data.

Like PCA, it is an unsupervised method, where the visualization will not depend on any class labels or values.

However, these visualisations can often tell us something about labelled data, that we might not have otherwise known.



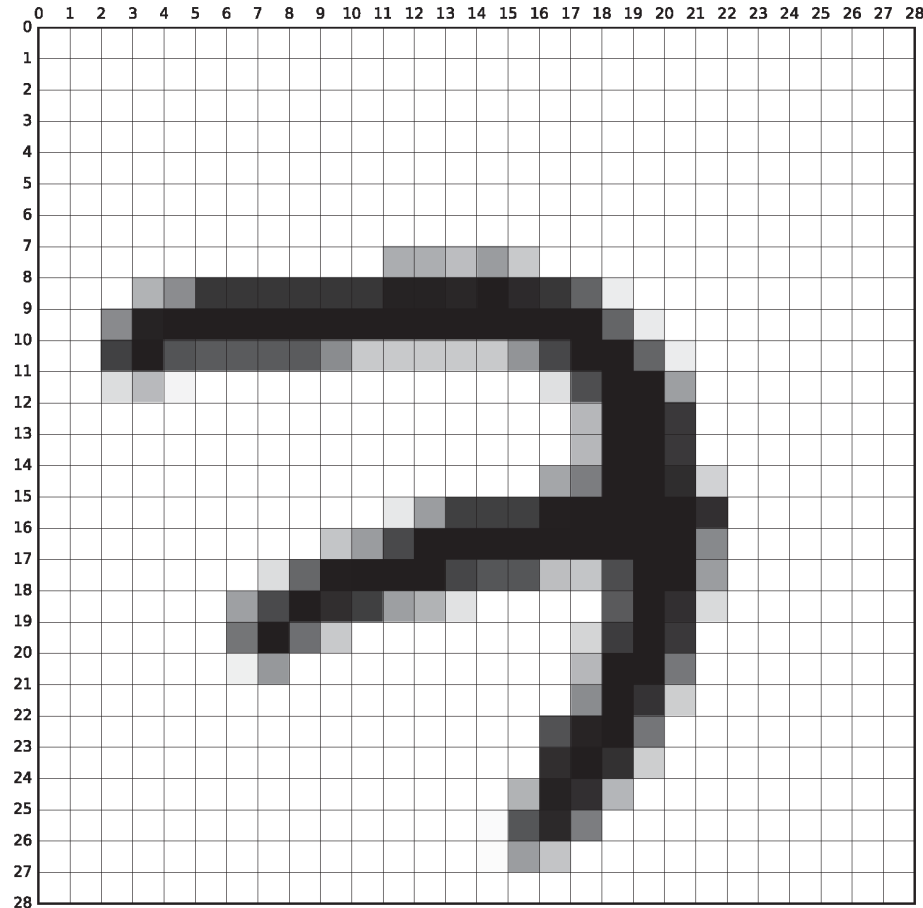
# t-SNE Example – MNIST Dataset

As an example, a very common scenario used to demonstrate t-SNE is on the MNIST dataset.

This very famous dataset contains 70k *labelled* examples of handwritten digits (0-9), in a 28x28 pixel grid.

Each example has been flattened into a 784-dimensional vector, containing a value between 0 and 255, corresponding to that pixel's (grayscale) intensity.

# t-SNE Example – MNIST Dataset



(a) MNIST sample belonging to the digit '7'.

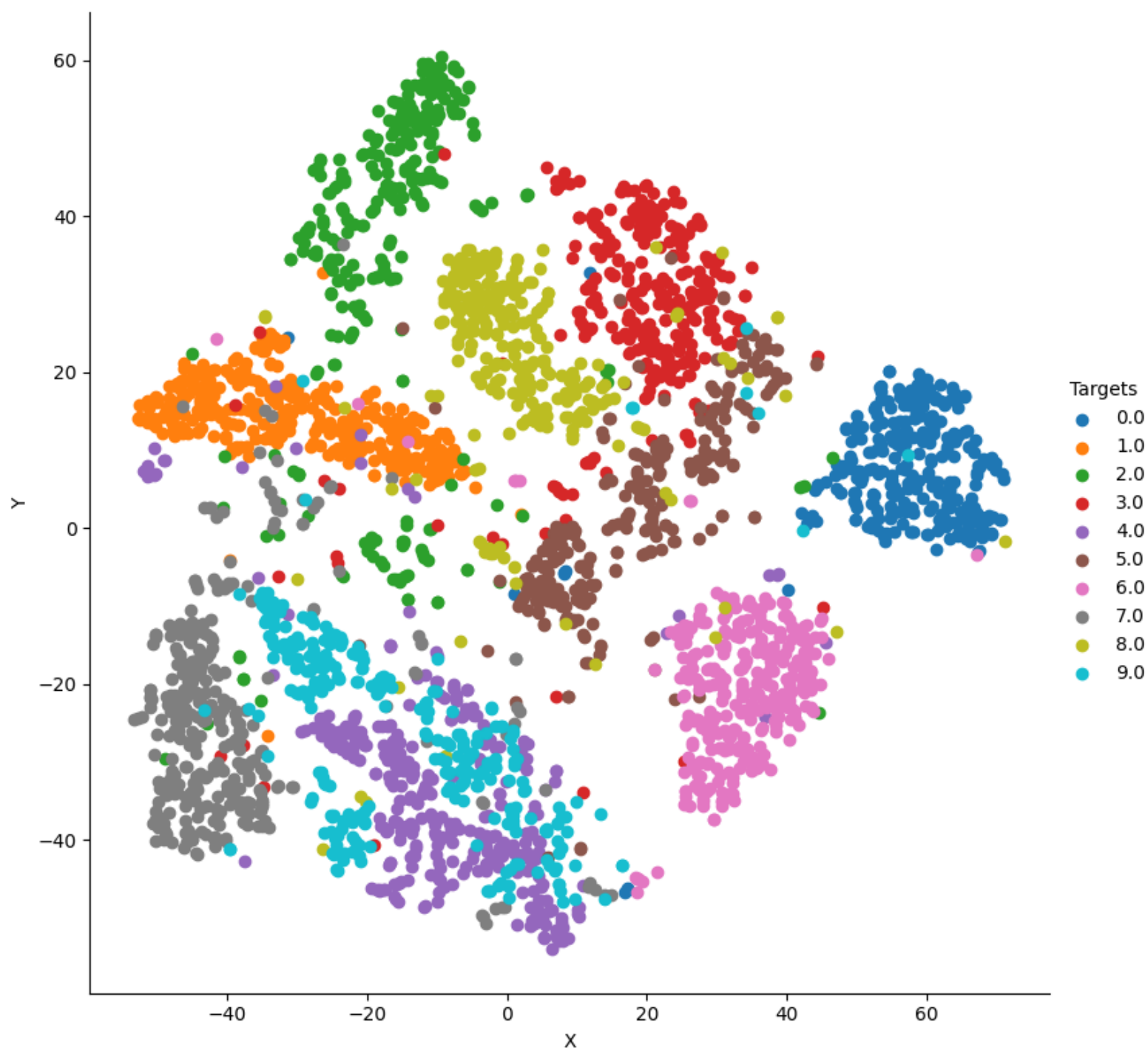


(b) 100 samples from the MNIST training set.

# t-SNE Example –

And if we apply t-SNE to this dataset of 786-dimensional data, and plot the output,

using the class labels to determine the colour of each datapoint...



# t-SNE: Advantages

t-SNE is very **effective for visualizing** high-dimensional data.

Unlike PCA and LDA, it can **capture non-linear relationships**.

It also **preserves the local structure**, keeping similar objects close together.

# t-SNE: Disadvantages

However, t-SNE does result in the **loss of global structure**, in that it doesn't preserve the scale of distances between points.

It is **non-deterministic** (i.e. relying on a stochastic projection), and **relies on the tuning of hyperparameters**.

It is **comparatively slow and computationally more complex** than alternative dimensionality reduction techniques.

# Uniform Manifold Approximation and Projection



# UMAP

Uniform Manifold Approximation and Projection (UMAP) is another approach to non-linear dimensionality reduction.

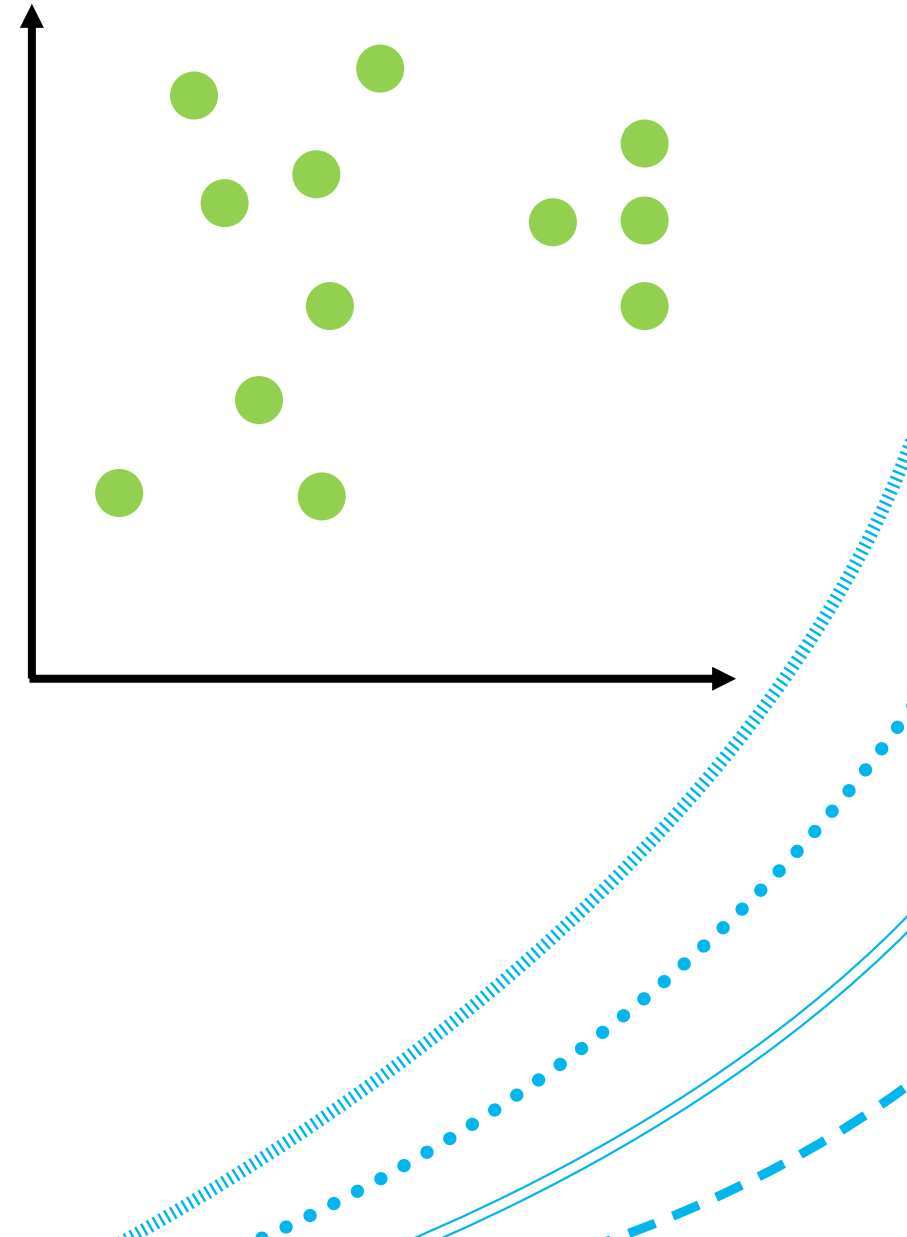
While many of UMAP's steps may appear similar to that of t-SNE, it is fundamentally a different mathematical approach.

In doing so, UMAP manages to address many of t-SNE's major shortcomings, and as a result, it is rapidly becoming a preferred approach to non-linear dimensionality reduction.

# UMAP: Step 1

The first thing that UMAP does, like t-SNE, is to find out which points are close to each other in the high-dimensional space.

However, unlike t-SNE, UMAP does this by using a *nearest neighbour search* to build a *fuzzy topological graph*.

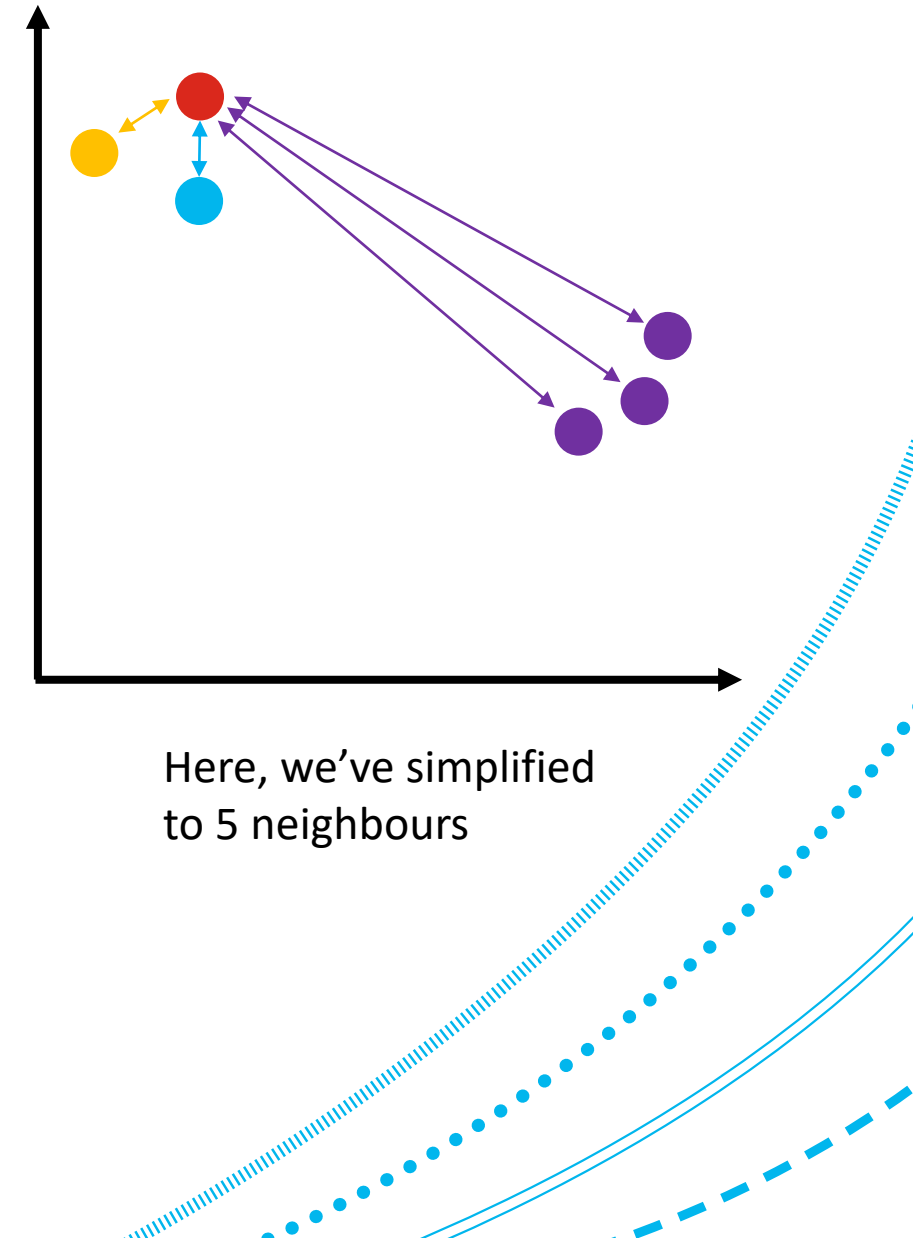




# UMAP: Step 1

So, for each datapoint, we do a nearest neighbour search for its closest 15 or so\* neighbours (using Euclidean or cosine distance).

\* Note that this number is a user-defined hyperparameter.

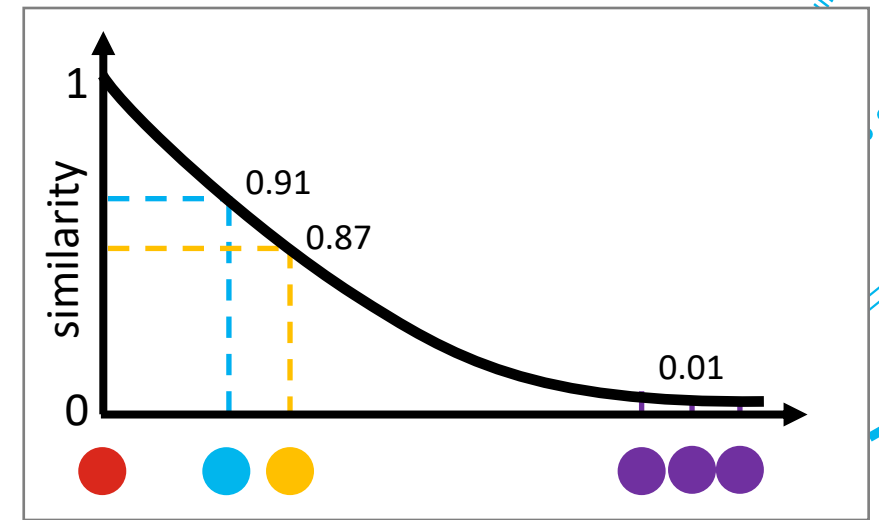
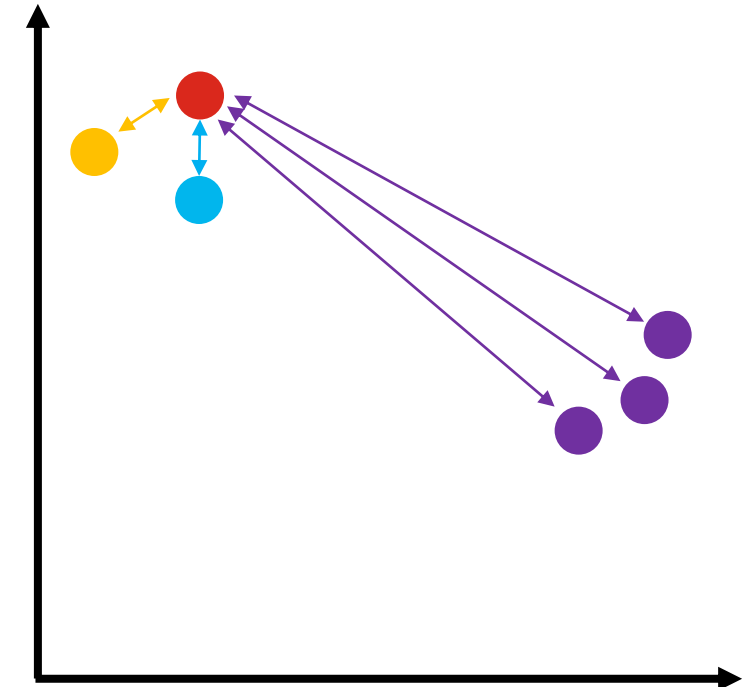


# UMAP: Step 1

For each datapoint, we generate a curve, which assigns fuzzy weights.

In other words, closer points have a weight closer to 1, and further points have a weight closer to 0.

This curve is generated mathematically, based on the number of nearest neighbours.

















# UMAP: Step 2

We then generate a symmetric fuzzy graph, containing the pairwise combination of each pair of points.

This takes into account that the weight of red to blue may not be the same as the weight of blue to red.

**Note that, unlike t-SNE, this is not an average of the two values. It is a specific fuzzy union formula, that is used by UMAP.**

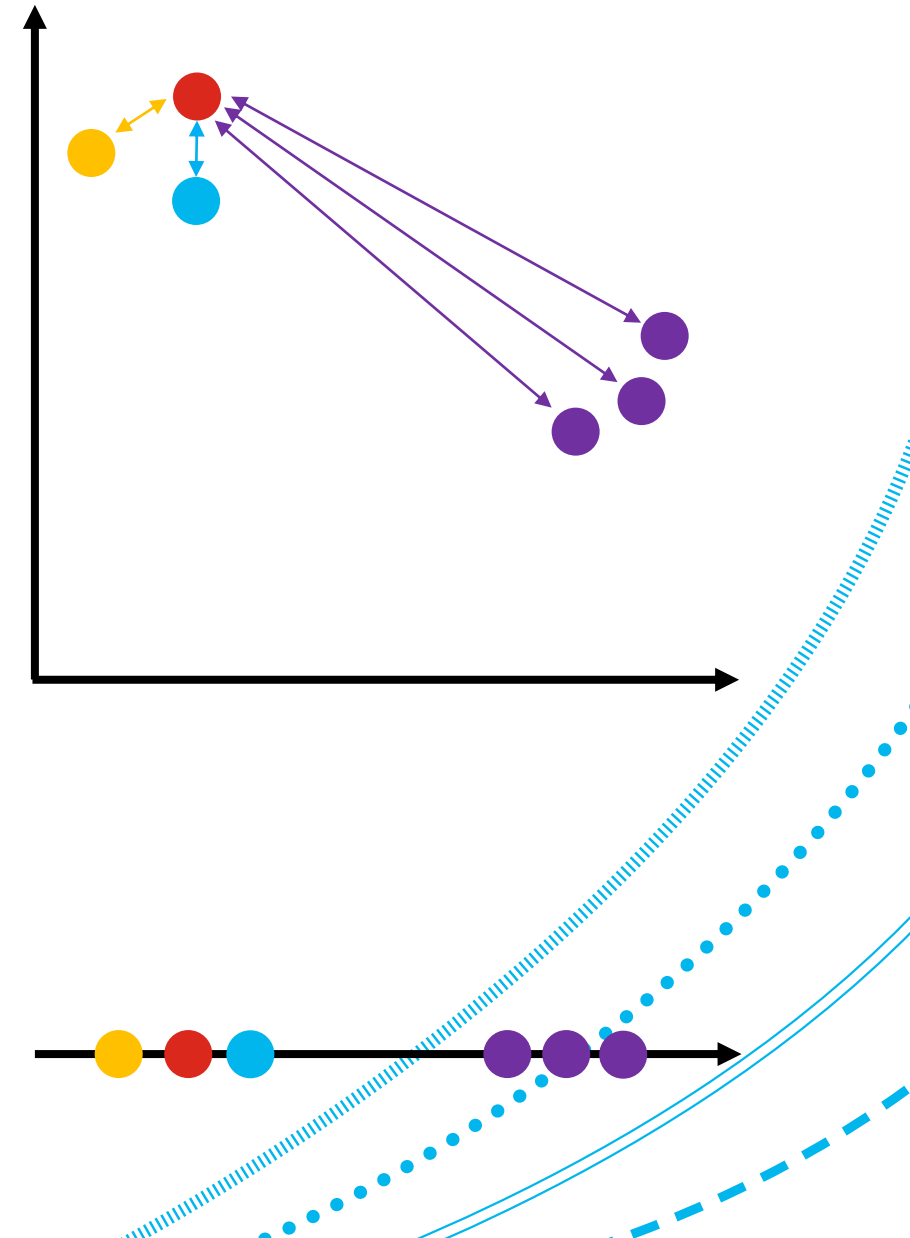
Symmetric Similarity Scores	
 	0.85
 	0.8
 	0.01
 	0.01
 	0.01
 	0.75
 	0.01
...	...

$$\text{Symm Score} = (w1 + w2) - (w1 * w2)$$

# UMAP: Step 3

Next, we project our datapoints onto our desired number of dimensions.

However, instead of doing this randomly (like in t-SNE), **we can use PCA to map to a lower number of dimensions**, to get to a better, deterministic, starting point.



# UMAP: Step 4

Next, using a similar, but not identical, process to that of Step 1 and 2, UMAP then creates a new set of symmetric similarity scores – this time for the low-dimensional mapping.

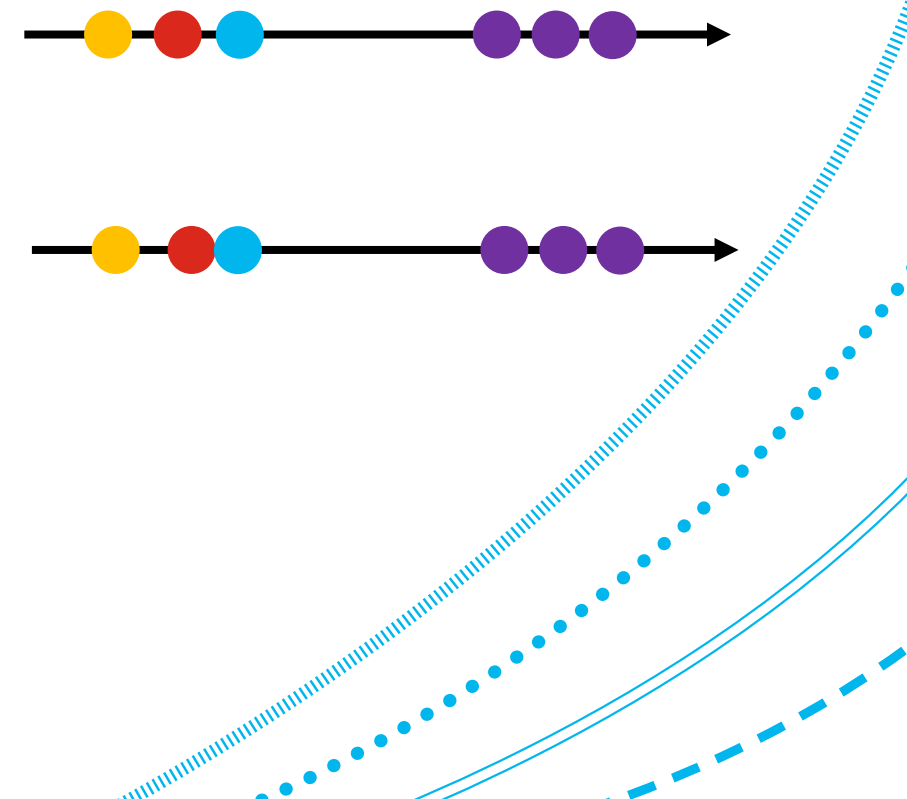
Symmetric  
Similarity Scores

●	●	...
●	●	...
●	●	...
●	●	...
●	●	...
●	●	...
●	●	...
...		...

# UMAP: Step 5

UMAP then randomly selects a pair of datapoints, proportional to their high-dimensional similarity score.

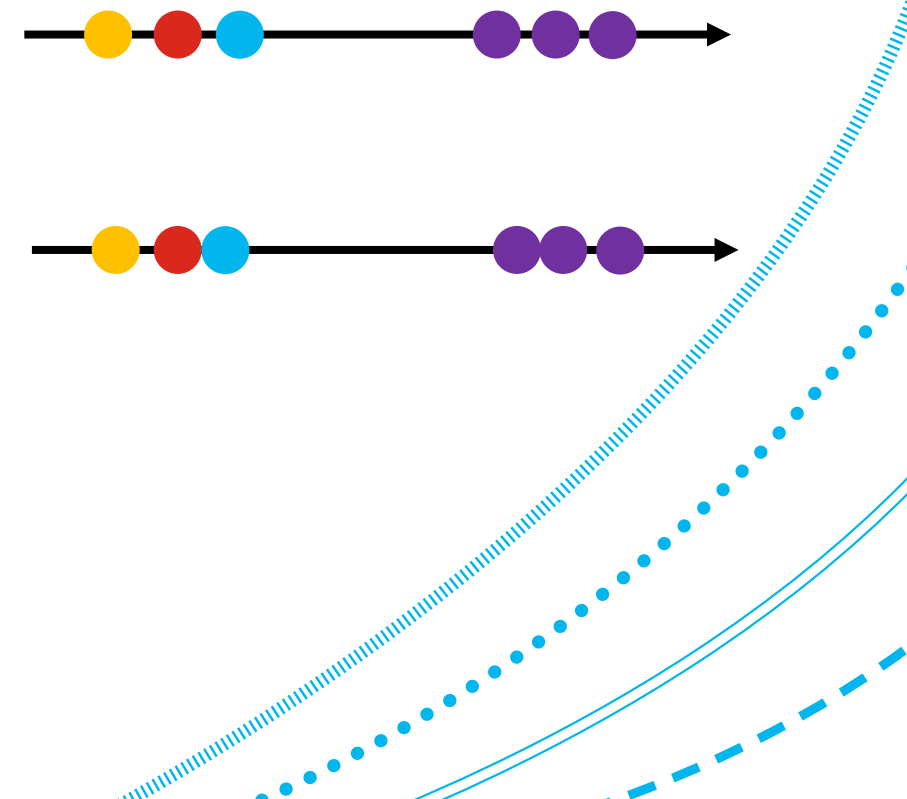
Then, it uses gradient descent to nudge those two points on the lower-dimensionality mapping closer together.



# UMAP: Step 5

In parallel, UMAP also randomly selects points that aren't neighbours, and uses gradient descent to nudge them apart.

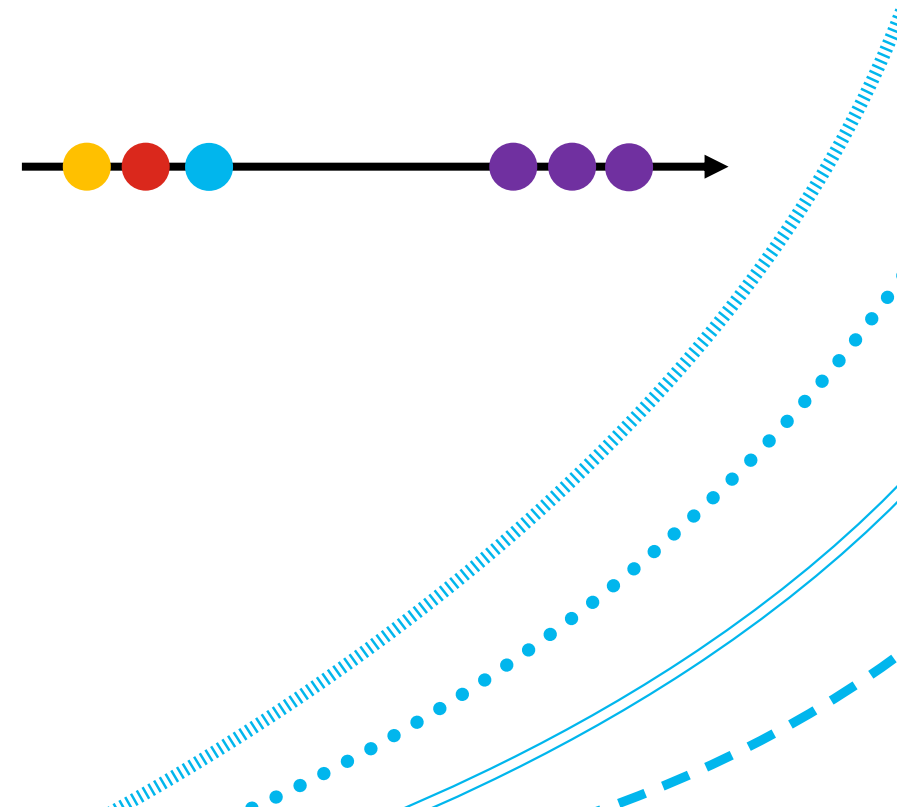
The result is that our low-dimensionality representation reflects the overall structure of the high-dimensional data, while simultaneously avoiding crowding.



# UMAP: Step 5

This pulling and pushing process is iterative, and repeats for a set number of iterations (usually ~200 by default).

After this process is complete, UMAP is finished, and we receive our low-dimensional data as the output.





# UMAP: Advantages

Compared to t-SNE, UMAP is **significantly faster and more scalable**, particularly when dealing with large, complex data.

It also **preserves more global structure** than t-SNE, which means the output is more suitable for downstream tasks (further analysis, etc.).

UMAP also becomes **deterministic** when used with a fixed seed value, which is not always the case with t-SNE implementations (e.g. sklearn).

# UMAP: Disadvantages

Again, UMAP does require the **tuning of certain hyperparameters**.

Despite it being comparatively fast (compared to t-SNE), its complexity means that it is **not always the best solution**, particularly for very small, or non-linear, datasets.

Its complexity and non-linear nature also means that the **explainability of low-dimensional representations** can still pose a challenge.

# Summary



# Summary

Dimensionality reduction gives us a way to reduce high-dimensional data, while preserving its important information.

One of the crucial advantages of dimensionality reduction is that it can make very high-dimensional datasets easier to visualise and explore.

There are a range of implementations for applying this technique to a wide variety of datasets.

# Summary

Using dimensionality reduction *as a pre-processing technique* can also lead to improved computational efficiency on downstream tasks, through simplifying the data.

And by removing some variance, it can also be used to help reduce model overfitting, leading to better generalisation.