

# Deep Unsupervised Learning (Overview)

Naeemullah Khan

[naeemullah.khan@kaust.edu.sa](mailto:naeemullah.khan@kaust.edu.sa)



جامعة الملك عبدالله  
للعلوم والتكنولوجيا  
King Abdullah University of  
Science and Technology

KAUST Academy  
King Abdullah University of Science and Technology

June 10, 2025

# Table of Contents

1. Definition
2. Applications
3. Challenges
4. Types
  - 4.1 Clustering
  - 4.2 Dimensionality Reduction
  - 4.3 Anomaly Detection
5. Dimensionality Reduction
  - 5.1 PCA
  - 5.2 t-SNE
  - 5.3 UMAP

- ▶ **Supervised learning** requires large labeled datasets, which are often unavailable in real-world scenarios.
- ▶ **Deep Unsupervised Learning** enables discovery of hidden structures, patterns, and representations in data **without labeled outputs**.
- ▶ Systems can learn autonomously from vast amounts of raw, unstructured data.
- ▶ Understanding unsupervised methods (clustering, dimensionality reduction, anomaly detection) helps:
  - Unlock insights missed by supervised approaches
  - Reduce dependency on human annotation

# Learning Outcomes

By the end of this presentation, you will be able to:

1. **Define** deep unsupervised learning and **differentiate** it from supervised and semi-supervised approaches.
2. **Understand** the importance and real-world applications of unsupervised learning in:
  - Natural language processing
  - Computer vision
  - Fraud detection
3. **Identify** key challenges in unsupervised learning:
  - Model evaluation
  - Lack of labeled ground truth

**4. Classify** main types of unsupervised learning tasks:

- Clustering
- Dimensionality reduction
- Anomaly detection

**5. Explain** core dimensionality reduction techniques:

- Principal Component Analysis (PCA)
- t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Uniform Manifold Approximation and Projection (UMAP)

**6. Appreciate** the role of deep learning in enhancing traditional unsupervised methods through representation learning.

# Unsupervised Learning - Definition

- ▶ We have a dataset without labels. Our goal is to learn something interesting about the underlying structure of the data:
  - Clusters hidden in the dataset.
  - Outliers: particularly unusual and/or interesting data points.
  - Useful signals hidden in the noise, e.g., human speech over a noisy background.

# Components of Unsupervised Learning



- ▶ **Data:** Unlabeled data, e.g., images, text, or sensor readings.
- ▶ **Model:** A mathematical representation of the data, e.g., a mixture model or a neural network.
- ▶ **Objective function:** A measure of how well the model fits the data, e.g., likelihood or reconstruction error.
- ▶ **Optimization algorithm:** An algorithm to minimize the objective function, e.g., gradient descent or expectation-maximization.
- ▶ **Evaluation metrics:** Measures to assess the quality of the learned model, e.g., silhouette score or clustering accuracy.
- ▶ **Applications:** Use cases for unsupervised learning, e.g., clustering, dimensionality reduction, or anomaly detection.

# Supervised vs Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Objective	Learn a function $f$ from labeled input–output pairs.	Discover structure or representations in unlabeled data.
Evaluation	Accuracy, precision/recall on held-out labels.	Clustering validity indices (e.g. silhouette), reconstruction error.
Cost	Methods range from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$ per fit.	k-means $\mathcal{O}(nkd)$ , hierarchical $\mathcal{O}(n^2)$ , PCA $\mathcal{O}(nd^2)$ .
Labels/Clusters	Fixed, known set of classes.	Number of clusters unknown; must be chosen or inferred.
Output	Classifier or regressor for new inputs.	Cluster assignments, embeddings, density models, or generative samples.

Table 1: Key differences between Supervised and Unsupervised Learning

Unsupervised learning is used in various fields and applications, including:

- ▶ **Visualisation:** Identifying and making accessible useful hidden structures in the data.
- ▶ **Anomaly Detection:** Identifying factory components that are likely to break soon.
- ▶ **Signal denoising:** Extracting human speech from a noisy recording.
- ▶ **Generative Models:** Learning to generate new data points similar to the training data.
- ▶ **Feature Learning:** Automatically discovering useful representations of the data.
- ▶ **Data Preprocessing:** Cleaning and transforming data for better performance in supervised learning tasks.

# Application: Discovering Structure in Digits

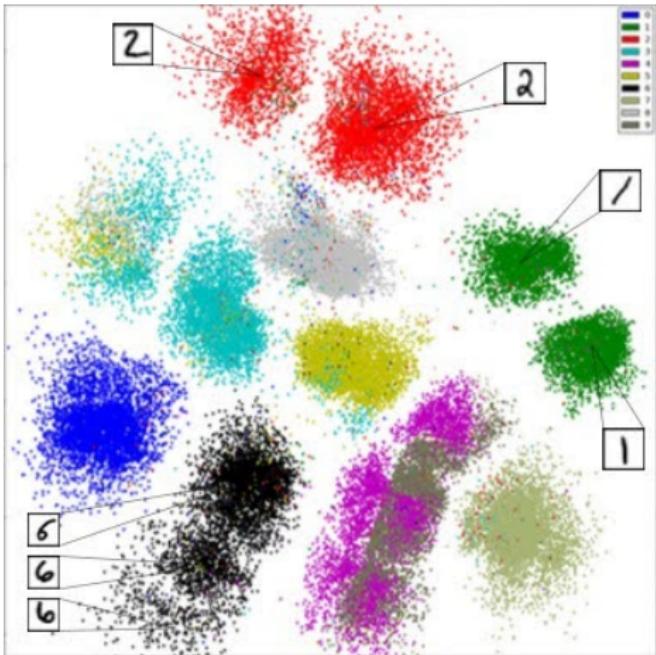


Figure 2: Unsupervised learning can discover structure in digits without any labels.

# Application: DNA Analysis

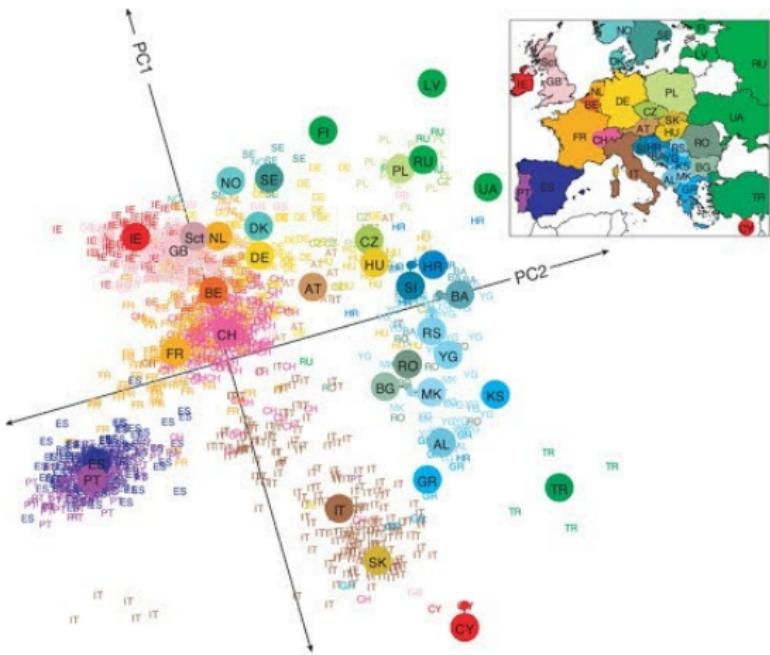


Figure 3: Dimensionality reduction applied to DNA reveal the geography of European countries.

# What is Deep Unsupervised Learning?



# What is Deep Unsupervised Learning? (cont.)

- ▶ Capturing rich patterns in raw data with deep networks in a label-free way.

# What is Deep Unsupervised Learning? (cont.)

- ▶ Capturing rich patterns in raw data with deep networks in a label-free way.
  - **Generative Models:** Recreate raw data distribution.

Why is unsupervised learning challenging?

- ▶ **Exploratory data analysis:** Unsupervised learning is often used for exploratory data analysis, where the goal is to discover patterns or structures in the data without any prior knowledge of the labels.
- ▶ **Difficult to assess performance:** Evaluating the performance of unsupervised learning algorithms can be challenging, as there are no ground truth labels to compare against ("right answer" unknown).
- ▶ **Sensitivity to noise:** Unsupervised learning algorithms can be sensitive to noise and outliers in the data, which can lead to misleading results.
- ▶ **Curse of dimensionality:** As the number of features increases, the data becomes sparse, making it difficult to find meaningful patterns.

## ► Cluster Analysis:

- For identifying homogenous subgroups of samples.
- **Examples:** K-means, hierarchical clustering, DBSCAN.

## ► Dimensionality Reduction:

- For finding a low-dimensional representation to characterize and visualize the data.
- Reducing the number of features in a dataset while preserving important information.
- **Examples:** PCA, t-SNE, UMAP.

## ► Anomaly Detection:

- **Finding outliers in the dataset:** Identifying unusual (rare items, events, or observations) data points that do not conform to expected patterns.
- **Examples:** Isolation Forest, One-Class SVM, Autoencoders.

A set of methods for finding subgroups within the dataset.

- ▶ Observations should share common characteristics within the same group, but differ across groups.
- ▶ Groupings are determined from attributes of the data itself — differs from classification.



Figure 4: Taking a 2 dimensional dataset and separating it into 3 distinct clusters.  
[\[Source\]](#)

**Input:** Dataset  $D = \{x_1, x_2, \dots, x_n\}$ , number of clusters  $k$

**Output:** Cluster assignments for each data point

**Initialization:** Randomly initialize  $k$  cluster centroids or seeds;

**repeat**

**Assignment Step:** Assign each data point  $x_i$  to the nearest cluster based on a distance metric;

**Update Step:** Recompute cluster centroids using current assignments;

**until** convergence or maximum iterations reached;

**return** Final cluster assignments;

**Algorithm:** Generic Clustering Algorithm

# Clustering Vs Classification

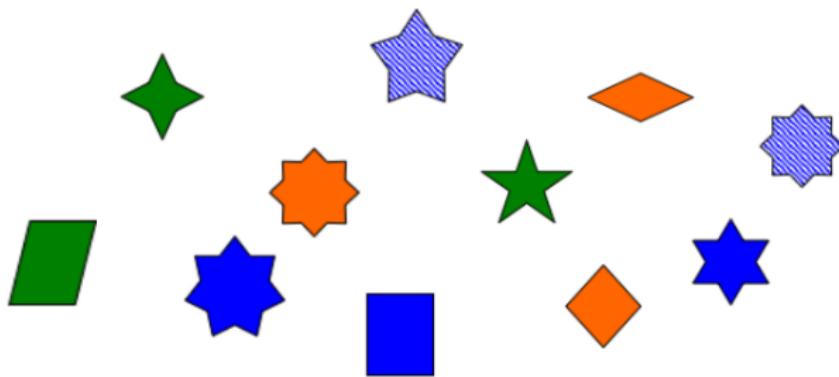


Figure 5: Sample data points.

# Clustering Vs Classification (cont.)

## Classification

- ▶ Labels available
- ▶ Assigning to known classes
- ▶ Supervised

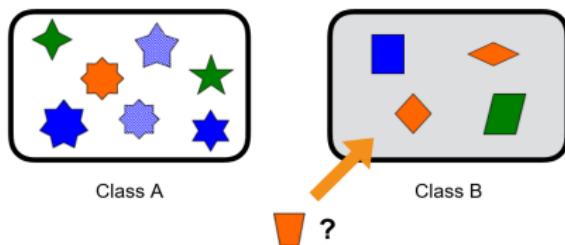


Figure 6: Classification result.

## Clustering

- ▶ No labels
- ▶ Grouping based on similarity
- ▶ Unsupervised

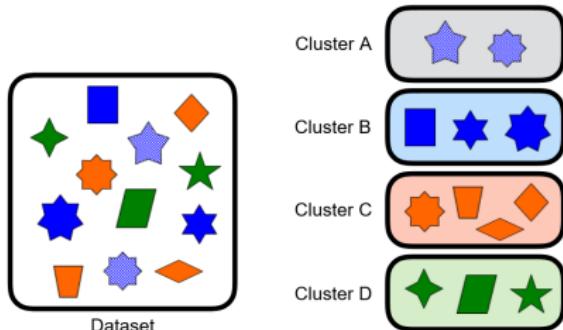


Figure 7: Clustering result.

- ▶ **Centroid-Based Clustering:** Groups data points based on their proximity to a central point, such as K-means or K-medoids.
- ▶ **Hierarchical Clustering:** Builds a hierarchy of clusters using either agglomerative (bottom-up) or divisive (top-down) approaches.
- ▶ **Model-Based Clustering:**
  - Each cluster is represented by a parametric distribution.
  - Dataset is a mixture of distributions.
  - Assumes a probabilistic model for the data and uses statistical methods to identify clusters, such as Gaussian Mixture Models (GMM).
- ▶ **Hard Clustering:**
  - Each data point is assigned exclusively to exactly one cluster.
  - **Example algorithms:** K-means, Hierarchical clustering.

- **interpretation:** No ambiguity — clusters are crisp and non-overlapping.

## ► Soft/Fuzzy Clustering:

- Each data point can belong to multiple clusters simultaneously with varying degrees of membership (probabilities or weights).
- **Example algorithms:** Gaussian Mixture Models (GMM), Fuzzy C-means.
- **interpretation:** Reflects uncertainty or mixed membership — clusters can overlap.

# Clustering - K-means

Groups data into  $K$  clusters that satisfy two properties.

1. Each observation belongs to at least one of the  $K$  clusters.
2. Clusters are non-overlapping. No observation belongs to more than one cluster.

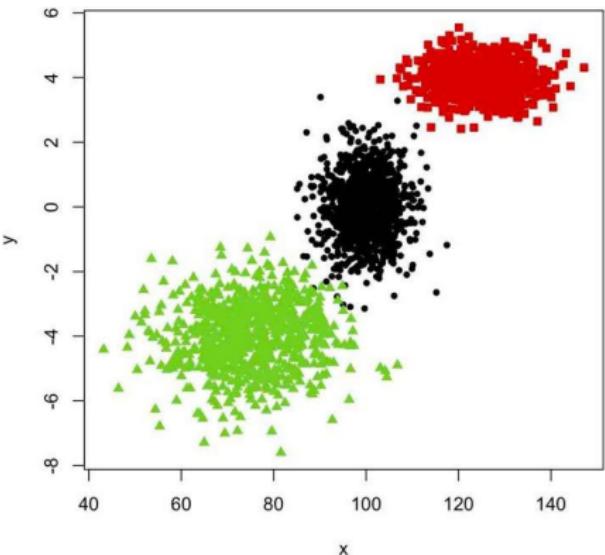


Figure 8: Clusters.

# Clustering - K-means (cont.)

A good clustering is one for which the *within-cluster variation* is as small as possible.

Denote each cluster by  $C_k$ , and let  $W(C_k)$  be a measure of the within-cluster variation.

K-means aims to solve the following optimization problem:

$$\underset{C_1, \dots, C_K}{\text{minimise}} \left\{ \sum_{k=1}^K W(C_k) \right\} \quad (1)$$

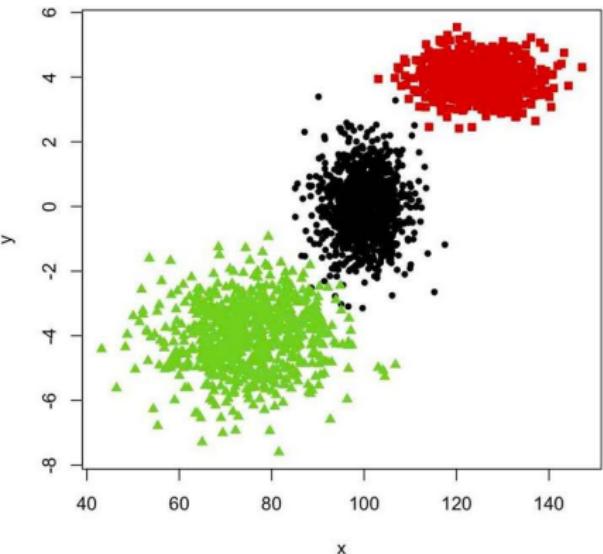


Figure 9: Clusters.

# Clustering - K-means (cont.)

How to measure within-cluster variation?

The most common choice is squared Euclidean distance:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \quad (2)$$

where  $|C_k|$  is the number of points in cluster  $C_k$  and  $x_{ij}$  is the  $j^{th}$  feature of the  $i^{th}$  point.

Which means overall we solve:

$$\underset{C_1, \dots, C_K}{\text{minimise}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\} \quad (3)$$

# Clustering - K-means (cont.)

- ▶ It turns out that this optimization problem is difficult to solve, as it is discrete and there are nearly  $K^n$  ways to split  $n$  samples into  $K$  clusters.
- ▶ In practice, use an iterative algorithm that finds a local minimum to this optimization.

# Clustering - K-means (cont.)

**Input:** Dataset  $D = \{x_1, x_2, \dots, x_n\}$ , number of clusters  $k$

**Output:** Cluster assignments for each data point

**Initialization:** Randomly initialize  $k$  cluster centroids or seeds;

**Repeat until convergence:**

- ▶ **Assignment Step:** Assign each data point  $x_i$  to the nearest cluster based on a distance metric;
- ▶ **Update Step:** Recompute cluster centroids using current assignments;
- ▶ **Convergence Check:** Check if cluster assignments have changed or if centroids have stabilized;

**Return:** Final cluster assignments and centroids;

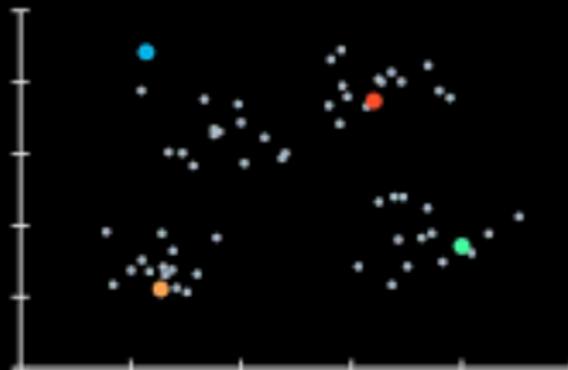
**Algorithm:** K-means Clustering Algorithm

# Clustering - K-means (cont.)

Watch the K-means clustering algorithm in action:

## K-Means Algorithm

1. Initialize centroid positions ( $k = 4$ )
2. Assign labels ( $\mu$ ) to all data ( $X$ )  
 $\mu_n = \underset{i}{\operatorname{argmin}} \|x_n - c_i\|$



# Clustering - K-means (cont.)

1. It can be shown that the value of the objective function will never increase at each iteration of  $k$ -means.
2. Since the algorithm finds local minima, however, it will result in different clusters with different initializations.

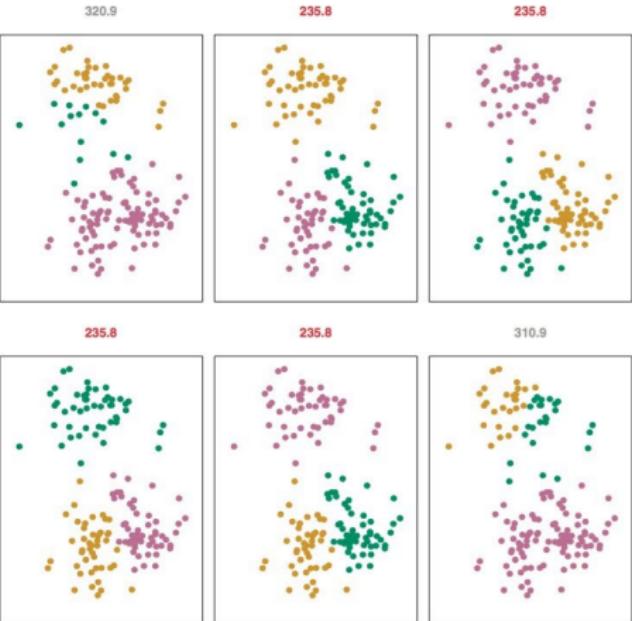


Figure 10: Different initializations of K-means.

## Pros

- ▶ Simple and easy to implement
- ▶ Efficient for large datasets
- ▶ Works well with spherical clusters
- ▶ Scalable to large datasets

## Cons

- ▶ Not robust to data perturbations and different initializations
- ▶ Sensitive to initial centroid placement
- ▶ Assumes spherical clusters
- ▶ Requires specifying the number ' $K$ ' of clusters in advance
- ▶ Sensitive to outliers
- ▶ May converge to local minima
- ▶ Not suitable for non-convex shapes

# Clustering - Hierarchical

Cluster based on distances between observations.

Represented as a tree hierarchy (*dendrogram*) rather than a partition of data.

Does not require committing to a choice of  $K$ .

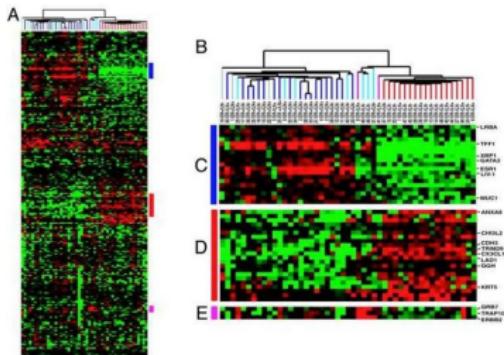


Figure 11: Sørlie, Therese, et al. (2003) "Repeated observation of breast tumor subtypes in independent gene expression data sets," PNAS.

# Clustering - Hierarchical: Dendrograms

- ▶ Each leaf in a dendrogram is a sample/ observation.
- ▶ As we move up the dendrogram, observations that are similar to each other begin to fuse into branches.
- ▶ Branches then fuse into bigger branches.
- ▶ Observations that fuse later (near the top of the tree, or root) are more different than observations that fuse earlier (near the leaves).

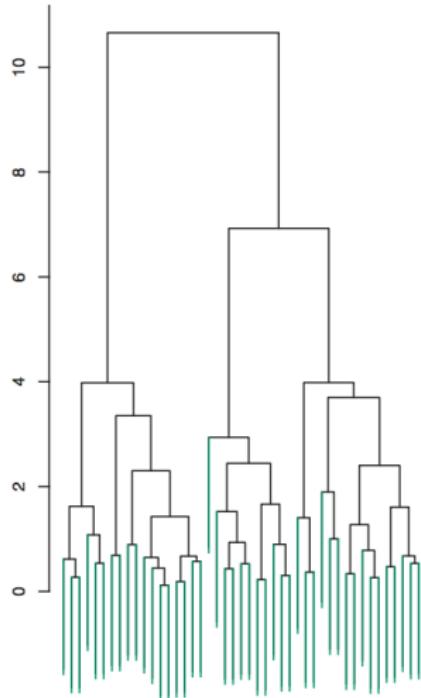


Figure 12: ISL (8th printing 2017)

# Clustering - Hierarchical: Dendrograms (cont.)

Note that the horizontal distance between observations on a dendrogram is not the appropriate assessment of observation similarity. Instead, look at vertical axis where branches are first fused.

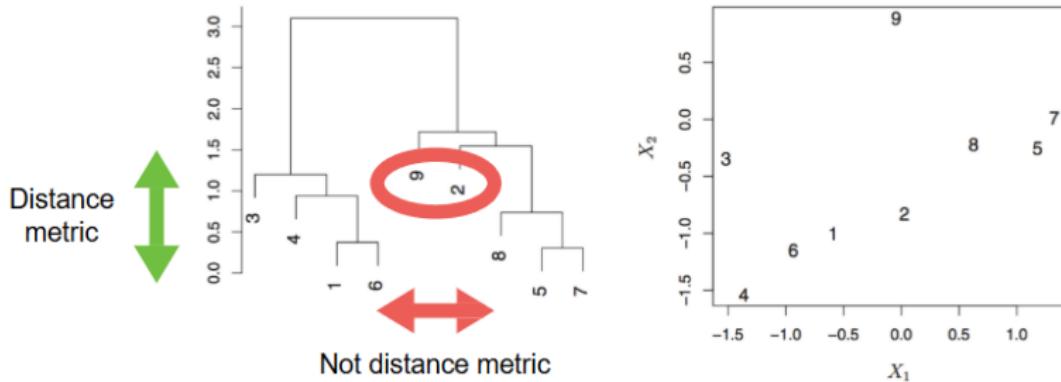


Figure 13: ISL (8th printing 2017)

# Clustering - Hierarchical: Dendograms (cont.)

Clusters are created by making a horizontal cut across the dendrogram. Clusters are the separate trees below the cut.

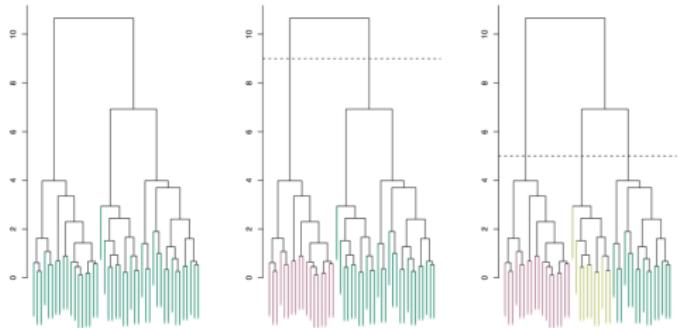


Figure 14: ISL (8th printing 2017)

## Building a Dendrogram

A dendrogram is most commonly built using a bottom-up or agglomerative algorithm.

We start at the leaves and group observations until we reach the root containing the entire dataset.

Like in  $k$ -means, we need a measure of similarity. Again, the most common is Euclidean distance.

- ▶ Compute the distance between each pair of observations.
- ▶ Merge the two closest observations into a cluster.
- ▶ Compute the distance between the new cluster and all other observations.
- ▶ Repeat until all observations are in one cluster.
- ▶ The distance between clusters is computed using a linkage method.

**Input:** Dataset  $D = \{x_1, x_2, \dots, x_n\}$

**Output:** Dendrogram representing the hierarchical structure of clusters

**Initialization:** Treat each data point as a separate cluster;

**Compute distance matrix:** Calculate pairwise distances between all clusters;

**Repeat until only one cluster remains:**

- ▶ Find the two closest clusters based on the distance matrix;
- ▶ Merge the two clusters into a new cluster;
- ▶ Update the distance matrix to reflect the new cluster;
- ▶ Recompute distances between the new cluster and all other clusters using a linkage method;

**Return:** Dendrogram representing the hierarchical structure of clusters;

**Algorithm:** Hierarchical Clustering Algorithm

w3schools: Codes and Playground

# Clustering - Hierarchical: Dendograms (cont.)

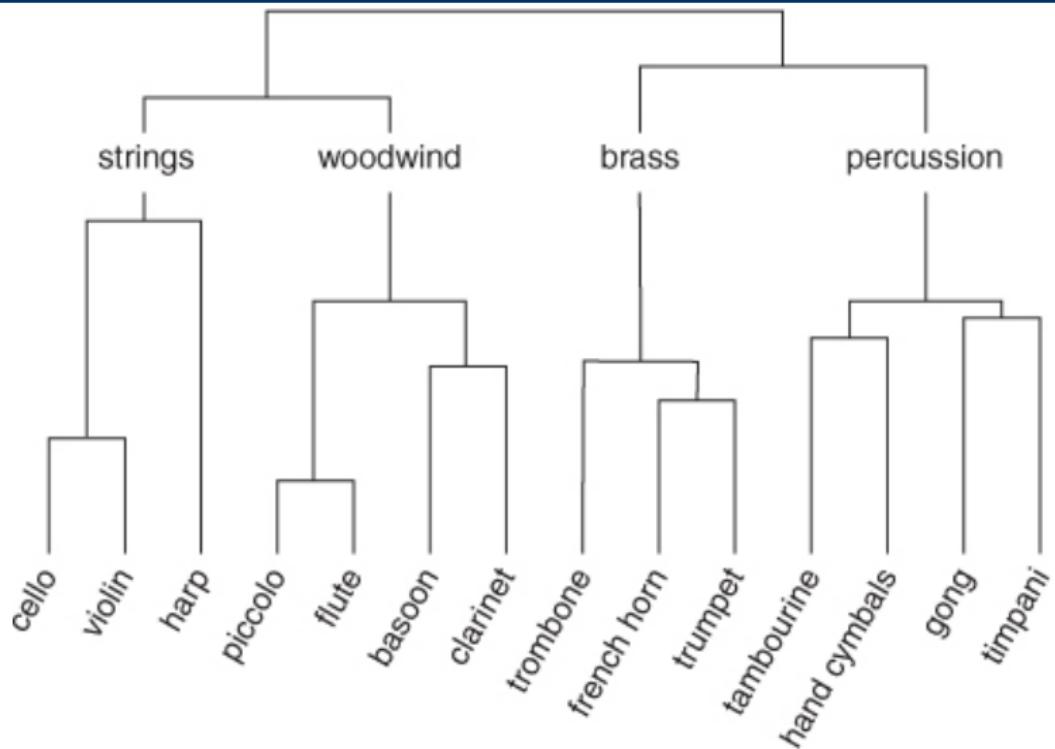


Figure 15: Dendrogram interpretation.

## Distance between groups

It's easy to compute Euclidean distance between two observations. What is the distance or similarity between two groups or clusters of observations?

**Linkage:** defines the dissimilarity between two groups of observations. Most common types are *complete*, *average*, *single*, and *centroid*.

- ▶ **Single Linkage:** Distance between two clusters is the minimum distance between any two points in the clusters.
- ▶ **Complete Linkage:** Distance between two clusters is the maximum distance between any two points in the clusters.
- ▶ **Average Linkage:** Distance between two clusters is the average distance between all pairs of points in the clusters.
- ▶ **Centroid Linkage:** Distance between two clusters is the distance between their centroids.
- ▶ **Ward's Linkage:** Distance between two clusters is the increase in variance when the two clusters are merged.

# Clustering - Hierarchical: Distance Metrics (cont.)

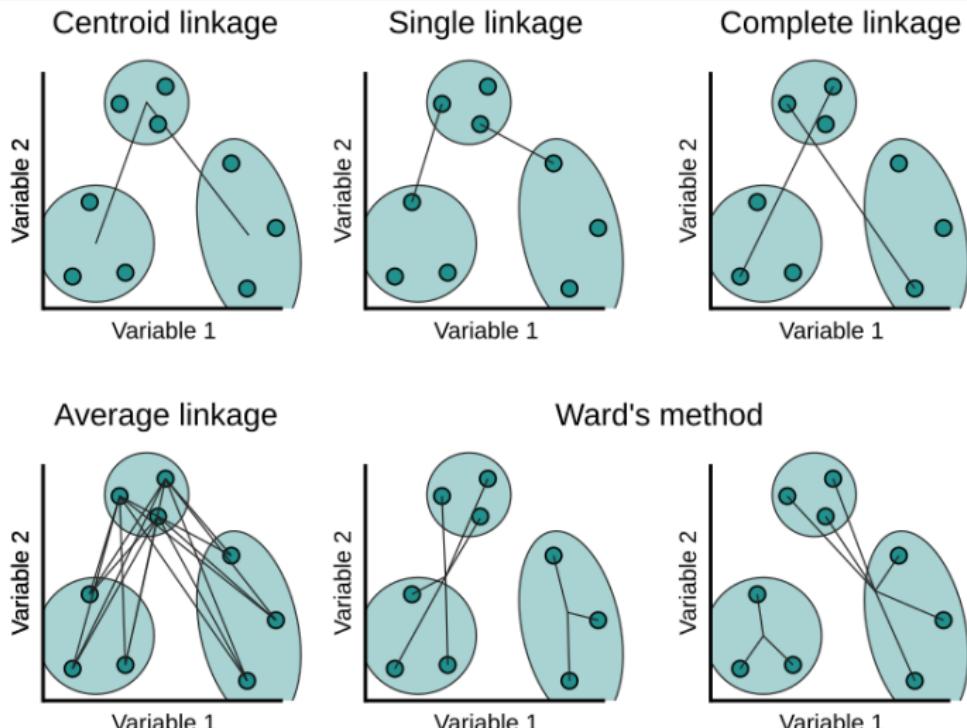


Figure 16: Linkage methods.

# Clustering - Hierarchical: Distance Metrics (cont.)

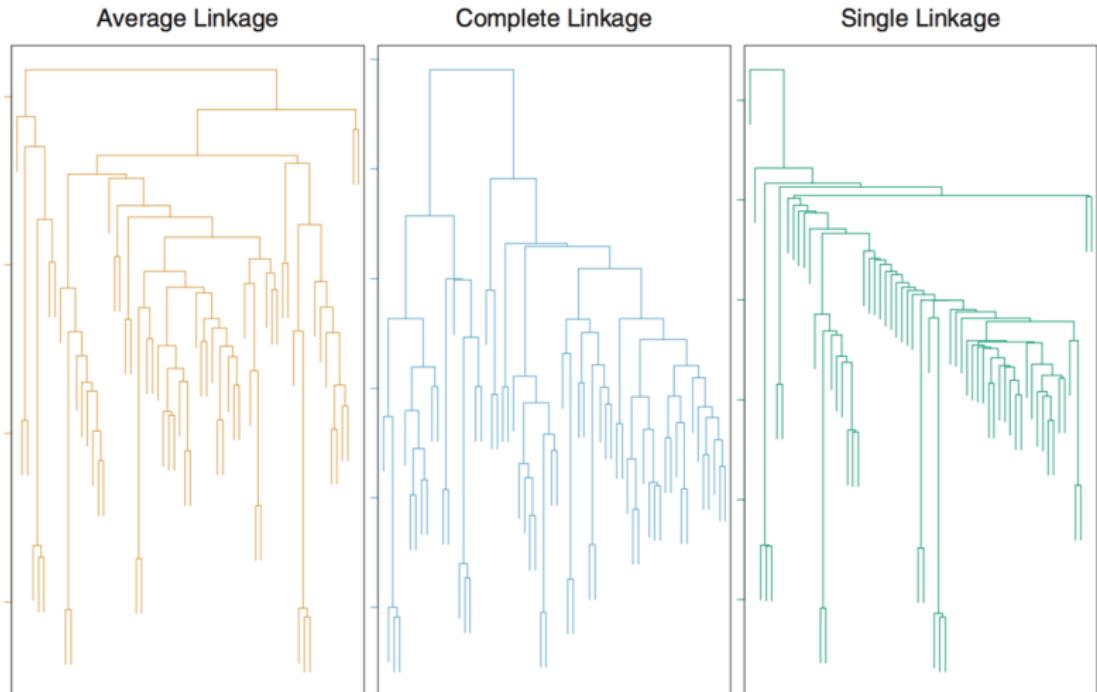


Figure 17: Dendrogram with different linkage types.

# Clustering - Hierarchical: Pros and Cons

## Pros

- ▶ No need to specify the number of clusters  $K$  in advance.
- ▶ Dendograms provide a visual representation of the clustering process.
- ▶ Can capture complex cluster shapes and relationships.

## Cons

- ▶ Computationally expensive for large datasets.
- ▶ Do have to pick where to cut the dendrogram to obtain clusters
- ▶ Sensitive to similarity measure and type of linkage used.
- ▶ Sensitive to noise and outliers.
- ▶ Difficult to interpret and choose the optimal number of clusters.

- ▶ Clustering based on density (local cluster criterion), such as density-connected points or based on an explicitly constructed density function.
- ▶ **Major features:**
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters

## ► Major algorithms:

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Ester, et al. (KDD'96)
- OPTICS (Ordering Points to Identify the Clustering Structure): Ankerst, et al. (SIGMOD'99)
- HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise): Campello, et al. (ACM TIST'15)
- DENCLUE (DENsity-based CLUstEring): Hinneburg and Gabriel (KDD'97)
- CLIQUE (CLustering In QUEst): Karypis, Han, and Kumar (SIGMOD'98)



- ▶ **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise):
  - Density = number of points within a specified radius  $\epsilon$ .
  - A point is a core point if it has more than a specified number of points (MinPts) within  $\epsilon$ . These are points that are at the interior of a cluster.
  - A border point has fewer than MinPts within  $\epsilon$ , but is in the neighborhood of a core point
  - A noise point is any point that is not a core point or a border point.
  - Groups together points that are closely packed together, marking as outliers points that lie alone in low-density regions.
  - Parameters:
    - ▶  $\epsilon$ : Maximum distance between two points for them to be considered as in the same neighborhood.
    - ▶ MinPts: Minimum number of points required to form a dense region.

# Clustering - DBSCAN (cont.)

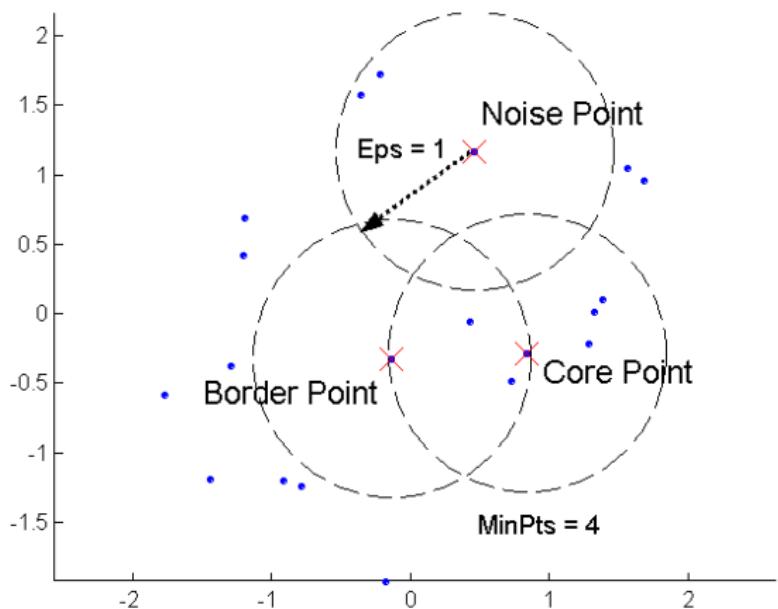


Figure 18: DBSCAN features: Core, Border, and Noise Points

# Clustering - DBSCAN: Algorithm

**Input:** Set of points  $P$ , distance threshold  $\varepsilon$ , minimum number of points  $minPts$

**Output:** A set of clusters

Construct a directed graph  $G = (V, E)$  where each node in  $V$  corresponds to a point in  $P$ ;

**foreach** point  $c \in P$  **do**

**if**  $c$  is a core point (i.e.,  $|\mathcal{N}_\varepsilon(c)| \geq minPts$ ) **then**

**foreach** point  $p \in \mathcal{N}_\varepsilon(c)$  **do**

            Add a directed edge  $(c \rightarrow p)$  to  $E$ ;

**end**

**end**

**end**

$N \leftarrow V$ ;

**while** there exists a core point  $c \in N$  **do**

    Let  $X$  be the set of nodes reachable from  $c$  via directed edges in  $G$ ;

    Form a cluster  $C = X \cup \{c\}$ ;

    Remove all nodes in  $C$  from  $N$ ;

**end**

# Clustering - DBSCAN: Core, Border, and Noise Points



Figure 19: Original Points

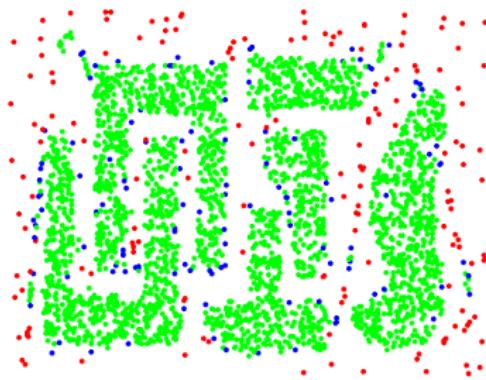


Figure 20: Point types: **Core**, **Border**, and **Noise** Points

$Eps = 10$ ,  $MinPts = 4$

# When DBSCAN Works Well



Figure 21: Original Points

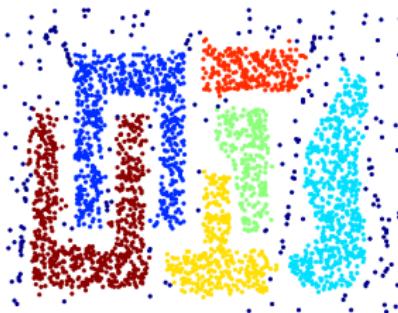


Figure 22: Clusters identified by DBSCAN

DBSCAN works well when:

- ▶ Clusters are of varying shapes and sizes.
- ▶ There is a clear distinction between dense and sparse regions.
- ▶ The data contains noise or outliers.

# When DBSCAN Doesn't Work Well

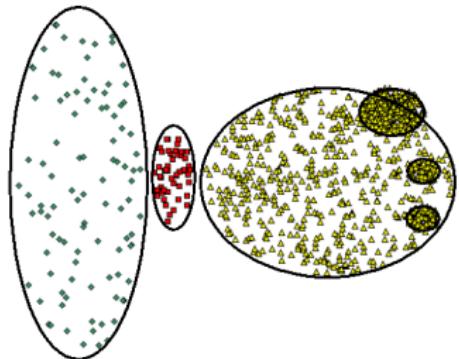


Figure 23: Original Points

DBSCAN does not work well when:

- ▶ Clusters are of varying densities.
- ▶ The data has varying scales or dimensions.
- ▶ There are overlapping clusters.
- ▶ The choice of  $\epsilon$  and MinPts is not suitable for the data distribution.

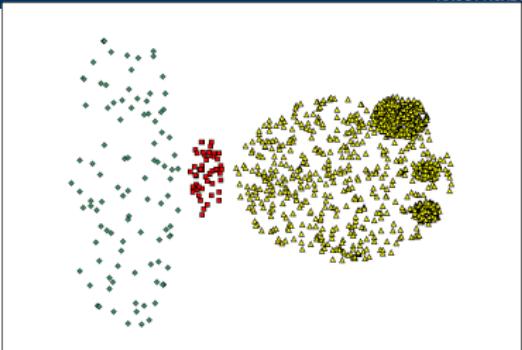


Figure 24:  $\text{MinPts}=4$ ,  $\epsilon=9.75$

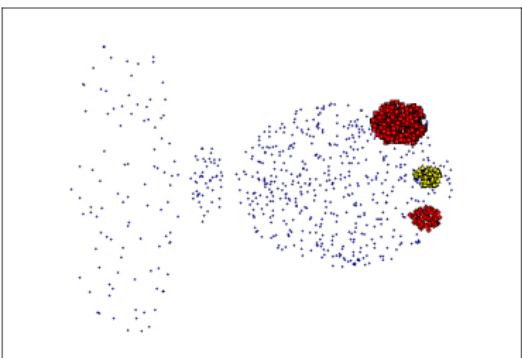


Figure 25:  $\text{MinPts}=4$ ,  $\epsilon=9.92$

# DBSCAN: Determining $\epsilon$ and MinPts

- ▶ A common approach to determine  $\epsilon$  is to use a k-distance graph:
  - For each point, compute the distance to its k-th nearest neighbor.
  - Plot these distances in ascending order.
  - Look for a "knee" in the plot, which indicates a suitable value for  $\epsilon$ .
- ▶ MinPts is often set based on domain knowledge or heuristics:
  - A common rule of thumb is to set MinPts to at least the dimensionality of the data plus one (e.g., for 2D data, MinPts = 3).
  - Higher values of MinPts can lead to fewer clusters and more noise points.

# DBSCAN: Determining $\epsilon$ and MinPts (cont.)

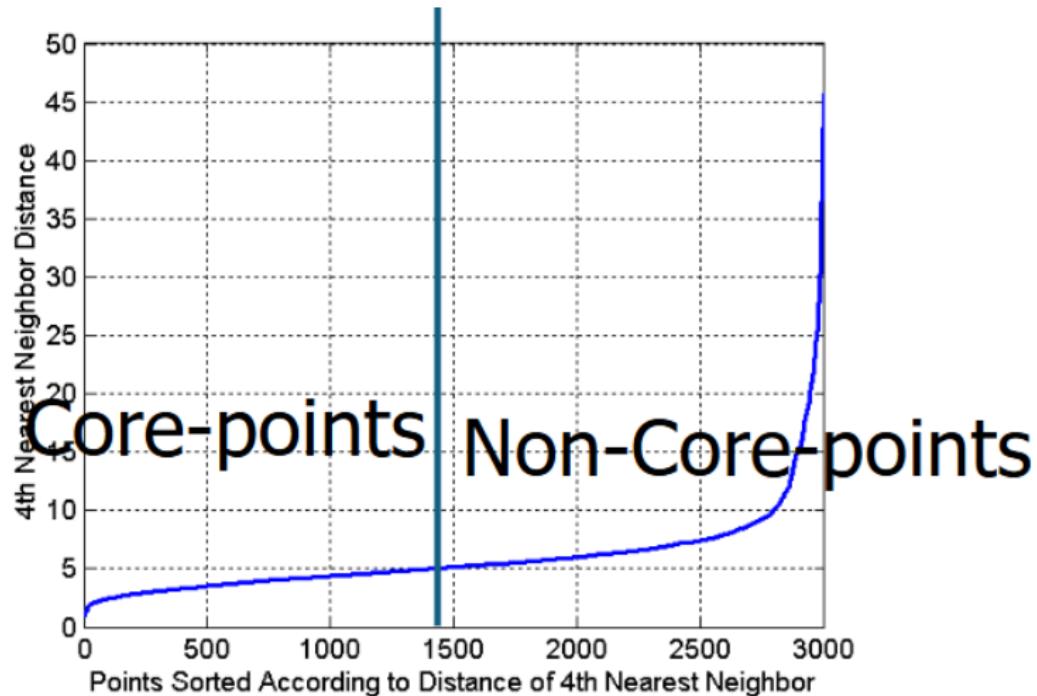


Figure 26: Run K-means for Minp=4 and not fixed  $\epsilon$

## ► Complexity:

- The time complexity of DBSCAN is  $O(n \log n)$  for spatial data structures like KD-trees or R-trees, where  $n$  is the number of points.
- Without spatial indexing, the complexity can degrade to  $O(n^2)$ .
- Space complexity is  $O(n)$ , as it needs to store the points and their cluster assignments.

## ► Limitations:

- Sensitive to the choice of  $\epsilon$  and MinPts parameters.
- Struggles with clusters of varying densities.
- Not suitable for high-dimensional data due to the curse of dimensionality.
- Cannot handle clusters that are not well-separated in terms of density.

► **Good:**

- Can discover clusters of arbitrary shape.
- Effectively handles noise and outliers.
- Requires only one scan of the data.
- Suitable for datasets with varying densities.
- Does not require prior knowledge of the number of clusters.

► **Bad:**

- Does not work well in high-dimensional datasets.
- Parameter selection is tricky.
- Has problems identifying clusters of varying densities ( $\rightarrow$  SSN algorithm).
- Density estimation is simplistic ( $\rightarrow$  does not create a real density function, but rather a graph of density-connected points).

# DBSCAN: Algorithm Revisited

```
current_cluster_label ← 1
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    end if
    for all points in the  $Eps$ -neighborhood, except  $i^{th}$  the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
        end if
    end for
end for
```

Figure 27: DBSCAN Algorithm Steps

# Where are we heading?

Gene	Description	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
Inpp5d	inositol polyphosphate-5-phosphatase D	7.00	5.45	5.89	6.03	5.75
Aim2	absent in melanoma 2	3.01	4.37	4.59	4.38	4.18
Gldn	gliomedin	3.48	3.63	4.61	4.70	4.74
Frem2	Fras1 related extracellular matrix protein 2	4.75	4.66	3.46	3.74	3.45
Rps3a1	ribosomal protein S3A1	6.10	7.23	7.44	7.36	7.34
Slc38a3	solute carrier family 38, member 3	1.90	3.16	3.52	3.61	3.19
Mt1	metallothionein 1	5.07	6.49	6.46	6.04	6.05
C1s1	complement component 1, s subcomponent 1	2.74	3.02	3.86	4.10	4.10
Cds1	CDP-diacylglycerol synthase 1	4.55	4.22	3.80	3.16	3.12
Ifi44	interferon-induced protein 44	4.82	4.52	3.87	3.42	3.59
Lefty2	left-right determination factor 2	6.95	6.28	5.88	5.60	5.61
Fmr1nb	fragile X mental retardation 1 neighbor	4.28	2.78	3.10	3.25	2.57
Tagln	transgelin	7.93	7.91	7.20	7.02	6.68

Figure 28: Original Points

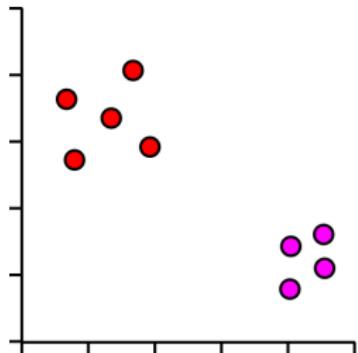
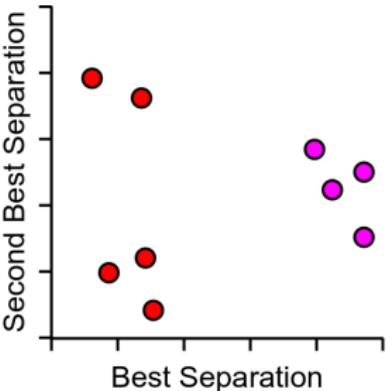
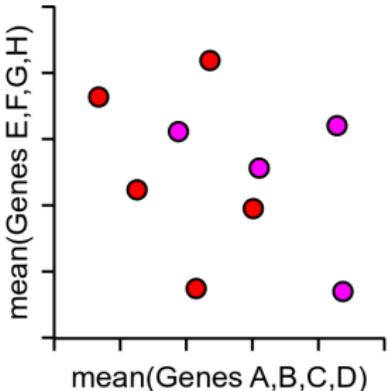
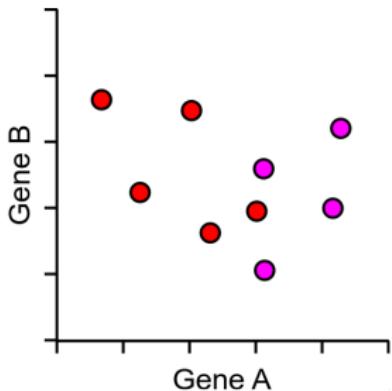


Figure 29: Clusters identified by DBSCAN

- ▶ Each dot is a cell.
- ▶ Groups of dots are similar cells.
- ▶ Separation of groups could be interesting biology.

# Too much data!

- ▶ 5000 cells and 2500 measured genes
- ▶ Realistically only 2 dimensions we can plot (x, y)



# Principal Component Analysis (PCA)

- ▶ Principal Component Analysis (PCA) is a method for optimally summarizing large, multi-dimensional datasets.
- ▶ PCA identifies a smaller number of dimensions (ideally 2) that retain most of the useful information present in the original data.
- ▶ The technique constructs a set of new variables, called Principal Components (PCs), which are linear combinations of the original variables.
- ▶ Each Principal Component represents a specific weighted sum of the original features. For example:

$$\text{PC} = (10 \times \text{GeneA}) + (3 \times \text{GeneB}) + (-4 \times \text{GeneC}) + (-20 \times \text{GeneD}) + \dots$$

- ▶ By projecting the data onto these principal components, PCA reduces dimensionality while preserving as much variance (information) as possible.

# Principal Component Analysis (PCA) (cont.)

- ▶ Principal Component Analysis (PCA) is a method for optimally summarizing large, multi-dimensional datasets.
- ▶ PCA identifies a smaller number of dimensions (ideally 2) that retain most of the useful information present in the original data.
- ▶ The technique constructs a set of new variables, called Principal Components (PCs), which are linear combinations of the original variables.
- ▶ Each Principal Component represents a specific weighted sum of the original features. For example:

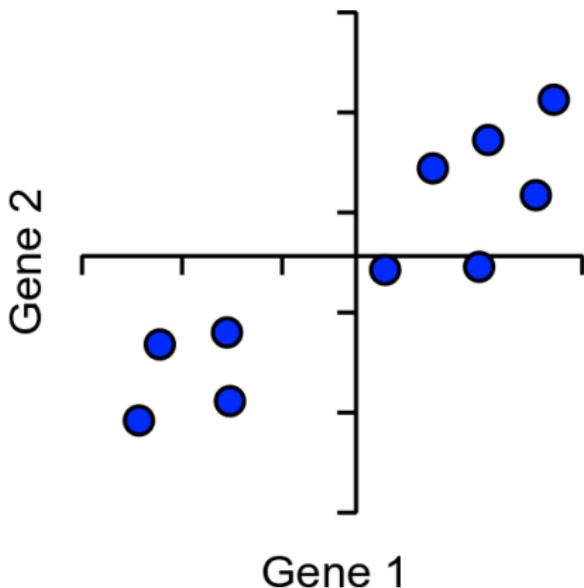
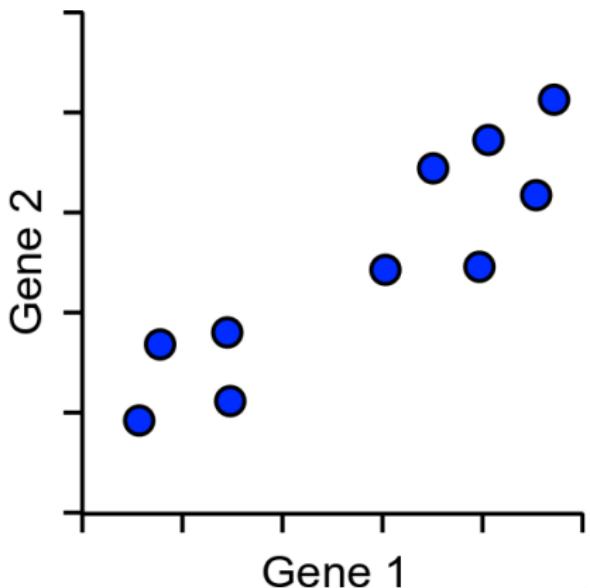
$$\text{PC} = (10 \times \text{GeneA}) + (3 \times \text{GeneB}) + (-4 \times \text{GeneC}) + (-20 \times \text{GeneD}) + \dots$$

- ▶ By projecting the data onto these principal components, PCA reduces dimensionality while preserving as much variance (information) as possible.

Simple example using 2 genes and 10 cells

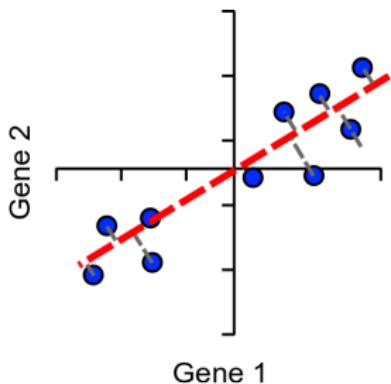
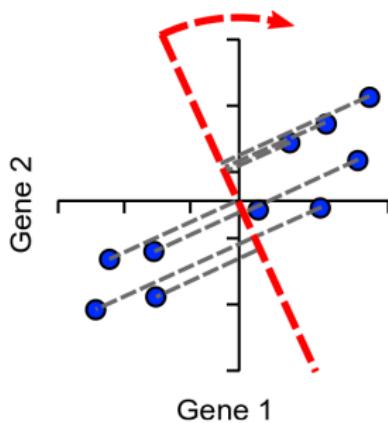
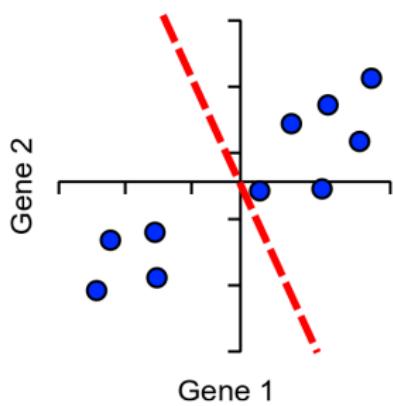
# How does PCA work?

Simple example using 2 genes and 10 cells



# How does PCA work? (cont.)

Find line of best fit, passing through the origin



# Assigning Loadings to Genes

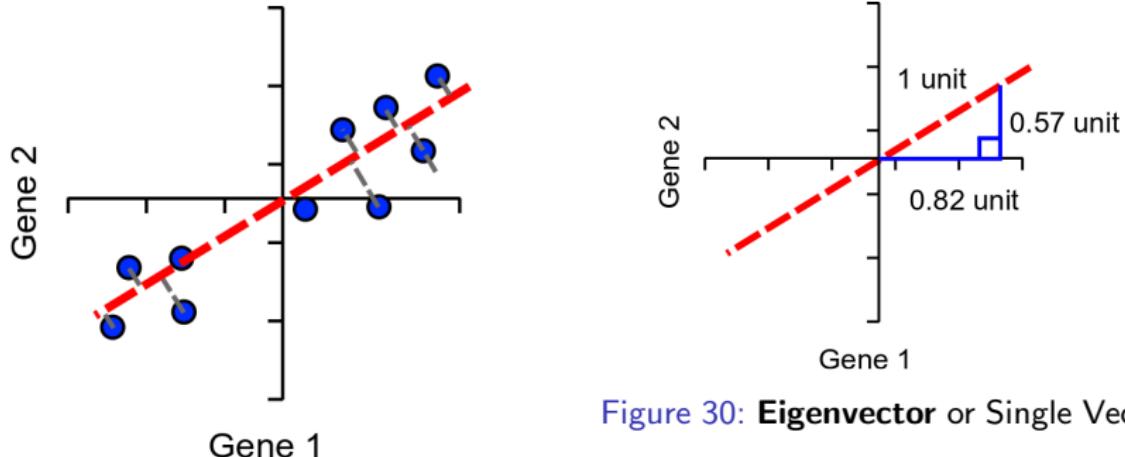
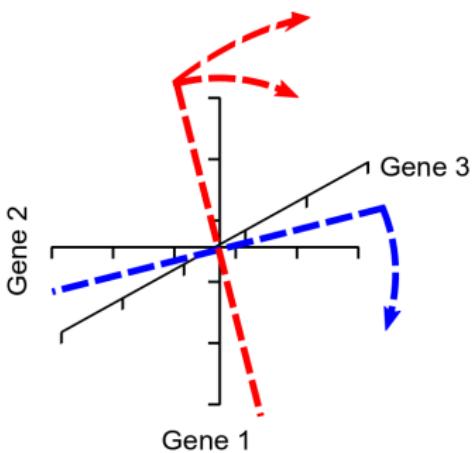


Figure 30: **Eigenvector** or Single Vector

- ▶ **Loadings** represent the contribution of each gene to the principal component.
- ▶ For this example using 2 genes and 10 cells:
  - Gene1 loading: 0.82
  - Gene2 loading: 0.57
- ▶ A higher loading means the gene has a greater influence on the principal component.

# More Dimensions

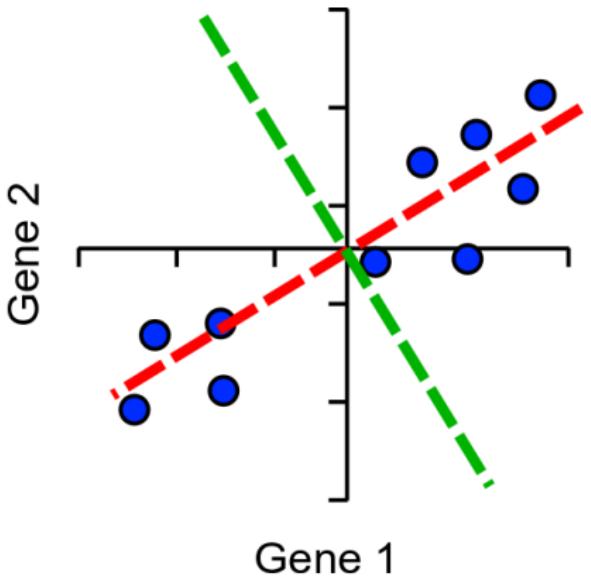


- ▶ The same idea extends to datasets with more dimensions ( $n$  genes).
- ▶ The first principal component (PC1) finds the direction of maximum variance, rotating in  $(n - 1)$  dimensions.
- ▶ The next principal component (PC2) is perpendicular to PC1 and rotates in the remaining  $(n - 2)$  dimensions.
- ▶ Each subsequent principal component is always perpendicular to the previous ones, capturing the next largest variance.
- ▶ The last principal component is the only remaining perpendicular direction.
- ▶ The number of principal components always equals the number of original genes (features).



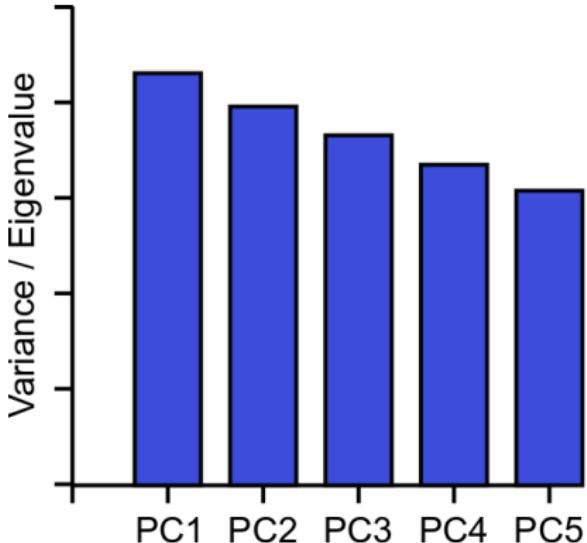
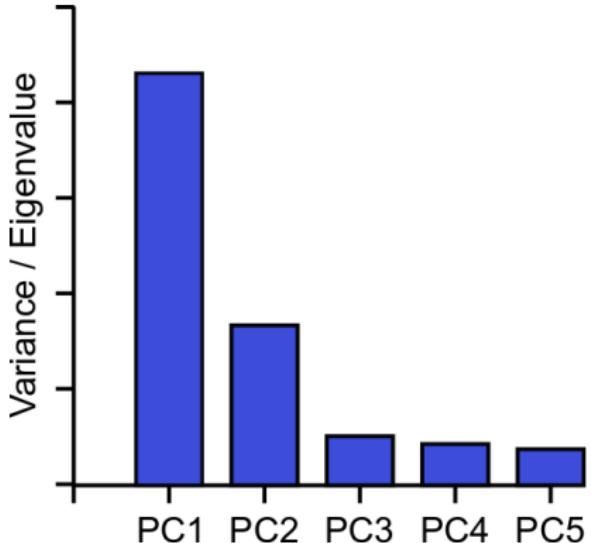
- ▶ Each principal component (PC) explains a certain proportion of the total variance in the data. Together, all PCs account for 100% of the variance.
- ▶ PC1 always explains the largest amount of variance.
- ▶ PC2 explains the next largest amount, and so on for subsequent PCs.
- ▶ Since we typically plot only 2 dimensions (PC1 and PC2), it's important to check how much of the total variance these two PCs explain.
- ▶ **How do we calculate this?**

# Explaining Variance (cont.)

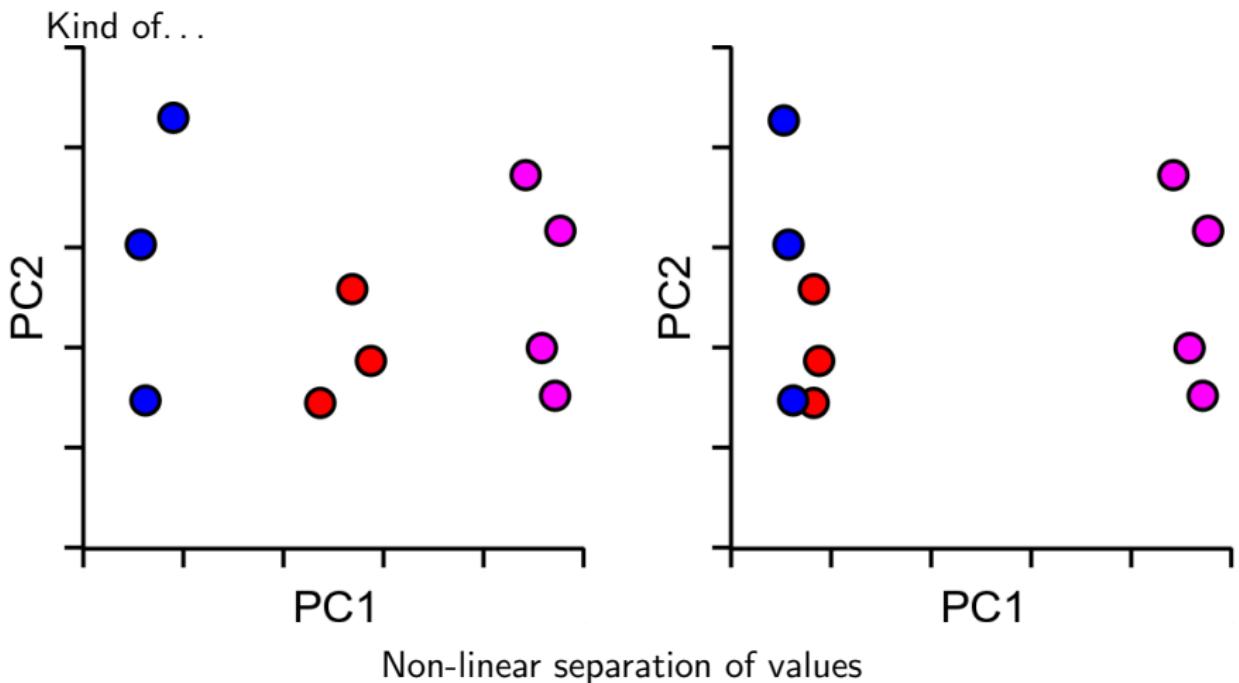


- ▶ Project each data point onto the principal component (PC).
- ▶ Calculate the distance of each projected point to the origin.
- ▶ Compute the sum of squared differences (SSD) from the origin.
- ▶ This SSD is a measure of variance, called the *eigenvalue*.
- ▶ Divide the SSD by  $(\text{number of points} - 1)$  to obtain the actual variance explained by the PC.

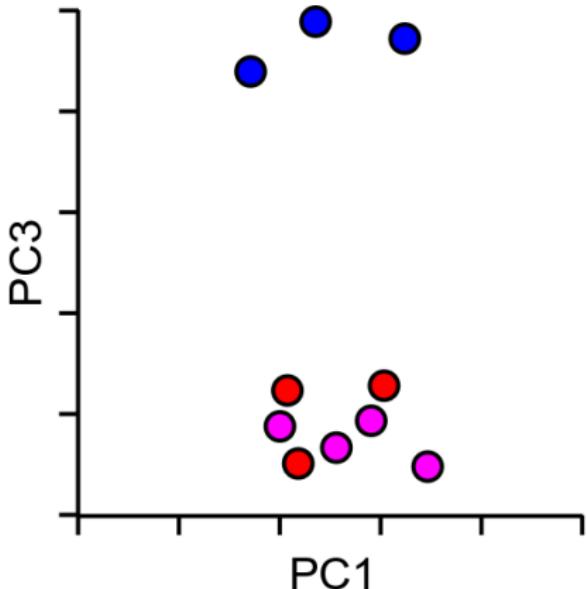
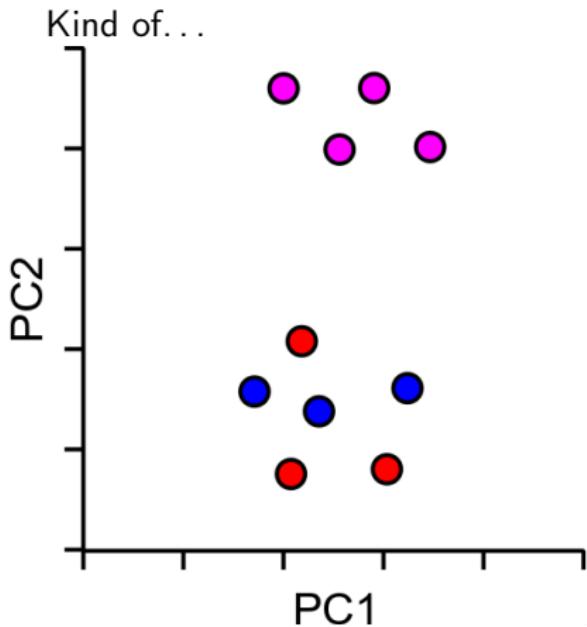
# Explaining Variance (cont.)



# So PCA is great then?



# So PCA is great then? (cont.)



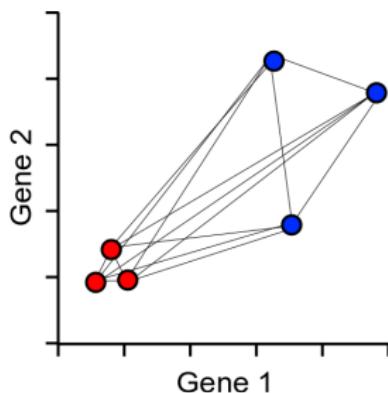
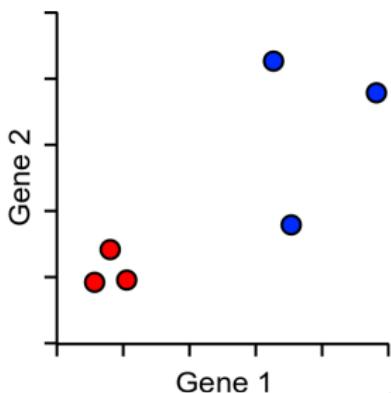
Not optimised for 2-dimensions

# tSNE to the rescue...

- ▶ T-Distributed Stochastic Neighbour Embedding
- ▶ Aims to solve the problems of PCA
- ▶ Non-linear scaling to represent changes at different levels
- ▶ Optimal separation in 2-dimensions

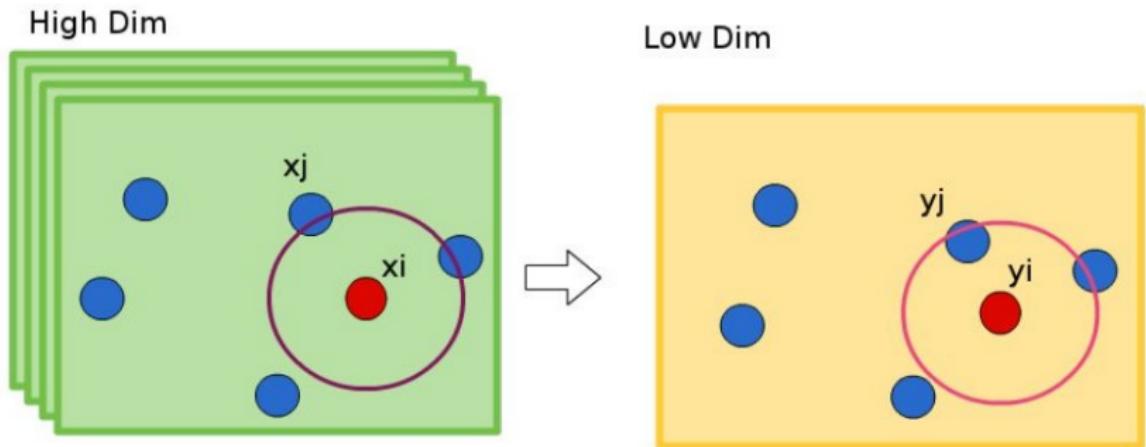
# tSNE: How does it work?

- ▶ Based around all-vs-all table of pairwise cell to cell distances
- ▶ Each cell is represented as a point in a high-dimensional space
- ▶ The pairwise distances are converted into probabilities



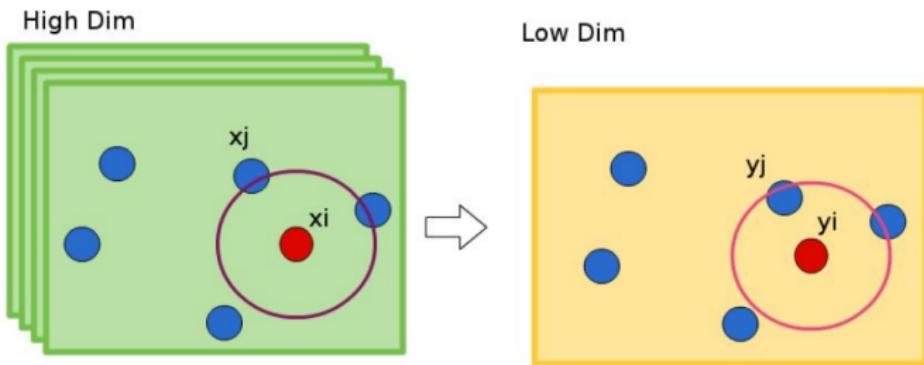
	0	10	10	295	158	153
9	0	0	1	217	227	213
1	8	0	154	225	238	
205	189	260	0	23	45	
248	227	246	44	0	54	
233	176	184	41	36	0	

# tSNE: Underlying idea



# tSNE: Stochastic Neighbor Embedding

Measure pairwise similarities between high-dimensional and low-dimensional objects



$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities

- Similarity of datapoints in High Dimension

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

- Similarity of datapoints in Low Dimension

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$$

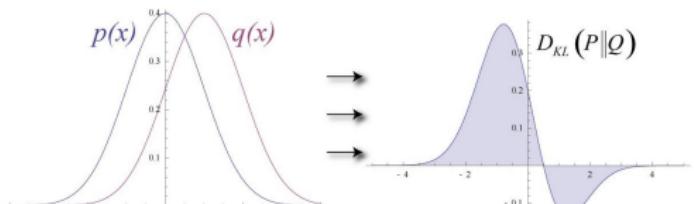
- Cost function

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

# KL Divergence

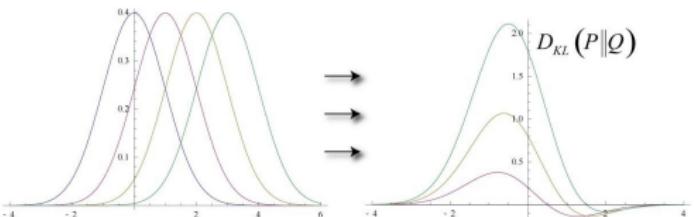
Measures the similarity between two probability distributions it is asymmetric.

$$D_{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$



Original Gaussian PDF's

KL Area to be Integrated



D<sub>KL</sub>(P||Q)

# tSNE: Stochastic Neighbor Embedding

Gradient has a surprisingly simple form

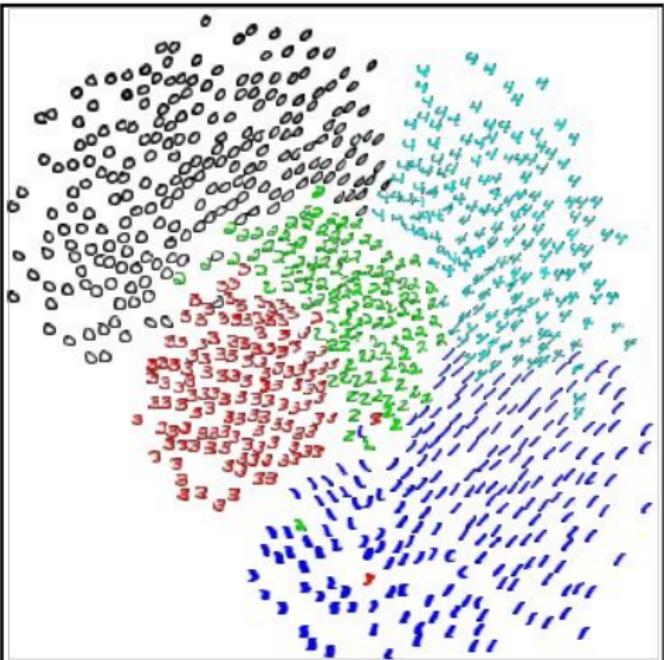
$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The gradient update with momentum term is given by

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \beta(t)(Y^{(t-1)} - Y^{(t-2)})$$

# tSNE: Stochastic Neighbor Embedding (cont.)

- ▶ The result of running the SNE algorithm on 3000 256-dimensional grayscale images of handwritten digits.
- ▶ The classes are quite well separated even though SNE had no information about class labels.
- ▶ Furthermore, within each class, properties like orientation, skew, and stroke thickness tend to vary smoothly across the space.



Such that  $p_{ij} = p_{ji}$ ,  $q_{ij} = q_{ji}$ , the main advantage is simplifying the gradient

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

However, in practice we symmetrize (or average) the conditionals

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Set the bandwidth  $\sigma_i$  such that the conditional has a fixed perplexity (effective number of neighbors)  $Perp(P_i) = 2^{H(P_i)}$ , typical value is about 5 to 50

Use heavier tail distribution than Gaussian in low-dim space, we choose

$$q_{ij} \propto (1 + \|y_i - y_j\|^2)^{-1}$$

Then the gradient could be

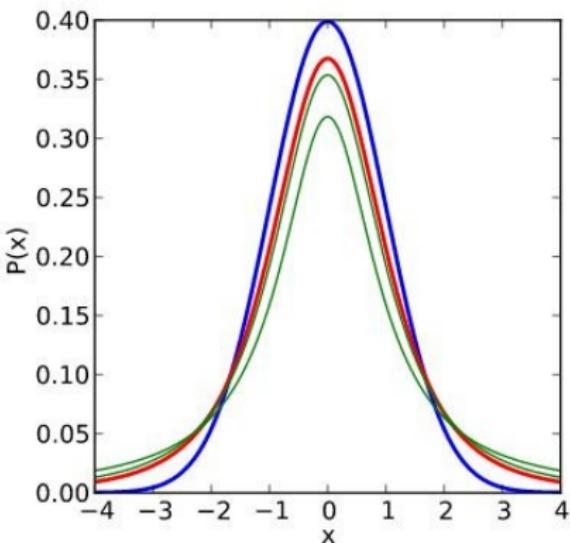
$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j)$$

# Student's t-Distribution

- ▶ Why do we define map similarities as:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|y_k - y_i\|^2)^{-1}}$$

- ▶ Suppose data is intrinsically high dimensional
- ▶ We try to model the local structure of this data in the map
- ▶ Result: Dissimilar points have to be modeled as too far apart in the map!



# t-Distributed Stochastic Neighbor Embedding

- Similarity of datapoints in High Dimension

$$p_{ij} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma^2)}$$

- Similarity of datapoints in Low Dimension

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||y_k - y_i||^2)^{-1}}$$

- Cost function

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

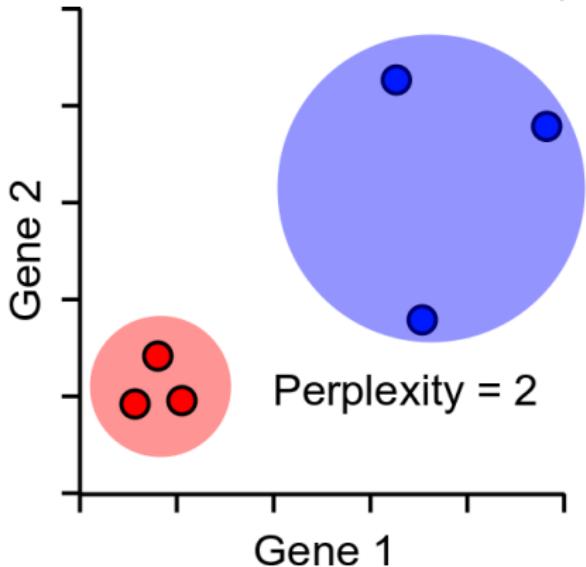
- Large  $p_{ij}$  modeled by small  $q_{ij}$ : Large penalty
- Small  $p_{ij}$  modeled by large  $q_{ij}$ : Small penalty
- t-SNE mainly preserves local similarity structure of the data

- Gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j)$$

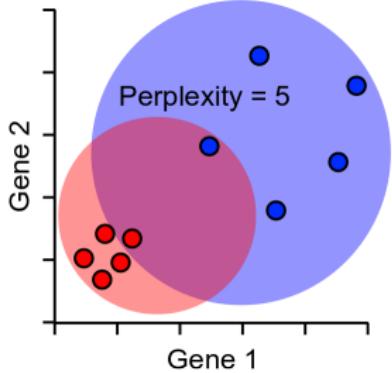
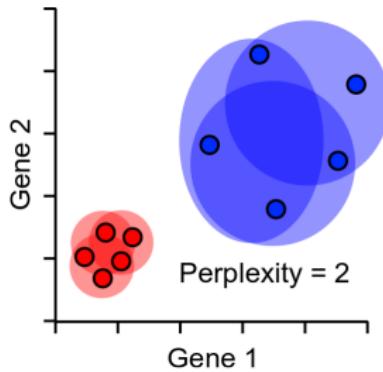
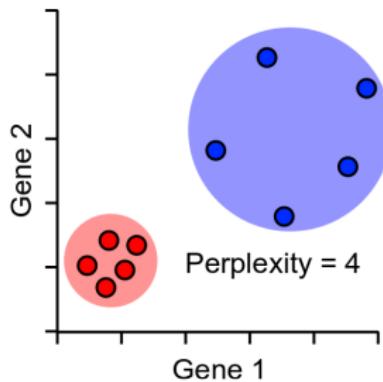
# Distance scaling and perplexity

- ▶ Perplexity = expected number of neighbours within a cluster
- ▶ Distances scaled relative to perplexity neighbours



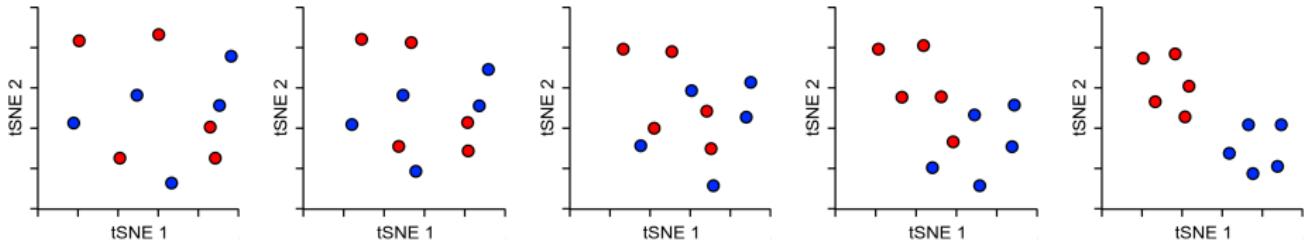
	0	4	6	586	657	836
0	0	4	4	815	527	776
4	3	0	0	752	656	732
6	31	28	29	0	4	7
586	31	24	25	4	0	7
657	40	37	32	8	8	0
836						

# Perplexity Robustness



- ▶ Randomly scatter all points within the space (normally 2D)
- ▶ Start a simulation
  - Aim is to make the point distances match the distance matrix
  - Shuffle points based on how well they match
  - Stop after fixed number of iterations, or
  - Stop after distances have converged

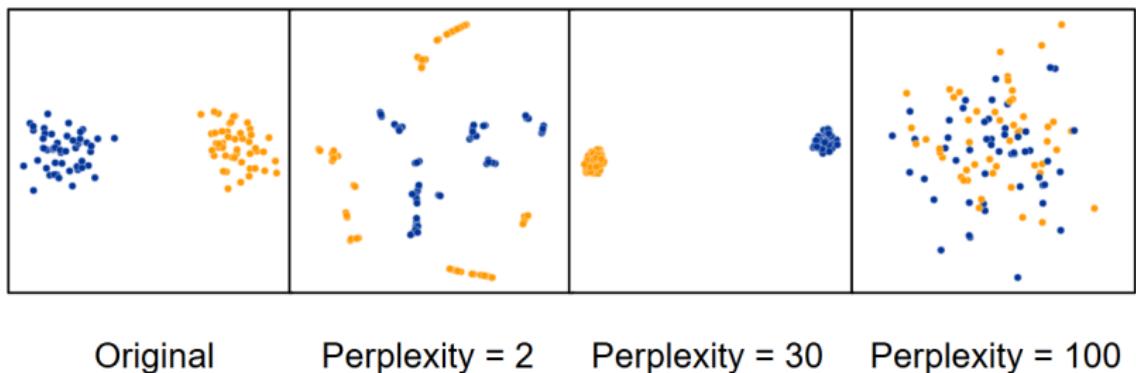
# tSNE Projection (cont.)



- ▶ X and Y don't mean anything (unlike PCA)
- ▶ Distance doesn't mean anything (unlike PCA)
- ▶ Close proximity is highly informative
- ▶ Distant proximity isn't very interesting
- ▶ Can't rationalise distances, or add in more data

# tSNE: Practical Examples

Perplexity Settings Matter



Original

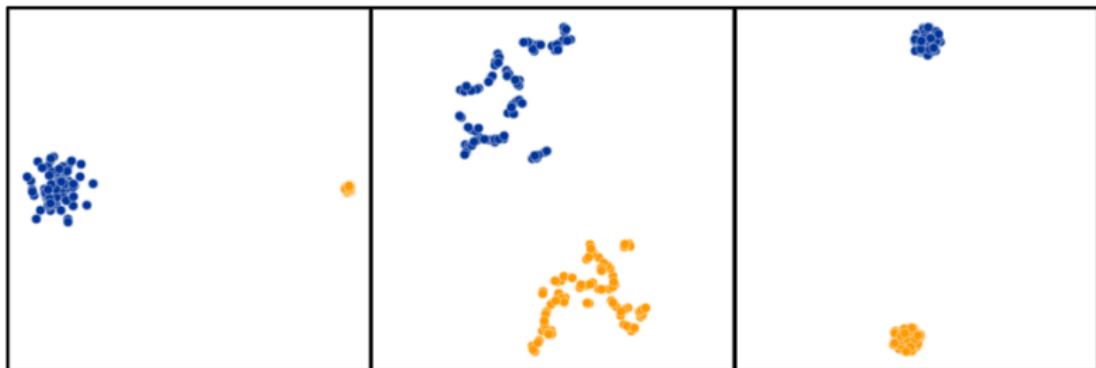
Perplexity = 2

Perplexity = 30

Perplexity = 100

# tSNE: Practical Examples (cont.)

Cluster Sizes are Meaningless



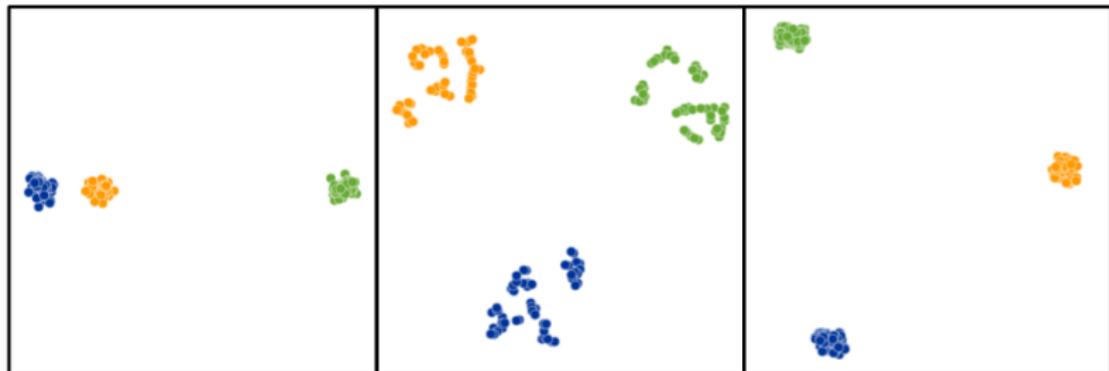
Original

Perplexity = 5

Perplexity = 50

# tSNE: Practical Examples (cont.)

Distances between clusters can't be trusted



Original

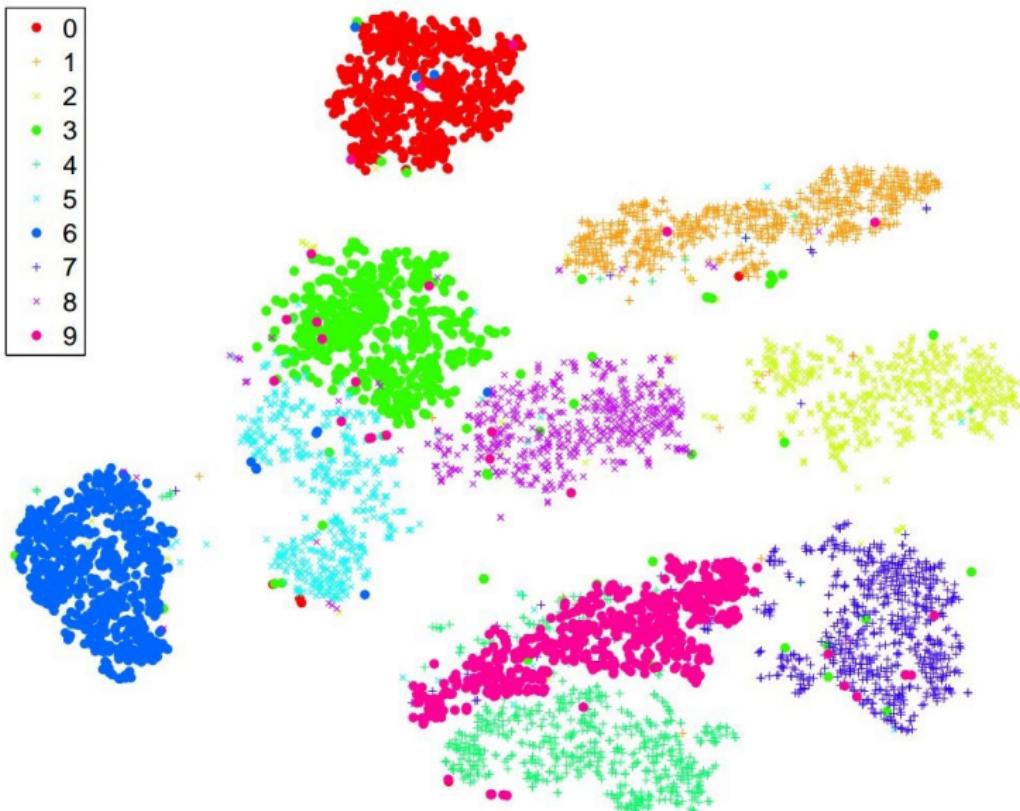
Perplexity = 5

Perplexity = 30

# tSNE: Results on MNIST



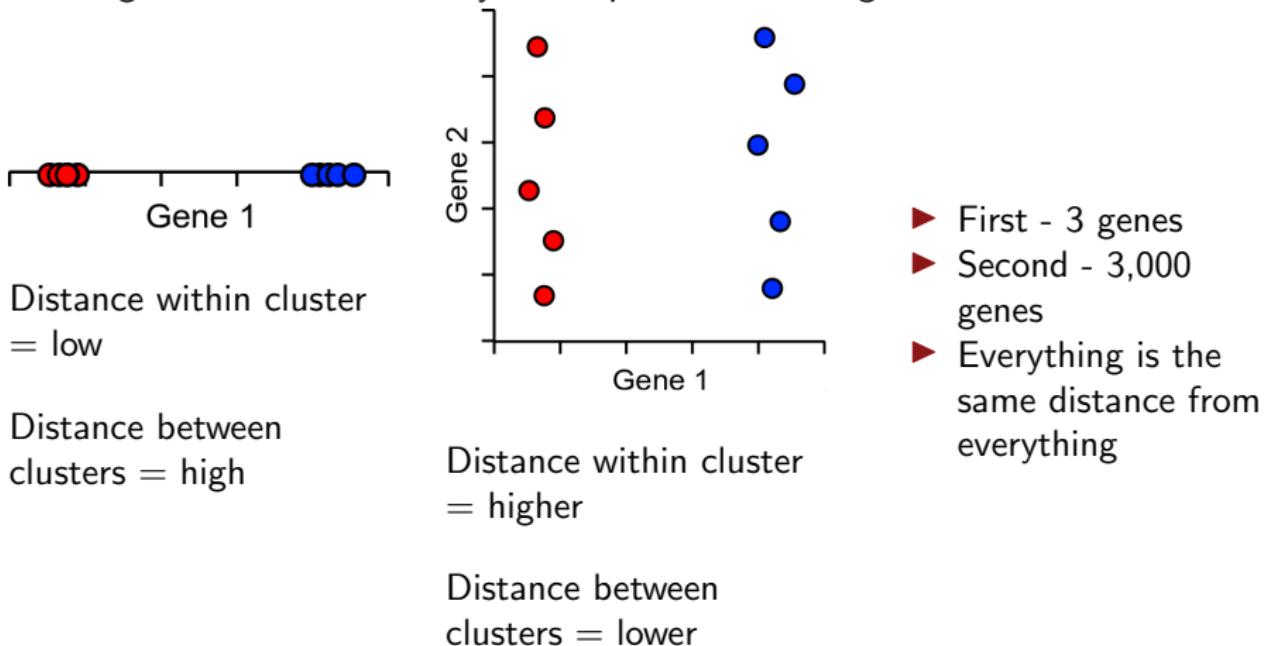
## tSNE: Results on MNIST (cont.)



# So tSNE is great then?

Kind of...

Imagine a dataset with only one super informative gene



# So there is no clear solution?

PCA:

- ▶ Requires more than 2 dimensions
- ▶ Expects linear relationships

tSNE:

- ▶ Can't cope with noisy data
- ▶ Loses the ability to cluster

# So there is no clear solution? (cont.)

PCA:

- ▶ Requires more than 2 dimensions
- ▶ Expects linear relationships

tSNE:

- ▶ Can't cope with noisy data
- ▶ Loses the ability to cluster

**Answer: Combine the two methods, get the best of both worlds!**

PCA:

- ▶ Good at extracting signal from noise
- ▶ Extracts informative dimensions

tSNE:

- ▶ Can reduce to 2D well
- ▶ Can cope with non-linear scaling

**This is what many pipelines do in their default analysis!**

# So PCA + tSNE is great then?

Kind of...

- ▶ tSNE is slow—this is its biggest drawback.
  - tSNE does not scale well to large datasets (10,000+ samples).
- ▶ tSNE provides reliable information only about the closest neighbors; large distance information is mostly irrelevant.

# UMAP to the rescue!

- ▶ UMAP is a replacement for t-SNE, designed to fulfill the same role in dimensionality reduction.
- ▶ It is conceptually very similar to t-SNE, but introduces several relevant (and somewhat technical) changes.
- ▶ In practice, UMAP offers several advantages:
  - UMAP is significantly faster than t-SNE.
  - It can preserve more global structure compared to t-SNE.<sup>1</sup>
  - UMAP can operate directly on raw data without requiring PCA preprocessing.<sup>2</sup>
  - It allows new data points to be added to an existing projection, enabling incremental updates.

---

<sup>1</sup>Preservation of global structure may depend on the dataset and parameter choices.

<sup>2</sup>Although PCA preprocessing can still be beneficial in some cases.



- ▶ Unlike t-SNE, which uses a single perplexity parameter, UMAP introduces two key parameters:
  - **Number of Nearest Neighbours:** Specifies how many nearest neighbours are considered for each point. This is conceptually similar to perplexity in t-SNE and controls the balance between local and global structure in the embedding.
  - **Minimum Distance:** Determines how closely UMAP packs points together in the low-dimensional space. Lower values allow points to be closer, resulting in more compact clusters, while higher values spread points apart.
- ▶ The **nearest neighbours** parameter influences the trade-off between preserving local versus global structure in the data.
- ▶ The **minimum distance** parameter affects the density and compactness of clusters in the resulting visualization.
- ▶ Together, these parameters provide more flexibility and control over the embedding compared to t-SNE's single perplexity value.

# UMAP differences (cont.)

**Structure preservation** – mostly in the 2D projection scoring

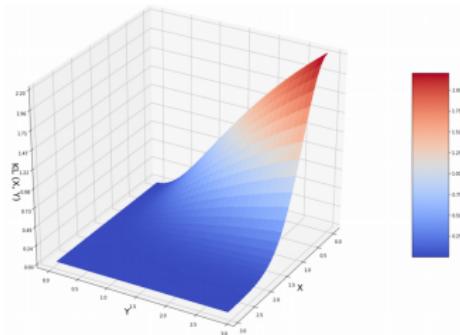


Figure 31: t-SNE

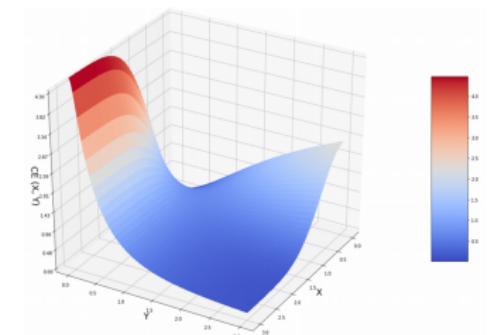
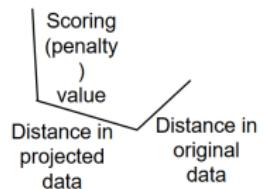
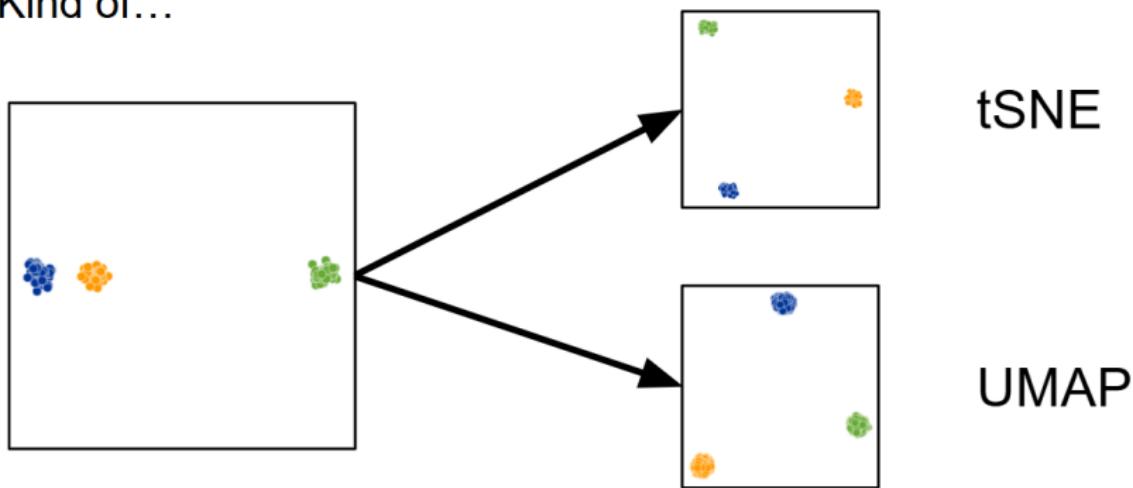


Figure 32: UMAP



# So UMAP is great then?

Kind of...

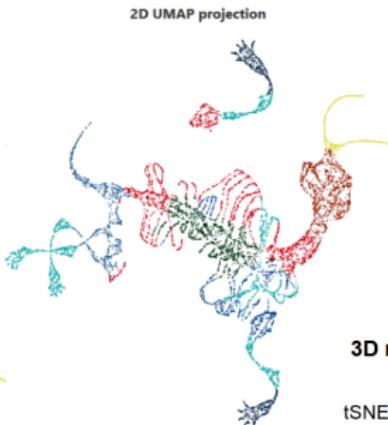
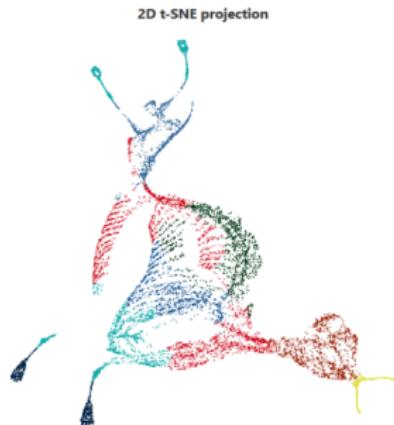


tSNE

UMAP

# So UMAP is all hype then?

No, it really does better for some datasets...



**3D mammoth skeleton projected into 2D**

tSNE: Perplexity 2000 2h 5min

UMAP: Neigh 200, mindist 0.25, 3min

<https://pair-code.github.io/understanding-umap/>

## ► Filter the data:

- Remove poorly behaving cells to ensure data quality.
- Select only expressed genes for further analysis.
- Identify and retain variable genes to focus on informative features.

## ► Apply PCA (Principal Component Analysis):

- Extract the most interesting and informative signals from the data.
- Select the top principal components (PCs) to reduce dimensionality, but do not reduce directly to 2 dimensions.

## ► Run t-SNE or UMAP:

- Calculate distances between samples using the PCA projections.
- Scale these distances appropriately.
- Project the data into 2 dimensions for visualization using t-SNE or UMAP.

# So PCA + UMAP is great then?

- ▶ Kind of... as long as you only have one dataset.
  - In 10X experiments, each library is considered a **batch**.
  - Over time or distance, additional biases can be introduced.
  - These biases make direct comparisons between datasets challenging.
  - To enable meaningful comparisons, it is necessary to align the datasets and correct for batch effects.

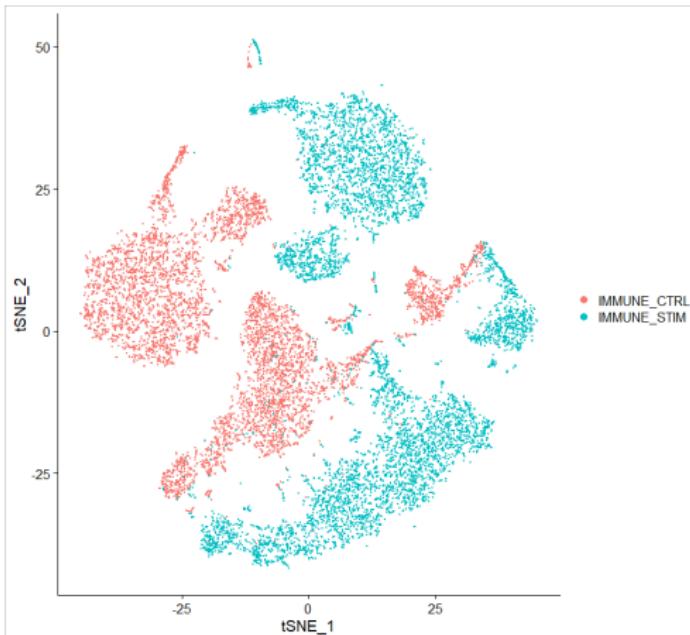


Figure 33: UMAP

- ▶ Sherrie Wang, CME 250: Introduction to Machine Learning, Stanford University. <https://web.stanford.edu/class/cme250/>
- ▶ Volodymyr Kuleshov, CS 5785: Applied Machine Learning, Cornell University. [https://www.cs.cmu.edu/~arielpro/15381f16/slides/NLP\\_guest\\_lecture.pdf](https://www.cs.cmu.edu/~arielpro/15381f16/slides/NLP_guest_lecture.pdf)
- ▶ Aykut Erdem, COMP547: Deep Unsupervised Learning, Koç University.  
<https://aykuterdem.github.io/classes/comp547.s24/>
- ▶ Christoph F. Eick, Machine Learning, University of Houston.  
<http://www2.cs.uh.edu/~ceick/ML/Topic9.ppt>
- ▶ Zaur Fataliyev, Seoul AI.  
<https://seoulai.com/presentations/t-SNE.pdf>
- ▶ Simon Andrews, "Dimensionality Reduction", Babraham Bioinformatics. <https://www.bioinformatics.babraham.ac.uk/training/10XRNASeq/Dimension%20Reduction.pptx>

## Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

[p.aparajeya@aisimply.uk](mailto:p.aparajeya@aisimply.uk)

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.