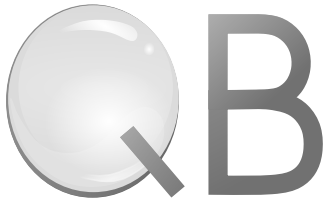


Fundamentals of Quantitative Biology

STATS 05: High Dimensional Data - Non-linear embedding

Winter semester 2024/2025

High-dimensional data

High-dimensional data

Representation of **high-dimensional** data as feature matrix:

$$\begin{aligned} \boldsymbol{C}(n, m) &= \text{feature matrix} \\ n &= \text{number of rows (samples or data points)} \\ m &= \text{number of columns (features or dimensions)} \\ c(i, j) &= \text{ value for row i } j \end{aligned}$$

$$i = 1, \ldots, n \quad j = 1, \ldots, m$$

The features span the m-dimensional vectorspace. Each axis should be sufficiently populated ($n \gg m$).
If $m > n$ we have fat data and suffer from the curse of dimensionality.

Feature Selection

Some features may be more relevant than others.

- **Explanatory Features:** Capture the essential structure or patterns in the data.
- **Noisy/Redundant Features:** Irrelevant or redundant information that obscures meaningful patterns.

Extraction Techniques:

- **Variance Thresholding:** Remove features with low variance.
- **Correlation analysis:** Remove highly correlated feature to reduce redundancy .
- **Random forest:** Identifies and ranks highly explanatory features through ensemble learning.
- **Autoencoder:** Learn compressed feature representations.
- **Entropy-Based Methods:** Maximize information gain.

Feature Selection - Example

Example from NLP (Natural Language Processing)

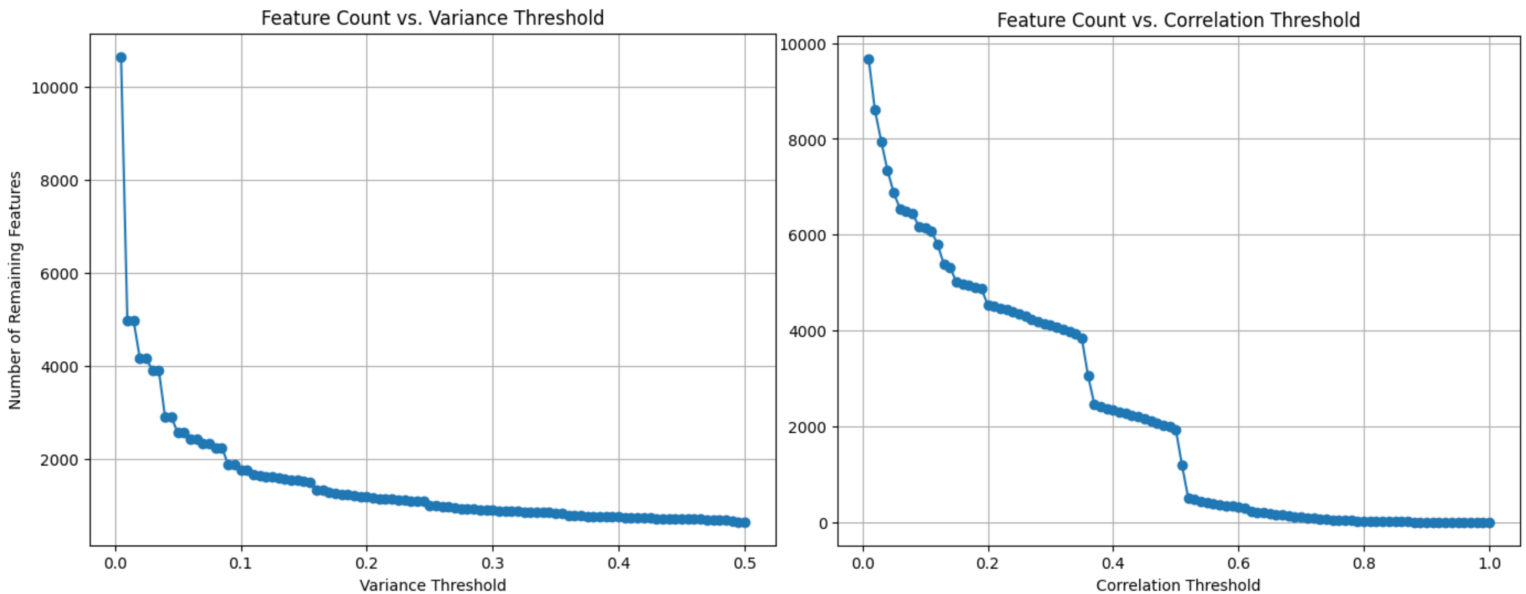
- take $n(= 100)$ random wikipedia articles
- set the occuring words as features (BoW - Bag of Words)
- construct the feature matrix: articles as rows (samples) and words as columns (features)
- elements in the matrix are word frequencies (better: TF-IDF values)

This results in more than 10.000 unique words and thus 10k features!

Feature Selection - Example

Feature selection techniques:

- variance (the higher the better)
- correlation (uncorrelated is good)



Principal Component Analysis (PCA)

Linear methods

Classical techniques: Principal Component Analysis (PCA) or Multidimensional Scaling (MDS) aim to reveal underlying structures in data that may lie on or near a linear subspace.

PCA finds a low-dimensional embedding that best preserves the variance measured in the m-dimensional space

MDS computes an embedding that preserves pairwise Euclidean distances.

\+ Computationally efficient, easy to implement

\− Restricted to linear relations

Principal Component Analysis (PCA)

- We have m features (genes) and assume that the data points (cells) are scattered around and primarily concentrated near a linear subspace of dimension $p \ll m$ (a p-dimensional plane).
- The new variables that span the subspace are called **principal components**, each principal component is a m-dimensional vector.
- For m-dimensional measurements we get m principal components, but we use only p of them. The choice of p depends on the percentage of variance/variation we would like to maintain in the smaller subspace.

Principal Component Analysis (PCA)

Goal: Reduce the dimensionality of data while preserving as much variance as possible.

Concept:

- PCA transforms the original data into a new coordinate system.
- The first principal component captures the direction of the highest variance.
- Each subsequent component captures the next highest variance, orthogonal to the previous ones.

Data Representation:

- Feature matrix: $\mathbf{X} \in \mathbb{R}^{n \times m}$
  - n: Number of samples (rows)
  - m: Number of features (columns)
- Center the data by subtracting the mean.

Principal Components in PCA

Finding Principal Components:

- Compute the **covariance matrix**:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$$

- Perform **eigen decomposition** of $\mathbf{C}$:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

- $\mathbf{v}_i$: Eigenvectors (principal components)
  λ_i: Eigenvalues (variance value)

Interpretation:

- Eigenvectors define the new axes (principal components).
- Eigenvalues indicate the importance (variance captured) of each component.

PCA Transformation

Projecting Data onto Principal Components:

- Select the top k (=2,3) eigenvectors to form a projection matrix $\mathbf{W}$:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

- Transform the original data:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}$$

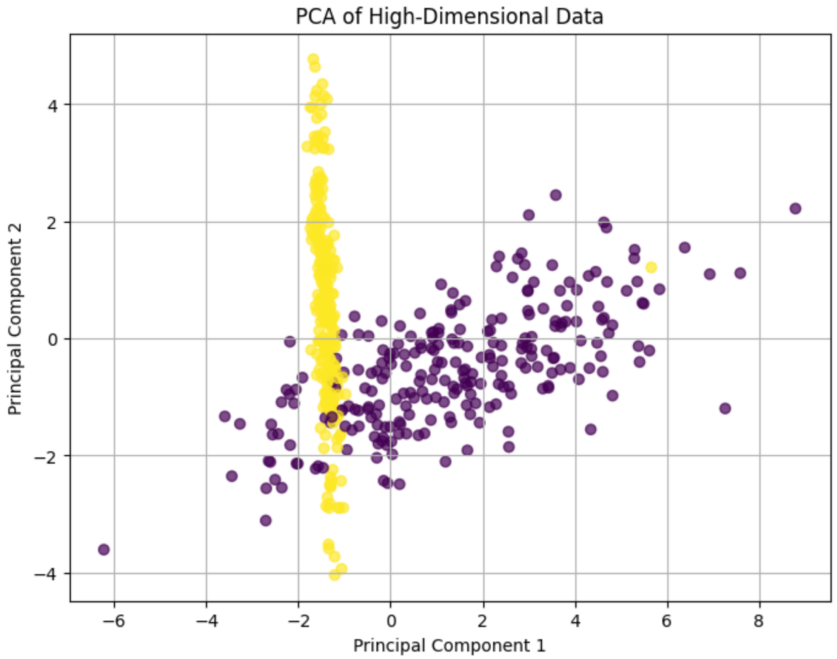
- $\mathbf{Z}$ is the lower-dimensional representation of $\mathbf{X}$.

Outcome:

- Data is now expressed in terms of principal components.
- Dimensionality is reduced with minimal loss of variance.

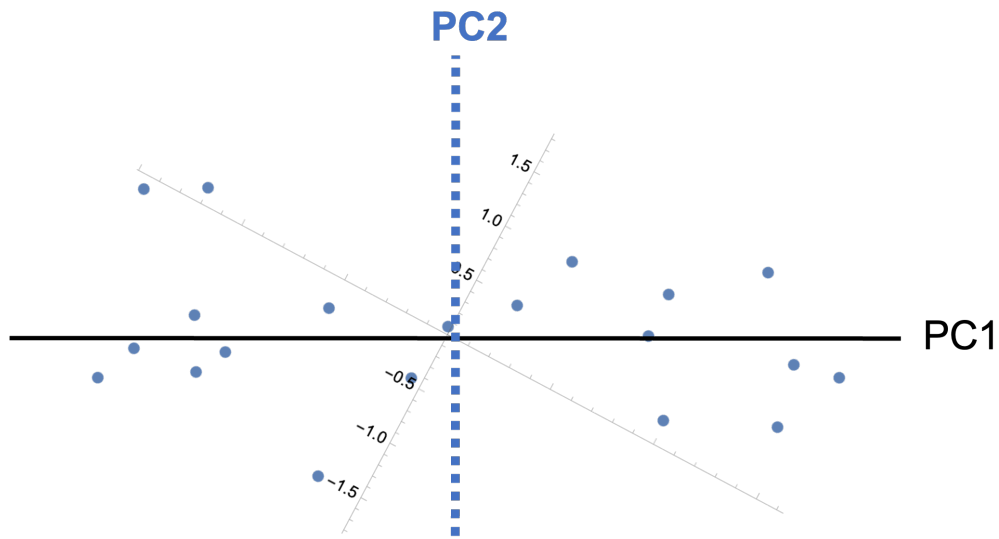
PCA Transformation

$$\mathbb{R}^{500\times 10} \rightarrow \mathbb{R}^{500\times 2}$$



Principal Component Analysis (PCA)

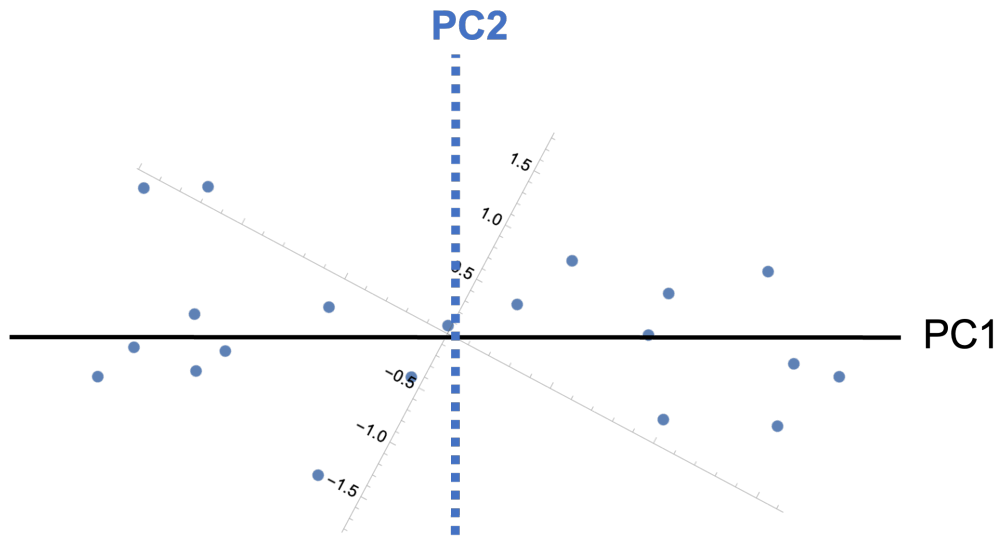
The principal components describe a new coordinate system:



We observe: $\lambda_1 > \lambda_2 \Rightarrow$ More variation in the first principal component.

Principal Component Analysis (PCA)

Total variation: $\lambda_1 + \lambda_2$. The ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$ describes the percentage of total variation explained by PC1.

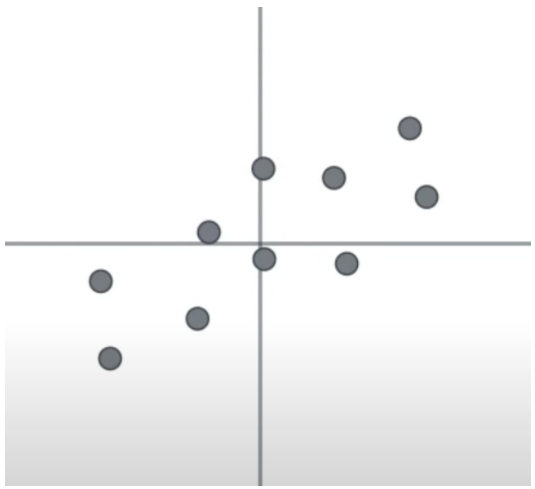


If $\frac{\lambda_1}{\lambda_1+\lambda_2}$ is large, then simply drawing PC1 and the projections of the data on PC1 is a good approximation of the 2-dim graph.

Principal Component Analysis (PCA)

An algorithm to compute the eigenvalues and eigenvectors

Center the data.



Compute the covariance between gene 1 and gene 2

$$\text{cov}(x_1, x_2) = \sum_{i=1}^{n} x_i(i)x_2(i)$$

and obtain the symmetric covariance matrix, where the variances are on the main-diagonal and the covariances on the off-diagonal elements

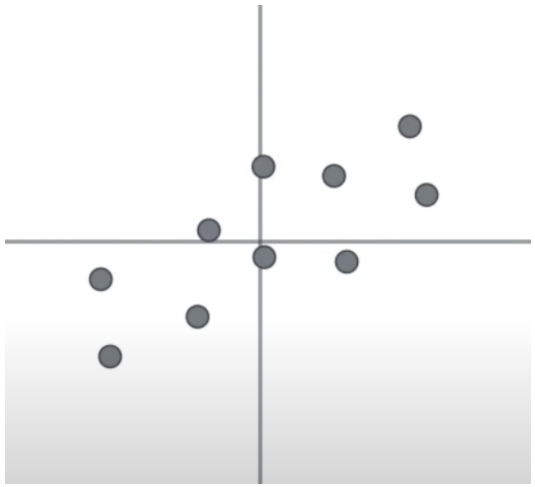
$$Cov = \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) \end{bmatrix}$$

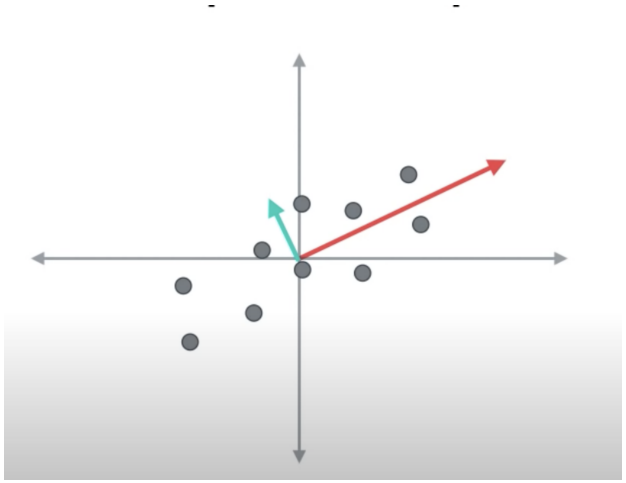
Principal Component Analysis (PCA)

An algorithm to compute the eigenvalues and eigenvectors

Compute the eigenvalues and eigenvectors of

$$Cov = \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) \end{bmatrix}$$





Principal Component Analysis (PCA)

Summary:

- **Input:** Large table, i.e, $\boldsymbol{C}$
- Column-wise normalize $\boldsymbol{C}$ by subtracting the mean for each $i = 1, \ldots, m$
- Use the normalized data matrix to compute the $m \times m$ covariance matrix
- Compute the m real eigenvalues and eigenvectors.
- Take the biggest eigenvalue(s) and project the high m-dimensional data on the smaller space.

PCA with Python

Don't worry, the computer is your friend.

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# 1. Standardize the data
scaler = StandardScaler()
C_scaled = scaler.fit_transform(C)

# 2. Apply PCA to get the first two components
pca = PCA(n_components=2)
C_pca = pca.fit_transform(C_scaled)

# 3. Plot the PCA results
plt.figure(figsize=(8, 6))
plt.scatter(C_pca[:, 0], C_pca[:, 1], alpha=0.7)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

Make sure that C is a proper feature matrix with samples as rows and features as columns.

Non-linear Embedding

PCA recap

Principal Component Analysis (PCA):

- A dimensionality reduction technique that projects high-dimensional data into a lower-dimensional space.
- Identifies directions (principal components) that maximize variance in the data.
- Each principal component is a linear combination of the original variables.
- PCA ensures that the principal components are orthogonal to each other.

Key Applications:

- Visualizing high-dimensional data in 2D or 3D.
- Preprocessing for clustering or machine learning tasks.
- Noise reduction by retaining only significant components.

PCA recap

Pros:

- Reduces data dimensionality while preserving most of the variance.
- Simplifies complex datasets, making patterns easier to interpret.
- Enhances computational efficiency for downstream tasks.
- Does not require labeled data (unsupervised method).

Cons:

- Assumes linear relationships in the data.
- Sensitive to outliers and scaling of data.
- Principal components may not have a clear physical interpretation.
- Loses interpretability of original features after projection.

t-SNE

t-SNE: t-Distributed Stochastic Neighbor Embedding

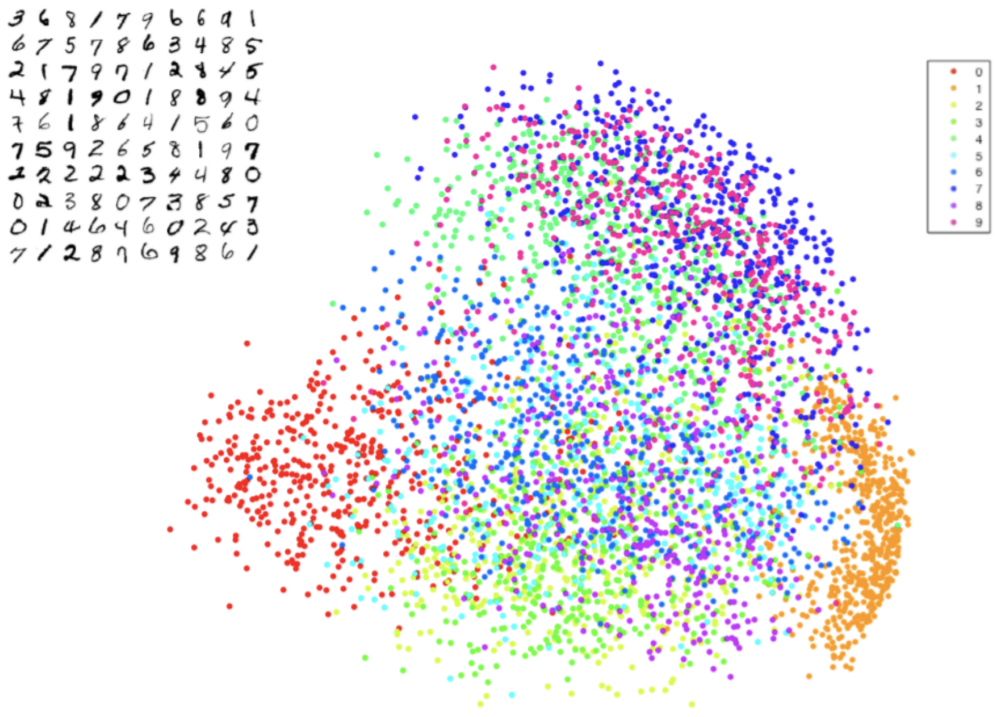
- An alternative dimensionality reduction algorithm.
- Aims to preserve **local structure**:
  - Low-dimensional neighborhoods resemble original high-dimensional neighborhoods.
- Unlike PCA, primarily used for **visualization**.

Key Differences from PCA:

- PCA seeks to find **global structure**: a low-dimensional subspace that may lead to local inconsistencies.
  - Far away points can become nearest neighbors.
- t-SNE focuses on **local structure**.

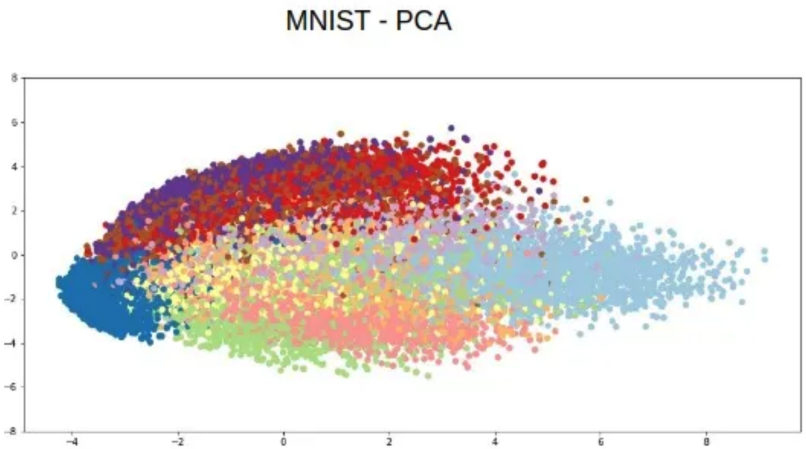
t-SNE: t-Distributed Stochastic Neighbor Embedding

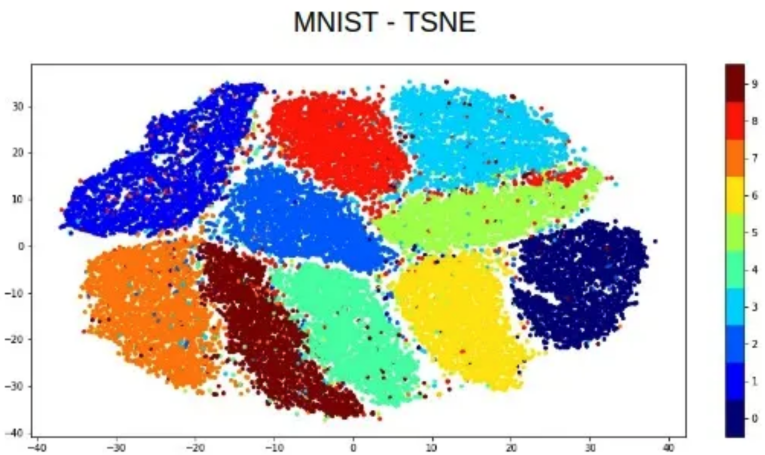
The MNIST dataset is a collection of 70,000 grayscale images of handwritten digits ($0 - 9$), each 28x28 pixels, commonly used as a benchmark for training and testing machine learning and computer vision models.



t-SNE: t-Distributed Stochastic Neighbor Embedding

t-SNE preserves local structures, making it better for visualizing neighborhood relationships than PCA.





t-SNE procedure

Goal: Compute pairwise similarities for all points in the high-dimensional space ($\mathbb{R}^d$).

- Construct a $N \times N$ similarity matrix, where N is the number of points.
- Similarity is modeled using a Gaussian distribution centered at each point:

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

- σ_i: Perplexity-based bandwidth for each point.
- $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$: Symmetric joint probabilities.

Key Idea: The probabilities p_{ij} represent the similarity between points i and j in the original space.

t-SNE procedure

Goal: Map the high-dimensional similarities to a low-dimensional space ($\mathbb{R}^2$ or $\mathbb{R}^3$) while preserving local structure.

- Similarities in the low-dimensional space are modeled using a **t-distribution**:

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

- The t-distribution has heavier tails than the Gaussian, preventing crowding of distant points.
- q_{ij}: Pairwise similarity in the low-dimensional space.

Why t-Distribution?

- Ensures distant points in the original space remain relatively far apart in the embedding.
- Balances local and global structure for better visualization.

t-SNE procedure

Goal: Minimize the difference between pairwise similarities in high- and low-dimensional spaces.

- The mismatch is measured using the Kullback-Leibler (KL) divergence:

$$C = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

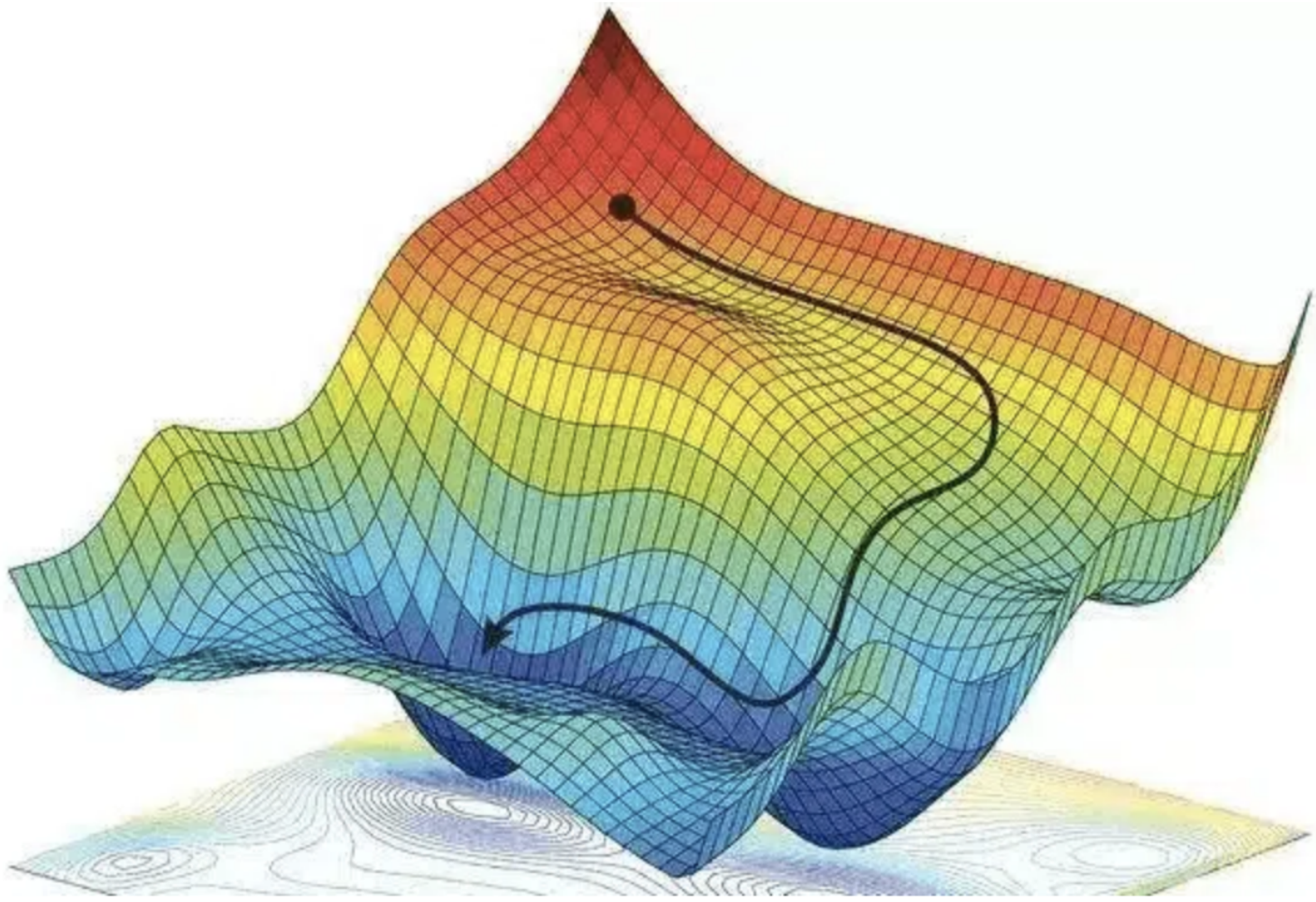
- p_{ij}: Pairwise similarities in high-dimensional space.
- q_{ij}: Pairwise similarities in low-dimensional space.

Key Points:

- KL divergence ensures that high-dimensional neighbors remain close in low-dimensional embedding.
- Optimization is performed using gradient descent to minimize C.

t-SNE procedure

Stochastic gradient descent optimizes the KL divergence by updating embeddings iteratively using gradients from random data points.



t-SNE procedure

Gradient of the KL Divergence:

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

- $p_{ij} - q_{ij}$: Difference between high- and low-dimensional similarities.
- $y_i - y_j$: Difference in low-dimensional coordinates.
- $\left(1 + \|y_i - y_j\|^2\right)^{-1}$: Weight from the t-distribution.

Update Rule for y_i:

$$y_i \leftarrow y_i - \eta \cdot \frac{\partial C}{\partial y_i}$$

- η: Learning rate controlling the step size.
- Iteratively adjusts y_i to minimize KL divergence C.

UMAP

UMAP: Uniform Manifold Approximation and Projection

- A non-linear dimensionality reduction technique.
- Preserves both local and global structure of data.
- Designed for high-dimensional data visualization and analysis.
- Based on principles from topology and manifold theory.

UMAP: Uniform Manifold Approximation and Projection

Key Goals:

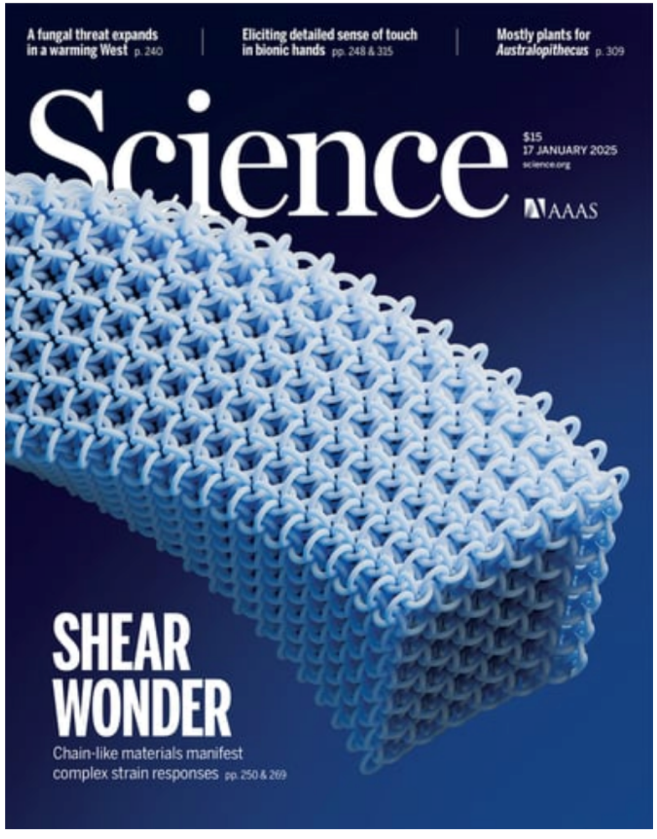
- Embed high-dimensional data into a low-dimensional space (e.g., 2D or 3D).
- Retain meaningful relationships between data points.

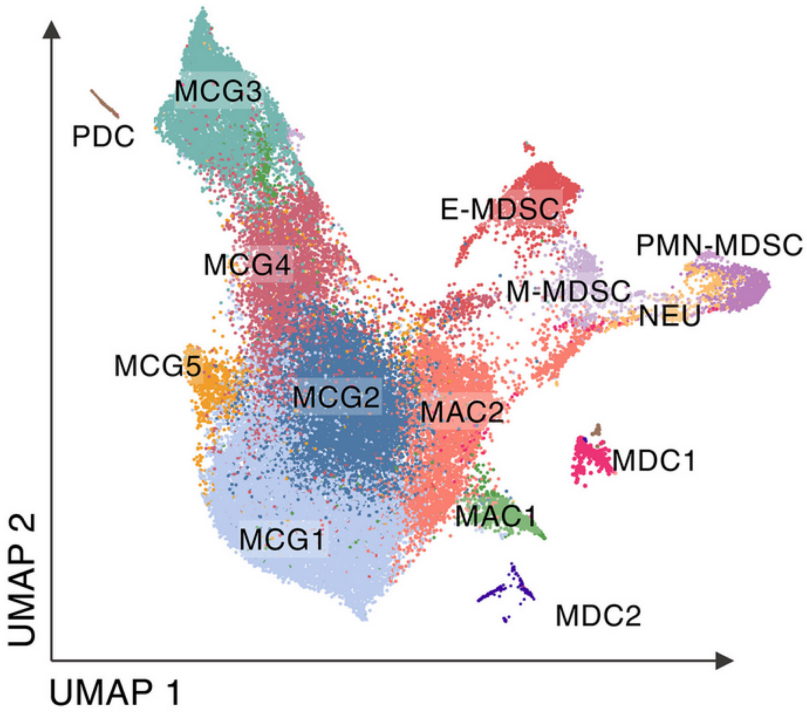
Applications:

- Visualizing clusters in high-dimensional datasets (e.g., single-cell RNA-seq).
- Preprocessing for machine learning tasks.
- Exploring hidden patterns in complex data.

UMAP

UMAP is a highly versatile technique, widely adopted in bio-medical research, and frequently featured in high-impact journals like *Science*.

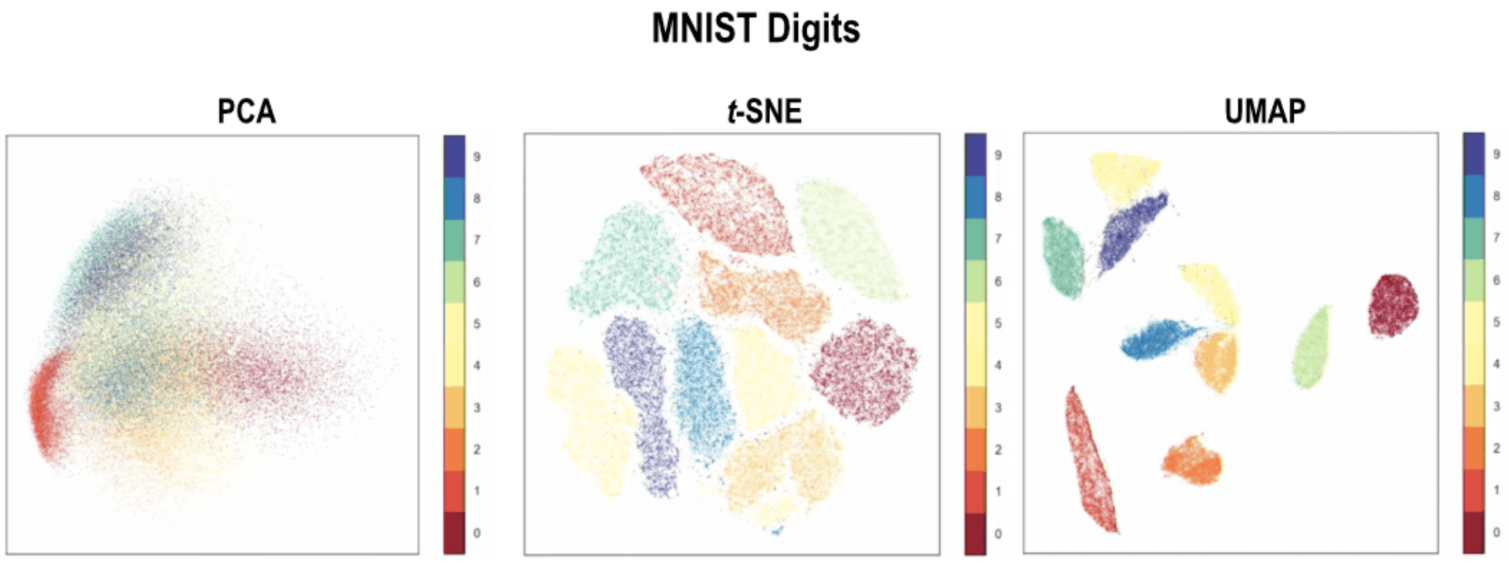




Christina Jackson *et al*.
Distinct myeloid-derived suppressor cell populations in human glioblastoma.
Science **387** (2025)

UMAP on MNIST

The MNIST dataset is a collection of 70,000 grayscale images of handwritten digits ($0 - 9$), each 28x28 pixels, commonly used as a benchmark for training and testing machine learning and computer vision models.



Constructing the k-NN Graph for UMAP

UMAP begins by constructing a weighted k-nearest neighbor (k-NN) graph to capture local relationships in the high-dimensional data.

- Select a metric (euclidean, cosine,..., or use your own)
- Compute pairwise distances between points in X.
- For each point x_i, determine its k-nearest neighbors.
- Assign weights to edges:

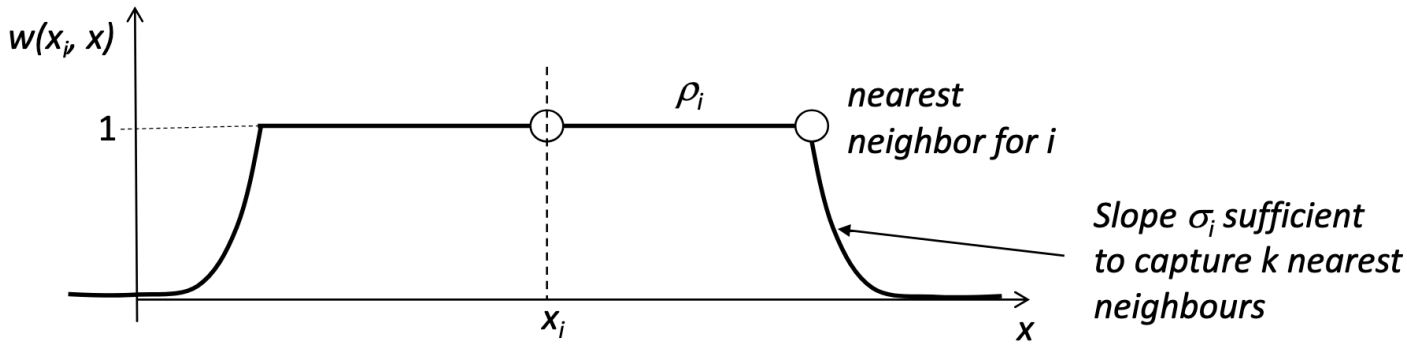
$$w(x_i, x_j) = \exp\left(-\frac{\max(0, d(x_i, x_j) - \rho_i)}{\sigma_i}\right),$$

with

$$\rho_i = \min\{d(x_i, x_j) \mid 1 \leq j \leq k, d(x_i, x_j) > 0\}.$$

Properties of the k-NN Graph in UMAP

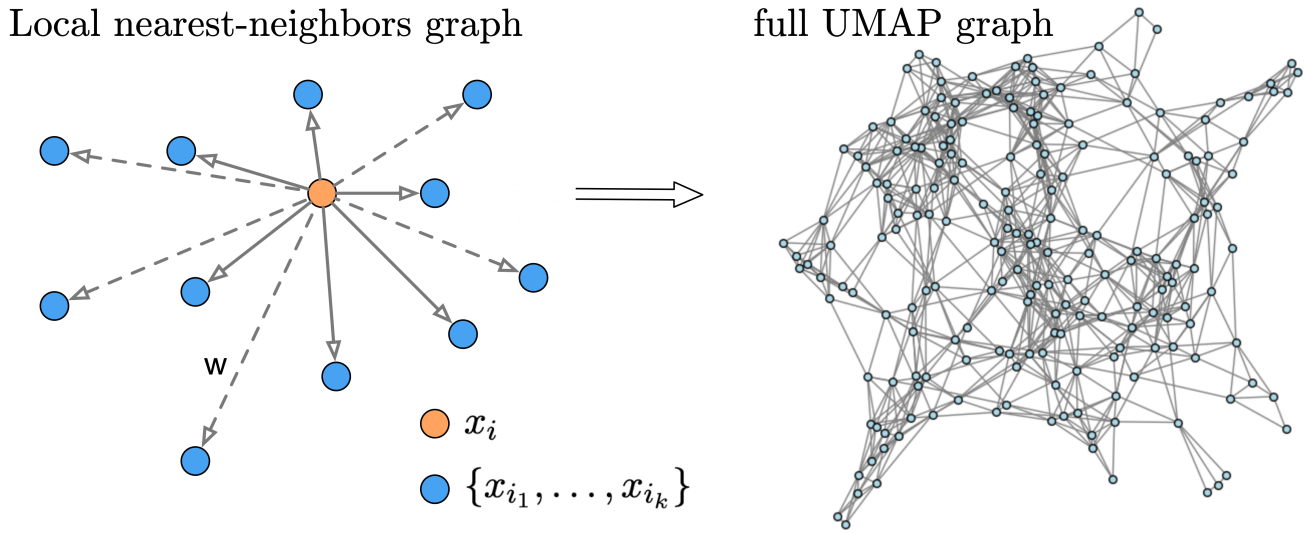
- **Local Connectivity:** The parameter ρ_i ensures that each point has at least one connection, preventing isolated nodes.
- **Adaptive Bandwidth:** The perplexity σ indicates how broadly the similarity kernel accounts for the distance between two data points.
- **Asymmetric Relationships:** Initial similarities may be asymmetric; UMAP symmetrizes the graph to ensure consistency.



Layout of the k-NN Graph

Force directed layout algorithm:

- nodes (data points) behave like equally charged particles $\Rightarrow$ repelling force
- edges act like springs inducing an attractive force between connected nodes
- Convergence to a local minimum is ensured by gradually reducing attractive and repulsive forces



UMAP Parameters

Key Parameters in UMAP:

- `n_neighbors`:
  - Controls the size of the local neighborhood used for manifold approximation.
  - Affects the balance between local and global structure in the embedding.
  - **Small values:** Preserve fine-grained local structure.
  - **Large values:** Capture broader global relationships.
- `min_dist`:
  - Determines the minimum spacing between points in the low-dimensional embedding.
  - Affects the compactness of clusters:
    - **Low values:** Points are tightly packed, emphasizing local structure.
    - **High values:** Points are more evenly spread, preserving global structure.

Layout of the k-NN Graph

Example from my work:
The human interactome - protein-protein interaction network.
Samples are proteins; features are associated diseases (e.g. PTEN: Cowden Syndrome, Bannayan-Riley-Ruvalcaba Syndrome, Proteus Syndrome, prostate, endometrial, and glioblastoma cancers)