Dimensionality Reduction

Iran University of Science and Technology

M. S. Tahaei PhD.

Fall 2024

Text Books

- DATA MINING : CONCEPTS AND TECHNIQUES, Jiawei Han, 4TH, 2022
- Pattern Recognition and Machine Learning, C. Bishop, Springer, 2006.
- Machine Learning,T. Mitchell, MIT Press,1998.

Course information and Policies (cont.)

Grading Policy:

Homework and Programming Assignments	30%
Short Quizzes	10%
Midterm exam	25+5%
Final exam (comprehensive)	35+5%

Extra Point: Submitted Paper on ISI Journal or Workshop, +(15-20)%

Dimensionality Reduction

- What Is Dimensionality Reduction?
- Dimensionality Reduction Methods
  - Principal Component Analysis
  - Attribute Subset Selection
  - Nonlinear Dimensionality Reduction Methods

What Is Dimensionality Reduction?

- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Reducing the number of random variables under consideration, via obtaining a set of principal variables
- **Advantages of dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization

Dimensionality Reduction Methods

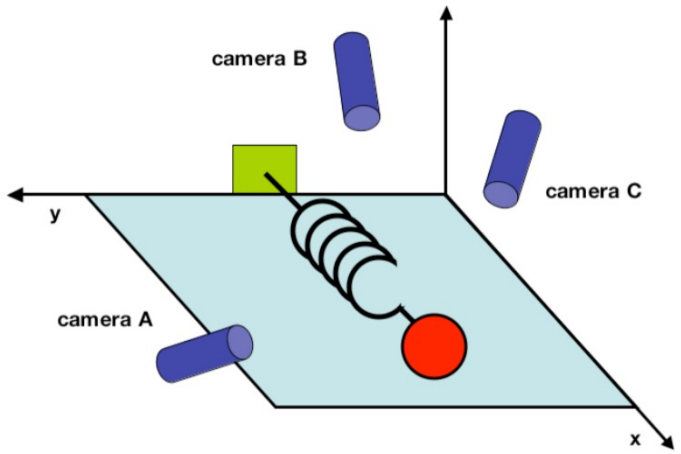
- **Dimensionality reduction methodologies**
  - Feature selection: Find a subset of the original variables (or features, attributes)
  - Feature extraction: Transform the data in the high-dimensional space to a space of fewer dimensions
- **Some typical dimensionality reduction methods**
  - Principal Component Analysis
  - Attribute Subset Selection
  - Nonlinear Dimensionality Reduction

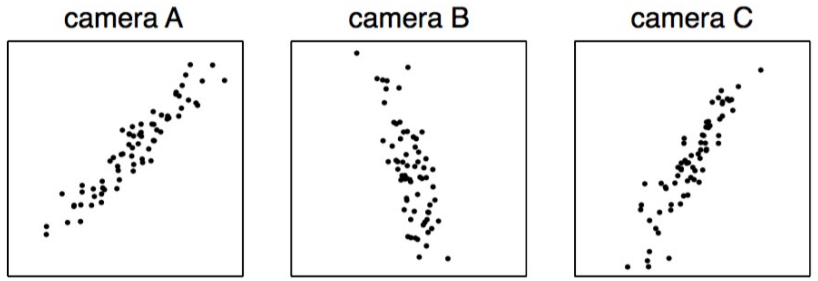
Dimensionality Reduction Methods (SOTA)

- **PCA:** Linear method maximizing variance through eigenvalue decomposition of the covariance matrix.
- **Dual PCA:**
- **KPCA:** Extends PCA to non-linear relationships using kernels to map data to higher dimensions.
- **UMAP:** Non-linear method preserving local and global structure through graph-based optimization.
- **t-SNE:** Non-linear method emphasizing local structure by minimizing the divergence between high-dimensional and low-dimensional distributions.

Principal Component Analysis (PCA)

- **PCA:** A statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
- The original data are projected onto a much smaller space, resulting in dimensionality reduction
- **Method:** Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space

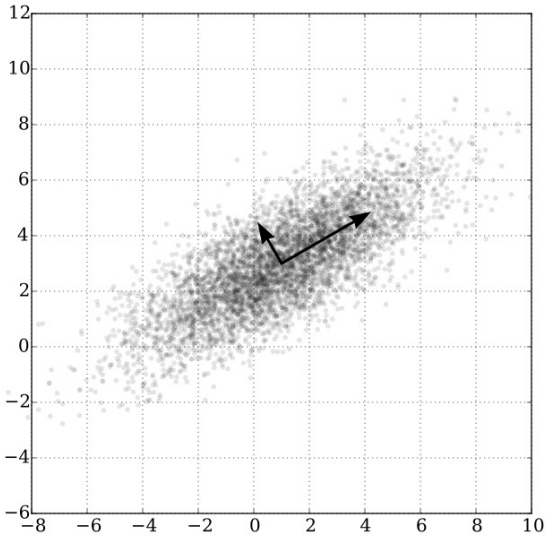




Ball travels in a straight line. Data from three cameras contain much redundancy

Principal Component Analysis (Method)

- **Given N data vectors from n-dimensions, find k ≤ n orthogonal vectors (principal components) best used to represent data**
  - Normalize input data: Each attribute falls within the same range
  - Compute k orthonormal (unit) vectors, i.e., principal components
  - Each input data (vector) is a linear combination of the k principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance (i.e., using the strongest principal components, to reconstruct a good approximation of the original data)
- **Works for numeric data only**



Ack. Wikipedia: Principal Component Analysis

Variance for Single Variable (Numerical Data)

- **The variance of a random variable X provides a measure of how much the value of X deviates from the mean or expected value of X:**
  - $$\sigma^2 = \mathrm{var}(X) = E[(\mathrm{X} - \mu)^2] = \begin{cases} \sum_x (x-\mu)^2 f(x) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx & \text{if } X \text{ is continuous} \end{cases}$$
  - where σ2 is the variance of X, σ is called standard deviation
    - μ is the mean, and μ = E[X] is the expected value of X
  - That is, variance is the expected value of the square deviation from the mean
  - It can also be written as:

$$\sigma^2 = \mathrm{var}(X) = E[(\mathrm{X} - \mu)^2] = E[\mathrm{X}^2] - \mu^2 = E[\mathrm{X}^2] - [E(x)]^2$$

- **Sample variance**

$$s^2 = \frac{1}{n}\sum_i^n (x_i - \hat{\mu})^2 \qquad s^2 = \frac{1}{n-1}\sum_i^n (x_i - \hat{\mu})^2$$

Covariance for Two Variables

- **Covariance between two variables X1 and X2**

$$\sigma_{12} = E[(X_1 - \mu_1)(X_2 - \mu_2)] = E[X_1 X_2] - \mu_1 \mu_2 = E[X_1 X_2] - E[X_1]E[X_2]$$

  - where μ1 = E[X1] is the respective mean or expected value of X1; similarly for μ2

- **Sample covariance between X1 and X2:** $\hat{\sigma}_{12} = \frac{1}{n}\sum_{i=1}^{n}(x_{i1} - \widehat{\mu_1})(x_{i2} - \widehat{\mu_2})$
- **Sample covariance is a generalization of the sample variance:**

$$\hat{\sigma}_{11} = \frac{1}{n}\sum_{i=1}^{n}(x_{i1} - \widehat{\mu_1})(x_{i1} - \widehat{\mu_1})$$

- **Positive covariance: If σ12 > 0**
- **Negative covariance: If σ12 < 0**

Correlation between Two Numerical Variables

- Correlation between two variables X1 and X2 is the standard covariance, obtained by normalizing the covariance with the standard deviation of each variable

$$\rho_{12} = \frac{\sigma_{12}}{\sigma_1 \sigma_2} = \frac{\sigma_{12}}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

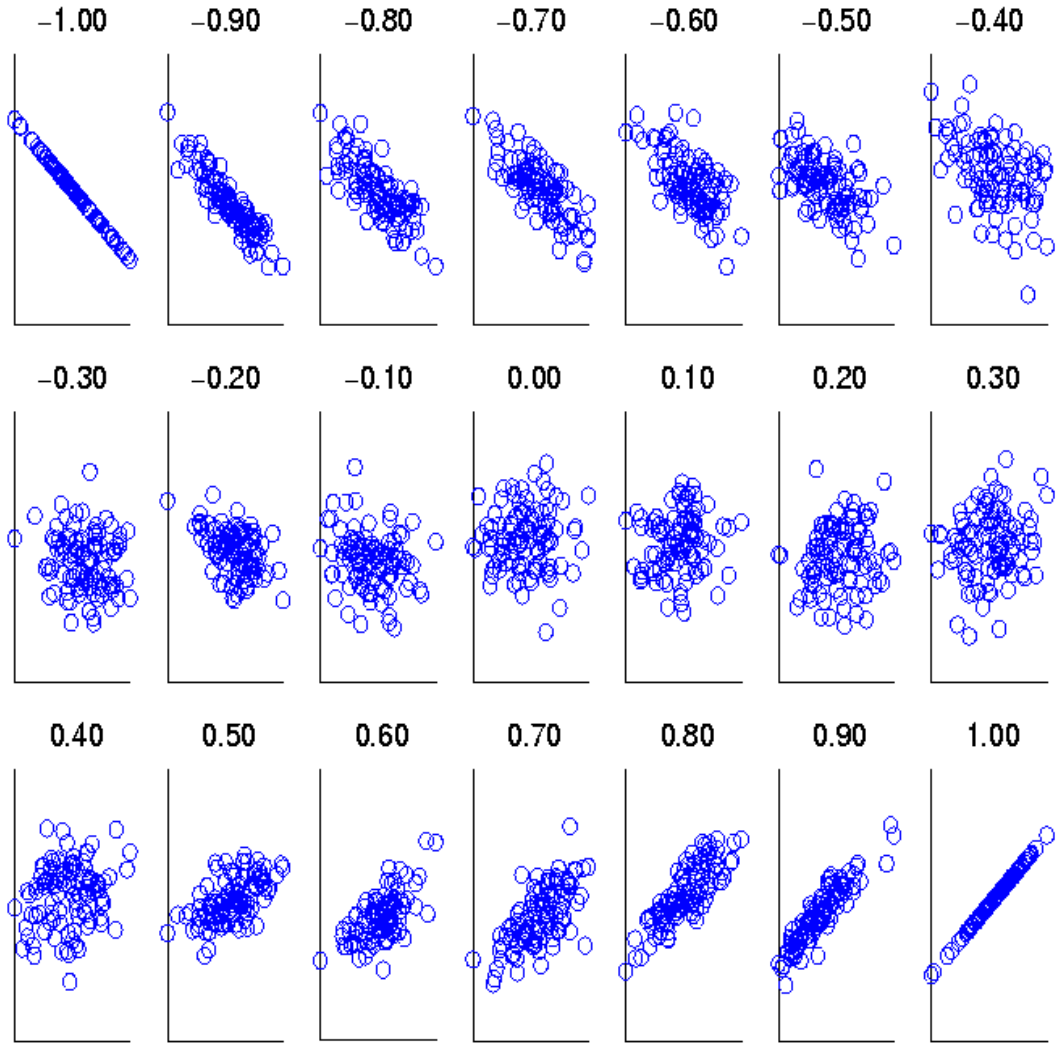
- **Sample correlation for two attributes X1 and X2:**

$$\hat{\rho}_{12} = \frac{\hat{\sigma}_{12}}{\hat{\sigma}_1 \hat{\sigma}_2} = \frac{\sum_{i=1}^{n} (x_{i1} - \hat{\mu}_1)(x_{i2} - \hat{\mu}_2)}{\sqrt{\sum_{i=1}^{n} (x_{i1} - \hat{\mu}_1)^2 \sum_{i=1}^{n} (x_{i2} - \hat{\mu}_2)^2}}$$

  - where n is the number of tuples, μ1 and μ2 are the respective means of X1 and X2 , σ1 and σ2 are the respective standard deviation of X1 and X2

- **If ρ12 > 0: A and B are positively correlated (X1's values increase as X2's)**
  - The higher, the stronger correlation
- **If ρ12 = 0: independent (under the same assumption as discussed in co-variance)**
- **If ρ12 < 0: negatively correlated**

Visualizing Changes of Correlation Coefficient



- **Correlation coefficient value range: [–1, 1]**
- **A set of scatter plots shows sets of points and their correlation coefficients changing from –1 to 1**

PCA vs Others

Linear Discriminant Analysis (LDA)

- LDA is a supervised version of PCA. While PCA maximizes variance without regard to class labels, LDA maximizes the separation between classes in the data, using similar linear projections.
- LDA is used in classification tasks, particularly in fields like **Natural Language Processing** and **Pattern Recognition**, where it helps classify data into distinct categories.

Autoencoders (AEs)

- Autoencoders, particularly **linear autoencoders**, function similarly to PCA in that they reduce the dimensionality of data by compressing it into a lower-dimensional space and then reconstructing it.
- AEs are widely used in deep learning for unsupervised learning, data compression, and denoising tasks. Unlike PCA, which is purely linear, AEs can handle non-linear data, making them more powerful in modern applications like image and video analysis.

PCA vs Others

Singular Value Decomposition (SVD)

- PCA is often performed using **SVD** of the data matrix, as PCA decomposes data into orthogonal components that explain the variance. SVD similarly breaks a matrix down into singular values and vectors, capturing data's essential structure.

Eigenvalues and Eigenvectors

- PCA works by finding the eigenvalues and eigenvectors of the covariance matrix, which capture the direction and magnitude of variance in the data.

Covariance Matrix

- The variance and covariance information for the two variables X_1 and X_2 can be summarized as 2 X 2 covariance matrix as

$$\Sigma = E[(\mathbf{X}-\mu)(\mathbf{X}-\mu)^T] = E[\begin{pmatrix} X_1-\mu_1 \\ X_2-\mu_2 \end{pmatrix}(X_1-\mu_1 \quad X_2-\mu_2)]$$

$$= \begin{pmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] \end{pmatrix}$$

$$= \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix}$$

- Generalizing it to *d* dimensions, we have,

$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} & \cdots & S_{1d} \\ S_{21} & S_{22} & \cdots & S_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ S_{d1} & S_{d2} & \cdots & S_{dd} \end{pmatrix} \qquad \mathbf{\Sigma} = E[(\mathbf{X}-\mu)(\mathbf{X}-\mu)^T] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{pmatrix}$$

Covariance Matrix

- **The covariance matrix S is a d×d matrix:**

$$S_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ji} - \mu_j)(x_{ki} - \mu_k)$$

$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} & \dots & S_{1d} \\ S_{21} & S_{22} & \dots & S_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ S_{d1} & S_{d2} & \dots & S_{dd} \end{pmatrix}$$

- Express the Variance in Matrix Form

$$\mathrm{Var}(\mathbf{U}) = \frac{1}{n-1}\sum_{i=1}^{n}(U_i - \bar{U})^2 \qquad \bar{U} = \frac{1}{n}\sum_{i=1}^{n} U_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{u}^T\mathbf{x}_i = \mathbf{u}^T\mu$$

$$\mathrm{Var}(\mathbf{U}) = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{u}^T\mathbf{x}_i - \mathbf{u}^T\mu)^2$$

$$\mathrm{Var}(\mathbf{U}) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{u}^T(\mathbf{x}_i - \mu)\right)^2$$

$$\mathrm{Var}(\mathbf{U}) = \mathbf{u}^T\left(\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T\right)\mathbf{u}$$

$$\mathrm{Var}(\mathbf{U}) = \mathbf{u}^T\mathbf{S}\mathbf{u}$$

Direct PCA Algorithm

Recover basis (PCs): Calculate $XX^\top = \sum_{i=1}^{n} x_i x_i^\top$ and let $U =$ eigenvectors of $XX^\top$ corresponding to the top p eigenvalues.

Encode training data: $Y = U^\top X$ where Y is a $p \times n$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = UU^\top X$.

Encode test example: $y = U^\top x$ where y is a p-dimensional encoding of x.

Reconstruct test example: $\hat{x} = Uy = UU^\top x$.

Dual PCA Algorithm

Recover basis: Calculate $X^\top X$ and let V = eigenvectors of $X^\top X$ corresponding to the top d eigenvalues. Let Σ = diagonal matrix of *square roots* of the top d eigenvalues.

Encode training data: $Y = U^\top X = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = XV\Sigma^{-1}\Sigma V^\top = XVV^\top$.

Encode test example: $y = U^\top x = \Sigma^{-1}V^\top X^\top x = \Sigma^{-1}V^\top X^\top x$ where y is a d dimensional encoding of x.

Reconstruct test example: $\hat{x} = Uy = UU^\top x = XV\Sigma^{-2}V^\top X^\top x = XV\Sigma^{-2}V^\top X^\top x$.

Kernel PCA Algorithm

Recover basis: Calculate $K = \Phi(X)^\top \Phi(X)$ using the kernel K and let $V =$ eigenvectors of $\Phi(X)^\top \Phi(X)$ corresponding to the top d eigenvalues. Let $\Sigma =$ diagonal matrix of *square roots* of the top d eigenvalues.

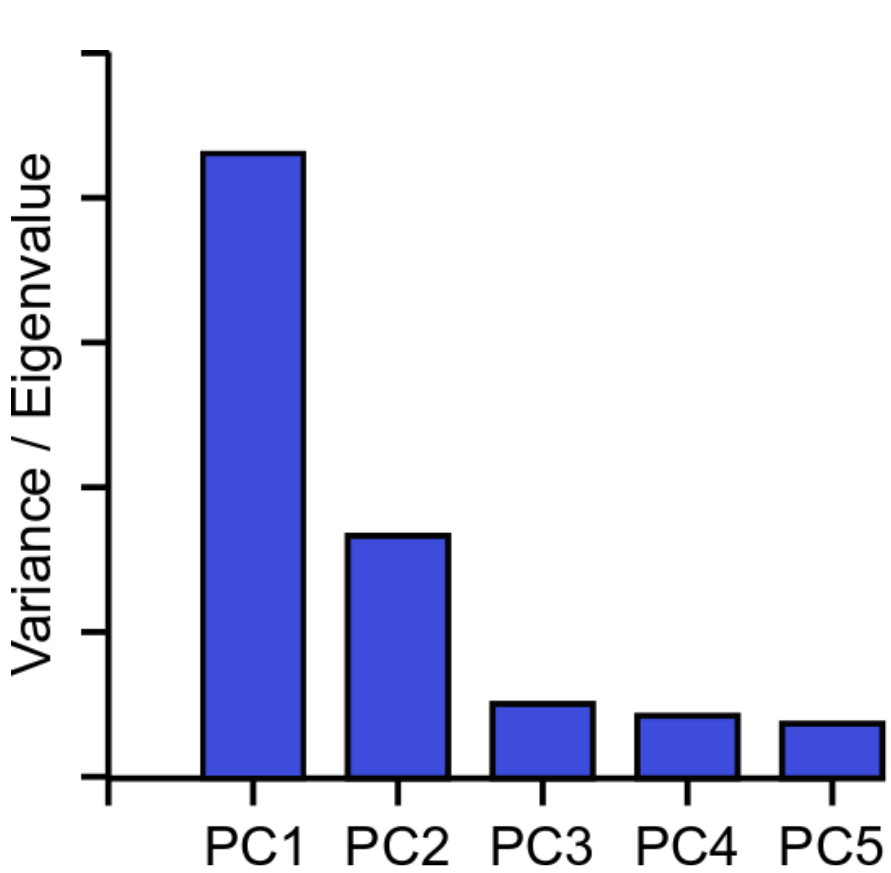
Encode training data: $Y = U^\top \Phi(X) = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data. This can be done since the calculation of Y does not depend on $\Phi(X)$

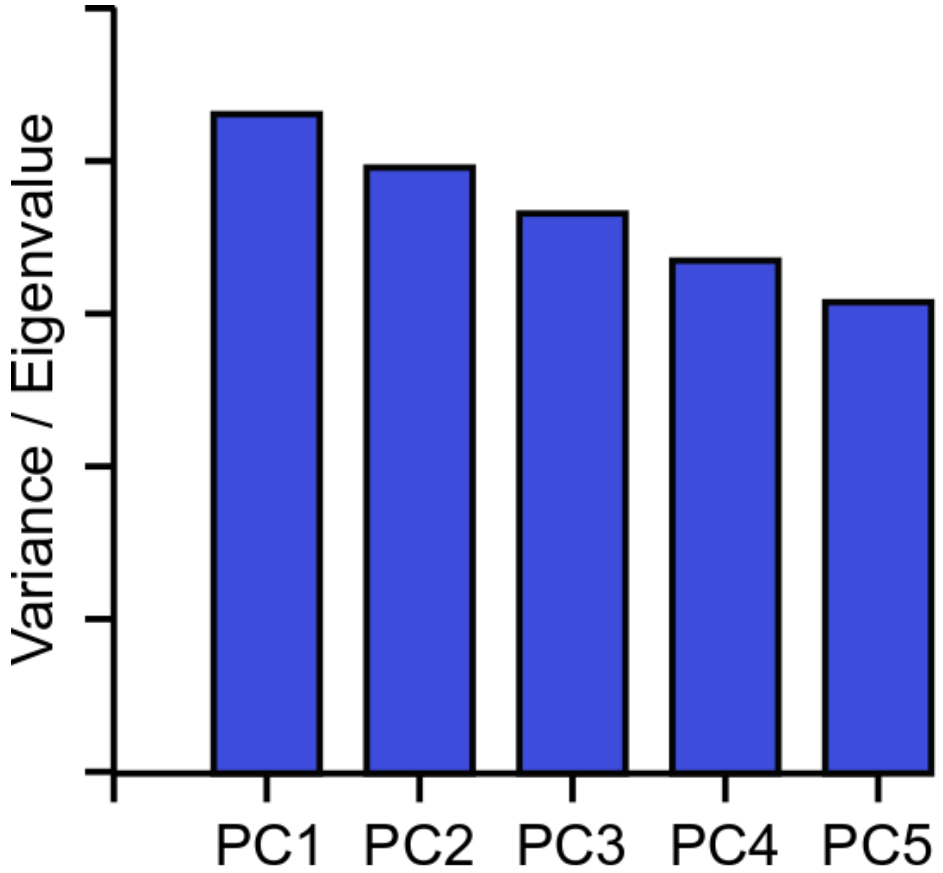
Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = \Phi(X)V\Sigma^{-1}\Sigma V^\top = \Phi(X)VV^\top$. $\Phi(X)$ is unknown so reconstruction **can not be done**.

Encode test example: $y = U^\top \Phi(x) = \Sigma^{-1} V^\top \Phi(X)^\top \Phi(x) = \Sigma^{-1} V^\top \Phi(X)^\top \Phi(x)$. This result can be calcuated because the quantity $\Phi(X)^\top \Phi(x) = K(X, x)$.

Reconstruct test example: $\hat{x} = Uy = UU^\top \Phi(x) = \Phi(X)V\Sigma^{-2}V^\top \Phi(X)^\top \Phi(x) = \Phi(X)V\Sigma^{-2}V^\top \Phi(X)^\top \Phi(x)$. $\Phi(X)$ is unknown so reconstruction **can not be done**.

PCA Challenges - Explaining Variance





Stochastic neighbor embedding (t-SNE) and UMAP

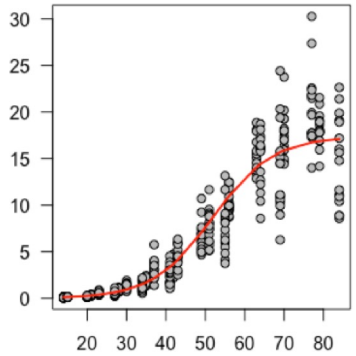
PCA vs tSNE

- PCA
  - Requires more than 2 dimensions
  - Thrown off by quantised data
  - Expects linear relationships
- tSNE
  - Can’t cope with noisy data
  - Loses the ability to cluster

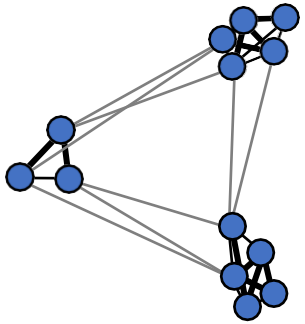
Answer: Combine the two methods, get the best of both worlds

- PCA
  - Good at extracting signal from noise
  - Extracts informative dimensions
- tSNE
  - Can reduce to 2D well
  - Can cope with non-linear scaling

Graphs



This is a PLOT



This is GRAPH
(a.k.a. network)

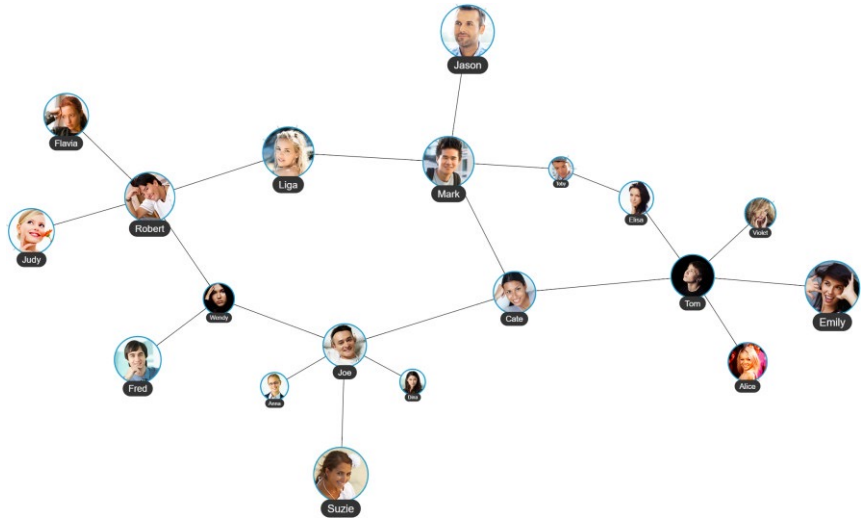
- Each dot is a cell (or a gene)
- Each line represents a connection between 2 cells
- Each connection can be weighted as a proximity between cells
  - Correlation (high and positive)
  - Euclidean distance (low)
  - etc.

Graph-based dimensionality reduction algorithms can be divided into 2 main steps:

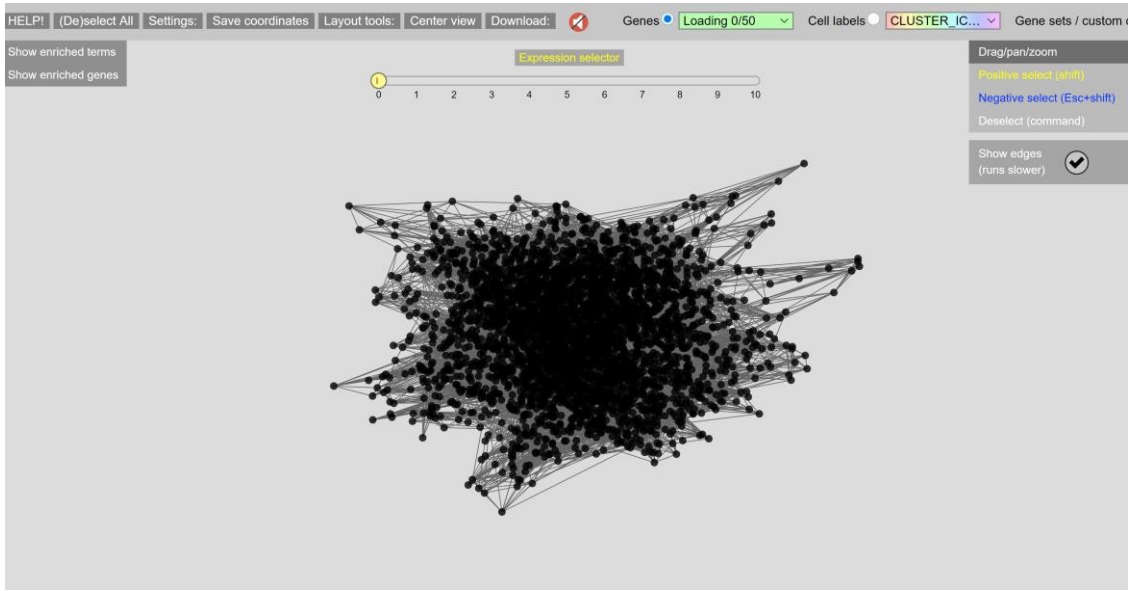
1. Construct a weighted graph based on the top k connections (a.k.a. k-nearest neighbors, KNN)
2. The low dimensional layout of the graph is computed and optimized

Using graph layouts to reduce data dimensionality

- Simple algorithm:
  - Compute the KNN-graph
  - Apply graph drawing (layouting) algorithm to visualize it in 2D (3D)



https://demo.zoomcharts.com/net-chart/examples/layout/layout-forced.html



https://www.ihes.fr/~zinovyev/mosaic/SPRING/springViewer.html?datasets/CHLA9_nufp

Problems with drawing knn-graphs

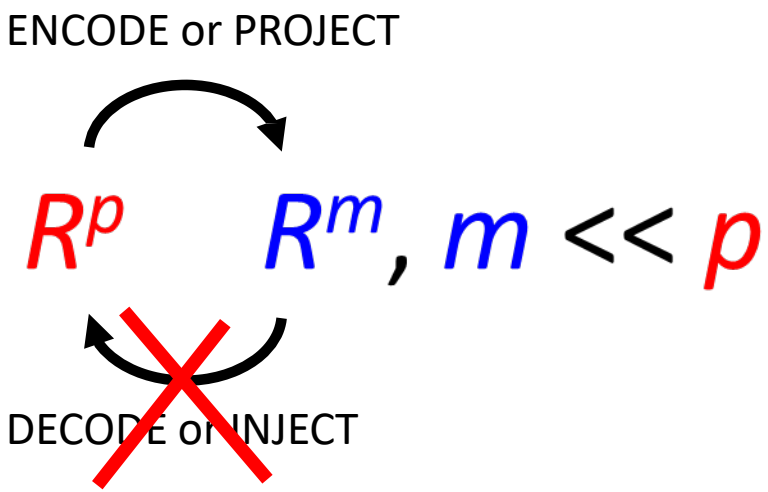
- In many situations, it is already a usefull solution for visualizing the data, but…

- The structure of knn graph heavily depends on the local density
- Point crowding
- The graph is very ‘rigid’

- Solutions:
  - Makes the structure of the neighbourhood graph more adapted to the density variations
  - Introduce a ‘softer’ probabilistic model in constructing the neighborhood graph – instead of a ‘hard’ connection a probability of being connected

- Two methods have become famous for this in recent years: t-SNE and UMAP

t-SNE and UMAP are projective methods

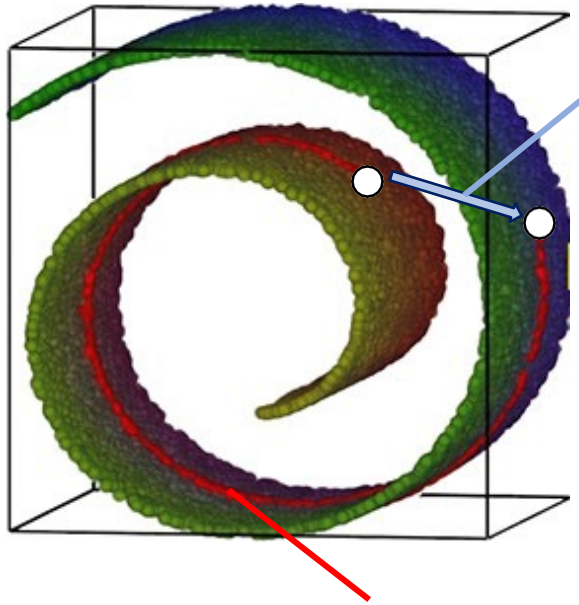


Variant 2: The projector is known only for $\boldsymbol{y} \in \boldsymbol{X}$

How t-SNE works

It is a graph-based NON-LINEAR dimensionality reduction

Manifold:



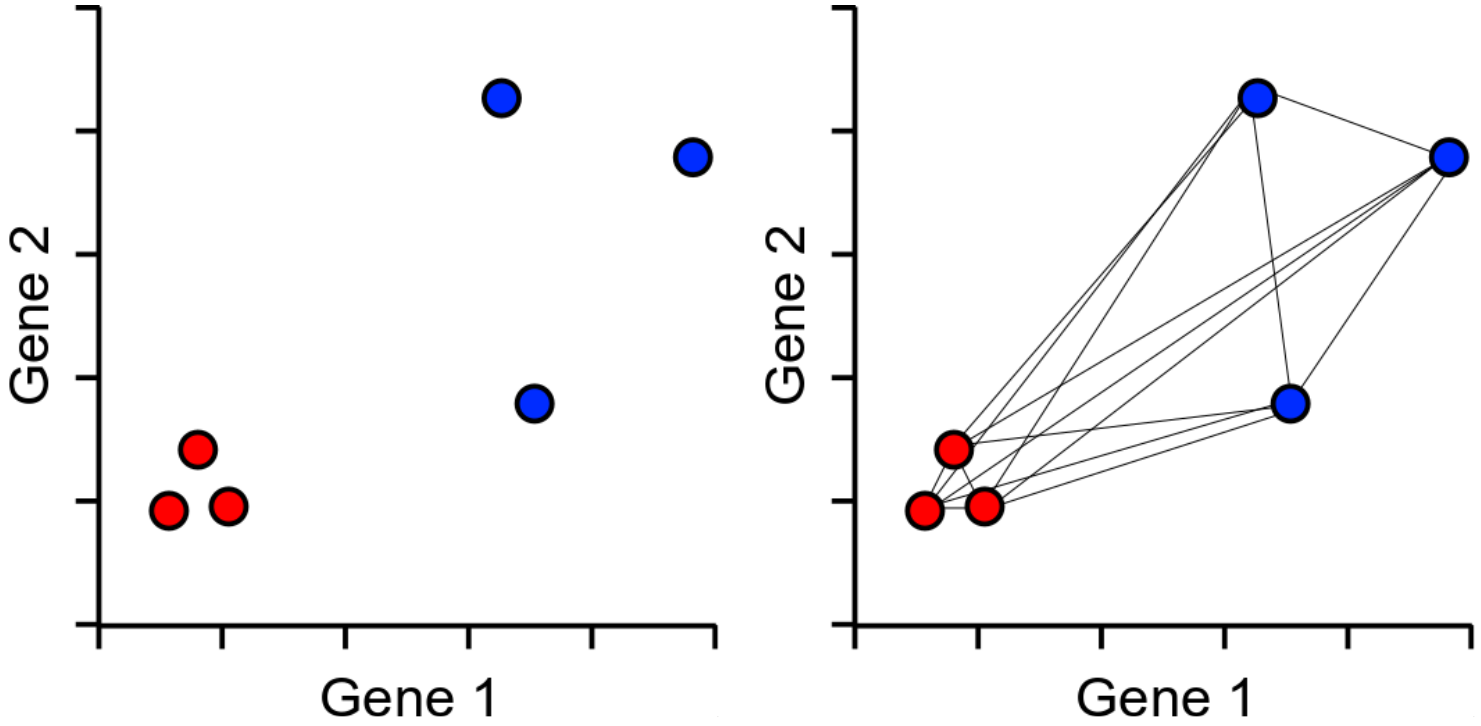
distance in PCA space (euclidean distance)

distance in t-SNE space

In other words, t-SNE calculates the distances based on the distance to the neighbor cell

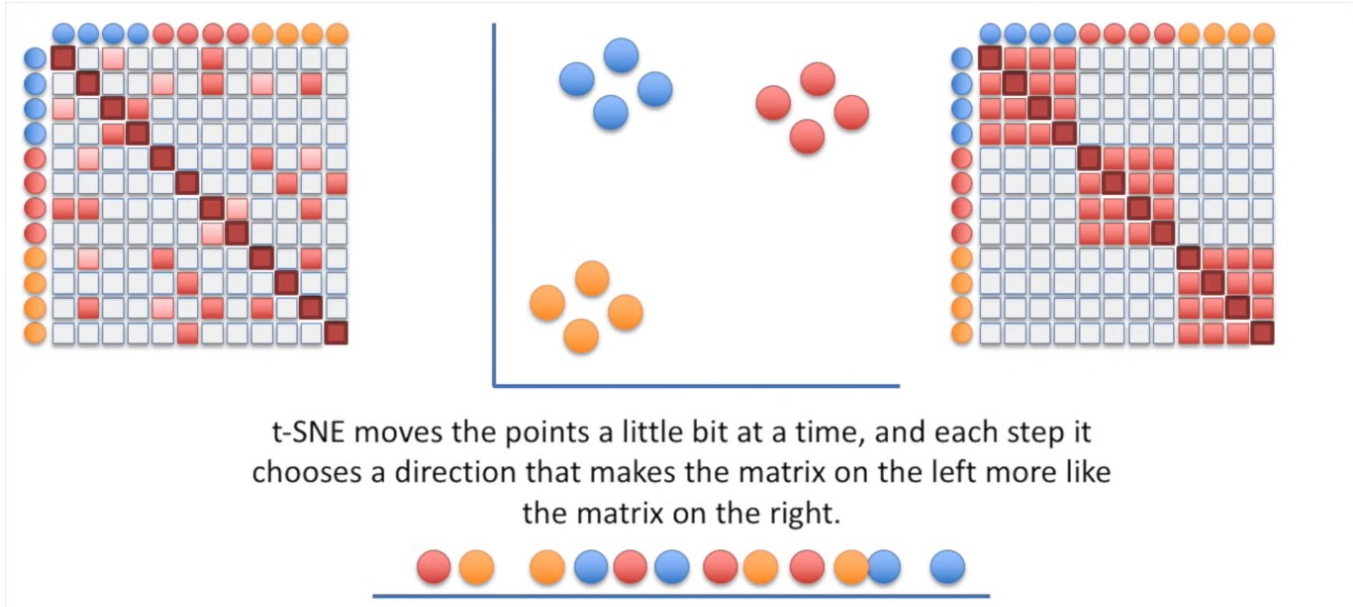
How does tSNE work?

- Based around all-vs-all table of pairwise cell to cell distances

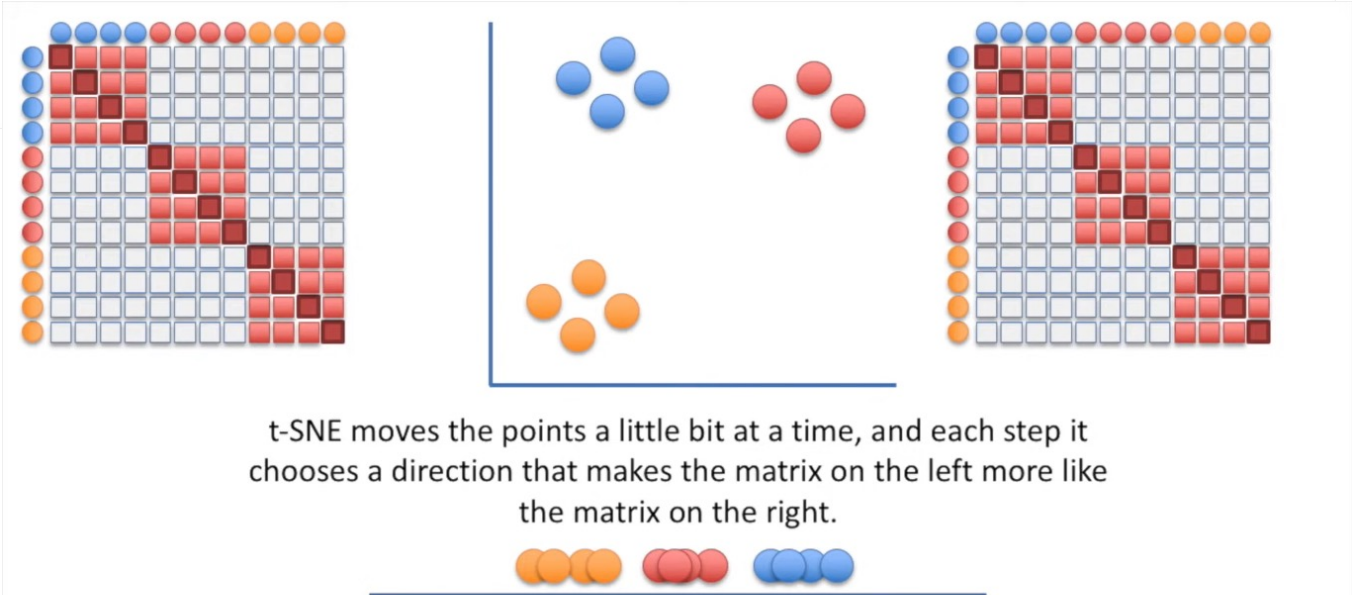


	0	10	10	295	158	153
	9	0	1	217	227	213
	1	8	0	154	225	238
	205	189	260	0	23	45
	248	227	246	44	0	54
	233	176	184	41	36	0

How t-SNE works?

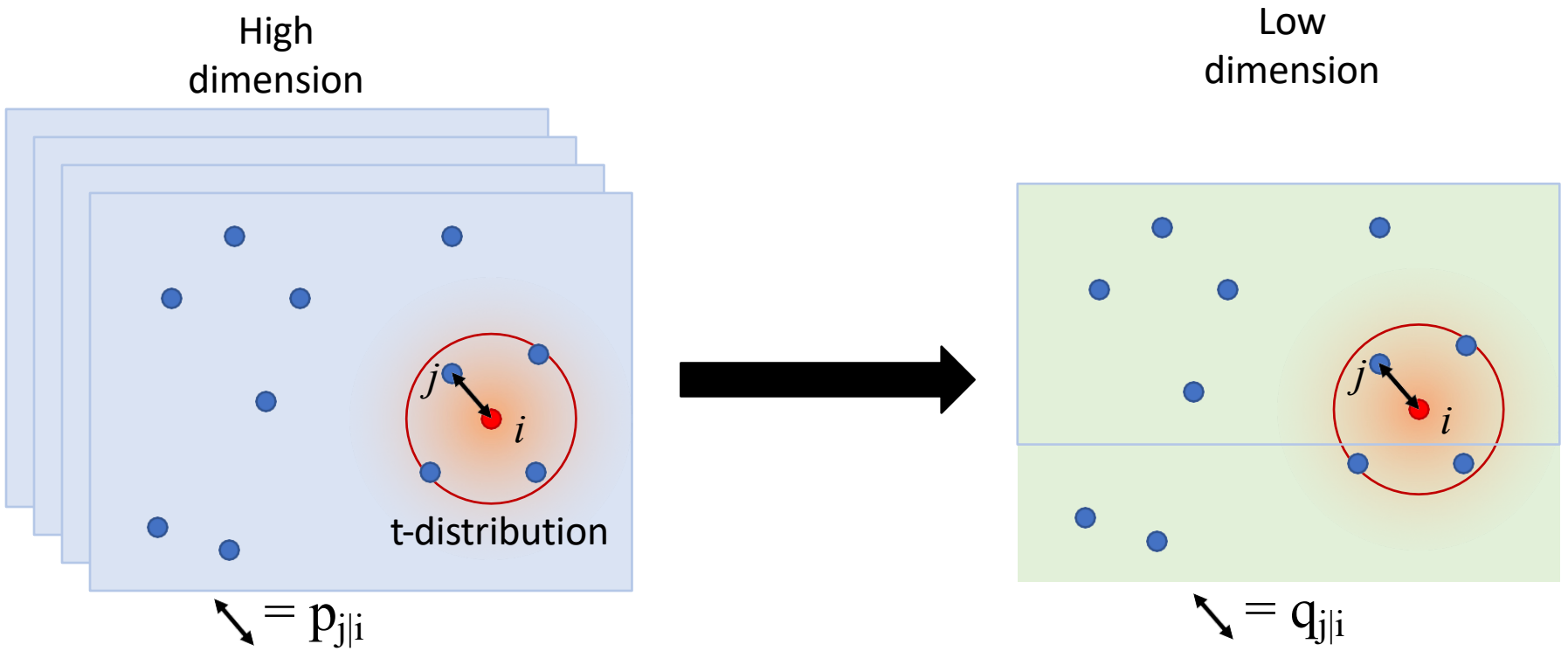


t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

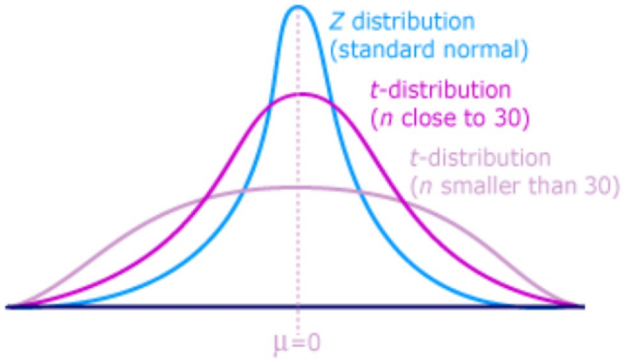


t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

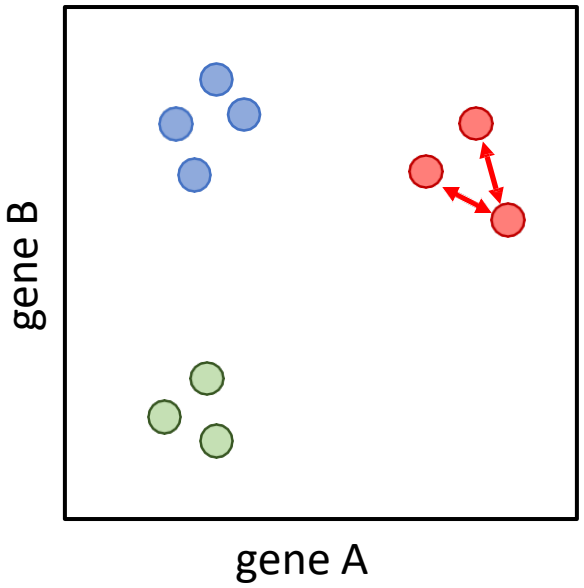
How t-SNE works

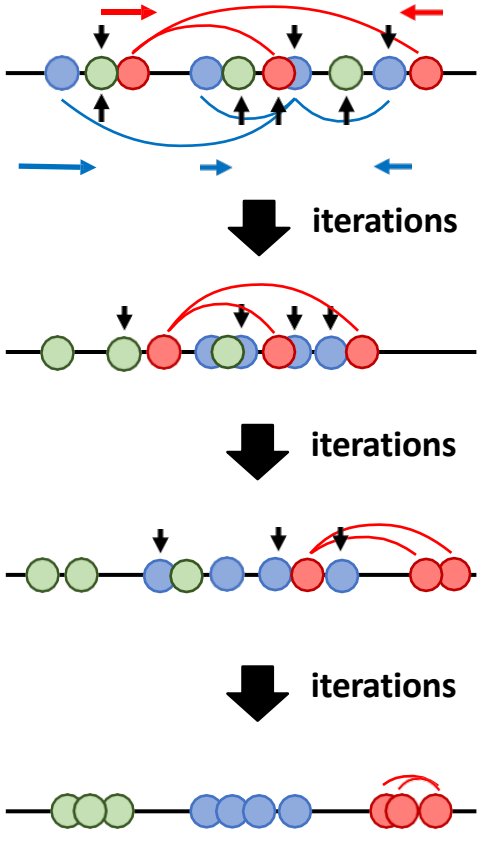


$p_{j|i}$ and $q_{j|i}$ measure the conditional probability that a point i would pick point j as it's nearest neighbor, in high (p) and low (q) dimensional space respectively.



How t-SNE works





Higher KL divergence
(cost / error)

Lower KL divergence
(cost / error)

The same concept applies to embedding into 2 dimensions

Gaussian Weight Formula

- Closer points j to i have smaller distances dij , resulting in larger exponent values and higher probabilities pj|i
- Farther points j from i have larger distances dij , resulting in smaller exponent values and lower probabilities pj|i

$$p_{j|i} = \frac{\exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d_{ik}^2}{2\sigma_i^2}\right)}$$

Where:

- d_{ij} is the **Euclidean distance** between points i and j.
- σ_i is the **bandwidth** or **scale parameter** for the Gaussian centered at point i. It determines how quickly the weight decreases as distance increases.
- The denominator $\sum_{k \neq i} \exp\left(-\frac{d_{ik}^2}{2\sigma_i^2}\right)$ ensures that the probabilities sum to 1.

t-distributed Stochastic neighbor embedding (t-SNE)

R^p $\qquad$ R^m, $m << p$

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} \qquad q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k\neq l}\left(1+\|y_k - y_l\|^2\right)^{-1}}$$

We try to minimize Kullback-Leibler divergence between *p* and *q*

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

t-SNE uses the Kullback-Leibler (KL) divergence

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

- **KL measures ‘distance’ from distribution P to Q**
- **Not a metric function - not symmetric!**
- **Measures how much more information is contained in P with respect to Q (similar to the mutual information for which Q is the product of marginal distributions)**

Notion of perplexity

- In information theory, perplexity is a measurement of how well a probability distribution or probability model predicts a sample
- It may be used to compare probability models.
- A low perplexity indicates the probability distribution is good at predicting the sample
- The perplexity PP of a discrete probability distribution *p* is defined as

$$PP(p) := 2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

Notion of perplexity

$$PP(p) := 2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

- H(p) is the Shannon entropy of the probability distribution p(x), which measures the uncertainty or spread of the distribution.
- p(x) is the probability assigned to a neighbor x.
- Perplexity in t-SNE controls the effective number of neighbors.
- A high perplexity leads to a higher entropy H(pi), meaning more neighbors are considered important.
- A low perplexity results in a lower entropy, meaning the focus is only on a few closest neighbors.
- Mathematically, t-SNE searches for the optimal bandwidth σi such that the entropy matches the desired perplexity.

Perplexity in Gaussian distribution

- t-SNE searches for the optimal **σi** such that the entropy matches the perplexity. The equation for entropy with Gaussian weights:

$$H(p_i) = -\sum_j \frac{\exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right)}{Z_i} \log_2 \frac{\exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right)}{Z_i}$$

Here, $Z_i = \sum_{k \neq i} \exp\left(-\frac{d_{ik}^2}{2\sigma_i^2}\right)$ is the normalization term.

- optimization ensures that the **neighborhood size** adapts dynamically based on the distances between points and the chosen perplexity.

Notion of perplexity

- **Small σi**:
  The Gaussian is narrow, meaning only very close neighbors will get high weights. This emphasizes **local structure**.
- **Large σi**:
  The Gaussian is wider, meaning more neighbors will receive significant weights, capturing more **global structure**.
- In **t-SNE**, the goal is to find the optimal σi such that the entropy of the distribution matches the specified **perplexity**. This process is done separately for each point to adapt to the **local density** of the data.

- For Gaussian distribution $PP(p) = \frac{1}{2}\ln(\sqrt{2\pi e\sigma})$

Discussion on t-SNE

- Some remarks:

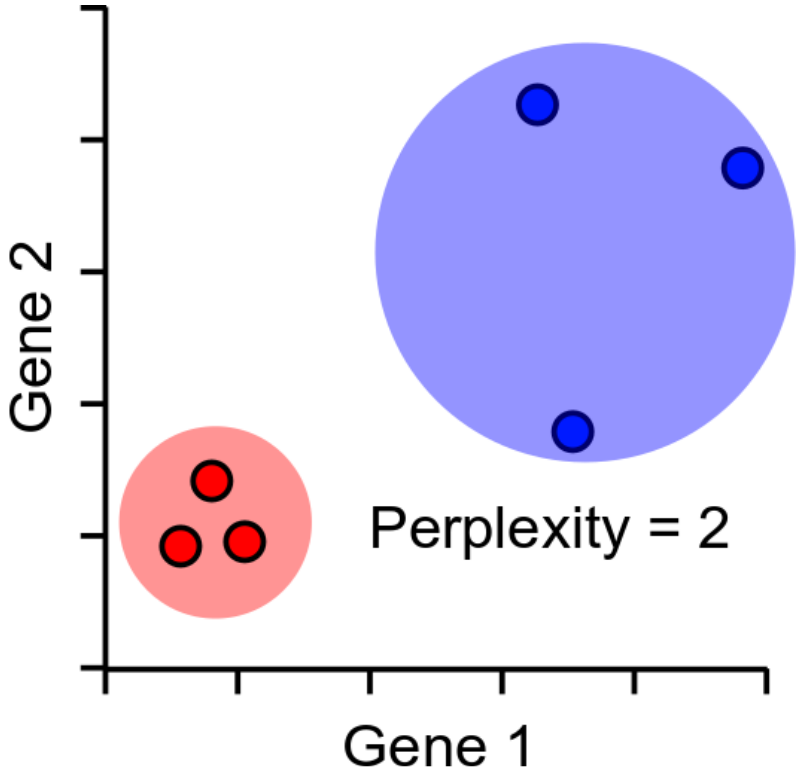
$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

- if $||x_i - x_j|| \gg \sigma_i$ then $p_{i|j} \approx 0$

- if $||x_i - x_j|| \ll \sigma_i$ (large perplexity) then

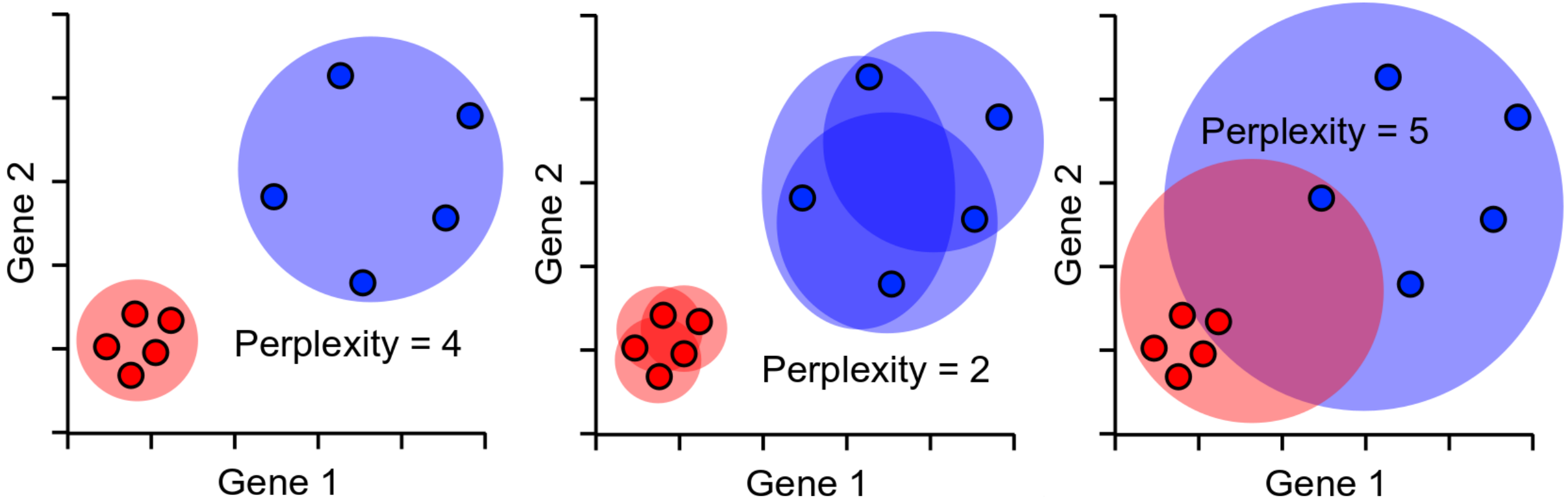
Distance scaling and perplexity

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



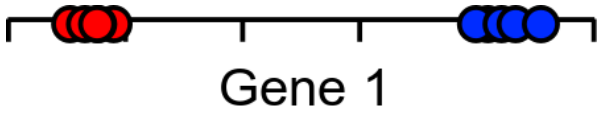
	0	4	6	586	657	836
	4	0	4	815	527	776
	9	3	0	752	656	732
	31	28	29	0	4	7
	31	24	25	4	0	7
	40	37	32	8	8	0

Perplexity Robustness

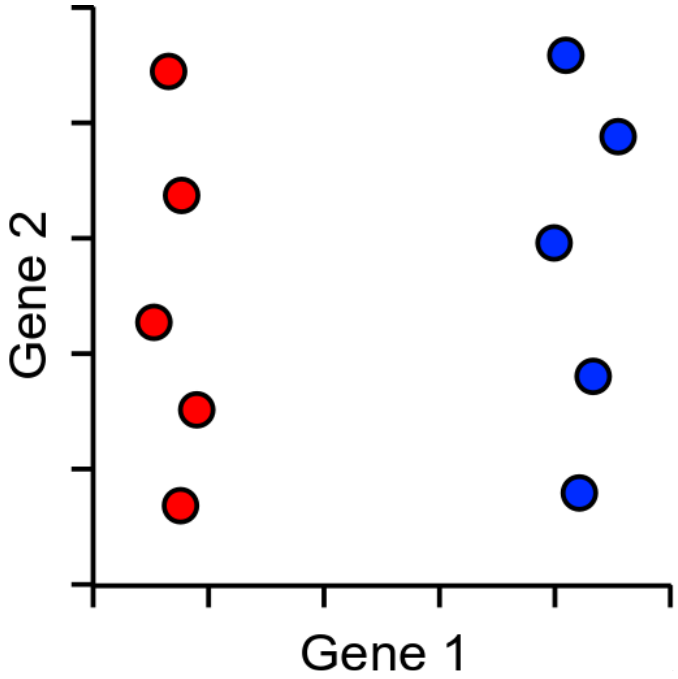


So tSNE is great then?

- Kind of…
- Imagine a dataset with only one super informative gene



Distance within cluster = low
Distance between clusters = high



Distance within cluster = higher
Distance between clusters = lower

- Now 3 genes
- Now 3,000 genes
- Everything is the same distance from everything

t-SNE hyper-parameters

- Barnes-Hut's tSNE implementation - O(n log n)
  - Rtsne & Seurat & viSNE (MATLAB)
    - Maaten (2014) *Journal of Machine Learning Research*

- The definition of the t-SNE and the chances of converging correctly depends on the hyper-parameters ("tuning" parameters).

- SNE has over 10 hyper-parameters that can be optimized for your specific data:

  - Perplexity
  - Number of iterations
  - Learning rate
  - …

t-SNE: summary

To keep in mind:

- It is a NON-LINEAR method of dimensionality reduction
- It is the current GOLD-STANDARD method in single cell data (including scRNA-seq)
- Can be run from the top PCs (e.g.: PC1 to PC10)

Problems:

- It does not learn an explicit function to map new points
- It's cost function is not convex – This means that the optimal t-SNE cannot be computed
- Too many hyper-parameters to be defined empirically (dataset-specific)
- It does not preserve a global data structure (only local)

Uniform Manifold Approximation and Projection (UMAP)

UMAP to the rescue!

- UMAP is a replacement for tSNE to fulfil the same role
- Conceptually very similar to tSNE, but with a couple of relevant (and somewhat technical) changes
- Practical outcome is:
  - UMAP is quite a bit quicker than tSNE
  - UMAP can preserve more global structure than tSNE*
  - UMAP can run on raw data without PCA preprocessing*
  - UMAP can allow new data to be added to an existing projection

Uniform Manifold Approximation and Projection (UMAP)

McInnes, L., & Healy, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

Compared to t-SNE, UMAP seems to be

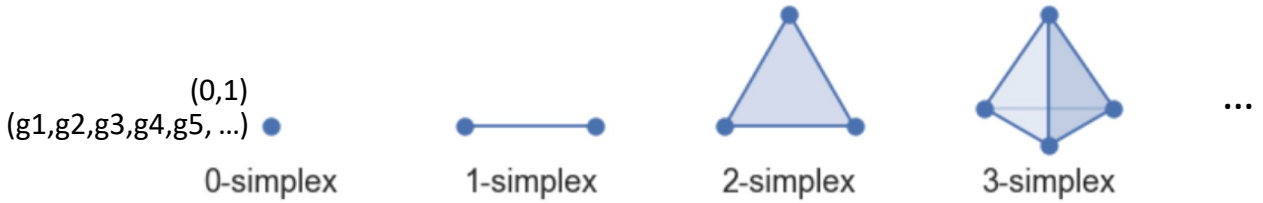
- faster
- deterministic
- better at preserving clusters

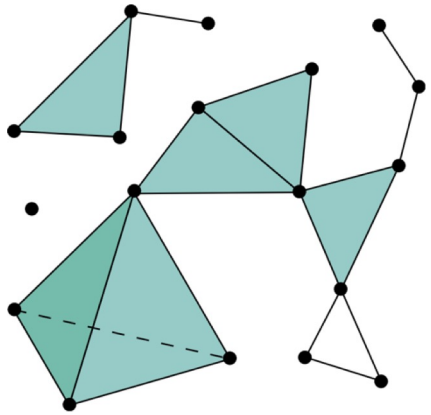
How UMAP works

It is based on **topological structures** in multidimensional space (simplices)

Points are connected with a line (edge) if the distance between them is below a threshold:

- Any distance metric can be used (euclidean)



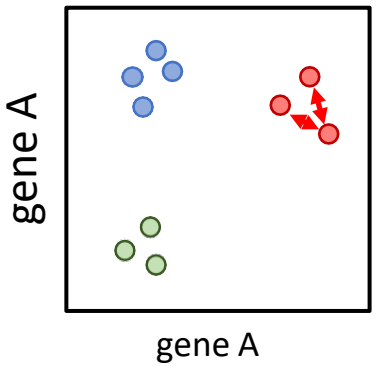


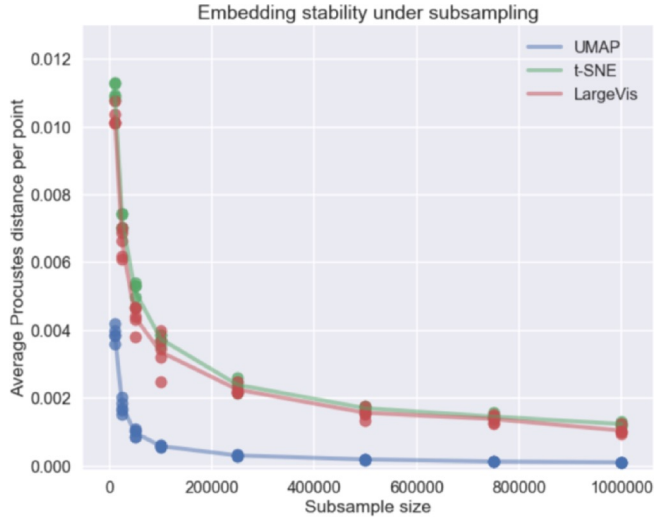
This way, by constructing the simplicial complexes beforehand allows UMAP to calculate the relative point distances in the lower dimension

(instead of randomly assigning as in tSNE)

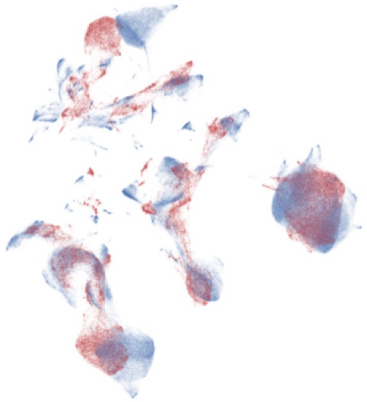
McInnes et al (2018) *BioRxiv*

UMAP

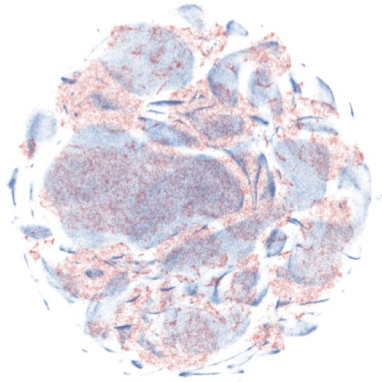




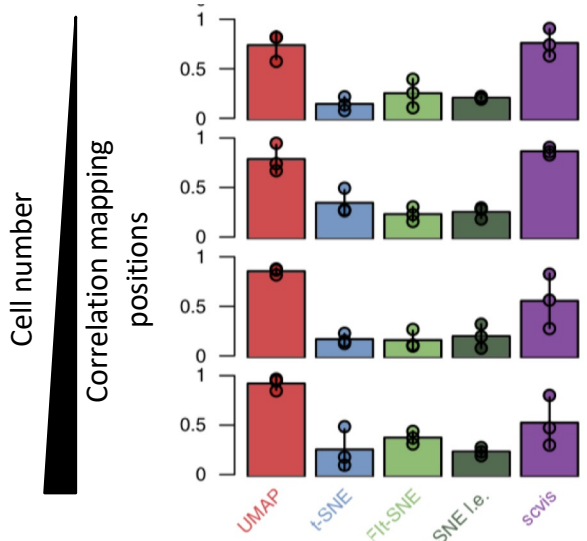
Since UMAP learns the global data structure and is less dependent on random initiators (like t-SNE), it can recreate low dimensional embedding regardless of the dataset size.



(a) UMAP



(b) t-SNE

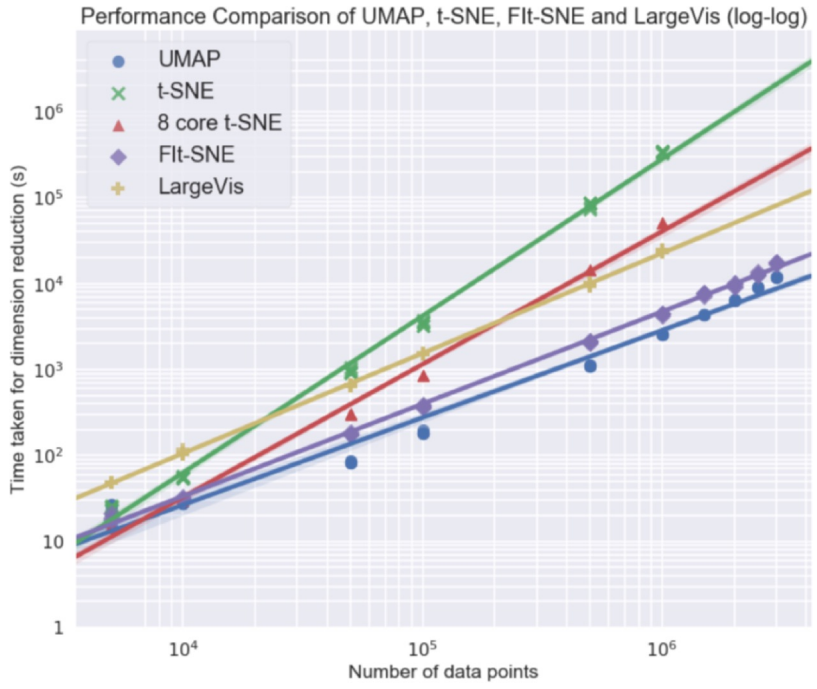


🔗 McInnes et al (2018) *BioRxiv*

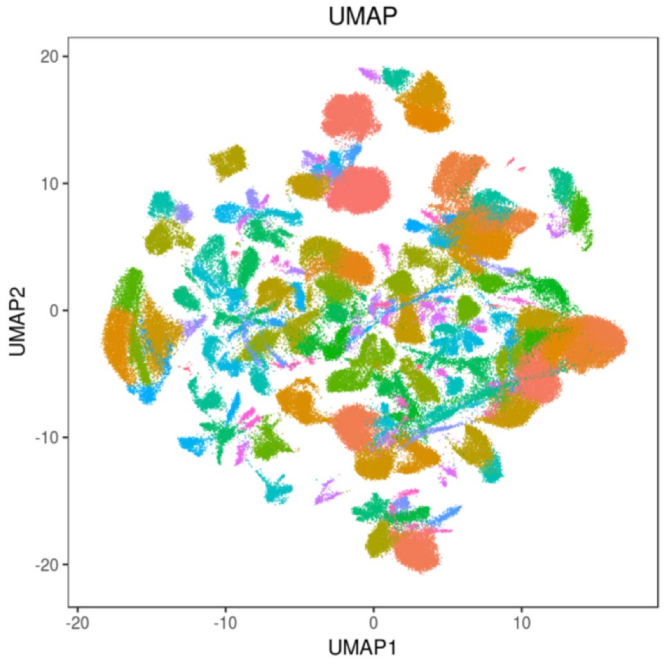
🔗 Becht & McInnes et al (2019) *Nat Biot*

UMAP hyper-parameters

UMAP's mathematical improvements allows much faster computations compared to current state-of-the-art methods.



250k cells
7 min



McInnes et al (2018) *BioRxiv*
Becht & McInnes et al (2019) *Nat Biot*

Constructing weighted neighbourhood graph

For each point *i,* define distance to the nearest neighbour

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

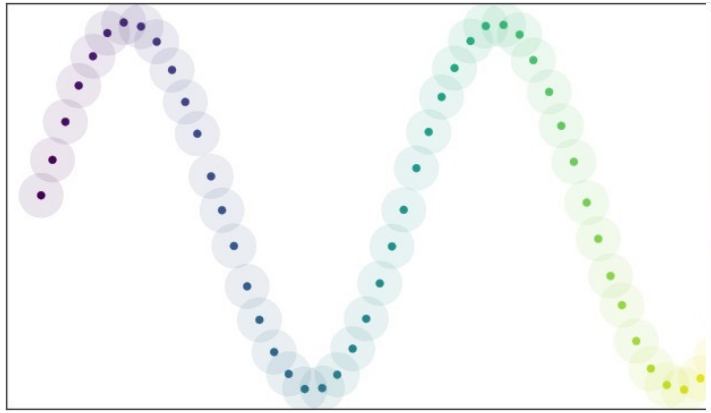
Define σ_i using the following equation (local dispersion)

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k)$$

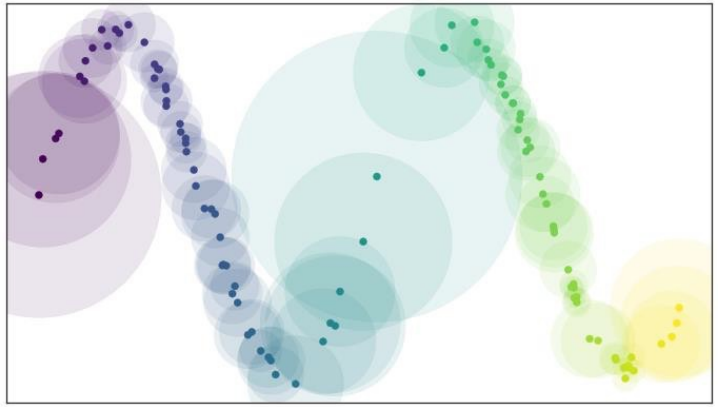
Weight of the neighbourhood graph

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

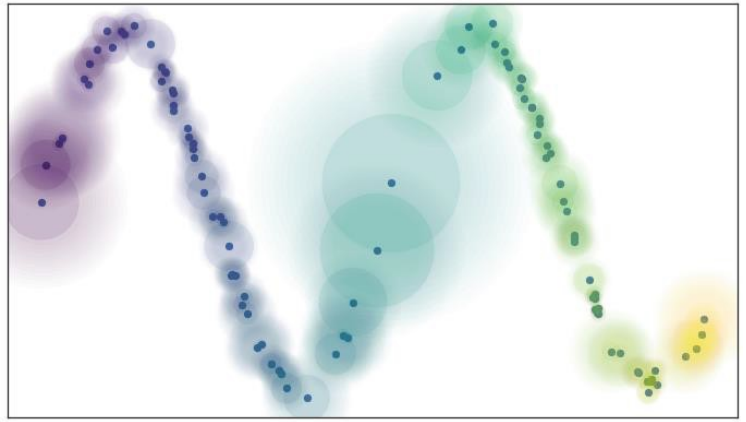
‘Uniform’ case



kNN graph



Fuzzy weighted neighbourhood graph (‘UMAP-modified kNN’)



k=4

UMAP differences

- Instead of the single perplexity value in tSNE, UMAP defines
  - **Nearest neighbours**: the number of expected nearest neighbours – basically the same concept as perplexity
  - **Minimum distance**: how tightly UMAP packs points which are close together
- Nearest neighbours will affect the influence given to global vs local information. Min dist will affect how compactly packed the local parts of the plot are.

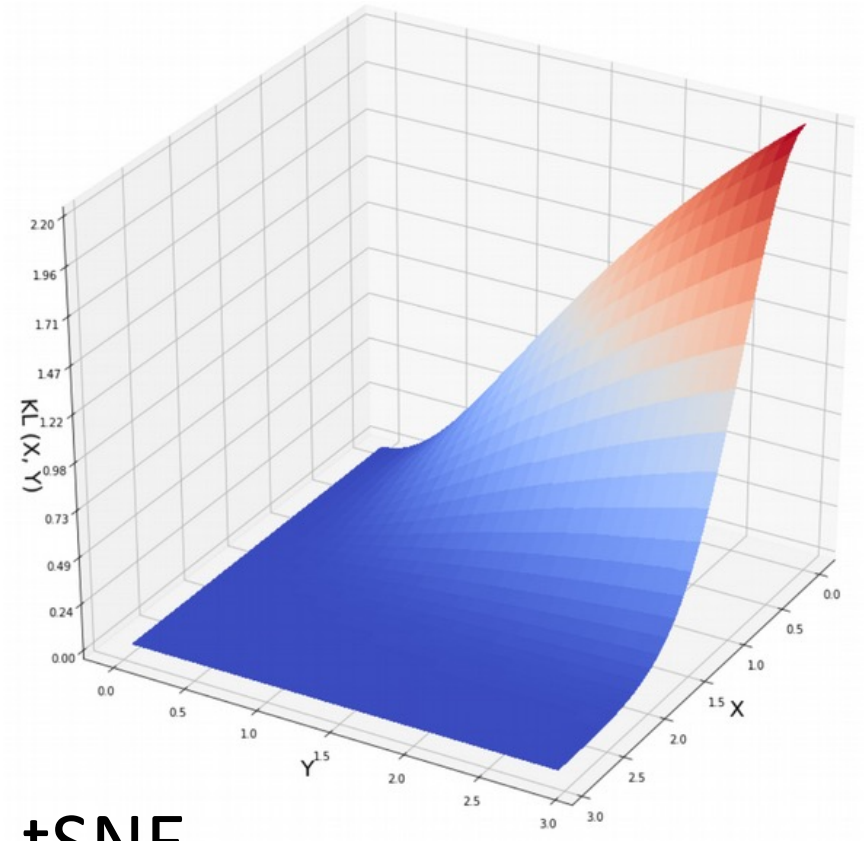
UMAP: Summary

- **To keep in mind:**
- It is a NON-LINEAR graph-based method of dimensionality reduction
- Very efficient -$O(n)$
- Can be run from the top PCs (e.g.: PC1 to PC10)
- Can use any distance metrics!
- Can integrate between different data types (text, numbers, classes)
- It is no longer completely stochastic as t-SNE
- Defines both LOCAL and GLOBAL distances
- Can be applied to new data points

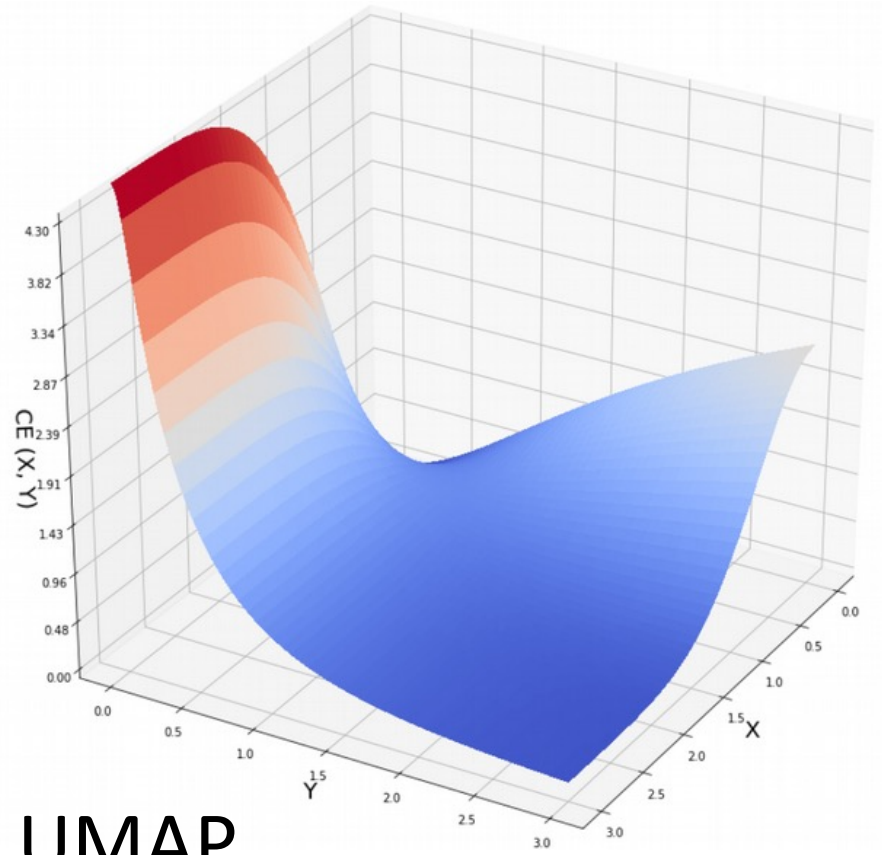
- McInnes et al (2018) *BioRxiv*
- Becht & McInnes et al (2019) *Nat Biot*

UMAP differences

- Structure preservation – mostly in the 2D projection scoring



tSNE

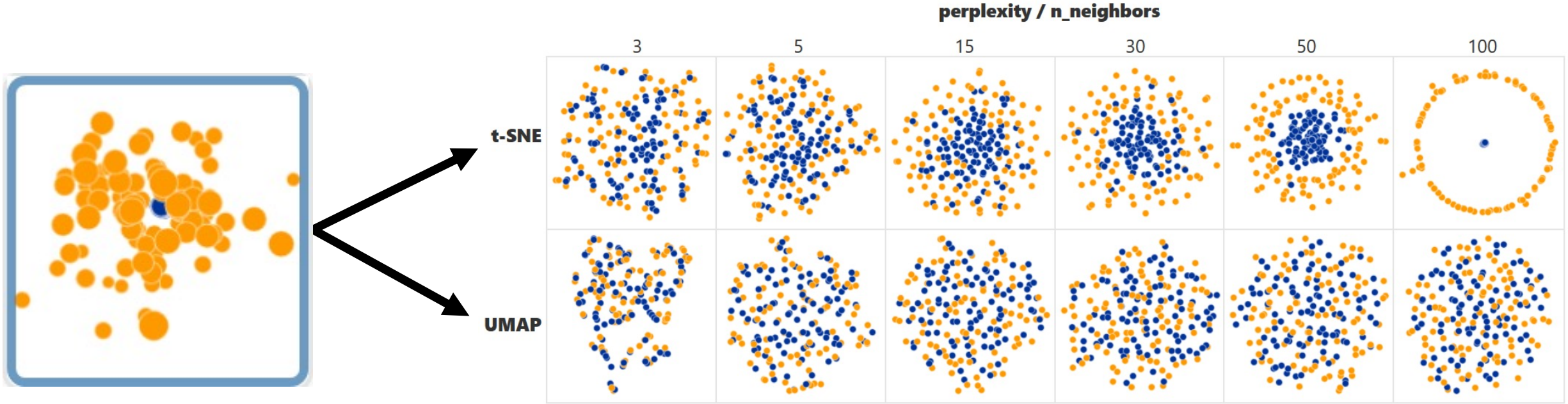


UMAP

https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

So UMAP is great then?

- Kind of…

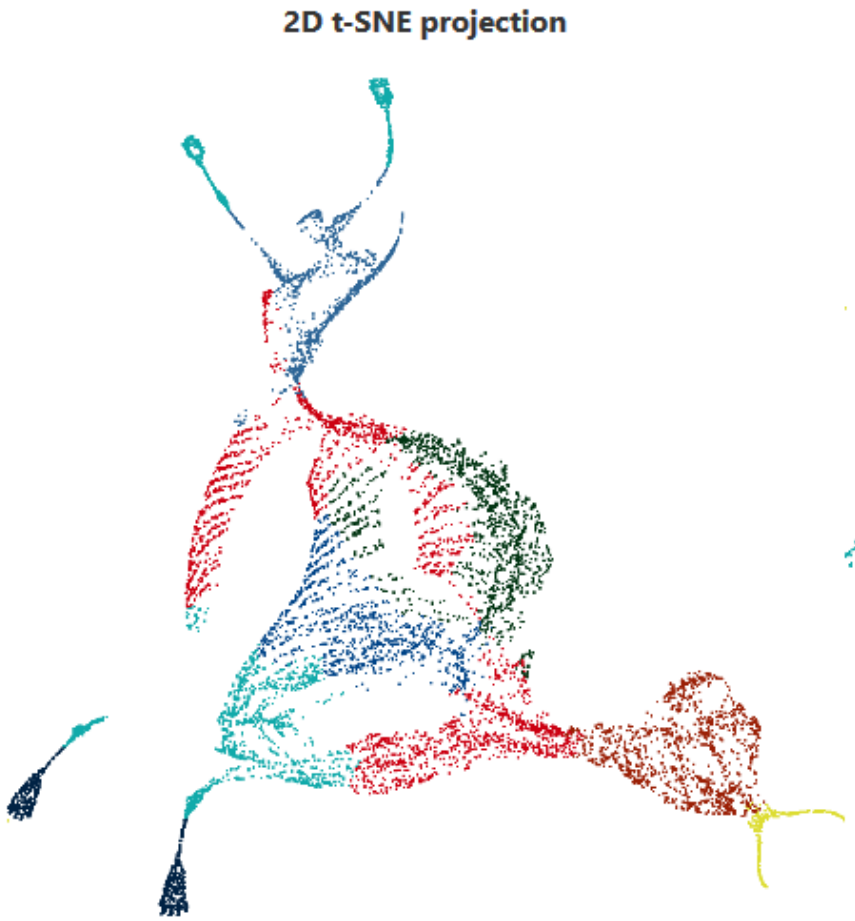


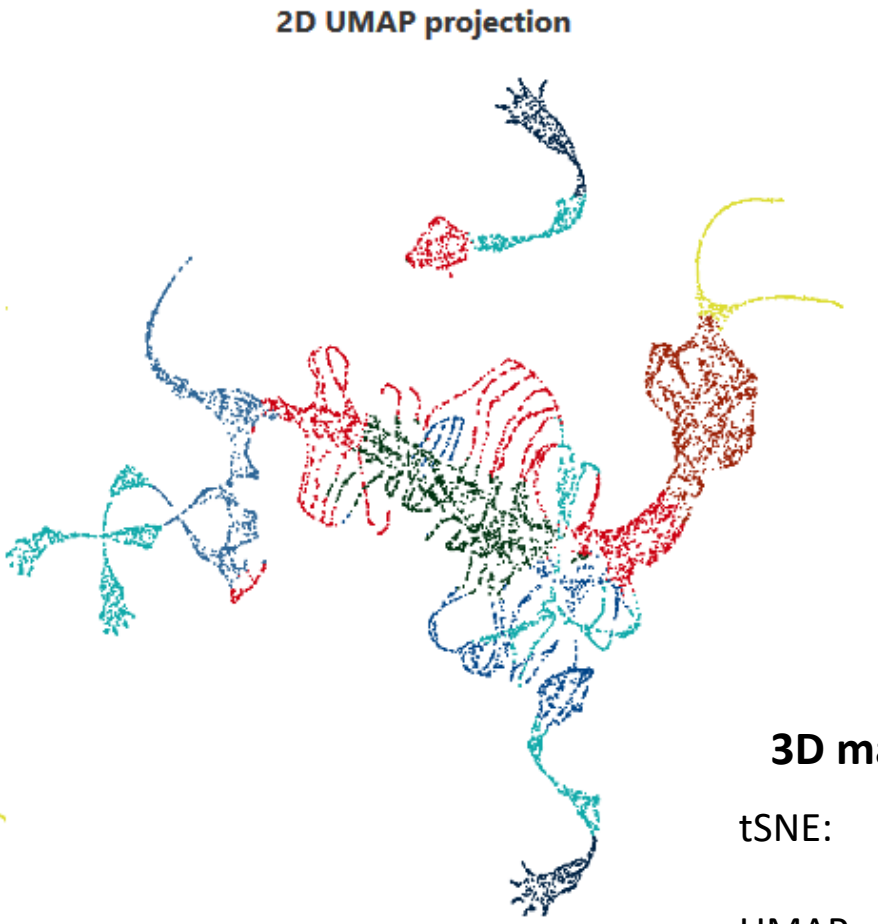
- It may perform better on more complex datasets
- It’s certainly quicker

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

So UMAP is all hype then?

- No, it really does better for some datasets…



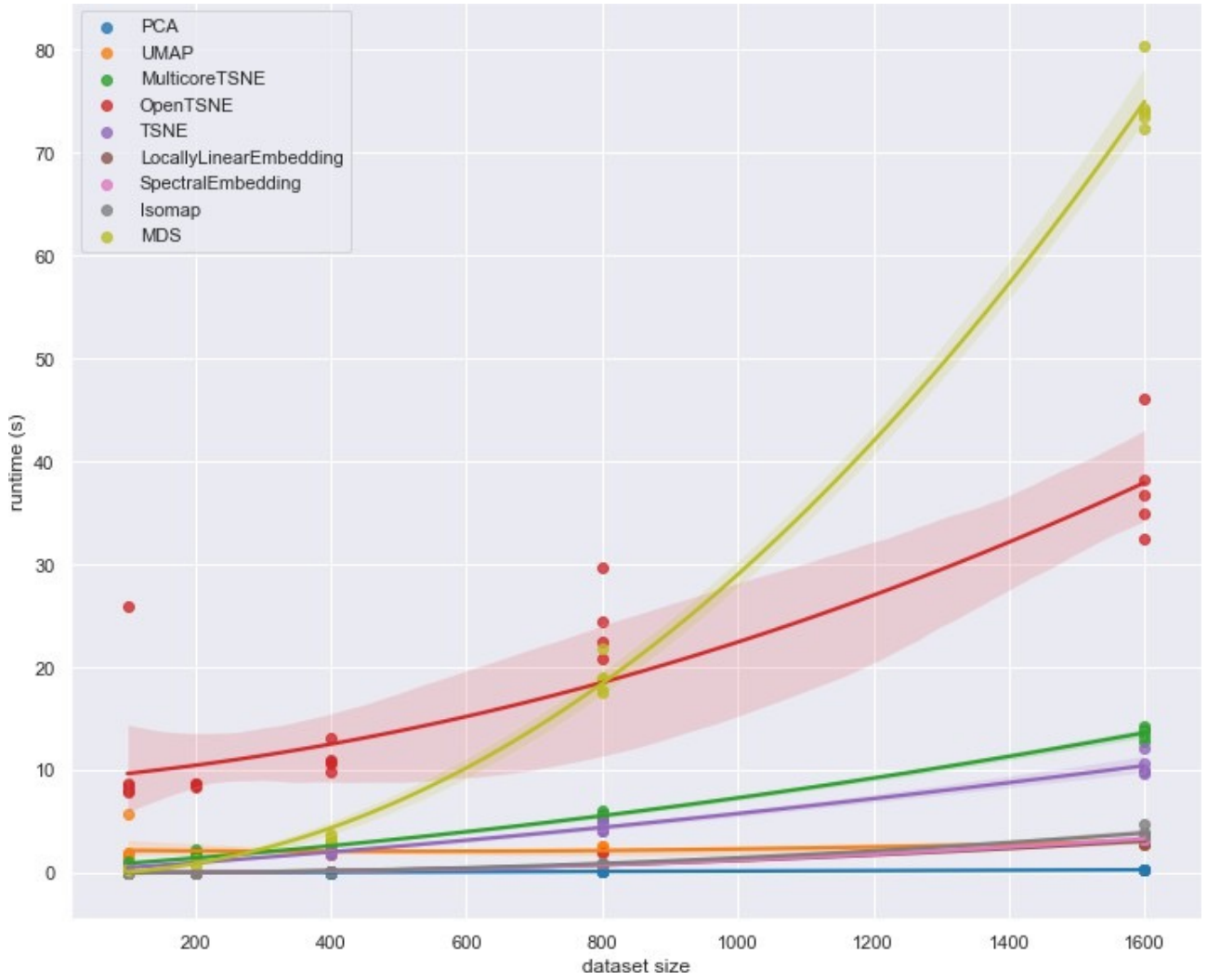


3D mammoth skeleton projected into 2D

tSNE:	Perplexity 2000	2h 5min
UMAP:	Nneigh 200, mindist 0.25,	3min

Comparing tSNE and UMAP

- Speed



Comparison of (many) methods:

https://colab.research.google.com/drive/1miQpnYAa9pZa-YWng1C7V78hM-nPR8Ly?usp=sharing

viz_results = apply_panel_of_manifold_learning_methods(X,color, methods_to_apply=['PCA','UMAP','TRIMAP','MDE','TSNE', 'LLE','MLLE','ISOMAP','MDS','SE', 'AUTOENCODER'])

MNIST dataset (downsampled to 2000 points)

PCA: 0.82 sec
LLE: 260 sec
Modified LLE: 270 sec
Isomap: 280 sec
MDS: 240 sec
SpectralEmbedding: 202 sec
t-SNE: 250 sec
UMAP: 44 sec
TRIMAP: 12 sec
MDE: 15 sec
Autoencoder: 170 sec