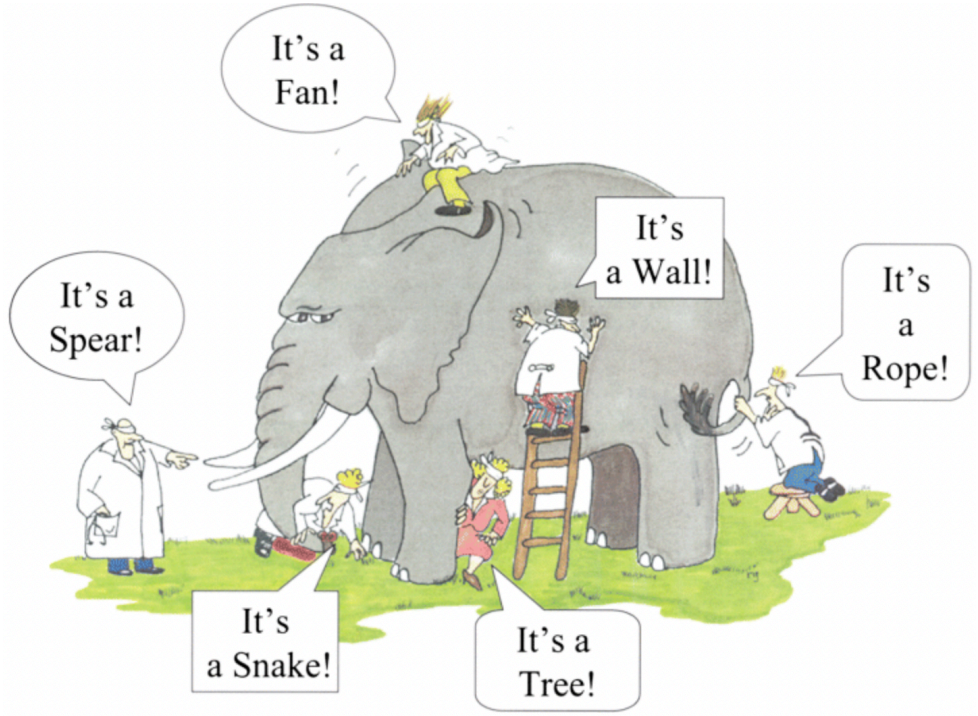
Dimensionality Reduction - PCA + t-SNE

We previously learned how to use the SVD as a tool for constructing low-rank matrices.

We now consider it as a tool for transforming (i.e., reducing the dimension of) our data.

Overview

Collected data is often high-dimensional. The high-dimensionality of, or the large number of features in a dataset is challenging to work with.



High-Dimensional Challenges

We have seen some of these challenges already, in particular:

How can we reduce the dimension of our data but still preserve the most important information in our dataset?

PCA

Dimensionality Reduction

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_m$ with $\mathbf{x}_i \in \mathbb{R}^n \ \ \forall\, i \in \{1, \ldots, n\}$.

Output: $\mathbf{y}_1, \ldots, \mathbf{y}_m$ with $\mathbf{y}_i \in \mathbb{R}^d \ \ \forall\, i \in \{1, \ldots, n\}$.

The goal is to compute the new data points $\mathbf{y}_i$ such that $d << n$ while still preserving the most information contained in the data points $\mathbf{x}_i$.

Be aware that row i of the data matrix X_0 is the n dimensional vector $\mathbf{x}_i$. This keeps our matrix in the structure m data rows and n features (columns). The same is true for Y (i.e., there are m rows with d features).

$$X_0 = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \xrightarrow[\text{PCA}]{} Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1d} \\ y_{21} & y_{22} & \cdots & y_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{md} \end{bmatrix}$$

PCA Application: Genes Mirror Geography

We consider a dataset from Novembre et al. (2008). The authors collected DNA data called SNPs (single nucleotide polymorphism) from 3000 individuals in Europe.

SNPs describe the changes in DNA from a common base pair (A,T,C,G). A value of 0 means no changes to the base pair, a value of 1 means 1 change to the base pair, and a value of 2 means both base pairs change.

The data for each individual consisted of approximately 500k SNPs. This means the data matrix we are working with is 3000 x 500k.

The authors performed PCA and plotted 1387 of the individuals in the reduced dimensions.

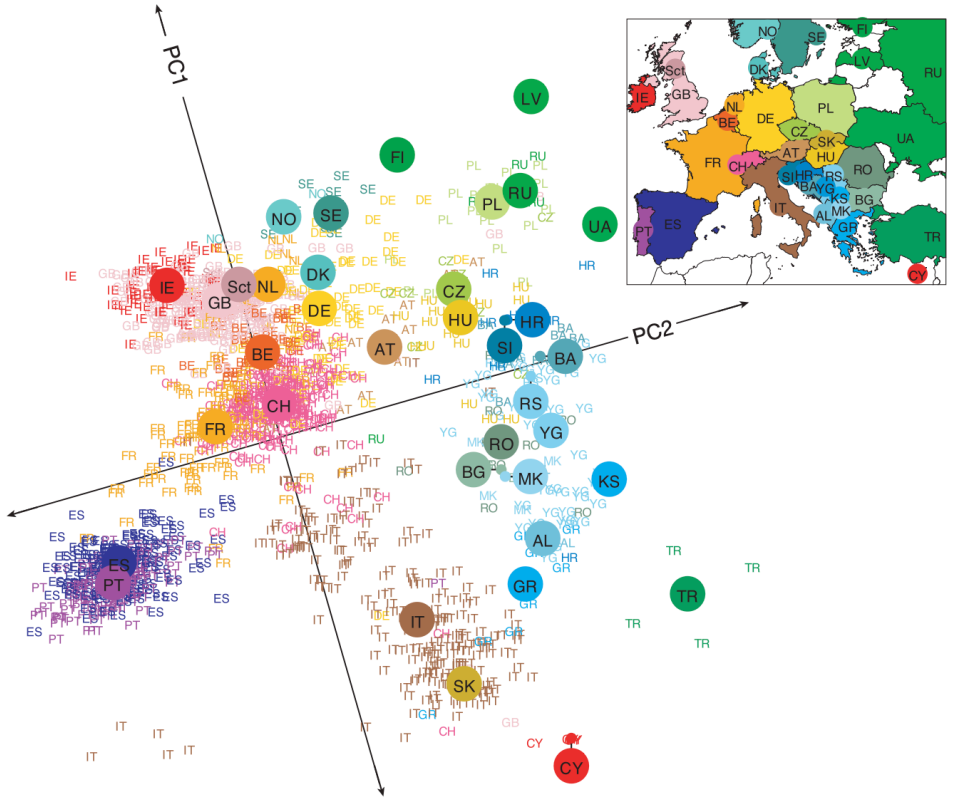


Image Credit Novembre et al. (2008)

For comparison, a color coded map of western Europe is added and the same color coding was applied to the data samples by country of origin.

Key observations:

PCA Overview

The following describes the process to perform PCA and obtain your reduced data set.

The goal is given:

Input: $X_0 \in \mathbb{R}^{m \times n}$, produce

Output: $Y \in \mathbb{R}^{m \times d}$ with $d << n$.

PCA Step 1: Center the Data

The first step is to center the data (subtract the mean of each column): $X_0 \rightarrow X$.

Example:

$$X_0 = \begin{bmatrix} 90 & 60 & 60 \\ 80 & 60 & 70 \end{bmatrix}$$

The mean, per column, across rows is:

$$\boldsymbol{\mu} = \begin{bmatrix} 85 & 60 & 65 \end{bmatrix}$$

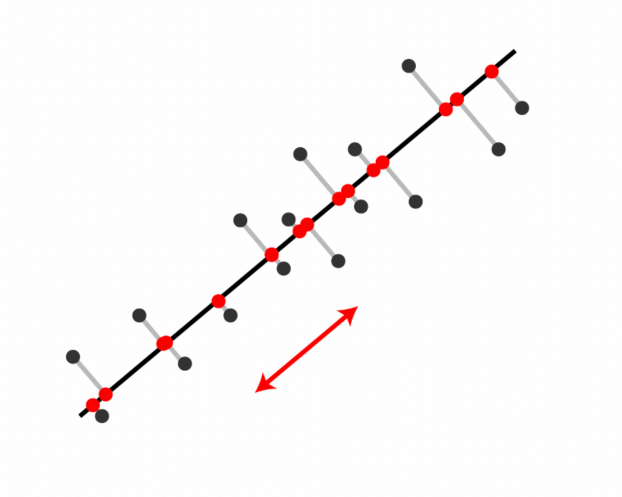
The mean-centered dataset is

$$X = \begin{bmatrix} 5 & 0 & -5 \\ -5 & 0 & 5 \end{bmatrix}$$

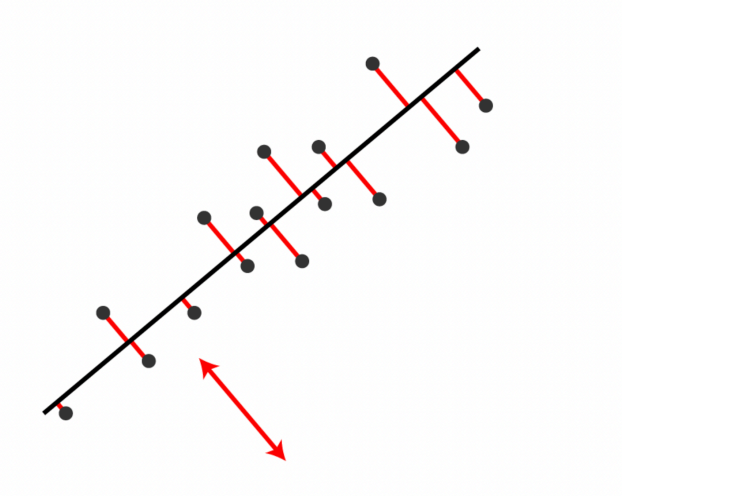
The next step is to determine the directions of the data that correspond to the largest variances. These are the principal components.

Least Squares Interpretation

Centered data often clusters along a line (or other low-dimensional subspace of $\mathbb{R}^n$).



The sum of **variances** (squared distances to the mean) of the projected points is a **maximum**.



The sum of **residuals** (squared distances from the points to the line) is a **minimum**.

Covariance Matrix

Let $X \in \mathbb{R}^{m \times n}$ contain the centered data. The sample covariance matrix for zero-mean data is:

$$S = \frac{1}{m-1} X^T X$$

(See notes for derivation.)

Example:

$$X = \begin{bmatrix} 5 & 0 & -5 \\ -5 & 0 & 5 \end{bmatrix}$$

$$S = \begin{bmatrix} 5 & -5 \\ 0 & 0 \\ -5 & 5 \end{bmatrix} \begin{bmatrix} 5 & 0 & -5 \\ -5 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 50 & 0 & -50 \\ 0 & 0 & 0 \\ -50 & 0 & 50 \end{bmatrix}$$

The matrix S is symmetric i.e. $S = S^T$

Spectral Decomposition

We use the fact that the covariance matrix S is symmetric to apply the Spectral Decomposition, which states:

> Every real symmetric matrix S has the factorization $V \Lambda V^T$, where Λ is a diagonal matrix that contains the eigenvalues of S and the columns of V are orthogonal eigenvectors of S.

This is a special case of eigendecomposition for symmetric matrices.

You can refresh your memory about this in the Linear Algebra Refresher.

The covariance matrix $S \in \mathbb{R}^{n \times n}$ has the spectral decomposition $S = V \Lambda V^T$ where

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ & 0 & & \lambda_{n-1} \\ & & & & \lambda_n \end{bmatrix}$$

with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} \geq \lambda_n$, and

$$V = \begin{bmatrix} | & | & & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_{n-1} & \mathbf{v}_n \\ | & | & & | & | \end{bmatrix}$$

with $S\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and $\mathbf{v}_i \perp \mathbf{v}_j$ for $i \neq j$.

The previous decomposition was for all n dimensions. To obtain our reduced data, we take the first d columns of V, i.e.,

$$V' = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_d \\ | & | & & | \end{bmatrix},$$

and the $d \times d$ upper block of matrix Λ

$$\Lambda' = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_d \end{bmatrix}.$$

PCA Interpretation

- The direction $\mathbf{v}_i$ is the i-th principal component and the corresponding λ_i accounts for the i-th largest variance in the dataset.
  - In other words, $\mathbf{v}_1$ is the first principal component and is the direction that accounts for the most variance in the dataset.
  - The vector $\mathbf{v}_d$ is the d-th principal component and is the direction that accounts for the d-th most variance in the dataset.

The reduced data matrix is obtained by computing

$$Y = XV'.$$

Spatial Interpretation – Dimensions

The operation $Y = XV'$ performs a **change of basis** and **projection** of your data:

- $X \in \mathbb{R}^{m \times n}$: each of the m rows is a data point in the original n-dimensional feature space
- $V' \in \mathbb{R}^{n \times d}$: contains the first d principal components as columns (these are orthogonal vectors)
- $Y \in \mathbb{R}^{m \times d}$: each row is now a data point in the new d-dimensional space

Geometric Interpretation

1. **Rotation**: The multiplication by V' rotates your coordinate system. The columns of V' define a new set of d orthogonal axes that point in the directions of maximum variance.
2. **Projection**: Each data point in X is projected onto this new coordinate system. For each row vector $\mathbf{x}_i$ (a point in $\mathbb{R}^n$):

$$\mathbf{y}_i = \mathbf{x}_i V' = \begin{bmatrix} \mathbf{x}_i \cdot \mathbf{v}_1 & \mathbf{x}_i \cdot \mathbf{v}_2 & \cdots & \mathbf{x}_i \cdot \mathbf{v}_d \end{bmatrix}$$

3. **Dimensionality Reduction**: You're effectively taking each point in n-dimensional space and expressing it using only d coordinates - the coordinates along the principal component directions. You're discarding the components along directions with low variance.

Intuitive View

Think of it as finding the "best viewing angle" for your data. If you have data in 3D that's mostly flat (like a pancake), PCA finds that the data lies approximately in a 2D plane. V' defines that plane's orientation, and Y gives you the coordinates of each point within that

Returning to our example

$$X = \begin{bmatrix} 5 & 0 & -5 \\ -5 & 0 & 5 \end{bmatrix}, \quad S = \begin{bmatrix} 50 & 0 & -50 \\ 0 & 0 & 0 \\ -50 & 0 & 50 \end{bmatrix}$$

Eigenvalues of S: $\lambda_1 = 100,\ \lambda_2 = \lambda_3 = 0$

Eigenvectors of S:

$$\mathbf{v}_1 = \begin{bmatrix} 0.7071 \\ 0 \\ -0.7071 \end{bmatrix},\ \mathbf{v}_2 = \begin{bmatrix} 0.7071 \\ 0 \\ 0.7071 \end{bmatrix},\ \mathbf{v}_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

What is the value of the total variance in the data?

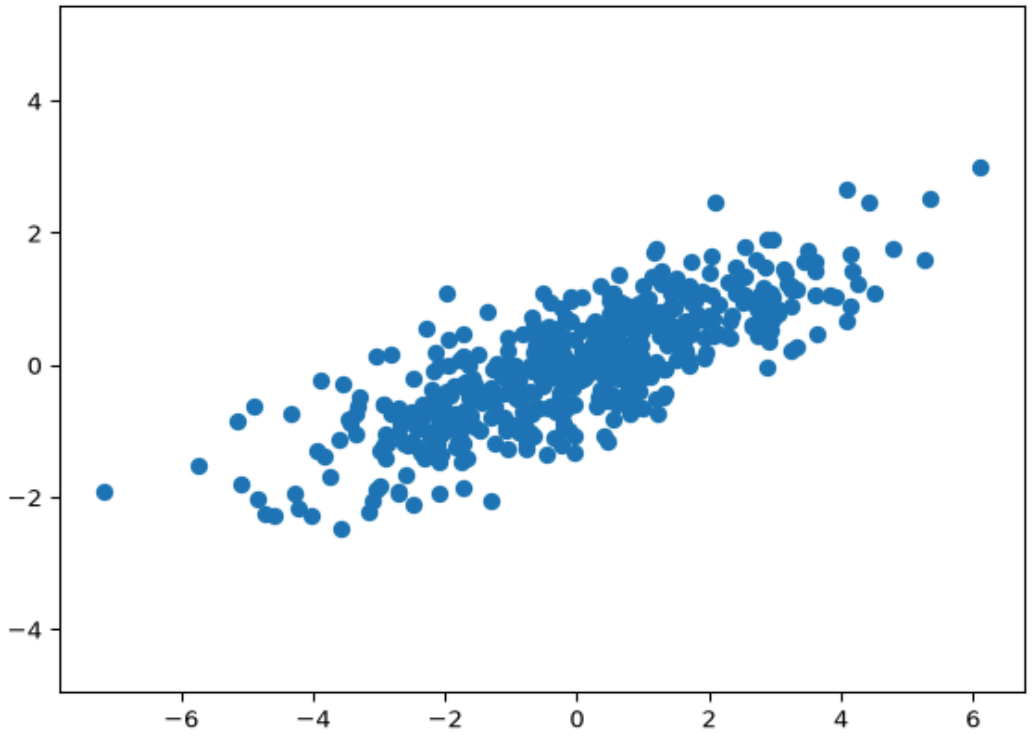
PCA Summary

- The columns of V are the principal directions (or components).
- The reduced dimension data is the projection of the data in the direction of the principal directions, i.e., $Y = XV'$.
- The total variance T in the data is the sum of all eigenvalues: $T = \lambda_1 + \lambda_2 + \cdots + \lambda_n$.
- The first eigenvector $\mathbf{v}_1$ points in the most significant direction of the data. This direction explains the largest fraction λ_1/T of the total variance.
- The second eigenvector $\mathbf{v}_2$ accounts for a smaller fraction λ_2/T.
- The **explained variance** of component i is the value λ_i/T. As $i \to n$ the explained variance gets smaller and approaches λ_n/T.
- The **cumulative (total) explained variance** is the sum of the explained variances up to component k, e.g. $\sum_{i=1}^{k} \frac{\lambda_i}{T}$. The total explained variance for all eigenvalues is 1.

Case Study: Random Data

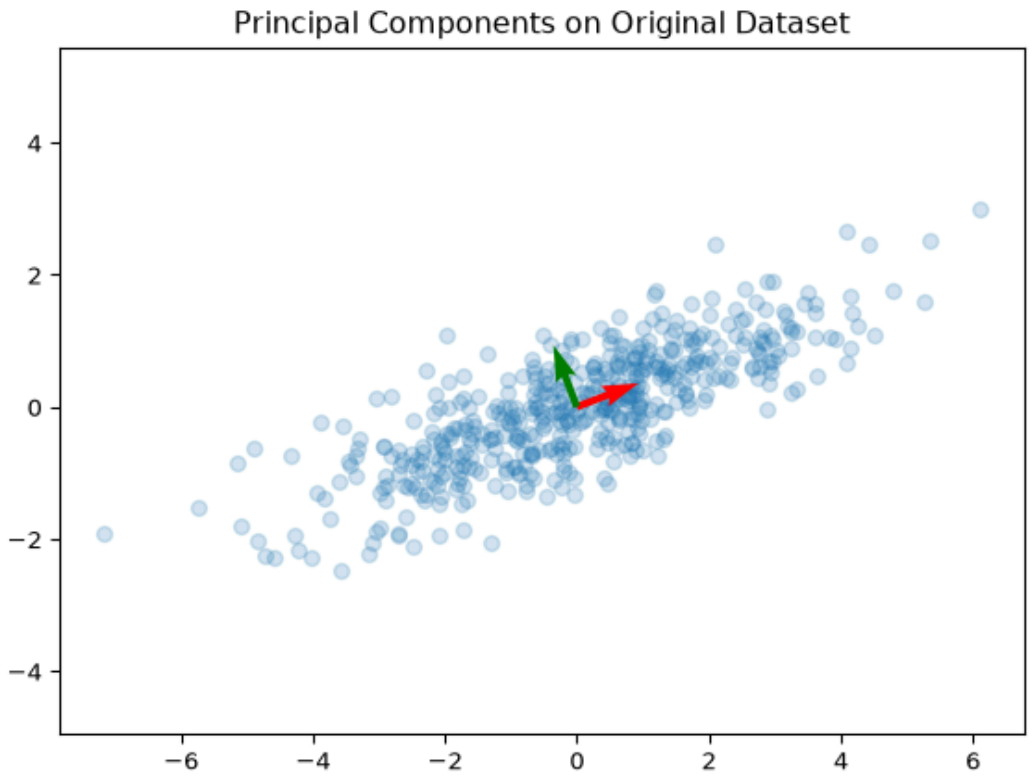
Let’s consider some randomly generated data consisting of 2 features.

▶ Code



Let’s do PCA and plot the principle components over our dataset. As expected, the principal components are orthogonal and point in the directions of the maximum variance.

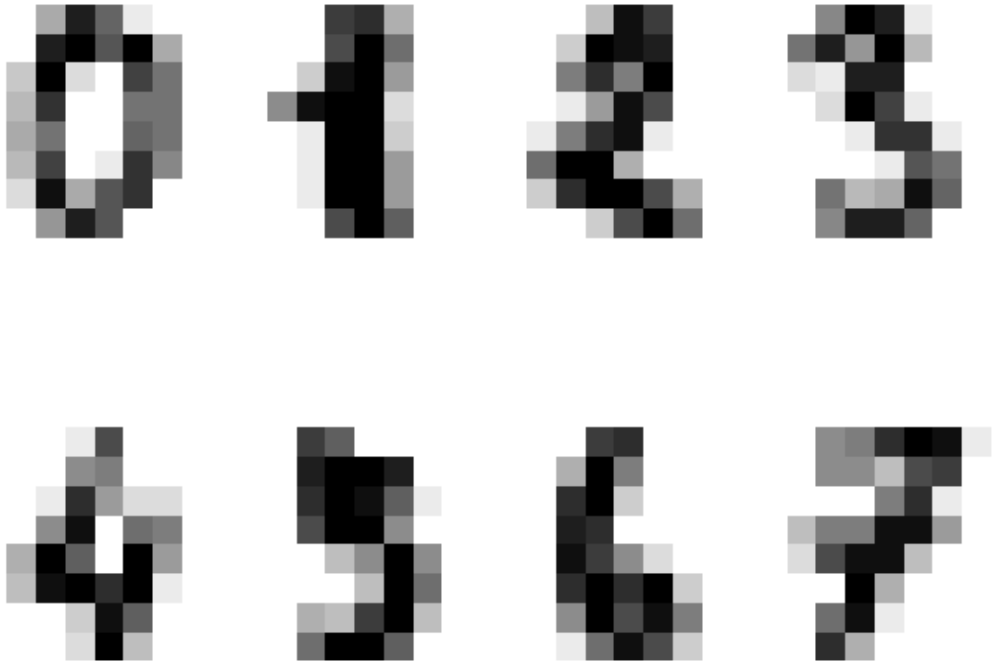
▶ Code



Case Study: Digits

Let’s consider another example using the MNIST dataset.

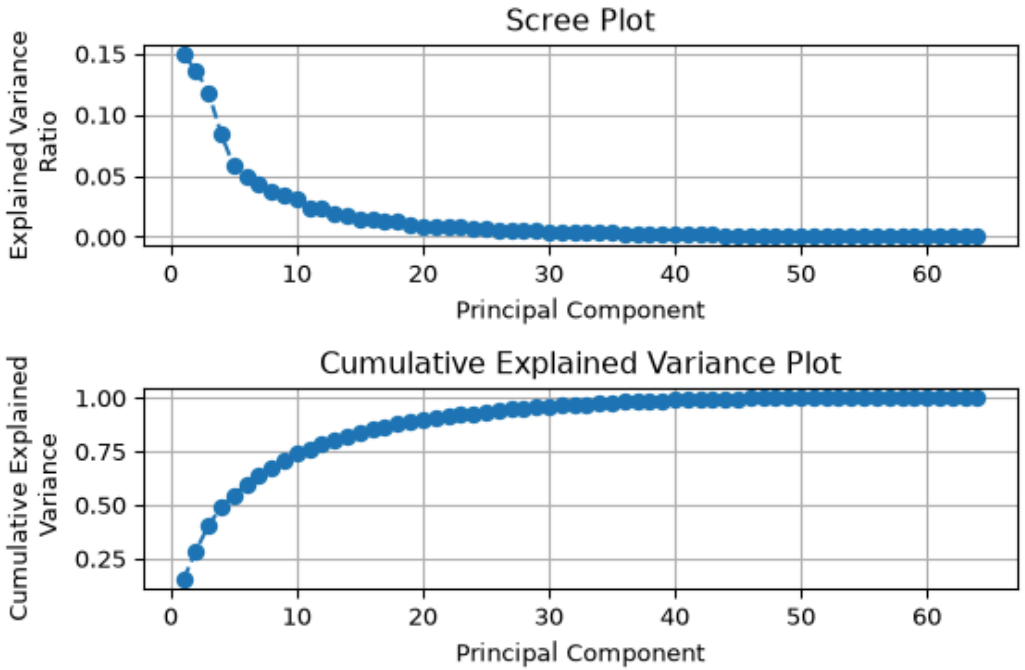
▶ Code



We first stack the columns of each digit on top of each other (this operation is called vectorizing) and perform PCA on the 64-D representation of the digits.

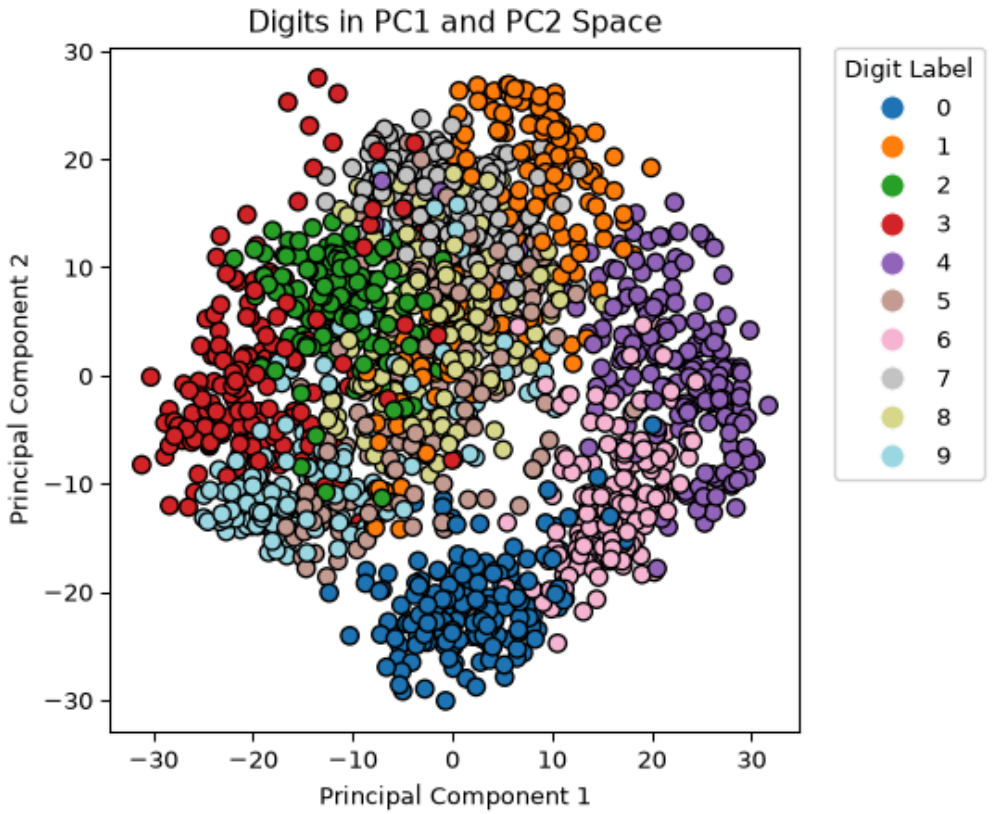
We can plot the explained variance ratio and the total (cumulative) explained variance ratio.

▶ Code



Let’s plot the data in the first 2 principal component directions. We’ll use the digit labels to color each digit in the reduced space.

▶ Code



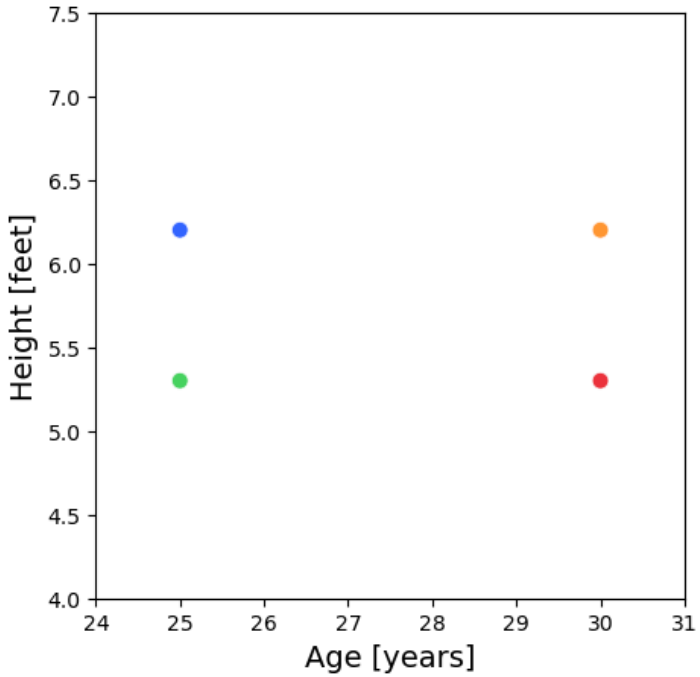
We observe the following in our plot of the digits in the first two principal components:

- There is a decent clustering of some of our digits, in particular 0, 2, 3, 4, and 6.
- The numbers 0 and 6 seem to be relatively close to each other in this space.
- There is not a very clear separation of the number 5 from some of the other points.

To scale or not to scale

Consider a situation where we have age (years) and height (feet) data for 4 people.

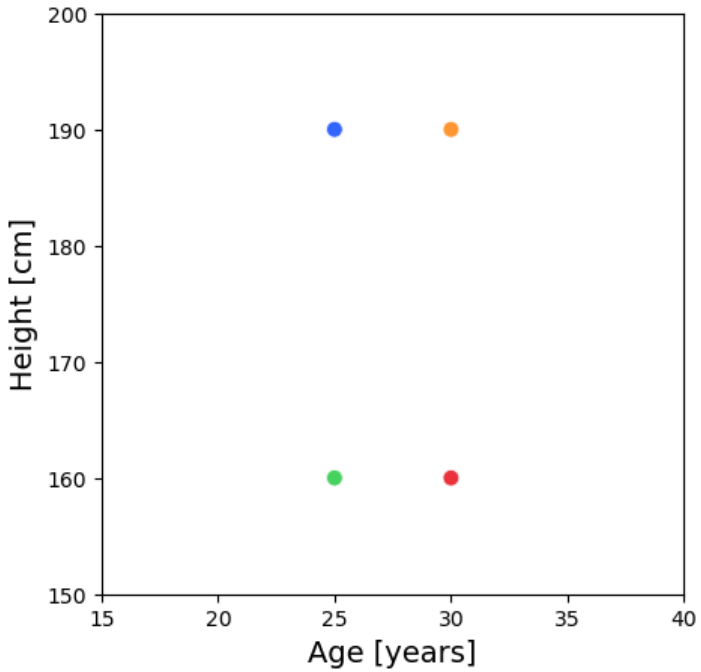
Person	Age [years]	Height [feet]
A	25	6.232
B	30	6.232
C	25	5.248
D	30	5.248



Notice that the dominant direction of the variance is aligned horizontally.

What if the height is in cm?

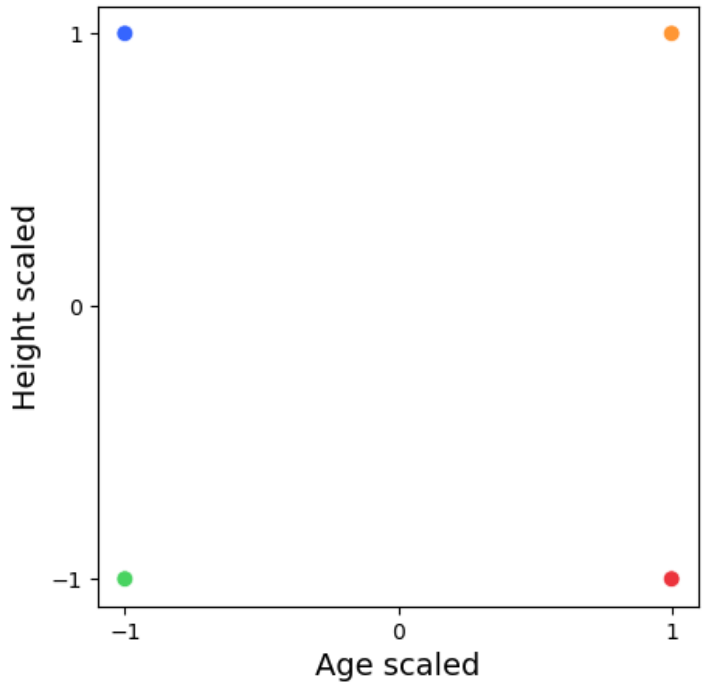
Person	Age [years]	Height [cm]
A	25	189.95
B	30	189.95
C	25	159.96
D	30	159.96



Notice that the dominant direction of the variance is aligned vertically.

Let’s standardize our data.

Person	Age [years]	Height [cm]
A	-1	1
B	1	1
C	-1	-1
D	1	-1



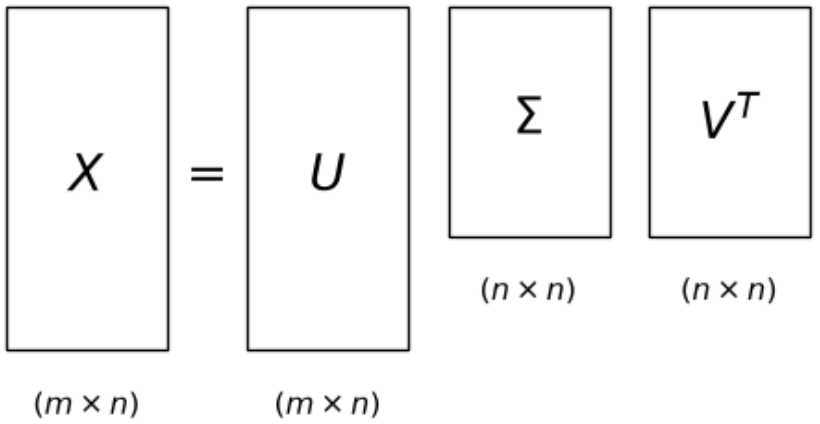
When we normalize our data, we observe an equal distribution of the variance.

What quantity is represented by $\frac{Cov(X,Y)}{\sigma_X \sigma_Y}$, where X and Y represent the random variables of a person's age and height, respectively?

Relationship to the SVD

Recall that the SVD of a (mean-centered) data matrix $X \in \mathbb{R}^{m \times n}$ $(m > n)$ is

$$X = U\Sigma V^T.$$



How can we relate this to what we learned in PCA?

In PCA, we computed an eigen-decomposition of the covariance matrix, i.e., $S = V\Lambda V^T$.

Consider the following computation using the SVD of X

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$$

We see that eigenvalues λ_i of the matrix S are the squares σ_i^2 of the singular values Σ (scaled by $\frac{1}{m-1}$).

In practice, PCA is done by computing the SVD of your data matrix as opposed to forming the covariance matrix and computing the eigenvalues.

The principal components are the right singular vectors V. The projected data is computed as XV. The eigenvalues of S are obtained from squaring the entries of Σ and scaling them by $\frac{1}{m-1}$.

Reasons to compute PCA this way are

Pros and Cons

Here are some advantages and disadvantages of using PCA.

Advantages

- Allows for visualizations
- Removes redundant variables
- Prevents overfitting
- Speeds up other ML algorithms

Disadvantages

- reduces interpretability
- can result in information loss
- can be less effective than non-linear methods

t-SNE

What is t-SNE?

t-SNE stands for **t-distributed Stochastic Neighbor Embedding**

- A dimensionality reduction technique specifically designed for visualization
- Transforms high-dimensional data (hundreds or thousands of features) into 2D or 3D
- Goal: Preserve the local structure of your data while making it viewable
- Developed by Laurens van der Maaten and Geoffrey Hinton in 2008, see Van der Maaten and Hinton (2008).

Think of it as creating a “map” of your data where similar points stay close together.

Why Do We Need t-SNE?

The Curse of Dimensionality

- Real datasets often have many features (dimensions)
- Impossible to visualize data with 100+ dimensions
- Traditional methods (like PCA) can lose important patterns

What t-SNE offers:

- Reveals clusters and patterns in complex data
- Preserves neighborhood relationships between points
- Creates intuitive, interpretable visualizations

How Does t-SNE Work?

The Big Picture

Step 1: Measure Similarity in High Dimensions

- Calculate how “similar” each pair of points is in the original space
- Similar points get high probability, distant points get low probability

Step 2: Create a Random Low-Dimensional Map

- Start with random positions for all points in 2D/3D

Step 3: Optimize the Map

- Iteratively move points around to match the high-dimensional similarities
- Points that were neighbors in high-D should be neighbors in low-D

Step 1: Compute Pairwise Similarities

- Calculate pairwise similarities between points in the high-dimensional space.
- Use a Gaussian distribution to convert distances into probabilities.
- The probability $p_{j|i}$ between points i and j is given by:

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

- The $p_{j|i}$ value represents the conditional probability that point x_i would choose x_j as its neighbor.
- Note that we set $p_{i|i} = 0$ – we don't consider a point to be its own neighbor.
- To create symmetric joint probabilities, t-SNE combines these two conditional probabilities into a single joint probability $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2d}$ where d is the dimension of the point. This is because the probabilities $p_{j|i}$ and $p_{i|j}$ are different. Using this value of p_{ij} is called symmetric t-SNE.

Step 1: Choosing σ

- The variance is calculated within a neighborhood of a point x_i determined by a hyperparameter called the **perplexity**.
- As a result the perplexity controls the effective number of neighbors.
- A high perplexity means more neighbors for each point. A low perplexity means less neighbors are considered for each point.

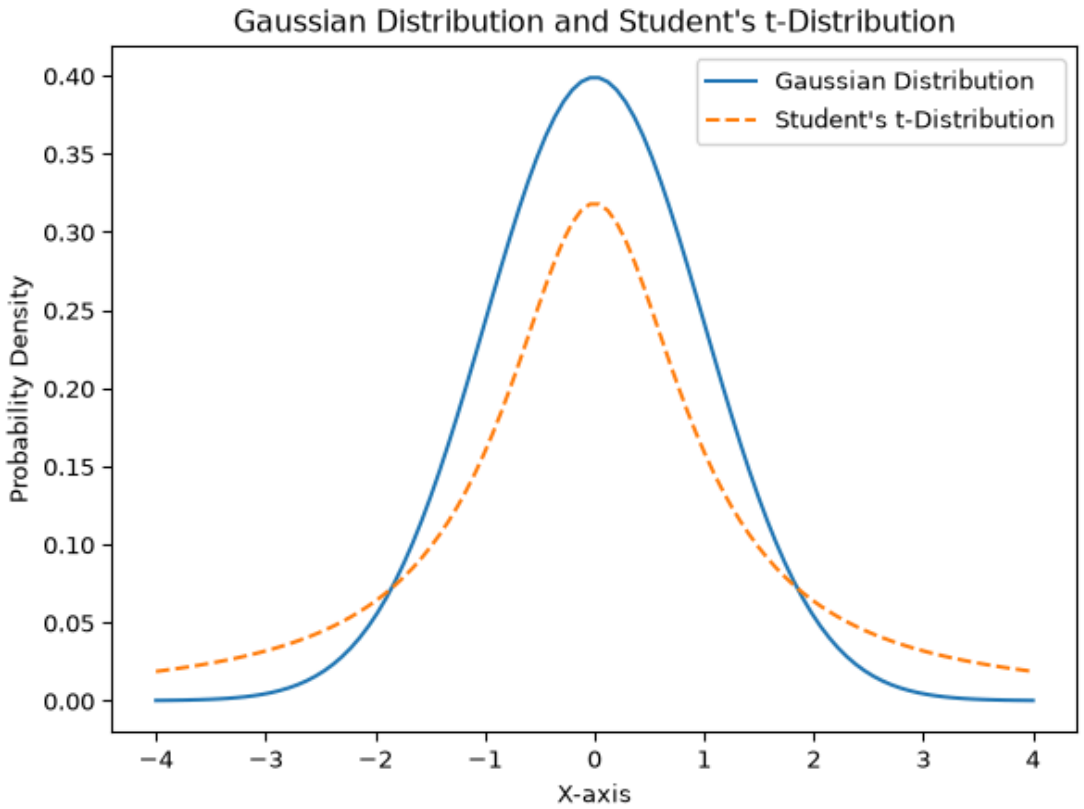
> See notes for definition of perplexity.

The perplexity commonly ranges between 5 and 50.

Visualizing the Pairwise Similarities

The Gaussian distribution is centered at point x_i and the points x_j that are further away have less probability of being chosen as neighbor.

▶ Code



Step 2: Define Low-Dimensional Map

- Initialize points randomly in the low-dimensional space.
- Define a similar probability distribution using a simplified Student-t distribution:

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

Step 3: Minimize Kullback-Leibler Divergence

- Minimize the Kullback-Leibler (KL) divergence between the high-dimensional and low-dimensional distributions:

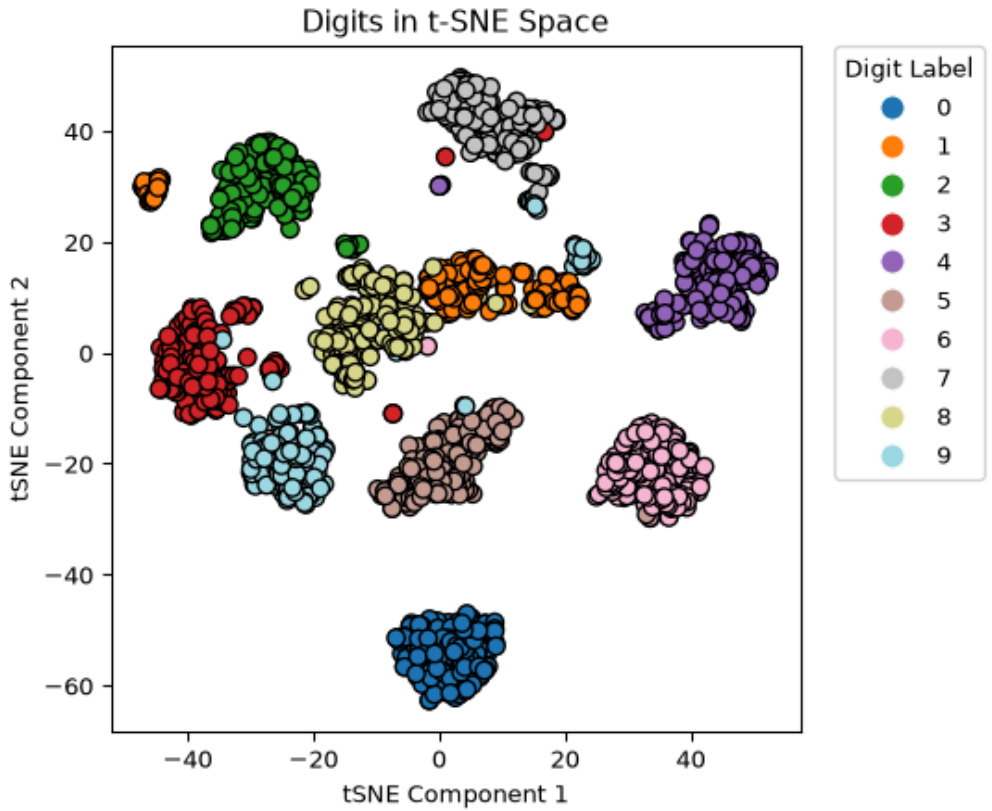
$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

- The KL divergence measures how different distribution P is from distribution Q. In other words, how different are each of the higher dimensional P_i point distributions from the lower dimensional Q_i distributions.
- By minimizing this function, we ensure that the lower dimensional point clusters are similar to the higher dimensional clusters.
- To minimize the KL divergence we use gradient descent on $\partial KL / \partial y_i$ to iteratively adjust the positions of points, y_i, in the low-dimensional space.

Case Study: Digits

Let’s consider how t-SNE performs clustering the MNIST digits datset.

▶ Code



The Key Insight

t-SNE uses different probability distributions for each step:

- **High dimensions**: Gaussian (normal) distribution
  - Measures similarity based on Euclidean distance
- **Low dimensions**: Student's t-distribution
  - Has "heavier tails" than Gaussian
  - Allows moderate distances in low-D to represent larger distances in high-D
  - Helps prevent crowding and creates better separation

This asymmetry is what makes t-SNE effective!

When to Use t-SNE

Great for:

- Exploratory data analysis
- Visualizing clusters in labeled data
- Understanding complex datasets (images, text, genomics)
- Finding patterns you didn’t know existed

Common applications:

- Image datasets (like MNIST digits)
- Single-cell RNA sequencing data
- Word embeddings
- Customer segmentation

Important Limitations

1. It's for visualization only

- Cannot use for dimensionality reduction before modeling
- The low-dimensional representation changes each run

2. Computationally expensive

- Slow on large datasets (>10,000 points)
- Consider using approximations or sampling

3. Hyperparameters matter

- Perplexity (neighborhood size) significantly affects results
- Typical range: 5-50, default: 30

4. Distances are not meaningful

- Size of clusters doesn't mean anything
- Distance between clusters is not interpretable
- Only focus on local neighborhoods

Key Hyperparameter: Perplexity

Perplexity controls how t-SNE balances local vs. global structure

- Think of it as “how many neighbors to consider”
- Low perplexity (5-10): Focuses on very local structure
- High perplexity (30-50): Considers broader neighborhood

Rule of thumb: - Should be less than the number of points - Try multiple values to see which reveals the best structure - Default of 30 often works well

Practical Tips

1. Preprocessing matters

- Scale/normalize your features first
- Consider PCA preprocessing for high dimensions

2. Try different perplexities

- Run 3-5 different values
- Look for consistent patterns across runs

3. Run multiple times

- Results vary due to random initialization
- Look for stable structures

4. Don’t over-interpret

- Focus on local clusters, not global geometry
- Use alongside other analysis methods

Summary

t-SNE is a powerful visualization tool that:

- Makes high-dimensional data visible
- Preserves local neighborhood structure
- Reveals clusters and patterns effectively

Remember:

- Only for visualization, not for modeling
- Computationally intensive
- Hyperparameters (especially perplexity) matter
- Always complement with other analysis methods

Recap: t-SNE vs PCA

Feature	PCA	t-SNE
Speed	Fast	Slow
Purpose	Dimensionality reduction	Visualization
Preserves	Global variance	Local neighborhoods
Deterministic?	Yes	No (random initialization)
Distances	Meaningful	Not meaningful
Further analysis?	Yes	No

Best practice: Use PCA first to reduce to ~50 dimensions, then apply t-SNE

In-Class Exercise: Dimensionality Reduction with PCA and t-SNE

Open in Colab Open on GitHub

> The activity notebook goes live on the day of the lecture. Colab is optional — you can also open it on GitHub and run it locally.

Novembre, John, Toby Johnson, Katarzyna Bryc, et al. 2008. “Genes Mirror Geography Within Europe.” *Nature* 456 (7218): 98–101.

Strang, Gilbert. 2016. *Introduction to Linear Algebra*. 5th ed. Wellesley-Cambridge Press Welleslye, MA.

Van der Maaten, Laurens, and Geoffrey Hinton. 2008. “Visualizing Data Using t-SNE.” *Journal of Machine Learning Research* 9 (11).