Dimensionality Reduction

A Gentle (?) Introduction

Bernhard Geiger[1,2]

[1]SPSC Laboratory
Graz University of Technology

[2]Know Center Research GmbH

CLARIAH-AT Summer School
Machine Learning for Digital Scholarly Editions
September 2025

Table of Contents

Table of Contents

What is this?

What is this? A Projection!

What symbols are on the backside?

Information is lost

Information Loss

Dimensionality reduction potentially destroys information

Select method and parameterization such that

- lost information is **irrelevant** for the **task**, or
- little (**relevant**) information is lost.

Information is lost

Information Loss

Dimensionality reduction potentially destroys information

Select method and parameterization such that

- lost information is **irrelevant** for the **task**, or
- little (**relevant**) information is lost.

Take-Away

What we do after dimensionality reduction is as important for selecting method and parameterization as what we did before!

Information is lost – but is it?

Whitney Embedding Theorem

(Strong) Whitney Embedding Theorem

Any smooth real m-dimensional manifold can be smoothly embedded in $\mathbb{R}^{2m}$.

- Mathematical justification for dimensionality reduction
- Assumes smooth manifold (difficult to justify in practice) and embedding function (no tears or overlaps)
- ...but the function that we need for the embedding may not be linear (may not be a projection)
- $\rightarrow$ Nonlinear dimensionality reduction

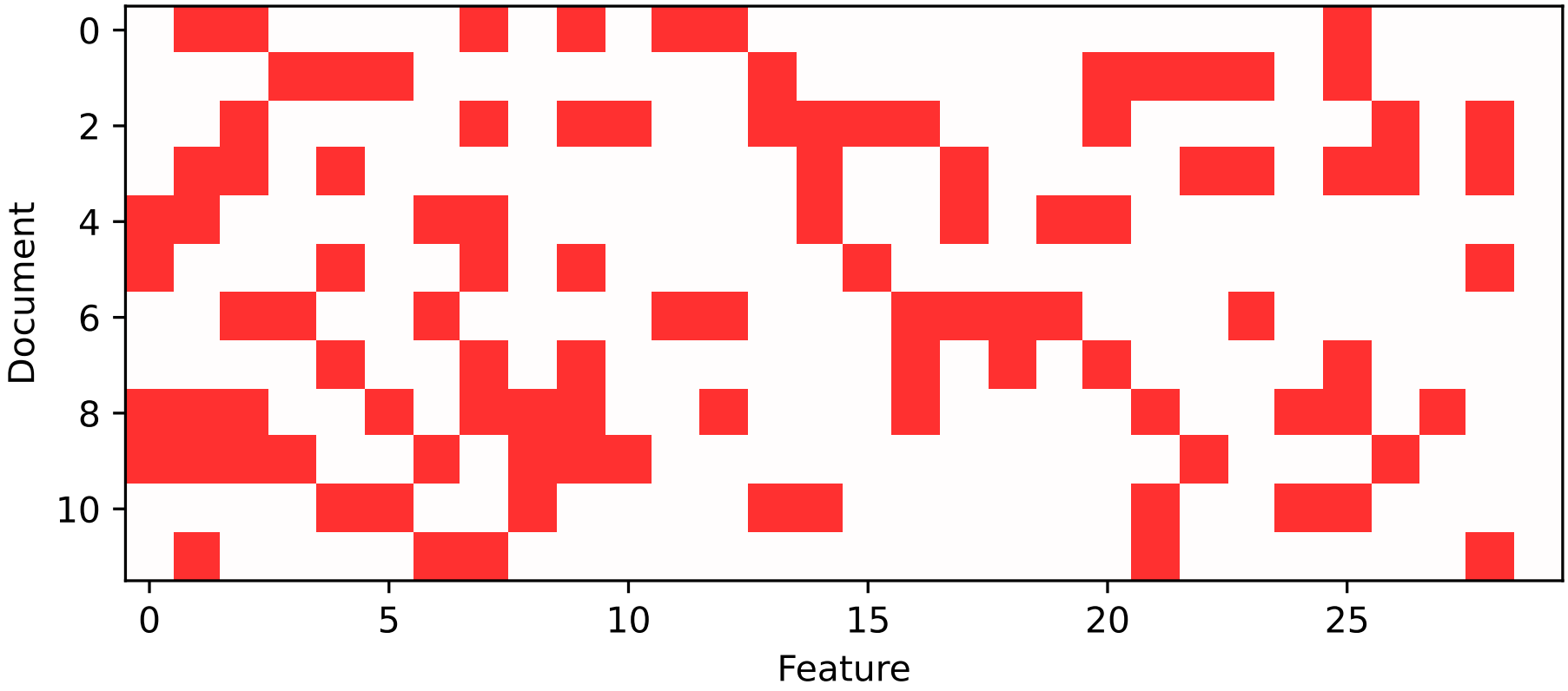
Notation

- x_ℓ is the embedding of document $\ell = 1, \ldots, n$
- x_ℓ is a D-dimensional embedding, i.e., $x_\ell \in \mathbb{R}^D$
- we assume $D > n$ (e.g., bag of words), but $D < n$ is also possible (neural embeddings)
- X is the matrix of embeddings, i.e., $X \in \mathbb{R}^{n \times D}$

$$X = \begin{bmatrix} — & x_1 & — \\ — & x_2 & — \\ \vdots & \vdots & \vdots \\ — & x_n & — \end{bmatrix}$$

  - column ℓ contains the embedding x_ℓ of document ℓ
  - row k contains the k-th dimension of all n embeddings
- $Y \in \mathbb{R}^{n \times r}$ is the matrix of reduced embeddings y_ℓ

E.g., bag of 30 words for 12 documents



Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)
- High-dimensional embeddings are "noisy"

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)
- High-dimensional embeddings are "noisy"
  - Not all information in $x_\ell \in \mathbb{R}^D$ is *relevant*

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)
- High-dimensional embeddings are "noisy"
  - Not all information in $x_\ell \in \mathbb{R}^D$ is *relevant*
  - Dimensionality reduction removes noise and (hopefully) preserves dominant topic structure

Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)
- High-dimensional embeddings are "noisy"
  - Not all information in $x_\ell \in \mathbb{R}^D$ is *relevant*
  - Dimensionality reduction removes noise and (hopefully) preserves dominant topic structure
  - Information is lost on purpose!

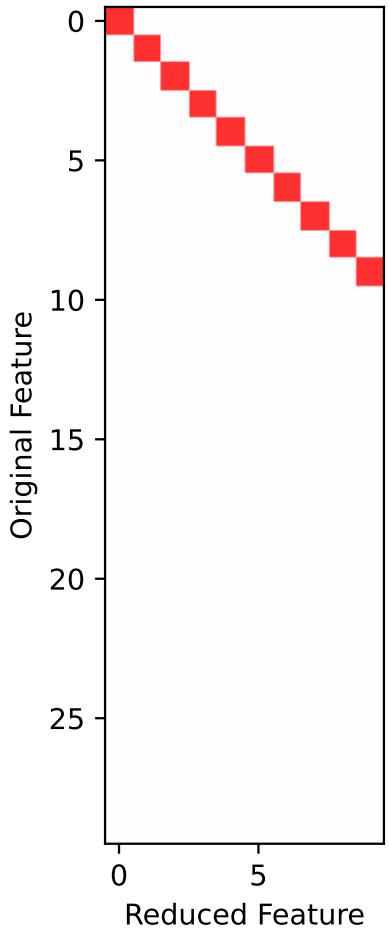
Why Dimensionalty Reduction in BERTopic?

- Embeddings are usually high-dimensional ($D > 100$)
  - In high-dimensional spaces, distances become meaningless
  - $\rightarrow$ The closest points are as far apart as the most distant points
  - Clustering often relies on distances $\rightarrow$ clustering methods applied directly on X may fail (e.g., a single large cluster)
- High-dimensional embeddings are "noisy"
  - Not all information in $x_\ell \in \mathbb{R}^D$ is *relevant*
  - Dimensionality reduction removes noise and (hopefully) preserves dominant topic structure
  - Information is lost on purpose!
- Working with high-dimensional data has a high computational complexity $\rightarrow$ dimensionality reduction increases speed of subsequent processing steps

Table of Contents

Orthogonal Projection

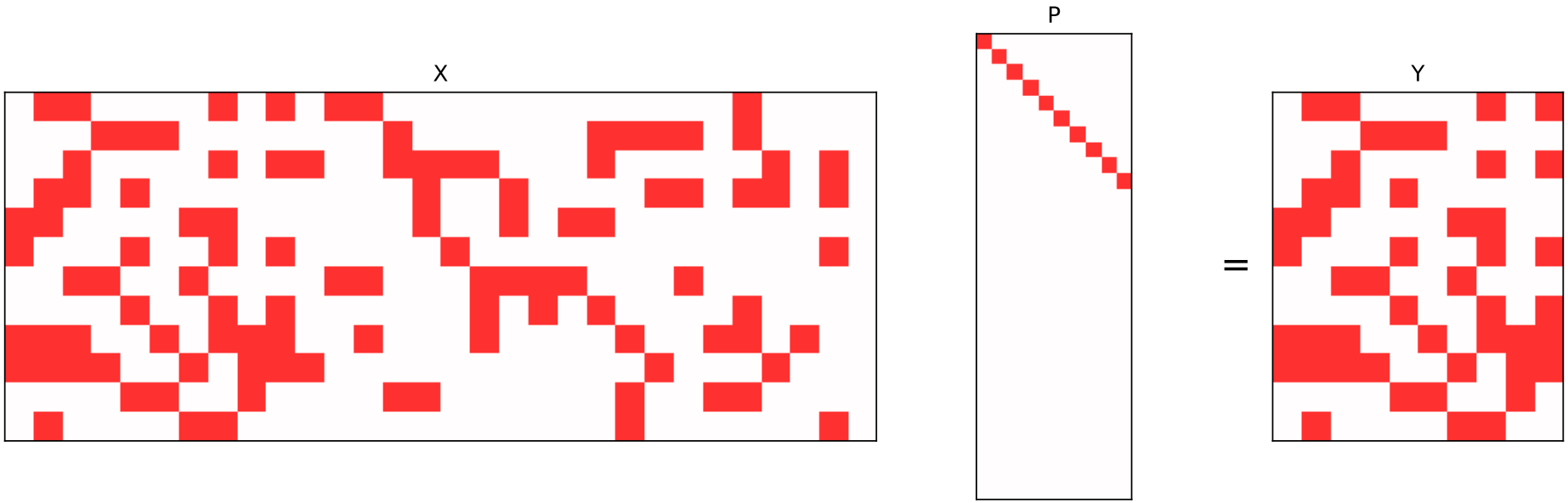
A projection matrix P selects a given number of coordinates



Orthogonal Projection (cont'd)

$$Y = XP$$

Orthogonal Projection (cont'd)



Singular Value Decomposition (SVD)

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

where

- U and V are unitary matrices (rotations) and
- U and V contain the left and right singular vectors of X
- Σ is a diagonal matrix of singular values $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$

Singular Value Decomposition (SVD)

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

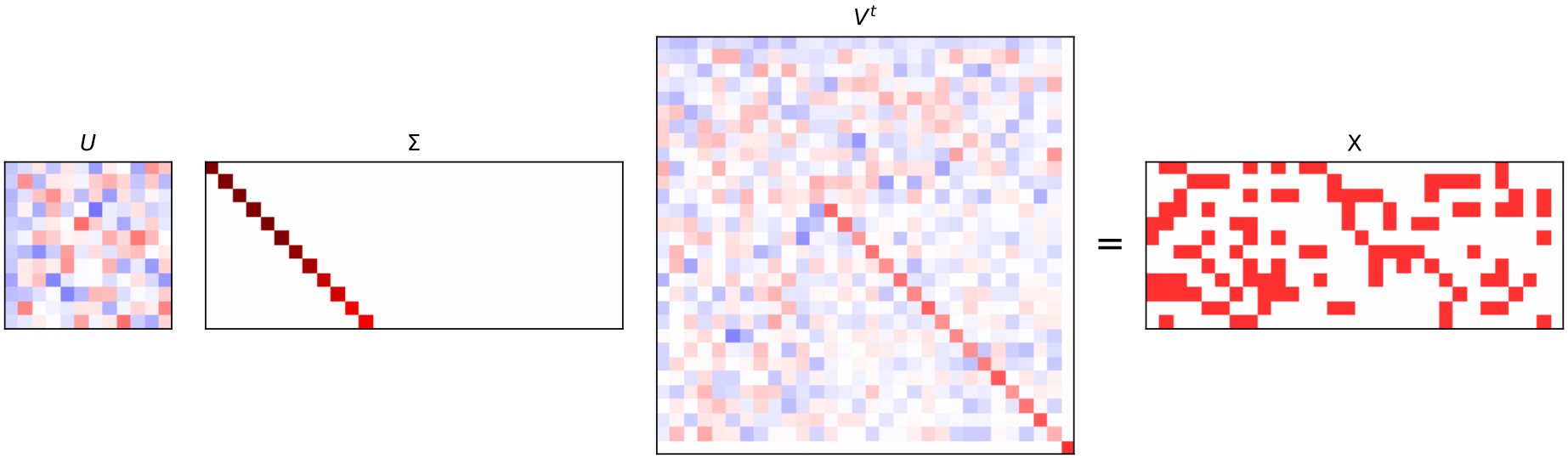
where

- U and V are unitary matrices (rotations) and
- U and V contain the left and right singular vectors of X
- Σ is a diagonal matrix of singular values $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$

Take-Away

The matrices U and V allow us to look at the data "from the best angle" – it's a change of perspective!

SVD (cont'd)



Truncated SVD

Effectively, SVD writes X as a sum of *simple* matrices:

$$X = \sigma_1 X_1 + \sigma_2 X_2 + \cdots + \sigma_n X_n$$

Approximate X using a smaller number $r < n$ of *simple* matrices:

$$X \approx \tilde{X} = \sigma_1 X_1 + \sigma_2 X_2 + \cdots + \sigma_r X_r$$

Take-Away

For a given matrix X and a fixed *rank* r, $\tilde{X}$ is the best linear approximation of X (in some well-defined sense).

Truncated SVD (cont'd)

If

- U and V allow us to change perspective
- SVD allows us to approximate X as a sum of $r < n$ simple matrices (weighted by singular values σ_k from Σ)

can we combine changing perspective with orthogonal projection?

$$Y = XVP = U\Sigma P = U_r\Sigma_r$$

where Σ_r is a diagonal matrix with the r largest singular values and U_r contains the corresponding r columns of U.

Truncated SVD (cont'd)

If

- U and V allow us to change perspective
- SVD allows us to approximate X as a sum of $r < n$ simple matrices (weighted by singular values σ_k from Σ)

can we combine changing perspective with orthogonal projection?

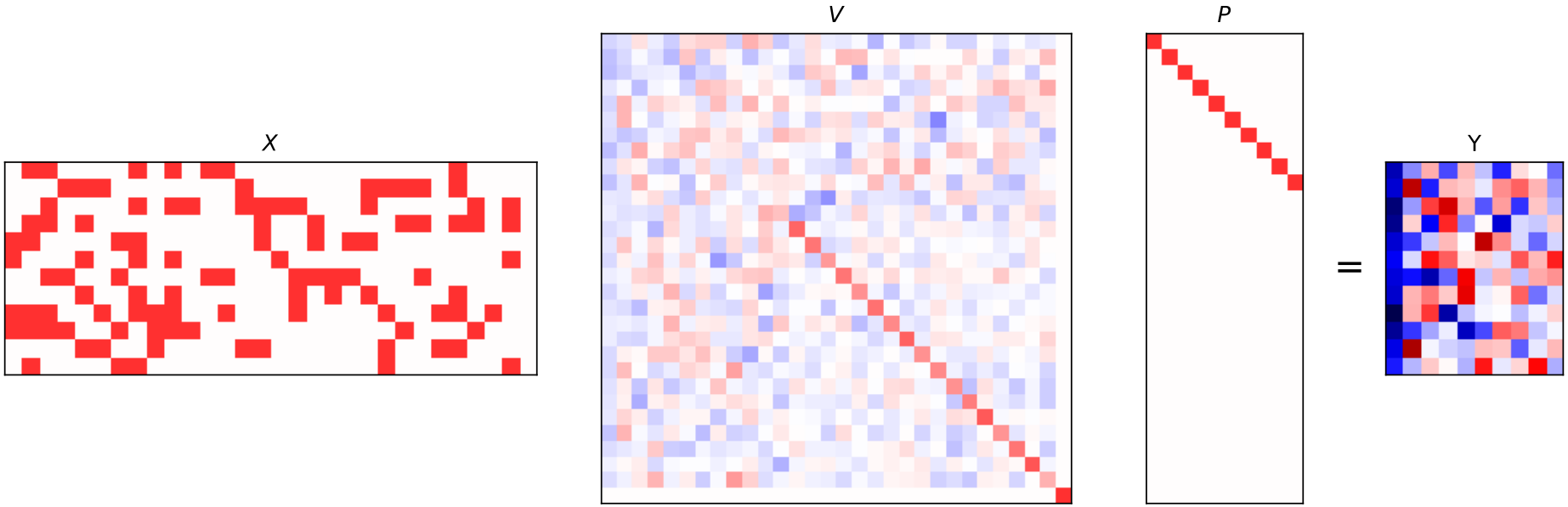
$$Y = XVP = U\Sigma P = U_r\Sigma_r$$

where Σ_r is a diagonal matrix with the r largest singular values and U_r contains the corresponding r columns of U.

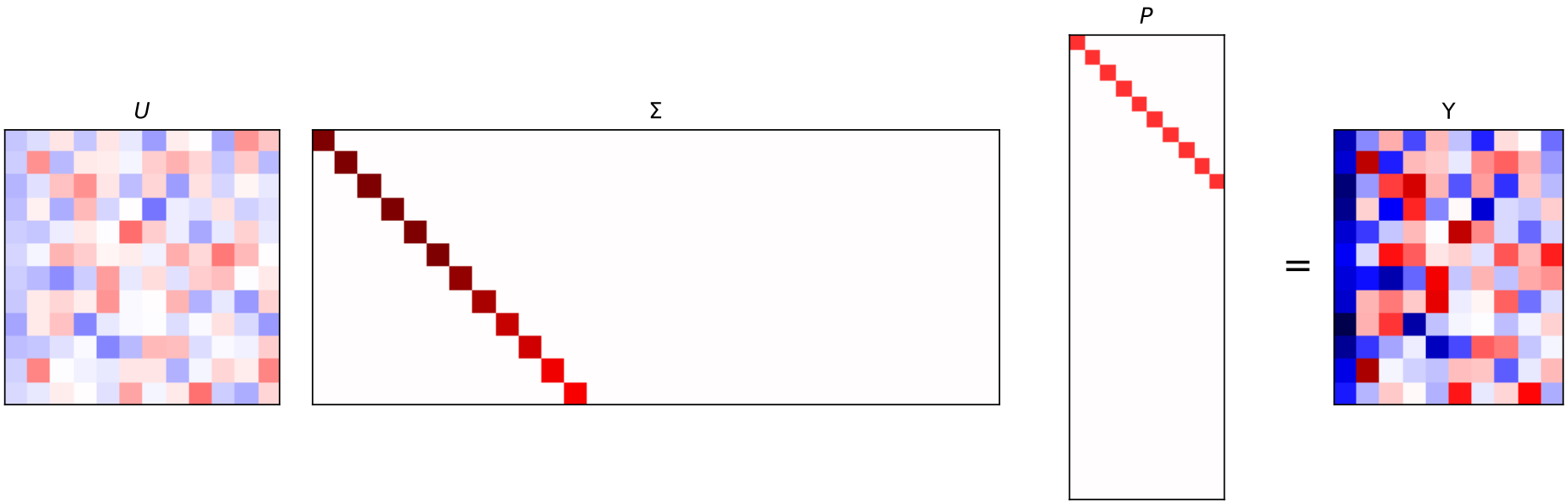
Take-Away

With SVD, we take the photo "from the best angle".

Truncated SVD (cont'd)



Truncated SVD (cont'd)

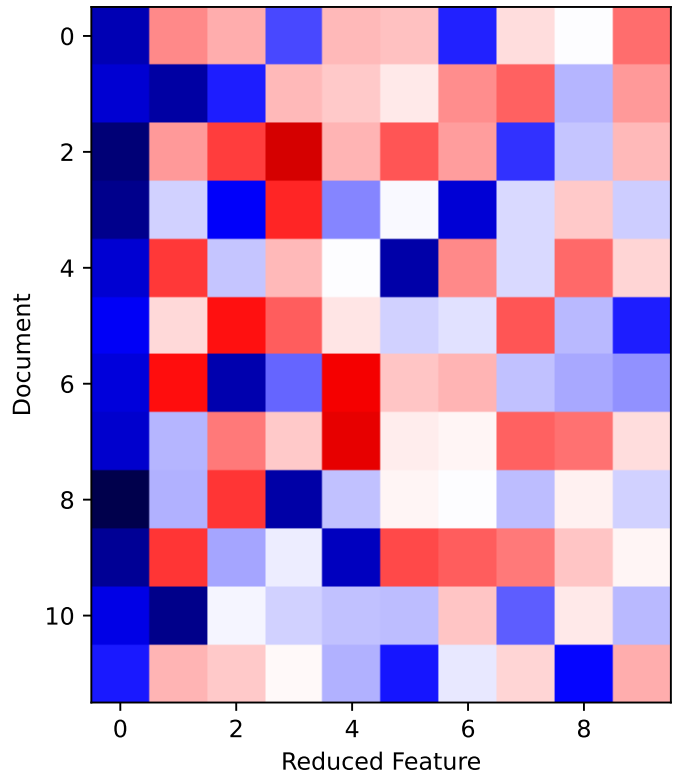


Truncated SVD (cont'd)

```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=10,random_state=42)

svd.fit(X)
Y=svd.transform(X)
```



(Columns could change signs.)

Principal Components Analysis (PCA)

1. Remove the mean of each feature
2. Compute the Gramian
3. Perform eigenvalue decomposition $C = VDV^T$
4. Use eigenvectors V to project the data X

Principal Components Analysis (PCA)

1. Remove the mean of each feature
2. Compute the Gramian
3. Perform eigenvalue decomposition $C = VDV^T$
4. Use eigenvectors V to project the data X

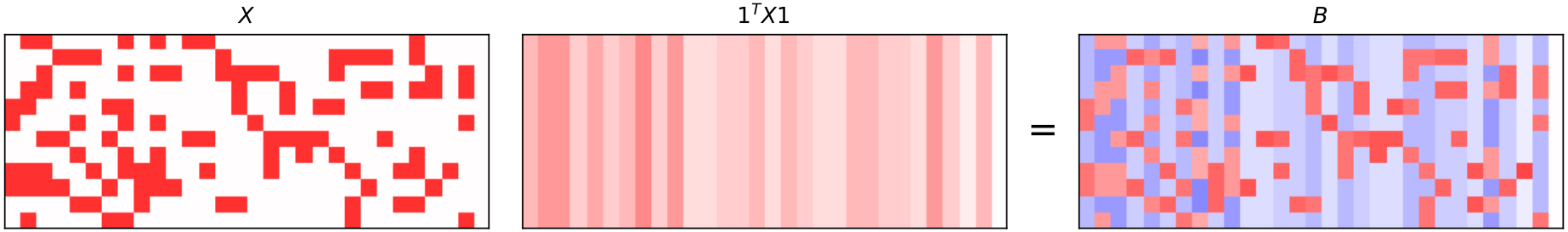
Take-Away

PCA is just SVD on centered data.

PCA (cont'd)

1) Remove the mean of each feature

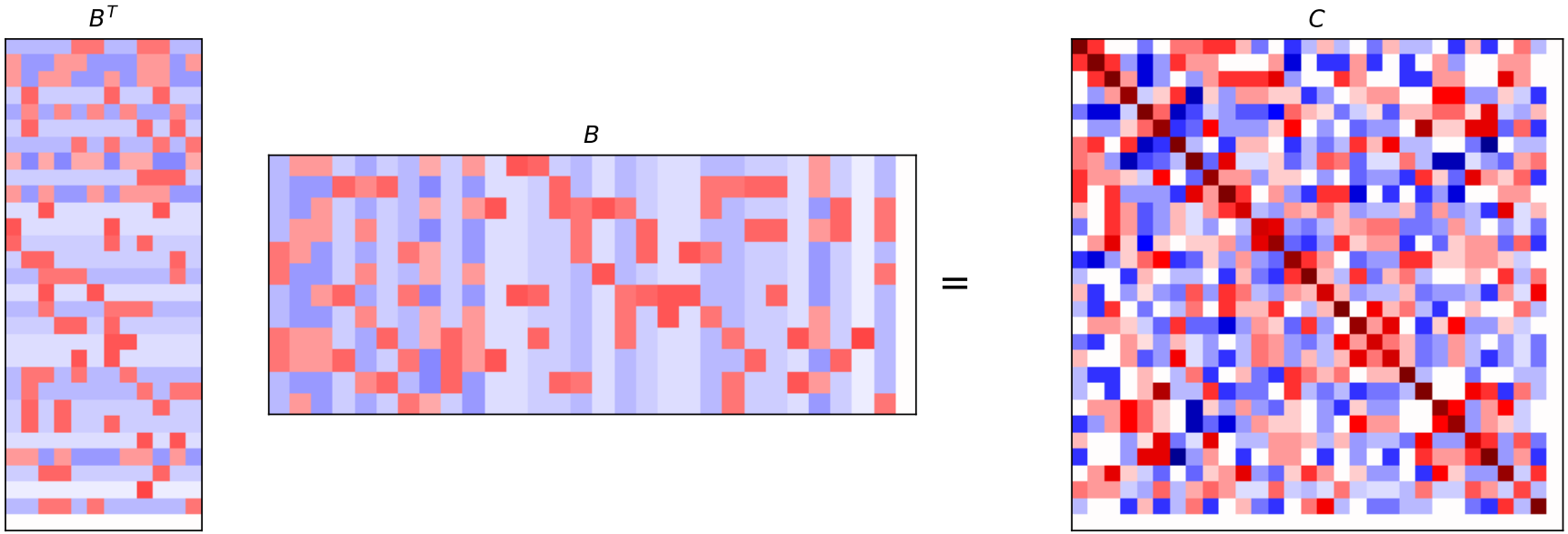
$$B = X - \frac{1}{n} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} X \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$



PCA (cont'd)

2) Compute the Gramian

$$C = B^T B$$

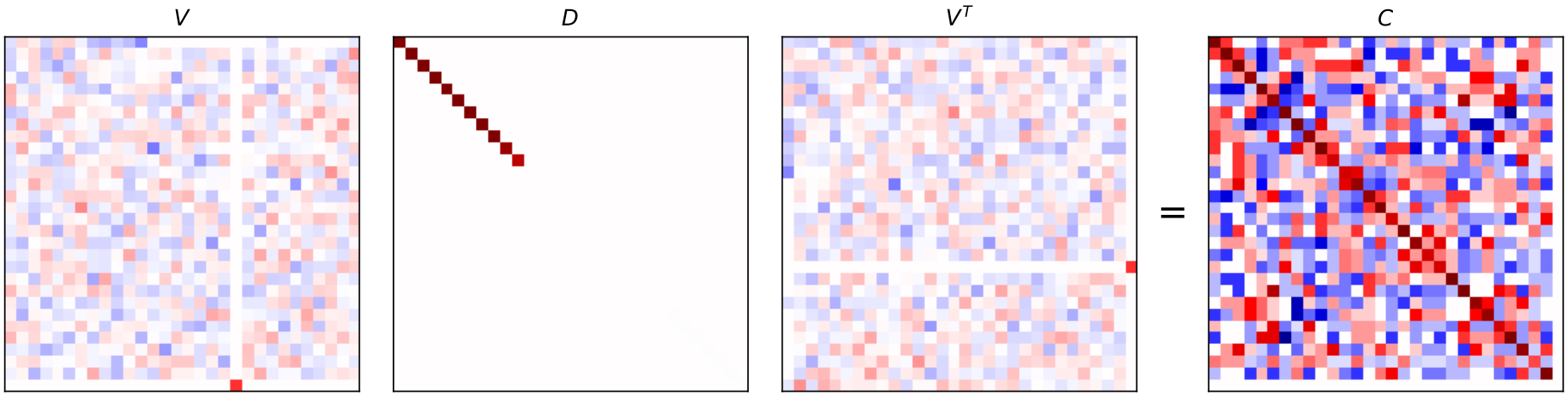


The Gramian describes the correlation between (centered) features.

PCA (cont'd)

3) Perform eigenvalue decomposition

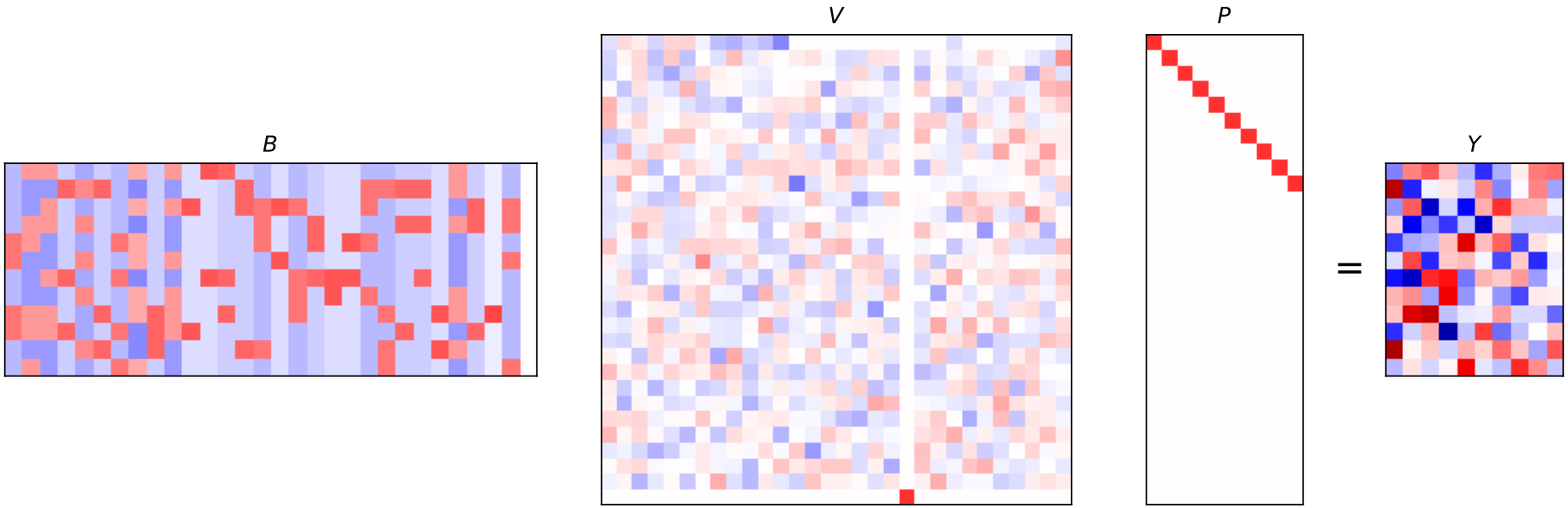
$$C = VDV^T$$



PCA (cont'd)

3) Use eigenvectors V to project the data X

$$Y = BVP$$

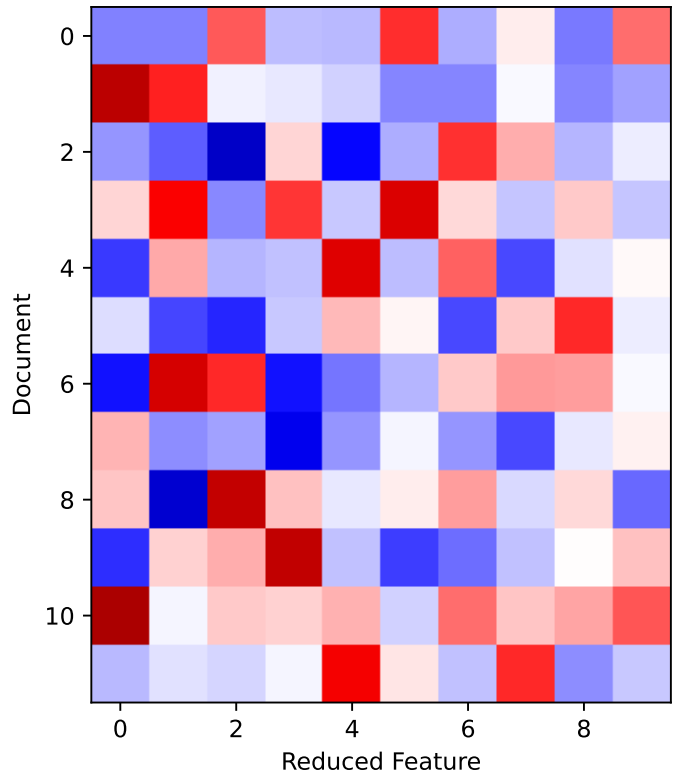


PCA (cont'd)

```
from sklearn.decomposition import PCA

pca = PCA(n_components=10)

pca.fit(X)
Y=pca.transform(X)
```



(Columns could change signs.)

Experiments

Experiments!

t-distributed stochastic neighbor embedding (t-SNE)

Motivation

For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping.

van der Maaten & Hinton, "Visualizing Data using t-SNE", JMLR, 9:2579-2605, 2008.

t-distributed stochastic neighbor embedding (t-SNE)

Motivation

For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping.

van der Maaten & Hinton, "Visualizing Data using t-SNE", JMLR, 9:2579-2605, 2008.

Two-stage process for dimensionality reduction:

1. Construct probability distribution between pairs of documents based on their high-dimensional embeddings x_ℓ
2. Define similar distribution over points y_ℓ in a low-dimensional space

t-SNE (cont'd)

1) Construct probability distribution between pairs of documents

1.1) Define the probability

$$p_{j|i} \propto e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}$$

where σ_i^2 is selected such that $H(P_i)$ has a fixed value (e.g., $\log 5$ or $\log 50$)

$$H(P_i) = -\sum_{j=1}^{n} p_{j|i} \log p_{j|i}$$

Intuition

The perplexity [this fixed value] can be interpreted as a smooth measure of the effective number of neighbors.

van der Maaten & Hinton, "Visualizing Data using t-SNE", JMLR, 9:2579-2605, 2008.

t-SNE (cont'd)

1) Construct probability distribution between pairs of documents

1.2) Symmetrize distribution via

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

Intuition

$p_{j|i}$ is the conditional probability that x_i would pick x_j as its neighbor – the closer x_j is to x_i, the higher is the probability

van der Maaten & Hinton, "Visualizing Data using t-SNE", JMLR, 9:2579-2605, 2008.

t-SNE (cont'd)

2) Define similar distribution over points y_ℓ in a low-dimensional space $\mathbb{R}^r$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{\ell \neq k} (1 + \|y_k - y_\ell\|^2)^{-1}} \qquad q_{ii} = 0$$

and find points $\{y_\ell\}$ such that

$$\mathbb{D}(p\|q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

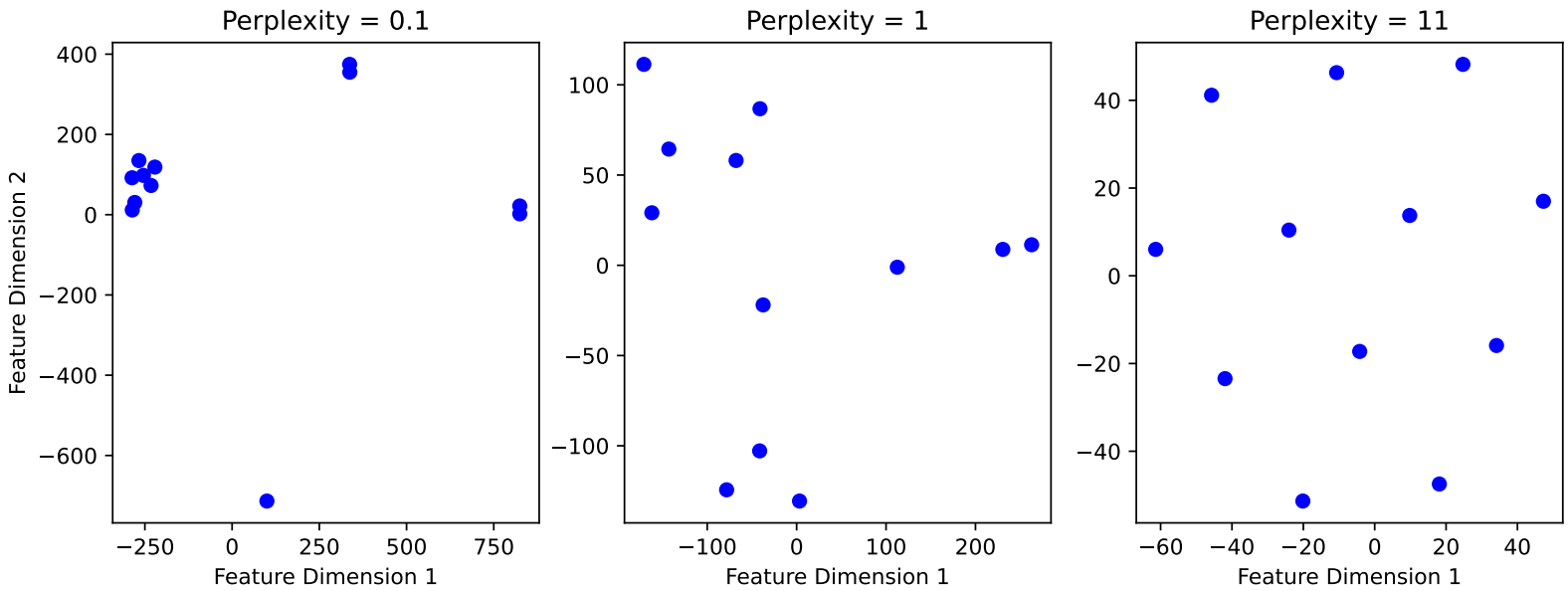
is minimized.

t-SNE (cont'd)

```
from sklearn.manifold import TSNE

t_sne=TSNE(n_components=2, learning_rate='auto',
        init='random', perplexity=.25)

Y=t_sne.fit_transform(X)
```



t-SNE (cont'd)

- t-SNE is often used for visualization ($r = 2, 3$)
- Depends on Euclidean distance (curse of dimensionality if D is large!)
- Perplexity is a critical parameter
- Stochastic implementation (different results for every run)
- Creates clusters when there are none or splits larger clusters (problematic for topic modeling!)

Intuition

[...] preserves neighborhoods, but not distances and densities [...]

Schubert & Gertz, "Intrinsic t-Stochastic Neighbor Embedding for Visualization and Outlier Detection – A Remedy Against the Curse of Dimensionality?", Int. Conf. Similarity Search and Applications, 2017.

t-SNE (cont'd)

Take-Away

The problem with t-SNE (and UMAP) is that it does not preserve distances nor density. It only to some extent preserves nearest-neighbors. The difference is subtle, but affects any density- or distance based algorithm. While clustering after t-SNE will sometimes (often?) work, you will never know whether the "clusters" you find are real, or just artifacts of t-SNE. You will not be able to explain the clusters. You may just be seeing 'shapes in clouds'.

Erich Schubert, `https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-t-sne`

t-SNE (cont'd)

Take-Away

In conclusions, use t-SNE for visualization (and try different parameters to get something visually pleasing!), but rather do not run clustering afterwards, in particular do not use distance- or density based algorithms, as this information was intentionally (!) lost.

Erich Schubert, `https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-t-sne`

Take-Away

Different ways in which t-SNE can be misleading:

`https://distill.pub/2016/misread-tsne/`

Uniform manifold approximation and projection (UMAP)

Underlying Assumptions

1. There exists a manifold on which the data would be uniformly distributed.
2. The underlying manifold of interest is locally connected.
3. Preserving the topological structure of this manifold is the primary goal.

McInnes et al. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction", `arXiv:1802.03426`

Take-Away

Also UMAP focuses on topological structure (neighborhood) but not density.

UMAP (cont'd)

Two-stage process for dimensionality reduction:

1. Construct a weighted graph with edge weights based on the high-dimensional embeddings x_ℓ
2. Define a graph layout in a low-dimensional space to find reduced embeddings y_ℓ

UMAP (cont'd)

Two-stage process for dimensionality reduction:

1. Construct a weighted graph with edge weights based on the high-dimensional embeddings x_ℓ
2. Define a graph layout in a low-dimensional space to find reduced embeddings y_ℓ

Remember t-SNE?

1. Construct probability distribution between pairs of documents based on their high-dimensional embeddings x_ℓ
2. Define similar distribution over points y_ℓ in a low-dimensional space

UMAP (cont'd)

1) Construct a weighted graph

1.1) k-nearest neighbor graph with edge weight

$$w_{i\to j} = e^{-\frac{\max\{0, d(x_i,x_j)-\rho_i\}}{\sigma_i}} \quad \text{or} \quad w_{i\to j} = 0$$

where $d(\cdot,\cdot)$ is some distance function and where σ_i is such that

$$\sum_{j=1}^{n} w_{i\to j} = \log k$$

Intuition

The selection of [nearest neighbor distance] ρ_i ensures that x_i connects to at least one other data point with an edge of weight 1; [...; it] improves the representation on very high dimensional data where other algorithms such as t-SNE begin to suffer from the curse of dimensionality.

McInnes et al., `arXiv:1802.03426`

UMAP (cont'd)

1) Construct a weighted graph

1.2) Symmetrize the weights via

$$w_{ij} = w_{i \to j} + w_{j \to i} - w_{i \to j} w_{j \to i}$$

Intuition

This formula derives from the use of the probabilistic t-conorm used in unioning the fuzzy simplicial sets.

McInnes et al., `arXiv:1802.03426`

UMAP (cont'd)

2) Define a graph layout in a low-dimensional space

2.1) Start with a spectral layout with $y_\ell \in \mathbb{R}^r$

2.2) Move embeddings y_ℓ such that the correspondingly weighted graph in $\mathbb{R}^r$ has minimum cross-entropy with the original weighted graph in $\mathbb{R}^D$.

Take-Away

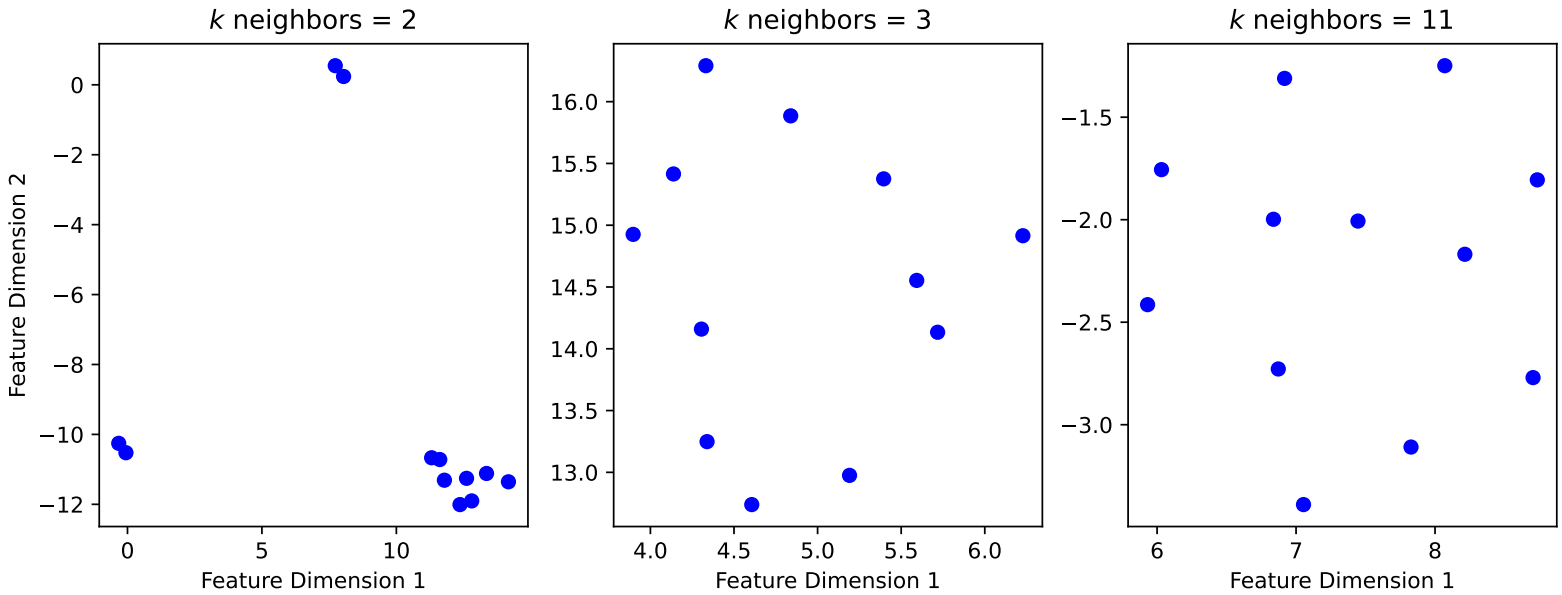
Since UMAP has a deterministic initialization, the results are more stable over multiple run than for t-SNE.

UMAP (cont'd)

```
import umap

reducer = umap.UMAP(n_neighbors=5,n_components=2,
        metric='euclidean',min_dist=0.1)

Y=reducer.fit_transform(X)
```



UMAP (cont'd)

- UMAP is often used for dimensionality reduction ($r > 2$)
- Distance function can be chosen (Euclidean distance $\rightarrow$ curse of dimensionality if D is large!)
- Number k of neighbors is a critical parameter
- Stochastic implementation (but relatively stable)
- Can create clusters when there are none, split clusters, and also does not preserve density (problematic for topic modeling!)

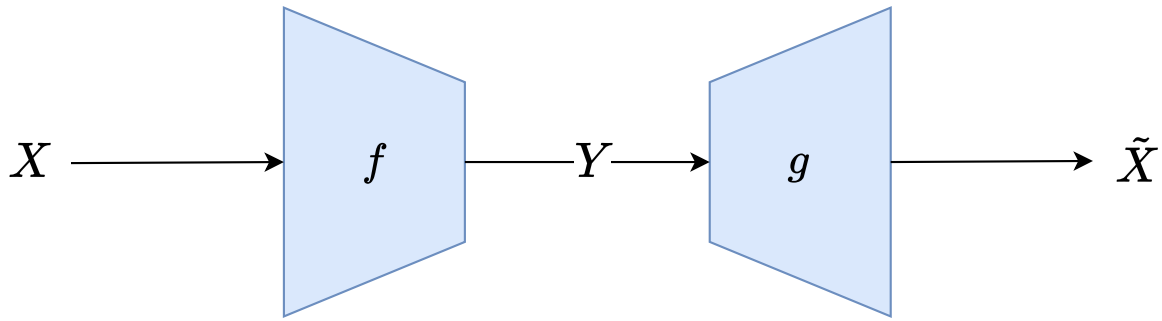
Take-Away

UMAP suffers from similar problems as t-SNE, but is much faster, more stable during multiple runs, and its hyperparameters are easier to interpret.

Experiments

Experiments!

Autoencoders



Neural networks that directly learn a function $f\colon \mathbb{R}^D \to \mathbb{R}^r$ *and* an "inverse" function $g\colon \mathbb{R}^r \to \mathbb{R}^D$

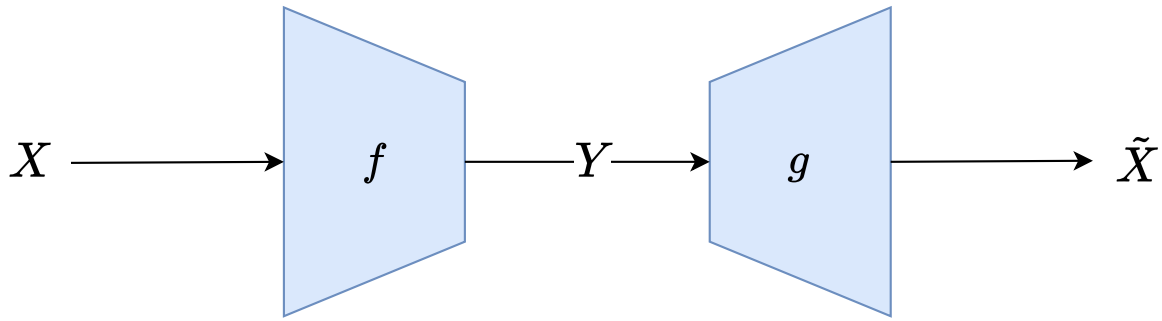
$$y_\ell = f(x_\ell) \qquad \tilde{x}_\ell = g(y_\ell) = g(f(x_\ell))$$

such that

$$\sum_{i=1}^{n} \|x_i - \tilde{x}_i\|^2$$

is minimized.

Autoencoders (cont'd)



- Work well for large and high-dimensional datasets
- Can model nonlinear behavior very well
- Require separate training (experience!) and large datasets
- Are not (easily) interpretable

Table of Contents

How to Choose the Method?

Linear Methods

- If you believe that information lies in a *linear* subspace
- Especially useful for large, high-dimensional datasets ($n, D \gg$)
- Usually fast (computationally efficient)
- PCA
  - dense, high-dimensional embeddings (multilingual embeddings)
  - sometimes used as a preprocessing step before UMAP
- Truncated SVD
  - sparse, high-dimensional embeddings (BoW, TF-IDF)

How to Choose the Method? (cont'd)

Nonlinear Methods

- If you believe that information lies on a *nonlinear* manifold
- UMAP
  - preserves local and some global structure
  - fast and scalable to large corpora
  - recommended choice for BERTopic
- t-SNE
  - unreliable for clustering and slow, do not use
- Autoencoders
  - require complex training and tuning
  - purportedly useful for large corpora of domain-specific language

How to Parameterize the Method?

PCA/Truncated SVD

Heuristics to choose r based on the approximation

$$X \approx \tilde{X} = \sigma_1 X_1 + \sigma_2 X_2 + \cdots + \sigma_r X_r$$

- "Elbow rule"
- Percentage of variation explained

How to Parameterize the Method? (cont'd)

UMAP

- Documentation recommends $r = 5$?
- Number k of neigbors
  - k is small $\rightarrow$ many small clusters, fine-grained topics
  - k is large $\rightarrow$ fewer, broader topics
- Metric
  - Cosine similarity is often recommended as metric for sentence embeddings
  - Euclidean distance may summer from curse of dimensionality

How to Parameterize the Method? (cont'd)

Autoencoder

- Loss function
- Encoder and decoder architecture (layers, layer widths, activation functions, dropout, batch normalization,...)
- Optimizer settings (learning rate, batch size, epochs,...)
- Modeling assumptions (stochastic vs. deterministic, dimension of latent space/reduced embedding r,...)

How to Parameterize the Method? (cont'd)

Autoencoder

- Loss function
- Encoder and decoder architecture (layers, layer widths, activation functions, dropout, batch normalization,...)
- Optimizer settings (learning rate, batch size, epochs,...)
- Modeling assumptions (stochastic vs. deterministic, dimension of latent space/reduced embedding r,...)

Take-Away

It's an art.

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...
- How do we select *good combinations* of embedding, dimensionality reduction, and clustering?

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...
- How do we select *good combinations* of embedding, dimensionality reduction, and clustering?
- Can we reduce dimensionality and cluster in one step?

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...
- How do we select *good combinations* of embedding, dimensionality reduction, and clustering?
- Can we reduce dimensionality and cluster in one step?
  - e.g., deep clustering techniques

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...
- How do we select *good combinations* of embedding, dimensionality reduction, and clustering?
- Can we reduce dimensionality and cluster in one step?
  - e.g., deep clustering techniques
- How do we know if we must/do not need to reduce dimensionality?

Further Topics (not covered here)

- Dimensionality reduction is related to other, popular topics: Representation learning, manifold learning, metric learning,...
- How do we select *good combinations* of embedding, dimensionality reduction, and clustering?
- Can we reduce dimensionality and cluster in one step?
  - e.g., deep clustering techniques
- How do we know if we must/do not need to reduce dimensionality?
- How can we assess the performance of dimensionality reduction in isolation and within the BERTopic framework?

Experiments

Experiments!