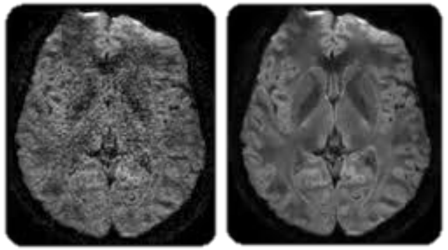
Unsupervised Learning: Dimensionality Reduction (PCA and other methods)

CS771: Introduction to Machine Learning

Piyush Rai

Dimensionality Reduction



- Goal: Reduce the dimensionality of each input $\boldsymbol{x}_n \in \mathbb{R}^D$

$\boldsymbol{z}_n \in \mathbb{R}^K$ $(K \ll D)$ is a compressed version of $\boldsymbol{x}_n$

$$\boldsymbol{z}_n = f(\boldsymbol{x}_n)$$

- Also want to be able to (approximately) reconstruct $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$

Often $\tilde{\boldsymbol{x}}_n$ is a "cleaned" version of $\boldsymbol{x}_n$ (the loss in information is often the noise/redundant information in $\boldsymbol{x}_n$)

$$\tilde{\boldsymbol{x}}_n = g(\boldsymbol{z}_n) = g(f(\boldsymbol{x}_n)) \approx \boldsymbol{x}_n$$

- Sometimes f is called "encoder" and g is called "decoder". Can be linear/nonlinear

- These functions are learned by minimizing the distortion/reconstruction error of inputs

$$\mathcal{L} = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n||^2 = \sum_{n=1}^{N} ||\boldsymbol{x}_n - g(f(\boldsymbol{x}_n))||^2$$

Image source: Manjon et al (2013): Diffusion Weighted Image Denoising using overcomplete Local PCA

Dimensionality Reduction

- Choosing f and g as linear transformations $\boldsymbol{W}^T$ $(K \times D)$ and $\boldsymbol{W}$, respectively

$\boldsymbol{z}_n$

$\tilde{\boldsymbol{x}}_n = \boldsymbol{W}\boldsymbol{z}_n$

$$\mathcal{L} = \sum_{n=1}^{N} \|\boldsymbol{x}_n - g(f(\boldsymbol{x}_n))\|^2 = \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_n\|^2$$

Principal Component Analysis (PCA). Proof shortly

- Minimizer of $\mathcal{L}$, if the K columns of $\boldsymbol{W}$ are orthonormal, are top K eigenvectors of

$D \times D$ empirical covariance matrix of inputs

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\top = \frac{1}{N}\boldsymbol{X}_c^\top \boldsymbol{X}_c$$

$\boldsymbol{X}_c$ is the $N \times D$ matrix of inputs after centering each input (subtracting off the mean of inputs from each input)

- The matrix $\boldsymbol{W}$ does a "linear projection" of each input $\boldsymbol{x}_n \in \mathbb{R}^D$ into a K dim space
  - $\boldsymbol{z}_n = \boldsymbol{W}^T\boldsymbol{x}_n \in \mathbb{R}^K$ denotes this linear projection
- Note: If we use $K = D$ eigenvectors for $\boldsymbol{W}$, the reconstruction will be perfect $(\mathcal{L} = 0)$

Dimensionality Reduction

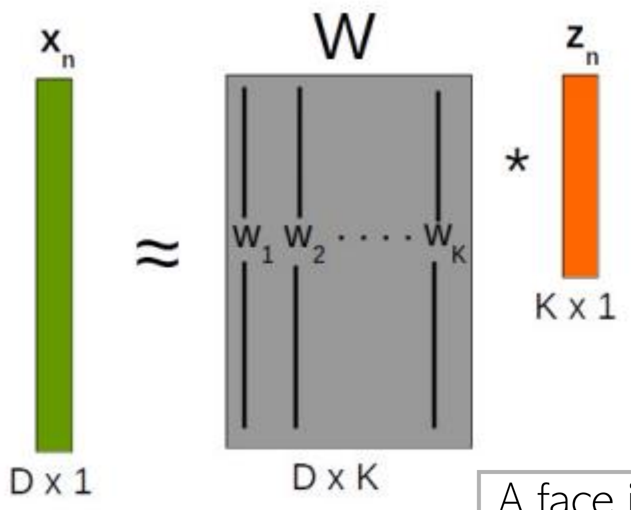
- Consider a linear model of the form

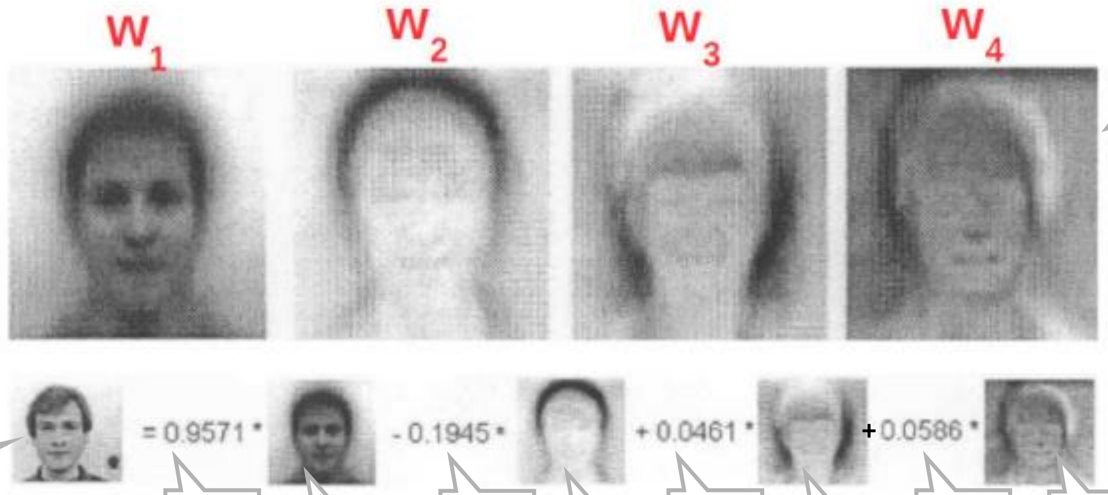
Not necessarily PCA where the columns of $\boldsymbol{W}$ were orthonormal

$$\boldsymbol{x}_n \approx \tilde{\boldsymbol{x}}_n = \boldsymbol{W}\boldsymbol{z}_n = \sum_{k=1}^{K} z_{nk}\boldsymbol{w}_k$$

$\boldsymbol{w}_k$ is the k-th column of $\boldsymbol{W}$

- Above means that each $\boldsymbol{x}_n$ is appox a linear comb of K vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K$





$K = 4$ "basis" face images

Each "basis" image is like a "template" that captures the common properties of face images in the dataset

A face image $\boldsymbol{x}_n \in \mathbb{R}^D$

z_{n1} $\boldsymbol{w}_1$ z_{n2} $\boldsymbol{w}_2$ z_{n3} $\boldsymbol{w}_3$ z_{n4} $\boldsymbol{w}_4$

- In this example, $\boldsymbol{z}_n \in \mathbb{R}^K$ $(K = 4)$ is a low-dim feature rep. for each image $\boldsymbol{x}_n \in \mathbb{R}^D$

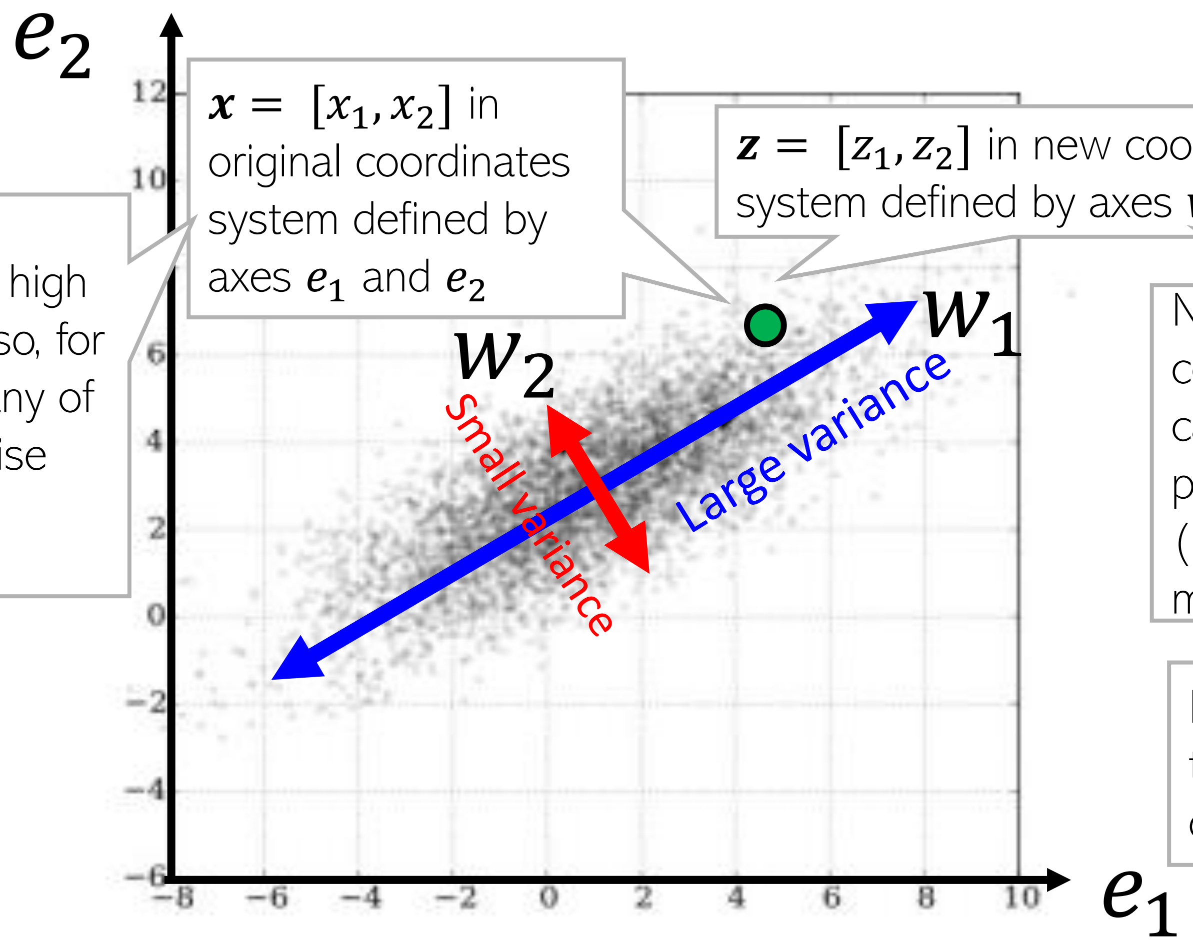
Principal Component Analysis (PCA)

- PCA learns a different and more economical coordinate system to represent data

With this representation, both coordinates x_1 and x_2 have a high variance across all the points so, for any point $\boldsymbol{x}$, we can't ignore any of these two coordinates otherwise we will lose considerable information about the data

$\boldsymbol{x} = [x_1, x_2]$ in original coordinates system defined by axes $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$

$\boldsymbol{z} = [z_1, z_2]$ in new coordinates system defined by axes $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$



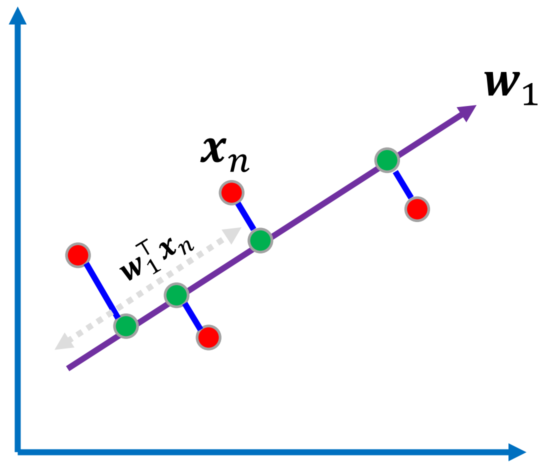
Not much variance in the z_2 coordinates of all the points so we can ignore z_2 and represent each point using just the first coordinate (i.e., $\boldsymbol{z} = [z_1]$) without losing much information about the data

Dimensionality reduction! From two dimensions ($\boldsymbol{x} \in \mathbb{R}^2$) to one dimension ($\boldsymbol{z} \in \mathbb{R}$)

- Top eigenvectors of the covariance matrix of inputs give us the large variance directions

Finding Max. Variance Directions

- Consider projecting an input $\boldsymbol{x}_n \in \mathbb{R}^D$ along a direction $\boldsymbol{w}_1 \in \mathbb{R}^D$
- Projection/embedding of $\boldsymbol{x}_n$ (red points below) will be $\boldsymbol{w}_1^\top \boldsymbol{x}_n$ (green pts below)



Mean of projections of all inputs:

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{w}_1^\top \boldsymbol{x}_n = \boldsymbol{w}_1^\top\left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\right) = \boldsymbol{w}_1^\top \boldsymbol{\mu}$$

$\boldsymbol{S}$ is the $D \times D$ cov matrix of the data:

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\top$$

Variance of the projections:

$$\frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{w}_1^\top \boldsymbol{x}_n - \boldsymbol{w}_1^\top \boldsymbol{\mu})^2 = \frac{1}{N}\sum_{n=1}^{N} \{\boldsymbol{w}_1^\top(\boldsymbol{x}_n - \boldsymbol{\mu})\}^2 = \boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1$$

For already centered data, $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top$

- Want $\boldsymbol{w}_1$ such that variance $\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1$ is maximized

$$\underset{\boldsymbol{w}_1}{\operatorname{argmax}}\ \boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 \quad \text{s.t.} \quad \boldsymbol{w}_1^\top \boldsymbol{w}_1 = 1$$

Need this constraint otherwise the objective's max will be infinity

Max. Variance Direction

Variance along the direction $\boldsymbol{w}_1$

- Our objective function was $\arg\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1$ s.t. $\boldsymbol{w}_1^\top \boldsymbol{w}_1 = 1$
- Can construct a Lagrangian for this problem

$$\arg\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 + \lambda_1 (1 - \boldsymbol{w}_1^\top \boldsymbol{w}_1)$$

- Taking derivative w.r.t. $\boldsymbol{w}_1$ and setting to zero gives $\boldsymbol{S}\boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_1$
- Therefore $\boldsymbol{w}_1$ is an eigenvector of the cov matrix $\boldsymbol{S}$ with eigenvalue λ_1
- Claim: $\boldsymbol{w}_1$ is the eigenvector of $\boldsymbol{S}$ with largest eigenvalue λ_1. Note that

$$\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_1^\top \boldsymbol{w}_1 = \lambda_1$$

- Thus variance $\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1$ will be max. if λ_1 is the largest eigenvalue (and $\boldsymbol{w}_1$ is the corresponding top eigenvector; also known as the first Principal Component)
- Other large variance directions can also be found likewise (with each being orthogonal to all others) using the eigendecomposition of cov matrix $\boldsymbol{S}$ (this is PCA)

Note: Total variance of the data is equal to the sum of eigenvalues of $\boldsymbol{S}$, i.e., $\sum_{d=1}^{D} \lambda_d$

PCA would keep the top $K < D$ such directions of largest variances

Note: In general, $\boldsymbol{S}$ will have D eigvecs

The PCA Algorithm

- Center the data (subtract the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$ from each data point)
- Compute the $D \times D$ covariance matrix $\mathbf{S}$ using the centered data matrix $\mathbf{X}$ as

$$\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$$

(Assuming **X** is arranged as $N \times D$)

- Do an eigendecomposition of the covariance matrix $\mathbf{S}$ (many methods exist)
- Take top $K < D$ leading eigvectors $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K\}$ with eigvalues $\{\lambda_1, \lambda_2, \dots, \lambda_K\}$
- The K-dimensional projection/embedding of each input is

$$\mathbf{z}_n \approx \mathbf{W}_K^\top \boldsymbol{x}_n$$

$\mathbf{W}_\mathrm{K} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K]$ is the "projection matrix" of size $D \times K$

Note: Can decide how many eigvecs to use based on how much variance we want to capture (recall that each λ_k gives the variance in the k^{th} direction (and their sum is the total variance)

The Reconstruction Error View of PCA

- Representing a data point $\boldsymbol{x}_n = [x_{n1}, x_{n2}, \dots, x_{nD}]^\top$ in the standard orthonormal basis $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_D\}$

x_{nd} is the coordinate of $\boldsymbol{x}_n$ along the direction $\boldsymbol{e}_d$

$$\boldsymbol{x}_n = \sum_{d=1}^{D} x_{nd}\boldsymbol{e}_d$$

$\boldsymbol{e}_d$ is a vector of all zeros except a single 1 at the d^{th} position. Also, $\boldsymbol{e}_d{}^\top \boldsymbol{e}_{d'} = 0$ for $d \neq d'$

- Let's represent the same data point in a new orthonormal basis $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_D\}$

z_{nd} is the projection/coordinate of $\boldsymbol{x}_n$ along the direction $\boldsymbol{w}_d$ since $z_{nd} = \boldsymbol{w}_d^\top \boldsymbol{x}_n = \boldsymbol{x}_n^\top \boldsymbol{w}_d$ (verify)

$$\boldsymbol{x}_n = \sum_{d=1}^{D} z_{nd}\boldsymbol{w}_d$$

$\boldsymbol{z}_n = [z_{n1}, z_{n2}, \dots, z_{nD}]^\top$ denotes the co-ordinates of $\boldsymbol{x}_n$ in the new basis

- Ignoring directions along which projection z_{nd} is small, we can approximate $\boldsymbol{x}_n$ as

$$\boldsymbol{x}_n \approx \hat{\boldsymbol{x}}_n = \sum_{d=1}^{K} z_{nd}\boldsymbol{w}_d = \sum_{d=1}^{K} (\boldsymbol{x}_n^\top \boldsymbol{w}_d)\boldsymbol{w}_d = \sum_{d=1}^{K} (\boldsymbol{w}_d \boldsymbol{w}_d^\top)\boldsymbol{x}_n$$

Note that $\left\|\boldsymbol{x}_n - \sum_{d=1}^{K} (\boldsymbol{w}_d \boldsymbol{w}_d^\top)\boldsymbol{x}_n\right\|^2$ is the reconstruction error on $\boldsymbol{x}_n$. Would like it to minimize w.r.t. $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K$

- Now $\boldsymbol{x}_n$ is represented by $K < D$ dim. rep. $\boldsymbol{z}_n = [z_{n1}, z_{n2}, \dots, z_{nK}]$ and

Also, $\boldsymbol{x}_n \approx \mathbf{W}_K \boldsymbol{z}_n$

$$\boldsymbol{z}_n = \boldsymbol{W}_K^\top \boldsymbol{x}_n$$

$\boldsymbol{W}_K = [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_K]$ is the "projection matrix" of size $D \times K$

PCA Minimizes Reconstruction Error

- We plan to use only K directions $[\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$ so would like them to be such that the total reconstruction error is minimized

$$\mathcal{L}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} \|\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n\|^2 = \sum_{n=1}^{N} \left\| \boldsymbol{x}_n - \sum_{d=1}^{K} (\boldsymbol{w}_d \boldsymbol{w}_d^\top) \boldsymbol{x}_n \right\|^2$$

Constant; doesn't depend on the $\boldsymbol{w}_d$'s

Variance along $\boldsymbol{w}_d$

$$= C - \sum_{d=1}^{K} \boldsymbol{w}_d^\top \mathbf{S} \boldsymbol{w}_d \text{ (verify)}$$

- Each optimal $\boldsymbol{w}_d$ can be found by solving

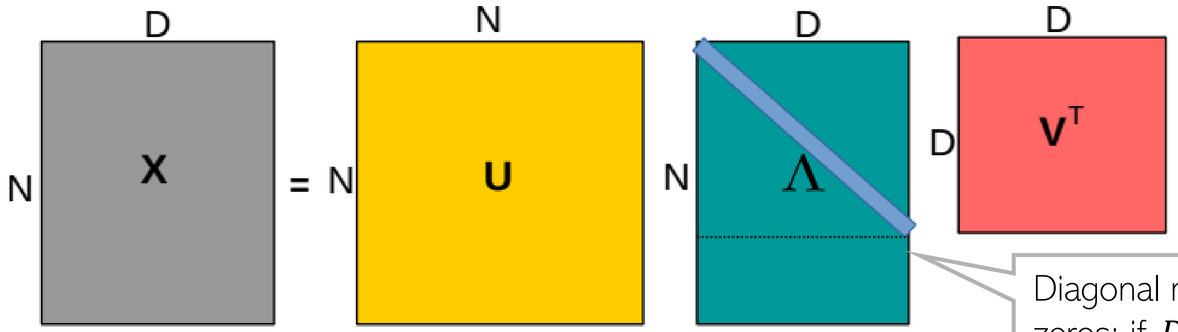
$$\underset{\boldsymbol{w}_d}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = \underset{\boldsymbol{w}_d}{\operatorname{argmax}}\ \ \boldsymbol{w}_d^\top \mathbf{S} \boldsymbol{w}_d$$

Subject to $\boldsymbol{w}_d^\top \boldsymbol{w}_d = 1$

- Thus minimizing the reconstruction error is equivalent to maximizing variance
- The K directions can be found by solving the eigendecomposition of $\mathbf{S}$

Singular Value Decomposition (SVD)

- Any matrix $\mathbf{X}$ of size $N \times D$ can be represented as the following decomposition



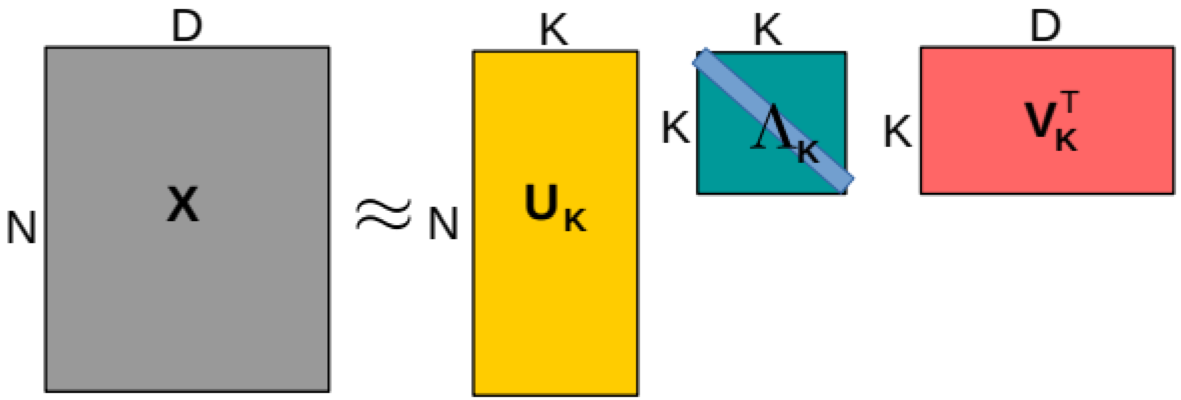
$$\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top = \sum_{k=1}^{\min\{N,D\}} \lambda_k \boldsymbol{u}_k \boldsymbol{v}_k^\top$$

Diagonal matrix. If $N > D$, last $D - N$ rows are all zeros; if $D > N$, last $D - N$ columns are all zeros

- $\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N]$ is $N \times N$ matrix of left singular vectors, each $\boldsymbol{u}_n \in \mathbb{R}^N$
  - $\mathbf{U}$ is also orthonormal ($\boldsymbol{u}_n^\top \boldsymbol{u}_n = 1 \; \forall n$ and $\boldsymbol{u}_n^\top \boldsymbol{u}_{n'} = 0 \; \forall n \neq n'$)
- $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N]$ is $D \times D$ matrix of right singular vectors, each $\boldsymbol{v}_d \in \mathbb{R}^D$
  - $\mathbf{V}$ is also orthonormal($\boldsymbol{v}_d^\top \boldsymbol{v}_d = 1 \; \forall d$ and $\boldsymbol{v}_d^\top \boldsymbol{v}_{d'} = 0 \; \forall d \neq d'$)
- Λ is $N \times D$ with only $\min(N, D)$ diagonal entries - singular values
- Note: If $\mathbf{X}$ is symmetric then it is known as eigenvalue decomposition ($\mathbf{U} = \mathbf{V}$)

Low-Rank Approximation via SVD

- If we just use the top $K < \min\{N, D\}$ singular values, we get a rank-K SVD



This is a rank-1 matrix

$$\mathbf{X} \approx \hat{\mathbf{X}} = \sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k \boldsymbol{v}_k^\top = \mathbf{U}_K \Lambda_K \mathbf{V}_K^\top$$

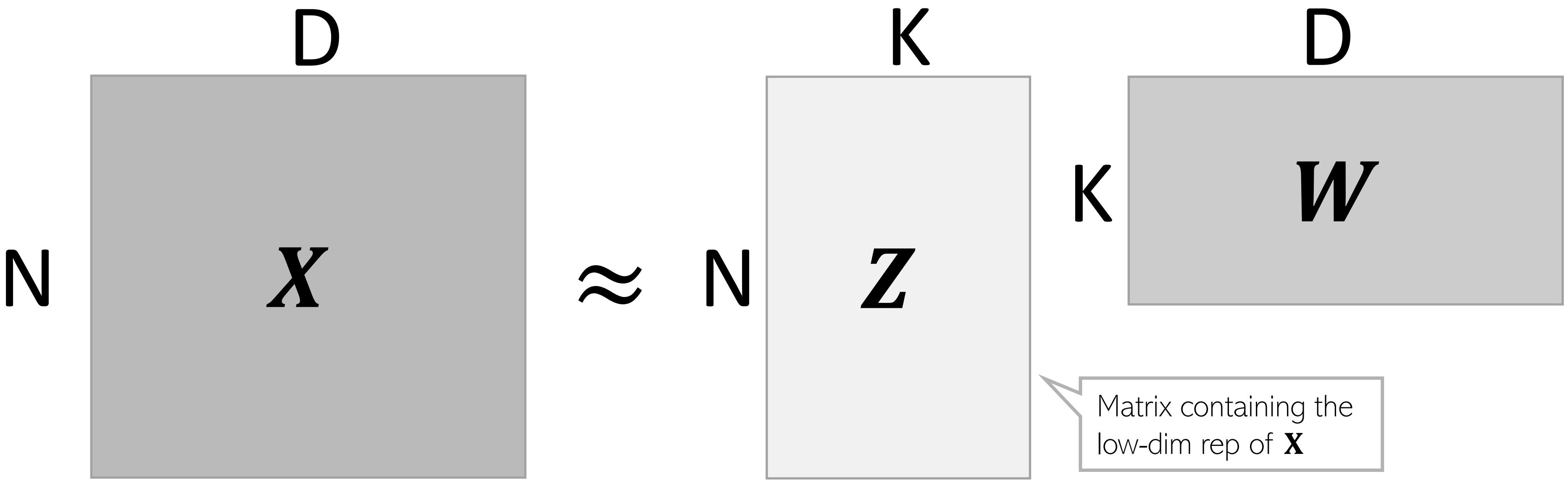
- Above SVD approx. can be shown to minimize the reconstruction error $\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\|$
  - Fact: SVD gives the best rank-K approximation of a matrix

- PCA is done by doing SVD on the covariance matrix $\boldsymbol{S}$ (left and right singular vectors are the same and become eigenvectors, singular values become eigenvalues)

Dimensionality Reduction: Beyond PCA

Dim-Red as Matrix Factorization

- If we don't care about the orthonormality constraints on $\boldsymbol{W}$, then dim-red can also be achieved by solving a matrix factorization problem on the data matrix $\mathbf{X}$



Matrix containing the low-dim rep of $\mathbf{X}$

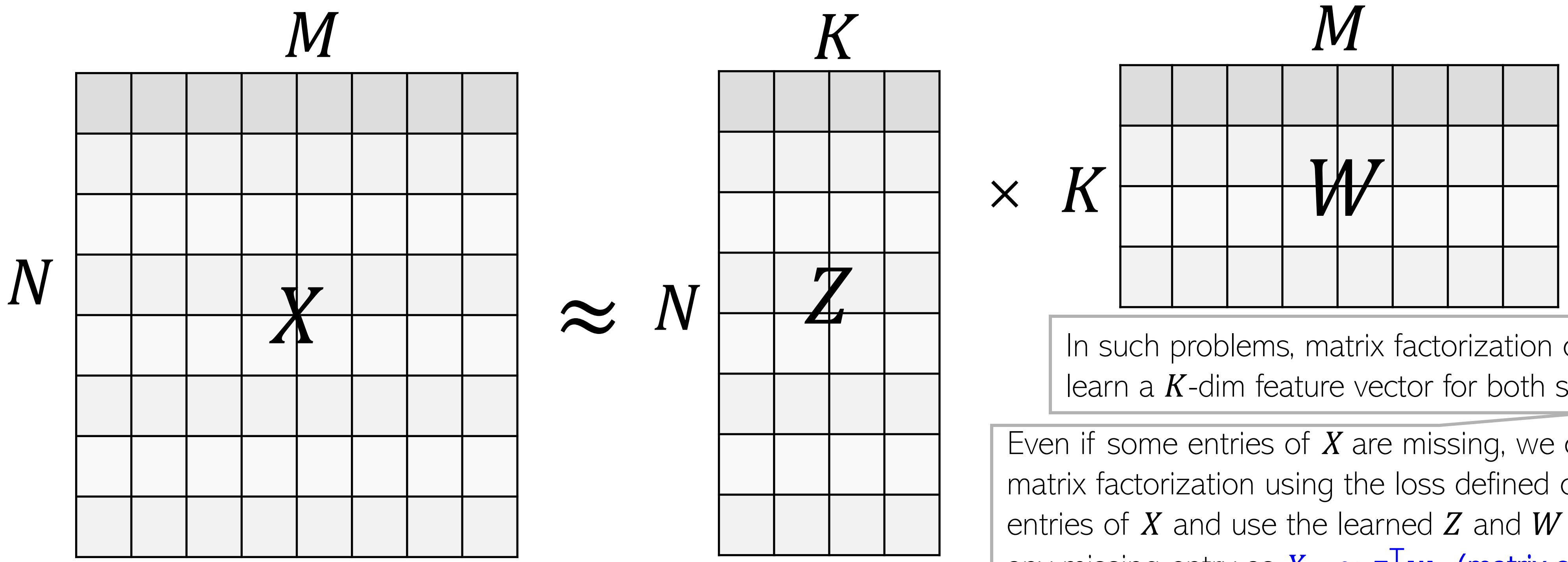
$$\{\hat{\boldsymbol{Z}}, \widehat{\boldsymbol{W}}\} = \operatorname{argmin}_{\boldsymbol{Z},\boldsymbol{W}} ||\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{W}||^2$$

If $K < \min\{D, N\}$, such a factorization gives a low-rank approximation of the data matrix $\boldsymbol{X}$

- Can solve such problems using ALT-OPT
- Can impose various constraints on $\mathbf{Z}$ and $\mathbf{W}$ (e.g., sparsity, non-negativity, etc)

Matrix Factorization is a very useful method!

- In many problems, we are given co-occurrence data in form of an $N \times M$ matrix X
- Data consists of relationship b/w two sets of entities containing N and M entities
- Each entry X_{ij} denotes how many times the pair (i, j) co-occurs, e.g.,
  - Number of times a document i (total N docs) contains word j of a vocabulary (total M words)
  - Rating user i gave to item (or movie) j on a shopping (or movie streaming) website

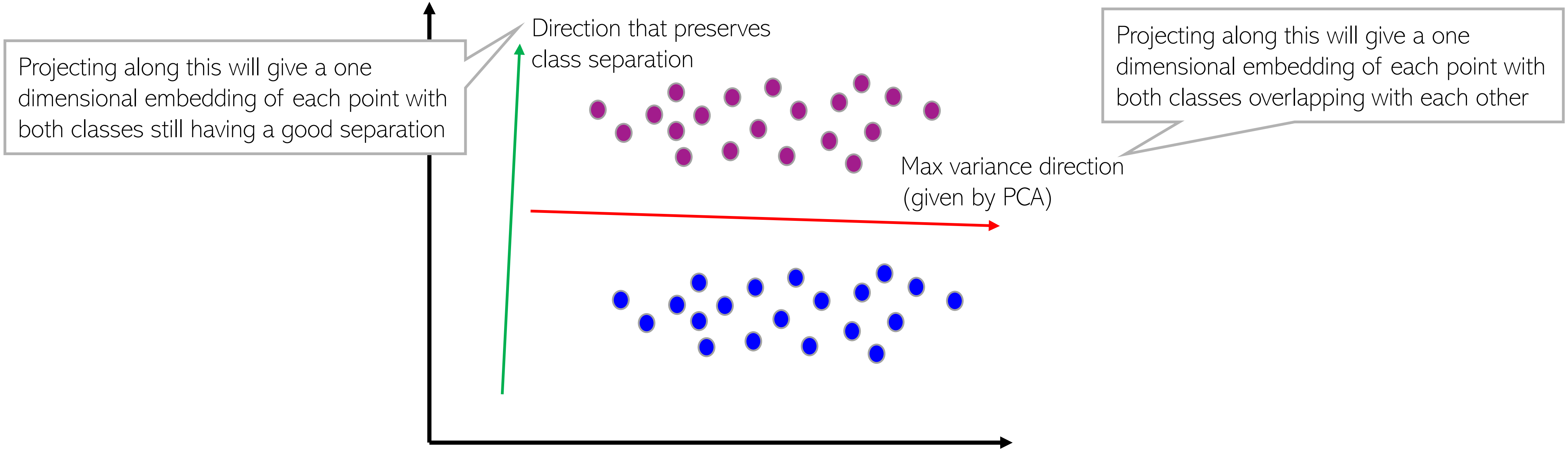


In such problems, matrix factorization can be used to learn a K-dim feature vector for both set of entities

Even if some entries of X are missing, we can still do matrix factorization using the loss defined on the given entries of X and use the learned Z and W to predict any missing entry as $X_{ij} \approx \boldsymbol{z}_i^\top \boldsymbol{w}_j$ (matrix completion)

Supervised Dimensionality Reduction

- Maximum variance directions may not be aligned with class separation directions (focusing only on variance/reconstruction error of the inputs x_n, is not always ideal)



- Be careful when using methods like PCA for supervised learning problems
- A better option would be to find projection directions such that after projection
  - Points within the same class are close (low intra-class variance)
  - Points from different classes are well separated (the class means are far apart)

Dim. Reduction by Preserving Pairwise Distances

- PCA/SVD etc assume we are given points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ as vectors (e.g., in D dim)
- Often the data is given in form of distances d_{ij} between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ $(i, j = 1,2, \ldots, N)$
- Would like to project data such that pairwise distances between points are preserved

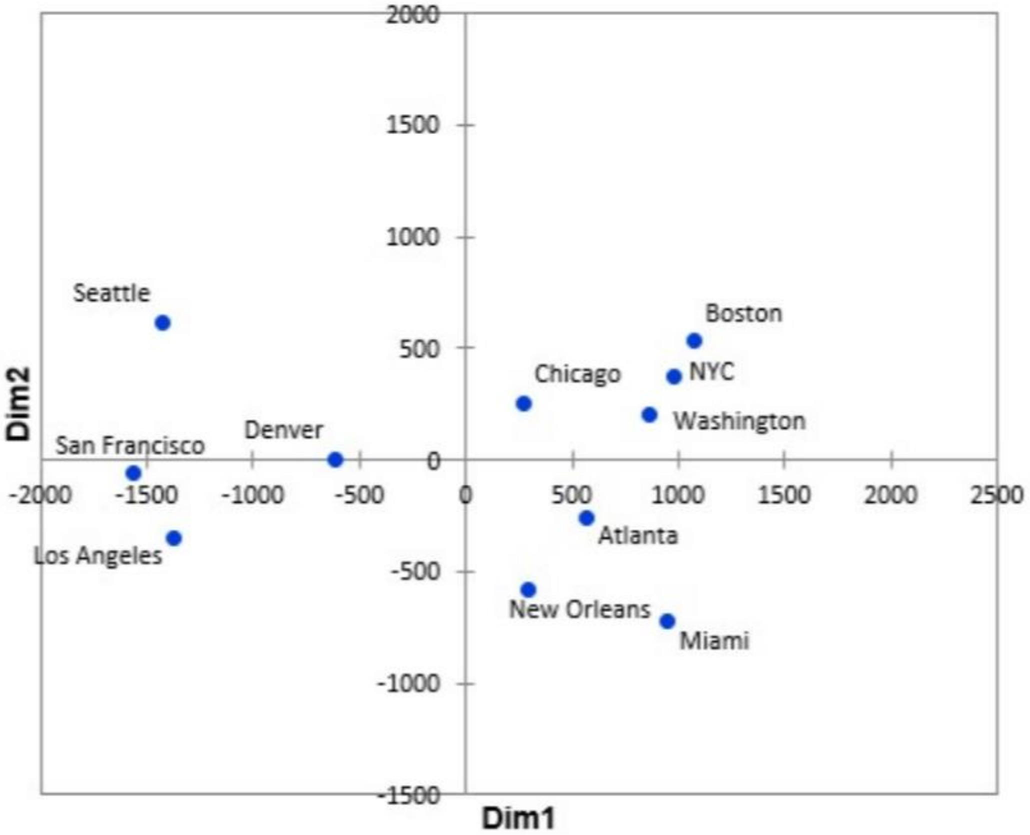
$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{Z}) = \arg\min_{\mathbf{Z}} \sum_{i,j=1}^{N} (d_{ij} - ||\mathbf{z}_i - \mathbf{z}_j||)^2$$

$\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ denote low-dim embeddings/projections of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively

- Basically, if d_{ij} is large (resp. small), would like $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|$ to be large (resp. small)
- Multi-dimensional Scaling (MDS) is one such algorithm
- Note: If d_{ij} is the Euclidean distance, MDS is equivalent to PCA

MDS: An Example

- Result of applying MDS (with $K = 2$) on pairwise distances between some US cities

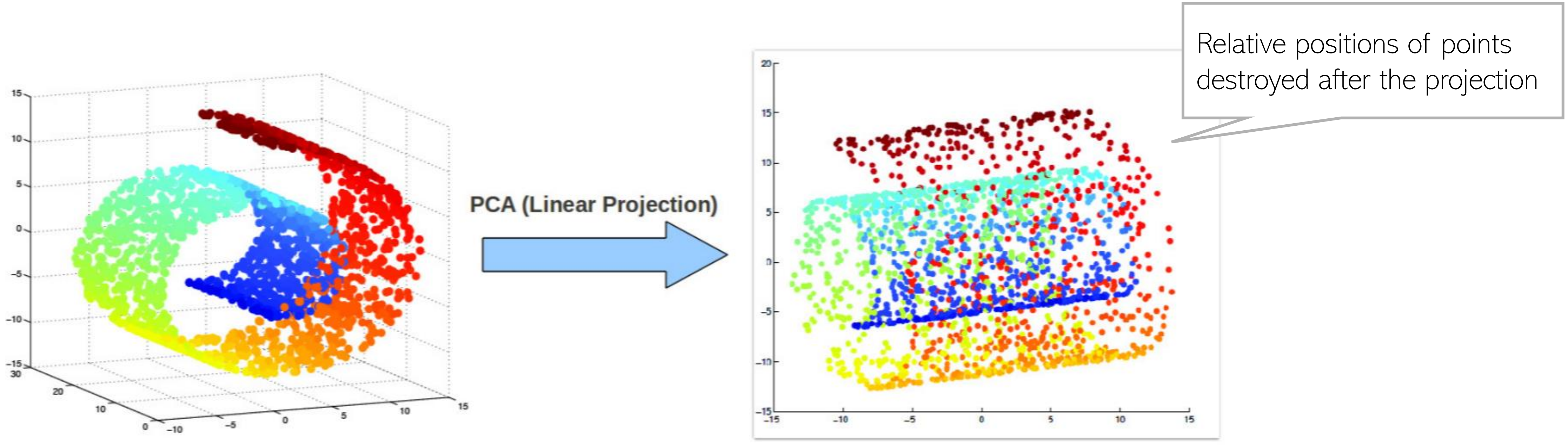


- Here MDS produces 2D embedding of each city such that geographically close cities are also close in 2D embedding space

Nonlinear Dimensionality Reduction

Beyond Linear Projections

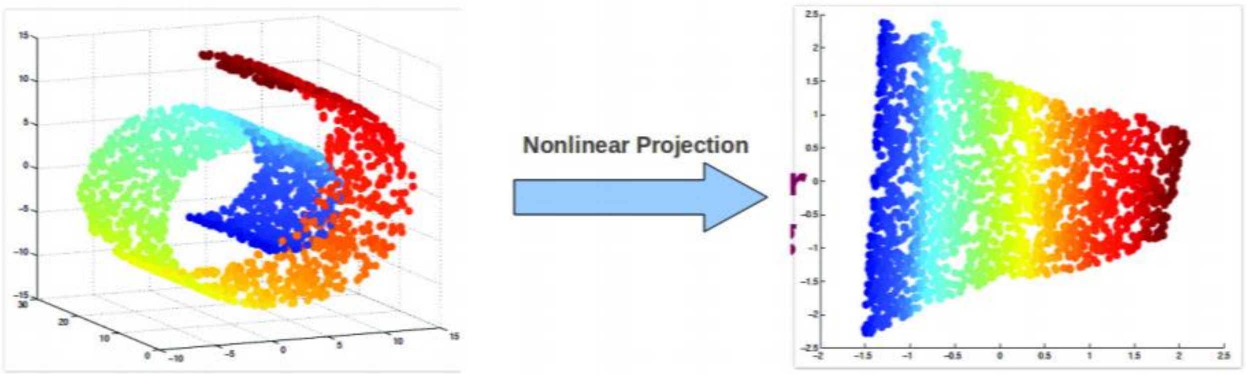
- Consider the swiss-roll dataset (points lying close to a manifold)



- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities
  - Maximum variance directions may not be the most interesting ones

Nonlinear Dimensionality Reduction

- We want to a learn nonlinear low-dim projection



Relative positions of points preserved after the projection

- Some ways of doing this
  - Nonlinearize a linear dimensionality reduction method. E.g.:
    - Cluster data and apply linear PCA within each cluster (mixture of PCA)
    - Kernel PCA (nonlinear PCA)
  - Using manifold based methods that intrinsically preserve nonlinear geometry, e.g.,
    - Locally Linear Embedding (LLE), Isomap
    - Maximum Variance Unfolding
    - Laplacian Eigenmap, and others such as SNE/tSNE, etc.
- .. or use unsupervised deep learning techniques (later)

Will look at KPCA, LLE, SNE/tSNE

Kernel PCA

- Recall PCA: Given N observations $\boldsymbol{x}_n \in \mathbb{R}^D$, $n = 1,2,\dots,N$,

$D \times D$ cov matrix assuming centered data

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top \qquad \mathbf{S}\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i \;\forall i = 1,\dots,D$$

D eigenvectors of $\mathbf{S}$

- Assume a kernel k with associated M dimensional nonlinear map ϕ

$M \times M$ cov matrix assuming centered data in the kernel-induced feature space

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^\top \qquad \mathbf{C}\mathbf{v}_i = \lambda_i\mathbf{v}_i \;\forall i = 1,\dots,M$$

M eigenvectors of $\mathbf{C}$

- Would like to do it without computing $\mathbf{C}$ and the mappings $\phi(\boldsymbol{x}_n)'s$ since M can be very large (even infinite, e.g., when using an RBF kernel)
- Boils down to doing eigendecomposition of the $N \times N$ kernel matrix $\mathbf{K}$ (PRML 12.3)
  - Can verify that each $\boldsymbol{v}_i$ above can be written as a lin-comb of the inputs: $\boldsymbol{v}_i = \sum_{n=1}^{N} a_{in}\phi(\boldsymbol{x}_n)$
  - Can show that finding $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \dots, a_{iN}]$ reduces to solving an eigendecomposition of $\mathbf{K}$
  - Note: Due to req. of centering, we work with a centered kernel matrix $\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$

$N \times N$ matrix of all 1s

Locally Linear Embedding

Several non-lin dim-red algos use this idea

Essentially, neighbourhood preservation, but only local

- Basic idea: If two points are local neighbors in the original space then they should be local neighbors in the projected space too
- Given N observations $\boldsymbol{x}_n \in \mathbb{R}^D$, $n = 1,2, \dots, N$, LLE is formulated as

Solve this to learn weights W_{ij} such that each point x_i can be written as a weighted combination of its local neighbors in the original feature space

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j \in \mathcal{N}(i)} W_{ij}\mathbf{x}_j||^2$$

$\mathcal{N}(i)$ denotes the local neighbors (a predefined number, say K, of them) of point $\boldsymbol{x}_i$

- For each point $\boldsymbol{x}_n \in \mathbb{R}^D$, LLE learns $\boldsymbol{z}_n \in \mathbb{R}^K$, $n = 1,2, \dots, N$ such that the same neighborhood structure exists in low-dim space too

Requires solving an eigenvalue problem

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\mathbf{z}_i - \sum_{j \in \mathcal{N}(i)} W_{ij}\mathbf{z}_j||^2$$

- Basically, if point $\boldsymbol{x}_i$ can be reconstructed from its neighbors in the original space, the same weights W_{ij} should be able to reconstruct $\boldsymbol{z}_i$ in the new space too

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science 290 (2000)

SNE and t-SNE

Thus very useful if we want to visualize some high-dim data in two or three dims

- Also nonlin. dim-red methods, especially suited for projecting to 2D or 3D
- SNE stands for Stochastic Neighbor Embedding (Hinton and Roweis, 2002)
- Uses the idea of preserving probabilistically defined neighborhoods
- SNE, for each point $\boldsymbol{x}_i$, defines the probability of a point $\boldsymbol{x}_j$ being its neighbor as

Neighbor probability in the original space

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

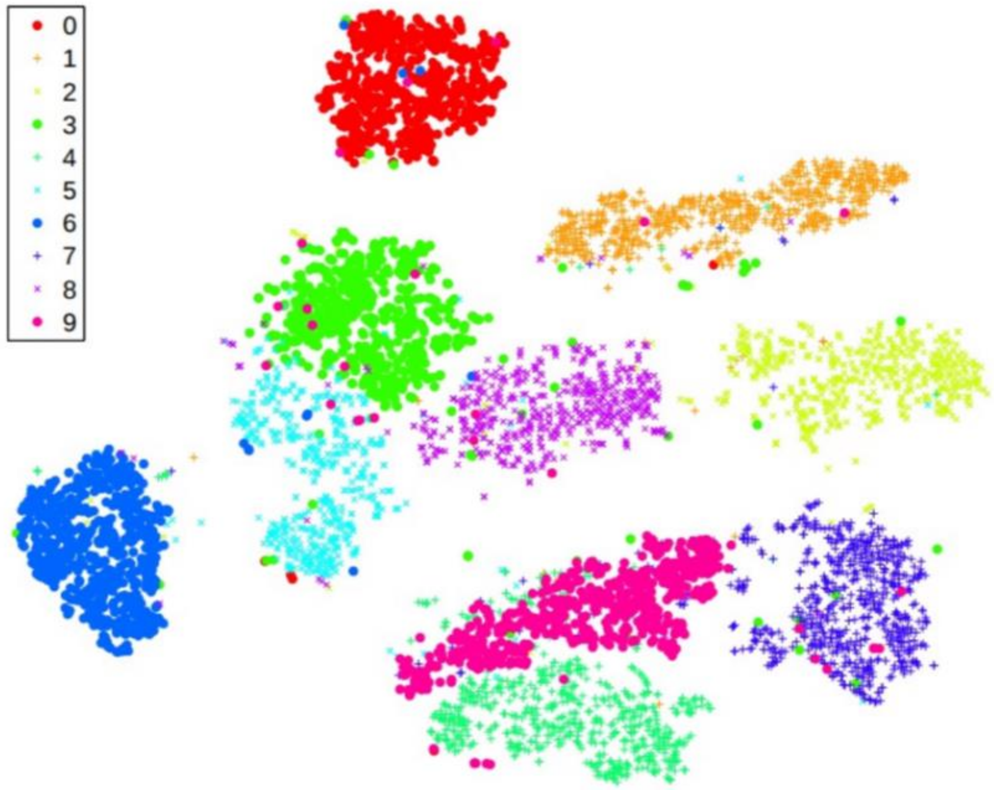
Neighbor probability in the projected/embedding space

$$q_{j|i} = \frac{\exp(-||\mathbf{z}_i - \mathbf{z}_j||^2/2\sigma^2)}{\sum_{k \neq i} \exp(-||\mathbf{z}_i - \mathbf{z}_k||^2/2\sigma^2)}$$

- SNE ensures that neighbourhood distributions in both spaces are as close as possible
  - This is ensured by minimizing their total mismatch (KL divergence) $\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
- t-SNE (van der Maaten and Hinton, 2008) offers a couple of improvements to SNE
  - Learns $\boldsymbol{z}_i$'s by minimizing symmetric KL divergence
  - Uses Student-t distribution instead of Gaussian for defining $q_{j|i}$

SNE and t-SNE

- Especially useful for visualizing data by projecting into 2D or 3D



Result of visualizing MNIST digits data in 2D (Figure from van der Maaten and Hinton, 2008)

Word Embeddings: Dim-Reduction for Words

Or sentences, paragraphs, documents, etc which are basically a set of words

- Feature representation/embeddings of words are very useful in many applications
- Naively we can a one-hot vector of size V for each word (where V is the vocab size)

1 only at the n-th location

Word n of the vocabulary

0	**0**	**0**	**0**	**0**	**0**	....	**0**	**1**	....	**0**	**0**	**0**	**0**

- One-hot representation of a word has two main issues
  - Very high dimensionality (V) for each word
  - One-hot vector does not capture word semantics (any pair of words will have zero similarity)

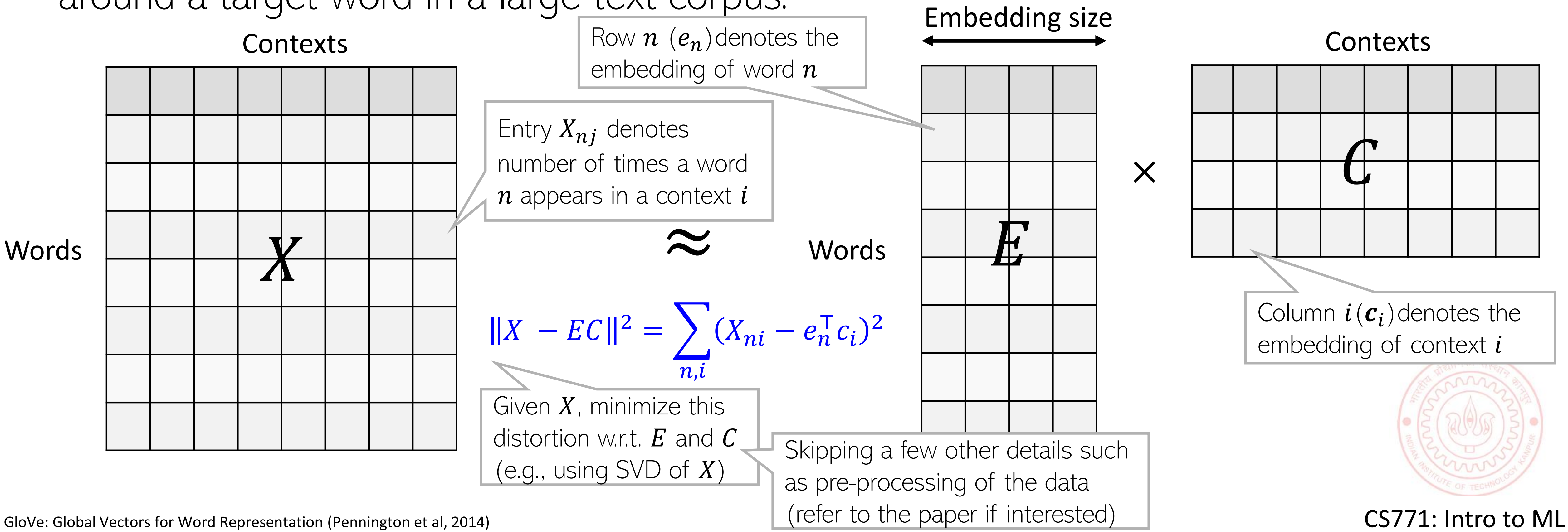
- Desirable: Learning low-dim word embeddings that capture the meaning/semantics

- We want embedding of each word $\boldsymbol{n}$ to be low-dimensional vector $\boldsymbol{e}_n \in \mathbb{R}^K$
  - If two words $\boldsymbol{n}$ and $\boldsymbol{n}'$ are semantically similar (dissimilar), we want $\boldsymbol{e}_n$ and $\boldsymbol{e}_{n'}$ to be close (far)

- Many methods to learn word embeddings (e.g., Glove and Word2Vec)

GloVe

- GloVe (Global Vectors for Word Representation) is a linear word embedding method
- Based on matrix factorization of a word-context co-occurrence matrix
- In GloVe, the context consists of words that co-occur within a specified context window around a target word in a large text corpus.



Word2Vec

- A deep neural network based nonlinear word embedding method
- Usually learned using one of the following two objectives
  - Skip-gram
  - Continuous bag of words (CBOW)
- Skip-gram: Probability of a context $\boldsymbol{i}$ occurring around a word $\boldsymbol{n}$

Conditional probability which can be estimated from training data

$$p(i|n) = \frac{\exp(\boldsymbol{c}_i^\top \boldsymbol{e}_n)}{\sum_i \exp(\boldsymbol{c}_i^\top \boldsymbol{e}_n)}$$

Embeddings are learned by optimizing a neural network based loss function which makes the difference b/w LHS and RHS small

- CBOW: Probability of word $\boldsymbol{n}$ occurring given a context window, e.g., $\boldsymbol{k}$ previous and $\boldsymbol{k}$ next words

Conditional probability which can be estimated from training data

Sum/average of the embeddings of the context window for word $\boldsymbol{n}$

$$p(n|n-k:n+k) = \frac{\exp(\boldsymbol{e}_n^\top \boldsymbol{c}_n)}{\sum_n \exp(\boldsymbol{e}_n^\top \boldsymbol{c}_n)}$$

Embeddings are learned by optimizing a neural network based loss function which makes the difference b/w LHS and RHS small

Efficient Estimation of Word Representations in Vector Space (Mikolov et al, 2013)