Maximum Variance Subspace

- Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_p$ denote the p principal components

$$\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0 \text{ for } i \neq j \text{ and } \boldsymbol{v}_i \cdot \boldsymbol{v}_j = 1 \text{ for } i = j$$

- Assume that data $\boldsymbol{X}$ is centered (mean = 0)
  - if not, we can preprocess it: $\widetilde{\boldsymbol{X}} = \boldsymbol{X} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{X}$
- Find vector that maximizes sample variance of projected data

$$\frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{v}^T\boldsymbol{x}_i)^2 = \boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}$$

$$\max_{\boldsymbol{v}} \boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} \quad s.t. \quad \boldsymbol{v}^T\boldsymbol{v} = 1$$

✓

$$= \boxed{\frac{1}{n-1} v^T X^T X v}$$

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

- Lagrangian: $\mathrm{L}(\boldsymbol{v}, \lambda) = \frac{1}{n-1}\boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} - \lambda(\boldsymbol{v}^T\boldsymbol{v} - 1)$

$$\frac{\partial L}{\partial \boldsymbol{v}} = 0 \qquad \boxed{\left(\frac{1}{n-1}\boldsymbol{X}^T\boldsymbol{X} - \lambda\boldsymbol{I}\right)\boldsymbol{v} = 0}$$

Eigen Problem

$$\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_p] \in \mathbb{R}^{d\times p}, \boldsymbol{Y} = \boldsymbol{XV}$$

$$\mathbf{y_i} = \mathbf{V^T x_i} = \sum_{j=1}^{p} \boldsymbol{v}_j^T \boldsymbol{x}_i$$

Eigen Decomposition

- Eigen-decomposition

$$\max_{v_1,\cdots,v_p} trace(\boldsymbol{V}^T \boldsymbol{C}\boldsymbol{V}).s.t.\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I} \rightarrow max_V\ trace\left(\boldsymbol{V}^T\boldsymbol{C}\boldsymbol{V} - \Lambda(\boldsymbol{V}^T\boldsymbol{V} - \boldsymbol{I})\right)$$

The sum of variance after projection

  - ○ $\frac{1}{n-1}\sum_{i=1}^{n} y_i^T y_i = \frac{1}{n-1}\sum_{i=1}^{n}\sum_{j=1}^{p}(v_j^T x_i)^2 = \frac{1}{n-1} trace(\boldsymbol{V}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{V}),\ \ y_i = (v_1^T x_i, \dots, v_p^T x_i)$
  - ○ $cov(\boldsymbol{Y},\boldsymbol{Y}) = \frac{1}{n-1}\boldsymbol{Y}^T\boldsymbol{Y} = \frac{1}{n-1}\boldsymbol{V}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{V} = \boldsymbol{V}^T\mathbf{V}\Lambda\mathbf{V}^T\mathbf{V} = \Lambda$ Statistically uncorrelated

Eigen Decomposition

- In $\boldsymbol{Cv} = \lambda\boldsymbol{v}$, $\boldsymbol{v}$ (the first PC) is the eigenvector of sample covariance matrix $\mathrm{C} = \frac{1}{\mathrm{n}-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$
  - Sample variance of projection becomes $\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{Cv} = \lambda\boldsymbol{v}^{T}\boldsymbol{v} = \lambda$
- For the eigenvectors and values for the sample variance matrix $\boldsymbol{C}$,
  - Eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots$
  - The 1st PC $\boldsymbol{v_1}$ is the eigenvector of the sample covariance matrix $\boldsymbol{C}$ associated with the largest eigenvalue
  - The 2nd PC $\boldsymbol{v_2}$ is the eigenvector of the sample covariance matrix $\boldsymbol{C}$ associated with the second largest eigenvalue
  - $\underline{\sum_j var(\boldsymbol{y_j})} = trace\big(\boldsymbol{cov}(\boldsymbol{Y},\boldsymbol{Y})\big) = \underline{\lambda_1 + \lambda_2 + \cdots}$
    - $\boldsymbol{cov}(\boldsymbol{Y},\boldsymbol{Y}) = \frac{\boldsymbol{1}}{\boldsymbol{n}-\boldsymbol{1}}\boldsymbol{Y}^{\boldsymbol{T}}\boldsymbol{Y} = \frac{\boldsymbol{1}}{\boldsymbol{n}-\boldsymbol{1}}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{V} = \boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{V}\boldsymbol{\Lambda}\mathbf{V}^{\mathbf{T}}\mathbf{V} = \boldsymbol{\Lambda}$

$$\underline{trace}\big(\overset{P}{\boxed{cov(Y,Y)}}\big) = \lambda_1 + \cdots + \lambda_p$$

$\lambda_i = \frac{\sigma_i^2}{n-1}$

Singular Value Decomposition

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

$\frac{1}{n-1} X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$

- For an n × d matrix $\boldsymbol{X}$ of rank r there exists a factorization (Singular Value Decomposition = SVD) as follows:

$$\boldsymbol{X} = \boldsymbol{U\Sigma V^T}$$

  - The columns of $\boldsymbol{U}$ are orthogonal eigenvectors of $\boldsymbol{XX^T}$
  - The columns of $\boldsymbol{V}$ are orthogonal eigenvectors of $\boldsymbol{X^TX}$
  - Eigenvalues $\mu_1, \dots, \mu_r$ of $\boldsymbol{XX^T}$ are the eigenvalues of $\boldsymbol{X^TX}$ $(rank(\boldsymbol{X}) = r)$
  - Singular values $\sigma_i = \sqrt{\mu_i}$ where $\boldsymbol{\Sigma} = diag(\sigma_1, \dots, \sigma_r)$
  - For eigenvalue λ_i of $\boldsymbol{C} = \frac{1}{n-1}\boldsymbol{X^TX}$, $\lambda_i = \frac{\sigma_i^2}{n-1}$

$\sigma_1 > \sigma_2 > \dots > \sigma_r$.

- Minimum Reconstruction Error
  - Find the subspace that yields minimum MSE reconstruction
    - MSE: $\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{VV^Tx}_i\|^2$ s.t. $\boldsymbol{V^TV} = \boldsymbol{I}$
  - Low rank approximation
    - $\min\limits_{A:rank(A)=p} \|\boldsymbol{X} - \boldsymbol{A}\|_F = \sigma_{p+1}$

$V V^T x_i$

v_2

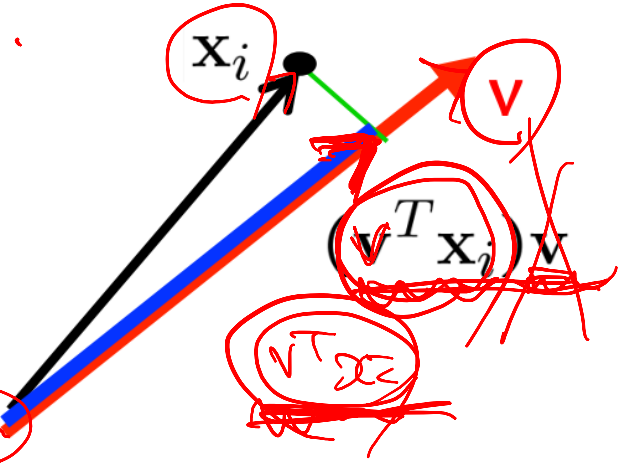
$v_1^T x_i$

$(v_2^T x_i) v_2 + (v_1^T x_i) v_1$

blue² + green² = black²

black² is fixed (it's just the data)

So, maximizing blue² is equivalent to minimizing green²



Dimensionality Reduction using PCA

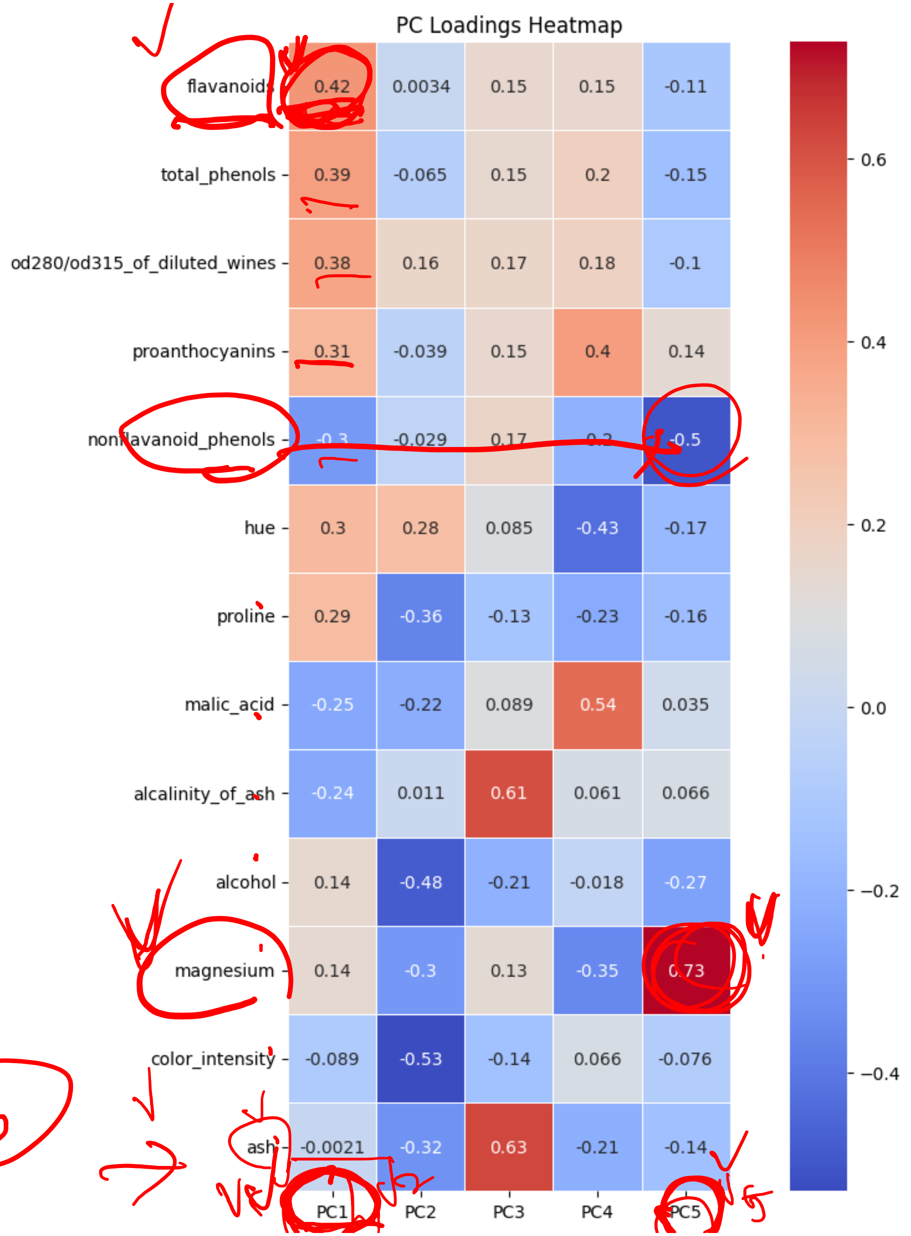
- Original representation
  - Data point: $\boldsymbol{x}_i = (x_i^1, \dots, x_i^d)$ ✓
- Transformed representation
  - Principal components (PCs): a normalized linear combination of the original features $(\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_p)$
  - PC loadings: how much each original variable contributes to that principal component
    - Loading for the i-th PC are given by the column of V $(\boldsymbol{v}_i)$
    - The j-th variable on the i-th PC is v_{ij}
  - PC score (Projection): $\boldsymbol{V}^T \boldsymbol{x}_i = (\boldsymbol{v}_1^T \boldsymbol{x}_i, \dots, \boldsymbol{v}_p^T \boldsymbol{x}_i)$
- Ratio of Explained variance: $\sum_{i=1}^{p} \lambda_i / trace(\boldsymbol{C})$

λ_i λ_p

V_1 V_p

$$0 \leq \frac{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5}{trace(C)} \leq 1$$

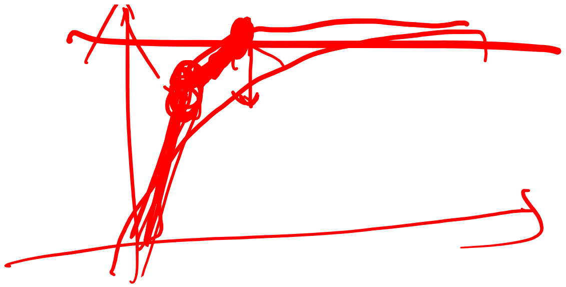
13



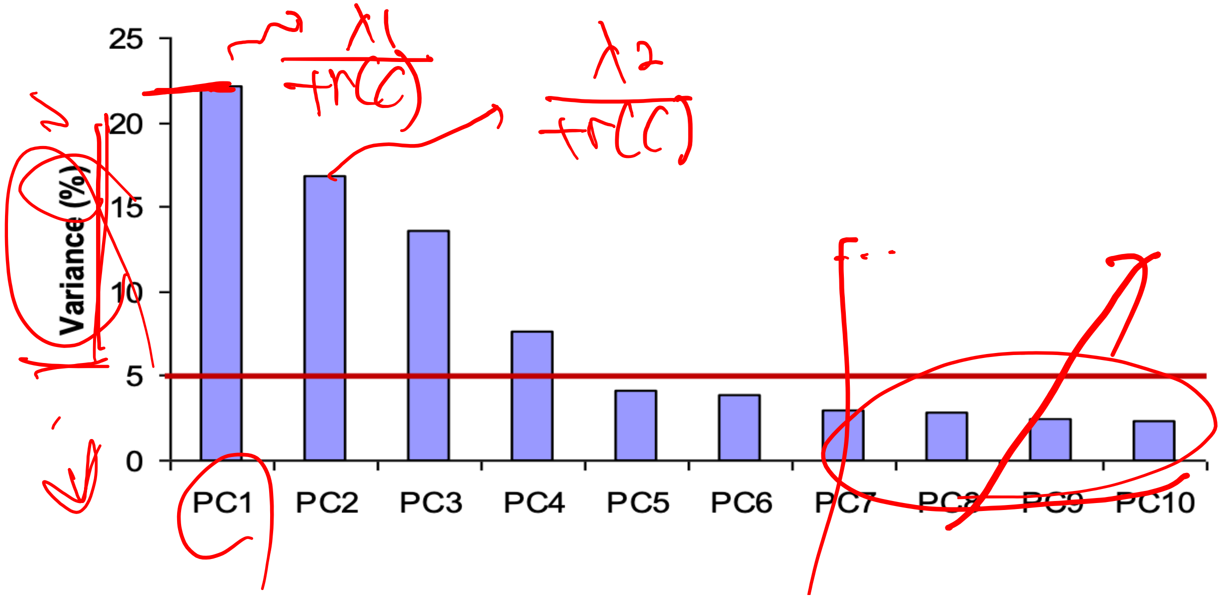
PC Loadings Heatmap

	PC1	PC2	PC3	PC4	PC5
flavanoids	0.42	0.0034	0.15	0.15	-0.11
total_phenols	0.39	-0.065	0.15	0.2	-0.15
od280/od315_of_diluted_wines	0.38	0.16	0.17	0.18	-0.1
proanthocyanins	0.31	-0.039	0.15	0.4	0.14
nonflavanoid_phenols	-0.3	-0.029	0.17	-0.2	-0.5
hue	0.3	0.28	0.085	-0.43	-0.17
proline	0.29	-0.36	-0.13	-0.23	-0.16
malic_acid	-0.25	-0.22	0.089	0.54	0.035
alcalinity_of_ash	-0.24	0.011	0.61	0.061	0.066
alcohol	0.14	-0.48	-0.21	-0.018	-0.27
magnesium	0.14	-0.3	0.13	-0.35	0.73
color_intensity	-0.089	-0.53	-0.14	0.066	-0.076
ash	-0.0021	-0.32	0.63	-0.21	-0.14

Dimensionality Reduction using PCA

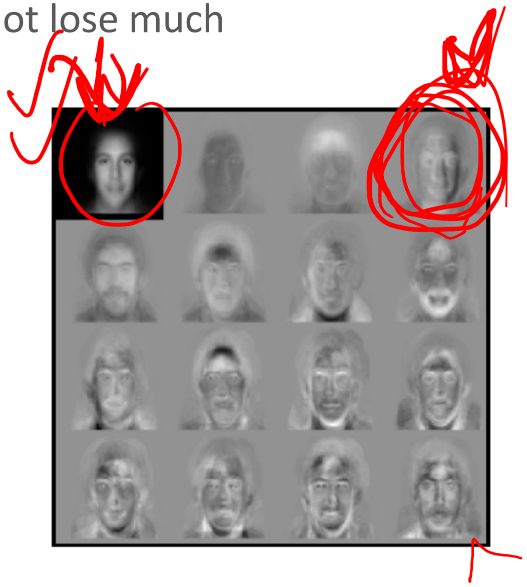


- In high-dimensional problems, data sometimes lies near a linear subspace, as noise introduces small variability
- Only keep data projections onto principal components with large eigenvalues
  - Can ignore the components of smaller significance
  - Might lose some info, but if eigenvalues are small, do not lose much



$$\frac{\lambda_1}{tr(C)}$$

$$\frac{\lambda_2}{tr(C)}$$



Eigenfaces from 7562 images:

top left image is linear combination of rest.

Sirovich & Kirby (1987)
Turk & Pentland (1991)

PCA wrap-up

- Strengths
  - Eigenvector method
  - No tuning of the parameters
  - No local optima

- Weaknesses ✓
  - Limited to ~~second order statistics~~ ✓
  - Limited to linear projections ✓
    - If the structure of data is non-linear, the transformed representation can not capture the original data structure and can lose the important information.

PCA vs LDA

	PCA	LDA
Objective	Maximize variance in the dataset	Maximize separability between known classes
Data Requirements	Feature matrix only	Feature matrix and corresponding class labels
Number of Components	Up to the number of original features (d)`	At most C−1 where C is the number of classes (min(C-1, d))
Usage	Dimensionality reduction, visualization, noise reduction, feature extraction	Classification and dimensionality reduction for classification
Optimization Criteria	Maximizes total variance	Maximizes ratio of between-class variance to within-class variance

Kernel PCA

- Non-linear feature mapping
  - PCA aims to find the subspace that consists of eigenvectors with largest eigenvalues of covariance matrix
- Covariance matrix for non-linear feature mapping
  - Consider a non-linear feature mapping $\boldsymbol{x} \rightarrow \Phi(\boldsymbol{x})$

$$C_\Phi V = \lambda V$$

$$\boldsymbol{C}_\Phi = \frac{1}{n-1}\sum_{i=1}^{n} \Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_i)^T$$

  - Eigenvectors of $\boldsymbol{C}_\Phi$ are linear combinations of the feature vectors $\Phi(\boldsymbol{x}_1), \dots, \Phi(\boldsymbol{x}_n)$

$$\lambda \boldsymbol{v} = \boldsymbol{C}_\Phi \boldsymbol{v} = \frac{1}{n-1}\sum_{i=1}^{n} \Phi(\boldsymbol{x}_i)\Phi(\boldsymbol{x}_i)^T \boldsymbol{v} = \frac{1}{n-1}\sum_{i=1}^{n} \langle \Phi(\boldsymbol{x}_i), \boldsymbol{v}\rangle\, \Phi(\boldsymbol{x}_i) = \lambda \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)$$

$$C_\Phi$$

  - $\alpha_i = \frac{1}{\lambda(n-1)}\langle \Phi(\boldsymbol{x}_i), v\rangle$

$$V = \sum_i \alpha_i \Phi(x_i)$$

$\langle \Phi(x_k), \Phi(x_i) \rangle = K(x_k, x_i)$

Kernel PCA

$$\alpha_i = \frac{1}{\lambda(n-1)}\langle \Phi(\boldsymbol{x}_i), \boldsymbol{v}\rangle, \ \boldsymbol{v} = \textstyle\sum_i \alpha_i \Phi(\boldsymbol{x_i})$$

✓

$\langle \Phi(x_i), v \rangle = \alpha_i \lambda(n-1)$

$\tilde{1}_n = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$

$\langle a, b+c \rangle = \langle a, b \rangle + \langle a, c \rangle$

- Kernel PCA
  - Suppose that $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ and kernel matrix $\boldsymbol{K}$ where $K_{ij} = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)\rangle$

$\forall k$

$$\sum_{i=1}^{n} \underbrace{\langle \Phi(\boldsymbol{x}_k), \Phi(\boldsymbol{x}_i)\rangle}_{K(x_k, x_i)} \alpha_i = \left\langle \Phi(\boldsymbol{x}_k), \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i)\right\rangle = \langle \Phi(\boldsymbol{x}_k), \boldsymbol{v}\rangle = (n-1)\lambda\alpha_k$$

$$\begin{bmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = (n-1) \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

k-th row

  - Thus, we need to solve the kernel eigenvalue problem

K_{x_i}

$$\boldsymbol{K\alpha} = (n-1)\lambda\boldsymbol{\alpha}$$

C_Φ

  - Normalizing Eigenvectors $\langle \boldsymbol{v}, \boldsymbol{v}\rangle = 1$ leads to the condition: $\sum_{i,j} \alpha_i \alpha_j \Phi(x_i)^T \Phi(x_j) = \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha K \alpha} = (n-1)\lambda\langle \boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle = 1$

$\langle v, v \rangle$

$\|\alpha\| = \frac{1}{(n-1)\lambda}$

- For new test data $\boldsymbol{x}_{new}$

$$\boldsymbol{\beta} = \begin{bmatrix} - - - \boldsymbol{\alpha}_1^T - - - \\ \vdots \\ - - - \boldsymbol{\alpha}_p^T - - - \end{bmatrix} \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}^{(1)}) \\ k(\boldsymbol{x}, \boldsymbol{x}^{(2)}) \\ \vdots \\ k(\boldsymbol{x}, \boldsymbol{x}^{(N)}) \end{bmatrix}$$

PC score

$$\langle \boldsymbol{v}, \Phi(\boldsymbol{x}_{new})\rangle = \left\langle \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{new})\right\rangle = \sum_{i=1}^{n} \alpha_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_{new})\rangle = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_{new})$$

- How to obtain the centered feature map?

$K \rightarrow \tilde{K}$ $\quad K = XX^T \quad \Phi\Phi^T$

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} - \frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^T\boldsymbol{X} = \boldsymbol{X} - \tilde{\boldsymbol{1}}_n\boldsymbol{X}$$

$\tilde{K} = \tilde{X}\tilde{X}^T = (X - \tilde{1}_n X)(X - \tilde{1}_n X)^T$

$$\tilde{K} = K - \tilde{1}_n \boxed{K} - K\tilde{1}_n + \tilde{1}_n K \tilde{1}_n$$ ✓

$$K = XX^T$$

$$\Phi = \begin{bmatrix} -\phi(x_1)- \\ \vdots \\ -\phi(x_n)- \end{bmatrix} \in \mathbb{R}^{n\times \cancel{d}}$$

$$K = \Phi\Phi^T \in \mathbb{R}^{n\times n}$$

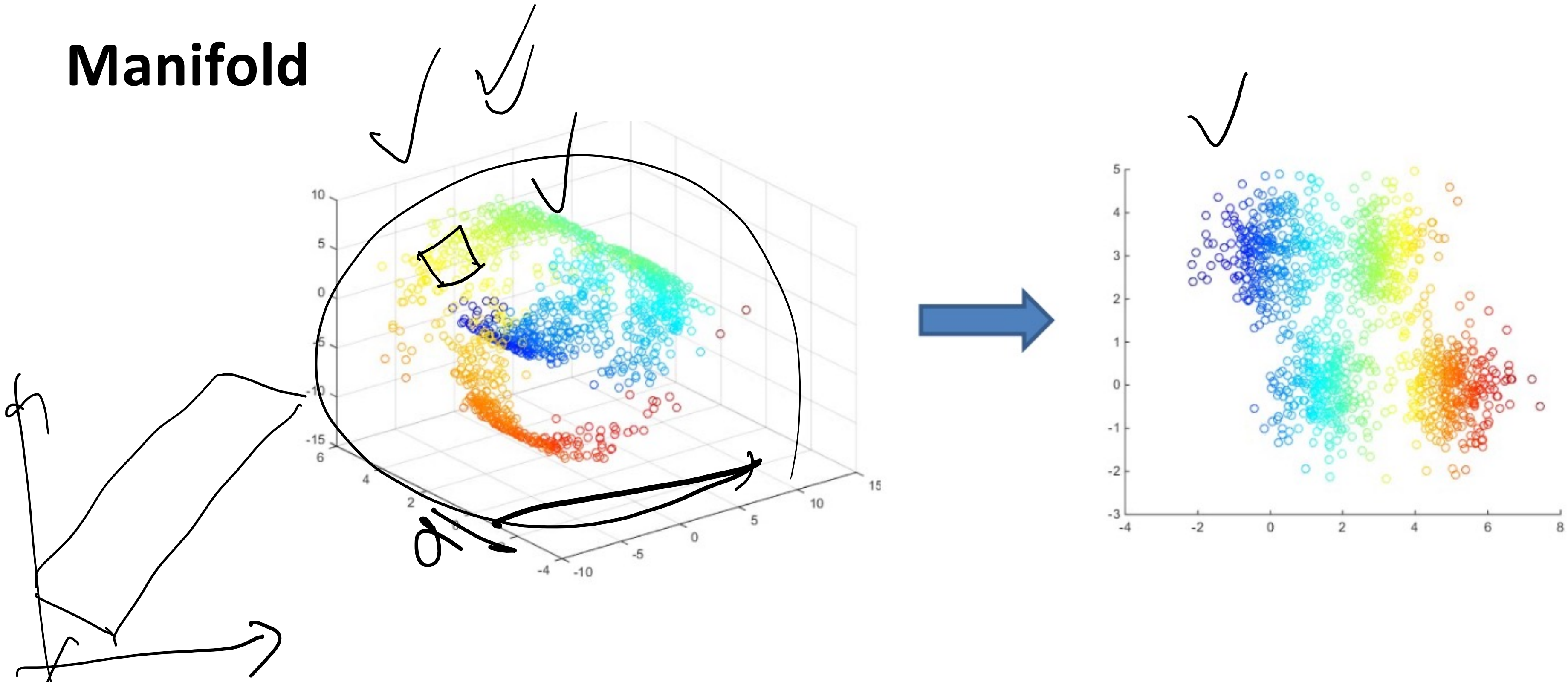
Manifold Learning

- Isomap
- Multi-dimensional Scaling
- Laplacian Eigenmap

$\boxed{\text{RBF}}$

Data Mining
Prof. Saerom Park

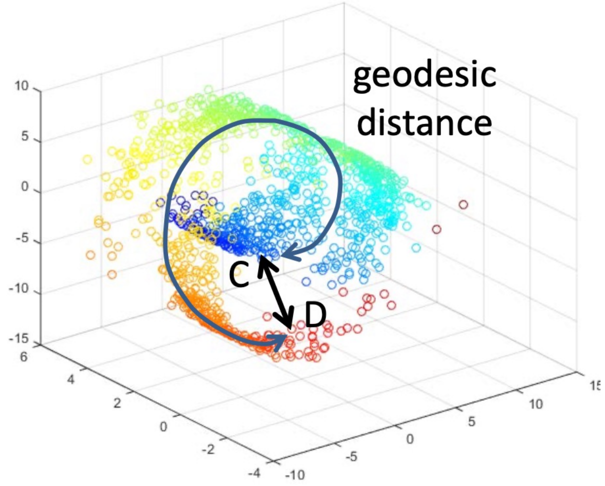
Manifold



- **Key assumption**: High dimensional data actually resides in a *lower dimensional space* that is *locally Euclidean*

- Informally, the manifold is a subset of points in the high-dimensional space that locally looks like a low-dimensional space

Similarity graph models data geometry

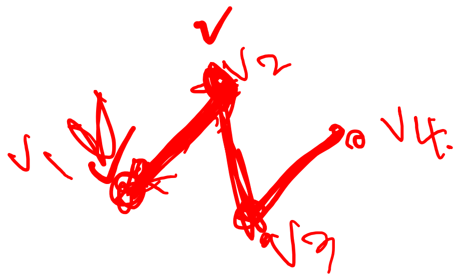


- Manifold Structure
  - Global distance ignores the manifold structure of high-dimensional data
  - Geodesic distance: we can consider the distance along the manifold
- Manifold Learning
  - The idea is that (hopefully) once the manifold is “unfolded”, the analysis, such as clustering becomes easy
  - In essence, “unfolding” a manifold is achieved via dimensionality reduction by considering the relationship between local data points

→ datapoints

- Graph based on local distances
  - Graph $G = \{V, E\}$, where V is a set of vertices, and $E \subset V \times V$ is a set of edges
  - Graph models pairwise relations between objects (similarity or distance)
  - Weighted graph: each vertex v_{ij} has an associated weight w_{ij} (the strength of the relation between objects)

Similarity graph models data geometry



- We will consider weighted undirected graphs with non-negative weights $w_{ij} \geq 0$. Moreover, we will assume that $\mathrm{w_{ij}} = 0$, if and only if vertices i and j are not connected
- The degree of a vertex $v_i \in \boldsymbol{V}$ is defined as

$$\deg(i) = \sum_{j=1}^{n} w_{ij}$$

$= d(x_i, x_j)$

$w_{ij} \geq 0$

$w_{13} = 0$

- The weighted undirected graph can be represented with a weighted adjacency matrix $\boldsymbol{W}$ that contains weights w_{ij} as its elements

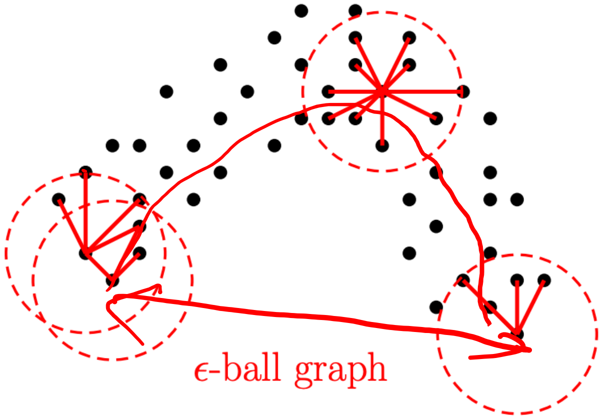
Similarity graph models data geometry

- Geodesic distances can be approximated using a graph in which vertices represent data points
- Let $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the Euclidian distance between the points in the original data space

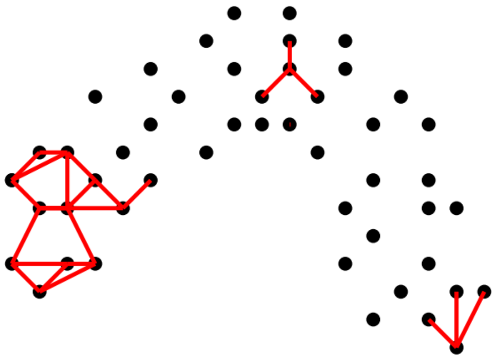
- **Option 1**: define some local radius ϵ

  Connect vertices i and j with an edge if $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \epsilon$
- **Option 2**: define nearest neighbor threshold k ✓ ✓

  Connect vertices i and j if i is among the k nearest neighbors of j OR j is among the k nearest neighbors of i



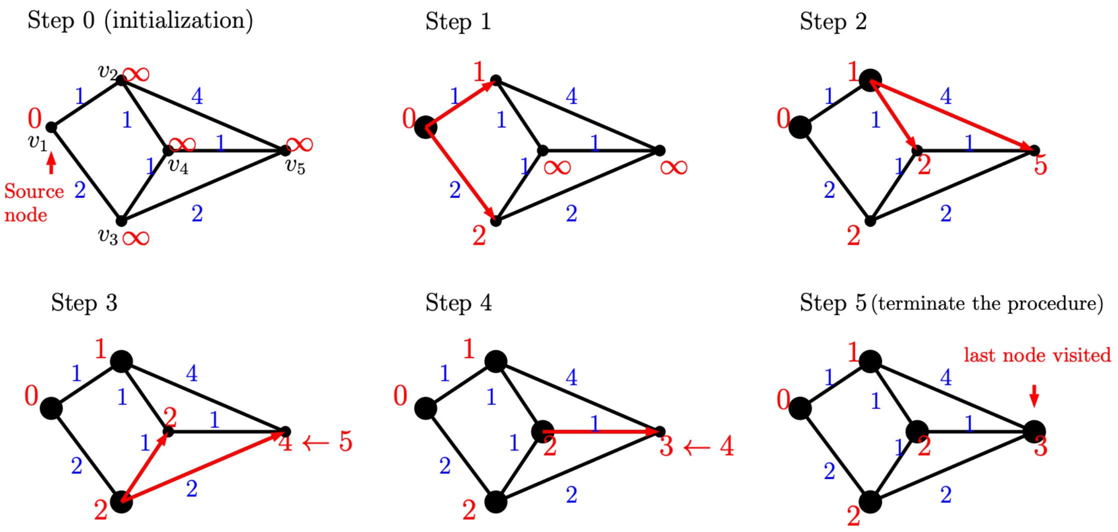
ϵ-ball graph



rNN graph $(r = 3)$

Isomap

- Given n data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$
  1. Compute the k-nearest neighbor graph $\mathrm{G} = (V, E)$
  2. Compute graph distances (geodesic distance) $d_G(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between all points using Dijkstra’s algorithm
  3. Embed the points into low dimensions using metric MDS


Step 0 (initialization)
v_1
v_2
v_3
v_4
v_5
0
∞
Source node
Step 1
Step 2
Step 3
4 ← 5
Step 4
3 ← 4
Step 5 (terminate the procedure)
last node visited