Three-Dimensional Sensors
Lecture 5: Point-Cloud Processing

Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inria.fr
http://perception.inrialpes.fr/

3D Data

- Data representation: PCA and its variants, KD-trees, KNN-graphs.
- Data segmentation: K-means, Gaussian mixtures, spectral clustering.
- Data registration: Iterative closest point (ICP), soft assign methods, robust registration.

Data Representation

- Principal component analysis (PCA): The data are represented in an intrinsic coordinate system and projected onto a lower dimensional space (2D or 1D).
- There are many interesting variants of PCA: probabilistic PCA (PPCA), mixture of PPCA, kernel PCA, etc.
- KD-trees (K-dimensional trees): The 3D point cloud is represented as a binary 3D-tree by recursively splitting the point cloud into two subsets. Provides an efficient way to manipulate the point cloud.
- KNN-graph (K nearest neighbor graph): The 3D point cloud is represented as a sparse *undirected weighted graph*.

Data Segmentation

- K-means clustering: The data are grouped into K spherical clusters. The number of clusters is provided in advance. This algorithm is often use to initialize other clustering methods.
- The Gaussian mixture model (GMM): A more sophisticated clustering method is based on a mixture of Gaussian distributions. This is generally solved using the expectation-maximization algorithm (EM).
- K-means and GMM work well on spherical or ellipsoidal groups of points. Spectral clustering operates in the *spectral space* spanned by the eigenvectors of the symmetric matrix associated with a KNN-graph.
- Clustering methods need KD-trees for efficiently accessing the data points.

Data Registration

- "Fuse" data gathered with several 3D sensors or with a moving 3D sensor.
- Each sensor provides a point cloud in a sensor-centered coordinate frame.
- There is only a partial overlap between the two point clouds.
- The two point clouds must be *registered* or represented in the same coordinate frame.
- The registration process requires point-to-point correspondences which is a difficult problem.

Data Registration Methods

- Iterative closest point (ICP) is the most popular rigid registration method that needs proper initialization of the registration parameters (rotation and translation).
- A number of robust variants of ICP were proposed for eliminating bad points (outliers).
- An alternative to ICP is to use a generative probabilistic model such as GMM, or EM-based point registration.

Some Notations and Definitions

- Let's start with a few more notations:
- The input (observation) space: $\mathbf{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_i \dots \boldsymbol{x}_n]$, $\boldsymbol{x}_i \in \mathbb{R}^3$
- The output (latent) space: $\mathbf{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_i \dots \boldsymbol{y}_n]$, $\boldsymbol{y}_i \in \mathbb{R}^d, 1 \leq d \leq 3$
- **Projection:** $\mathbf{Y} = \mathbf{Q}^\top \mathbf{X}$ with $\mathbf{Q}^\top$ a $d \times 3$ matrix.
- $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_d$

Computing the Spread of the Data

- We start with n scalars $x_1 \dots x_n$; the mean and the variance are given by:

$$\overline{x} = \frac{1}{n}\sum_i x_i \quad \sigma_x = \frac{1}{n}\sum_i (x_i - \overline{x})^2 = \frac{1}{n}\sum_i x_i^2 - \overline{x}^2$$

- More generally, for the data set $\mathbf{X}$:
- The mean: $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_i \boldsymbol{x}_i$
- The covariance matrix is *semi-definite positive symmetric* of dimension 3×3:

$$\mathbf{C}_X = \frac{1}{n}\sum_i (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\top = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \overline{\boldsymbol{x}}\,\overline{\boldsymbol{x}}^\top$$

Maximum-Variance Formulation of PCA

- Let's center and project the data $\mathbf{X}$ onto a line along a unit vector $\boldsymbol{u}$. The variance along this line writes:

$$\begin{aligned}\sigma_u &= \frac{1}{n}\sum_i (\boldsymbol{u}^\top(\boldsymbol{x}_i - \overline{\boldsymbol{x}}))^2 \\ &= \boldsymbol{u}^\top \left(\frac{1}{n}\sum_i (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\top\right)\boldsymbol{u} \\ &= \boldsymbol{u}^\top \mathbf{C}_X \boldsymbol{u}\end{aligned}$$

- Find $\boldsymbol{u}$ maximizing the variance under the constraint that $\boldsymbol{u}$ is a unit vector:

$$\boldsymbol{u}^\star = \underset{\boldsymbol{u}}{\operatorname{argmax}} \left\{\boldsymbol{u}^\top \mathbf{C}_X \boldsymbol{u} + \lambda(1 - \boldsymbol{u}^\top \boldsymbol{u})\right\}$$

Maximum-Variance Solution

- First note that the 3×3 covariance matrix is a symmetric semi-definite positive matrix. (The associated quadratic form above is non-negative).
- Taking the derivative with respect to $\boldsymbol{u}$ and setting the derivatives equal to 0, yields: $\mathbf{C}_X \boldsymbol{u} = \lambda \boldsymbol{u}$
- Making use of the fact that $\boldsymbol{u}$ is a unit vector we obtain: $\sigma_u = \lambda$
- **Solution:** The *principal* or largest eigenvector–eigenvalue pair $(\boldsymbol{u}_{\max}, \lambda_{\max})$ of the covariance matrix.

Eigendecomposition of the Covariance Matrix

- Assume that the data are centred:

$$n\mathbf{C}_X = \mathbf{X}\mathbf{X}^\top = n\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

  Where $\mathbf{U}$ is a 3×3 orthogonal matrix and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues:

$$\mathbf{\Lambda} = [\lambda_1 \ \lambda_2 \ \lambda_3]$$

- If the point-cloud lies on a lower d-dimensional space (collinear or planar points):

$$d = \text{rank}(\mathbf{X}) < 3$$

  and

$$\mathbf{\Lambda}_d = [\lambda_1 \ \lambda_d]$$
$$\mathbf{C}_X = \widetilde{\mathbf{U}}\mathbf{\Lambda}_d\widetilde{\mathbf{U}}^\top$$

- $\widetilde{\mathbf{U}} = \mathbf{U}\mathbf{I}_{3\times d}$ is a $3 \times d$ column-orthgonal matrix
- $\widetilde{\mathbf{U}}^\top = \mathbf{I}_{d\times 3}^\top\mathbf{U}^\top$ is a $d \times 3$ row-orthgonal matrix

Data Representation in the Eigen (Sub)space

- Coordinate change: $\mathbf{Y} = \mathbf{Q}\mathbf{X}$; We have

$$\mathbf{Y}\mathbf{Y}^\top = \mathbf{Q}\mathbf{X}\mathbf{X}^\top\mathbf{Q}^\top = n\mathbf{Q}\widetilde{\mathbf{U}}\mathbf{\Lambda}_d\widetilde{\mathbf{U}}^\top\mathbf{Q}^\top$$

1. The projected data have a diagonal covariance matrix: $\frac{1}{n}\mathbf{Y}\mathbf{Y}^\top = \mathbf{\Lambda}_d$, by identification we obtain

$$\mathbf{Q} = \widetilde{\mathbf{U}}^\top$$

2. The projected data have an identity covariance matrix, this is called *whitening the data*: $\frac{1}{n}\mathbf{Y}\mathbf{Y}^\top = \mathbf{I}_d$

$$\mathbf{Q} = \mathbf{\Lambda}_d^{-\frac{1}{2}}\widetilde{\mathbf{U}}^\top$$

- Projection of the data points onto principal direction $\boldsymbol{u}_i$:

$$(y_{1i} \dots y_{ni}) = \underbrace{\lambda_i^{-1/2}}_{\text{whitening}} \boldsymbol{u}_i^\top (\boldsymbol{x}_1 \dots \boldsymbol{x}_n)$$

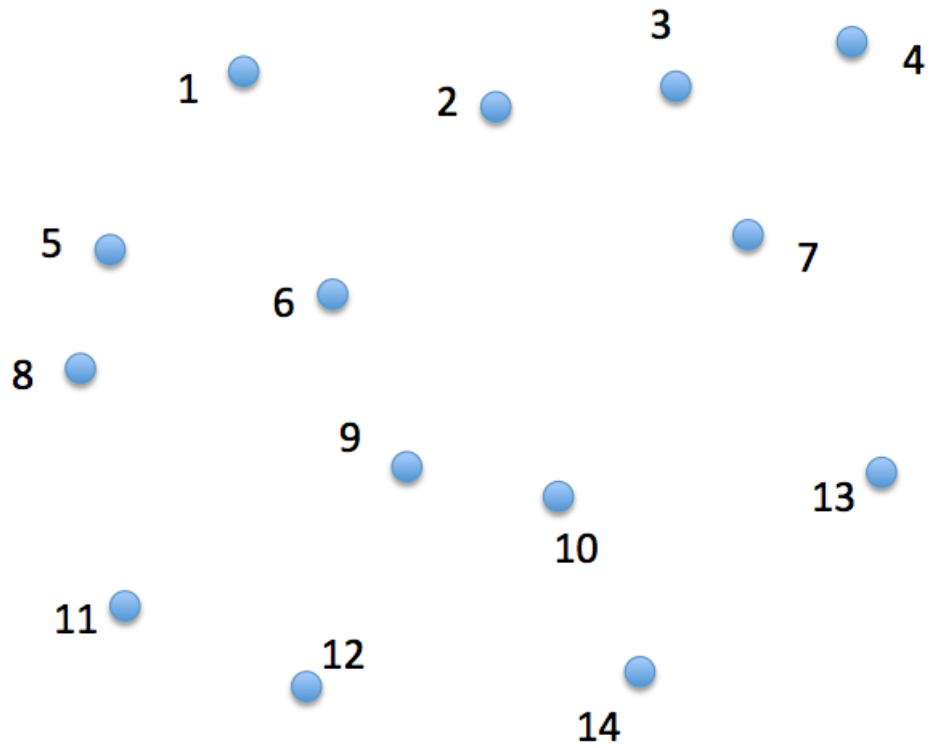
Summary of PCA

- The eigenvector-eigenvalue pairs of the covariance matrix correspond to a *spectral* representation of the point cloud, or a *within representation*.
- This eigendecomposition allows to reduce the dimensionality of the point cloud to one plane or one line and then to project the cloud onto such a linear subspace.
- The largest eigenvalue-eigenvector pair defines the direction of maximum variance. By projecting the data onto this line one can order the data (useful for data organization, i.e., KD-trees).
- The eigenvalue-eigenvector pairs can be efficiently computed using the power method: get a random unit vector $\boldsymbol{x}^{(0)}$ and iterate $\boldsymbol{x}^{(k+1)} = \mathbf{C}\boldsymbol{x}^{(k)}$, normalize $\boldsymbol{x}^{(k+1)}$, etc., until $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| < \varepsilon$. Then $\boldsymbol{u}_{\max} = \boldsymbol{x}^{(k+1)}$.

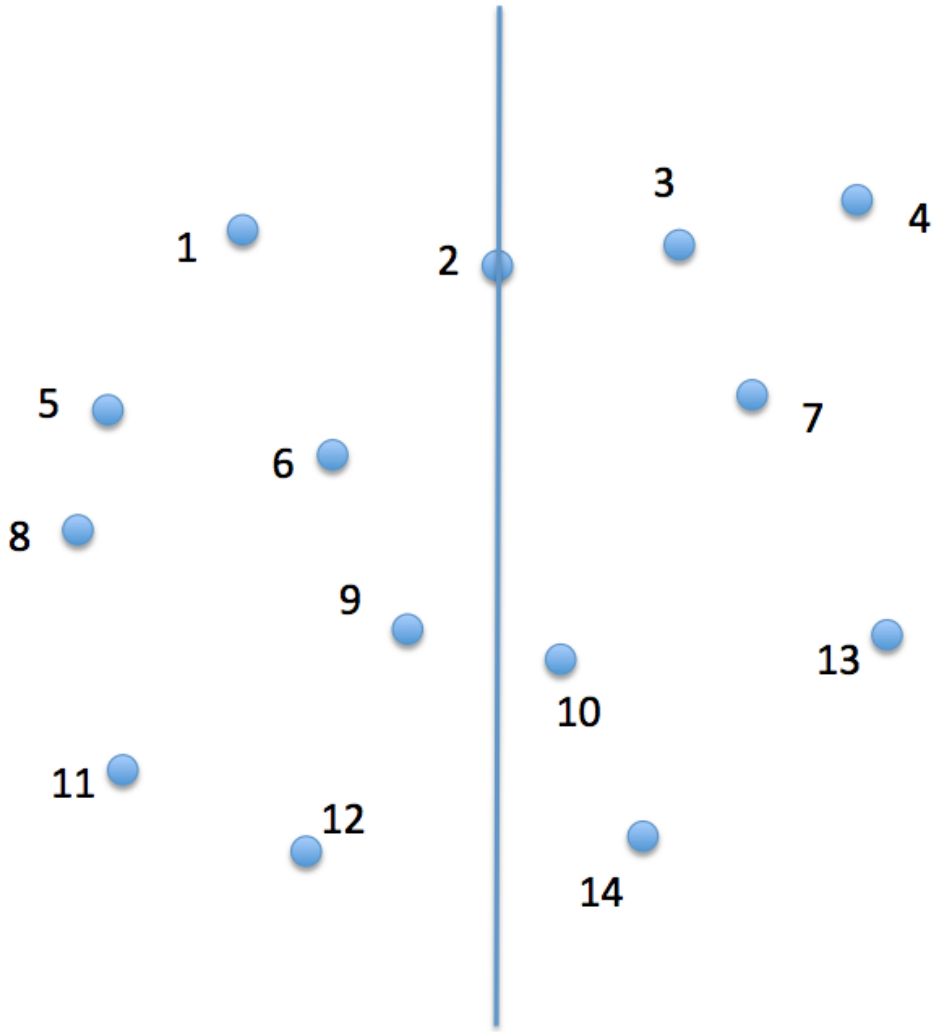
KD Trees

- KD-tree (K-dimensional tree) is a data structure that allows to organize a point cloud under the form of a binary tree.
- The basic idea is to recursively and alternatively project the points onto the x, y, z, x, y, z, etc., axes, to order the points along each axis and to split the set into two halves.
- This point-cloud organization facilitates and accelerates the search of nearest neighbors (at the price of kd-tree construction).
- A more elaborate method (requiring more pre-processing time) is to search for the principal direction and split the data using a plane orthogonal to this direction, and apply this strategy recursively.

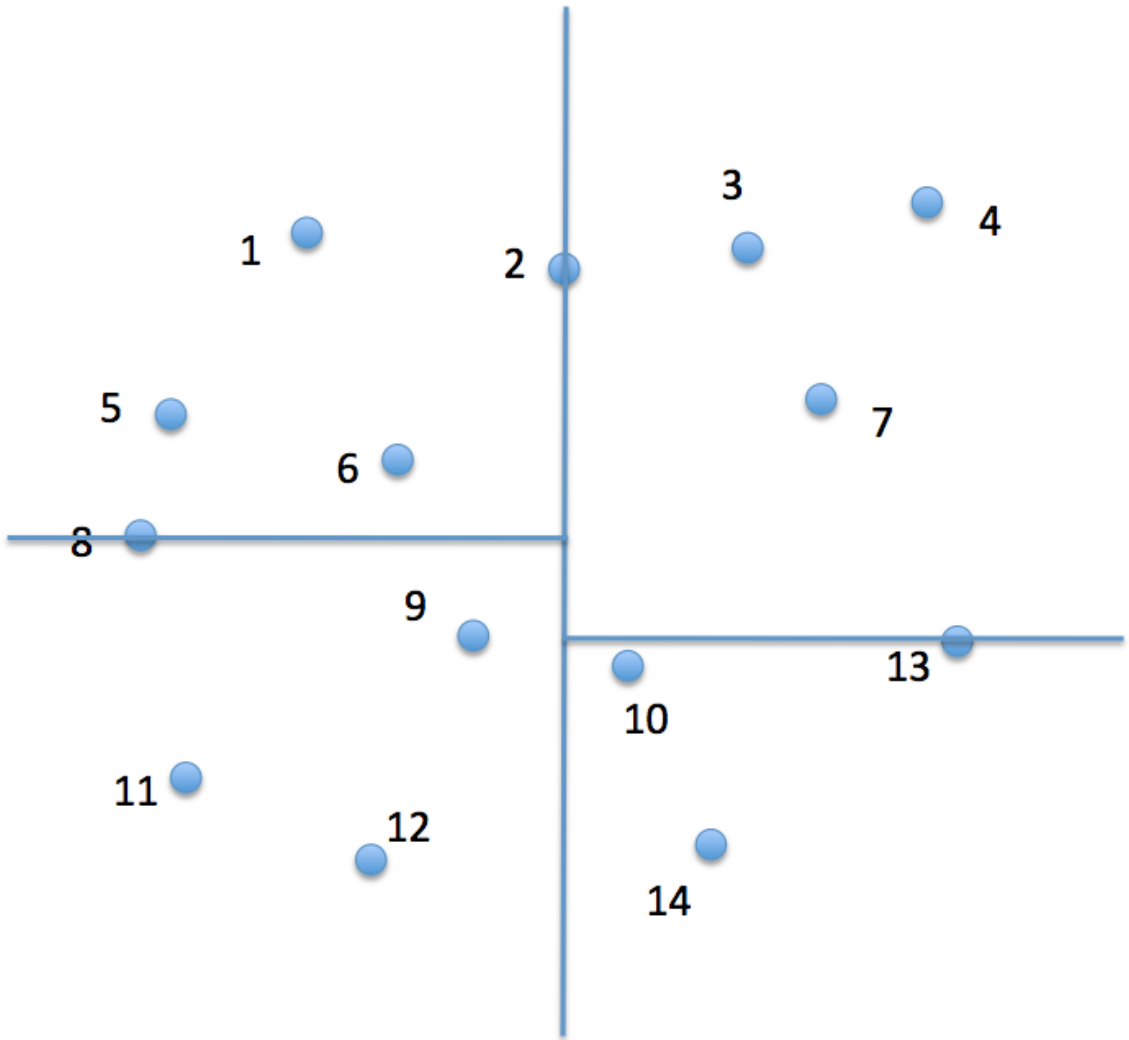
An Example of a 2D-tree (1)



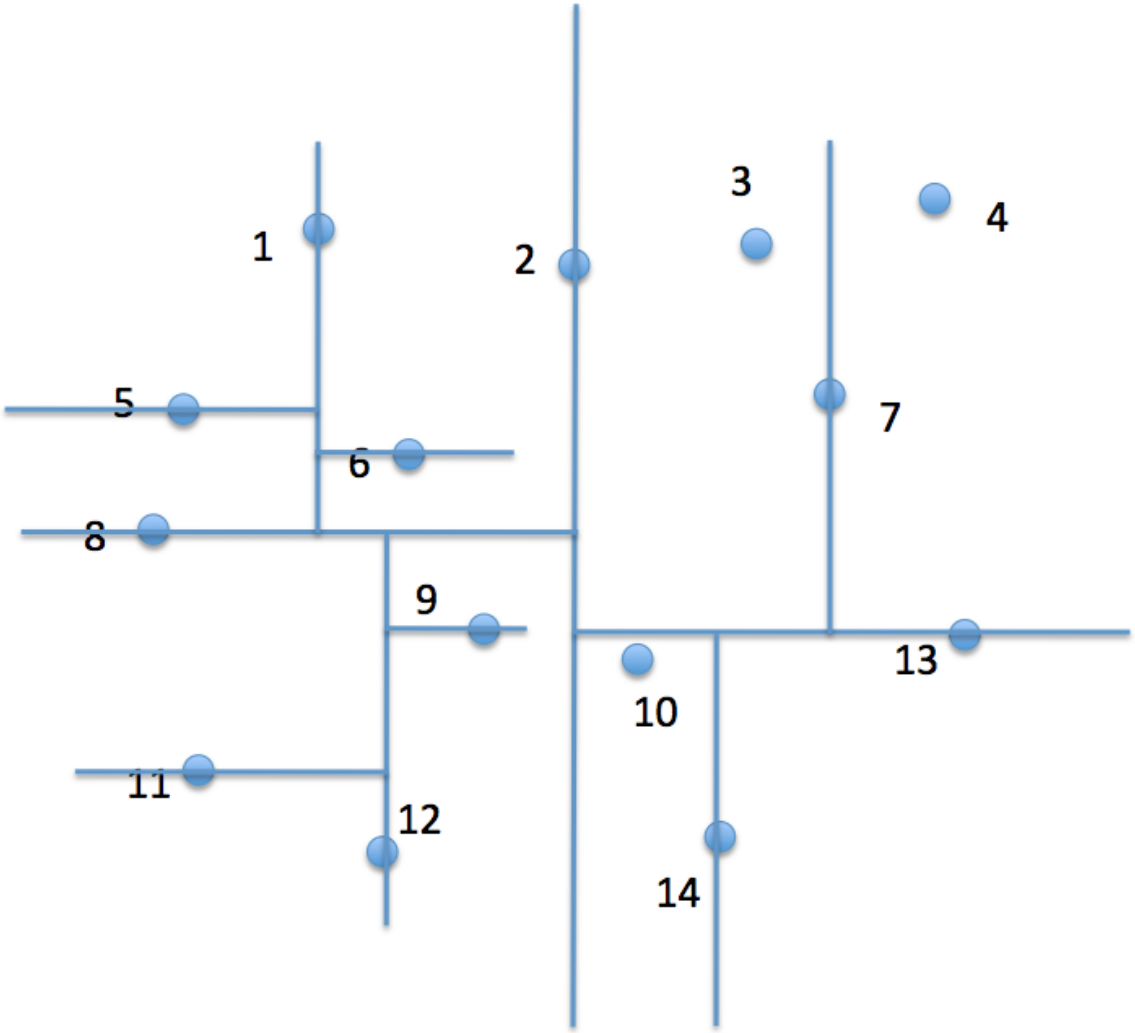
An Example of a 2D-tree (2)



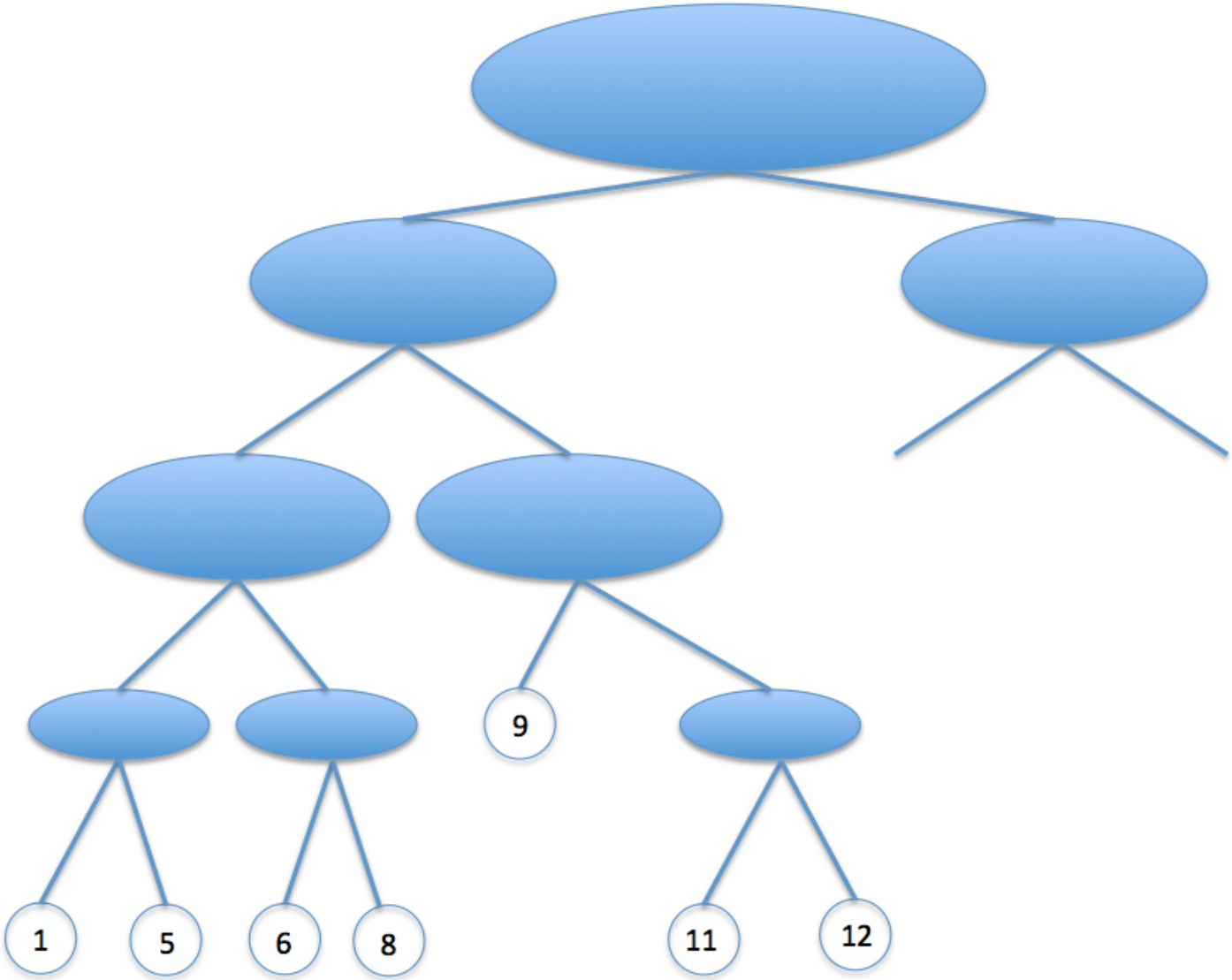
An Example of a 2D-tree (3)



An Example of a 2D-tree (4)



An Example of a 2D-tree (5)



K-means Clustering

- What is a cluster: a group of points whose inter-point distance are small compared to distances to points outside the cluster.
- Cluster centers: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m$.
- Goal: find an assignment of points to clusters as well as a set of mean-vectors $\boldsymbol{\mu}_k$.
- Notations: For each point $\boldsymbol{x}_j$ there is a *binary indicator variable* $r_{jk} \in \{0, 1\}$.
- Objective: minimize the following *distorsion measure*:

$$J = \sum_{j=1}^{n} \sum_{k=1}^{m} r_{jk} \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\|^2$$

The K-means Algorithm

1. Initialization: Choose m and initial values for $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m$.
2. First step: Assign the j-th point to the closest cluster center:

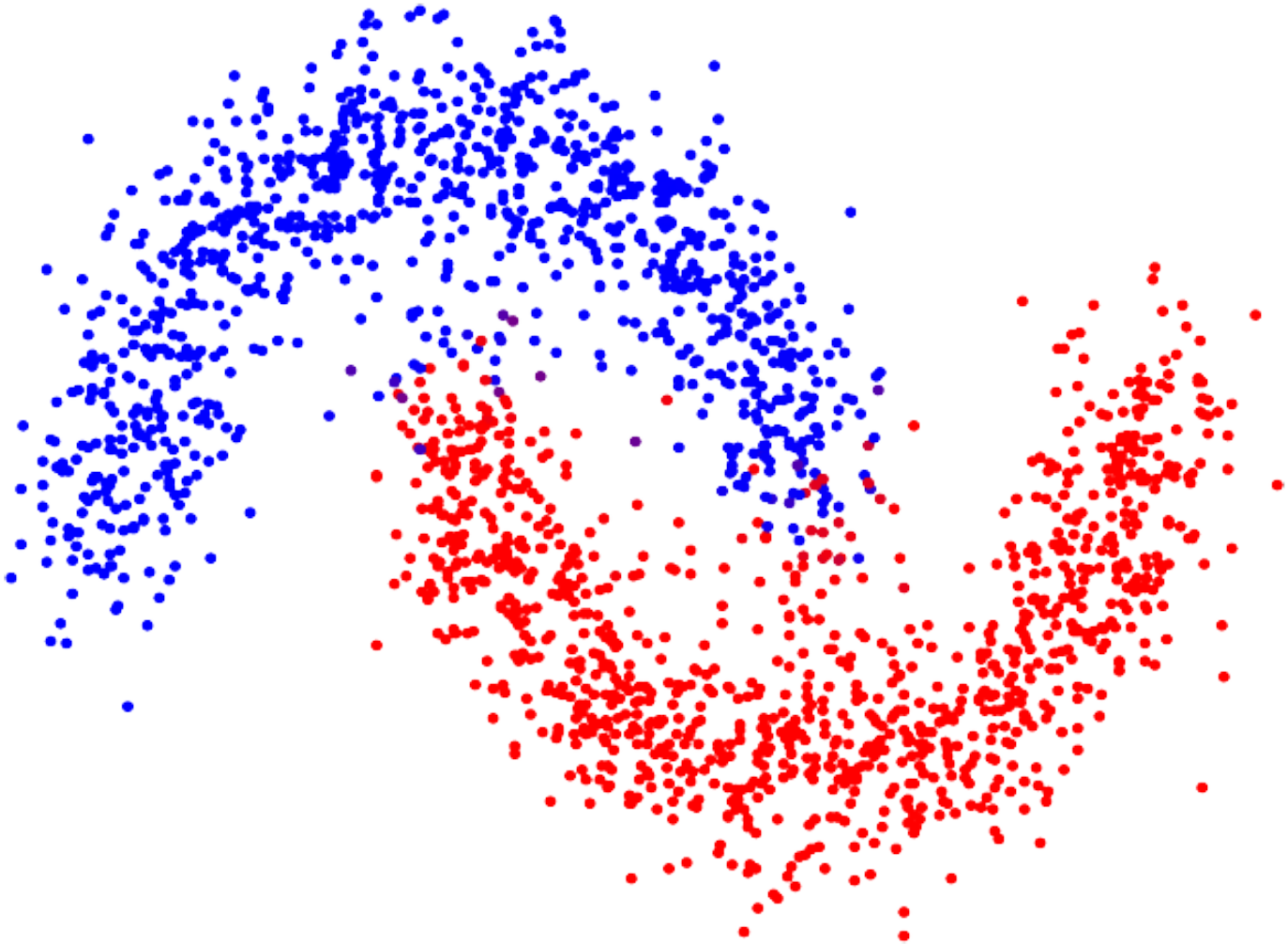
$$r_{jk} = \begin{cases} 1 & \text{if } k = \arg\min_l \|\boldsymbol{x}_j - \mu_l\|^2 \\ 0 & \text{otherwise} \end{cases}$$

3. Second Step: Minimize J to estimate the cluster centers:

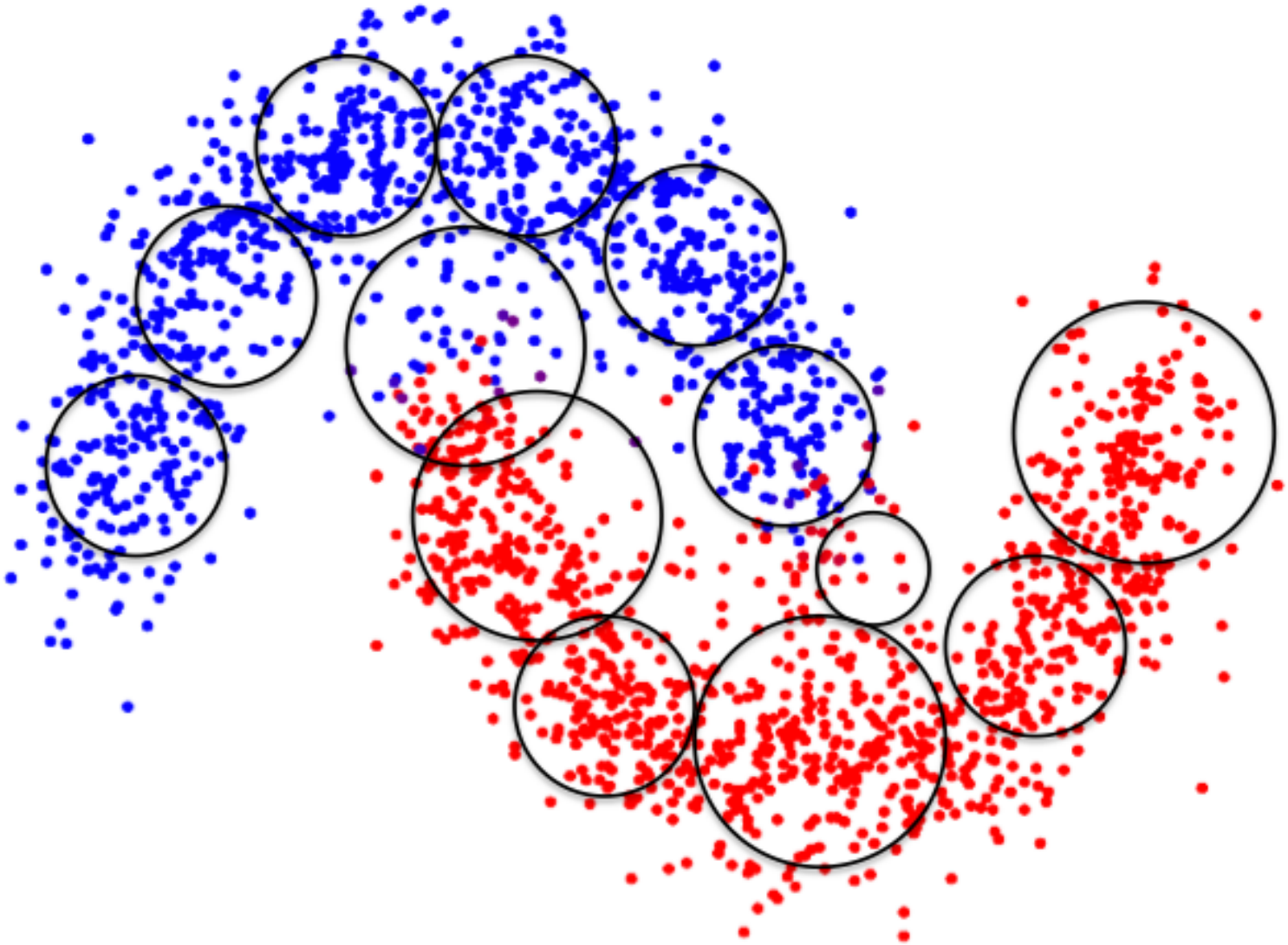
$$\boldsymbol{\mu}_k = \frac{\sum_{j=1}^{n} r_{jk} \boldsymbol{x}_j}{\sum_{j=1}^{n} r_{jk}}$$

4. Convergence: Repeat until no more change in the assignments.

How to Represent This Point Cloud?

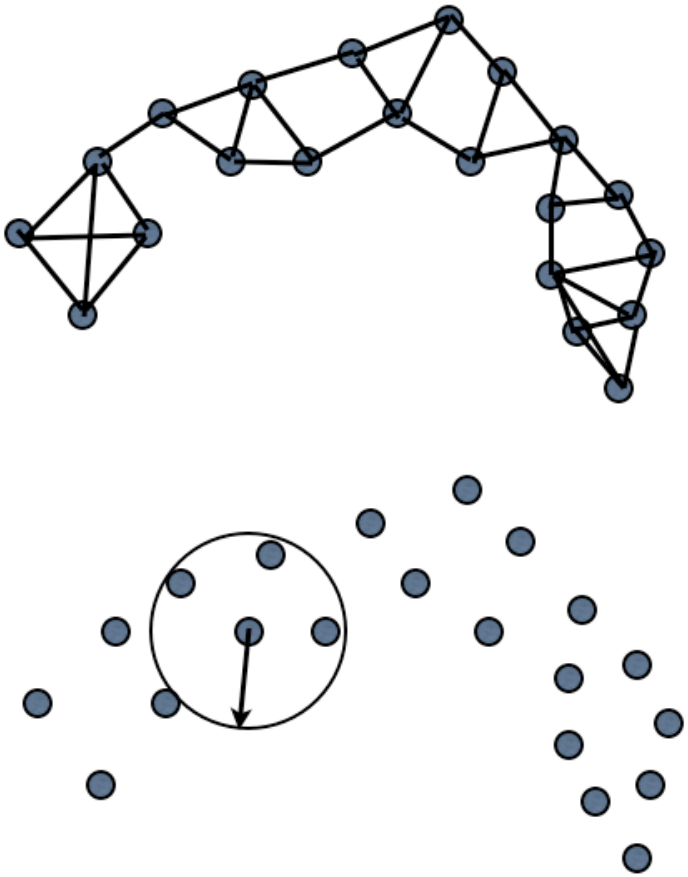


Spherical Clusters



Building a Graph from a Point Cloud

- K-nearest neighbor (KNN) rule
- ε-radius rule
- Other more sophisticated rules can be found in the literature, i.e., Lee and Verleysen. Nonlinear Dimensionality Reduction (Appendix E). Springer. 2007.



- Remark: The KD-tree data structure can be used to facilitate graph construction when the number of points is large.

The Graph Partitioning Problem

- We want to find a partition of the graph such that the edges between different groups have very low weight, while the edges within a group have high weight.
- **The mincut problem**:
  1. Edges between groups have very low weight, and
  2. Edges within a group have high weight.
  3. Choose a partition of the graph into k groups that mimimizes the following criterion:

$$\text{mincut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i)$$

- with

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

RatioCut and NormalizedCut

- Often, the mincut solution isolates a vertex from the rest of the graph.
- Request that the groups are reasonably large.
- **Ratio cut** (Hagen & Kahng 1992) minimizes:

$$\text{RatioCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{|A_i|}$$

- Here $|A|$ refers to the number of vertices in group A.
- **Normalized cut**: (Shi & Malik 2000)

$$\text{NCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{\text{vol}(A_i)}$$

What is Spectral Clustering?

- Both ratio-cut and normalized-cut minimizations are NP-hard problems
- Spectral clustering is a way to solve relaxed versions of these problems:
  1. Build the Laplacian matrix of the graph
  2. Compute the smallest (non-null) eigenvalue-eigenvector pairs of this matrix
  3. Map the graph vertices into the space spanned by these eigenvectors
  4. Apply the K-means algorithm to the new point cloud

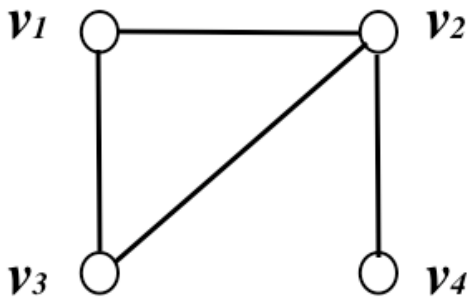
The Laplacian Matrix of a Graph

- $f : \mathcal{V} \longrightarrow \mathbb{R}$, i.e., $f(v_1), \ldots, f(v_n)$.
- $(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- The degree matrix: $\mathbf{D} := D_{ii} = d(v_i)$.

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



Case of an Undirected Weighted Graph

- We consider *undirected weighted graphs*; Each edge e_{ij} is weighted by $w_{ij} > 0$. We obtain:

$$\mathbf{\Omega} := \begin{cases} \Omega_{ij} = w_{ij} & \text{if there is an edge } e_{ij} \\ \Omega_{ij} = 0 & \text{if there is no edge} \\ \Omega_{ii} = 0 & \end{cases}$$

- The degree matrix: $\mathbf{D} = \sum_{i \sim j} w_{ij}$

The Laplacian on an Undirected Weighted Graph

- Often we will consider:

$$w_{ij} = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2\right)$$

- $\mathbf{L} = \mathbf{D} - \mathbf{\Omega}$
- $\mathbf{L}$ is symmetric and positive semi-definite $\leftrightarrow$ $w_{ij} \geq 0$.
- $\mathbf{L}$ has n non-negative, real-valued eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

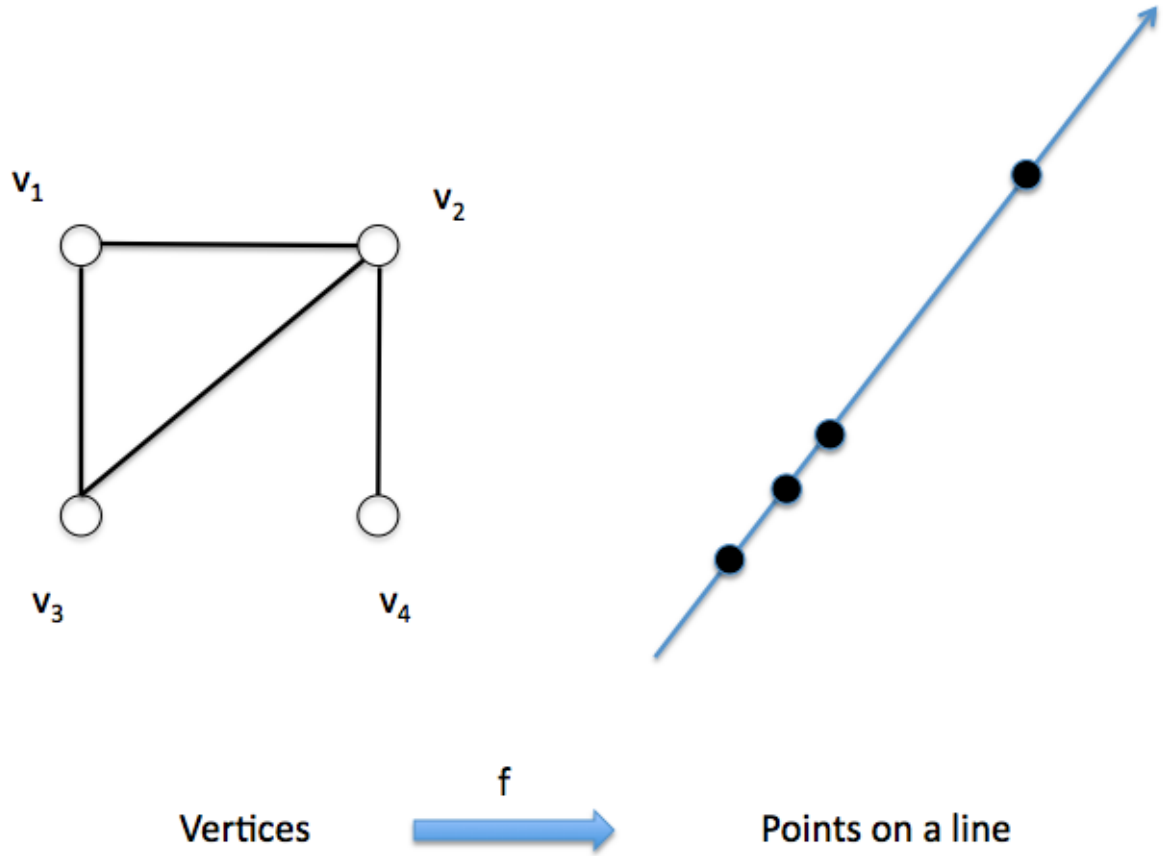
Laplacian embedding: Mapping a graph on a line

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize $\sum_{i,j=1}^{n} w_{ij}(f(v_i) - f(v_j))^2$, or:

$$\arg\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} \text{ with: } \boldsymbol{f}^\top \boldsymbol{f} = 1 \text{ and } \boldsymbol{f}^\top \mathbf{1} = 0$$

- The solution is the eigenvector associated with the smallest nonzero eigenvalue of the eigenvalue problem: $\mathbf{L}\boldsymbol{f} = \lambda \boldsymbol{f}$, (the Fiedler vector) $\boldsymbol{u}_2$.
- Practical computation of the eigenpair $\lambda_2, \boldsymbol{u}_2$): the shifted inverse power method (see lecture 2).

Mapping the Graph's Vertices on the Eigenvector



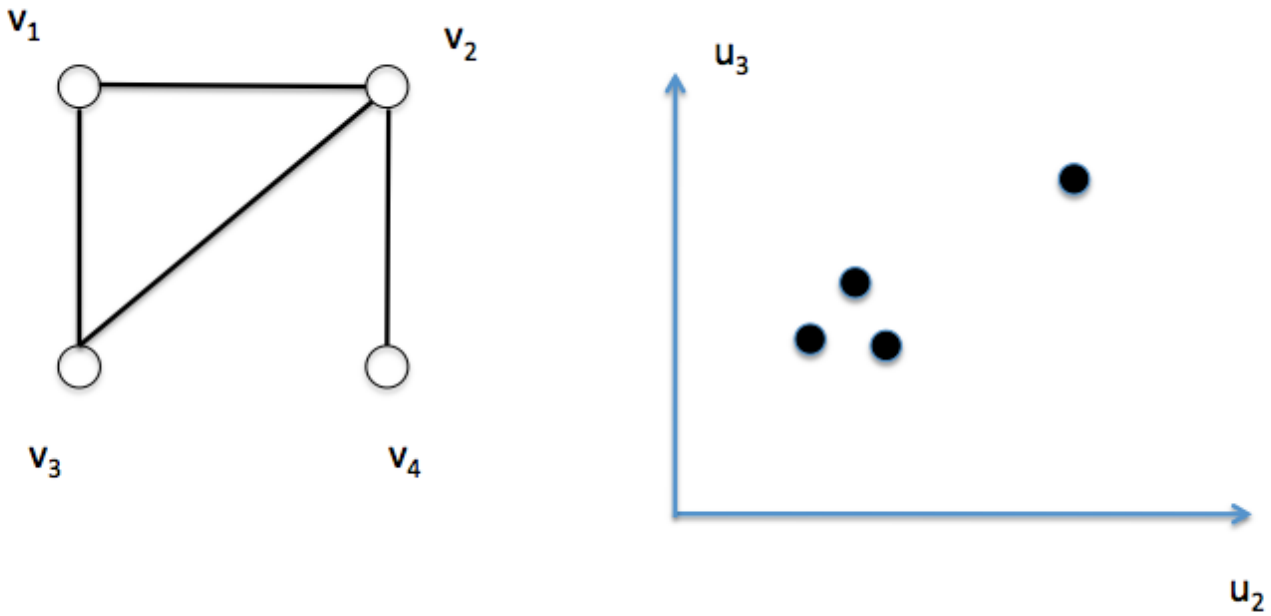
Spectral Embedding using the Laplacian

- Compute the eigendecomposition $\mathbf{L} = \mathbf{D} - \mathbf{\Omega} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$.
- Select the k smallest non-null eigenvalues $\lambda_2 \leq \ldots \leq \lambda_{k+1}$
- $\lambda_{k+2} - \lambda_{k+1} =$ **eigengap**.
- We obtain the $n \times k$ column-orthogonal matrix $\widetilde{\mathbf{U}} = [\boldsymbol{u}_2 \ldots \boldsymbol{u}_{k+1}]$:

$$\widetilde{\mathbf{U}} = \begin{bmatrix} \boldsymbol{u}_2(v_1) & \ldots & \boldsymbol{u}_{k+1}(v_1) \\ \vdots & & \vdots \\ \boldsymbol{u}_2(v_n) & \ldots & \boldsymbol{u}_{k+1}(v_n) \end{bmatrix}$$

- Embedding: The i-row of this matrix correspond to the representation of vertex v_I in the $\mathbb{R}^k$ basis spanned by the orthonormal vector basis $\boldsymbol{u}_2, \ldots, \boldsymbol{u}_{k+1}$.
- Therefore: $\mathbf{Y} = [\boldsymbol{y}_1 \ldots \boldsymbol{y}_i \ldots \boldsymbol{y}_n] = \widetilde{\mathbf{U}}^\top$

Laplacian Eigenmap



Next Lecture: Data Registration

- "Fuse" data gathered with several 3D sensors or with a moving 3D sensor.
- Each sensor provides a point cloud in a sensor-centered coordinate frame.
- There is only a partial overlap between the two point clouds.
- The two point clouds must be *registered* or represented in the same coordinate frame.
- The registration process requires point-to-point correspondences which is a difficult problem.