NFFT meets Krylov methods:

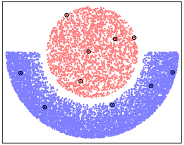
Fast matrix-vector products for the graph Laplacian of fully connected networks

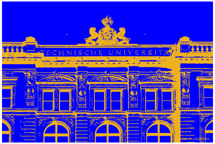
Dominik Alfke, Daniel Potts, Martin Stoll, Toni Volkmer

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Outline

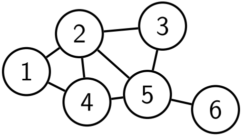
- (Fully-Connected) Graphs and Graph Laplacian Matrix
- NFFT-based fast summation
- Application to Learning (Classification)
  - Semi-Supervised
  - Unsupervised





Graph terminology: Undirected graph

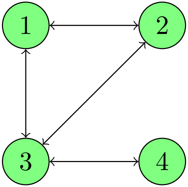
A set of **nodes** that may be connected by (undirected) **edges**



- Nodes refer to data points that may contain information
- Edges show that two data points are related
- Nodes are numbered, e.g., from 1 to n
- Edges encoded in the symmetric adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$

Unweighted graph

Graph sketch



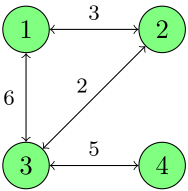
Adjacency matrix $\mathbf{W}$

	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

- If i and j are connected: $w_{ij} = 1$
- If i and j are not connected: $w_{ij} = 0$
- On the diagonal: $w_{ii} = 0$

Weighted graph

Graph sketch



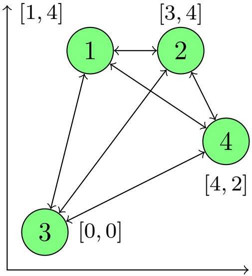
Adjacency matrix $\mathbf{W}$

	1	2	3	4
1		3	6	
2	3		2	
3	6	2		5
4			5	

- If i and j are connected: $w_{ij} =$ edge weight
- If i and j are not connected: $w_{ij} = 0$
- On the diagonal: $w_{ii} = 0$

Graph types: Fully connected graph with node features

Graph sketch



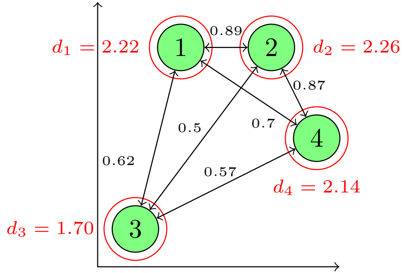
Adjacency matrix $\mathbf{W}$

	1	2	3	4
1		0.89	0.62	0.7
2	0.89		0.5	0.87
3	0.62	0.5		0.57
4	0.7	0.87	0.57	

- Each node i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$
- For all $i \neq j$: $\quad w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$ $\qquad \sigma = 6$
  - Large (close to one) entries for similar features
  - Small (close to zero) entries for dissimilar features
- ~~If i and j are not connected: $w_{ij} = 0$~~
- On the diagonal: $w_{ii} = 0$

Node degrees

Graph sketch



Adjacency matrix $\mathbf{W}$

	1	2	3	4
1		0.89	0.62	0.7
2	0.89		0.5	0.87
3	0.62	0.5		0.57
4	0.7	0.87	0.57	

Degree matrix $\mathbf{D}$

	1	2	3	4
1	2.22			
2		2.26		
3			1.70	
4				2.14

- Node degree $d_i = \sum_{j=1}^{n} w_{ij}$: Sum of all weights of edges connected to node i
- Degree matrix $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n) = \mathrm{diag}(\mathbf{W} \cdot \mathbf{1})$

Graph Laplacian matrix

- Most important tool in graph-based data science
- Symmetrically normalized version:

$$\mathbf{L} = \mathbf{D}^{-1/2}\,(\mathbf{D} - \mathbf{W})\,\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\,\mathbf{W}\,\mathbf{D}^{-1/2}$$

- Entries:

$$L_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{w_{ij}}{\sqrt{d_i d_j}} & \text{if } i \neq j \end{cases}$$

Adjacency matrix $\mathbf{W}$

	1	2	3	4
1		0.89	0.62	0.7
2	0.89		0.5	0.87
3	0.62	0.5		0.57
4	0.7	0.87	0.57	

Degree matrix $\mathbf{D}$

	1	2	3	4
1	2.22			
2		2.26		
3			1.70	
4				2.14

Laplacian matrix $\mathbf{L}$

	1	2	3	4
1	1	-0.40	-0.322	-0.320
2	-0.40	1	-0.255	-0.395
3	-0.322	-0.255	1	-0.301
4	-0.320	-0.395	-0.301	1

Eigenvalues of the graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$

Eigenvalue λ and eigenvector $\mathbf{v} \neq \mathbf{0}$ with $\mathbf{L}\mathbf{v} = \lambda\mathbf{v}$

- All eigenvalues are in $[0, 2)$
- Smallest eigenvalue 0 is always present
  (multiple times if the graph is not connected)
- Small eigenvalues $\lambda > 0$: $\mathbf{v}$ contains clustering information
- Large eigenvalues $\lambda < 2$: $\mathbf{v}$ contains noise

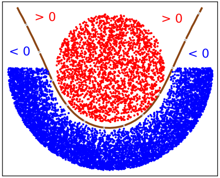
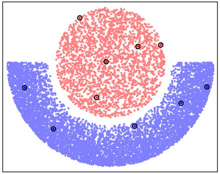
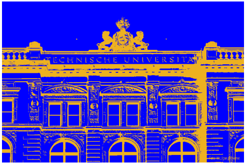
Computation (of $k \ll n$ eigenpairs) using Lanczos algorithm (MATLAB eigs), requires only matrix-vector multiplications with matrix $\mathbf{L}$, in particular with matrix $\mathbf{W}$

Matrix-vector multiplication with $\mathbf{W} \cdot \boldsymbol{\alpha}$

$$\mathbf{W} \cdot \boldsymbol{\alpha} = \Big(g(\mathbf{x}_i)\Big)_{i=1}^{n}, \qquad g(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2},$$

requires $\mathcal{O}(n^2)$ arithmetic operations

Many different learning tasks benefit from fast methods for computing $\mathbf{W} \cdot \boldsymbol{\alpha}$:

type	supervised	semi-supervised	unsupervised
method	kernel ridge regression	kernel method	spectral clustering
approach	$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\mathbf{f} - \mathbf{W}\,\boldsymbol{\alpha}\|_2^2 + \beta\,\boldsymbol{\alpha}^\top \mathbf{W}\,\boldsymbol{\alpha}$	$\min_{\mathbf{u} \in \mathbb{R}^n} \|\mathbf{u} - \mathbf{f}\|_2^2 + \beta\,\mathbf{u}^\top\,\mathbf{L}\,\mathbf{u}$	compute eigenvectors of $\mathbf{L}$, apply kmeans
example			$n = 426\,400$ 

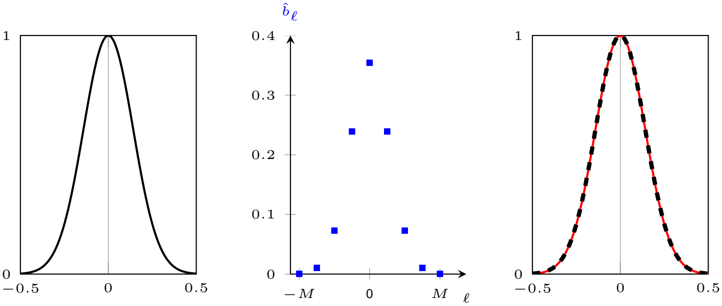
Outline

- (Fully-Connected) Graphs and Graph Laplacian
- NFFT-based fast summation
- Application to Learning (Classification)
  - Semi-Supervised
  - Unsupervised

Fourier method

$$\mathbf{W} \cdot \boldsymbol{\alpha} = \Big(g(\mathbf{x}_i) \Big)_{i=1}^{n}, \qquad g(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$$

Ansatz: Approximate kernel $\mathcal{K}(\mathbf{x})$, e.g. $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$,
by truncated Fourier series $\mathcal{K}_{\mathrm{RF}}(\mathbf{x}) = \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}}$



$\mathrm{Re}\left(\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}}\right)$
$= \cos(2\pi\boldsymbol{\ell} \cdot \mathbf{x})$

$\mathcal{K}_{\mathrm{RF}}(x) = 0.35$
$+ 0.4782 \cos(2\pi x)$
$+ 0.1457 \cos(4\pi x)$
$+ 0.0207 \cos(6\pi x)$
$+ 0.0009 \cos(8\pi x)$

Fourier method

$$\mathbf{W} \cdot \boldsymbol{\alpha} = \Big(g(\mathbf{x}_i)\Big)_{i=1}^{n}, \quad g(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$$

Ansatz: Approximate kernel $\mathcal{K}(\mathbf{x})$, e.g. $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$,
by truncated Fourier series $\mathcal{K}_{\mathrm{RF}}(\mathbf{x}) = \sum_{\boldsymbol{\ell} \in \{-M,\dots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}}$

$$(\mathbf{W} \cdot \boldsymbol{\alpha})_i \approx \sum_{j=1}^{n} \alpha_j \, \mathcal{K}_{\mathrm{RF}}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \sum_{\boldsymbol{\ell} \in \{-M,\dots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \cdot (\mathbf{x}_i - \mathbf{x}_j)}$$

Fourier method

$$\mathbf{W} \cdot \boldsymbol{\alpha} = \Big(g(\mathbf{x}_i)\Big)_{i=1}^{n}, \quad g(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$$

Ansatz: Approximate kernel $\mathcal{K}(\mathbf{x})$, e.g. $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$,
by truncated Fourier series $\mathcal{K}_{\mathrm{RF}}(\mathbf{x}) = \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}}$

$$(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{j=1}^{n} \alpha_j \, \mathcal{K}_{\mathrm{RF}}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot(\mathbf{x}_i - \mathbf{x}_j)}$$

$$= \sum_{j=1}^{n} \alpha_j \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j}$$

Fourier method

$$\mathbf{W} \cdot \boldsymbol{\alpha} = \Big(g(\mathbf{x}_i)\Big)_{i=1}^{n}, \quad g(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x}_i - \mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$$

Ansatz: Approximate kernel $\mathcal{K}(\mathbf{x})$, e.g. $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$,
by truncated Fourier series $\mathcal{K}_{\mathrm{RF}}(\mathbf{x}) = \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}}$

$$\begin{aligned}(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{j=1}^{n} \alpha_j \, \mathcal{K}_{\mathrm{RF}}(\mathbf{x}_i - \mathbf{x}_j) &= \sum_{j=1}^{n} \alpha_j \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot(\mathbf{x}_i - \mathbf{x}_j)} \\ &= \sum_{j=1}^{n} \alpha_j \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j} \\ &= \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \left(\sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j} \right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}\end{aligned}$$

NFFT-based fast summation ([Potts, Steidl 2004])

Fast computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$:

0. $\hat{b}_{\boldsymbol{\ell}} := \sum_{\mathbf{j} \in \{-M,\ldots,M\}^d} \mathcal{K}\left(\frac{\mathbf{j}}{M}\right) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{j}/M}$, $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$, by d-dim. FFT
1. Compute (nonequispaced adjoint) DFT:
   $\hat{c}_{\boldsymbol{\ell}} := \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_j}$ for all $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$
2. Multiply Fourier coefficients: $\hat{f}_{\boldsymbol{\ell}} := \hat{b}_{\boldsymbol{\ell}} \, \hat{c}_{\boldsymbol{\ell}}$ for all $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$
3. Compute (nonequispaced) DFT:
   $(\mathbf{W} \cdot \boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{f}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_i}$ for all $i = 1, \ldots, n$

$$(\mathbf{W} \cdot \boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \left(\sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_j} \right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_i}$$

NFFT-based fast summation ([Potts, Steidl 2004])

Fast computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$:

0. $\hat{b}_{\boldsymbol{\ell}} := \sum_{\mathbf{j} \in \{-M,\ldots,M\}^d} \mathcal{K}\left(\frac{\mathbf{j}}{M}\right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{j}/M}$, $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$, by d-dim. FFT
1. Compute (nonequispaced adjoint) DFT:
   $\hat{c}_{\boldsymbol{\ell}} := \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j}$ for all $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$
2. Multiply Fourier coefficients: $\hat{f}_{\boldsymbol{\ell}} := \hat{b}_{\boldsymbol{\ell}} \, \hat{c}_{\boldsymbol{\ell}}$ for all $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$
3. Compute (nonequispaced) DFT:
   $(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{f}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$ for all $i = 1,\ldots,n$

$$(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \left(\sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j} \right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$$

NFFT-based fast summation ([Potts, Steidl 2004])

Fast computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$:

0. $\hat{b}_{\boldsymbol{\ell}} := \sum_{\mathbf{j} \in \{-M,\ldots,M\}^d} \mathcal{K}\left(\frac{\mathbf{j}}{M}\right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{j}/M}$, $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$, by d-dim. FFT
1. Compute (nonequispaced adjoint) DFT:
   $\hat{c}_{\boldsymbol{\ell}} := \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j}$ for all $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$
2. Multiply Fourier coefficients: $\hat{f}_{\boldsymbol{\ell}} := \hat{b}_{\boldsymbol{\ell}} \, \hat{c}_{\boldsymbol{\ell}}$ for all $\boldsymbol{\ell} \in \{-M,\ldots,M\}^d$
3. Compute (nonequispaced) DFT:
   $(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{f}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$ for all $i = 1,\ldots,n$

$$(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \left(\sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j} \right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$$

NFFT-based fast summation ([Potts, Steidl 2004])

Fast computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$:

0. $\hat{b}_{\boldsymbol{\ell}} := \sum_{\mathbf{j} \in \{-M,\dots,M\}^d} \mathcal{K}\left(\frac{\mathbf{j}}{M}\right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{j}/M}$, $\boldsymbol{\ell} \in \{-M,\dots,M\}^d$, by d-dim. FFT
1. Compute (nonequispaced adjoint) DFT:
   $\hat{c}_{\boldsymbol{\ell}} := \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j}$ for all $\boldsymbol{\ell} \in \{-M,\dots,M\}^d$
2. Multiply Fourier coefficients: $\hat{f}_{\boldsymbol{\ell}} := \hat{b}_{\boldsymbol{\ell}} \, \hat{c}_{\boldsymbol{\ell}}$ for all $\boldsymbol{\ell} \in \{-M,\dots,M\}^d$
3. Compute (nonequispaced) DFT:
   $(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\dots,M\}^d} \hat{f}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$ for all $i = 1,\dots,n$

$$(\mathbf{W}\cdot\boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\dots,M\}^d} \hat{b}_{\boldsymbol{\ell}} \left(\sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_j} \right) \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\ell}\cdot\mathbf{x}_i}$$

NFFT-based fast summation ([Potts, Steidl 2004])

Fast computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$:

0. $\hat{b}_{\boldsymbol{\ell}} := \sum_{\mathbf{j} \in \{-M,\ldots,M\}^d} \mathcal{K}\left(\frac{\mathbf{j}}{M}\right) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{j}/M}$, $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$, by d-dim. FFT
1. Compute (nonequispaced adjoint) DFT:
   $\hat{c}_{\boldsymbol{\ell}} := \sum_{j=1}^{n} \alpha_j \, \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_j}$ for all $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$
2. Multiply Fourier coefficients: $\hat{f}_{\boldsymbol{\ell}} := \hat{b}_{\boldsymbol{\ell}} \, \hat{c}_{\boldsymbol{\ell}}$ for all $\boldsymbol{\ell} \in \{-M, \ldots, M\}^d$
3. Compute (nonequispaced) DFT:
   $(\mathbf{W} \cdot \boldsymbol{\alpha})_i \approx \sum_{\boldsymbol{\ell} \in \{-M,\ldots,M\}^d} \hat{f}_{\boldsymbol{\ell}} \, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\ell} \cdot \mathbf{x}_i}$ for all $i = 1, \ldots, n$

Nonequispaced Fast Fourier Transform (NFFT):
NFFT3 software library (github.com/NFFT/nfft) by Keiner, Kunis, Potts

- Fast computation of 1. and 3. using approximative algorithm NFFT

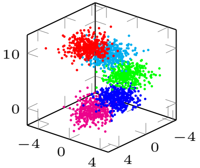
$\Rightarrow$ Computation of $\mathbf{W} \cdot \boldsymbol{\alpha}$ requires only $\mathcal{O}(n)$ runtime (for fixed accuracy)

Fast computation of eigenpairs of graph Laplacian

Compute k eigenvectors belonging to k-smallest eigenvalues of $\mathbf{L}$ using NFFT-based fast summation (Lanczos algorithm, MATLAB eigs)

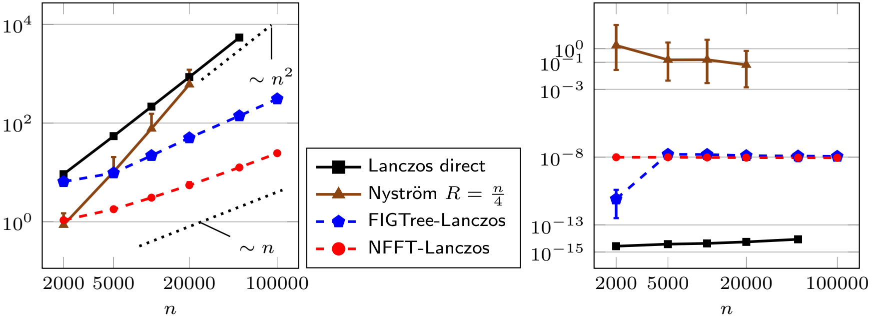
Computation of 10 largest eigenvalues of $\mathbf{I} - \mathbf{L}$ and corresponding eigenvectors:

Spiral data set example



Comparison of runtimes

Comparison of eigenvector accuracies



Outline

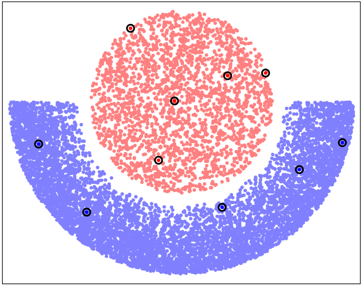
- (Fully-Connected) Graphs and Graph Laplacian
- NFFT-based fast summation
- Application to Learning (Classification)
  - Semi-Supervised
  - Unsupervised

Semi-Supervised learning – Kernel method

- Learn model based on labeled training data and remaining unlabelled data
- Benefits from cluster recognition and given training labels

Training data encoded in vector $\mathbf{f} \in \mathbb{R}^n$:

$$f_i = \begin{cases} 1 & \text{if node } i \text{ has label of class 1,} \\ -1 & \text{if node } i \text{ has label of class 2,} \\ 0 & \text{if label of node } i \text{ is unknown.} \end{cases}$$



- Ansatz: $\min\limits_{\mathbf{u} \in \mathbb{R}^n} \left(\|\mathbf{u} - \mathbf{f}\|^2 + \beta \mathbf{u}^T \mathbf{L} \mathbf{u} \right)$, $\beta \geq 0$ regularization parameter
- Compute $\mathbf{u}$ by solving $(\mathbf{I} + \beta \mathbf{L}) \cdot \mathbf{u} = \mathbf{f}$ via conjugate gradient method

$\Rightarrow$ Assign class labels based on the sign of entries of $\mathbf{u}$

Semi-Supervised learning – Kernel method

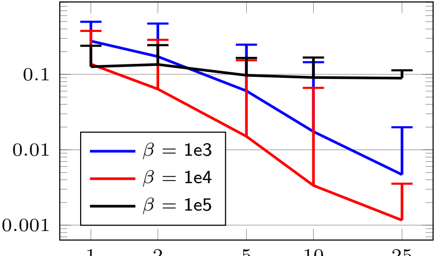
- Learn model based on labeled training data and remaining unlabelled data
- Benefits from cluster recognition and given training labels

Training data encoded in vector $\mathbf{f} \in \mathbb{R}^n$:

Example crescent-fullmoon, $n = 100\,000$

$$f_i = \begin{cases} 1 & \text{if node } i \text{ has label of class 1,} \\ -1 & \text{if node } i \text{ has label of class 2,} \\ 0 & \text{if label of node } i \text{ is unknown.} \end{cases}$$

Misclassification rate (avg. & max.)



- Ansatz: $\min_{\mathbf{u} \in \mathbb{R}^n} \left(\|\mathbf{u} - \mathbf{f}\|^2 + \beta \mathbf{u}^T \mathbf{L} \mathbf{u} \right)$, $\beta \geq 0$ regularization parameter
- Compute $\mathbf{u}$ by solving $(\mathbf{I} + \beta \mathbf{L}) \cdot \mathbf{u} = \mathbf{f}$ via conjugate gradient method

$\Rightarrow$ Assign class labels based on the sign of entries of $\mathbf{u}$

Unsupervised learning – Clustering: Find clusters among unlabelled data points

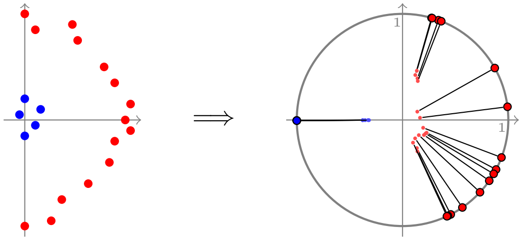
Spectral clustering (k classes):

1. Compute k eigenvectors $\mathbf{v}_\ell$ belonging to the k-smallest eigenvalues of $\mathbf{L}$ (Lanczos algorithm, MATLAB eigs) and put them in the columns of matrix
$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_k \end{bmatrix} \in \mathbb{R}^{n \times k}$$
2. Compute spectral points as normalized rows of $\mathbf{V}$:
$$\tilde{\mathbf{v}}_i = \frac{\begin{pmatrix} V_{i1} & \cdots & V_{ik} \end{pmatrix}^T}{\left\| \begin{pmatrix} V_{i1} & \cdots & V_{ik} \end{pmatrix} \right\|} \in \mathbb{R}^k, \qquad i = 1, \ldots, n$$
3. Use standard clustering tool `kmeans` on $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_n$

Unsupervised learning – Clustering: Find clusters among unlabelled data points

Spectral clustering (k classes):

1. Compute k eigenvectors $\mathbf{v}_\ell$ belonging to the k-smallest eigenvalues of $\mathbf{L}$ (Lanczos algorithm, MATLAB eigs) and put them in the columns of matrix
$$\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_k\end{bmatrix} \in \mathbb{R}^{n\times k}$$
2. Compute spectral points as normalized rows of $\mathbf{V}$:
$$\tilde{\mathbf{v}}_i = \frac{\begin{pmatrix}V_{i1} & \cdots & V_{ik}\end{pmatrix}^T}{\left\|\begin{pmatrix}V_{i1} & \cdots & V_{ik}\end{pmatrix}\right\|} \in \mathbb{R}^k, \qquad i = 1, \ldots, n$$
3. Use standard clustering tool `kmeans` on $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_n$



Unsupervised learning – Spectral clustering: Results

Image source: TU Chemnitz/Wolfgang Thieme

800x533 pixels RGB

$n = 426\,400$, $d = 3$, $\sigma = 90$

$k = 2$ classes

$k = 4$ classes

NFFT-based Lanczos method: 25 sec. eigenvector computation, (+ 18 sec. kmeans)

Intel Core i7 CPU 970 (3.20 GHz) @ 1 thread

---

without fast method: 31 hours on Intel Xeon E7-4880 CPUs (2.50 GHz) @ 32 threads

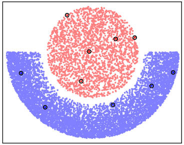
Conclusion

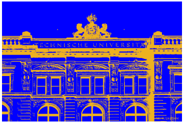
- Fully connected graph with node features
- NFFT-based fast summation

$\Rightarrow$ fast matrix-vector products with graph Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$

$\Rightarrow$ Enormous speed-up

- Various applications in learning
  - Semi-supervised: Kernel method
  - Unsupervised: Spectral clustering



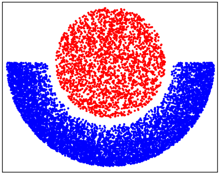


D. Alfke, D. Potts, M. Stoll, T. V.
NFFT Meets Krylov Methods: Fast Matrix-Vector Products for the Graph Laplacian of Fully Connected Networks.
Front. Appl. Math. Stat. 4:61, 2018. DOI: 10.3389/fams.2018.00061

Example Code:
`https://www.tu-chemnitz.de/mathematik/wire/codes.php`

NFFT - Nonequispaced FFT: `https://github.com/NFFT/nfft`
`https://www.tu-chemnitz.de/~potts/nfft/`

Types of learning (for classification): Supervised



Supervised learning

- Learn model (parameters) based on a training dataset with known class labels
- Apply (learned) model to data points (with unknown class labels)
- Data points (to be classified) not required for training
- Sufficient training data covering all relevant cases required

Example methods: Trees, Linear Regression, SVM, Kernel Ridge Regression, NNs

Supervised learning – Kernel Ridge Regression

- Training data: n data points with feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ with desired output $f_i \in \mathbb{R}$ $(i = 1, \ldots, n)$
- Goal: find (parameters of) prediction function $F : \mathbb{R}^d \to \mathbb{R}$,
  $F(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x} - \mathbf{x}_j)$ with $f_i \approx F(\mathbf{x}_i)$
- Build fully connected adjacency matrix $\mathbf{W}$, e.g. use $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$:
  $w_{ij} = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$
- Ansatz: $\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left(\|\mathbf{f} - \mathbf{W}\,\boldsymbol{\alpha}\|_2^2 + \beta\, \boldsymbol{\alpha}^\top \mathbf{W}\, \boldsymbol{\alpha} \right)$, $\beta \geq 0$ regularization parameter
- Compute coefficients $\boldsymbol{\alpha}$ by solving $(\mathbf{W} + \beta \mathbf{I}) \cdot \boldsymbol{\alpha} = \mathbf{f}$ via Conjugate Gradient
- Prediction function for new data points $\mathbf{x} \in \mathbb{R}^d$: $F(\mathbf{x})$

Supervised learning – Kernel Ridge Regression

- Training data: n data points with feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ with desired output $f_i \in \mathbb{R}$ $(i = 1, \ldots, n)$
- Goal: find (parameters of) prediction function $F : \mathbb{R}^d \to \mathbb{R}$,
  $F(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x} - \mathbf{x}_j)$ with $f_i \approx F(\mathbf{x}_i)$
- Build fully connected adjacency matrix $\mathbf{W}$, e.g. use $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$:
  $w_{ij} = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$
- Ansatz: $\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left(\|\mathbf{f} - \mathbf{W}\,\boldsymbol{\alpha}\|_2^2 + \beta\,\boldsymbol{\alpha}^\top \mathbf{W}\,\boldsymbol{\alpha} \right)$, $\beta \geq 0$ regularization parameter
- Compute coefficients $\boldsymbol{\alpha}$ by solving $(\mathbf{W} + \beta \mathbf{I}) \cdot \boldsymbol{\alpha} = \mathbf{f}$ via Conjugate Gradient
- Prediction function for new data points $\mathbf{x} \in \mathbb{R}^d$: $F(\mathbf{x})$

Supervised learning – Kernel Ridge Regression

- Training data: n data points with feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ with desired output $f_i \in \mathbb{R}$ $(i = 1, \ldots, n)$
- Goal: find (parameters of) prediction function $F : \mathbb{R}^d \to \mathbb{R}$, $F(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x} - \mathbf{x}_j)$ with $f_i \approx F(\mathbf{x}_i)$
- Build fully connected adjacency matrix $\mathbf{W}$, e.g. use $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$: $w_{ij} = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$
- Ansatz: $\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left(\|\mathbf{f} - \mathbf{W}\,\boldsymbol{\alpha}\|_2^2 + \beta\,\boldsymbol{\alpha}^\top \mathbf{W}\,\boldsymbol{\alpha} \right)$, $\beta \geq 0$ regularization parameter
- Compute coefficients $\boldsymbol{\alpha}$ by solving $(\mathbf{W} + \beta \mathbf{I}) \cdot \boldsymbol{\alpha} = \mathbf{f}$ via Conjugate Gradient
- Prediction function for new data points $\mathbf{x} \in \mathbb{R}^d$: $F(\mathbf{x})$

Supervised learning – Kernel Ridge Regression

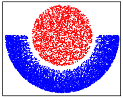
- Training data: n data points with feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ with desired output $f_i \in \mathbb{R}$ $(i = 1, \dots, n)$
- Goal: find (parameters of) prediction function $F : \mathbb{R}^d \to \mathbb{R}$, $F(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x} - \mathbf{x}_j)$ with $f_i \approx F(\mathbf{x}_i)$
- Build fully connected adjacency matrix $\mathbf{W}$, e.g. use $\mathcal{K}(\mathbf{x}) = \mathrm{e}^{-\|\mathbf{x}\|^2/\sigma^2}$: $w_{ij} = \mathrm{e}^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}$
- Ansatz: $\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left(\|\mathbf{f} - \mathbf{W}\,\boldsymbol{\alpha}\|_2^2 + \beta\,\boldsymbol{\alpha}^\top \mathbf{W}\,\boldsymbol{\alpha} \right)$, $\beta \geq 0$ regularization parameter
- Compute coefficients $\boldsymbol{\alpha}$ by solving $(\mathbf{W} + \beta\mathbf{I}) \cdot \boldsymbol{\alpha} = \mathbf{f}$ via Conjugate Gradient
- Prediction function for new data points $\mathbf{x} \in \mathbb{R}^d$: $F(\mathbf{x})$

Supervised learning – Kernel Ridge Regression: Results

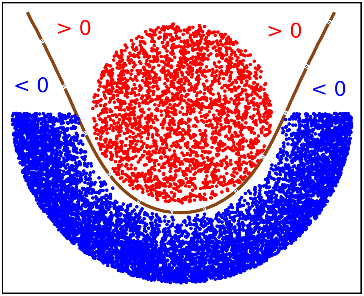
Example: Classification of 2D feature vectors into two classes

- Desired output for training nodes:

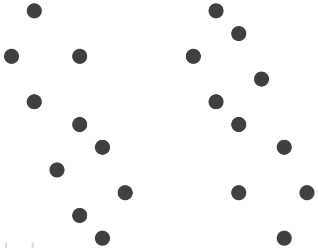
$$f_i = \begin{cases} 1 & \text{if training node } i \text{ is labelled for class } 1, \\ -1 & \text{if training node } i \text{ is labelled for class } 2. \end{cases}$$



- Classify a new data point $\mathbf{x}$ based on the sign of $F(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \, \mathcal{K}(\mathbf{x} - \mathbf{x}_j)$



Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

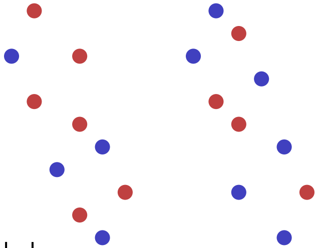
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

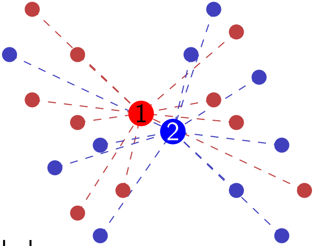
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – kmeans



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

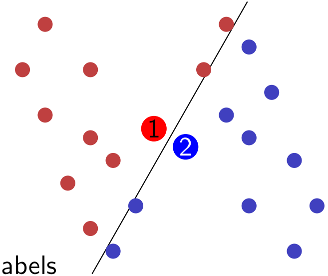
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

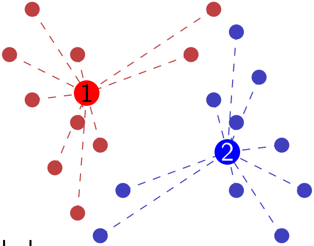
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

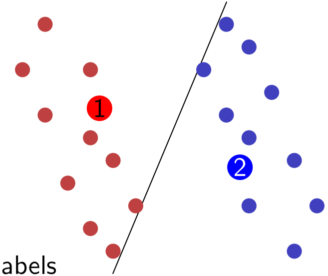
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

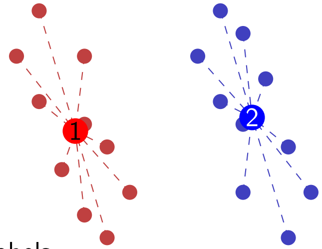
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

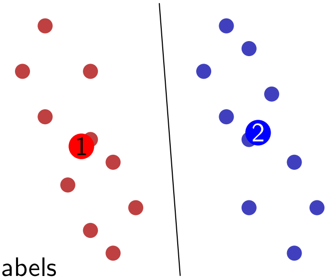
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

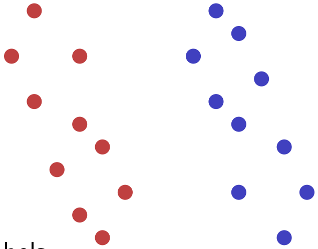
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



Step 1: Assign random labels

Step 2: Compute cluster centers

Step 3: Assign labels of closest center

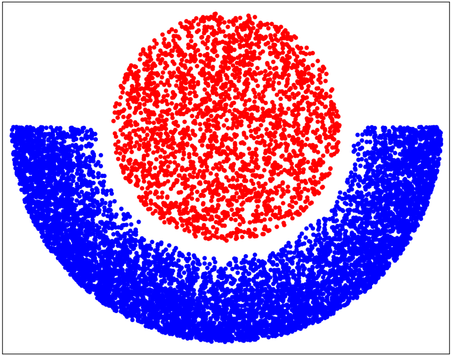
Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center

Repeat step 2: Compute cluster centers

Repeat step 3: Assign labels of closest center → no label changes → finished

Unsupervised learning without graph tools – `kmeans`



?

Unsupervised learning – Fiedler vector clustering

Fiedler vector $\mathbf{v}$: Eigenvector of $\mathbf{L}$ to the smallest non-zero eigenvalue

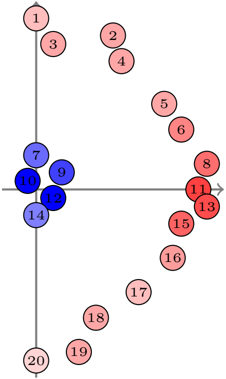
Example: fully connected graph for simple data set with 2-dim. feature vectors

Simple algorithm: Classify nodes according to sign of Fiedler vector entries

Unsupervised learning – Fiedler vector clustering

Fiedler vector $\mathbf{v}$: Eigenvector of $\mathbf{L}$ to the smallest non-zero eigenvalue

Example: fully connected graph for simple data set with 2-dim. feature vectors



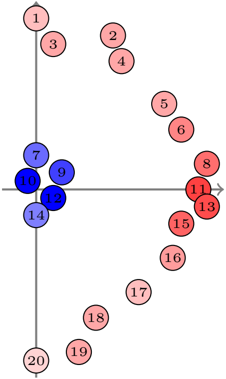
Simple algorithm: Classify nodes according to sign of Fiedler vector entries

Unsupervised learning – Fiedler vector clustering

Fiedler vector $\mathbf{v}$: Eigenvector of $\mathbf{L}$ to the smallest non-zero eigenvalue

Example: fully connected graph for simple data set with 2-dim. feature vectors

1	0.12
2	0.14
3	0.14
4	0.14
5	0.14
6	0.17
7	−0.33
8	0.19
9	−0.35
10	−0.38
11	0.23
12	−0.38
13	0.22
14	−0.32
15	0.20
16	0.15
17	0.12
18	0.14
19	0.14
20	0.10



Simple algorithm: Classify nodes according to sign of Fiedler vector entries