TECHNISCHE UNIVERSITÄT CHEMNITZ

Faculty of Mathematics
Scientific Computing, Applied Functional Analysis, Applied Analysis
Dominik Alfke, Daniel Potts, Martin Stoll, Toni Volkmer

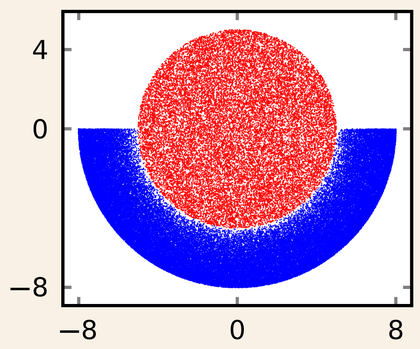
NFFT meets Krylov methods:

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

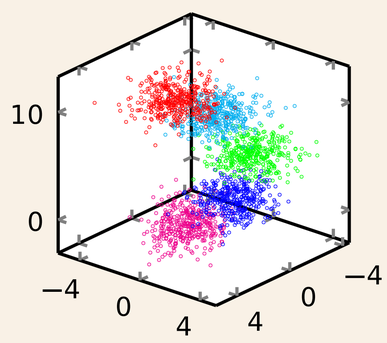
Introduction

The graph Laplacian is a standard tool in data science, machine learning, and image processing. For densely populated networks, matrix-vector products with the graph Laplacian are a hard task. A typical application is the computation of a number of its eigenvalues and eigenvectors. Standard methods become infeasible as the number of nodes in the graph is too large. We propose the use of the fast summation based on the nonequispaced fast Fourier transform (NFFT) to perform the dense matrix-vector products with the graph Laplacian fast without ever forming the whole matrix. The enormous flexibility of the NFFT algorithm allows for fast computations for a variety of kernels and applications.

Crescent-fullmoon example[1] with $n = 100\,000$ points



Spiral data set example[2] with $n = 2\,000$

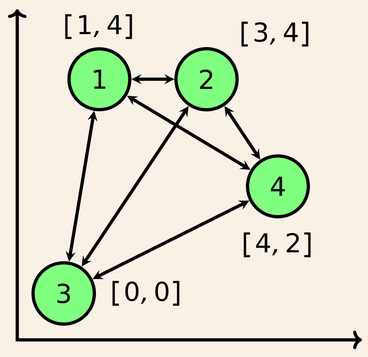


[1] https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets
[2] https://sites.google.com/site/kittipat/matlabtechniques

Fully connected networks and graph Laplacian

- each node $j = 1, \dots, n$ is associated with a feature vector $\mathbf{v}_j \in \mathbb{R}^d$
- weights $w_{ji} = K(\mathbf{v}_j - \mathbf{v}_i)$ for $j \neq i$, $w_{jj} = 0$, weight matrix $\mathbf{W} = (w_{ji})$
  - large entries for similar features, small entries for dissimilar features
- e.g. Gaussian RBF kernel $K(\mathbf{v}) = \exp\left(-\|\mathbf{v}\|^2/\sigma^2\right)$
- node degree $d_j = \sum_{i=1}^{n} w_{ji}$
  (sum of all weights of edges connected to node j)
- degree matrix $\mathbf{D} = \operatorname{diag}(d_1, \dots, d_n) = \operatorname{diag}(\mathbf{W}\,\mathbf{1})$
- graph Laplacian, unnormalized: $\mathbf{L} = \mathbf{D} - \mathbf{W}$,
  symmetric normalized: $\mathbf{L_s} = \mathbf{I} - \mathbf{D}^{-1/2}\,\mathbf{W}\,\mathbf{D}^{-1/2} =: \mathbf{I} - \mathbf{A}$

Graph sketch



Weight matrix $\mathbf{W}$

	1	2	3	4
1	0	0.89	0.62	0.70
2	0.89	0	0.50	0.87
3	0.62	0.50	0	0.57
4	0.70	0.87	0.57	0

Graph Laplacian $\mathbf{L_s}$

	1	2	3	4
1	1	−0.40	−0.32	−0.32
2	−0.40	1	−0.25	−0.40
3	−0.32	−0.25	1	−0.30
4	−0.32	−0.40	−0.30	1

Supervised learning

Kernel ridge regression

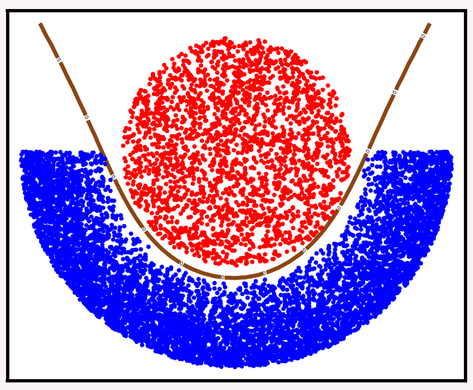
ansatz: $F(\mathbf{v}) = \sum_{i=1}^{n} \alpha_i\, K(\mathbf{v} - \mathbf{v}_i)$,

$\operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^n} \|\mathbf{f} - \mathbf{K}\boldsymbol{\alpha}\|_2^2 + \beta\,\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$

where $\mathbf{K} = \left(K(\mathbf{v}_j - \mathbf{v}_i)\right)_{j,i=1}^{n}$

$\Longrightarrow \boldsymbol{\alpha} = (\mathbf{K} + \beta\,\mathbf{I})^{-1}\,\mathbf{f}$

Crescent-fullmoon example $n = 10\,000$:



Semi-Supervised Learning

Kernel method

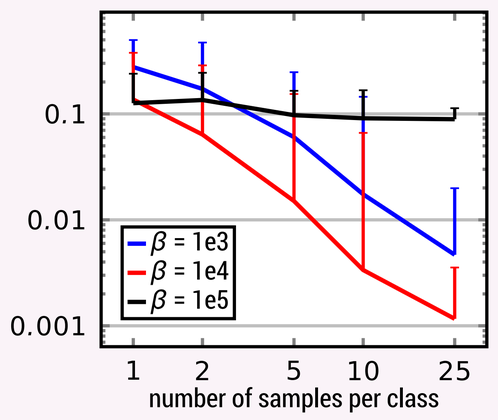
$\operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^n} \|\mathbf{u} - \mathbf{f}\|_2^2 + \beta\,\mathbf{u}^\top \mathbf{L_s}\,\mathbf{u}$,

$\mathbf{u} = (\mathbf{I} + \beta\,\mathbf{L_s})^{-1}\,\mathbf{f}$.

$f_j = \pm 1$ for training data (2 classes),
$f_j = 0$ otherwise.

Crescent-fullmoon example $n = 100\,000$:

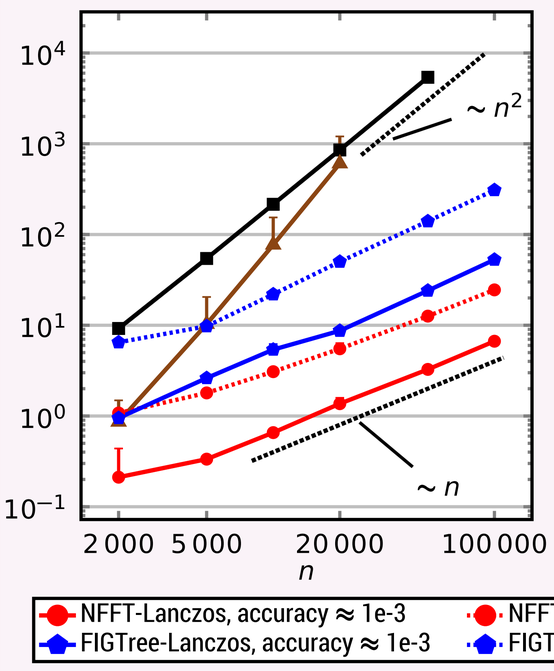
Misclassification rate



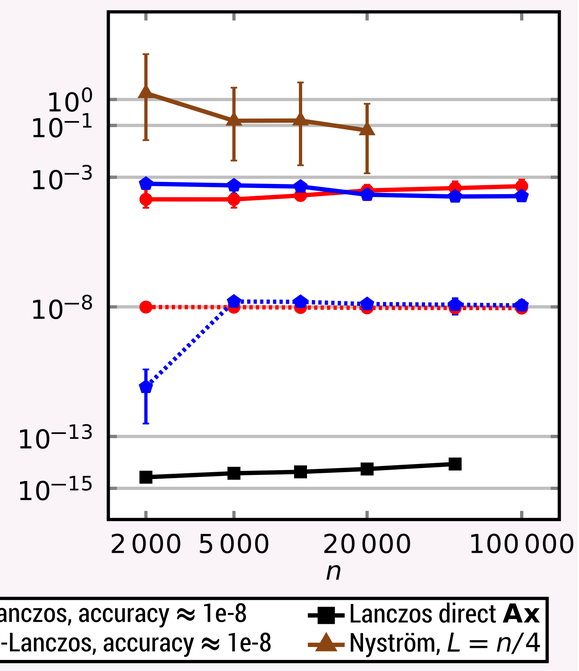
Eigenvalue and eigenvector computation

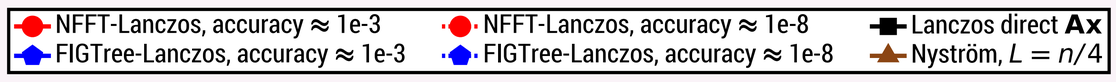
Compute 10 largest eigenvalues of $\mathbf{A} = \mathbf{I} - \mathbf{L_s}$ and corresponding eigenvectors of spiral data set

Comparison of runtimes



Comparison of eigenvector accuracies





Algorithm 1: Fast summation $\mathbf{K}\,\mathbf{x} = (\mathbf{W} + K(\mathbf{0})\,\mathbf{I})\,\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$

1. Apply d-dim. NFFT$^{\mathrm{H}}$ on $\mathbf{x}$ and obtain $\hat{x}_{\mathbf{h}} \approx \sum_{i=1}^{n} x_i\, \mathrm{e}^{-2\pi\mathrm{i}\mathbf{h}\mathbf{v}_i} \quad \forall \mathbf{h} \in \mathcal{I}_N$,
2. Multiply result by Fourier coefficients $\left(\hat{b}_{\mathbf{h}}\right)_{\mathbf{h}\in\mathcal{I}_N}$ of approximation of K and obtain $\hat{f}_{\mathbf{h}} := \hat{b}_{\mathbf{h}}\,\hat{x}_{\mathbf{h}} \quad \forall \mathbf{h} \in \mathcal{I}_N$.
3. Apply d-dim. NFFT on $\left(\hat{f}_{\mathbf{h}}\right)_{\mathbf{h}\in\mathcal{I}_N}$ and obtain output $(\mathbf{K}\,\mathbf{x})_j \approx \mathrm{Re}\left(\sum_{\mathbf{h}\in\mathcal{I}_N} \hat{f}_{\mathbf{h}}\, \mathrm{e}^{2\pi\mathrm{i}\mathbf{h}\mathbf{v}_j}\right) \quad \forall j = 1, \dots, n$.

Algorithm 2: Fast computation of $\mathbf{A}\,\mathbf{x} = (\mathbf{I} - \mathbf{L_s})\,\mathbf{x}$

0. Choose internal algorithm parameters, scale $\|\mathbf{v}_j\| < 1/4$, adjust σ, compute $\mathbf{D} = \operatorname{diag}(\mathbf{K}\,\mathbf{1} - K(\mathbf{0})\,\mathbf{1})$ via Algorithm 1.
1. Compute $\mathbf{z} = \mathbf{K}\left(\mathbf{D}^{-1/2}\,\mathbf{x}\right)$ via Algorithm 1.
2. Compute $\mathbf{y} = \mathbf{D}^{-1/2}\left(\mathbf{z} - K(\mathbf{0})\,\mathbf{D}^{-1/2}\,\mathbf{x}\right)$.
3. Rescale $\mathbf{y}$ and obtain output $\mathbf{y} \approx \mathbf{A}\,\mathbf{x} = (\mathbf{I} - \mathbf{L_s})\,\mathbf{x}$.

Complexity: $\mathcal{O}(n)$ for fixed accuracy.

Fast matrix-vector multiplication, eigenvalues

- smallest eigenvalues of $\mathbf{L_s}$ $\hat{=}$ largest eigenvalues of $\mathbf{A}$ (identical eigenvectors)
- use Lanczos-based eigenvalue routine on $\mathbf{A}$ via Algorithm 2

Example code: https://mytuc.org/khqm

Contact

Dominik Alfke
dominik.alfke@math.tu-chemnitz.de
tu-chemnitz.de/mathematik/wire/people/alfke.php

Daniel Potts
daniel.potts@math.tu-chemnitz.de
tu-chemnitz.de/~potts

Martin Stoll
martin.stoll@math.tu-chemnitz.de
tu-chemnitz.de/mathematik/wire/

Toni Volkmer
toni.volkmer@math.tu-chemnitz.de
tu-chemnitz.de/~tovo

See also the references in

D. Alfke, D. Potts, M. Stoll, T. Volkmer. 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