GSPBOX: A toolbox for signal processing on graphs
Nathanael Perraudin, Johan Paratte, David I Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst & David K.Hammond
Ecole Polytechnique Fédérale de Lausanne (EPFL), LTS2
Website: https://lts2.epfl.ch/gsp/

Abstract
The Graph Signal Processing toolbox (GSPBox) is a MATLAB/R open-source toolbox designed for graph signal processing and data mining tasks such as filtering, denoising, prediction, classification, data representation and visualization. Its purpose is to serve as a tool for accelerating new scientific developments in a reproducible research perspective.

We propose an overview of the current features of the toolbox: graph construction, graph operators, graph learning, filter design, spectral filtering methods, graph reduction, bindings with the optimization toolbox UNLocBox, etc. In order to prepare future collaborations between different research graphs, we additionally present the modules that are currently under development and will be released in the near future.

1 The box

The general design of the GSPBox focuses around the graph object [1], a structure containing the necessary information to use most of the algorithms.

Toolbox features

- MATLAB and Python libraries
- Efficient implementations of a large set of graph signal processing algorithms
- Documented, maintained and regularly tested
- Fast development at the state of the art of the graph signal processing field
- Binded with the UNLocBox to solve your graph regularized problems

2 Graph

To initialize a graph from a weight matrix W, use

```matlab
G = gsp_from_matrix(W);
```

Alternatively, the toolbox contains a lot of synthetic graphs and an optimized nearest-neighbor graph function.

```matlab
G = gsp_nearest_graph_matrix(W, k);
```

Finally, if you do not possess any coordinates, you can build a graph using learning methods.

```matlab
x = G.U[:, 1]
```

All these functions initialize the graph structure with the arguments inside Table 1.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of nodes</td>
</tr>
<tr>
<td>M</td>
<td>Number of edges</td>
</tr>
<tr>
<td>D</td>
<td>Degree matrix</td>
</tr>
<tr>
<td>L</td>
<td>Laplacian matrix</td>
</tr>
<tr>
<td>X</td>
<td>Vertex signal vector</td>
</tr>
<tr>
<td>W</td>
<td>Weight matrix</td>
</tr>
<tr>
<td>W^−1/2</td>
<td>Normalized Laplacian</td>
</tr>
</tbody>
</table>

Table 1: Attributes of the graph object

In order to speed-up computation with MATLAB, an optional field can be pre-computed:

```matlab
G = gsp_compute_fourier_basis(G, x);
```

3 Operators

The most central operator in graph signal processing is the Laplacian. It is stored in G.L.

Based on the Laplacian, the toolbox is able to perform

- Fourier transform [M]: gsp_fft
- Kron reduction [M]: gsp_kron_reduce
- Gradient computation [M]: gsp_grad
- Multi-resolution analysis using a pyramid transform [M]: gsp_pyramid_analysis

4 Filters

Filters are central in graph signal processing. They are implemented as:

```matlab
y = gsp_fft(x)
```

The toolbox contains a large set of predefined designs such as:

- Wavelet (Filters are scaled version of a mother window) [M]: gsp_design_mexican_hat & gsp_design_abalone
- Gabor (Filters are shifted version of a mother window) [M]: gsp_design_gabor
- Low pass filter (Filters to de-noise a signal) [M]: gsp_design_exponential

5 Plotting

The toolbox contains a few plotting functions

```matlab
figure(1); gsp_plot_signal(G, x); title('Original signal');
```

Demonstration in 7 steps

1. Start the code
2. Create a graph
3. Compute the Fourier basis
4. Create a smooth signal with noise
5. Filter the signal
6. Plot the results
7. Save the results

Use it as a black-box

- MATLAB code
- Python code

References


Acknowledgements

We would like to thank all coding authors of the GSPBox. The toolbox was written in Python by Basic Childbin, Alexandre Lalanne and Nicolas Red.

The toolbox was also improved by Nauman Shahid and Yann Schönenberger. The work has been supported by the Swiss National Science Foundation research project Towards Signal Processing on Graphs, grant number: 200021_154356/1.

Figure 1: Visualization of graph and signals using plotting functions.

6 Python

The Python port of the library works similarly. Each package described here is a module of the library. Graph functions are in [P]: pygsp.graphs, filters in [P]: pygsp.filters, operators in [P]: pygsp.ops and so on.

All mathematical operations are performed with matrices using numpy and scipy libraries. Plotting requires either matplotlib or pygraph to be installed.

7 Help

Starting with the GSPBox

1. Get a free version online [M]: https://lts2.epfl.ch/gsp
2. Do the tutorial [M]: Run gsp_demo [P]: https://lts2.epfl.ch/pygsp/tutorials
3. Get help from the documentation, the article [2], or by contacting gspbox-support@groupes.epfl.ch

If you need additional functions, please ask.

- Machine learning / Optimization
- Clustering
- Low rank extraction
- Hyperspectrals
- Bi-graphs, vertex-time signal processing

```matlab
G = gsp_from_matrix(W);
```

```python
G = gsp_from_matrix(W)
```