Advances in Deep Learning on Graphs

Michaël Defferrard

Joint work with Xavier Bresson, Alexandre Cherqui, Frank de Morsier, Nathanaël Perraudin, Tomasz Kacprzak, Andreas Loukas, Pierre Vandergheynst.

École Polytechnique Fédérale de Lausanne

Outline

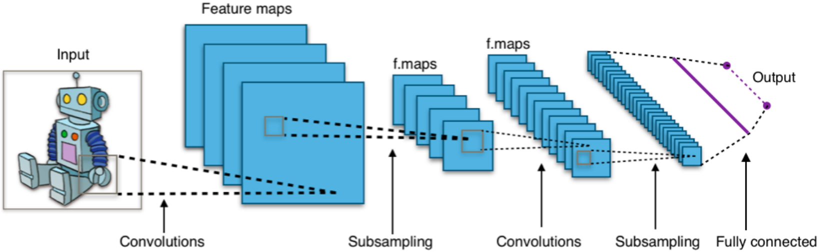
Deep Learning on Graphs

Applications

Current Challenges and Future Work

Convolutional Neural Networks

Main benefit (over MLPs): they **exploit the structure** of the data.



Key properties:

- Convolutional: translation invariance (stationarity).
- Localized: deformation stability & compact filters (independent of input size n).
- Multi-scale: hierarchical features extracted by multiple layers (compositionality).
- $\mathcal{O}(n)$ computational complexity.

ConvNets on graphs

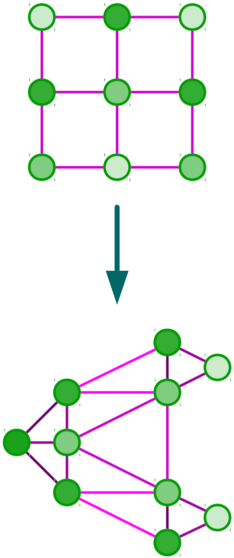
Graphs vs Euclidean grids:

- Irregular sampling.
- Weighted edges.
- No orientation or ordering (in general).

Ingredients:

- Convolution (local)
- Non-linearity (point-wise)
- Down-sampling (global / local)
- Pooling (local)

Challenge: efficient formulation of convolution and down-sampling on graphs.



Convolution on Graph, the GSP way

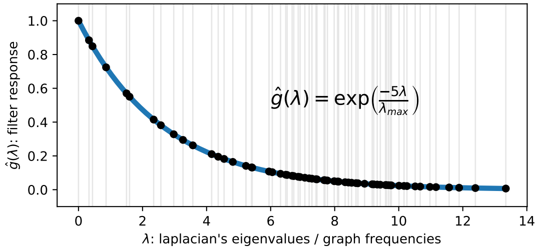
$$y = x *_{\mathcal{G}} g = U \begin{bmatrix} \hat{g}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_n) \end{bmatrix} U^T x = U\hat{g}(\Lambda)U^T x = \hat{g}(L)x$$

- Combinatorial $L = D - W$ or normalized $L = I_n - D^{-1/2}WD^{-1/2}$ Laplacian.
- The eigendecomposition of the Laplacian $L = U\Lambda U^T \in \mathbb{R}^{n\times n}$ gives eigenvectors u_k and eigenvalues λ_k. $U = [u_1, \dots, u_n] \in \mathbb{R}^{n\times n}$ forms the graph Fourier basis and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ are graph "frequencies".
- Fourier Transform: $\hat{x} = \mathcal{F}_{\mathcal{G}}\{x\} = U^T x \in \mathbb{R}^n$
- Inverse Fourier Transform: $x = \mathcal{F}_{\mathcal{G}}^{-1}\{\hat{x}\} = U\hat{x} = UU^T x = x$
- Convolution theorem: $y = x *_{\mathcal{G}} g = U\left(U^T g \odot U^T x\right) = U\left(\hat{g} \odot U^T x\right)$

Spectral filtering of graph signals

Non-parametric filter, can learn any filter (n degrees of freedom):

$$\hat{g}_\theta(\Lambda) = \text{diag}(\theta), \;\; \theta \in \mathbb{R}^n$$



- Non-localized in vertex domain
- Learning complexity is $\mathcal{O}(n)$
- Computational complexity is $\mathcal{O}(n^2)$ (& memory)

Polynomial parametrization

$$\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k = \sum_{k=0}^{K-1} \tilde{\theta}_k T_k(\tilde{\Lambda}), \quad \tilde{\Lambda} = \frac{2}{\lambda_n}\Lambda - I_n$$

Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$
with $T_0 = 1$ and $T_1 = x$

- Can learn any K-localized filter.
- Allows a distributed implementation: only access the K-neighborhood.

- K-localized
- Learning complexity is $\mathcal{O}(K)$
- Computational complexity is $\mathcal{O}(K|\mathcal{E}|)$ (same as classical ConvNets!)

Fast implementation by recursion

$$y = \hat{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x = \sum_{k=0}^{K-1} \theta_k \bar{x}_k, \quad \tilde{L} = \frac{2}{\lambda_n} L - I_n$$

Recurrence:

$$\begin{aligned} \bar{x}_k &= T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2} \\ \bar{x}_1 &= \tilde{L}x \\ \bar{x}_0 &= x \end{aligned}$$

- Can be implemented as an accumulator.
- Any polynomial can be used. They all have the same representative power. Optimization difficulty might vary.
- Any matrix can be used instead of the Laplacian L, including the adjacency matrix, or even a non-symmetric adjacency or "Laplacian".
- The learned filter parameters θ can be transferred across graphs (i.e. used with different L).

Spatial vs Spectral

In the end, almost all formulations are spatial.

Our formulation is **spectrally motivated**.

$$y = U \hat{g}_\theta(\Lambda) U^\intercal x$$

In the absence of an $O(n \log n)$ Fast Fourier Transform (FFT), which only exists for specific domains, that is however too expensive with $O(n^3)$ operations.

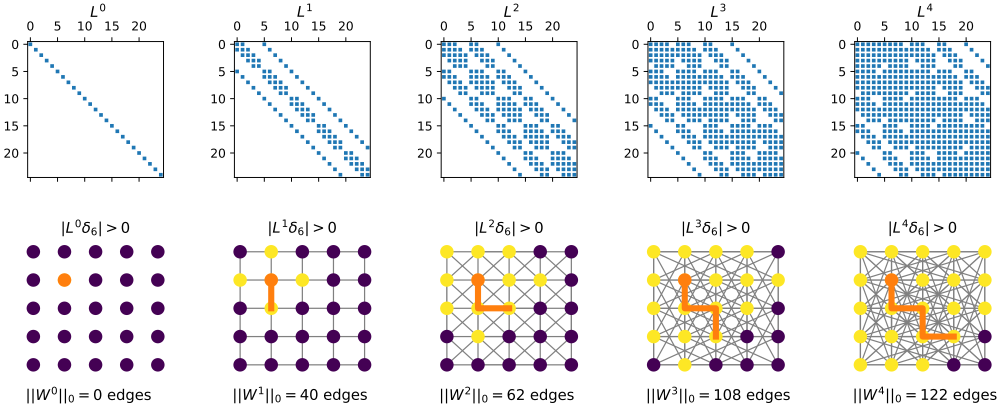
With polynomials, the **implementation is spatial**.

$$y = \hat{g}_\theta(L)x = \sum_k \theta_k L^k x = \sum_k \tilde{\theta}_k T_k(\tilde{L})x$$

Many papers get this wrong and imply that an eigendecomposition of the Laplacian or adjacency matrix is needed.

Filter localization

- Value at j of g_θ centered at i: $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$
- $d_\mathcal{G}(i,j) > K$ implies $(L^K)_{i,j} = 0$



Outline

Deep Learning on Graphs

Applications

Current Challenges and Future Work

Multiple kinds of problems

Graphs which model discrete relations

- Social networks
- Graph of citations or hyperlinks
- Molecules
- Knowledge graphs

Graphs which represent sampled manifolds

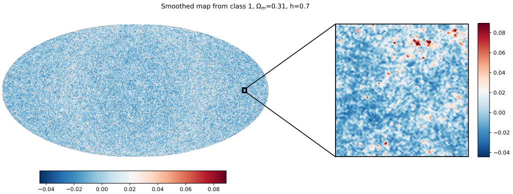
- Meshes
- Point clouds
- Data on spheres (planets, sky)
- Traffic on roads

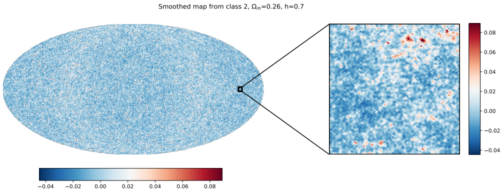
Problems:

- Node classification or regression (e.g. semi-supervized learning)
- Graph classification or regression
- Signal classification or regression $\to$ what I'm most interested about

Cosmology: Data & Problem

- Cosmologists devise models of how the universe works.
- We only get to observe one real universe.
- Problem: which simulation is closest to the real thing? A signal classification task.

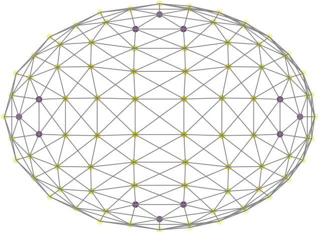




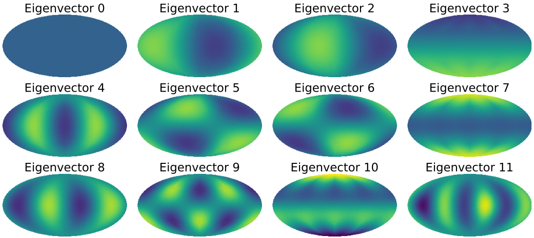
Two mass maps generated from different cosmological parameters.

Cosmology: Graph

- Data lives on the sky, a sphere.
- The sphere is discretized, and can be represented by a graph.
- Numerous kind of spherical sky maps in cosmology and astrophysics.
  Cosmic microwave background, galaxy clustering, gravitational lensing.

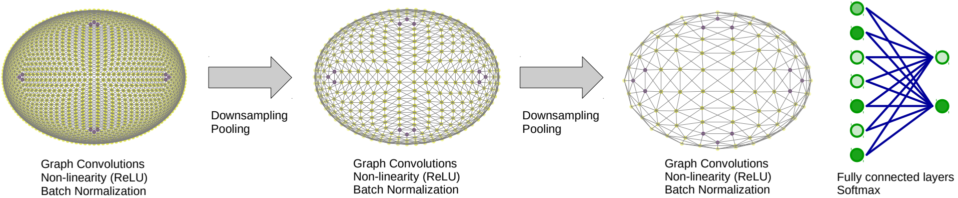


Sphere discretized by graph.

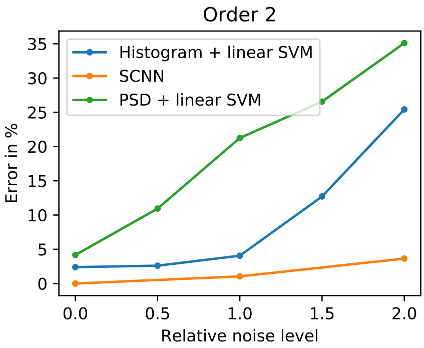


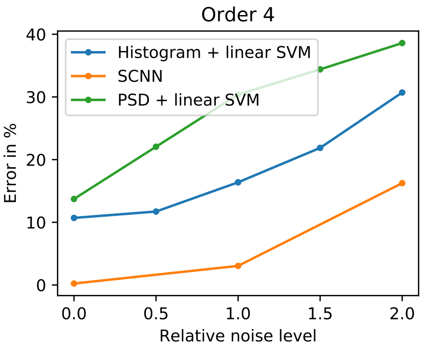
Fourier modes resemble spherical harmonics.

A classical ConvNet, but on graph.



Cosmology: Results





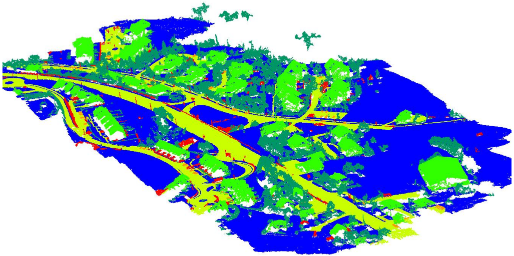
Standard benchmarks in cosmology:

- Histogram of values.
- Power spectral density.

Point Cloud Segmentation: Data & Problem

- Drones take aerial pictures of the ground.
- Each point is photographed multiple times from different point-of-views.
- Point cloud constructed by photogrammetry.
- Problem: assign a class to each point, a node classification task.

x,y,z coordinates with RGB features

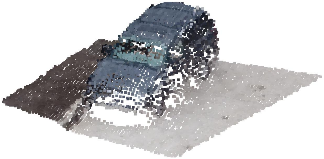


class labels

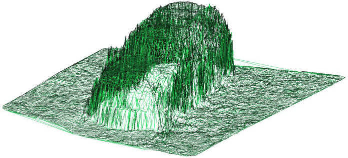
Point Cloud Segmentation: Graph

A graph gives:

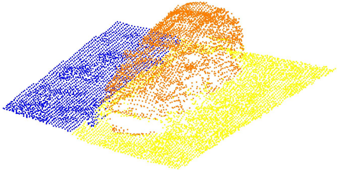
- Neighborhood information, needed for consistent labeling.
- A support, needed for efficient computation.



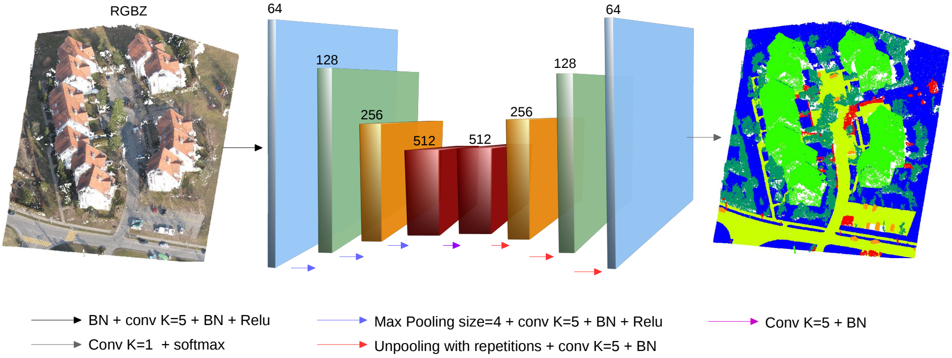
RGB features



Graph



Labels



Characteristics:

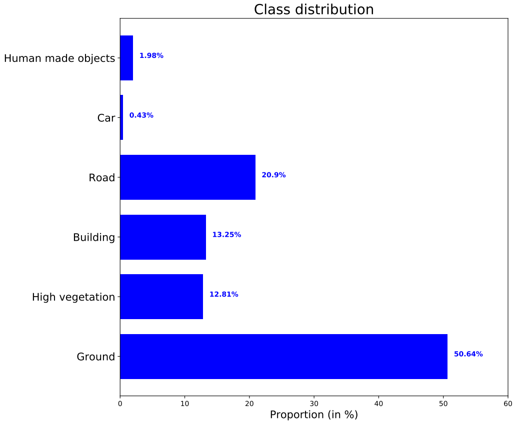
- Dense prediction.
- *Reason* at multiple scales.

Main difficulties:

- Large number of points.
- Training samples are of varying sizes.

Point Cloud Segmentation: Results

Model	Accuracy Overall (micro)	Accuracy Mean (macro)
Random Forest	75%	52%
Graph ConvNet	83%	68%



Point Cloud Segmentation: Results

Random forest baseline

True label \ Predicted label	Ground	High veg.	Building	Road	Car	Human made obj.
Ground	807919	111645	20379	20821	620	4548
High veg.	99011	134143	6184	2899	200	1159
Building	19336	8995	198616	22489	1716	6586
Road	42386	4866	75775	366655	2342	15912
Car	1891	260	5030	2175	1412	636
Human made obj.	12538	3360	11867	9200	436	3923

Graph ConvNet

True label \ Predicted label	Ground	High veg.	Building	Road	Car	Human-made obj.
Ground	812954	90364	24305	28224	1146	8939
High veg.	24180	210247	3780	2377	15	2997
Building	8065	7025	226223	10405	310	5710
Road	25005	7214	51894	418984	945	3894
Car	220	291	2562	2191	3894	2246
Human-made obj.	6878	9103	8950	2528	1325	12540

Outline

Deep Learning on Graphs

Applications

Current Challenges and Future Work

The need to consider multiple scales

Most signals on large graphs exhibit **patterns at multiple scales**.

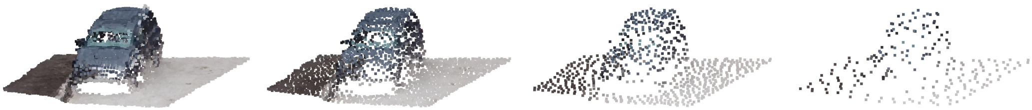
Some filters thus need to have larger receptive fields to capture longer-range dependencies. This can be achieved by:

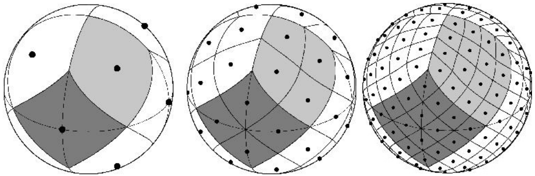
1. increasing the size of the filters (the polynomial order),
2. increasing the number of layers,
3. down-sampling the domain (pooling).

While we can easily do (1) and (2), it can drastically increase the number of parameters to learn. For now, we don't yet have a generic and functional approach to (3).

Coarsening

Graph coarsening is certainly an answer to the down-sampling problem.





- Feature or structure-based coarsening can be used when the sampling is regular.
- It is however much harder on non-regular graph (with power-law degree distributions and hubs), like social networks.

Conclusion

Successes:

- Convolution operation mostly solved (many formulations have been proposed for specific tasks) and understood (with multiple interpretations, including message-passing, local aggregation function, attention).
- The framework can be applied to many problems.

Challenges:

- Multiple scales, down-sampling, coarsening.
- Unified framework.
- Better knowledge of method - problem fit.

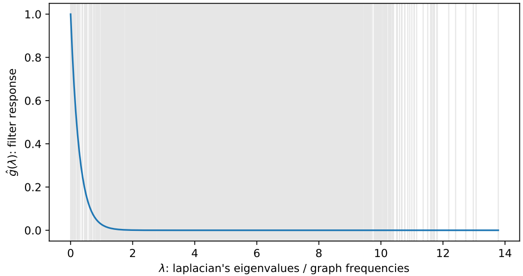
Last year I told the audience that DL was coming to GSP. This year I think it has been realized, with many of you gaining interest in DL and many ML researchers gaining interest in GSP.

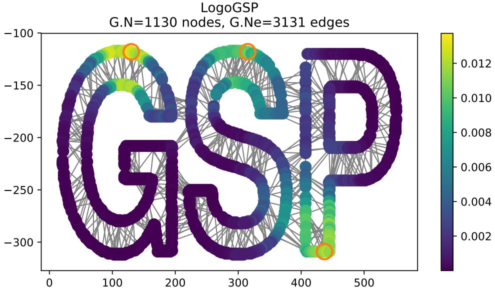
PyGSP: Graph Signal Processing in Python

```
import numpy as np
import matplotlib.pyplot as plt

G = graphs.Logo()
G.compute_fourier_basis()
g = filters.Heat(G, tau=50)
g.plot()

DELTAS = [20, 30, 1090]
s = np.zeros(G.N)
s[DELTAS] = 1
s = g.filter(s)
G.plot_signal(s, highlight=DELTAS)
```





Slides `https://doi.org/10.5281/zenodo.1286818`

Paper Defferrard, Bresson and Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS, 2016.

Code `https://github.com/mdeff/cnn_graph`

Paper Seo, Defferrard, Bresson and Vandergheynst, Structured Sequence Modeling with Graph Convolutional Recurrent Networks, arXiv, 2017.

Code `https://github.com/youngjoo-epfl/gconvRNN`

GSP in Python `https://github.com/epfl-lts2/pygsp`

Thanks Questions?