Applied Machine Learning Days
AI & Networks track

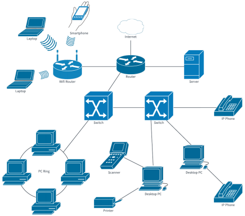
Lausanne, 2019-01-28

Learning on Graphs

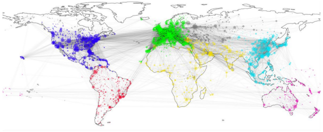
Michaël Defferrard

École Polytechnique Fédérale de Lausanne

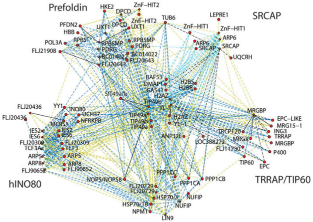
Networks and graphs



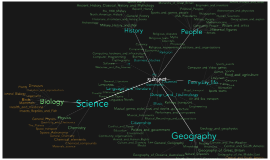
communication



transportation (flights)



protein interaction



hyperlinks

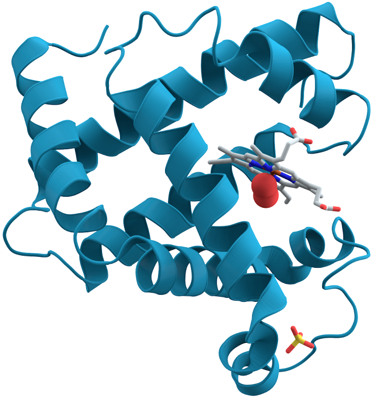
transportation (roads)

More:

- brain networks
- social networks

Image sources: 1, 2, 3, 4, 5

Motivation

$x =$ 

$$y = f(x) = \begin{cases} \text{“toxic”} \\ \text{“non-toxic”} \end{cases}$$

$$y = f(x) = 80\% \text{ toxic}$$

Goal: learn the **unknown** function f, using both **structure** and **features**.

Structure and features

Structure: graph (or network)

- Graph: a set of nodes (vertices) and a set of pairwise relations (edges)
- Relations: interactions, similarity, geometry

Features: data on the graph (or signal)

- Features: set of characteristics (or properties) about each node

Traditional ML uses features only. Our goal is to combine features and structure!

Using the structure

Extrinsic: embed the graph in an Euclidean space.

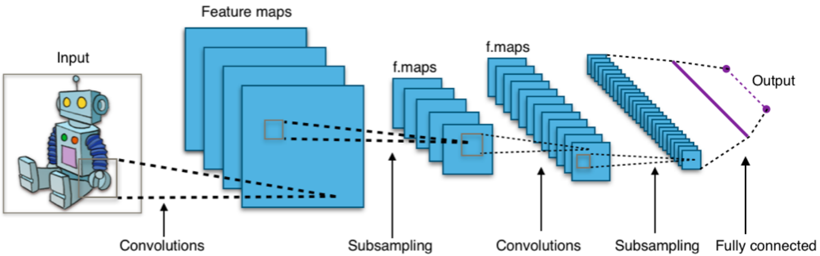
- Compute or learn a vector representation of each node.
- Use that embedding as additional features for a classifier.

Intrinsic: a Neural Net defined on graphically structured data.

- Exploit geometric structure for learning and computational efficiency.
- Starting point: ConvNet, an intrinsic formulation for Euclidean grids.

Convolutional Neural Networks

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



Key properties:

- Convolutional: translation equivariance (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

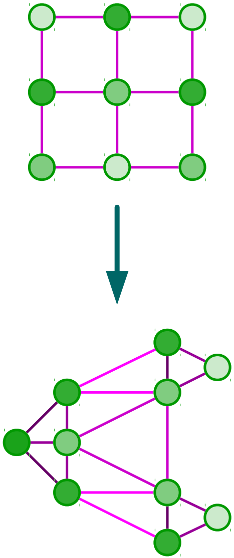
Graphs vs Euclidean grids:

- Irregular sampling.
- Weighted edges.
- No orientation or ordering (in general)
  $\rightarrow$ permutation invariance.

Ingredients:

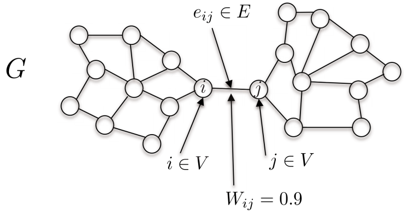
- Convolution (local)
- Non-linearity (point-wise)
- Down-sampling (global / local)
- Pooling (local)

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



Notation

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$: undirected and connected graph



- $\mathcal{V}$: set of $|\mathcal{V}| = n$ vertices
- $\mathcal{E}$: set of edges
- $W \in \mathbb{R}^{n \times n}$: weighted adjacency matrix
- $D_{ii} = \sum_j W_{ij}$: diagonal degree matrix

Graph Laplacians (core operator to spectral graph theory):

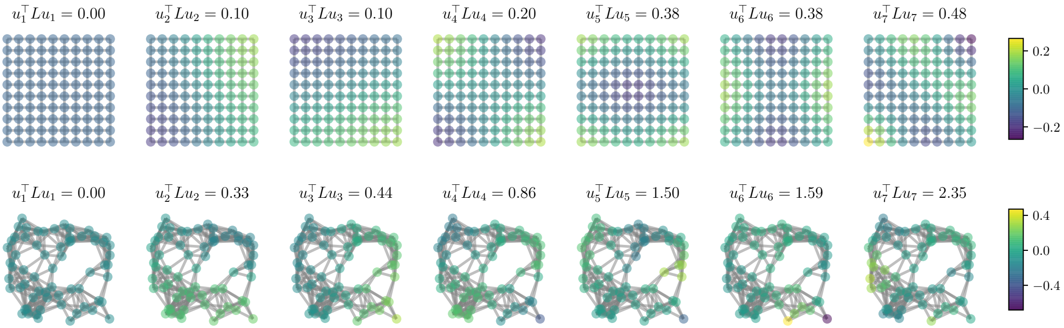
- combinatorial Laplacian $L = D - W \in \mathbb{R}^{n \times n}$
- normalized Laplacian $L = I_n - D^{-1/2} W D^{-1/2} \in \mathbb{R}^n$

Graph Fourier basis

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

Definition: the Fourier basis diagonalizes the Laplacian operator $\rightarrow L = U \Lambda U^\top$

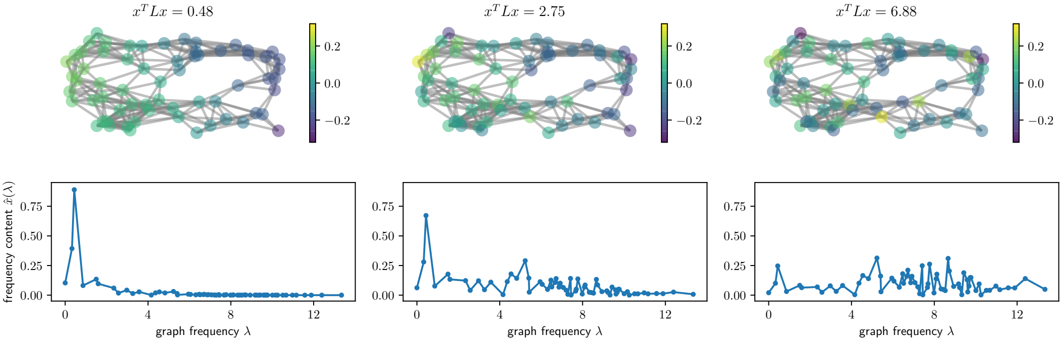
- Graph Fourier basis $U = [u_1, \dots, u_n] \in \mathbb{R}^{n \times n}$
- Graph "frequencies" $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n) = U^\top L U \in \mathbb{R}^{n \times n}$



Graph Fourier Transform

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

- Graph signal $x : \mathcal{V} \to \mathbb{R}$ seen as $x \in \mathbb{R}^n$
- Transform: $\hat{x} = \mathcal{F}_\mathcal{G}\{x\} = U^\top x \in \mathbb{R}^n$
- Inverse: $x = \mathcal{F}_\mathcal{G}^{-1}\{x\} = U\hat{x} = UU^\top x = x$



Filtering

kernel a function $g : \mathbb{R} \to \mathbb{R}$ that defines the action of the filter

filter an operator acting on signals represented by $g(L)$

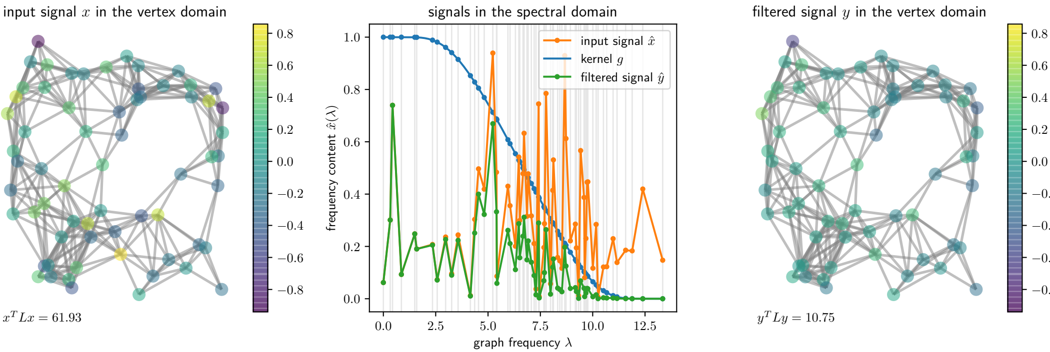
A signal $x \in \mathbb{R}^{|\mathcal{V}|}$ is filtered by the kernel g as:

$$y = g(L)x = Ug(\Lambda)U^\top x$$

Step by step

1. take the Fourier transform: $\hat{x} = U^\top x$
2. take an element-wise product with the kernel evaluated at the eigenvalues: $\hat{y} = (g(\lambda_1), \dots, g(\lambda_{|\mathcal{V}|})) \odot \hat{x}$
3. take the inverse Fourier transform: $y = U\hat{y}$

Example



Observation: the *low-pass filtered* signal y is much smoother than x!

Filter design

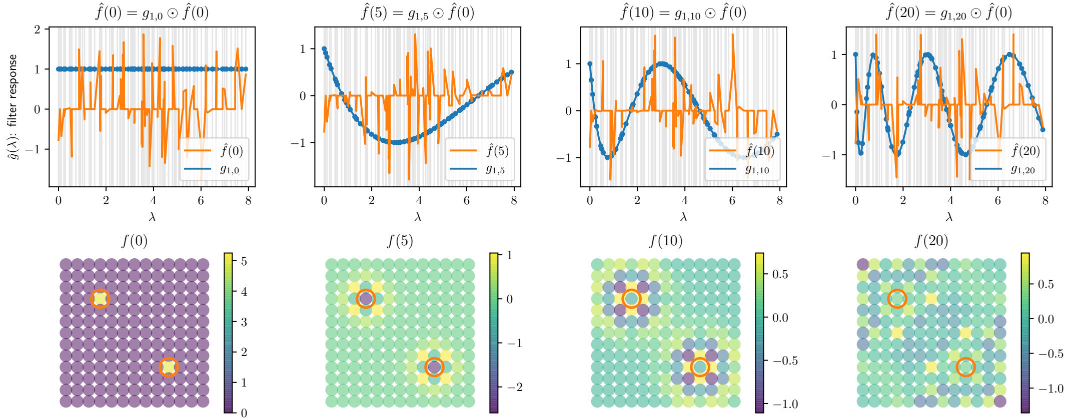
Task: design a kernel $g : \mathbb{R} \to \mathbb{R}$ such that $y = g(L)x$ is the solution of something interesting.

Examples

- Heat diffusion: $g_{\tau t}(\lambda) = \exp(-\tau t \lambda)$
- Wave propagation: $g_{\tau t}(\lambda) = \cos\left(t \arccos\left(1 - \frac{\tau^2}{2}\lambda\right)\right)$
- Projection on a subspace: $g(\lambda) = \begin{cases} 1 & \text{if } \lambda_{min} < \lambda < \lambda_{max}, \\ 0 & \text{otherwise.} \end{cases}$
- Denoising with $\arg\min_y \|y - x\|_2^2 + \tau y^\top L y$: $g(\lambda) = \frac{1}{1+\tau\lambda}$

Example: wave propagation

$$-\tau^2 L f(t) = \partial_{tt} f(t) \quad \Rightarrow \quad f(t) = g_{\tau t}(L) f(0) \text{ with } g_{\tau t}(\lambda) = \cos\left(t \arccos\left(1 - \frac{\tau^2}{2}\lambda\right)\right)$$



Learning

What if we don't know the process by which y depends on x, and can't derive g?
Answer: learn the kernel from examples.

Task: approximate the optimal unknown mapping $y = g(L)x$ by a parameterized approximation $y \approx \tilde{y} = g_\theta(L)x$, where θ are the parameters to be learned.

We got:

- a set of examples $\{(x_n, y_n)\}_{n=1}^N$, hopefully large enough
- a cost function to measure how good our approximation is,
  for example $c(\tilde{y}, y) = \|\tilde{y} - y\|_2^2$

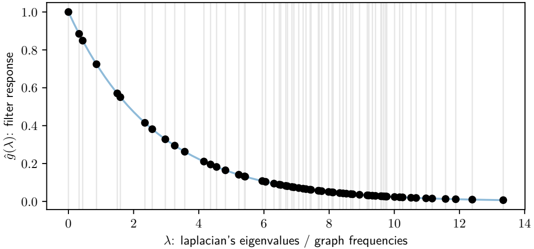
Goal: $\hat{\theta} = \arg\min_\theta \mathbf{E}_{(x,y)}[c(g_\theta(L)x, y)]$

Kernel parameterization

Defferrard, Bresson, and Vandergheynst 2016

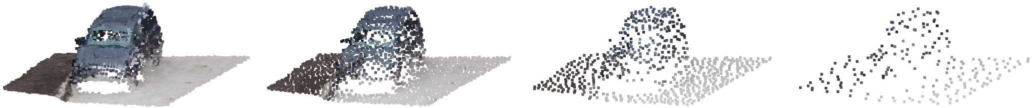
Non-parametric filter, can learn any filter (n degrees of freedom):

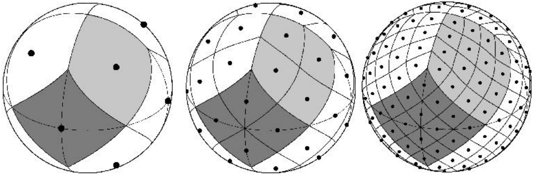
$$g_\theta(\Lambda) = \operatorname{diag}(\theta),\ \theta \in \mathbb{R}^n \ \Rightarrow\ y = U \operatorname{diag}(\theta) U^\top x$$



Coarsening: hierarchical representation

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



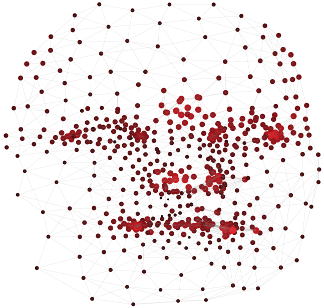


- Easy and well-defined when the domain has a hierarchical structure.

Learned coarsening: an attention mechanism

Defferrard and Loukas 2018

hard combinatorial problem $\Rightarrow$ learn a **continuous relaxation** of the operation



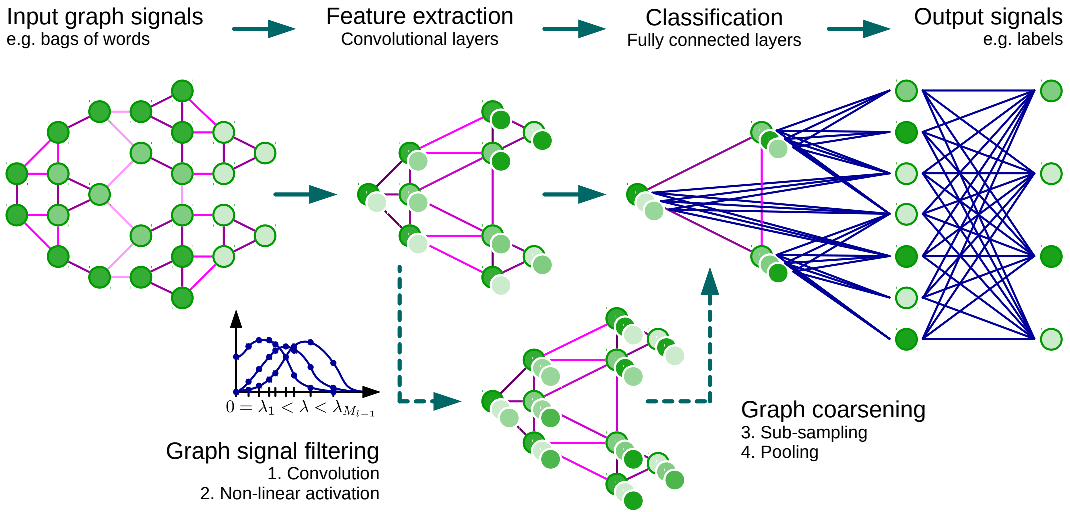
Conditioned on:

1. the structure
2. the features
3. the task

introspection!

Graph ConvNet architecture

Defferrard, Bresson, and Vandergheynst 2016



Multiple kinds of problems: combination of data and tasks

Graphs that model discrete relations

- Social networks
- Graph of citations or hyperlinks
- Molecules (proteins)
- Knowledge graphs

Graphs that represent sampled manifolds

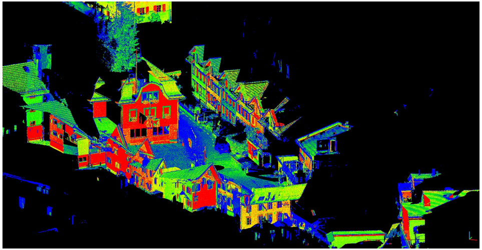
- Meshes (shapes, surfaces)
- Point clouds
- Data on spheres (planets, sky)
- Traffic on roads

Tasks:

- Node classification or regression (semi-supervized learning)
- Graph classification or regression
- Signal classification or regression

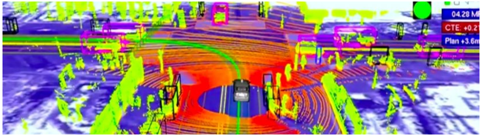
Application: segmentation of point clouds

remote sensing / surveying



outdoor mapping

indoor mapping



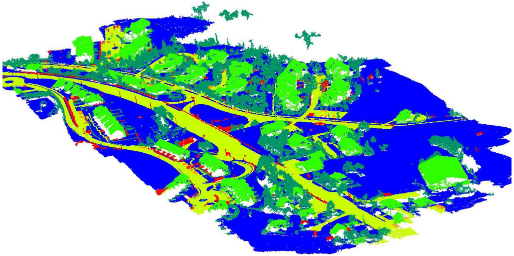
autonomous driving

Data

input a set of features associated to a set of points

output a label associated to each point

x,y,z coordinates with RGB colors



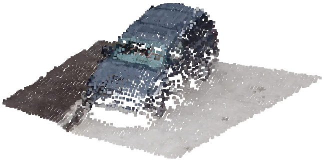
class labels

Graph

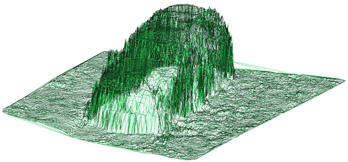
Cherqui, Morsier, and Defferrard 2018

A graph gives:

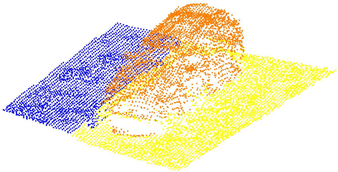
- Neighborhood information, needed for consistent labeling.
- A support, needed for efficient computation.



RGB features



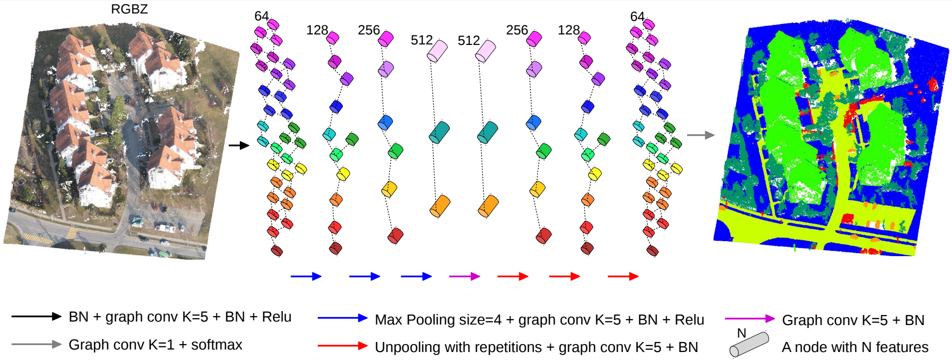
graph



labels

Model

Cherqui, Morsier, and Defferrard 2018



Characteristics:

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

Main difficulties:

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

Conclusion

Filters can be **designed** to solve known problems.
If the transformation is unknown, **learn** filters from examples.

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).
- Applications to many scientific and industrial problems

Challenges:

- Multiple scales, down-sampling, coarsening.
- Better understanding of the method – problem fit.