Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Objective

Data Science. Analyze and extract information for decision-making from large-scale and high-dimensional **unstructured** data.

1. Extand convolutional neural networks to graph-structured data.
2. Reproduce the breakthrough of ConvNets beyond Euclidean data!

Challenge: Efficiently formulate convolution and down-sampling on graphs.
Contribution: Generalization to graphs with same computational complexity.

Convolutional Neural Networks

ConvNets are extremely efficient at leveraging statistical properties of data, in particular stationarity and compositionality through local statistics.

Ingredients, well defined and efficient on Euclidean grids

1. Convolution → translate filter, fast Fourier transform (FFT)
2. Down-sampling → pick one pixel out of n
3. Non-linearity
4. Pooling

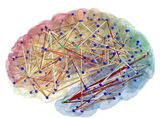
Non-Euclidean Data Structured with Graphs

Modeling versatility: graphs model heterogeneous pairwise relationships.

Graph-structured data

- Social networks: Facebook, Twitter.
- Biological networks: genes, molecules, brain connectivity.
- Infrastructure networks: energy, transportation, Internet, telephony.

Social network

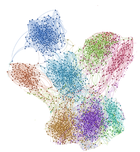


Brain structure

Telecommunication

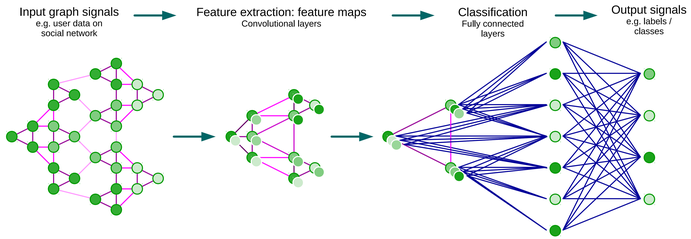
Constructed graphs

- Graph between samples, useful for semi-supervised learning.
- Graph between features, useful to reduce computational complexity.



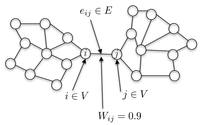
Alternative approach:

1. Embed nodes in an Euclidean space.
2. Use that embedding as features.



Spectral Graph Theory

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



- $\mathcal{V}$: 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 Laplacian

- Combinatorial: $L = D - W \in \mathbb{R}^{n \times n}$
- Normalized: $L = I_n - D^{-1/2} W D^{-1/2}$

L is symmetric and positive semidefinite → $L = U \Lambda U^T$ (eigendecomposition)

- Graph Fourier basis $U = [u_0, \dots, u_{n-1}] \in \mathbb{R}^{n \times n}$
- Graph "frequencies" $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{R}^{n \times n}$

Graph Fourier Transform

1. Graph signal $x : \mathcal{V} \to \mathbb{R}$ seen as $x \in \mathbb{R}^n$.
2. Spectral representation $\hat{x} = \mathcal{F}_\mathcal{G}\{x\} = U^T x \in \mathbb{R}^n$.
3. Inverse: $x = U\hat{x} = UU^T x = x$.

Graph Convolution

Convolution theorem:

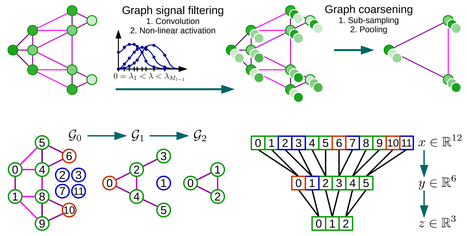
$$x *_\mathcal{G} g = U \left(U^T g \odot U^T x \right) = U \left(\hat{g} \odot U^T x \right)$$

Conveniently written as:

$$x *_\mathcal{G} g = U \, \mathrm{diag}\left(\hat{g}(\lambda_1), \dots, \hat{g}(\lambda_n) \right) U^T x = U \hat{g}(\Lambda) U^T x = \hat{g}(L) x$$

Graph Coarsening and Pooling

1. Coarsening (sub-sampling) with balanced cut models, using efficient greedy approximations (Graclus, Metis).
2. Parallel pooling (as 1D pooling) with coarsened graphs arranged as binary tree.



Learning Fast Localized Spectral Filters

Spectral filtering of graph signals: $y = \hat{g}_\theta(L)x = U\hat{g}_\theta(\Lambda)U^T x$

Non-parametric filter $\hat{g}_\theta(\Lambda) = \mathrm{diag}(\theta),\ \theta \in \mathbb{R}^n$

Non-localized Learning complexity in $\mathcal{O}(n)$ Computations & memory in $\mathcal{O}(n^2)$

Polynomial parametrization $\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k,\ \theta \in \mathbb{R}^K$

- 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 $(L^K)_{i,j} = 0$

K-localized Learning complexity in $\mathcal{O}(K)$ Computational complexity in $\mathcal{O}(n^2)$

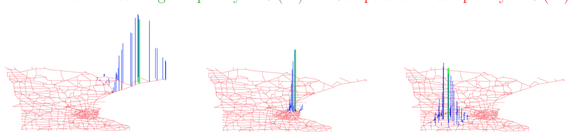


Figure: Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$.

Recursive Formulation for Fast Filtering $\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}),\ \ \tilde{\Lambda} = 2\Lambda/\lambda_n - I_n$

- Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$
- Filtering: $y = \hat{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x$
- Recurrence: $y = \hat{g}_\theta(L)x = [\bar{x}_0, \dots, \bar{x}_{K-1}]\theta$, $\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ with $\bar{x}_0 = x$ and $\bar{x}_1 = \tilde{L}x$

K-localized Learning complexity in $\mathcal{O}(K)$ Computational complexity in $\mathcal{O}(K|\mathcal{E}|)$

Learning Filters $y_{s,j} = \sum_{i=1}^{F_{in}} \hat{g}_{\theta_{i,j}}(L) x_{s,i} \in \mathbb{R}^n$

- $x_{s,i}$: feature map i of sample s, $\theta_{i,j}$: $F_{in} \times F_{out} \times K$ trainable parameters
- Gradients for backprop: $\frac{\partial E}{\partial \theta_{i,j}} = \sum_{s=1}^{S} [\bar{x}_{s,i,0}, \dots, \bar{x}_{s,i,K-1}]^T \frac{\partial E}{\partial y_{s,j}}$, $\frac{\partial E}{\partial x_{s,i}} = \sum_{j=1}^{F_{out}} g_{\theta_{i,j}}(L) \frac{\partial E}{\partial y_{s,j}}$

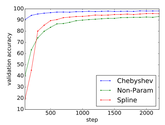
Overall cost of $\mathcal{O}(K|\mathcal{E}|F_{in}F_{out}S)$ operations

Results: Sanity Check on MNIST

1. Comparable to classical ConvNets and better than other parametrizations!
2. Isotropic filters → rotation invariance.

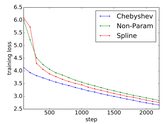
Model	Architecture	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	99.33
Proposed graph CNN	GC32-P4-GC64-P4-FC512	99.14

Table: Comparison to classical ConvNets on MNIST (grid graph).



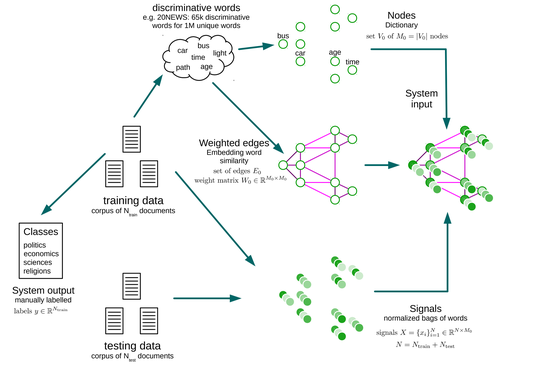
	Accuracy		
Architecture	Non-Param	Spline	Chebyshev
GC10	95.75	97.26	97.48
GC32-P4-GC64-P4-FC512	96.28	97.15	99.14

Table: Comparison between spectral filters, $K = 25$.



Results: Documents Classification on 20NEWS

1. Structuring documents as bag-of-words on vocabulary graph.
2. Make graph ConvNets practical!



Model	Accuracy
Linear SVM	65.90
Multinomial Naive Bayes	68.51
Softmax	66.28
FC2500	64.64
FC2500-FC500	65.76
GC32	68.26