MotifNet: a Motif-based GCNN for Directed Graphs

Federico Monti[1,2], Karl Otness[5], Michael M. Bronstein[1,2,3,4,5]

USI	Fabula AI	TAU	Intel	Harvard
Switzerland	UK	Israel	Israel	USA

IEEE DSW2018, 6 June 2018, Lausanne

MotifNet: a Motif-based GCNN for Directed Graphs

Federico Monti[1,2], Karl Otness[5], Michael M. Bronstein[1,2,3,4,5]

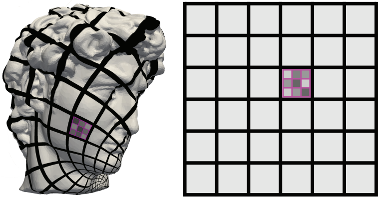
USI Switzerland | Fabula AI UK | TAU Israel | Intel Israel | Harvard USA

IEEE DSW2018, 6 June 2018, Lausanne

Different formulations of CNN on graphs

Spectral domain[1,2,3]

Spatial domain[4,5,6]



Embedding domain[7,8]

[1]Bruna et al. 2014; [2]Henaff, Bruna, LeCun 2015; [3]Defferrard, Bresson, Vandergheynst 2016; [4]Masci et al. 2015; [5]Boscaini et al. 2016; [6]Monti et al. 2017; [7]Sinha, Bai, Ramani 2016; [8]Maron et al. 2017

Directed graphs

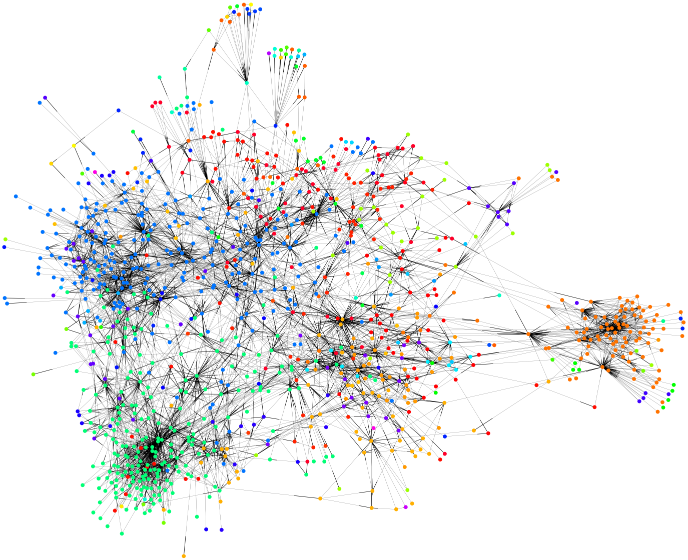


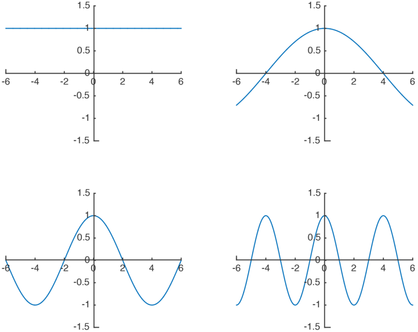
Figure: Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann 2017

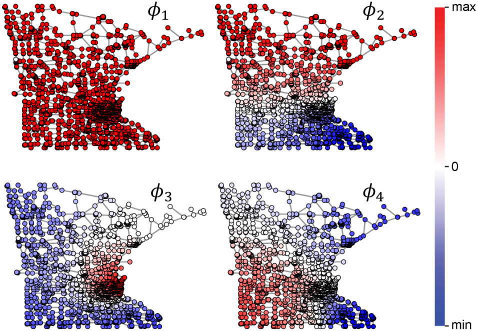
Graph Convolutional Neural Networks

Laplacian eigenfunctions

$$\Delta e^{j\omega x} = \frac{d^2}{dx^2} e^{j\omega x} = -\omega^2 e^{j\omega x}$$

$$\mathbf{\Delta} = \mathbf{D} - \mathbf{W} = \mathbf{\Phi} \mathbf{\Lambda} \mathbf{\Phi}^{\mathrm{T}}$$





Convolution Theorem

- Given two functions $f, h : Z \to \mathbb{R}$ their convolution is a function

$$(f \star h)(x) = \sum_{x'=-K/2}^{K/2} f(x - x')h(x') = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}$$

Convolution Theorem

- Given two functions $f, h : Z \to \mathbb{R}$ their convolution is a function

$$(f \star h)(x) = \sum_{x'=-K/2}^{K/2} f(x - x')h(x') = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}$$

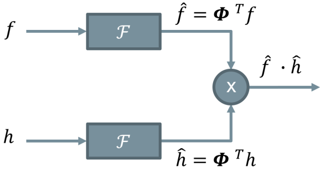
- Given two functions $f, h : \mathcal{V} \to \mathbb{R}$ their convolution is a function

Convolution Theorem

- Given two functions $f, h : Z \to \mathbb{R}$ their convolution is a function

$$(f \star h)(x) = \sum_{x'=-K/2}^{K/2} f(x - x')h(x') = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}$$

- Given two functions $f, h : \mathcal{V} \to \mathbb{R}$ their convolution is a function

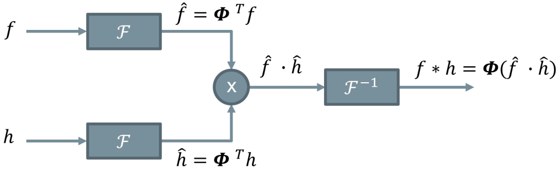


Convolution Theorem

- Given two functions $f, h : Z \to \mathbb{R}$ their convolution is a function

$$(f \star h)(x) = \sum_{x'=-K/2}^{K/2} f(x - x')h(x') = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}$$

- Given two functions $f, h : \mathcal{V} \to \mathbb{R}$ their convolution is a function



$$f * h = \boldsymbol{\Phi}\,(\boldsymbol{\Phi}^T h) \cdot (\boldsymbol{\Phi}^T f)$$

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi \left(\sum_{l'=1}^{p} \mathbf{\Phi} \begin{bmatrix} \hat{h}_1^{(l,l')} & & \\ & \ddots & \\ & & \hat{h}_N^{(l,l')} \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}_{l'} \right) \qquad \begin{array}{l} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{array}$$

where q is the number of features in output and p in input.

Bruna et al. 2014

Spectral graph CNN

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi \left(\sum_{l'=1}^{p} \mathbf{\Phi} \begin{bmatrix} \hat{h}_1^{(l,l')} & & \\ & \ddots & \\ & & \hat{h}_N^{(l,l')} \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}_{l'} \right) \qquad \begin{array}{l} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{array}$$

where q is the number of features in output and p in input.

- ☹ $\mathcal{O}(N)$ parameters per layer
- ☹ $\mathcal{O}(N^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)
- ☹ No guarantee of spatial localization of filters

Bruna et al. 2014

Spectral graph CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters

Represent spectral transfer function as a Chebyshev polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\tilde{\lambda}) = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\lambda})$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters,

$$T_j(\tilde{\lambda}) = 2\tilde{\lambda}T_{j-1}(\tilde{\lambda}) - T_{j-2}(\tilde{\lambda}) \qquad T_0(\tilde{\lambda}) = 1, \quad T_1(\tilde{\lambda}) = \tilde{\lambda}$$

and $-1 \leq \tilde{\lambda} \leq 1$ is normalized frequency.

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters

Represent spectral transfer function as a Chebyshev polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\tilde{\lambda}) = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\lambda})$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters,

$$T_j(\tilde{\lambda}) = 2\tilde{\lambda} T_{j-1}(\tilde{\lambda}) - T_{j-2}(\tilde{\lambda}) \qquad T_0(\tilde{\lambda}) = 1, \quad T_1(\tilde{\lambda}) = \tilde{\lambda}$$

and $-1 \leq \tilde{\lambda} \leq 1$ is normalized frequency.

Application of the filter to scaled Laplacian $\tilde{\boldsymbol{\Delta}} = \mathbf{D}^{-0.5} \boldsymbol{\Delta} \mathbf{D}^{-0.5} - \mathbf{I} = \boldsymbol{\Phi} \tilde{\boldsymbol{\Lambda}} \boldsymbol{\Phi}^T$:

$$\mathbf{g} = \boldsymbol{\Phi}(\sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Lambda}}))\boldsymbol{\Phi}^T \mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}$$

Defferrard, Bresson, Vandergheynst 2016

Spectral graph CNN with polynomial filters

Represent spectral transfer function as a Chebyshev polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\tilde{\lambda}) = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\lambda})$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters,

$$T_j(\tilde{\lambda}) = 2\tilde{\lambda} T_{j-1}(\tilde{\lambda}) - T_{j-2}(\tilde{\lambda}) \qquad T_0(\tilde{\lambda}) = 1, \quad T_1(\tilde{\lambda}) = \tilde{\lambda}$$

and $-1 \leq \tilde{\lambda} \leq 1$ is normalized frequency.

Application of the filter to scaled Laplacian $\tilde{\boldsymbol{\Delta}} = \mathbf{D}^{-0.5} \boldsymbol{\Delta} \mathbf{D}^{-0.5} - \mathbf{I} = \boldsymbol{\Phi} \tilde{\boldsymbol{\Lambda}} \boldsymbol{\Phi}^T$:

$$\mathbf{g} = \boldsymbol{\Phi}\Big(\sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Lambda}})\Big)\boldsymbol{\Phi}^T \mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}$$

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(r|\mathcal{E}|)$ computational complexity

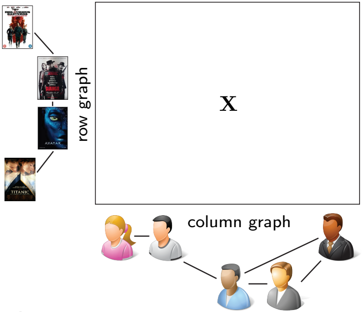
Defferrard, Bresson, Vandergheynst 2016

Multi-Graph CNN

Multi-graph spectral convolutional layer

$$\mathbf{Y}_l = \xi\left(\sum_{l'=1}^{p}\sum_{j,j'=0}^{r}\theta_{jj'll'}T_j(\tilde{\boldsymbol{\Delta}}_{\mathrm{r}})\mathbf{X}_{l'}T_{j'}(\tilde{\boldsymbol{\Delta}}_{\mathrm{c}})\right) \quad \begin{matrix} l = 1,\ldots,q \\ l' = 1,\ldots,p \end{matrix}$$

applied to p input channels ($m \times n$ matrices $\mathbf{X}_1,\ldots,\mathbf{X}_p$) and producing q output channels ($m \times n$ matrices $\mathbf{Y}_1,\ldots,\mathbf{Y}_q$)



Monti, Bresson, Bronstein 2017

Multi-Graph CNN

Multi-graph spectral convolutional layer

$$\mathbf{Y}_l = \xi\left(\sum_{l'=1}^{p}\sum_{j,j'=0}^{r}\theta_{jj'll'}T_j(\tilde{\boldsymbol{\Delta}}_{\mathrm{r}})\mathbf{X}_{l'}T_{j'}(\tilde{\boldsymbol{\Delta}}_{\mathrm{c}})\right) \quad \begin{array}{l} l = 1,\ldots,q \\ l' = 1,\ldots,p \end{array}$$

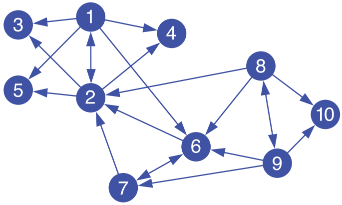
applied to p input channels ($m \times n$ matrices $\mathbf{X}_1,\ldots,\mathbf{X}_p$) and producing q output channels ($m \times n$ matrices $\mathbf{Y}_1,\ldots,\mathbf{Y}_q$)

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support on both graphs
- ☺ $\mathcal{O}(nm)$ computational complexity

Monti, Bresson, Bronstein 2017

Dealing with Directed Graphs

Directed graphs

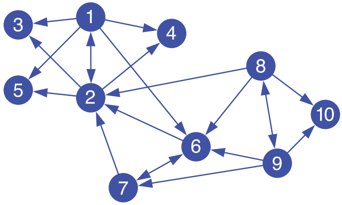


Directed graph

Asymmetric adjacency matrix

$$\begin{bmatrix}
. & 1 & 1 & 1 & 1 & 1 & . & . & . & . \\
1 & . & 1 & 1 & 1 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & . & . & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & 1 & . & . & . & 1 & . & . & 1 & 1 \\
. & . & . & . & . & 1 & 1 & 1 & . & 1 \\
. & . & . & . & . & . & . & . & . & .
\end{bmatrix}$$

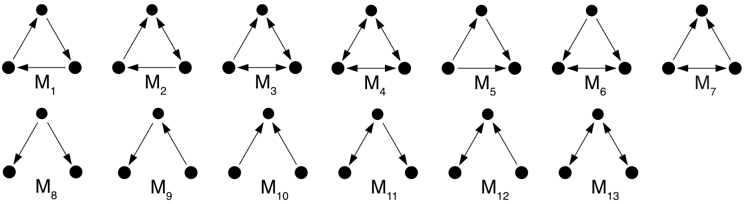
Motif-based graph analysis



Directed graph

Asymmetric adjacency matrix

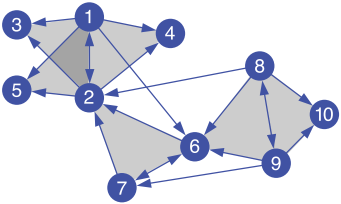
$$\begin{bmatrix} . & 1 & 1 & 1 & 1 & 1 & . & . & . & . \\ 1 & . & 1 & 1 & 1 & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & 1 & . & . & . & . \\ . & 1 & . & . & . & 1 & . & . & 1 & 1 \\ . & . & . & . & . & 1 & 1 & 1 & . & 1 \\ . & . & . & . & . & . & . & . & . & . \end{bmatrix}$$



Graph motifs

Benson et al. 2016

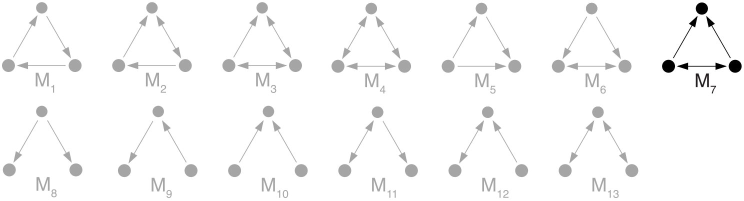
Motif-based graph analysis



Directed graph

Motif adjacency matrix for M_7

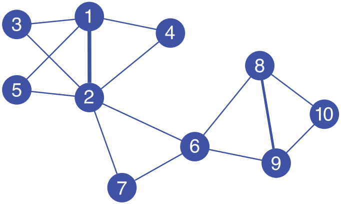
$$\begin{bmatrix}
. & 3 & 1 & 1 & 1 & . & . & . & . & . \\
3 & . & 1 & 1 & 1 & 1 & 1 & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & 1 & 1 & 1 & . \\
. & 1 & . & . & . & 1 & . & . & . & . \\
. & . & . & . & . & 1 & . & . & 2 & 1 \\
. & . & . & . & . & 1 & . & 2 & . & 1 \\
. & . & . & . & . & . & . & 1 & 1 & .
\end{bmatrix}$$



Graph motifs

Benson et al. 2016

Motif Laplacians



Undirected weighted graph

Motif adjacency matrix for M_7

$$\begin{bmatrix} . & 3 & 1 & 1 & 1 & . & . & . & . & . \\ 3 & . & 1 & 1 & 1 & 1 & 1 & . & . & . \\ 1 & 1 & . & . & . & . & . & . & . & . \\ 1 & 1 & . & . & . & . & . & . & . & . \\ 1 & 1 & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & 1 & 1 & 1 & . \\ . & 1 & . & . & . & 1 & . & . & . & . \\ . & . & . & . & . & 1 & . & . & 2 & 1 \\ . & . & . & . & . & 1 & . & 2 & . & 1 \\ . & . & . & . & . & . & . & 1 & 1 & . \end{bmatrix}$$

Motif Laplacian for motif $k = 1, \ldots, K$

$$\tilde{\mathbf{\Delta}}_k = \mathbf{I} - \tilde{\mathbf{D}}_k^{-1/2} \tilde{\mathbf{W}}_k \tilde{\mathbf{D}}_k^{-1/2}$$

Benson et al. 2016

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1, \ldots, k_j \in \{1, \ldots, K\}} \theta_{k_1, \ldots, k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K, \ldots, K})$ is the vector of filter parameters

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \theta_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K,\ldots,K})$ is the vector of filter parameters

☺ Explicitly accounts for directed graph structures

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\mathbf{\Delta}}_1, \ldots, \tilde{\mathbf{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \theta_{k_1,\ldots,k_j} \tilde{\mathbf{\Delta}}_{k_1} \cdots \tilde{\mathbf{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K,\ldots,K})$ is the vector of filter parameters

☺ Explicitly accounts for directed graph structures

☺ Anisotropic kernels

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\mathbf{\Delta}}_1, \ldots, \tilde{\mathbf{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \theta_{k_1,\ldots,k_j} \tilde{\mathbf{\Delta}}_{k_1} \cdots \tilde{\mathbf{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K,\ldots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\Delta}}_1, \dots, \tilde{\boldsymbol{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1, \dots, k_j \in \{1, \dots, K\}} \theta_{k_1, \dots, k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_{K, \dots, K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support
- ☺ $\mathcal{O}(1)$ parameters per layer

Monti, Otness, Bronstein 2018

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \theta_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K,\ldots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support
- ☹ $\frac{1-K^{r+1}}{1-K}$ parameters per layer, intractable in practice

MotifNet: multivariate polynomial

Apply K-variate polynomial of order r to the motif Laplacians

$$\tau_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \theta_0 \mathbf{I} + \sum_{j=1}^{r} \sum_{k_1,\ldots,k_j \in \{1,\ldots,K\}} \theta_{k_1,\ldots,k_j} \tilde{\boldsymbol{\Delta}}_{k_1} \cdots \tilde{\boldsymbol{\Delta}}_{k_j}$$

where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{K,\ldots,K})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support
- ☹ $\frac{1-K^{r+1}}{1-K}$ parameters per layer, intractable in practice
- ☹ $\mathcal{O}(n\frac{1-K^{r+1}}{1-K})$ computational complexity

Monti, Otness, Bronstein 2018

MotifNet: simplified multivariate polynomial

Two possible simplications:

- We consider only $K = 2$ simple motifs corresponding to incoming and outgoing edges (MotifNet-d):

$$\tau_{\boldsymbol{\theta}} = \theta_0 \mathbf{I} + \theta_1 \tilde{\boldsymbol{\Delta}}_1 + \theta_2 \tilde{\boldsymbol{\Delta}}_2 + \theta_{11} \tilde{\boldsymbol{\Delta}}_1^2 + \ldots + \theta_{22} \tilde{\boldsymbol{\Delta}}_2^2 + \ldots \tag{1}$$

MotifNet: simplified multivariate polynomial

Two possible simplications:

- We consider only $K = 2$ simple motifs corresponding to incoming and outgoing edges (MotifNet-d):

$$\tau_{\boldsymbol{\theta}} = \theta_0 \mathbf{I} + \theta_1 \tilde{\boldsymbol{\Delta}}_1 + \theta_2 \tilde{\boldsymbol{\Delta}}_2 + \theta_{11} \tilde{\boldsymbol{\Delta}}_1^2 + \ldots + \theta_{22} \tilde{\boldsymbol{\Delta}}_2^2 + \ldots \tag{1}$$

- We consider a simplified version of multivariate polynomials (MotifNet-m) defined recursively as:

MotifNet: simplified multivariate polynomial

Two possible simplications:

- We consider only $K = 2$ simple motifs corresponding to incoming and outgoing edges (MotifNet-d):

$$\tau_{\boldsymbol{\theta}} = \theta_0 \mathbf{I} + \theta_1 \tilde{\boldsymbol{\Delta}}_1 + \theta_2 \tilde{\boldsymbol{\Delta}}_2 + \theta_{11} \tilde{\boldsymbol{\Delta}}_1^2 + \ldots + \theta_{22} \tilde{\boldsymbol{\Delta}}_2^2 + \ldots \tag{1}$$

- We consider a simplified version of multivariate polynomials (MotifNet-m) defined recursively as:

$$\begin{aligned}
\mathbf{P}_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) &= \sum_{j=0}^{r} \theta_j \mathbf{P}_j; \qquad (2) \\
\mathbf{P}_j(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) &= \sum_{k=1}^{K} \alpha_{k,j} \tilde{\boldsymbol{\Delta}}_k \mathbf{P}_{j-1}, \quad j = 1, \ldots, p \\
\mathbf{P}_0 &= \mathbf{I}, \\
\sum_{k=1}^{K} \alpha_{k,j} &= 1 \qquad j = 1, \ldots, p
\end{aligned}$$

Monti, Otness, Bronstein 2018

MotifNet-m: simplified multivariate polynomial

Apply attention-based recursive polynomial of order r

$$\mathbf{P}_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \sum_{j=0}^{r} \theta_j \mathbf{P}_j;$$

$$\mathbf{P}_j(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \sum_{k=1}^{K} \alpha_{k,j} \tilde{\boldsymbol{\Delta}}_k \mathbf{P}_{j-1}, \qquad \mathbf{P}_0 = \mathbf{I},$$

where $\boldsymbol{\Theta} = (\theta_0, \ldots, \theta_r, \alpha_{1,1}, \ldots, \alpha_{K,r})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support

Monti, Otness, Bronstein 2018

MotifNet-m: simplified multivariate polynomial

Apply attention-based recursive polynomial of order r

$$\mathbf{P}_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \sum_{j=0}^{r} \theta_j \mathbf{P}_j;$$

$$\mathbf{P}_j(\tilde{\boldsymbol{\Delta}}_1, \ldots, \tilde{\boldsymbol{\Delta}}_K) = \sum_{k=1}^{K} \alpha_{k,j} \tilde{\boldsymbol{\Delta}}_k \mathbf{P}_{j-1}, \qquad \mathbf{P}_0 = \mathbf{I},$$

where $\boldsymbol{\Theta} = (\theta_0, \ldots, \theta_r, \alpha_{1,1}, \ldots, \alpha_{K,r})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support
- ☺ $Kr + 1$ parameters per layer

Monti, Otness, Bronstein 2018

MotifNet-m: simplified multivariate polynomial

Apply attention-based recursive polynomial of order r

$$\mathbf{P}_{\mathbf{\Theta}}(\tilde{\mathbf{\Delta}}_1, \ldots, \tilde{\mathbf{\Delta}}_K) = \sum_{j=0}^{r} \theta_j \mathbf{P}_j;$$

$$\mathbf{P}_j(\tilde{\mathbf{\Delta}}_1, \ldots, \tilde{\mathbf{\Delta}}_K) = \sum_{k=1}^{K} \alpha_{k,j} \tilde{\mathbf{\Delta}}_k \mathbf{P}_{j-1}, \qquad \mathbf{P}_0 = \mathbf{I},$$

where $\mathbf{\Theta} = (\theta_0, \ldots, \theta_r, \alpha_{1,1}, \ldots, \alpha_{K,r})$ is the vector of filter parameters

- ☺ Explicitly accounts for directed graph structures
- ☺ Anisotropic kernels
- ☺ Filters have guaranteed r-hops support
- ☺ $Kr + 1$ parameters per layer
- ☺ $\mathcal{O}(nr)$ computational complexity

Monti, Otness, Bronstein 2018

Directed CORA

- Citations network representing papers (vertices) and citations (directed edges)
- Goal: vertex-wise classification (paper topic)
- 19,793 documents, 70 classes
- Training/validation/test set: 10% samples (1979 vertices)

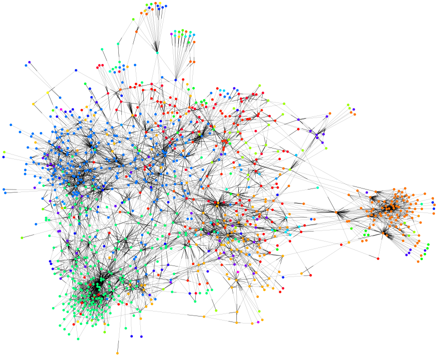
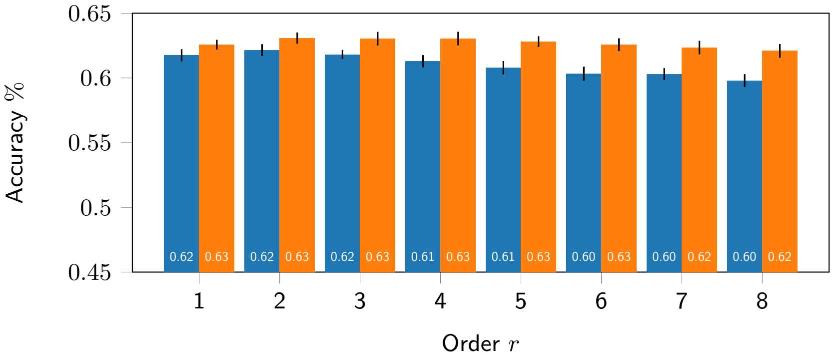


Figure: Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann 2017

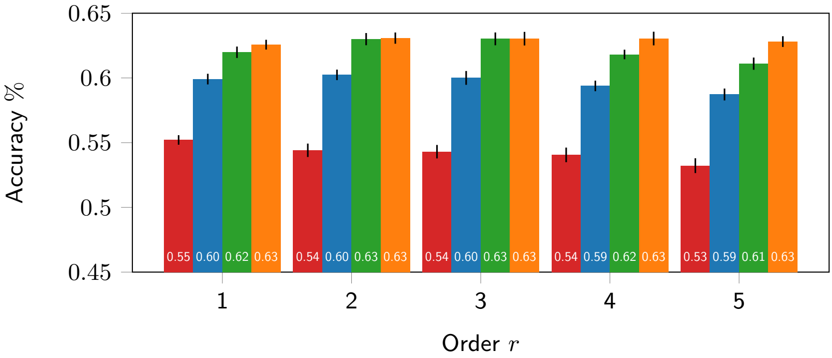
Example: directed citation networks (CORA)



Classification accuracy obtained with ChebNet on undirected graph (blue) and MotifNet-m (orange).

Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann, 2017

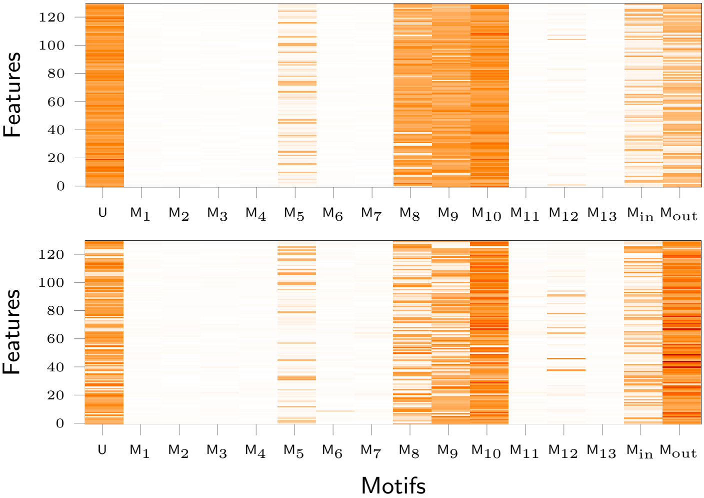
Example: directed citation networks (CORA)



Classification accuracy obtained with ChebNet applied with directed adjacency matrix $\mathbf{W}$ (blue) / $\mathbf{W}^\top$ (red), MotifNet-d (green) and MotifNet-m (orange).

Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann, 2017

Example: directed citation networks (CORA)



Attention scores α obtained with MotifNet-m ($r = 1$) from 1st and 2nd graph convolutional layers. Dark/bright colors represent high/low probabilities.

Monti, Otness, Bronstein 2018; data: Bojchevski, Günnemann, 2017

Conclusions

- We presented MotifNet, a motif-based GCN for directed graphs.
- Our solution realizes anisotropic filters on the provided domain generalizing and enriching classic undirected GCNs.
- Thanks to the proposed simplifications, MotifNet-m requires a number of parameters and a computational complexity comparable with the ones of ChebNet.
- Experimental evaluations on the directed version of CORA show the superior performance of our approach.

Thank You

Bibliography

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, “Geometric deep learning: going beyond Euclidean data”, arXiv:1611.08097, 2016. **First review paper of geometric deep learning**

M. Gori, G. Monfardini, F. Scarselli, “A new model for learning in graph domains”, *Proc. IJCNN* 2005. **First neural net on graphs (GNN framework)**

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, “The graph neural network model”, *IEEE Trans. Neural Networks* 20(1):61–80, 2009.

J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, “Spectral networks and locally connected networks on graphs”, *Proc. ICML* 2014. **First Spectral CNN on graphs**

M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data”, arXiv:1506.05163, 2015. **Spectral CNN with smooth multipliers**

J. Masci, D. Boscaini, M. M. Bronstein, P. Vandergheynst, “Geodesic convolutional neural networks on Riemannian manifolds”, arXiv:1501.06297v2, 2015. **First spatial CNN on manifolds (Geodesic CNN framework)**

Bibliography

M. Defferrard, X. Bresson, P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering”, *Proc. NIPS* 2016. **Spectral CNN with Chebychev polynomial filters (ChebNet)**

T. N. Kipf, M. Welling, “Semi-supervised classification with graph convolutional networks”, arXiv:1609.02907, 2016. **Graph convolutional network (GCN) framework, a simplification of ChebNet**

R. Levie, F. Monti, X. Bresson, M. M. Bronstein, “CayleyNets: Graph convolutional neural networks with complex rational spectral filters”, arXiv:1705.07664, 2017. **Spectral CNN with complex rational filters (CayleyNet)**

J. Atwood and D. Towsley, “Diffusion-convolutional neural networks”, arXiv:1511.02136v2, 2016. **Diffusion CNN framework**

F. Monti, K. Otness, M. M. Bronstein, “MotifNet: a motif-based Graph Convolutional Network for directed graphs”, arXiv:1802.01572, 2018. **MotifNet**

Bibliography

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs", *Proc. CVPR* 2017. **Mixture Model network (MoNet) framework**

D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks", *Computer Graphics Forum* 34(5):13-23, 2015. **Localized Spectral CNN framework**

D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks", *Proc. NIPS* 2016. **Anisotropic CNN framework**

F. Monti, M. M. Bronstein, X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks", arXiv:1704.06803, 22 April 2017. **CNNs on multiple graphs and application to matrix completion**

Bibliography

A. Sinha, J. Bai, K. Ramani, "Deep learning 3D shape surfaces using geometry images", Proc. ECCV 2016. **CNN on spherical authalic parametrization**

H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers", Trans. Graphics 36(4), 2017. **CNN on planar flat-torus parametrization**

L. Yi, H. Su, X. Guo, L. Guibas, "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation", Proc. CVPR 2017. **Spectral transformer networks**

C. R. Qi, H. Su, K. Mo, L. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", Proc. CVPR 2017. **PointNet**

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, "Dynamic Graph CNN for learning on point clouds", arXiv:1712.00268, 2018. **Edge convolution**

Bibliography

O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, M. M. Bronstein, "Deep functional maps: Structured prediction for dense shape correspondence", Proc. ICCV 2017. **Intrinsic structured prediction (FMNet)**

O. Litany, A. M. Bronstein, M. M. Bronstein, A. Makadia, "Deformable shape completion with graph convolutional autoencoders", Proc. CVPR 2018. **Shape completion with intrinsic variational autoencoder**

Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition", *Neural Computation* 1(4):541–551, 1989. **Classical Euclidean CNN**