

Graph Neural Networks in Big Data Analytics: Introduction V

Alexander Schönhuth



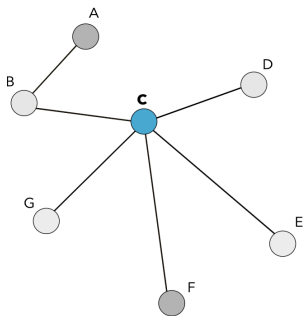
Bielefeld University
November 17, 2022

CONTENTS TODAY

- ▶ Reminder: Polynomial Filters
- ▶ Modern GNN's
- ▶ Global Convolution

Reminder: Polynomial Filters on Graphs

THE GRAPH LAPLACIAN: EXAMPLE



Input Graph G

$$\begin{array}{c} \text{A} \\ \text{B} \\ \mathbf{C} \\ \text{D} \\ \text{E} \\ \text{F} \\ \text{G} \end{array} \begin{bmatrix} & \text{A} & \text{B} & \mathbf{C} & \text{D} & \text{E} & \text{F} & \text{G} \\ \left[\begin{array}{cccccccc} 1 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 5 & -1 & -1 & -1 & -1 & \\ & & -1 & 1 & & & & \\ & & -1 & & 1 & & & \\ & & -1 & & & 1 & & \\ & & -1 & & & & 1 & \end{array} \right. \end{bmatrix}$$

Laplacian L of G

Zeros are not displayed. The Laplacian depends only on the graph structure.

From <https://distill.pub/2021/understanding-gnns/>

POLYNOMIALS OF THE LAPLACIAN

One can build *polynomials of the Laplacian* of the form

$$p_w(L) = w_0 I_n + w_1 L + w_2 L^2 + \dots + w_d L^d = \sum_{i=0}^d w_i L^i \quad (1)$$

where I_n is the n -dimensional identity matrix.

Alternatively, each such polynomial can be represented by its *vector of coefficients*

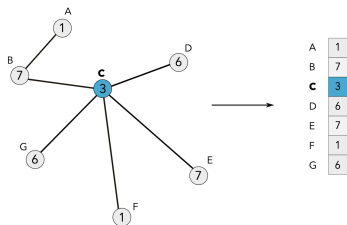
$$w = [w_0, \dots, w_d] \quad (2)$$

Remark:

- ▶ $p_w(L)$ is an $n \times n$ -Matrix for each w , just like L
- ▶ The $p_w(L)$ represent the equivalent of filters in CNN's
- ▶ We will see why that is...

POLYNOMIALS OF THE LAPLACIAN II

- ▶ In the following, each node $v \in V$ stores information $x_v \in \mathbb{R}$
 - ▶ For ease of presentation only
 - ▶ Everything applies also for multi-dimensional vectors
- ▶ Stack real-valued features into vector $x \in \mathbb{R}^n$



Collecting node information into vector.

From <https://distill.pub/2021/understanding-gnns/>

POLYNOMIAL FILTERS: DEFINITION

- ▶ In the following, each node $v \in V$ stores information $x_v \in \mathbb{R}$
 - ▶ For ease of presentation only
 - ▶ Everything applies also for multi-dimensional vectors
- ▶ Stack real-valued features into vector $x \in \mathbb{R}^n$
- ▶ *Convolution with a polynomial filter p_w* is then defined as

$$x' = p_w(L)x \quad (3)$$

that is, by applying the matrix $p_w(L) \in \mathbb{R}^{n \times n}$ to the vector $x \in \mathbb{R}^n$

POLYNOMIAL FILTERS: EXAMPLES

Examples:

- ▶ $w = [w_0, 0, \dots, 0]$:

$$x' = p_w(L) = w_0 I_n x + 0 + \dots + 0 = w_0 x$$

- ▶ $w = [0, 1, 0, \dots, 0]$:

$$x' = p_w(L) = Lx$$

Let $\mathcal{N}(v)$ is the *neighborhood* of v , that is all nodes attached to v via an edge, so

$$x'_v = (Lx)_v = \sum_{u \in G} L_{vu} x_u = \sum_{u \in G} (D_{vu} - A_{vu}) x_u = D_{vv} x_v - \sum_{u \in \mathcal{N}(v)} x_u$$

- ▶ *Interpretation:* Features of v are combined with features of immediate neighbors \Leftrightarrow message passing

POLYNOMIAL FILTERS: POLYNOMIAL DEGREE

- ▶ Let $\text{dist}(u, v)$ be the length of the shortest path between nodes $u, v \in V$
 - ▶ For example, $(u, v) \in E$ corresponds to $\text{dist}(u, v) = 1$
- ▶ Basic calculations imply

$$\text{dist}(u, v) > i \quad \text{implies} \quad (L^i)_{uv} = \underbrace{(L \times \dots \times L)}_{i \text{ times}}_{uv} = 0 \quad (4)$$

- ▶ Let $p_w(L)$ have polynomial degree d . One obtains

$$x'_v = (p_w(L)x)_v = \sum_{i=0}^d w_i \sum_{u \in V} (L^i)_{vu} x_u = \sum_{i=0}^d w_i \sum_{\substack{u \in V \\ \text{dist}_G(v, u) \leq i}} (L^i)_{vu} x_u \quad (5)$$

- ▶ (5): convolution at node v only with nodes at most d hops away



Summary: Degree of localization governed by degree of polynomial filter

POLYNOMIAL FILTERS: STACKING LAYERS

Start with the original features.

$$h^{(0)} = x$$

Color Codes:

-  Computed node embeddings.
-  Learnable parameters.

Then iterate, for $k = 1, 2, \dots$ upto K :

$$p^{(k)} = p_{w^{(k)}}(L)$$

Compute the matrix $p^{(k)}$ as the polynomial defined by the filter weights $w^{(k)}$ evaluated at L .

$$g^{(k)} = p^{(k)} \times h^{(k-1)}$$

Multiply $p^{(k)}$ with $h^{(k-1)}$: a standard matrix-vector multiply operation.

$$h^{(k)} = \sigma(g^{(k)})$$

Apply a non-linearity σ to $g^{(k)}$ to get $h^{(k)}$.

Note: weights re-used at every node, as in CNN's.

From <https://distill.pub/2021/understanding-gnns/>

Modern GNN's

MODERN GNN'S

- ▶ Re-consider $p_w(L) = L$, yielding

$$(Lx)_v = D_v x_v - \sum_{u \in \mathcal{N}(v)} x_u \quad (6)$$

- ▶ (6) decomposes into
 - ▶ Aggregating over immediate neighbor features $x_u, u \in \mathcal{N}(v)$
 - ▶ Combining with node v 's own feature x_v
- ▶ *Idea*: Generalize by considering different kinds of aggregation and combination steps
- ▶ *Caveat*: Aggregation needs to be node-order invariant
- ▶ Iteratively repeating 1-hop localized convolutions K times: receptive field including all nodes up to K hops away

GRAPH CONVOLUTIONAL NETWORKS (GCN'S)

For $k = 1, \dots, K$




$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)} \right) \quad \text{for all } v \in V.$$

Node v 's
embedding at
step k .

Mean of v 's
neighbour's
embeddings at
step $k - 1$.

Node v 's
embedding at
step $k - 1$.

Color Codes:

-  Embedding of node v .
-  Embedding of a neighbour of node v .
-  (Potentially) Learnable parameters.

From <https://distill.pub/2021/understanding-gnns/>

GRAPH CONVOLUTIONAL NETWORKS (GCN'S) I

$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)} \right) \quad \text{for all } v \in V.$$

From <https://distill.pub/2021/understanding-gnns/>

- ▶ Derive predictions from $h_v^{(K)}$
- ▶ Function $f^{(k)}$, matrices $W^{(k)}$, $B^{(k)}$ shared across nodes
- ▶ Dividing by $|\mathcal{N}(v)|$ implements normalization; alternative normalization schemes conceivable

GRAPH ATTENTION NETWORKS (GAN'S)

$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \left[\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k-1)} h_u^{(k-1)} + \alpha_{vv}^{(k-1)} h_v^{(k-1)} \right] \right) \quad \text{for all } v \in V.$$

Node v 's
embedding at
step k .

Weighted mean of
 v 's neighbour's
embeddings at
step $k - 1$.

Node v 's
embedding at
step $k - 1$.

for $k = 1, \dots, K$, where normalized attention weights $\alpha^{(k)}$ are generated by $A^{(k)}$

$$\alpha_{vu}^{(k)} = \frac{A^{(k)}(h_v^{(k)}, h_u^{(k)})}{\sum_{w \in \mathcal{N}(v)} A^{(k)}(h_v^{(k)}, h_w^{(k)})} \quad \text{for all } (v, u) \in E.$$

Color Codes:

- Embedding of node v .
- Embedding of a neighbour of node v .
- (Potentially) Learnable parameters.

From <https://distill.pub/2021/understanding-gnns/>

GRAPH ATTENTION NETWORKS (GAN'S) II

$$h_v^{(k)} = f^{(k)} \left(W^{(k)} \cdot \left[\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k-1)} h_u^{(k-1)} + \alpha_{vv}^{(k-1)} h_v^{(k-1)} \right] \right) \quad \text{for all } v \in V.$$

From <https://distill.pub/2021/understanding-gnns/>

- ▶ Derive predictions from $h_v^{(K)}$
- ▶ Function $f^{(k)}$, matrices $W^{(k)}$ and attention mechanism $A^{(k)}$ (generally another neural network) shared across nodes
- ▶ Here: single-headed attention; multi-headed attention similar

REFERENCES

- ▶ **ChebNet:** <https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf>
- ▶ **Graph Convolutional Networks (GCN's):**
<https://openreview.net/forum?id=SJU4ayYg1>
- ▶ **Graph Attention Networks (GAN's):**
<https://openreview.net/forum?id=rJXMpikCZ>

Global Convolution

GLOBAL CONVOLUTION: MOTIVATION

As before, for sake of clarity, let feature vectors x be one-dimensional.

Question:

- ▶ Let $x \in \mathbb{R}^{|V|}$ be a feature vector: how smooth is x w.r.t. G ?
- ▶ In other words: how similar are features x_i, x_j within x for edges (i, j) ?

Hint:

- ▶ Normalize x such that $\sum_i x_i^2 = 1$
- ▶ Consider the Laplacian based quantity

$$R_L(x) = \frac{x^T L x}{x^T x} = \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2} = \sum_{(i,j) \in E} (x_i - x_j)^2 \quad (7)$$

- ▶ Similar values for neighboring nodes imply small $R_L(x)$

GLOBAL CONVOLUTION: MOTIVATION II

$$R_L(x) = \frac{x^T L x}{x^T x} = \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

Laplacian Eigenvectors:

- ▶ L is a real symmetric matrix \Leftrightarrow All eigenvalues $\lambda_1 \leq \dots \leq \lambda_n$ are real
- ▶ The corresponding eigenvectors u_1, \dots, u_n can be taken *orthonormal*:

$$u_{k_1}^T u_{k_2} = \begin{cases} 1 & \text{if } k_1 = k_2 \\ 0 & \text{if } k_1 \neq k_2 \end{cases} \quad (8)$$

- ▶ One can compute

$$\arg \min_{x, x \perp \{u_1, \dots, u_{i-1}\}} R_L(x) = u_i \quad \text{and} \quad \min_{x, x \perp \{u_1, \dots, u_{i-1}\}} R_L(x) = \lambda_i \quad (9)$$

\Leftrightarrow Eigenvectors u_1, \dots, u_n are successively less smooth

GLOBAL CONVOLUTION: MOTIVATION II

Global Convolution: Idea

- ▶ Let u_1, \dots, u_n be the (orthonormal) eigenvectors of L
- ▶ *Intuition:* According to (9), eigenvectors reflect weights on nodes determined such that information is most smooth with respect to the structure of the graph
- ▶ *Goal:* Exchange information between similar neighbors more than between different neighbors

GLOBAL CONVOLUTION: MOTIVATION III

Global Convolution: Idea

- ▶ Knowing about (9), base convolution on suitable representations of x over u_1, \dots, u_n
 - ▶ In particular, according to (9), make preferable use of eigenvectors referring to small eigenvalues
 - ▶ *Global convolution*: convolution acting on eigenvectors u_i virtually acts on *all* nodes simultaneously
 - ▶ *Reminder*: local convolution only refers to neighborhoods of nodes
- ☞ Consider *spectral convolutions* as a suitable form of global convolution

SPECTRAL CONVOLUTIONS: FOUNDATION

- ▶ Let $\Lambda := \text{diag}(\lambda_1, \dots, \lambda_n)$ be the diagonal matrix having $\lambda_1 \leq \dots \leq \lambda_n$ on the diagonal
- ▶ Let U be the matrix having columns u_1, \dots, u_n (in that order)
- ▶ One obtains

$$L = U\Lambda U^T$$

- ▶ The n eigenvectors u_1, \dots, u_n form a basis, so any feature vector x can be represented as a linear combination of the u_i

$$x = \sum_{i=1}^n \hat{x}_i u_i = U\hat{x}$$

where \hat{x} is the vector of coefficients

- ▶ The orthonormality condition yields

$$x = U\hat{x} \quad \Leftrightarrow \quad U^T x = \hat{x}$$

SPECTRAL CONVOLUTIONS: PROTOCOL

- ▶ Compute *spectral representation*

$$\hat{x} = U^T x =: [\hat{x}_1, \dots, \hat{x}_n]$$

- ▶ Truncate \hat{x} to first m components

$$\hat{x}[m] := [\hat{x}_1, \dots, \hat{x}_m, 0, \dots, 0] \quad (10)$$

where $\hat{x}[m]$ can also be computed by

$$\hat{x}[m] = U_m^T x \quad \text{where} \quad (U_m)_{ij} = \begin{cases} U_{ij} & 1 \leq j \leq m \\ 0 & m > j \leq n \end{cases} \quad (11)$$

U_m is defined by turning the rightmost $n - m$ columns in U to zero

- ▶ One can view $\lambda_1 \leq \dots \leq \lambda_n$ as frequencies:
 - ▶ Lower frequencies capture basic, globally applicable relationships
 - ▶ Higher frequencies capture local details
 - ▶ Omitting higher frequencies omits details, but keeps global structure

SPECTRAL CONVOLUTIONS: PROTOCOL II

$$\hat{x}[m] = U_m^T x \quad \text{where} \quad (U_m)_{ij} = \begin{cases} U_{ij} & 1 \leq j \leq m \\ 0 & m > j \leq n \end{cases}$$

where U_m has rightmost $n - m$ columns in U turned to zero

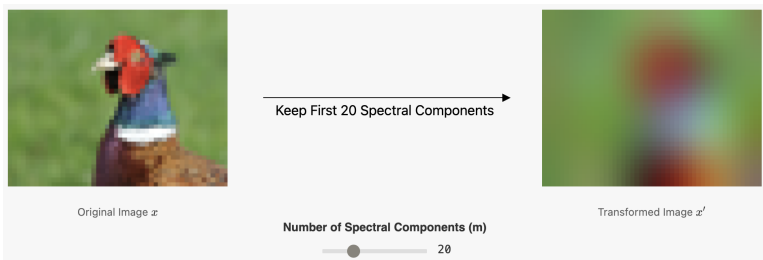
- ▶ This virtually turns the original x into

$$x[m] := U \cdot \hat{x}[m] = U \cdot U_m^T x = U_m \cdot U_m^T x$$

- ▶ $x[m]$ can be considered an approximation of x that optimally caters to global convolution
- ▶ Because relying small eigenvalues (i.e. "small frequencies"):

$x[m]$ **still captures all essential structure of x**

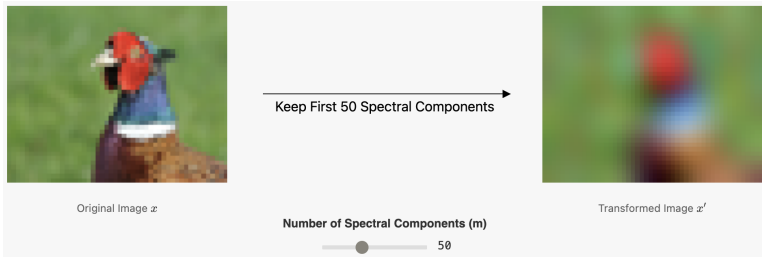
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 20 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

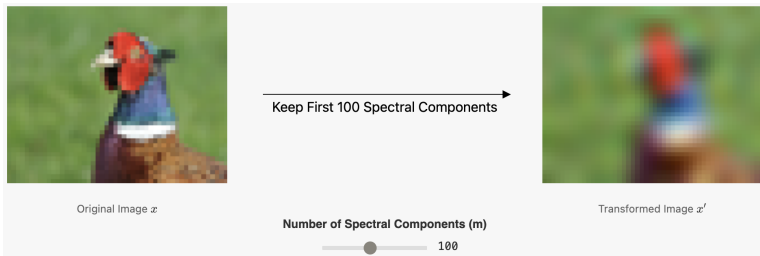
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 50 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

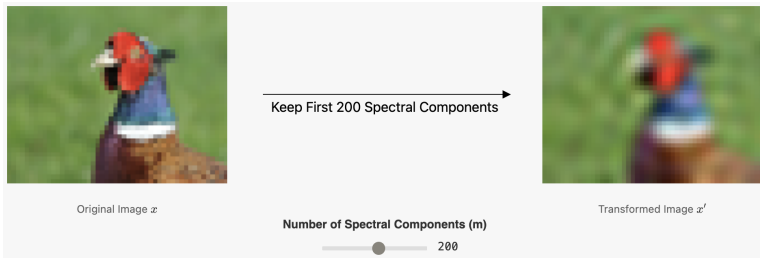
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 100 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

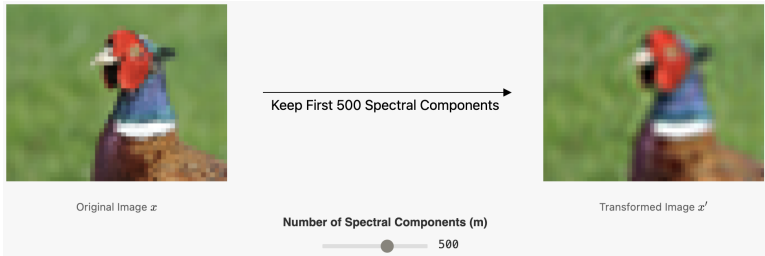
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 200 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

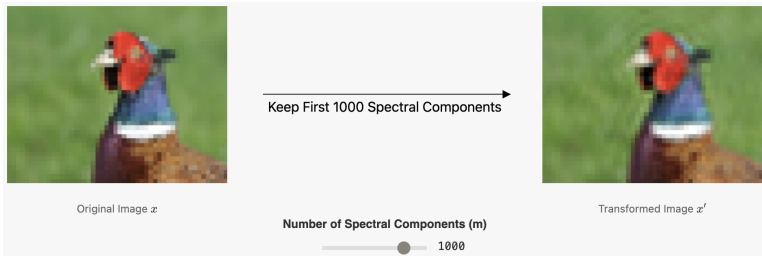
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 500 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

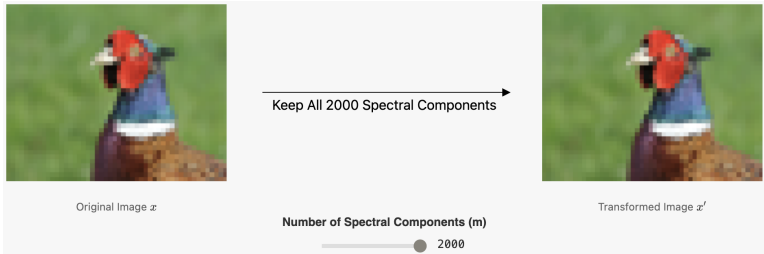
SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using first 1000 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

SPECTRAL REPRESENTATIONS: EXAMPLES



Approximation using all 2000 eigenvectors

From <https://distill.pub/2021/understanding-gnns/>

SPECTRAL CONVOLUTION GNN: PROTOCOL

- ▶ Consider a GNN having K layers
- ▶ Computing layer $k + 1$ from layer $k, k = 0, \dots, K - 1$, the GNN implements spectral (global) convolution
- ▶ Let

$$[h_1^k, \dots, h_n^k]^T =: h^k$$

be the vector storing node information in layer k where

$$h^0 = x$$

is the original node information vector

SPECTRAL CONVOLUTION GNN: PROTOCOL

Start with the original features.

$$h^{(0)} = x$$

Then iterate, for $k = 1, 2, \dots$ upto K :

$$\hat{h}^{(k-1)} = U_m^T h^{(k-1)}$$

Convert previous feature $h^{(k-1)}$ to its spectral representation $\hat{h}^{(k-1)}$.

$$\hat{g}^{(k)} = \hat{w}^{(k)} \odot \hat{h}^{(k-1)}$$

Convolve with filter weights $\hat{w}^{(k)}$ in the spectral domain to get $\hat{g}^{(k)}$.
 \odot represents element-wise multiplication.

$$g^{(k)} = U_m \hat{g}^{(k)}$$

Convert $\hat{g}^{(k)}$ back to its natural representation $g^{(k)}$.

$$h^{(k)} = \sigma \left(g^{(k)} \right)$$

Apply a non-linearity σ to $g^{(k)}$ to get $h^{(k)}$.

Pass in spectral GNN from layer k to layer $k + 1$

Only m parameters required: $\hat{w}^{(k)}$ consists of only m weights

From <https://distill.pub/2021/understanding-gnns/>

Color Codes:

■ Computed node embeddings.

■ Learnable parameters.

Thanks for your attention!