Mathematics in Graph Convolutional Network

Zhenmiao Zhang

March 24, 2021

Contents

Overview

First Part: Theoriems in GCN

- From Fourier Series to Fourier Transform
- Spectral Graph Theory: Graph Laplacian
- Graph Fourier Transform and Convolution

Second Part: Three classical GCN models

- Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering
  (NIPS 2016)
- Semi-Supervised Classification with Graph Convolutional Networks
  (ICLR 2017)
- Modeling Relational Data with Graph Convolutional Networks
  (ESWC 2018)

Contents

Inner Product

For real vector space ($\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$):

$$< \boldsymbol{x}, \boldsymbol{y} >= \sum_{i=1}^{n} x_i y_i.$$

For complex vector space ($\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$):

$$< \boldsymbol{x}, \boldsymbol{y} >= \sum_{i=1}^{n} x_i \overline{y_i}. \ (\textit{this is to ensure } < \boldsymbol{x}, \boldsymbol{x} > \geq 0)$$

For complex and square integrable function space:
($f, g \in L^2[a, b]$, i.e., $f : [a, b] \to \mathbb{C}, \int_a^b |f(t)|^2 \ dt < \infty$)

$$< f, g >= \int_a^b f(t) \overline{g(t)} \ dt.$$

Orthogonal Functions Basis

$\mathcal{E} = \{e_1, e_2, ..., e_n\}$ is orthogonal functions basis if:

$$< e_i, e_j > \begin{cases} = 0 & \text{for } i \neq j, \\ > 0 & \text{for } i = j. \end{cases}$$

Any function h lies in the space of $\mathcal{E}$ can be written as:

$$h = \sum_{i=1}^{n} \frac{< h, e_i > e_i}{< e_i, e_i >}.$$

Fourier Series (Definition)

Fourier Series

Let f(x) be a function of period T. If f(x) is integrable and absolutely integrable on $[-\frac{T}{2}, \frac{T}{2}]$, it can be written as:

$$f(x) \sim A_0 + \sum_{k=1}^{\infty} A_k \sin(kwx + \varphi_k) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kwx + b_k \sin kwx),$$

where

$$w = \frac{2\pi}{T} \text{ is the angular frequency, } A_k = \sqrt{a_k^2 + b_k^2} \text{ is amplitude,}$$

$$A_k \sin(kwx + \varphi_k) \text{ has period } T/k, \text{ frequency } k/T,$$

$$a_k = \frac{2}{T} \int_{-T/2}^{T/2} f(x) \cos kwx \, dx, \; b_k = \frac{2}{T} \int_{-T/2}^{T/2} f(x) \sin kwx \, dx.$$

Fourier Series (Deduction)

Assume $f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kwx + b_k \sin kwx)$ (uniform convergence):

$$\int_{-T/2}^{T/2} f(x) \cos nwx \; dx = \frac{a_0}{2} \int_{-T/2}^{T/2} \cos nwx \; dx$$

$$+ \sum_{k=1}^{\infty} \Big(a_k \int_{-T/2}^{T/2} \cos kwx \cos nwx \; dx + b_k \int_{-T/2}^{T/2} \sin kwx \cos nwx \; dx \Big)$$

$$= \sum_{k=1}^{\infty} \Big(a_k \int_{-T/2}^{T/2} \frac{\cos(k+n)wx + \cos(k-n)wx}{2} \; dx$$

$$+ b_k \int_{-T/2}^{T/2} \frac{\sin(k+n)wx + \sin(k-n)wx}{2} \; dx \Big)$$

$$= a_n \int_{-T/2}^{T/2} \frac{\cos 2nwx + 1}{2} \; dx + b_n \int_{-T/2}^{T/2} \frac{\sin 2nwx}{2} \; dx = \frac{T}{2} a_n$$

$$\Longrightarrow a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(x) \cos nwx \; dx.$$

Fourier Series (Deduction)

$$\int_{-T/2}^{T/2} f(x)\sin nwx\;dx = \frac{a_0}{2}\int_{-T/2}^{T/2} \sin nwx\;dx$$

$$+\sum_{k=1}^{\infty}\left(a_k\int_{-T/2}^{T/2}\cos kwx \sin nwx\;dx + b_k\int_{-T/2}^{T/2}\sin kwx \sin nwx\;dx\right)$$

$$=\sum_{k=1}^{\infty}\left(a_k\int_{-T/2}^{T/2}\frac{\sin(k+n)wx-\sin(k-n)wx}{2}\;dx\right.$$

$$\left.-\,b_k\int_{-T/2}^{T/2}\frac{\cos(k+n)wx-\cos(k-n)wx}{2}\;dx\right)$$

$$=a_n\int_{-T/2}^{T/2}\frac{\sin 2nwx}{2}\;dx - b_n\int_{-T/2}^{T/2}\frac{\cos 2nwx-1}{2}\;dx=\frac{T}{2}b_n$$

$$\Longrightarrow b_n=\frac{2}{T}\int_{-T/2}^{T/2} f(x)\sin nwx\;dx.$$

Fourier Series (Complex Form)

Euler's formula:

$$e^{ix} = \cos x + i \sin x, e^{-ix} = \cos x - i \sin x$$

$$\implies \cos x = \frac{e^{ix} + e^{-ix}}{2}, \sin x = \frac{e^{ix} - e^{-ix}}{2i}.$$

Therefore, Fourier series can be transformed to:

$$f(t) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nwt + b_n \sin nwt)$$

$$= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(\frac{a_n - ib_n}{2} e^{inwt} + \frac{a_n + ib_n}{2} e^{-inwt} \right).$$

Let $\dot{c}_0 = a_0, \dot{c}_n = a_n - ib_n, \dot{c}_{-n} = a_n + ib_n$ (complex amplitude), then:

$$f(t) \sim \frac{1}{2} \sum_{n=-\infty}^{\infty} \dot{c}_n e^{inwt}, \text{ with amplitude } A_n = |\dot{c}_{-n}| = |\dot{c}_n|.$$

Fourier Series (Complex Form)

Fourier Series (Complex Form)

$$f(t) \sim \frac{1}{2} \sum_{n=-\infty}^{\infty} \dot{c}_n e^{inwt}, \text{ where } \dot{c}_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) e^{-inwt} \, dt.$$

Orthogonality

Proof: $\{e^{inwt} \mid n \in \mathbb{Z}\}$ is orthogonal basis.

$$< e^{inwt}, e^{inwt} >= \int_{-T/2}^{T/2} e^{inwt}\overline{e^{inwt}}\ dt = \int_{-T/2}^{T/2} e^{inwt}e^{-inwt}\ dt = T > 0,$$

$$< e^{inwt}, e^{imwt} >= \int_{-T/2}^{T/2} e^{inwt}\overline{e^{imwt}}\ dt = \frac{e^{i(n-m)wt}}{i(n-m)w}\bigg|_{-T/2}^{T/2} = 0.\ (w = \frac{2\pi}{T})$$

$\square$

$$f(t) \sim \frac{1}{2}\sum_{n=-\infty}^{\infty} \dot{c}_n e^{inwt} = \frac{1}{T}\sum_{n=-\infty}^{\infty} [\int_{-T/2}^{T/2} f(x)e^{-inwx}\ dx]\, e^{inwt}$$

$$= \sum_{n=-\infty}^{\infty} \frac{< f(t), e^{inwt} >}{< e^{inwt}, e^{inwt} >} e^{inwt} = \sum_{n=-\infty}^{\infty} \frac{< f(t), e^{inwt} >}{T} e^{inwt}.$$

Fourier Transform (Definition)

Fourier Transform

f(x) is absolutely integrable on $(-\infty, +\infty)$. The Fourier Transform of f(x) is defined as:

$$\mathcal{F}[f] = F(w) = \int_{-\infty}^{+\infty} f(x) e^{-iwx}\, dx.$$

The Inverse Fourier Transform can be written as:

$$f(x) = \mathcal{F}^{-1}[\mathcal{F}[f]] = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(w) e^{iwx}\, dw.$$

Orthogonality

Proof: $\{e^{iwt} \mid w \in \mathbb{R}\}$ is orthogonal basis (namly Fourier basis).

$$< e^{iw_1 t}, e^{iw_2 t} >= \int_{-\infty}^{+\infty} e^{iw_1 t}\overline{e^{iw_2 t}}\, dt = \int_{-\infty}^{+\infty} e^{i(w_1 - w_2)t}\, dt$$

$$= \lim_{a\to\infty} \int_{-a}^{+a} e^{i(w_1 - w_2)t}\, dt = \lim_{a\to\infty} \frac{e^{i(w_1-w_2)t}}{i(w_1 - w_2)}\bigg|_{-a}^{a} = \lim_{a\to\infty} \frac{2i\sin(w_1 - w_2)a}{i(w_1 - w_2)}$$

$$= \lim_{a\to\infty} 2a\frac{\sin(w_1 - w_2)a}{(w_1 - w_2)a} = 2a\pi\delta((w_1 - w_2)a) = 2a\pi\frac{\delta(w_1 - w_2)}{a}$$

$= 2\pi\delta(w_1 - w_2)$, where δ is Dirac delta function:

$$\delta(x) \approx \begin{cases} +\infty, & x = 0, \\ 0, & x \neq 0. \end{cases}$$

$\square$

Orthogonality

Inverse Fourier Transform:

$$f(x) = \mathcal{F}^{-1}[\mathcal{F}[f]] = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(w) e^{iwx} \, dw.$$

We can write Fourier Transform F(w) as:

$$< f(x), e^{iwt} >$$

Then f(x) can be transformed to:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} < f(x), e^{iwt} > e^{iwx} \, dw$$

Fourier Series vs. Fourier Transform

Fourier Series:

- f(x) has period T.
- $f(x) = \frac{1}{2}\sum_{n=-\infty}^{+\infty} \dot{c}_n e^{inwx}$, where $\dot{c}_n = \frac{2}{T}\int_{-T/2}^{T/2} f(x)e^{-inwx}\,dx$.

Fourier Transform:

- f(x) is non-periodic function.
- $f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{iwx}\,dw$, where $F(w) = \int_{-\infty}^{+\infty} f(x)e^{-iwx}\,dx$.

From Fourier Series to Fourier Transform

For any non-periodic function $f(x)$, we can construct a periodic function $f_T(x)$ which satisfies $f_T(x) = f(x)$ for $|x| < T/2$. Then we have:

$$\lim_{T \to +\infty} f_T(x) = f(x).$$

We can write $f_T(x)$ using Fourier Series:

$$f_T(x) = \frac{1}{2} \sum_{n=-\infty}^{+\infty} \dot{c}_n e^{inwt} = \frac{1}{2} \sum_{n=-\infty}^{+\infty} \Big[\frac{2}{T} \int_{-T/2}^{T/2} f(x) e^{-inwx}\, dx\Big] e^{inwt}$$

$$= \frac{1}{T} \sum_{n=-\infty}^{+\infty} \Big[\int_{-T/2}^{T/2} f(x) e^{-inwx}\, dx\Big] e^{inwt}.$$

$$f(x) = \lim_{T \to +\infty} \frac{1}{T} \sum_{n=-\infty}^{+\infty} \Big[\int_{-T/2}^{T/2} f(x) e^{-iw_n x} \, dx \Big] e^{iw_n t}, \text{ where } w_n = nw = \frac{2\pi n}{T}.$$

Denote $\Delta w = w_n - w_{n-1} = \frac{2\pi}{T}$, then $T = \frac{2\pi}{\Delta w}$. Therefore,

$$f(x) = \lim_{\Delta w \to 0} \frac{1}{2\pi} \sum_{n=-\infty}^{+\infty} \Big[\int_{-T/2}^{T/2} f(x) e^{-iw_n x} \, dx \Big] e^{iw_n t} \Delta w$$

$$= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \Big[\int_{-\infty}^{\infty} f(x) e^{-iwx} \, dx \Big] e^{iwt} \, dw = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(w) e^{iwt} \, dw$$

Fourier Transform F(w) is the proposition of e^{iwt} in f(x). We say F(w) is the representation in frequency (or spectral) domain.

Fourier Transform Properties

- Linear combination

$$\mathcal{F}[af + bg] = a\mathcal{F}[f] + b\mathcal{F}[g]$$

- Translation

$$\mathcal{F}[f(x - c)](\lambda) = e^{-i\lambda c}\mathcal{F}[f](\lambda)$$

- Rescaling

$$\mathcal{F}[f(cx)](\lambda) = \frac{1}{c}e^{-i\lambda c}\mathcal{F}[f](\frac{\lambda}{c})$$

Fourier Transform: Convolution

Convolution

The convolution of two functions f and g is defined as

$$f * g\,(x) = \int_{-\infty}^{+\infty} f(x-t)g(t)\ dt.$$

Convolution Theorem

$$\mathcal{F}[f * g] = \mathcal{F}[f]\mathcal{F}[g]$$
$$\mathcal{F}[fg] = \mathcal{F}[f] * \mathcal{F}[g]$$
$$\mathcal{F}^{-1}[\mathcal{F}[f]\mathcal{F}[g]] = f * g$$
$$\mathcal{F}^{-1}[\mathcal{F}[f] * \mathcal{F}[g]] = fg$$

Convolution Theorem Proof

Proof: Convolution Theorem.

$$f(x-t) = \mathcal{F}^{-1}[e^{-i\lambda t}\mathcal{F}(f)](x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathcal{F}[f](\lambda)e^{i\lambda(x-t)}\ d\lambda,$$

Therefore, $f * g\ (x) =$

$$\int_{-\infty}^{+\infty} f(x-t)g(t)\ dt = \int_{-\infty}^{+\infty} \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathcal{F}[f](\lambda)e^{i\lambda(x-t)}\ d\lambda\ g(t)\ dt$$

$$= \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathcal{F}[f](\lambda)e^{i\lambda x}\int_{-\infty}^{+\infty} e^{-i\lambda t}\ g(t)\ dt\ d\lambda$$

$$= \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathcal{F}[f](\lambda)\mathcal{F}[g](\lambda)e^{i\lambda x}\ d\lambda = \mathcal{F}^{-1}[\mathcal{F}[f]\mathcal{F}[g]](x).$$

$\square$

Linear and Time Invariant Filter

A *filter* $\mathcal{L}$ maps f in to another function $\mathcal{L}[f]$.
Linear filter: $\mathcal{L}[af + bg] = a\mathcal{L}[f] + b\mathcal{L}[g]$.
Time invariant filter: $\mathcal{L}[f(x - c)](t) = \mathcal{L}[f(x)](t - c)$.

Theorem

Let L be a linear and time invariant filter. Then there exists a function h such that

$$\mathcal{L}[f] = f * h.$$

Therefore,

$$\mathcal{F}[\mathcal{L}[f]] = \mathcal{F}[f]\mathcal{F}[h].$$

Contents

Graph Laplacian (Definition)

Graph Laplacian

The Graph Laplacian is defined as

$$\mathcal{L} = D - W,$$

where W is adjacency matrix, D is a diagonal matrix with $D_{ii} = \sum_j w_{ij}$.
The normalized Graph Laplacian is defined as

$$\mathcal{L} = D^{-1/2}(D - W)D^{-1/2} = I - D^{-1/2}WD^{-1/2}$$

Graph Laplacian (Definition)

Non-normalized Graph Laplacian:

$$\mathcal{L}_{ij} = \begin{cases} d_i, & i = j \; (when \;\; W_{ii} = 0) \\ -w_{ij}, & (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$$

Normalized Graph Laplacian:

$$\mathcal{L}_{ij} = \begin{cases} 1, & i = j \; (when \;\; W_{ii} = 0) \\ -\frac{w_{ij}}{\sqrt{d_i d_j}}, & (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$$

Eigenvalues decomposition

Eigenvalues decomposition

We can decompose any square, symmetric $n \times n$ matrix S as

$$S = U \Lambda U^T = \sum_{i=0}^{n-1} \lambda_i u_i u_i^T,$$

where $\Lambda = diag(\lambda_0, ..., \lambda_{n-1})$, with $\lambda_0 \leq ... \leq \lambda_{n-1}$ the eigenvalues. $U = [u_0, ..., u_{n-1}]$ is a $n \times n$ orthonormal matrix ($U^T U = I$) that contains the eigenvectors.

Graph Laplacian (Spectral Properties)

Why Graph Laplacian (GL) is so important?

- GL is a symmetric matrix, that means GL has real eigenvalues and eigenvectors, and **the eigenvectors forms an orthogonal basis**.
- GL is positive semidefinite, and the eigenvalues are non-negative.
- GL has an eigenpair $(\lambda_0 = 0, \mathbf{u}_0 = \mathbf{1})$, this is because $\sum_j \mathcal{L}_{ij} = 0$, and $\mathcal{L}\mathbf{u}_0 = 0\mathbf{u}_0$.
- The multiplicity of the eigenvalue zero is equal to the number of connected components of the graph.
- Asymmetric GL: $D^{-1}(D - W) = I - D^{-1}W = P$ is the Markov random walk matrix.
- For Normalized GL, $0 = \lambda_0 \leq \lambda_1 ... \leq \lambda_{n-1} \leq 2$, with $\lambda_{n-1} = 2$ if and only if G is bipartite.

Graph Laplacian (Spectral Properties)

Property

If G is connected, the eigenvectors associated to the d smallest non-zero eigenvalues provides the best d-dimensional embedding of G.

Graph embedding:

Find $\mathbf{f} : V \to \mathbb{R}^d$, $\mathbf{f}(v_i) = [f_1(v_i), f_2(v_i), ..., f_d(v_i)]$, that minimizes

$$F^T \mathcal{L} F = \sum_{(i,j)\in\mathcal{E}} W_{ij} \|\mathbf{f}(v_i) - \mathbf{f}(v_j)\|^2, \text{ s.t. } F^T \mathbf{1} = \mathbf{0},$$

where $F = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_d]$. Imposing $\|\mathbf{f}_i\| = 1$, the solution is given by

$$F = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_d].$$

Contents

Graph Fourier Transform (Definition)

Graph Fourier Transform

Let $f : V \to \mathbb{R}$, the Graph Fourier Transform os f is defined as

$$\mathcal{F}[f](\lambda_l) = \hat{f}(\lambda_l) = < f, u_l > = \sum_{i=1}^{n} f(i) u_l(i)$$

Inverse Graph Fourier Transform

Let $f : V \to \mathbb{R}$, the Inverse Graph Fourier Transform os f is defined as

$$\mathcal{F}^{-1}[\mathcal{F}[f]](i) = f(i) = \sum_{l=0}^{n-1} \hat{f}(\lambda_l) u_l(i)$$

The eigenvalues of $\mathcal{L}$ represent the frequencies, and the eigenvectors are the Fourier basis. We say $\hat{f}$ is spectral domain, and f is vertex domain.

Graph Convolution (Definition)

The classical convolution is defined as

$$f * g\ (x) = \int_{-\infty}^{+\infty} f(x-t)g(t)\ dt.$$

This formula cannot be applied on graph because the translation $f(x-t)$ is not defined in the context of graphs.
However, convolution can be defined as

$$f * g = \mathcal{F}^{-1}[\mathcal{F}[f * g]].$$

We have $\mathcal{F}[f * g] = \mathcal{F}[f]\mathcal{F}[g]$. We can define Graph Convolution as

$$f * g\ (i) = \sum_{l=0}^{n-1} \hat{f}(\lambda_l)\hat{g}(\lambda_l)u_l(i) = \sum_{l=0}^{n-1} u_l(i)\hat{g}(\lambda_l)\sum_{j=1}^{n} u_l(j)f(j).$$

Graph Convolution (Properties)

Properties of Graph Convolution (it satisfies all convolution theorem):

- $\mathcal{F}[f * g] = \mathcal{F}[f]\mathcal{F}[g]$
- $f * g = g * f$
- $f * (g + h) = f * g + f * h$
- $\sum_{i=1}^{n} f * g\ (i) = \sqrt{n}\ \hat{f}(0)\ \hat{g}(0)$

Graph Filter (The Fundamental of GCN)

Consider a filter on $f(i)$ defined as $\mathcal{L}[f](i) = f * h\ (i)$.
According to definition of convolution,

$$\mathcal{L}[f](i) = f * h\ (i) = \sum_{l=0}^{n-1} \hat{f}(\lambda_l)\hat{h}(\lambda_l)u_l(i).$$

Supposing $\hat{h}(\lambda_l)$ **takes polynomial on spectral domain**,

$$\hat{h}(\lambda_l) = \sum_{k=0}^{K} a_k \lambda_l^k.$$

Then we get

$$\mathcal{L}[f](i) = \sum_{j=1}^{n} f(j) \sum_{k=0}^{K} a_k \sum_{l=0}^{n-1} \lambda_l^k u_l(j) u_l(i).$$

Graph Filter (The Fundamental of GCN)

$$\mathcal{L}[f](i) = \sum_{j=1}^{n} f(j) \sum_{k=0}^{K} a_k \sum_{l=0}^{n-1} \lambda_l^k u_l(j) u_l(i).$$

We have

$$(\mathcal{L}^k)_{ij} = \sum_{l=0}^{n-1} \lambda_l^k u_l(j) u_l(i),$$

and $(\mathcal{L}^k)_{ij} = 0$ if the shortest distance between nodes i and j is greater than k. Let

$$b_{ij} = \sum_{k=0}^{K} a_k (\mathcal{L}^k)_{ij},$$

we get

$$\mathcal{L}[f](i) = \sum_{j \in N_k(i)} b_{ij} f(j).$$

Graph Filter (The Fundamental of GCN)

Now we get

$$\mathcal{L}[f](i) = f * h\ (i) = \sum_{j \in N_k(i)} b_{ij} f(j)$$

with the assumption that

$$\hat{h}(\lambda_l) = \sum_{k=0}^{K} a_k \lambda_l^k.$$

Therefore, if the filter is a K-order polynomial on the spectral domain, the filtered f*h is a linear combination of the original signals in the K-neighborhood of node i.

Contents

GCN with Fast Localized Spectral Filtering

Overview of GCN with Fast Localized Spectral Filtering

- Task: New theoretical formulation
- Input: Undirected weighted graph G

Contribution

- Spectral filtering: Propose a theoretical formulation of CNNs on graph.
- Localized: The proposed spectral filters are provable to be strictly localized in a ball of radius K (K hops).
- Fast: Avoid the expensive eigenvalue decomposition for Fourier basis, leading to a linear complexity w.r.t the input data size n.

Spectral filtering

A signal x filtered by g is defined as

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x.$$

Recall that we analyzed

$$f * g\ (i) = \sum_{l=0}^{n-1} u_l(i)\hat{g}(\lambda_l) \sum_{j=1}^{n} u_l(j) f(j).$$

Polynomial parametrization

The polynomial filter is defined as

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k.$$

Recall that we analyzed

$$\hat{g}(\lambda_l) = \sum_{k=0}^{K} a_k \lambda_l^k.$$

The filter is K-localized. The learning complexity is O(k), the same complexity as classical CNNs.

Recursive formulation for fast filtering

- The cost to filter a signal x as $y = Ug_\theta(\Lambda)U^T x$ is still high with $O(n^2)$ operations because of the multiplication with the Fourier basis U.
- A solution to this problem is to parametrize $g_\theta(L)$ as a polynomial function that can be **computed recursively from L**, as K multiplications by a sparse L costs $O(K|E|) \ll O(n^2)$.
- Solution: Chebyshev polynomial $T_k(x)$ of order k may be computed by the stable recurrence $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$.
- Chebyshev polynomial form an orthogonal basis for $L^2([1, 1], dy/\sqrt{1 - y^2})$ (**square integrable on [-1, 1]**).

Recursive formulation for fast filtering

Rescaling Λ to [-1, 1]:

$$\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I.$$

Thus

$$\tilde{L} = U\tilde{\Lambda}U^T = U(\frac{2\Lambda}{\lambda_{max}} - I)U^T = \frac{2L}{\lambda_{max}} - UU^T = \frac{2L}{\lambda_{max}} - I.$$

We have $\bar{x}_0 = x, \bar{x}_1 = \tilde{L}x$, then

$$\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}.$$

Finally we get

$$y = g_\theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x = [\bar{x}_0, ..., \bar{x}_{K-1}]\theta.$$

Output of convolution layer

The j^{th} output feature map of the sample s is given by

$$y_{s,j} = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L) x_{s,i} \in \mathbb{R}^n.$$

Trainable parameters: $F_{in} \times F_{out}$ Chebyshev coefficients $\theta_{i,j} \in \mathbb{R}^k$.

Contents

Semi-Supervised Classification with GCN

Overview of Semi-Supervised Classification with GCN

- Task: Semi-Supervised Classification
- Input: Undirected graph G, not weighted

Contribution

- An efficient variant of GCN: Propose a localized first-order approximation of spectral graph convolutions.
- Semi-supervised classification: Propose a scalable approach for semi-supervised learning.

Semi-Supervised Classification with GCN

Recall that in the previous paper we get

$$y = g_\theta(L)x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L})x,$$

where $\tilde{L} = U\tilde{\Lambda}U^T = \frac{2L}{\lambda_{max}} - I$.

In this paper, K is limited to 1, and replace L to the normalized form $I - D^{-1/2}AD^{1/2}$.
For normalized Graph Laplacian, there is a theory says $\lambda_{max} <= 2$, and $\lambda_{max} = 2$ if and only id G is bipartite. Further the paper approximate $\lambda_{max} = 2$, and gets

$$y = g_\theta(L)x \approx \theta_0 x - \theta_1 D^{-1/2}AD^{-1/2}x$$

with two free parameters θ_0 and θ_1.

Semi-Supervised Classification with GCN

$$y = g_\theta(L)x \approx \theta_0 x - \theta_1 D^{-1/2} A D^{-1/2} x$$

To constrain the number of parameters further to address overfitting and to minimize the number of operations, the paper constrains $\theta = \theta_0 = -\theta_1$ and gets

$$y = g_\theta(L)x \approx \theta(I + D^{-1/2} A D^{1/2})x.$$

To avoid numerical instabilities and exploding/vanishing gradients, the paper introduces a renormalization trick with $\tilde{A} = A + I$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and gets

$$y = g_\theta(L)x \approx \theta \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} x.$$

The formula can be generalized to C input channels and F filters, and gets the well-know GCN formula

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)}),$$

where $W^{(l)} \in \mathbb{R}^{C \times F}$ is trainable parameters.

Semi-Supervised Classification with GCN

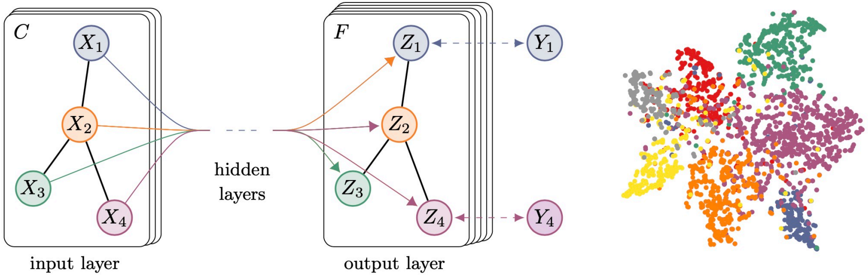
$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$$

can also be written as (here $d_i = |N_i|$)

$$h_i^{(l+1)} = \sigma(\sum_{j \in N_i} \frac{1}{\sqrt{d_i d_j}} h_j^{(l)} W^{(l)}).$$

Semi-supervised classification with GCN

$$Loss = -\sum_{l \in Y_L} \sum_{f=1}^{F} Y_{lf} ln(Z_{lf})$$



(a) Graph Convolutional Network

(b) Hidden layer activations

Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

Contents

R-GCN

Overview of R-GCN

- Task: (knowledge base) Link prediction (recovery of missing facts) and entity classification (recovery of missing entity attributes)
- Input: **Directed graph**

Contribution

- Model GCN to multiple directed graphs.
- Propose two regularization methods.

R-GCN

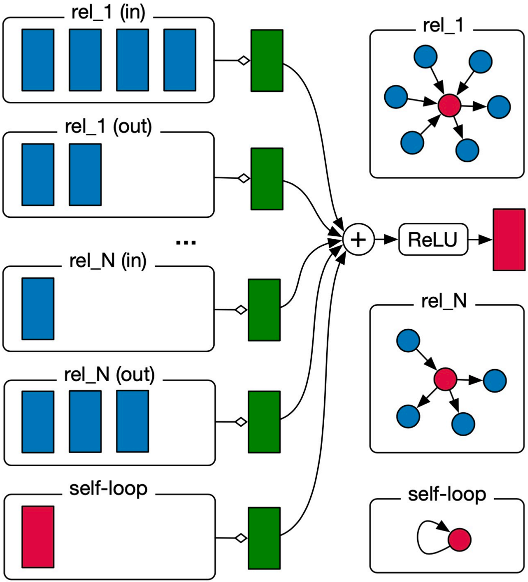
Input of R-GCN:

- Each relation is a directed graph (so input is directed multi-graph).
- Different relations share the same vertices set.
- To handle the directed graph, the paper separate it into two graphs, thus **consider the in-edges and out-edges independently**.
- It adds a self-loop graph.

R-GCN uses the propagation formula:

$$h_i^{(l+1)} = \sigma(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{d_i^r} W_r^{(l)} h_j^{(l)} + W_0 h_i^{(l)}).$$

R-GCN



rel_1 (in)
rel_1 (out)
...
rel_N (in)
rel_N (out)
self-loop
+
ReLU
rel_1
rel_N
self-loop

The normalization

- Basic decomposition constrains shared basis between relations:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)}.$$

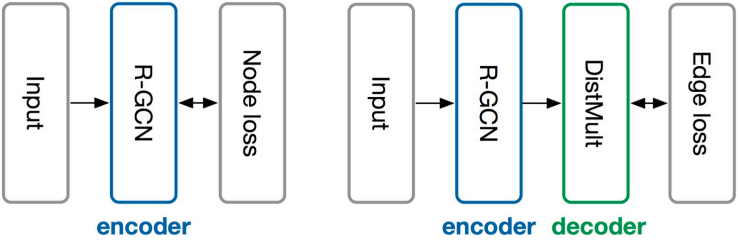
- Block-diagonal decomposition constrains sparse weight matrix:

$$W_r^{(l)} = \oplus_{b=1}^{B} Q_{br}^{(l)}.$$

In other words,

$$W_r^{(l)} = diag(Q_{1r}^{(l)}, ..., Q_{Br}^{(l)}).$$

R-GCN



(a) Entity classification (b) Link prediction

- Entity classification: predict missing entities.
- In link prediction, DistMult associates every relation r with a diagonal matrix $R_r \in \mathbb{R}^{d \times d}$, and give a score to a link $f(s, r, o) = e_s^T R_r e_o$.

The loss functions

- Entity classification:

$$Loss = -\sum_{i \in Y} \sum_{k=1}^{K} t_{ik} lnh_{ik}^{(L)}.$$

- Link prediction:

$$Loss = -\frac{1}{(1+w)|\hat{\mathcal{E}}|} \sum_{(s,r,o,y) \in \tau} ylog\, l(f(s,r,o)) + (1-y) log(1 - l(f(s,r,o))),$$

where l is the logistic sigmoid function.

Reference

- mathworld.wolfram.com
- Graph Signal Processing Seminars of Prof. Luis Gustavo Nonato
- Math behind GCN of Zhiping Xiao
- "Mathematical Analysis" of CHEN CHUANZHANG

The End