Spectral networks and locally connected networks on graphs

School of Industrial and Management Engineering, Korea University

Sae Rin Lim

KOREA UNIVERSITY

DMQA

• Contents

• Introduction

❖ Spectral Networks and Deep Locally Connected Networks on Graphs

- 2014 ICLR에 게재, 2021년 12월 17일 기준 2948회 인용
- Euclidean space(grid)에서 좋은 성능을 보인 CNN을 Non-euclidean space로 일반화하는 방법 두 가지로 제안
  - ➢ Hierarchical clustering을 기반으로 Spatial construction 정의
  - ➢ Spectrum of the graph Laplacian을 기반으로 spectral construction 정의
- 이번 스터디에서는 Spectral construction을 중점적으로 리뷰

Spectral Networks and Deep Locally Connected Networks on Graphs

Joan Bruna
New York University
bruna@cims.nyu.edu

Wojciech Zaremba
New York University
woj.zaremba@gmail.com

Arthur Szlam
The City College of New York
aszlam@ccny.cuny.edu

Yann LeCun
New York University
yann@cs.nyu.edu

• Background

❖ Fourier transform

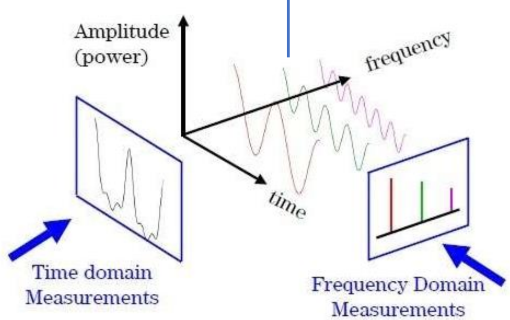
- Fourier transform : 임의의 입력 신호를 다양한 frequency(u)를 갖는 주기함수들의 합으로 분해하여 표현
- 일반적으로 sin, cos함수를 주기함수로 하여 입력 신호를 분해

입력 신호 $$f(x)=\int_{-\infty}^{\infty}F(u)e^{j2\pi ux}du \quad (1)$$ Inverse Fourier transform

https://harryjo97.github.io/theory/Graph-Fourier-Transform/

주기함수 성분의 계수
(성분의 강도 Amplitude) $$F(u)=\int_{-\infty}^{\infty}f(x)e^{-j2\pi ux}dx \quad (2)$$ Fourier transform

- 식(1)은 입력신호 $f(x)$가 $e^{j2\pi ux}$들의 합으로 표현(분해)된다는 의미
- 식(2)는 $f(x)$를 주기함수 성분으로 분해했을 때의 계수 F(u)가 식 (2)로 주어진다는 의미



DMQA

• Background

❖ Fourier transform

- 오일러 공식을 통한 주기함수($e^{j2\pi ux}$) 변형
- $e^{j\theta} = cos\theta + jsin\theta$ (3) $Euler's\ formula$
- $\rightarrow e^{j2\pi ux} = cos2\pi ux + jsin2\pi ux$ (4)
- sin(x)와 cos(x)의 주기는 2π이기 때문에 (4)식의 cos와 sin함수는 주기가 1/u, frequency 가 u가 됨
- 즉, 식 (1)은 입력 신호 $f(x)$를 모든 가능한 주파수(u)의 주기함수들($e^{j2\pi ux}$)의 일차결합으로 표현한 것이고 그 일차결합 계수F(u)는 식 (2)를 통해 주어짐
- 식(1)과 식(2)의 이러한 관계가 모든 임의의 입력신호에 대해 만족하기 때문에 신호함수는 항상 주기함수들의 일차 결합으로 분해될 수 있음(증명은 reference 참고)

DMQA

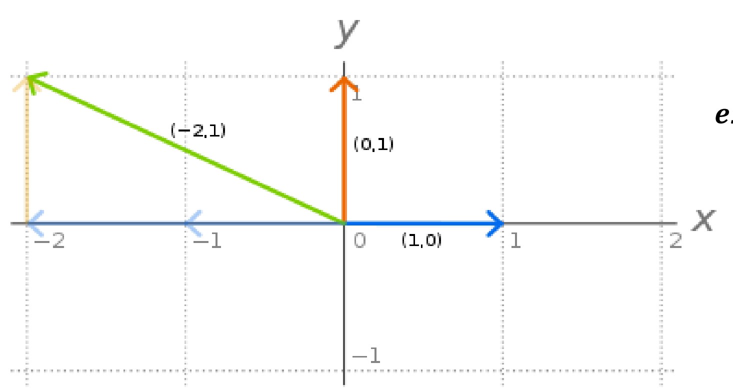
• Background

❖ Linear algebra and Fourier transform

- 식(1)을 통해 모든 신호 함수는 주기함수들의 선형결합으로 표현되며 이 때 주기함수가 discrete하다면 입력 신호에 대한 vector space의 basis vector가 됨(주기함수 집합 = 입력 신호 vector space의 basis vector 집합)
- 이러한 basis vector를 정규화하여 단위벡터로 변경하면 orthnormal한 특징을 가지게 됨
- 임의의 벡터 $v = a_1 v_1 + a_2 v_2 + \ldots + a_n v_n$ (v_i : *basis vectors*)에서 $a_i = v \cdot v_i$로 쉽게 계산이 가능

$$\because v \cdot v_i = (a_1 v_1 + a_2 v_2 + \ldots + a_n v_n) \cdot v_i = a_1 \underbrace{v_1 \cdot v_i}_{0} + \cdots + a_i v_i \cdot v_i + \cdots + a_n \underbrace{v_n \cdot v_i}_{0} = a_i \underbrace{v_i \cdot v_i}_{1} = a_i$$

- 즉, 만약 주기함수가 discrete 하다면 주기함수의 계수인 F(u)는 **내적을 통해 쉽게 구할 수 있음**
- 기하학적으로 해석하면 입력신호를 basis vector로 projection한 vector의 크기



$$ex.\,(-2,1) = \underline{\{(-2,1)\cdot(1,0)\}}(1,0) + \underline{\{(-2,1)\cdot(0,1)\}}(0,1)$$
$$= -1 * (1,0) + 1 * (0,1)$$

dot product of basis vector and input signal

Example of basis vector : 2d Euclidean space에서는 [1,0]과 [0,1]의 선형결합으로 모든 vector를 표현 할 수 있다
이러한 basis vector들은 서로의 내적이 0인 orthgonal 한 특징을 가진다

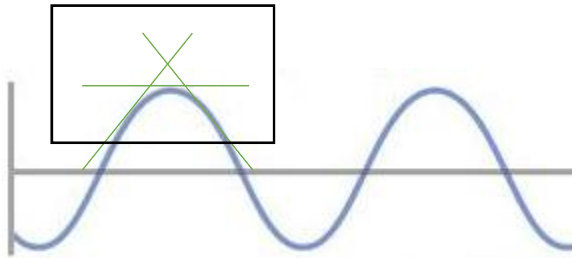
DMQA

• Background

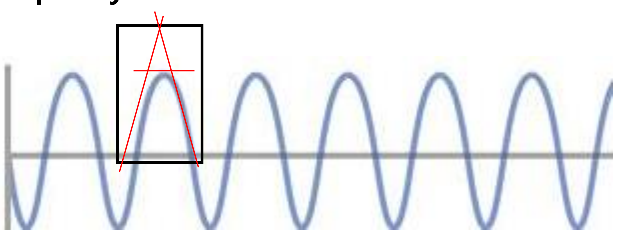
❖ Laplacian(Laplace Operator)

- Laplacian $\Delta f = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$ (5)
- Laplacian은 이계도 함수로 **"어떤 x에서의 변화(기울기)의 정도"**를 나타냄
- 변화의 정도는 frequency로 해석할 수 있으며 변화의 정도가 크면 frequency가 크고 작으면 frequency가 낮은 비례관계를 가짐
- 또한, smoothness로도 볼 수 있으며 smoothness는 변화의 정도와 반비례한 관계를 가짐

기울기 변화가 천천히
Low frequency = High smoothness



기울기 변화가 급격하게
High frequency = Low smoothness



https://www.everexceed.com/online-high-frequency-ups-and-its-advantages_n134

DMQA

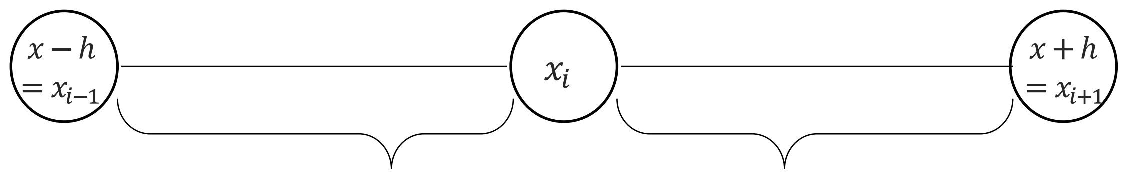
• Spectral Construction

❖ Graph Laplacian

- 1차원에서의 Laplacian은 다음과 같다

$$\Delta f = \nabla^2 f = \lim_{h\to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \qquad (6)$$

- $f(x_i)$ = **특정 노드 x_i 를 node signal f_i 로 mapping해주는 함수**라고 하고 식(6)을 그래프로 확장해보면,
  그래프는 discrete하기 때문에 Laplacian은 이웃 노드와의 차이로 표현된다
  즉, 그래프에서 한 노드에서 **"변화의 정도"**는 이웃노드와의 차이를 통해 알 수 있음



그래프는 discrete하기 때문에 naive하게 해석하면
이웃 노드의 의미가 됨

$$\lim_{h\to 0}\left(\frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x-h)-f(x)}{h}}{h}\right)$$

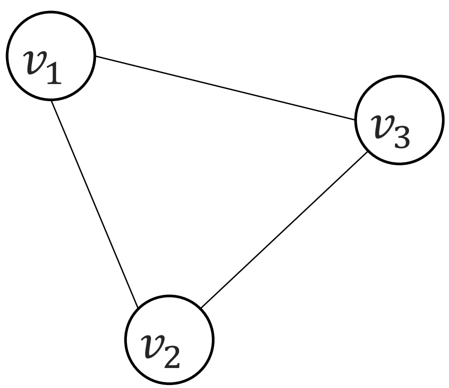
$$= \lim_{h\to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

DMQA

• Spectral Construction

❖ Laplacian Matrix

- Laplacian matrix(L) = Degree matrix(D) – Adjacency matrix(A)
- Matrix D와 matrix A는 real symmetric하기 때문에 Matrix L 역시 symmetric한 성질 가짐
- 임의의 vector x $\in R^n$ 에 대해, $x^T L x = x^T D x - x^T W x = \frac{1}{2} \sum_{i,j} W_{i,j} (x_i - x_j)^2 \geq 0$ (7) 을 만족하기 때문에 Matrix L 은 Positive semi-definite하고 이로 인해 Non-negative real eigenvalue를 가짐 **(Appendix A 참조)**
- 또한 real-valued symmetric matrix는 항상 eigen decomposition이 가능하며 eigen vector는 orthogonal함



Graph G(V,E)

D

2	0	0
0	2	0
0	0	2

–

A

0	1	1
1	0	1
1	1	0

=

L

2	-1	-1
-1	2	-1
-1	-1	2

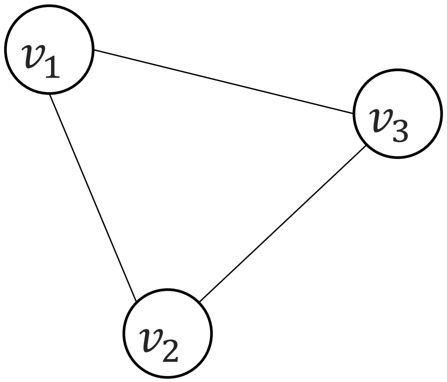
Positive semi-definite

• Spectral Construction

❖ Why D-A is called "Laplacian" Matrix?

- $In\ the\ Graph\ G(V,E), Let\ node\ v_i\ \in V\ with\ i = 1,2, \ldots, n\ and\ W = Weighted\ Maxtrix\ of\ G$
- $node\ v_i$의 $signal\ f_i \in R^{n}$에 대한 $Laplacian$은 이웃 노드와의 차이로 다음과 같음

$$\Delta f_i = \sum_{j \in N(v_i)} [f_i - f_j] = \sum_{j \in n} W_{ij}[f_i - f_j] \qquad (8)$$



Graph G(V,E)

0	1	1
1	0	1
1	1	0

A=W

- 식 (8)을 이용해 왼쪽의 그래프 G의 모든 노드에 대한 Laplacian

$$\Delta f_0 = 0 * (f_0 - f_0) + 1 * (f_0 - f_1) + 0 * (f_0 - f_2)$$
$$= 2f_0 - f_1 - f_2$$
$$\Delta f_1 = 2f_1 - f_0 - f_2$$
$$\Delta f_2 = 2f_2 - f_0 - f_1$$

- $$\Delta f = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \mathrm{L}f \qquad (9)$$

- Graph G의 모든 node signal 대한 Laplacian 값은 (D-A)라는 matrix에 node signal matrix f를 곱한 것이기 때문에 (D-A)가 Laplace operator, 즉 Laplacian이 됨

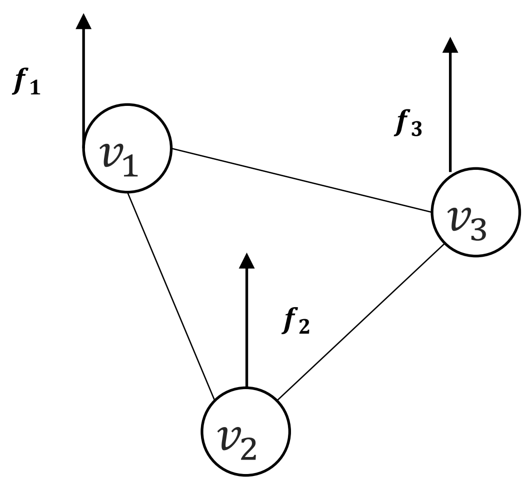
• Spectral Construction

❖ Laplacian quadratic form

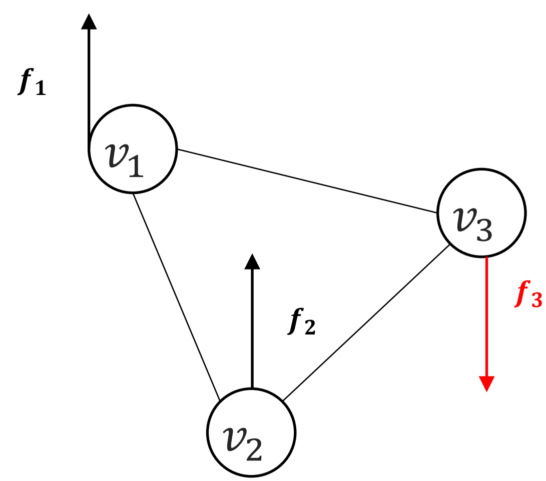
- 식 (8)을 통해 각각 node signal에 대한 Laplacian을 구할 수 있으며 이는 node signal smoothness를 나타냄
- 이를 모두 더하면 그래프 전체의 smoothness를 알 수 있지만 (+,-)부호 효과를 제거해야 함
- 때문에 식 (9)의 quadratic form을 이용하여 다음과 같이 정의(식 (7) 참고)

$$Laplacian\ quadratic\ form:\ f^T L f = \frac{1}{2}\sum_{i,j} W_{i,j}\left(f_i - f_j\right)^2 \qquad (10)$$

- Laplacian Quadratic form의 의미
  - ➢ Quadratic form 값이 작을수록 모든 노드에 대해 이웃 노드의 signal과 smoothness가 증가



Low frequency High smoothness graph signal



High frequency Low smoothness graph signal

DMQA

• Spectral Construction

❖ Eigen decomposition of Laplacian Matrix

- Laplacian Matrix 설명에서 L은 eigen decomposition이 가능하다는 것을 확인
- $Lu_i = \lambda_i u_i$ 를 만족하고 $u_i^T u_j = 1 \; if \; i = j \; else \; 0(orthnormal)$할 때,
- $eigen \; dicomposition \; of \; L : L = U\Lambda U^T \quad (11) where, \Lambda = diag(\lambda_i) = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}, \; U = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}^T$
- 식(11) $f^T L f$을변형하면$U^T L U = \Lambda \quad (\because U^T U = I \rightarrow U^{-1} = U^T) \qquad (12)$

  식(10)의 Laplacian quadratic form에서 f에 U를 대입하면 같은 식이 됨
- 특정 eigen vector u_i에서 Laplacian quadratic form의 값은 λ_i이며 Laplacian quadratic form의 값이 그래프의 smoothness(or frequency)를 나타냄
- 즉, $\boldsymbol{\lambda_i}$는 **그래프의 특정 frequency(<u>local minimum</u>)**을 나타내며 이 때, ***node signal f*** 가 **존재하는 공간은 $\boldsymbol{u_i}$를 basis vector**로 가지는 공간이기 때문에 *node signal f* 를 u_i의 선형결합으로 표현할 수 있음

- $minimize_f \; f^T L f$ $\quad s.t \; f \in R^n \; and \; ||f||_2 = 1$ $\xrightarrow{\text{Lagrange multiplier}}$ $minimize_f \; Lag = f^T L f - \lambda(f^T f - 1)$

  $\frac{\partial Lag}{\partial f} = 2Lf - 2\lambda f = 0 \rightarrow Lf = \lambda f$

• Spectral Construction

❖ Graph Fourier Transform

- 모든 *node signal* f 를 Laplacian Matrix의 eigen vector로 표현할 수 있기 때문에 eigen value= frequency, eigen vector = 특정 frequency를 나타내는 주기함수인 Fourier Transform으로 해석할 수 있음
- 따라서 Graph Fourier Transform은 아래와 같은 식으로 표현됨
- $let\ basis\ vector\ U = (u_1, u_2, \dots, u_n)\ u_i \in R, i = 1,2, \dots, n\ from\ Laplacian\ matrix\ L,$
- $Nodes'\ signal\ f = (f_1, f_2, \dots, f_n)^T$

$$Fourier\ transform: \hat{f} = \left\{\hat{f}(\lambda_1), \hat{f}(\lambda_2), \dots, \hat{f}(\lambda_n)\right\}^T = U^T f \qquad (13)$$

(p.8 기저벡터와 입력신호의 내적=계수)

$$Inverse\ Fourier\ transform: f = U\hat{f}\ (\because U^T = U^{-1})\ <-> \ f = \sum_{i=1}^{n} \hat{f}(\lambda_i) u_i \qquad (14)$$

• Spectral Construction

❖ Graph Fourier Transform

$$Fourier\ transform: \hat{f} = \{\hat{f}(\lambda_1), \hat{f}(\lambda_2), \dots, \hat{f}(\lambda_n)\}^T = U^T f$$

$$= \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} [f_1 \quad \cdots \quad f_n] = \begin{bmatrix} u_1 f_1 & u_1 f_i & u_1 f_n \\ \vdots & \vdots & \vdots \\ u_n f_1 & u_n f_i & u_n f_n \end{bmatrix} = \begin{bmatrix} \hat{f}(\lambda_1) \\ \vdots \\ \hat{f}(\lambda_n) \end{bmatrix}$$

$columns:\ node\ signal\ f_i$에 대한 계수 집합

$$Inverse\ Fourier\ transform: f = U\hat{f}\ (\because U^T = U^{-1}) <-> f = \sum_{i=1}^{n} \hat{f}(\lambda_i) u_i$$

$$= [u_1 \quad \cdots \quad u_n] \begin{bmatrix} u_1 f_1 & u_1 f_i & u_1 f_n \\ \vdots & \vdots & \vdots \\ u_n f_1 & u_n f_i & u_n f_n \end{bmatrix} = [(\underbrace{u_1^T u_1}_{1} f_1 + \underbrace{u_1^T u_2}_{0} f_1 + \cdots + \underbrace{u_1^T u_n}_{0} f_1 = f_1),\ f_2, \cdots, f_n]$$

• Spectral Construction

❖ Graph Filtering

- 고전적인 신호처리에서 frequency filtering은 Fourier basis들의 영향력(계수)를 조절하는 것

$$\underbrace{f}_{\text{signals}} * \underbrace{h}_{\text{filter}} \rightarrow FT(f * h) = \hat{f}\hat{h} \qquad (14)$$ **Spectral domain에서 frequency filtering**

역변환

$$ITF\left(\hat{f}\hat{h}\right) = f * h = \sum_{i=1}^{n} \hat{f}(\lambda_i)\hat{h}(\lambda_i)u_i \quad (15)$$ **frequency filtering 결과를 다시 Spatial domain으로**

- 여기서 $\hat{h}(\lambda_i) = \sum_{k=0}^{K} a_k \lambda_i^k (polynomial\ on\ the\ spectral\ domain)$ (16) 이라고 가정한다면,

$the\ filtered\ signal\ in\ each\ node\ i\ is\ a\ linear\ combination\ of\ the\ original\ signal\ in\ the\ neighborhood\ of\ i$

- 즉, spectral domain에서의 frequency filtering을 적용하는 것은 spatial domain에서 이웃 노드의 signal 정보를 활용하는 것(증명 : reference [9] 참고)

• Spectral Construction

❖ Convolution

$$definition\ of\ convolution\ between\ two\ function\ f\ and\ g:\ (f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \quad (17)$$

- Convolution연산은 현재의 f(x)를 출력하기 위해 이전의 입력값에 대한 영향을 고려하여 계산하는 것
- Convolution 과정
  1. 함수 g를 y축으로 반전시키고 t만큼 전이
  2. 이동시킨 함수 g를 함수 f와 곱하여 출력
  3. τ를 이동시키며 모든 출력값을 더함
- 이러한 Convolution 연산은 주로 행렬에 대한 convolution(딥러닝), 신호의 필터(샘플링), 라플라스 변환, 푸리에 변환 등에서 사용
- 일반적으로 함수 f는 본래의 신호, 행렬, 이미지 등이며 g는 필터, 가중치 등으로 표현
- 즉, 함수 f가 주어졌을 때 원하는 목적에 따라 함수g를 선정하여 분해, 변환, 필터링 할 수 있음

• Spectral Construction

❖ Convolution Theorem

- 이러한 Convolution연산은 두 함수를 각각 *Fourier transform* 한 것의 곱셈으로 표현 됨
- 식(2)의 *Fourier transform*에 식(17) convolution 연산식을 대입하면,

$$FT\{(f*g)(t)\} = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau\right]e^{-j2\pi \mathrm{ut}}dt$$

$$= \int_{-\infty}^{\infty} f(\tau)\left[\int_{-\infty}^{\infty} g(t-\tau)e^{-j2\pi \mathrm{ut}}\,dt\right]d\tau$$

이 때, t에 대한 적분식은 아래와 같이 $g(t-\tau)$의 *Fourier transform* 이므로, $\int_{-\infty}^{\infty} g(t-\tau)e^{-j2\pi \mathrm{ut}}\,dt = FT\{g(t-\tau)\}$

$$= FT\{g(t)\}e^{-j2\pi u\tau}$$

$$= FT\{g(t)\}\boxed{\int_{-\infty}^{\infty} f(\tau)e^{-j2\pi u\tau}d\tau} = FT\{f(t)\}$$

$$= FT\{g(t)\}\cdot FT\{f(t)\} \qquad (18)$$

$$\therefore (f*g)(t) = IFT\Big\{FT\{g(t)\}\cdot FT\{f(t)\}\Big\} \quad (19)$$

• Spectral Construction

❖ Graph Convolution in spectral domain

- 앞에서 설명한 Convolution과 Graph Fourier transform을 이용한 Graph convolution은 다음과 같음:

$$let\ f : V \rightarrow R^n\ node\ signals,\ \ g \in R^m\ \ convolution\ filter$$

$$f * g = IFT\{FT\{g\} \cdot FT\{f\}\} = IFT(U^T f \odot U^T g), \quad \odot: element\ wise\ product$$

$$= U(U^T f \odot U^T g)$$

$$Assume\ filter\ g_\theta = diag(U^T g), then\ the\ graph\ convolution\ of\ f * g$$

$$= U(U^T f g_\theta) = U g_\theta U^T f \quad (\because g_\theta : dagonal\ matrix) \qquad (20)$$

- 이 때, g_θ를 diagonal matrix로 사용한 이유는 식 (16)처럼 polynomial parameterization 하여 이웃 노드의 정보를 활용하기 위함(아마도…)과 parameter의 사이즈를 CNN과 같이 효과적으로 줄이기 위함

• Spectral Construction

❖ Graph Convolution in spectral domain

- 논 논문에서는 Graph Convolution을 이용하여 graph 구조를 가지는 데이터를 학습시키는 방법론 제안

$$g_\theta = \theta_{i,j}^{(k)}$$

$$x_{k+1,j} = \sigma\left(\sum_{i=1}^{f_{k-1}} U\theta_{i,j}^{(k)} U^T x_k, i\right), j = 1,2,3, \dots, f_k \quad (21)$$

- $k = layer\ index$
- $f_k = number\ of\ filters (= channels)$
- $x_{k,j} = k^{th}\ layer's\ input\ signal\ in\ filter\ j$
- $g_\theta = \theta_{i,j}^{(k)}$
- $\sigma = activation\ function$

- 식 (21)을 통해서 convolution의 입력값과 출력값을 만들며 filter의 weight, 즉 $\theta_{i,j}^{(k)}$의 대각 성분을 학습
- 하지만 그래프의 작은 변화에 고유벡터들이 변하게 되며, 각 필터들은 domain dependent하기 때문에 그래프의 구조가 달라지면 적용할 수 없음
- 또한 eigen decomposition을 계산하기 위한 복잡도가 높음

• Conclusion

❖ Spectral Networks and Deep Locally Connected Networks on Graphs

- 해당 논문은 Euclidean space에서 표현되는 이미지, 시그널 등에서 큰 성공을 거둔 CNN을 Non-euclidean space에서 표현되는 graph에 적용하기 위해 그래프 신호 처리 이론 기반의 방법론 제안
- Spectral domain에서의 convolution 적용은 다른 연구에 많은 영향을 미치며 GCN이 탄생하게 된 배경

❖ 소감

- 푸리에 변환과 Convolution에 대해 제대로 공부할 수 있는 기회가 되었음
- 그래프 신호 처리 이론을 기반으로 하기때문에 사전지식이 많이 필요
- 때문에 아직 모든 수식과 수식 전개 과정을 이해하지 못해 추후에 공부한 뒤 추가할 예정

• Reference

❖ Spectral Networks and Deep Locally Connected Networks on Graphs

1. Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.
2. Chen, X. (2020). Understanding Spectral Graph Neural Network. *arXiv preprint arXiv:2012.06660*.
3. Fourier transform 설명 : https://darkpgmr.tistory.com/171
4. Fourier transform 증명 : https://ghebook.blogspot.com/2012/07/fourier-series.html
5. Laplacian 설명 : https://micropilot.tistory.com/2970
6. Graph Fourier Transform 설명 : https://ahjeong.tistory.com/14
7. Graph Fourier Transform 설명 : https://ralasun.github.io/deep%20learning/2021/02/15/gcn/
8. Graph Fourier Transform 설명 : https://greeksharifa.github.io/machine_learning/2021/08/14/GFT/
9. Graph Fourier Transform 설명 : https://harryjo97.github.io/theory/Graph-Fourier-Transform/
10. Graph Fourier Transform 설명 : https://sites.icmc.usp.br/gnonato/gs/slide3.pdf
11. Graph Fourier Transform 설명 : https://thejb.ai/comprehensive-gnns-3/

• Appendix A

❖ **proof of the positive semi-definite of Laplacian**

임의의 $x \in \mathbb{R}^N$ 에 대해,

$$
\begin{aligned}
x^T L x = x^T D x - x^T W x &= \sum_i D_{ii} x_i^2 - \sum_{i,j} x_i W_{ij} x_j \\
&= \frac{1}{2}\left(2\sum_i D_{ii} x_i^2 - 2\sum_{i,j} W_{ij} x_i x_j\right) \\
&\overset{\text{(a)}}{=} \frac{1}{2}\left(2\sum_i \left\{\sum_j W_{ij}\right\} x_i^2 - 2\sum_{i,j} W_{ij} x_i x_j\right) \\
&\overset{\text{(b)}}{=} \frac{1}{2}\left(\sum_{i,j} W_{ij} x_i^2 + \sum_{i,j} W_{ij} x_j^2 - 2\sum_{i,j} W_{ij} x_i x_j\right) \\
&= \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2 \geq 0 \qquad (2)
\end{aligned}
$$

https://harryjo97.github.io/theory/Graph-Laplacian/

DMQA

Thank You