Graph Convolutional Neural Network: An Overview

沈华伟
中国科学院计算技术研究所
shenhuawei@ict.ac.cn

Convolution

- Convolution is a mathematical operation on two functions, e.g., f and g, defined as

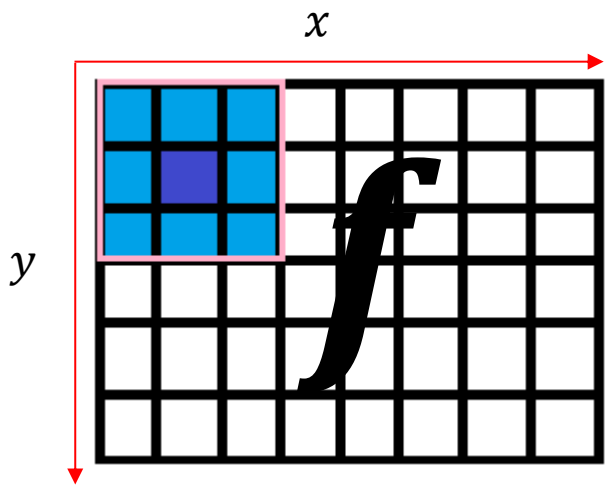
$$h(t) = (f * g)(t) \stackrel{\text{def}}{=} \int f(t)g(t-\tau)\,d\tau = \int f(t-\tau)g(\tau)\,d\tau$$

- Intuitively, convolution is a weighted average of the function $f(t-\tau)$ at the moment t where the weighting is given by $g(\tau)$
  - Also known as template matching, i.e., taking g as a template and using it to match f in a piece-wise manner

2-D Discrete Convolution

- 2-D discrete convolution

$$h(x,y) = (f * g)(x,y) \stackrel{\text{def}}{=} \sum_{m,n} f(x-m, y-n)\, g(m,n)$$



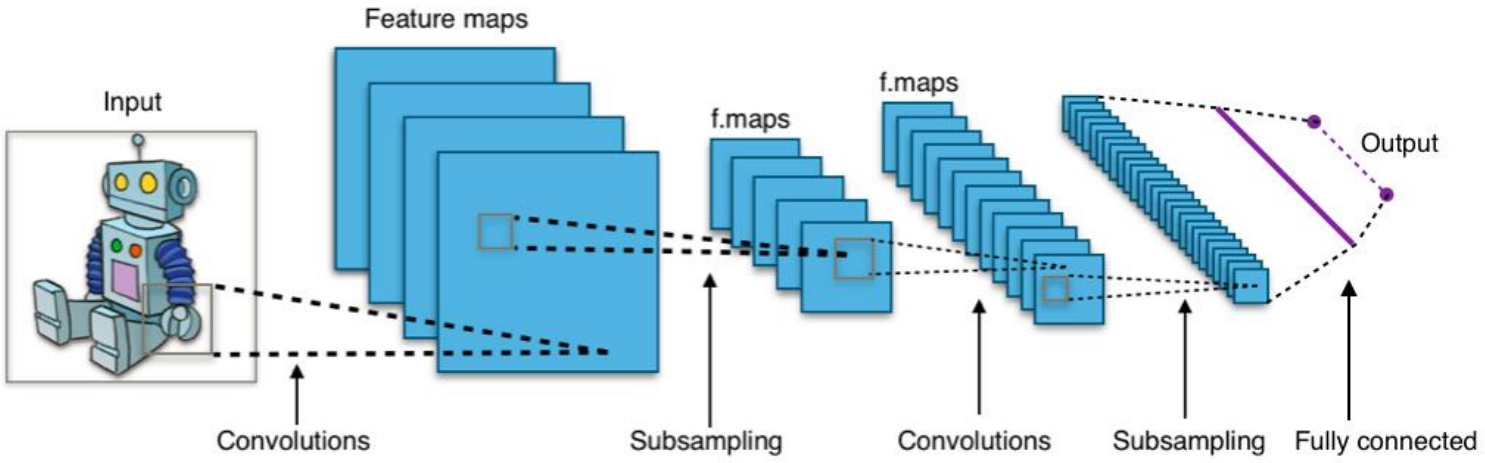
$g=$

$g(1,1)$	$g(0,1)$	$g(-1,1)$
$g(1,0)$	$g(0,0)$	$g(-1,0)$
$g(1,-1)$	$g(0,-1)$	$g(-1,-1)$

$$h(1,1) = f(0,0)g(1,1) + f(1,0)g(0,1) + f(2,0)g(-1,1)$$
$$+ f(0,1)g(1,0) + f(1,1)g(0,0) + f(2,1)g(-1,0)$$
$$+ f(0,2)g(1,-1) + f(1,2)g(0,-1) + f(2,2)g(-1,-1)$$

Convolutional Neural Network

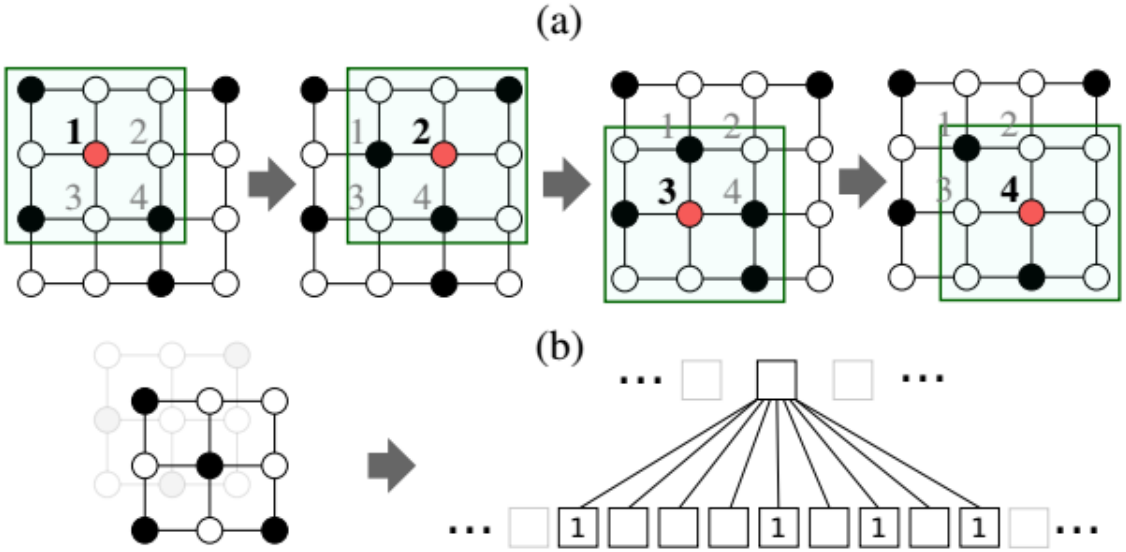
- CNN is a class of deep feed-forward artificial neural networks, most commonly applied to analyzing image, video, and audio data.
  - Weight sharing is its distinguishing feature, compared with fully-connected neural networks
  - Reduce the number of parameters to avoid overfitting when the model is deeper



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324, 1998.

Convolution in CNN

- Convolution operator in CNN



(a) A receptive field is moved over an image from left to right and top to bottom using a particular stride

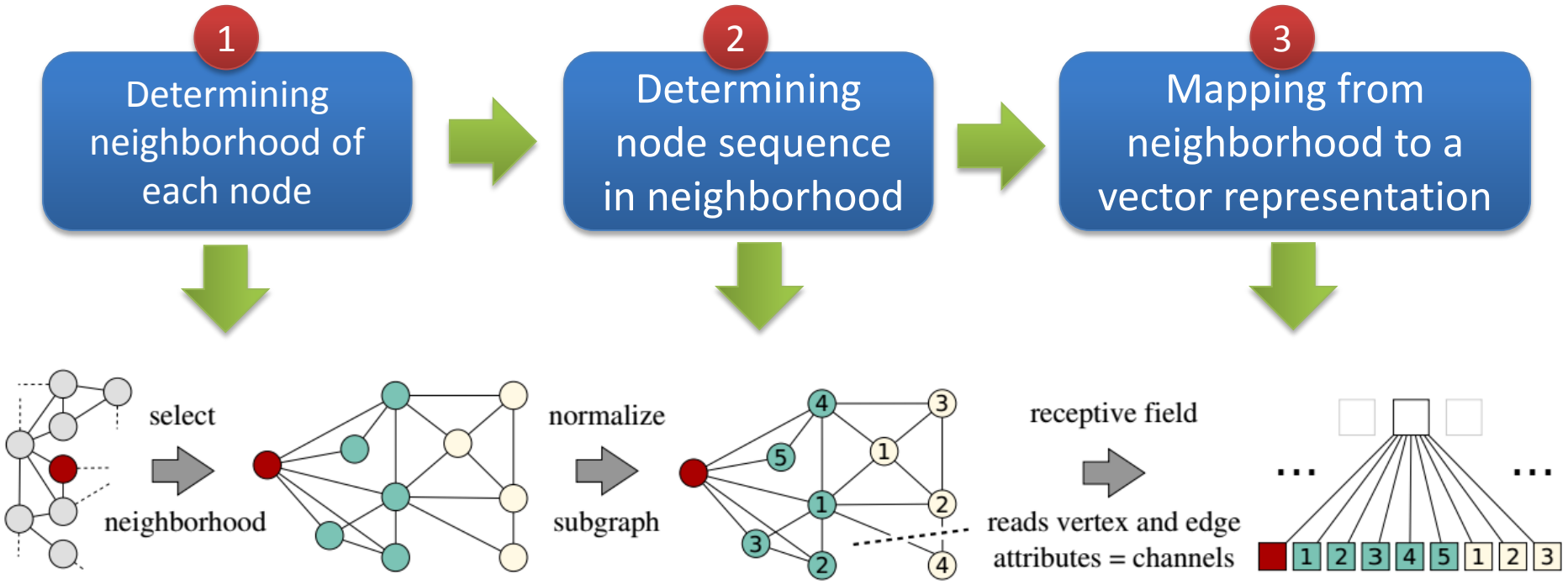
(b) The values read by the receptive field are transformed into a linear layer and fed to a convolutional architecture

1. Determining neighborhood of each node
2. Determining node sequence in neighborhood
3. Mapping from neighborhood to a vector representation

M. Niepert, M. Ahmed, K. Kutzkov. Learning Convolutional Neural Networks for Graphs. ICML, 2016.

Generalizing CNN to graph

- Convolutional neural network for graphs, by analogy



Cons:
1. High computational cost to determine neighborhood of each node.
2. Heuristic method

M. Niepert, M. Ahmed, K. Kutzkov. Learning Convolutional Neural Networks for Graphs. ICML, 2016.

Spectral methods

Bruna et al., ICLR, 2014;

Defferrard et al., NIPS 2016;

Kipf and Welling, ICLR 2017;

Fourier Transform

- Fourier transform
  - Time domain vs. frequency domain

$$\hat{f}(\xi) = \int f(t)\, e^{-2\pi i \xi t}\, dt$$

  - Fourier transform is the expansion of f in terms of the eigenfunctions of the Laplace operator, i.e., the second derivative

- Graph Fourier transform
  - Vertex domain vs. spectrum domain
  - Analogously, graph Fourier transform is defined by

$$\hat{f}(\lambda_i) = \langle f, u_i \rangle$$

  - λ_i and u_i are the i-th eigenvalue and eigenvector of Laplacian matrix

D. I. Shuman, B. Ricaud, and P. Vandergheynst, Vertex-frequency analysis on graphs, preprint, (2013)

Convolution on graph

- Convolution theorem

$$F\{f * g\} = F\{g\} \cdot F\{f\}$$

  - F denotes the Fourier transform of f

- According to convolution theorem, convolution operator on graph G is defined as

$$f *_G g = F^{-1}\{F\{g\} \cdot F\{f\}\} = U\big((U^T g) \odot (U^T f)\big)$$

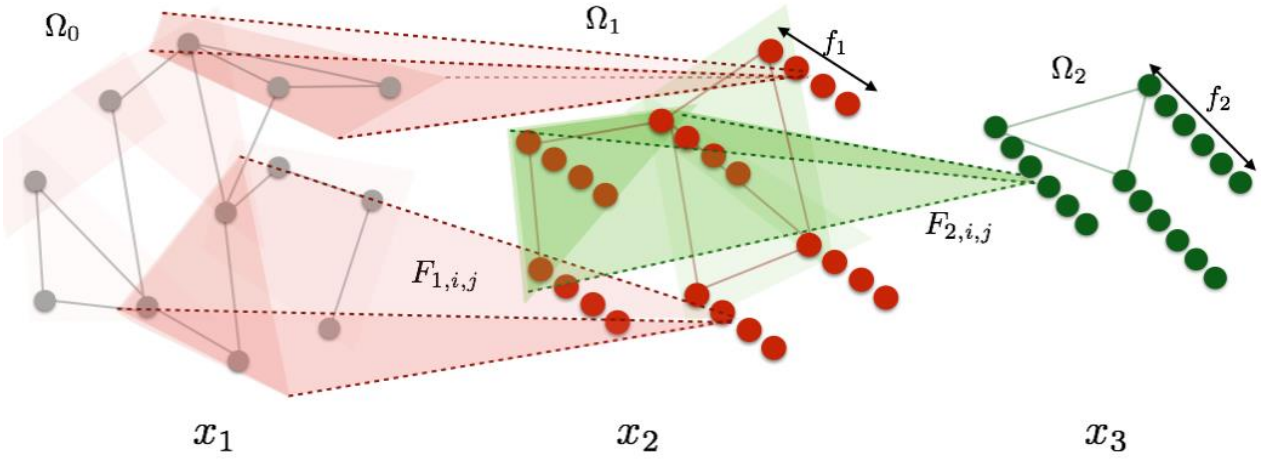
  - f and g are functions defined on nodes, i.e., vectors
  - U is the matrix of the eigenvector of the normalized Laplacian matrix $L = I - D^{-1/2} A D^{-1/2}$, forming the Fourier basis
  - I is identity matrix, A is adjacency matrix of graph, and D is the diagonal matrix of node degrees.

Convolution on graph: spectral method

- SCNN: Spectral method for graph CNN
  - The kth layer transforms an input vector x_k of size $|\Omega_{k-1}| \times f_{k-1}$ into an output of dimension $|\Omega_k| \times f_k$, as

$$x_{k+1,j} = h\left(\sum_{i=1}^{f_{k-1}} U F_{k,i,j} U^T x_{k,i}\right) \qquad i = 1 \cdots f_{k-1};\ j = 1 \cdots f_k$$

  - $F_{k,i,j}$ is a diagonal matrix, with its diagonal being convolutional kernel



J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv: 1312.6203, 2013.

Comments on spectral method

- Spectral method has solid principle for graph convolution
  - According to convolution theorem and graph Fourier transform

- Spectral methods explicitly rely on the spectrum of Laplacian matrix
  - Shortcoming 1: High computational cost to obtain the eigenvectors and to compute convolutions, generally $O(n^3)$ for a graph with n nodes
  - Shortcoming 2: Convolution is not localized, i.e., the receptive field of a target node is not located in its neighborhood

M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NIPS, 2016.

Spectrum-free spectral method

- ChebNet: Graph Convolutional Neural Network
  - Polynomial parameterization for localized filters

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

    - Λ is a diagonal matrix with its diagonal elements being eigenvalues of Laplacian matrix, and θ is the convolutional kernel
    - Reduce the number of free parameters from n to K
    - Spectral filters are K-localized, i.e., K hops from the central node
  - Recursive formulation for fast filtering

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

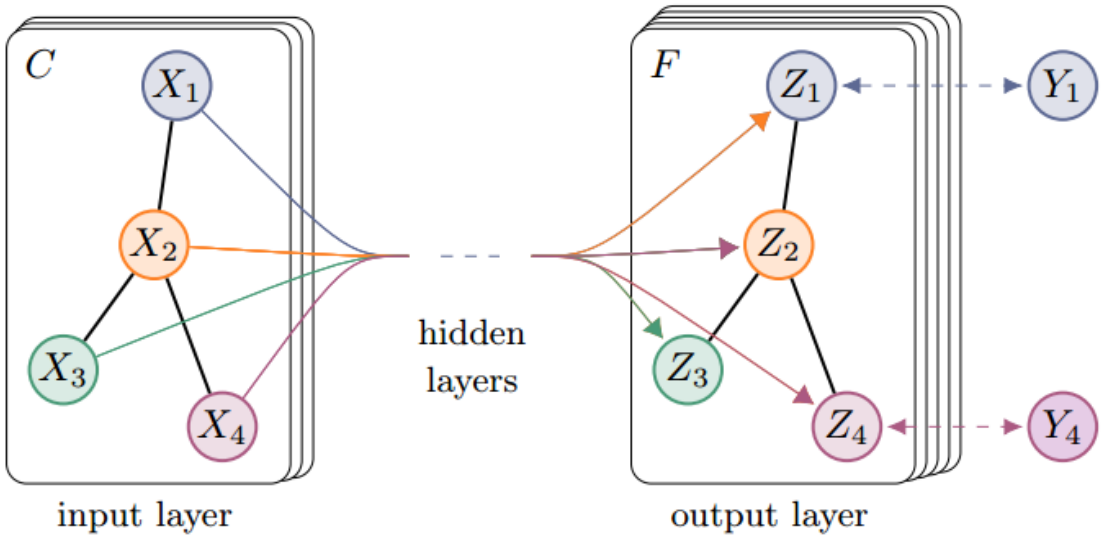
    - $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ is Chebyshev polynomial of order k
    - $\tilde{\Lambda} = 2\Lambda/\lambda_{\max} - I_n$ is a diagonal matrix of scaled eigenvalues that lie in [-1,1]
    - Reduce computational cost from $O(n^2)$ to $O(Km)$, where m is the number of edges

M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NIPS, 2016.

Simplified version of ChebNet

- GCN: Graph Convolution Network
  - Set $K = 2$ and view the center node as one of its neighbor, resulting in only one free parameter for each convolution filter
  - Offer an explanation of feature diffusion over graph

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \qquad H^{(0)} = X$$



T. N. Kipf and M. Welling. Semi-supervised classification with graph convolution networks, ICLR 2017.

Comments on GCN

- GCN offers us several key intuitions about graph CNN
  - Neighborhood is defined as nodes that are directly connected to the target nodes
  - Nodes in neighborhood are weighted by certain heuristics or domain knowledge
    - Similarity between nodes, e.g., $A = W_{ij}/\sqrt{d_i d_j}$
  - Open a new door for graph semi-supervised classification via feature diffusion (forward) and label propagation (backward)
  - Offer us a simple case to understand the spatial methods of graph CNN

Spatial methods

Masci et a., 3DRR, 2015;

Boscaini et al., NIPS 2016;

Monti et al., CVPR, 2017

Spatial methods of graph CNN

- Ideas behind spatial methods
  - Define multiple weighting function for points/nodes in neighborhood of target points/nodes
  - Convolution kernels are associated with weighting functions
    - Allowing target points/nodes with different neighbor size to share convolution kernels
- CNN is a special case of spatial method

(-1,-1)	(0,-1)	(1,-1)
(-1,0)	(0,0)	(1,0)
(-1,1)	(0,1)	(1,1)

- ✓ Define 9 weighting function over neighborhood, indexed by their coordinates, i.e., delta function
- ✓ For example, for a point (pixel), denoted by y, in the neighborhood, weighting function are

$$\delta(y-(-1,-1)),\ \delta(y-(0,-1)),\ \delta(y-(1,-1))$$

$$\delta(y-(-1,0)),\quad \delta(y-(0,0)),\quad \delta(y-(1,0))$$

$$\delta(y-(-1,1)),\quad \delta(y-(0,1)),\quad \delta(y-(1,1))$$

Spatial methods of graph CNN

- General framework of spatial method
  - Given a space, manifold or graph, we use x to denote the target point/node and denote with $y \in N(x)$ a point/node in the neighborhood $N(x)$ of x
  - For each y, we associate it with a pseudo-coordinate $u(x, y)$
  - Define multiple, i.e. J, weighting function $w_\Theta(u)$ with Θ being learnable parameters or without parameters (e.g., CNN)

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(\mathbf{u}(x,y))f(y), \quad j = 1, \ldots, J,$$

  - Convolution operator is defined with respect to weighting function, with clear template-matching explanation

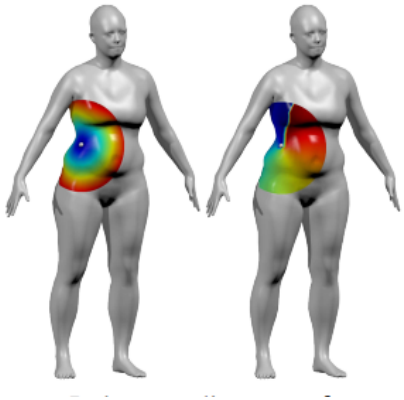
$$(f \star g)(x) = \sum_{j=1}^{J} g_j \, D_j(x) f$$

F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. CVPR 2017.

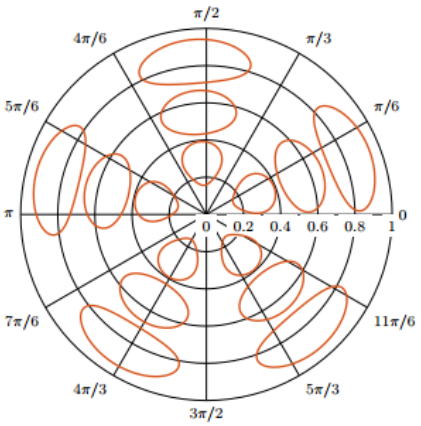
Spatial methods of graph CNN

- Examples of spatial methods
  - Application scenario: shape correspondence
  - Polar coordinates: $u(x,y) = \{\rho(x,y), \theta(x,y)\}$

- Three different weighting functions

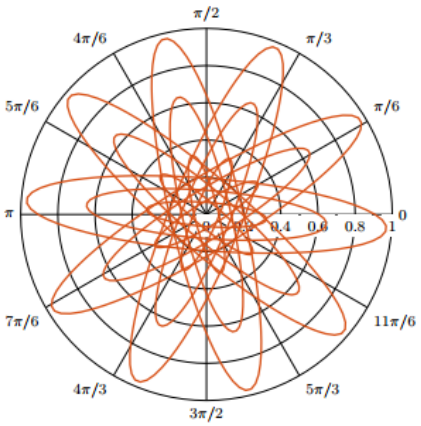


$\exp(-\frac{1}{2}(\mathbf{u}-\bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u}-\bar{\mathbf{u}}_j))$



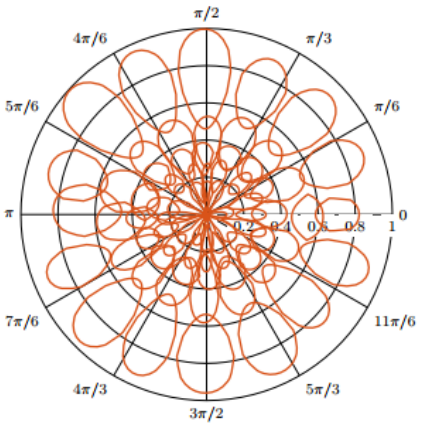
Geodesic CNN

$\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$



Anisotropic GNN

$\exp(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\mathbf{u}-\boldsymbol{\mu}_j))$



MoNet

Revisit CNN and GCN

- CNN
  - Pseudo-coordinate
    - $u(x, y)$=coordinate(y)-coordinate(x)
  - Weighting function
    - $\delta(u - u_j), j = 1 \cdots 9$

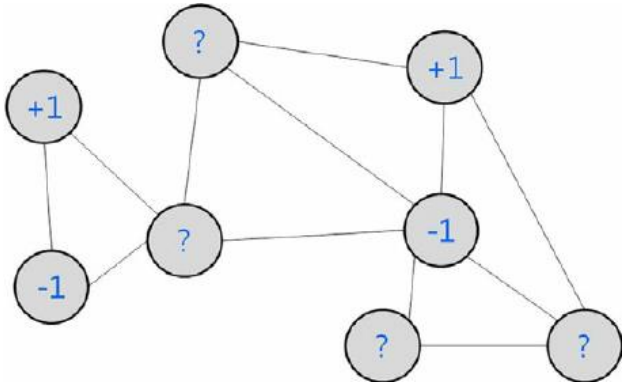
(-1,-1)	(0,-1)	(1,-1)
(-1,0)	(0,0)	(1,0)
(-1,1)	(0,1)	(1,1)

- GCN
  - Pseudo-coordinate
    - $d_x = \text{degree of } x,\ d_y = \text{degree of } y$
  - Weighing function
    - $W_{xy}/\sqrt{d_x d_y}$

Application:
Graph Semi-supervised Classification

Graph Semi-supervised Classification

- Given a graph
  - A few number of nodes are labeled, while other nodes are not
- Objective
  - Assign labels to unlabeled nodes
- Traditional methods
  - Label propagation
  - Network embedding



X. Zhu, Z. Ghahramai, J. D. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. ICML 2003.

B. Perozzi, R. Al-Rfou, S. Skiena. Deepwalk: online learning of social representations. KDD 2014.

Graph Semi-supervised Classification

- Loss function

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}$$

$$\mathcal{L}_{\text{reg}} = \sum_{ij} A_{ij} \left\| f(X_i) - f(X_j) \right\|^2 = f(X)^T \Delta f(X)$$

  - $\mathcal{L}_0$ is the supervised loss on labeled nodes
  - $\mathcal{L}_{\text{reg}}$ is the graph regularization, and Δ is Laplacian matrix
  - $f(\cdot)$ is the function we want to learn, e.g., a neural network
  - X is feature matrix
  - λ is hyper parameter

Z. Yang, W. W. Cohen, R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. ICML 2016

Graph Semi-supervised Classification

- Loss function with embedding regularization

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}$$

$$\mathcal{L}_{\text{reg}} = \sum_{ij} A_{ij} \left\| g(X_i) - g(X_j) \right\|^2$$

  - $\mathcal{L}_0$ is the supervised loss on labeled nodes
  - $\mathcal{L}_{\text{reg}}$ is the graph regularization
  - $g(\cdot)$ is an embedding function which learn representation of nodes according to its feature X
  - λ is hyper parameter

Z. Yang, W. W. Cohen, R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. ICML 2016

GCN for node classification

- Loss function

$$\mathcal{L} = f(X, A)$$

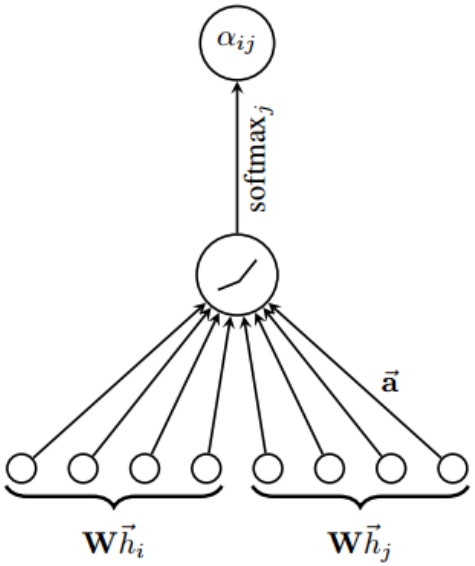
  - Take both feature matrix X and adjacency matrix A as input

  - Conditioning $f(\cdot)$ on the adjacency matrix A of the graph will allow the model to distribute gradient information from the supervised loss $\mathcal{L}_0$ and will enable it to learn representations of nodes both with and without labels

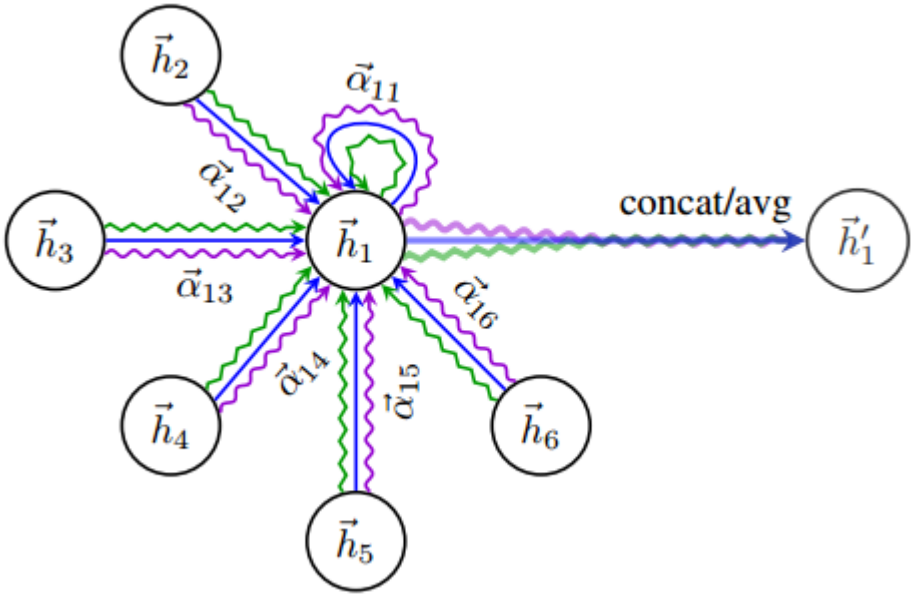
T. N. Kipf and M. Welling. Semi-supervised classification with graph convolution networks, ICLR 2017.

Recent advances

- GAT: Graph Attention Network
  - Leverage self-attention to learn the weighting function, i.e., the similarity of two nodes, according to the representation of nodes



Attention mechanism



Graph convolution using multi-head attention

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio. Graph attention networks. ICLR 2018

Comparison of methods

- Datasets
  - Cora: 2708 nodes, 5429 edges, 1433 features, 7 classes, 20 labels/class
  - Citeseer: 3327 nodes, 4732 edges, 3703 features, 6 classes, 20 labels/class
  - Pubmed: 19717 nodes, 44338 edges, 500 features, 3 classes, 20 labels/class
- Results

Feature only

Graph structure only

Methods	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
LP [Zhu et al., 2003]	68.0%	45.3%	63.0%
DeepWalk [Perozzi et al., 2014]	67.2%	43.2%	65.3%
ChebNet [Defferrard et al., 2016]	81.2%	69.8%	74.4%
GCN [Kipf & Welling, 2017]	81.5%	70.3%	79.0%
MoNet [Monti et al., 2017]	81.7±0.5%	N/A	78.8±0.3%
GAT [Velickovic et al., 2018]	83.0±0.7%	72.5±0.7%	79.0±0.3%

Graph CNN

Other applications

- Link prediction
  - M. Zitnik, M. Agrawal, J. Leskovec. Modeling polypharmacy side effects with graph convolutional networks, 2018.
  - W. L. Hamilton, R. Ying, J. Leskovec. Inductive representation learning on large graphs. NIPS 2017
- Recommendation
  - F. Monti, X. Bresson, M. M. Bronstein. Geometric matrix completion with recurrent multi-graph neural networks. 2017.
- Traffic prediction
  - Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, ICLR 2018

References

- M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. IEEE Signal Processing Magazine, 18-42, 2017.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324, 1998.
- D. I. Shuman, B. Ricaud, and P. Vandergheynst, Vertex-frequency analysis on graphs, preprint, (2013)
- M. Niepert, M. Ahmed, K. Kutzkov. Learning Convolutional Neural Networks for Graphs. ICML, 2016.
- J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv: 1312.6203, 2013.
- M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NIPS, 2016.
- T. N. Kipf and M. Welling. Semi-supervised classification with graph convolution networks, ICLR 2017.
- F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. CVPR 2017.
- W. L. Hamilton, R. Ying, J. Leskovec. Inductive representation learning on large graphs. NIPS 2017.
- P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio. Graph attention networks. ICLR 2018

谢谢！