Graph CNN

A/Prof Richard Yi Da Xu
Yida.Xu@uts.edu.au
Wechat: aubedata
https://github.com/roboticcam/machine-learning-notes

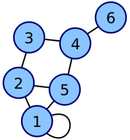
University of Technology Sydney (UTS)

April 12, 2019

degree matrix

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the associated degree matrix D is a $n \times n$ diagonal matrix:

$$d_{i,j} := \deg(v_i) \;\; \text{if } i = j, \qquad d_{i,j} := 0 \text{ otherwise}$$



$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- $\deg(v_i)$ counts the number of times an edge terminates at that vertex
- in undirected graph, each new loop increases degree of vertex by **two**, see node 1 above

matrix representation of a graph

- Given a simple graph $\mathcal{G}$ with n vertices, its Laplacian matrix $L^{n\times n}$:

$$L^{\mathcal{G}} = D - A$$

- D is the degree matrix and A is the adjacency matrix of the graph.

$$L^{\mathcal{G}}_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

- Since $\mathcal{G}$ is a simple graph, A only contains 1s or 0s and its diagonal elements are all 0s.
- Symmetric normalized Laplacian:

$$L^{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$$

About Graph Laplacian

- For a twice differentiable function f on the euclidean space, the Laplacian of f, Δf is **div(grad(f))** (divergence of the gradient of f)
- consider Laplacian in two dimensions:

$$\Delta f(x, y) = \nabla \cdot \nabla f(x, y) = \frac{d^2 f(x, y)}{dx^2} + \frac{d^2 f(x, y)}{dy^2}$$

- knowing that in 1-d, second derivative is approximated by:

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

- then we can see the approximation of

$$\begin{aligned}\Delta f(x, y) &= \lim_{t \to 0} \frac{f(x+h, y) + f(x-h, y) + f(x, y+h) + f(x, y-h) - 4f(x, y)}{h^2} \\ &\approx \frac{f(x+h, y) + f(x-h, y) + f(x, y+h) + f(x, y-h) - 4f(x, y)}{h^2}\end{aligned}$$

- this means that $\Delta f(x, y)$ is a **local gradient averaging operator**, is zero at a smooth image

$\mathbf{f}$ and $\nabla\mathbf{f}$

- Consider a function (or vector) $\mathbf{f}$ over vertices $\mathcal{V}$, $\mathbf{f} : \mathcal{V} \to \mathbb{R}$. This function is indexed by vertices, say v_i is the node value for i^{th} node, i.e., $\mathbf{f}$ is a vector, for example:

$$\mathbf{f} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

- then the product $\nabla\mathbf{f}$ is indexed by edges $\mathcal{E}$:
- each element of $\nabla\mathbf{f}$ is the difference between two end points of an edge e i.e. $v_i - v_j$:

$$\nabla\mathbf{f} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1 - v_2 \\ v_2 - v_3 \end{bmatrix}$$

- Matrix ∇ acts like a difference or gradient operator on $\mathcal{G}$

g and $\nabla^T \mathbf{g}^T$

- Consider a real function $\mathbf{g}$ over edges $\mathcal{E}$, $\mathbf{g} : \mathcal{E} \to \mathbb{R}$ indexed by edges.
- then, the product $\nabla^\top g^\top$ is a vector indexed by $\mathcal{V}$.
- Value at i^{th} vertex:

$$(\mathbf{g}\nabla)_i = \sum_{e \text{ exit node } i} g_e - \sum_{e \text{ enters node } i} g_e$$

- $(\mathbf{g}\nabla)_i$ is the net outbound flow on the vertex v which is **divergence**

- Consider the quantity $\nabla^T \nabla \mathbf{f}$
- this is divergence of the gradient.
- the matrix $L^{\mathcal{G}} = \nabla^T \nabla$ is the **Graph Laplacian**:

$$\nabla^\top \nabla = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

$$\implies \nabla^\top \nabla \mathbf{f} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 2 & -1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1 - v_2 \\ 2v_2 + v_1 - v_3 \\ v_2 - v_3 \end{bmatrix}$$

- for each i, j element of $L^{\mathcal{G}}$:

$$L^{\mathcal{G}}_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

- easy to see that $L^{\mathcal{G}} = D - A$:
- Laplacian is positive semidefinite:

$$\mathbf{f}^\top \nabla^\top \nabla \mathbf{f} = \|\nabla \mathbf{f}\|^2 = \sum_{(i,j)\in\mathcal{E}} (\mathbf{f}_i - \mathbf{f}_j)^2$$

Spectral Decomposition

- A Laplacian is written as:

$$L^{\mathcal{G}}\phi_k = \lambda_k \phi_k$$

- eigenvectors:

$$\langle \phi_k, \phi_{k'} \rangle = \delta_{k,k'}$$

- eigenvalues are non-negative:

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

- eigendecomposition of graph Laplacian:

$$L^{\mathcal{G}} = \Phi^{\top} \Lambda \Phi$$

where:

$$\Phi = [\phi_1, \ldots, \phi_n] \qquad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

Fourier Transform

- **traditional** Fourier series of a function:

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) \exp^{-ikt} \mathrm{d}t}_{\hat{f}_k = \langle \mathbf{f}, \exp^{-ikx} \rangle \text{ coefficient}} \underbrace{\exp^{-ikx}}_{\text{Fourier basis}}$$

- **graph** Fourier series: for the i^{th} element:

$$f_i = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle}_{\hat{f}_k \text{ coefficient}} \qquad \underbrace{\phi_{k,i}}_{\text{graph Fourier basis}}$$

in matrix notation:

$$\hat{\mathbf{f}} = \Phi^\top \mathbf{f} \qquad \text{and } \mathbf{f} = \Phi^\top \hat{\mathbf{f}}$$

Convolution in Euclidean space

- convolution theorem:

$$\widehat{\mathbf{f} \circledast \mathbf{g}} = \hat{\mathbf{f}} \odot \hat{\mathbf{g}}$$

- $\mathbf{f} = (f_1, \dots, f_n)^\top$ and $\mathbf{g} = (g_1, \dots, g_n)^\top$

$$(\mathbf{f} \circledast \mathbf{g})_i = \sum_k g(i-k) \bmod n \cdot f_m$$

$$\implies \mathbf{f} \circledast \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \dots & \dots & g_n \\ g_n & g_1 & g_2 & \dots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \dots & g_1 & g_2 \\ g_2 & g_3 & \dots & \dots & g_1 \end{bmatrix}}_{\text{Circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \Phi \underbrace{\begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix}}_{\Phi^\top \mathbf{g}} \Phi^\top \mathbf{f}$$

$$= \Phi(\Phi^\top \mathbf{g} \odot \Phi^\top \mathbf{f})$$

Convolution on graphs

- spectral convolution:

$$
\begin{aligned}
\mathbf{f} \circledast \mathbf{g} &= \Phi(\Phi^\top \mathbf{g} \odot \Phi^\top \mathbf{f}) \\
&= \underbrace{\Phi \text{diag}(\hat{g}_1, \ldots, \hat{g}_n)\Phi^\top}_{G} \mathbf{f}
\end{aligned}
$$

What is missing?

- CNN has a pooling step
- GraphCNN has Graph downsampling(**coarsening**) step