Notes on GNNs

Matthew Mo

2020 年 6 月 29 日

1 Vanilla GNN(Scarselli et al.)

Target node v ⇒representation $\boldsymbol{h}_v \in \mathbb{R}^s$ ⇒out $\boldsymbol{o}_v$
Domain undirected graph + node feature $\boldsymbol{x}_v$ + possible edge feature.
$co[v], ne[v]$:edges, neighbors of v.

1.1 Model

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}) \tag{1}$$

$$o_v = g(h_v, x_v) \tag{2}$$

where f:local transition function, g: local out function, both parametric. Let $\boldsymbol{H}, \boldsymbol{O}, \boldsymbol{X}, \boldsymbol{X_N}$ are those states in mat form

$$\boldsymbol{H} = F(\boldsymbol{H}, \boldsymbol{X}) \tag{3}$$

$$\boldsymbol{O} = F(\boldsymbol{H}, \boldsymbol{X}_N) \tag{4}$$

F:global transition function, G: global out function, both parametric, use fixed point iteration to calculate $\boldsymbol{H}$:

$$\boldsymbol{H}^{t+1} = F(\boldsymbol{H}^t, \boldsymbol{X}) \tag{5}$$

write loss using supervisory target info t_i

$$loss = \sum_{i=1}^{p}(t_i - o_i) \tag{6}$$

p is node count supervised.

Note An Implicit Neuron Representation!

Algorithm

1. update h'_v from (1) for T time steps
2. compute loss and gradients
3. update parameters

Note equals T layers of GNN of same parameters

1.2 Limitations

- computational inefficient
- each layer share same parameters⇒goto either RNN(GRU/LSTM)/ use different params
- edge features?
- large T ⇒graph representation be smooth, hard to distinguish nodes
- **Further Nets**
  - GGNN(computational efficiency)
  - R-GCN(directed graph)

2 Graph Convolutional Networks/GCN

2.1 Spectral Methods

2.1.1 Spectral CNN

$$\boldsymbol{g}_\theta * \boldsymbol{x} = \boldsymbol{U}\boldsymbol{g}_\theta(\boldsymbol{\Lambda})\boldsymbol{U}^T\boldsymbol{x} \tag{7}$$

$$\boldsymbol{\Delta} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \text{ normalized laplacian} \tag{8}$$

2.1.2 ChebNet a.k.a. GCNN

2.1.3 GCN