Representation Learning on Graphs and Networks (L45)
CST Part III / MPhil in ACS

Victor Zhao
xz398@cam.ac.uk

1 Primer on Graph Representations

1. Mathematical definition of graphs:

   A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a collection of nodes $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

   The edges can be represented by an *adjacency matrix*, $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, such that
$$A_{uv} = \begin{cases} 1 & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

2. Some interesting graph types:
   - **Undirected**: $\forall u, v \in \mathcal{V}.\ (u,v) \in \mathcal{E} \iff (u,v) \in \mathcal{E}$ (i.e., $\mathbf{A}^\top = \mathbf{A}$)
   - **Weighted**: provided *edge weight* w_{uv} for every edge $(u,v) \in \mathcal{E}$
   - **Multirelational**: various *edge types*, i.e. $(u,t,v) \in \mathcal{E}$ if there exists an edge (u,v) linked by type t
   - **Heterogeneous**: various *node types*
   - **Homophilic**: nodes tend to connect to other similar nodes (opposite: **heterophilic**)

3. Machine learning tasks on graphs by domain:
   - **Transductive**: training algorithm sees all observations, including the holdout observations
     - Task is to *propagate* labels from the training observations to the holdout observations
     - Also called *semi-supervised learning*
   - **Inductive**: training algorithm only sees the training observations during training, and only sees the holdout observations for prediction

4. Node statistics:
   - **Degree**: amount of edges the node is incident to:
$$d_u = \sum_{v \in \mathcal{V}} A_{uv}$$
   - **Centrality**: a measure of how "central" the node is in the graph (how often do infinite random walks visit the node)
$$d_u = \lambda^{-1} \sum_{v \in \mathcal{V}} A_{uv} e_v$$
     where $\mathbf{e} \in \mathbb{R}^{|\mathcal{V}|}$ is the largest eigenvector of $\mathbf{A}$, with corresponding eigenvalue λ
   - **Clustering coefficient**: a measure of "clusteredness" (are neighbours connected amongst each other)
$$c_u = \frac{\left|\left\{(v_1, v_2) \in \mathcal{E} \middle| v_1, v_2 \in \mathcal{N}(u)\right\}\right|}{\binom{d_u}{2}}$$

5. Graph Laplacian:

   Let $\mathbf{D}$ be the diagonal (out)-degree matrix of the graph, i.e., $D_{uu} = \sum_{v \in \mathcal{V}} A_{ij}$. Then:
   - The *unnormalised* graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$
   - The *symmetric* graph Laplacian: $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$
   - The *random walk* graph Laplacian: $\mathbf{L}_{\text{RW}} = \mathbf{D}^{-1} \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}$

   Properties:
   - For undirected graphs, $\mathbf{L}$ is *symmetric* ($\mathbf{L}^\top = \mathbf{L}$) and *positive semi-definite* ($\forall \mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}.\ \mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0$)
   - For undirected graphs:
$$\forall \mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}.\ \mathbf{x}^\top \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{u \in \mathcal{V}} \sum_{v \in \mathcal{V}} A_{uv} (x_u - x_v)^2 = \sum_{(u,v) \in \mathcal{E}} (x_u - x_v)^2$$
   - $\mathbf{L}$ has $|\mathcal{V}|$ nonnegative eigenvalues: $\lambda_1 \geq \cdots \geq \lambda_{|V|} = 0$

6. Spectral clustering:
    - Two-way cut: partition the graph into $\mathcal{A} \subseteq \mathcal{V}$ and its complement $\mathcal{A}_c \subseteq \mathcal{V}$:

    $$\text{Cut}(\mathcal{A}) = \big|\big\{(u, v) \in \mathcal{E} \big| u \in \mathcal{A} \wedge v \in \mathcal{A}_c\big\}\big|$$

    Ratio cut metric:

    $$\text{RCut}(\mathcal{A}) = \text{Cut}(\mathcal{A}) \left(\frac{1}{|\mathcal{A}|} + \frac{1}{|\mathcal{A}_c|}\right)$$

    - Minimising $\text{RCut}(\mathcal{A})$:
    Let $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}|}$ be a vector representing the cut $\mathcal{A}$, defined as follows:

    $$a_u = \begin{cases} \sqrt{\frac{\mathcal{A}_c}{\mathcal{A}}} & \text{if } u \in \mathcal{A} \\ -\sqrt{\frac{\mathcal{A}}{\mathcal{A}_c}} & \text{if } u \in \mathcal{A}_c \end{cases}$$

    Then

    $$\mathbf{a}^\top \mathbf{L} \mathbf{a} = \sum_{(u,v) \in \mathcal{E}} (a_u - a_v)^2 = |\mathcal{V}| \text{RCut}(\mathcal{A})$$

    Minimising $\mathbf{a}^\top \mathbf{L} \mathbf{a}$ corresponds to minimising $\text{RCut}(\mathcal{A})$ (NP-hard as the constraint is discrete)
    - Relaxing: minimise $\mathbf{a}^\top \mathbf{L} \mathbf{a}$ subject to $\mathbf{a} \perp \mathbf{1}$ and $||\mathbf{a}||^2 = |\mathcal{V}|$
    Rayleigh–Ritz Theorem: The solution is exactly the second-smallest eigenvector of $\mathbf{L}$
    To obtain the cut, place u into $\mathcal{A}$ or $\mathcal{A}_c$ depending on the sign of a_u
    - Can be generalised to k-clustering

2 Permutation Invariance and Equivariance

1. Informal definitions:
    - **Permutation invariance**: applying a permutation matrix does not modify the result
    - **Permutation equivariance**: transformation preserves the node order
    - **Locality**: signal remains stable under slight deformations of the domain
2. Setup:
    - **Node feature matrix**: $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_{|\mathcal{V}|} \end{bmatrix}^\top \in \mathbb{R}^{|\mathcal{V}| \times k}$, where $\mathbf{x}_i \in \mathbb{R}^k$ is the features of node i
    - **(1-hop) neighbourhood** of node i: $\mathcal{N}_i = \big\{j \big| (i, j) \in \mathcal{E} \vee (j, i) \in \mathcal{E}\big\}$
    - **Neighbourhood features**: $\mathbf{X}_{\mathcal{N}_i} = \{\{\mathbf{x}_j | j \in \mathcal{N}_i\}\}$
    - **Permutation matrix**: a $|\mathcal{V}| \times |\mathcal{V}|$ binary matrix that has exactly one entry of 1 in every row and column, and 0s elsewhere: $\mathbf{P} = \begin{bmatrix} \mathbf{e}_{\pi(1)} & \cdots & \mathbf{e}_{\pi(|\mathcal{V}|)} \end{bmatrix}^\top$
3. Learning on sets:
    - $f(\mathbf{X})$ is *permutation invariant* if for all permutation matrices $\mathbf{P}$: $f(\mathbf{P}\mathbf{X}) = f(\mathbf{X})$
    - $\boldsymbol{F}(\mathbf{X})$ is *permutataion equivariant* if for all permutation matrices $\mathbf{P}$: $\boldsymbol{F}(\mathbf{P}\mathbf{X}) = \mathbf{P}\boldsymbol{F}(\mathbf{X})$
    - Locality on sets: transform every node in isolation, through a shared function ψ: $\mathbf{h}_i = \psi(\mathbf{x}_i)$
    Stacking $\mathbf{h}_i$ into a matrix yields $\mathbf{H} = \boldsymbol{F}(\mathbf{X})$:

    $$\boldsymbol{F}(\mathbf{X}) = \begin{bmatrix} — & \psi(\mathbf{x}_1) & — \\ & \vdots & \\ — & \psi(\mathbf{x}_{|\mathcal{V}|}) & — \end{bmatrix}$$

    - Deep Sets (Zaheer *et al.*, NIPS 2017):

    $$f(\mathbf{X}) = \phi\left(\bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i)\right)$$

    Universality of Deep Sets: any permutation invariant model can be expressed as a Deep Sets

4. Learning on graphs:
   - $f(\mathbf{X})$ is *permutation invariant* if for all permutation matrices $\mathbf{P}$: $f(\mathbf{PX}, \mathbf{PAP}^\top) = f(\mathbf{X}, \mathbf{A})$
   - $\boldsymbol{F}(\mathbf{X})$ is *permutataion equivariant* if for all permutation matrices $\mathbf{P}$: $\boldsymbol{F}(\mathbf{PX}, \mathbf{PAP}^\top) = \mathbf{P}\boldsymbol{F}(\mathbf{X}, \mathbf{A})$
   - Locality on graphs: apply a local function ϕ over all neighbourhoods:

$$\boldsymbol{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} — & \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & — \\ & \vdots & \\ — & \phi(\mathbf{x}_{|\mathcal{V}|}, \mathbf{X}_{\mathcal{N}_{|\mathcal{V}|}}) & — \end{bmatrix}$$

   To ensure permutation equivariance, it is sufficient that ϕ is permutation invariant in $\mathbf{X}_{\mathcal{N}_i}$

3 Graph Neural Networks

1. Graph Networks (Battaglia *et al.*, 2018):

   Data flow:
   - Update edge features (using relevant nodes + graph)

$$\mathbf{h}_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_{uv}, \mathbf{x}_{\mathcal{G}})$$

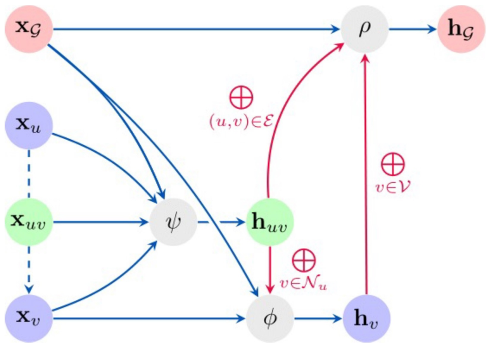
   - Update node features (using updated relevant edges + graph)

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_v} \mathbf{h}_{uv}, \mathbf{x}_{\mathcal{G}}\right)$$

   - Update graph features (using updated nodes + edges)

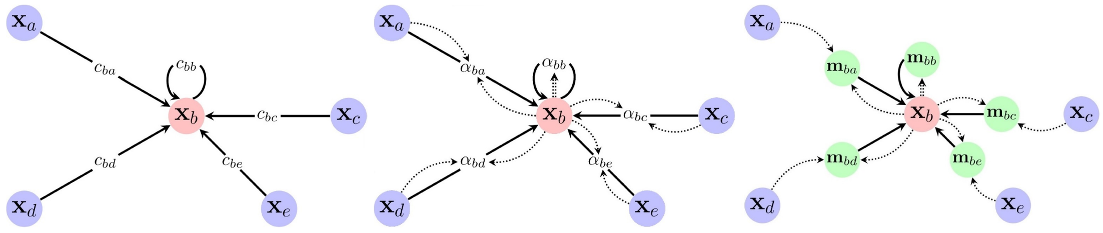
$$\mathbf{h}_{\mathcal{G}} = \rho\left(\bigoplus_{u \in \mathcal{V}} \mathbf{h}_u, \bigoplus_{(u,v) \in \mathcal{E}} \mathbf{h}_{uv}, \mathbf{x}_{\mathcal{G}}\right)$$

   Visualisation (**equivariant** and **invariant** layers):



$\mathbf{x}_{\mathcal{G}}$
ρ
$\mathbf{h}_{\mathcal{G}}$
$\bigoplus_{(u,v)\in\mathcal{E}}$
$\mathbf{x}_u$
$\bigoplus_{v\in\mathcal{V}}$
$\mathbf{x}_{uv}$
ψ
$\mathbf{h}_{uv}$
$\bigoplus_{v\in\mathcal{N}_u}$
$\mathbf{x}_v$
ϕ
$\mathbf{h}_v$

2. Three flavours of GNN layers:



(a) Convolutional

(b) Attentional

(c) Message-passing

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right) \quad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \quad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

3. Convolutional GNNs: features of neighbourhood aggregated with fixed weights
   - Graph Convolutional Network (GCN; Kipf & Welling, ICLR 2017):
     $$\mathbf{H} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}\right)$$
     where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{D}}$ is the corresponding degree matrix of $\tilde{\mathbf{A}}$
   - Simplified Graph Convolution (SGC; Wu *et al.*, ICML 2019):
     $$\mathbf{H} = \text{Softmax}\left(\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)^K \mathbf{X}\mathbf{W}\right)$$
     Near state-of-the-art on many tasks of interest, and very efficient to train
   - Chebyshev Networks (ChebyNet; Defferrard *et al.*, NIPS 2016):
     $$\mathbf{H} = \sigma\left(\sum_{k=0}^{K} \alpha_k \left(\frac{2}{\lambda_{\text{max}}}\mathbf{L}_{\text{sym}} - \mathbf{I}\right)^k \mathbf{X}\mathbf{W}_k\right)$$
     where
     - λ_{max} is the largest eigenvalue of $\mathbf{L}_{\text{sym}}$
     - α_k is the order-k coefficient of its Chebyshev polynomial

     GCN can be represented by a ChebyNet with $K = 1$ and $\lambda_{\text{max}} \approx 2$
4. Attentional GNNs: features of neighbourhood aggregated with implicit weights (attention)
   - Mixture Model CNN (MoNet; Monti *et al.*, CVPR 2017):
     $$\mathbf{h}_u = \sigma\left(\sum_{u \in \mathcal{N}(v)} w\big(\mathbf{e}(u, v)\big)\mathbf{W}\mathbf{x}_v\right)$$
     where
     - $\mathbf{e} : \mathcal{V}^2 \to \mathbb{R}^k$ is a pseudo-coordinate function that extracts a vector-based representation of the (u, v) edge
     - $w : \mathbb{R}^k \to \mathbb{R}$ is a weighting function converting the vector $\mathbf{e}$ into an aggregation coefficient

     MoNet can represent all isotropic GNNs and many standard anisotropic GNNs: for example, GCN can be represented by a MoNet with
     $$\mathbf{e} = \begin{bmatrix}\text{deg}(u)\\ \text{deg}(v)\end{bmatrix} \qquad w(\mathbf{e}) = \left(1 - \left|1 - \tfrac{1}{\sqrt{e_1}}\right|\right)\left(1 - \left|1 - \tfrac{1}{\sqrt{e_2}}\right|\right)$$
   - Graph Attention Networks (GAT; Veličković *et al.*, ICLR 2018):
     $$\mathbf{h}_u = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha(\mathbf{x}_u, \mathbf{x}_v)\mathbf{W}\mathbf{x}_v\right)$$
     In practice, to prevent over-fitting on the (now deprecated) datasets at that time, linear (static) attention mechanism was used:
     $$e(\mathbf{x}_u, \mathbf{x}_v) = \text{LeakyReLU}(\mathbf{a}^\top[\mathbf{x}_u \| \mathbf{x}_v])$$
     $$\alpha(\mathbf{x}_u, \mathbf{x}_v) = \text{Softmax}\big(e(\mathbf{x}_u, \mathbf{x}_v)\big) = \frac{\exp\big(e(\mathbf{x}_u, \mathbf{x}_v)\big)}{\sum_{w \in \mathcal{N}(u)} \exp\big(e(\mathbf{x}_u, \mathbf{x}_w)\big)}$$
   - GATv2 (Brody *et al.*, ICLR 2022): uses a universal approximator (a 2-layer MLP) that can learn any attention function, including dynamic attention
     $$e(\mathbf{x}_u, \mathbf{x}_v) = \mathbf{a}^\top\text{LeakyReLU}(\mathbf{W}[\mathbf{x}_u \| \mathbf{x}_v])$$
     GATv2 requires explicitly materialising the concatenation $[\mathbf{x}_u \| \mathbf{x}_v]$, and therefore has $O(|\mathcal{V}| + |\mathcal{E}|)$ storage complexity (in a graph, $|\mathcal{E}|$ is often significantly larger than $|\mathcal{V}|$)
   - Multi-head attention (Vaswani *et al.*, NIPS 2017):
     $$\mathbf{h}_u = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_1(\mathbf{x}_u, \mathbf{x}_v)\mathbf{W}_1\mathbf{x}_v\right) \Bigg\| \cdots \Bigg\| \, \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_K(\mathbf{x}_u, \mathbf{x}_v)\mathbf{W}_K\mathbf{x}_v\right)$$

Appendix: Mathematical Notations

a	A scalar (integer or real)
$\mathbf{a}$	A vector
$\mathbf{A}$	A matrix
$\mathcal{A}$ or $\{\cdot\}$	A set
$\{\{\cdot\}\}$	A multiset
$\lvert\mathcal{A}\rvert$	Cardinality of set $\mathcal{A}$
$\mathbb{R}$	The set of real numbers
a_i	Element i of vector $\mathbf{a}$, with indexing starting at 1
A_{ij}	Element i, j of matrix $\mathbf{A}$, with indexing starting at 1
f	A function
$\boldsymbol{F}$	A matrix-valued function
π	A permutation
$\phi, \psi, \rho, \cdots$	Learnable functions (e.g., MLPs)
σ	A non-linear activation function (e.g., sigmoid, ReLU)
$\oplus$	A permutation-invariant operator (e.g., sum, mean, min, max)