Machine Learning (part II)

Graph Neural Networks

Angelo Ciaramella

Introduction

- Structured data
  - in the form of sequences and images, corresponding to one-dimensional and two-dimensional arrays of variables respectively
  - many types of structured data are best described by a graph
    - set of nodes connected by edges

Structured data

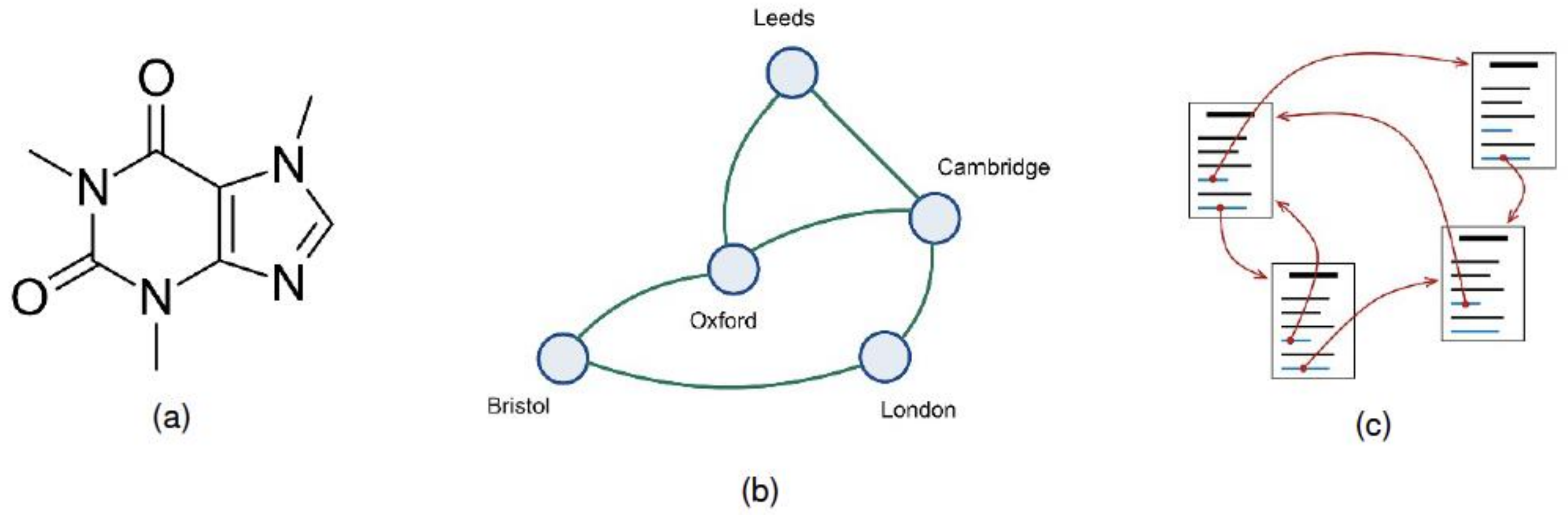


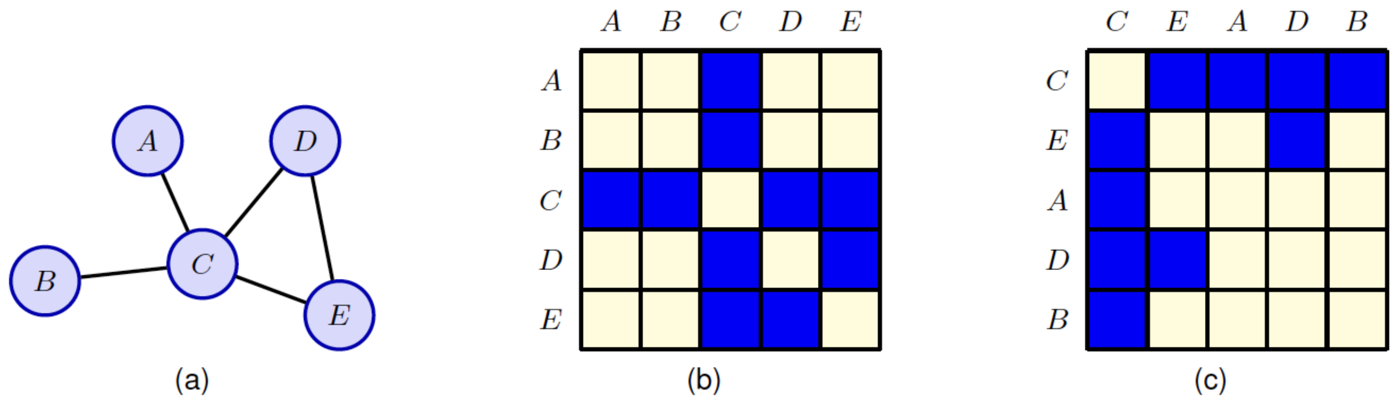
Figure 13.1 Three examples of graph-structured data: (a) the caffeine molecule consisting of atoms connected by chemical bonds, (b) a rail network consisting of cities connected by railway lines, and (c) the worldwide web consisting of pages connected by hyperlinks.

Graphs

$$\mathcal{G} = (\overset{\text{nodes}}{\mathcal{V}}, \overset{\text{edges}}{\mathcal{E}})$$

permutation

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$



$$\widetilde{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}}$$

Figure 13.2 An example of an adjacency matrix showing (a) an example of a graph with five nodes, (b) the associated adjacency matrix for a particular choice of node order, and (c) the adjacency matrix corresponding to a different choice for the node order.

Permutation equivariance

- Invariance
  - The network predictions must be invariant to node label reordering

$$y(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = y(\mathbf{X}, \mathbf{A})$$ Invariance

- Equivariance
  - node predictions should be equivariant with respect to node label reordering

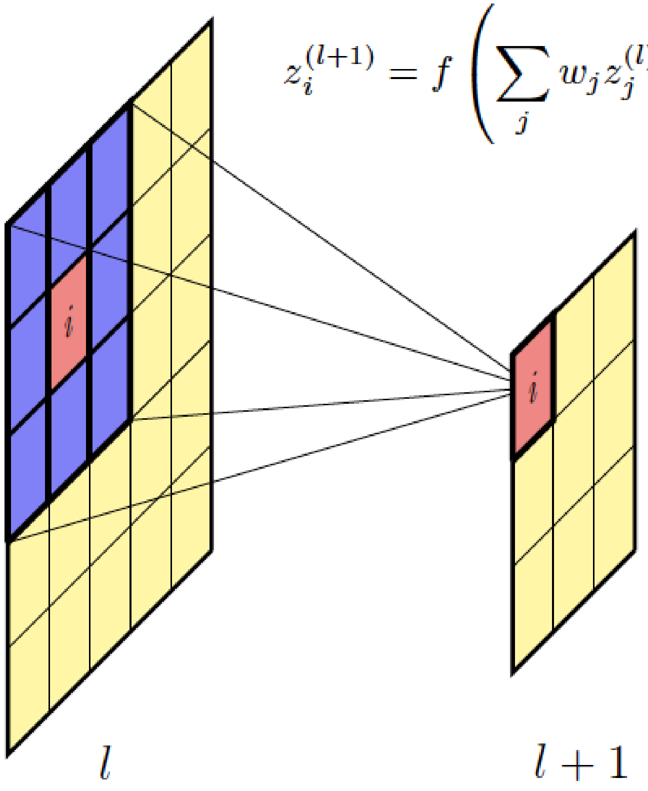
$$\mathbf{y}(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = \mathbf{P}\mathbf{y}(\mathbf{X}, \mathbf{A})$$ Equivariance

Convolution filter

$$z_i^{(l+1)} = f\left(\sum_j w_j z_j^{(l)} + b\right)$$

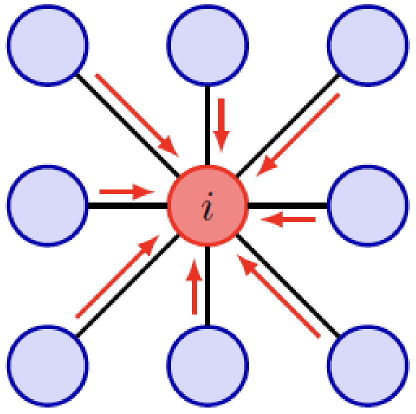
Invariant to any permutation

$$z_i^{(l+1)} = f\left(w_{\text{neigh}} \sum_{j \in \mathcal{N}(i)} z_j^{(l)} + w_{\text{self}} z_i^{(l)} + b\right)$$



(a)

aggregation



(b)

Figure 13.3 A convolutional filter for images can be represented as a graph-structured computation. (a) A filter computed by node i in layer $l+1$ of a deep convolutional network is a function of the activation values in layer l over a local patch of pixels. (b) The same computation structure expressed as a graph showing 'messages' flowing into node i from its neighbours.

Message-passing NN

Algorithm 13.1: Simple message-passing neural network

```
Input: Undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
       Initial node embeddings $\{\mathbf{h}_n^{(0)} = \mathbf{x}_n\}$
       Aggregate$(\cdot)$ function
       Update$(\cdot,\cdot)$ function
Output: Final node embeddings $\{\mathbf{h}_n^{(L)}\}$
--------------------------------------------------------------
// Iterative message-passing
for $l \in \{0, \ldots, L-1\}$ do
    $\mathbf{z}_n^{(l)} \leftarrow \mathrm{Aggregate}\left(\left\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\right\}\right)$
    $\mathbf{h}_n^{(l+1)} \leftarrow \mathrm{Update}\left(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}\right)$
end for
return $\{\mathbf{h}_n^{(L)}\}$
```

$\mathbf{h}_n^{(l)}$ D-dimensional column vector of node-embedding variables

Aggregation operators

sum

$$\text{Aggregate}\left(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}\right) = \sum_{m \in \mathcal{N}(n)} \mathbf{h}_m^{(l)}$$

average

$$\text{Aggregate}\left(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}\right) = \frac{1}{|\mathcal{N}(n)|} \sum_{m \in \mathcal{N}(n)} \mathbf{h}_m^{(l)}$$

element-wise

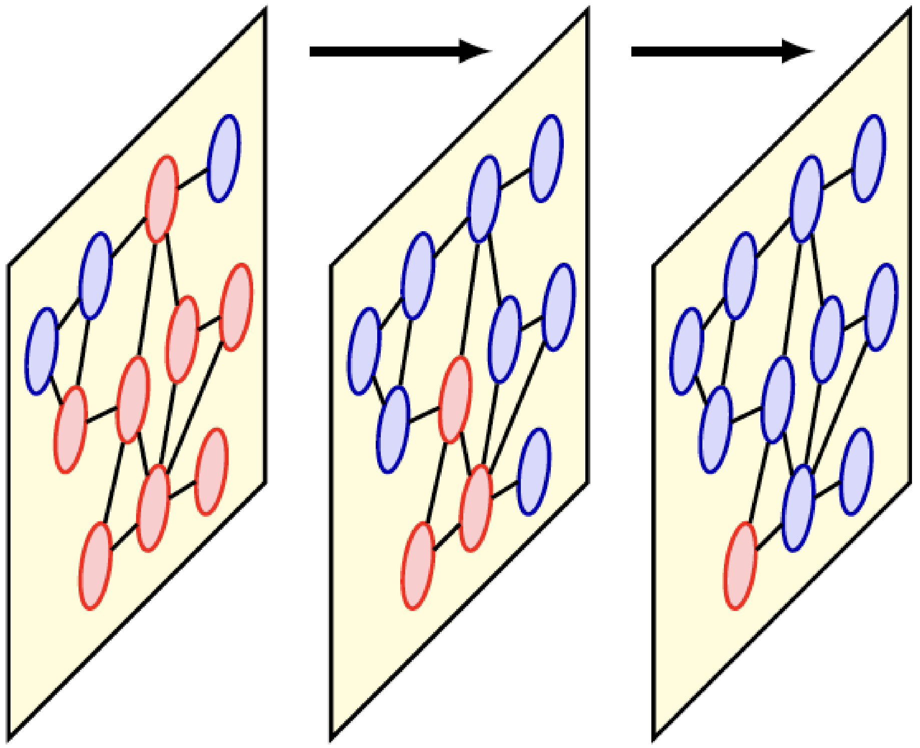
$$\text{Aggregate}\left(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}\right) = \sum_{m \in \mathcal{N}(n)} \frac{\mathbf{h}_m^{(l)}}{\sqrt{|\mathcal{N}(n)|\,|\mathcal{N}(m)|}}$$

learnable parameters

$$\text{Aggregate}\left(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}\right) = \text{MLP}_{\boldsymbol{\theta}}\left(\sum_{m \in \mathcal{N}(n)} \text{MLP}_{\phi}(\mathbf{h}_m^{(l)})\right)$$

Information flow

Figure 13.4 Schematic illustration of information flow through successive layers of a graph neural network. In the third layer a single node is highlighted in red. It receives information from its two neighbours in the previous layer and those in turn receive information from their neighbours in the first layer. As with convolutional neural networks for images, we see that the effective receptive field, corresponding to the number of nodes shown in red, grows with the number of processing layers.



Update operators

$$\text{Update}\left(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}\right) = f\left(\mathbf{W}_{\text{self}}\mathbf{h}_n^{(l)} + \mathbf{W}_{\text{neigh}}\mathbf{z}_n^{(l)} + \mathbf{b}\right)$$

$$\Big\Downarrow \quad \mathbf{W}_{\text{self}} = \mathbf{W}_{\text{neigh}}$$

$$\mathbf{h}_n^{(l+1)} = \text{Update}\left(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}\right) = f\left(\mathbf{W}_{\text{neigh}} \sum_{m \in \mathcal{N}(n), n} \mathbf{h}_m^{(l)} + \mathbf{b}\right)$$

$$\mathbf{h}_n^{(0)} = \mathbf{x}_n$$

Node classification

- GNN
  - can be viewed as a series of layers each of which transforms a set of node-embedding vectors $\{h_n^{(l)}\}$ into a new set $\{h_n^{(l+1)}\}$ of the same size and dimensionality

- Classifying nodes in a graph
  - readout layer

$$y_{ni} = \frac{\exp(\mathbf{w}_i^{\mathrm{T}} \mathbf{h}_n^{(L)})}{\sum_j \exp(\mathbf{w}_j^{\mathrm{T}} \mathbf{h}_n^{(L)})}$$

  - loss function

$$\mathcal{L} = - \sum_{n \in \mathcal{V}_{\mathrm{train}}} \sum_{i=1}^{C} y_{ni}^{t_{ni}}$$

Edge classification

- edge completion
  - edge present between two nodes

$$p(n,m) = \sigma\left(\mathbf{h}_n^{\mathrm{T}}\mathbf{h}_m\right)$$

- Example
  - predicting whether two people in a social network have shared interests and therefore might wish to connect

Graph classification

- sum of the node-embedding vectors
  - edge present between two nodes

$$\mathbf{y} = \mathbf{f}\left(\sum_{n\in\mathcal{V}} \mathbf{h}_n^{(L)}\right)$$

Graph attention network

- Aggregation

$$\mathbf{z}_n^{(l)} = \mathrm{Aggregate}\left(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}\right) = \sum_{m \in \mathcal{N}(n)} A_{nm} \mathbf{h}_m^{(l)}$$

$$A_{nm} \geqslant 0$$

$$\sum_{m \in \mathcal{N}(n)} A_{nm} = 1.$$

bilinear

$$A_{nm} = \frac{\exp\left(\mathbf{h}_n^{\mathrm{T}} \mathbf{W} \mathbf{h}_m\right)}{\sum_{m' \in \mathcal{N}(n)} \exp\left(\mathbf{h}_n^{\mathrm{T}} \mathbf{W} \mathbf{h}_{m'}\right)}$$

general

$$A_{nm} = \frac{\exp\left\{\mathrm{MLP}\left(\mathbf{h}_n, \mathbf{h}_m\right)\right\}}{\sum_{m' \in \mathcal{N}(n)} \exp\left\{\mathrm{MLP}\left(\mathbf{h}_n, \mathbf{h}_{m'}\right)\right\}}$$

Embeddings

- Edges

$$\mathbf{e}_{nm}^{(l+1)} = \mathrm{Update}_{\mathrm{edge}}\left(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_{n}^{(l)}, \mathbf{h}_{m}^{(l)}\right)$$
$$\mathbf{z}_{n}^{(l+1)} = \mathrm{Aggregate}_{\mathrm{node}}\left(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\}\right)$$
$$\mathbf{h}_{n}^{(l+1)} = \mathrm{Update}_{\mathrm{node}}\left(\mathbf{h}_{n}^{(l)}, \mathbf{z}_{n}^{(l+1)}\right).$$

- Graph

$$\mathbf{e}_{nm}^{(l+1)} = \mathrm{Update}_{\mathrm{edge}}\left(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_{n}^{(l)}, \mathbf{h}_{m}^{(l)}, \mathbf{g}^{(l)}\right)$$
$$\mathbf{z}_{n}^{(l+1)} = \mathrm{Aggregate}_{\mathrm{node}}\left(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\}\right)$$
$$\mathbf{h}_{n}^{(l+1)} = \mathrm{Update}_{\mathrm{node}}\left(\mathbf{h}_{n}^{(l)}, \mathbf{z}_{n}^{(l+1)}, \mathbf{g}^{(l)}\right)$$
$$\mathbf{g}^{(l+1)} = \mathrm{Update}_{\mathrm{graph}}\left(\mathbf{g}^{(l)}, \{\mathbf{h}_{n}^{(l+1)} : n \in \mathcal{V}\}, \{\mathbf{e}_{nm}^{(l+1)} : (n,m) \in \mathcal{E}\}\right)$$

Over-smoothing

- Modifying the operator

$$\mathbf{h}_n^{(l+1)} = \mathrm{Update}_{\mathrm{node}}\left(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}, \mathbf{g}^{(l)}\right) + \mathbf{h}_n^{(l)}$$

- Taking information from all previous layers

$$\mathbf{y}_n = \mathbf{f}\left(\mathbf{h}_n^{(1)} \oplus \mathbf{h}_n^{(2)} \oplus \cdots \oplus \mathbf{h}_n^{(L)}\right)$$