Graph NN

Abdul Samad

Adapted from Simon J.D. Prince

Graph Examples

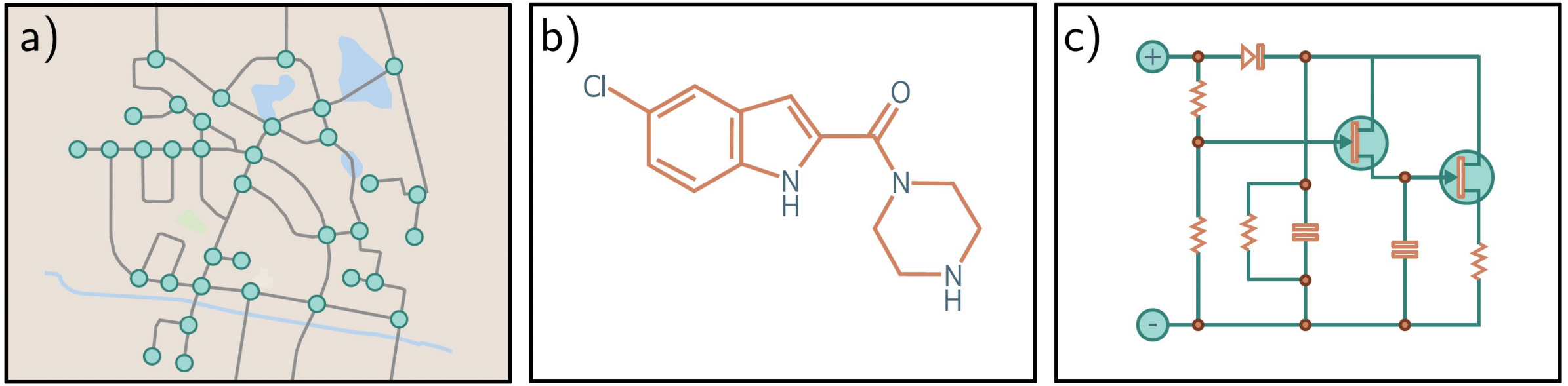


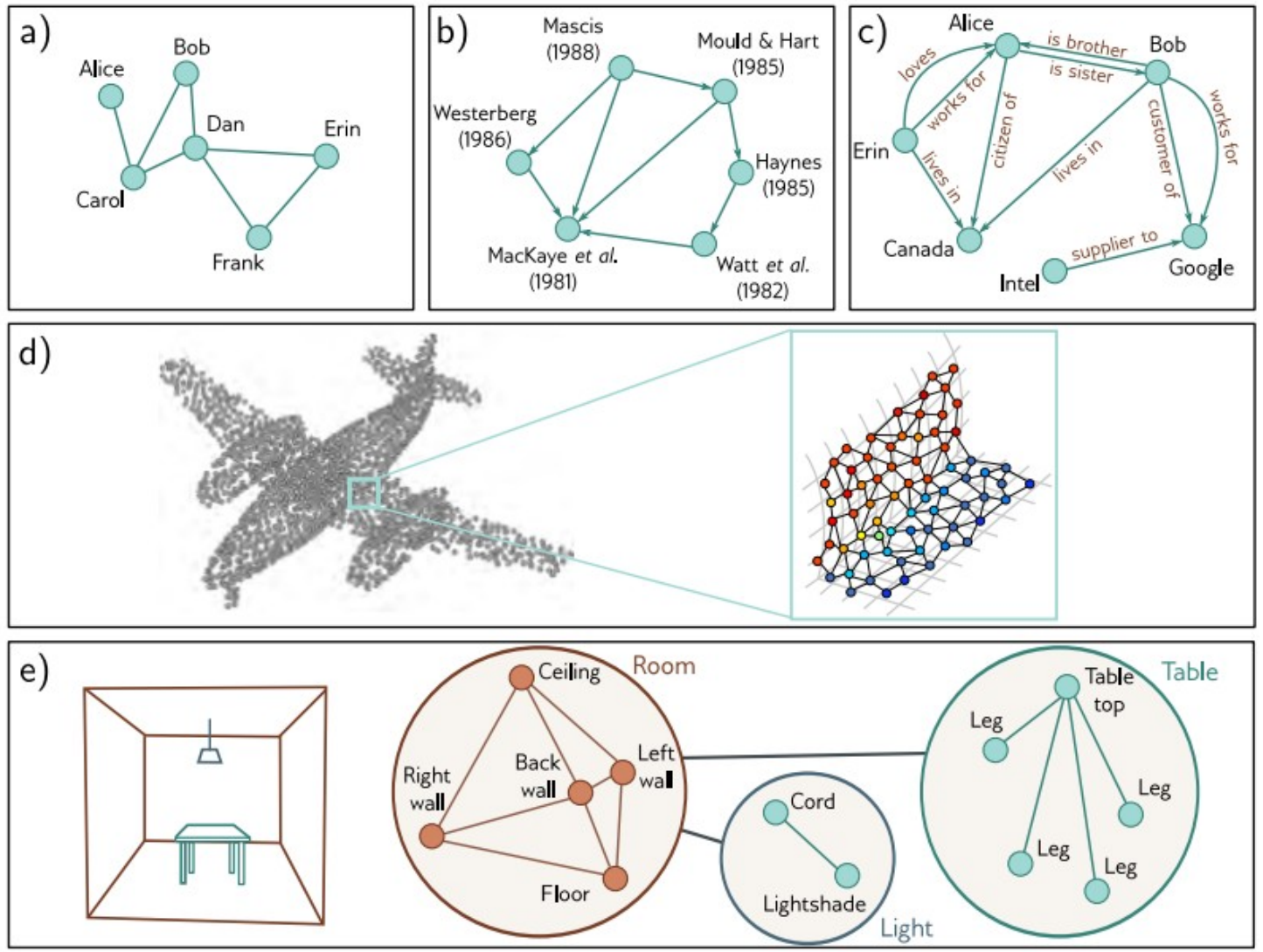
Figure 13.1 Real-world graphs. Some objects, such as a) road networks, b) molecules, and c) electrical circuits, are naturally structured as graphs.

More examples of Graphs

Some examples of graphs

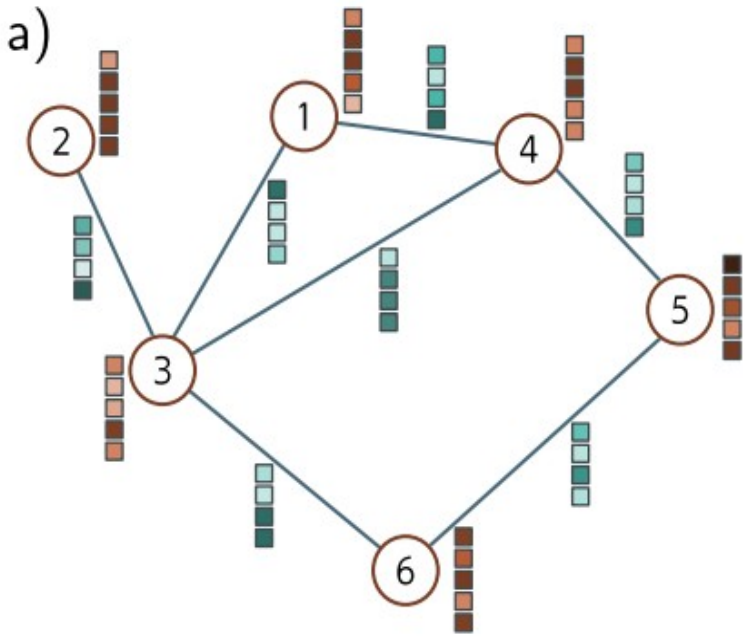
- Social networks are graphs where nodes are people, and the edges represent friendships between them.
- The scientific literature can be viewed as a graph where the nodes are papers, and the edges represent citations.
- Wikipedia can be considered a graph where the nodes are articles, and the edges represent hyperlinks between articles.
- Protein interactions in a cell can be expressed as graphs, where the nodes are the proteins, and there is an edge between two proteins if they interact.

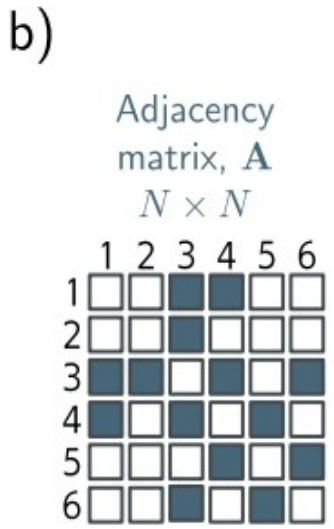
Types of Graphs

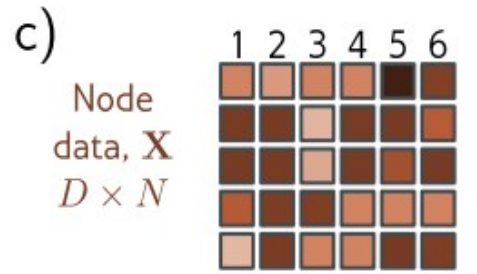


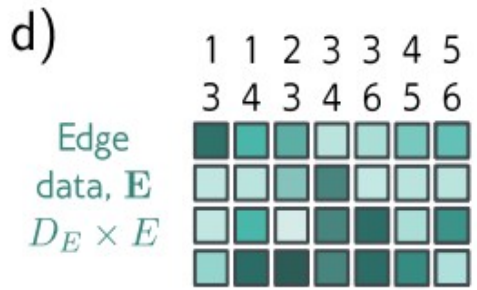
Graph Representation

The graph can be encoded by three matrices , , and , representing the **graph structure**, **node embeddings**, and **edge embeddings**, respectively



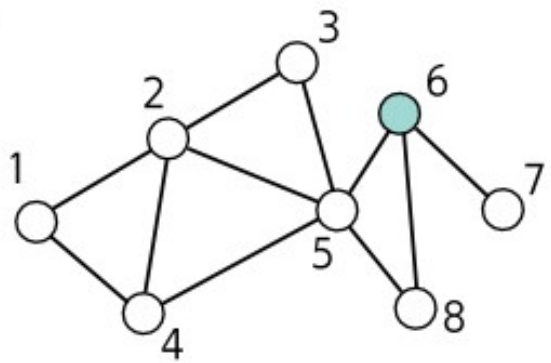






Properties of adjacency matrix

a)



b)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

c)

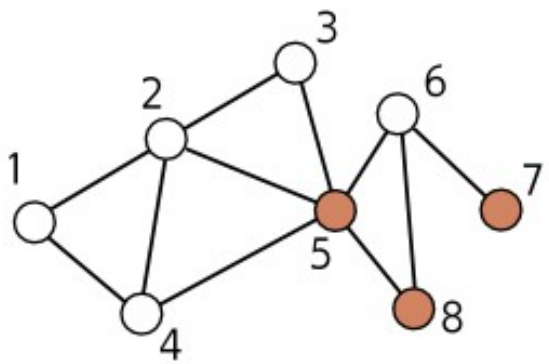
$$\mathbf{A}^2 = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 4 & 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 3 & 1 & 1 & 0 & 1 \\ 2 & 2 & 1 & 1 & 5 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

d)

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

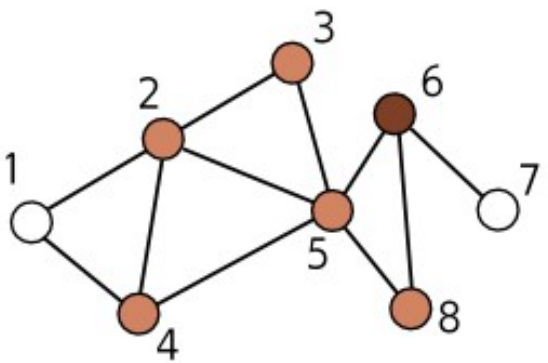
e)

$$\mathbf{Ax} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

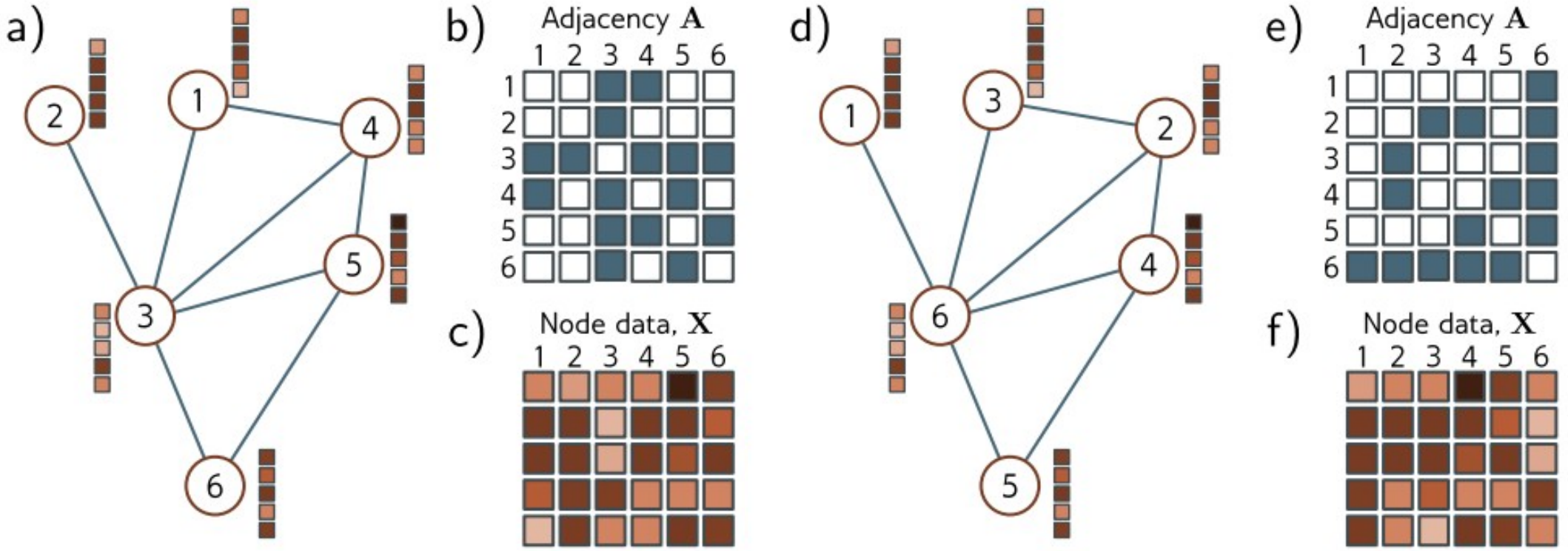


f)

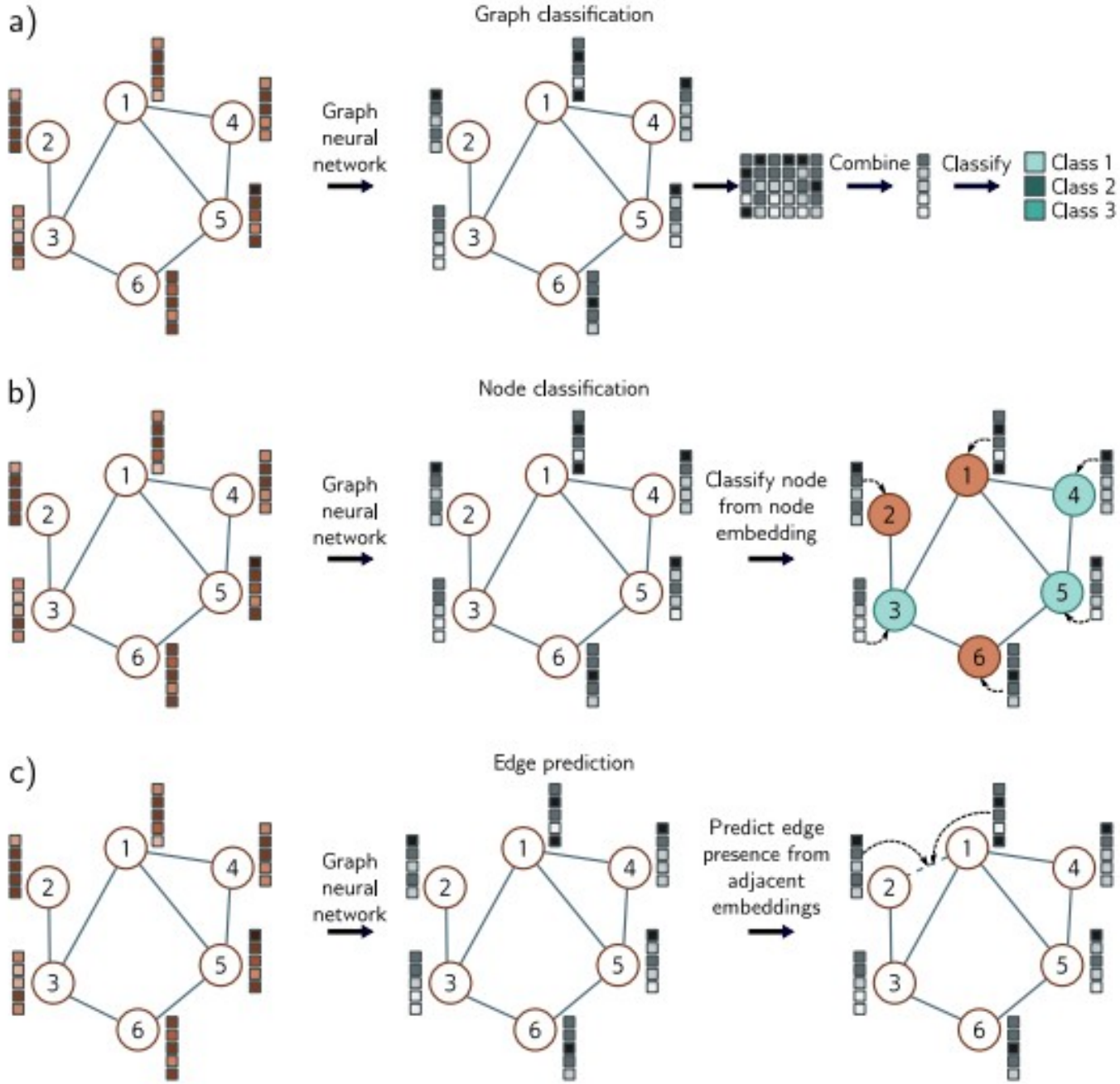
$$\mathbf{A}^2\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$



Permutation of node indices



Tasks

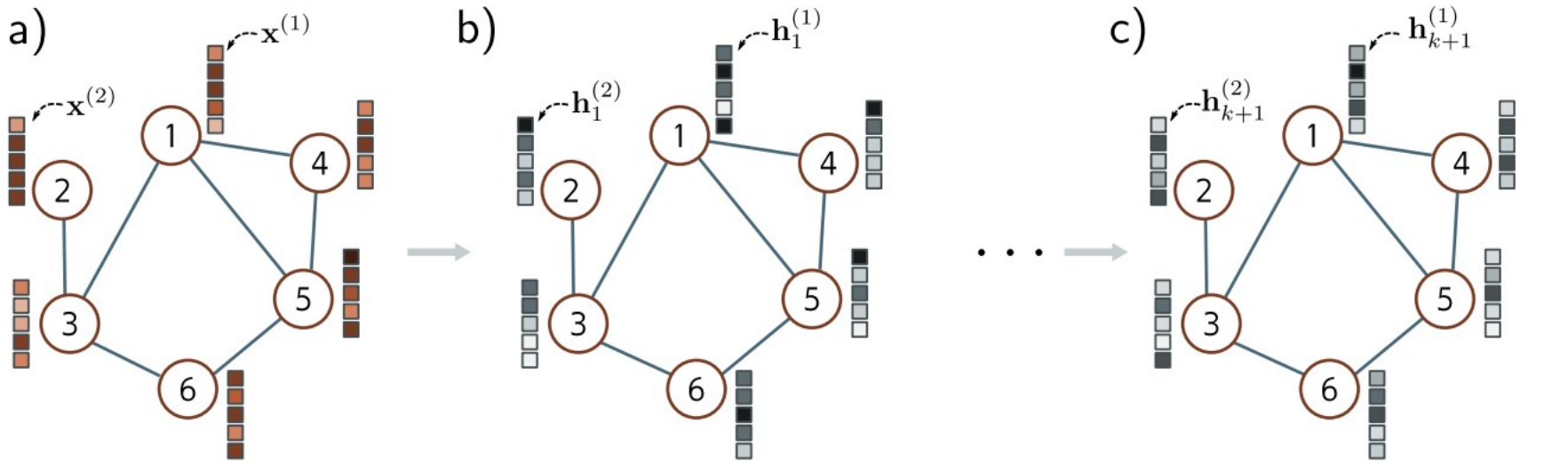

a)
Graph classification
Graph neural network
Combine
Classify
Class 1
Class 2
Class 3
b)
Node classification
Graph neural network
Classify node from node embedding
c)
Edge prediction
Graph neural network
Predict edge presence from adjacent embeddings

Graph Convolution NN

$$
\begin{aligned}
\mathbf{H}_1 &= \mathbf{F}[\mathbf{X}, \mathbf{A}, \boldsymbol{\phi}_0] \\
\mathbf{H}_2 &= \mathbf{F}[\mathbf{H}_1, \mathbf{A}, \boldsymbol{\phi}_1] \\
\mathbf{H}_3 &= \mathbf{F}[\mathbf{H}_2, \mathbf{A}, \boldsymbol{\phi}_2] \\
\vdots &= \vdots \\
\mathbf{H}_K &= \mathbf{F}[\mathbf{H}_{K-1}, \mathbf{A}, \boldsymbol{\phi}_{K-1}],
\end{aligned} \tag{13.5}
$$

where $\mathbf{X}$ is the input, $\mathbf{A}$ is the adjacency matrix, $\mathbf{H}_k$ contains the modified node embeddings at the k^{th} layer, and $\boldsymbol{\phi}_k$ denotes the parameters that map from layer k to layer $k+1$.

Example GCN



$$\mathbf{agg}[n,k] = \sum_{m \in \mathrm{ne}[n]} \mathbf{h}_k^{(m)}, \quad \text{where } \mathrm{ne}[n] \text{ returns the set of indices of the neighbors of node } n.$$

Example GCN

embeddings into the $D \times N$ matrix $\mathbf{H}_k$ and post-multiply by the adjacency matrix $\mathbf{A}$, the n^{th} column of the result is $\mathbf{agg}[n, k]$. The update for the nodes is now:

$$\begin{aligned} \mathbf{H}_{k+1} &= \mathbf{a}\left[\boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{H}_k + \boldsymbol{\Omega}_k \mathbf{H}_k \mathbf{A}\right] \\ &= \mathbf{a}\left[\boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{H}_k (\mathbf{A} + \mathbf{I})\right], \end{aligned} \tag{13.10}$$

where $\mathbf{1}$ is an $N \times 1$ vector containing ones. Here, the nonlinear activation function $\mathbf{a}[\bullet]$ is applied independently to every member of its matrix argument.

Graph classification

- Example of a network that classifies molecules as toxic or harmless.
- Network inputs are (Adjacency Matrix and node Embeddings)

  where is derived from molecule structure, are

  one-hot vectors indicating which of the 118 elements are present.

$$
\begin{aligned}
\mathbf{H}_1 &= \mathbf{a}\left[\boldsymbol{\beta}_0\mathbf{1}^T + \boldsymbol{\Omega}_0\mathbf{X}(\mathbf{A}+\mathbf{I})\right] \\
\mathbf{H}_2 &= \mathbf{a}\left[\boldsymbol{\beta}_1\mathbf{1}^T + \boldsymbol{\Omega}_1\mathbf{H}_1(\mathbf{A}+\mathbf{I})\right] \\
\vdots &= \vdots \\
\mathbf{H}_K &= \mathbf{a}\left[\boldsymbol{\beta}_{K-1}\mathbf{1}^T + \boldsymbol{\Omega}_{K-1}\mathbf{H}_{k-1}(\mathbf{A}+\mathbf{I})\right] \\
\mathrm{f}[\mathbf{X},\mathbf{A},\boldsymbol{\Phi}] &= \mathrm{sig}\left[\beta_K + \boldsymbol{\omega}_K\mathbf{H}_K\mathbf{1}/N\right],
\end{aligned}
$$

where and is a single value.

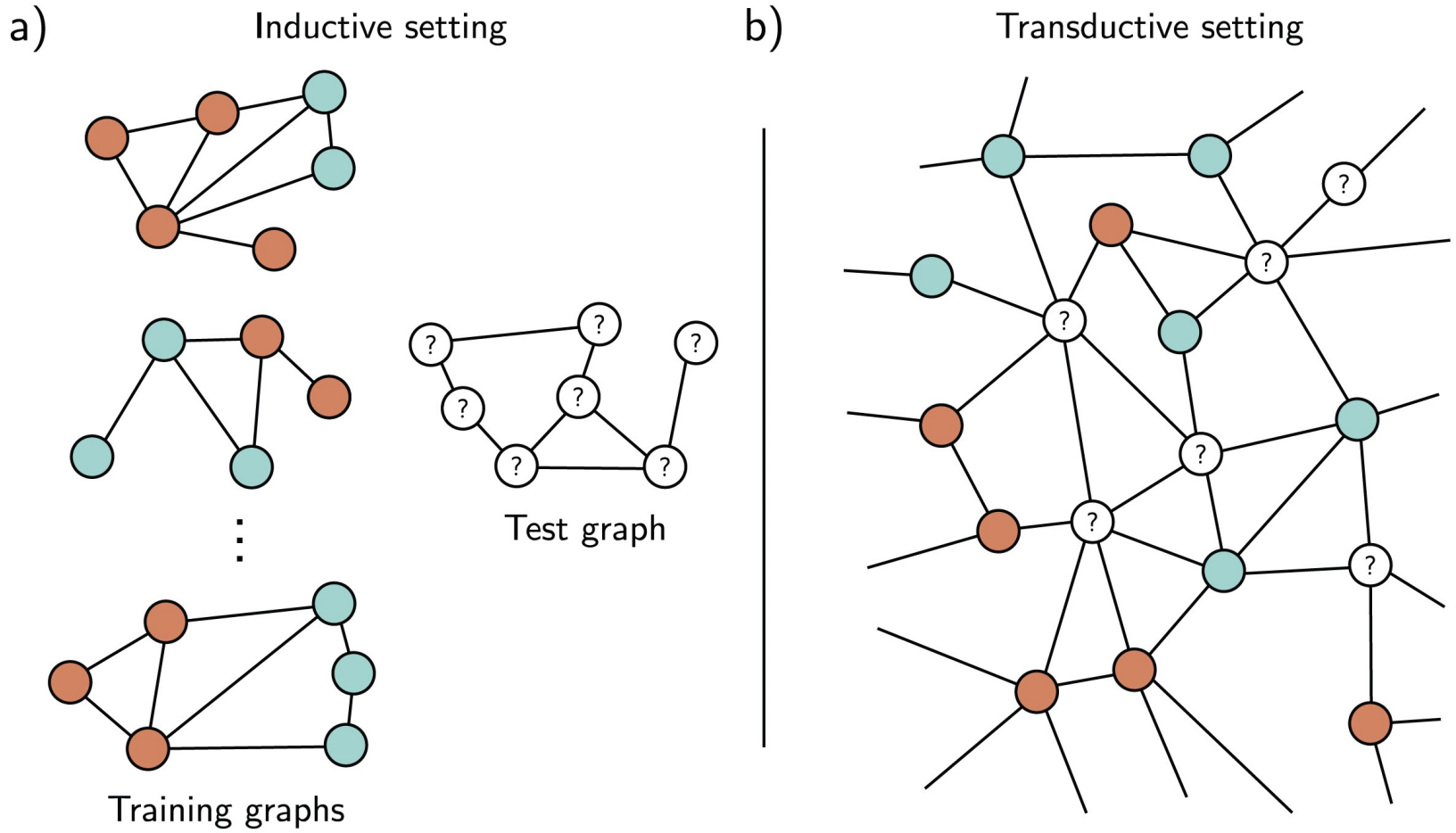


Figure 13.8 Inductive vs. transductive problems. a) Node classification task in the inductive setting. We are given a set of I training graphs, where the node labels (orange and cyan colors) are known. After training, we are given a test graph and must assign labels to each node. b) Node classification in the transductive setting. There is one large graph in which some nodes have labels (orange and cyan colors), and others are unknown. We train the model to predict the known labels correctly and then examine the predictions at the unknown nodes.

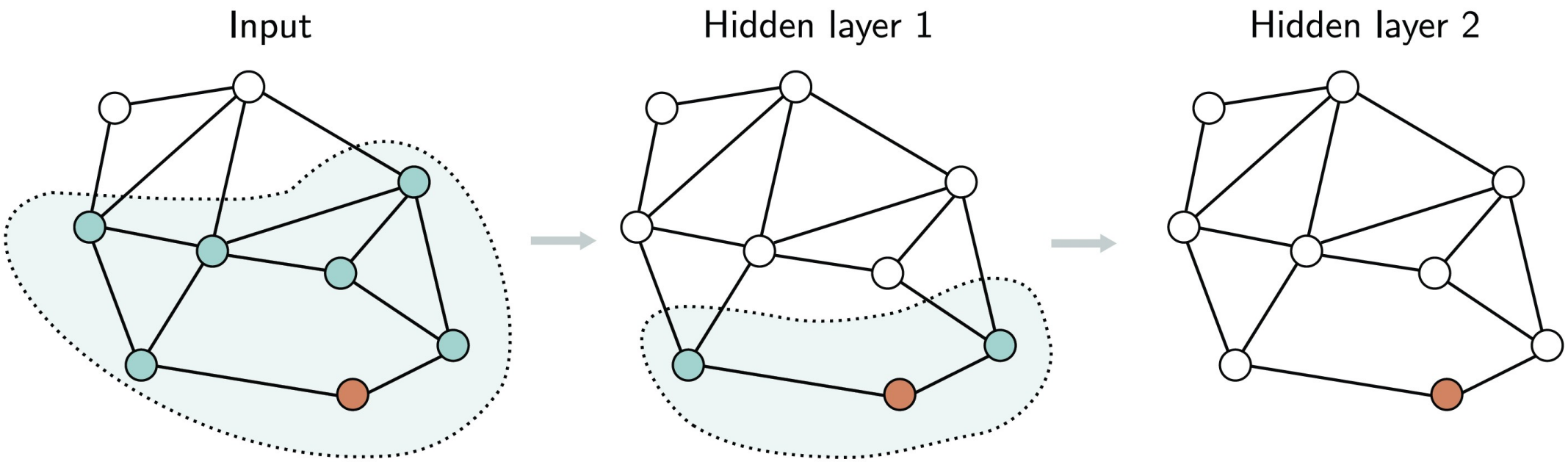


Figure 13.9 Receptive fields in graph neural networks. Consider the orange node in hidden layer two (right). This receives input from the nodes in the 1-hop neighborhood in hidden layer one (shaded region in center). These nodes in hidden layer one receive inputs from their neighbors in turn, and the orange node in layer two receives inputs from all the input nodes in the 2-hop neighborhood (shaded area on left). The region of the graph that contributes to a given node is equivalent to the notion of a receptive field in convolutional neural networks.

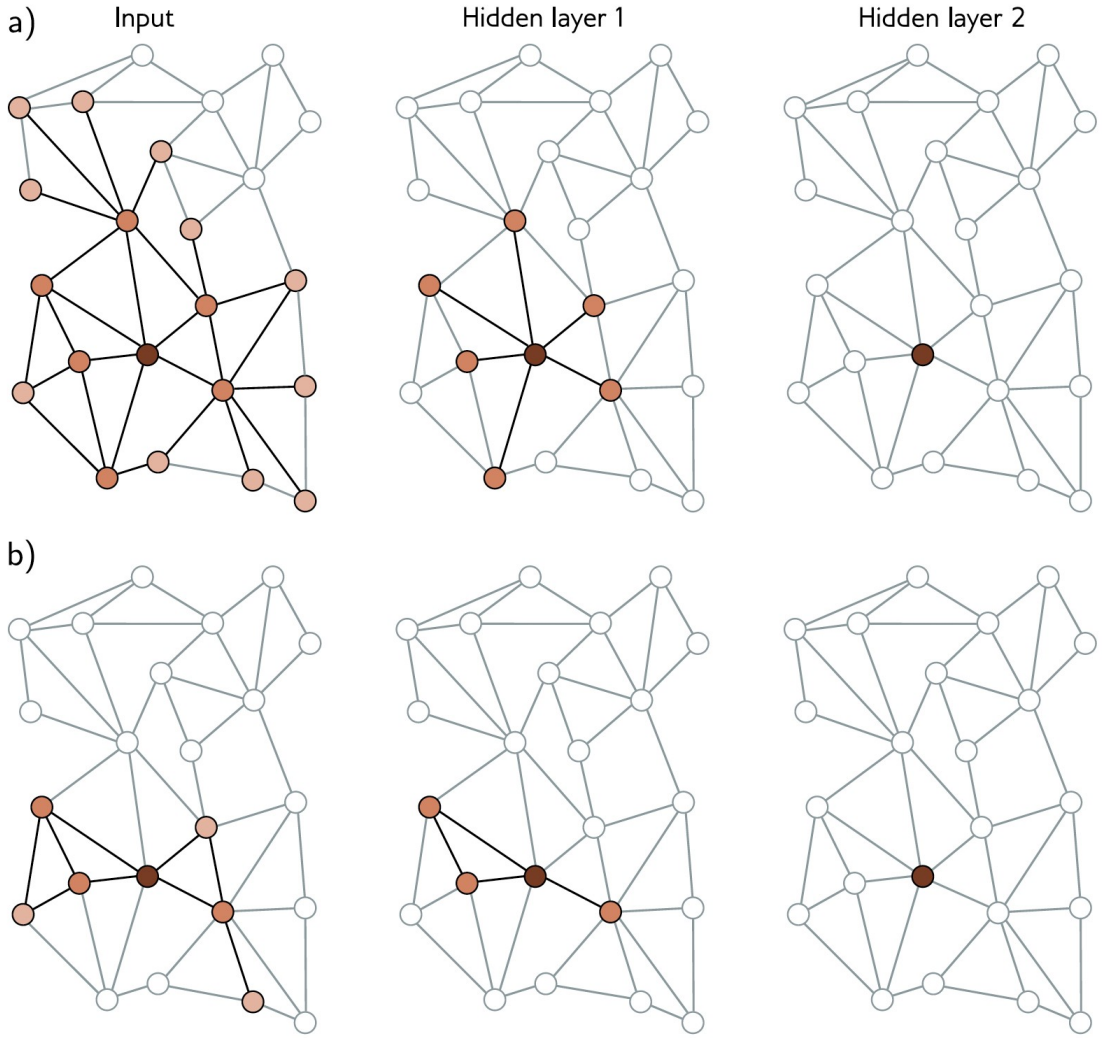


Figure 13.10 Neighborhood sampling. a) One way of forming batches on large graphs is to choose a subset of labeled nodes in the output layer (here, just one node in layer two, right) and then working back to find all of the nodes in the K-hop neighborhood (receptive field). Only this sub-graph is needed to train this batch. Unfortunately, if the graph is densely connected, this may retain a large proportion of the graph. b) One solution is neighborhood sampling. As we work back from the final layer, we select a subset of neighbors (here, three) in the layer before and a subset of the neighbors of these in the layer before that. This restricts the size of the graph for training the batch. In all panels, the brightness represents the distance from the original node.

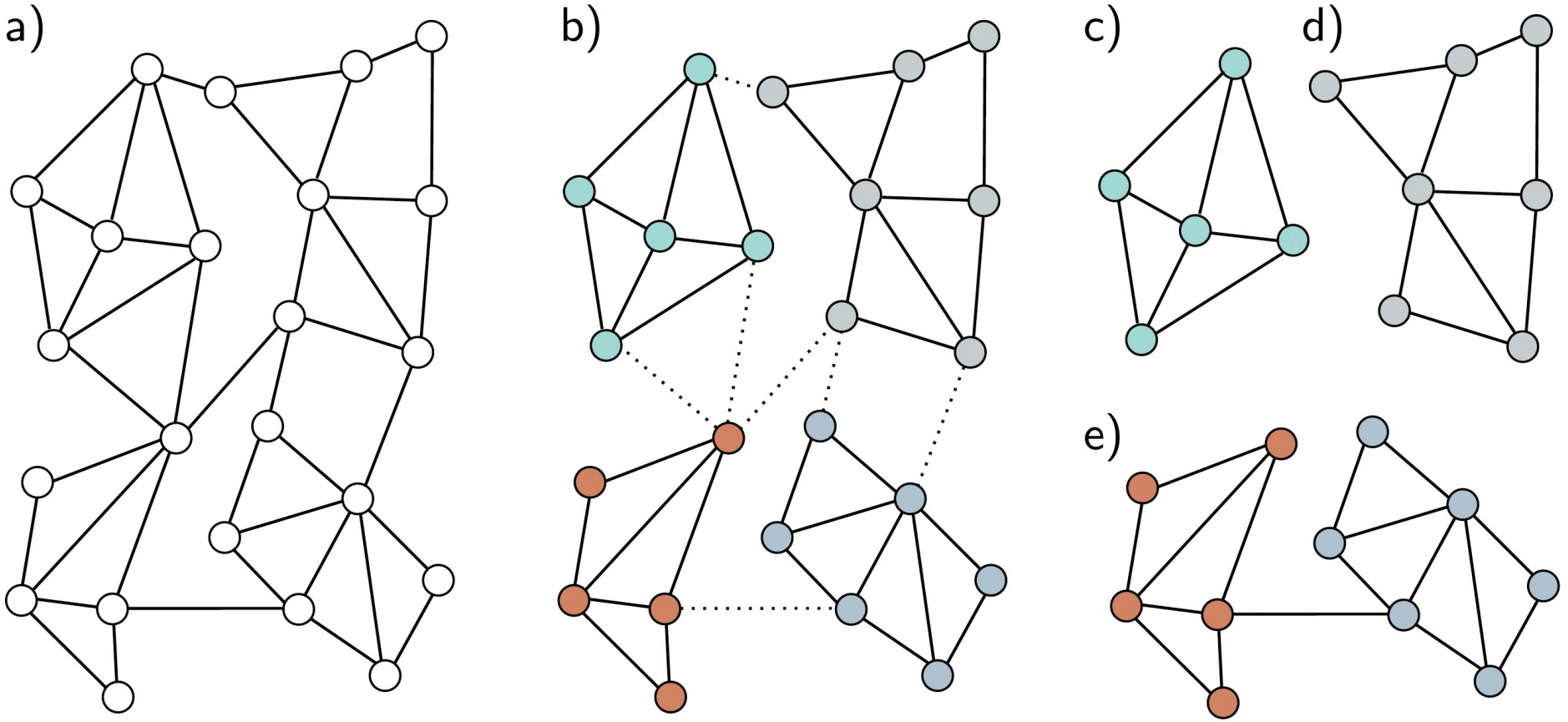


Figure 13.11 Graph partitioning. a) Input graph. b) The input graph is partitioned into smaller subgraphs using a principled method that removes the fewest edges. c-d) We can now use these subgraphs as batches to train in a transductive setting, so here, there are four possible batches. e) Alternatively, we can use combinations of the subgraphs as batches, reinstating the edges between them. If we use pairs of subgraphs, there would be six possible batches here.

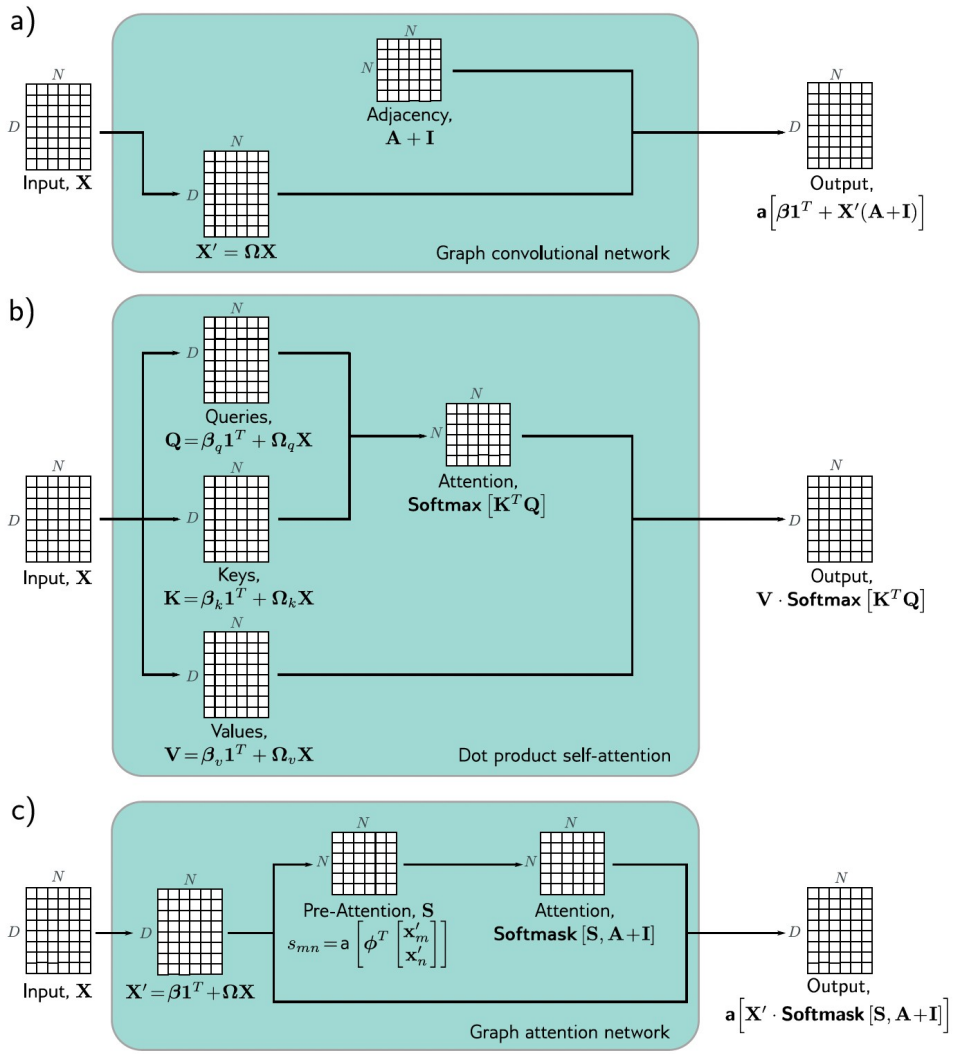


Figure 13.12 Comparison of graph convolutional network, dot product attention, and graph attention network. In each case, the mechanism maps N embeddings of size D stored in a $D \times N$ matrix $\mathbf{X}$ to an output of the same size. a) The graph convolutional network applies a linear transformation $\mathbf{X}' = \mathbf{\Omega X}$ to the data matrix. It then computes a weighted sum of the transformed data, where the weighting is based on the adjacency matrix. A bias $\boldsymbol{\beta}$ is added, and the result is passed through an activation function. b) The outputs of the self-attention mechanism are also weighted sums of the transformed inputs, but this time the weights depend on the data itself via the attention matrix. c) The graph attention network combines both of these mechanisms; the weights are both computed from the data and based on the adjacency matrix.

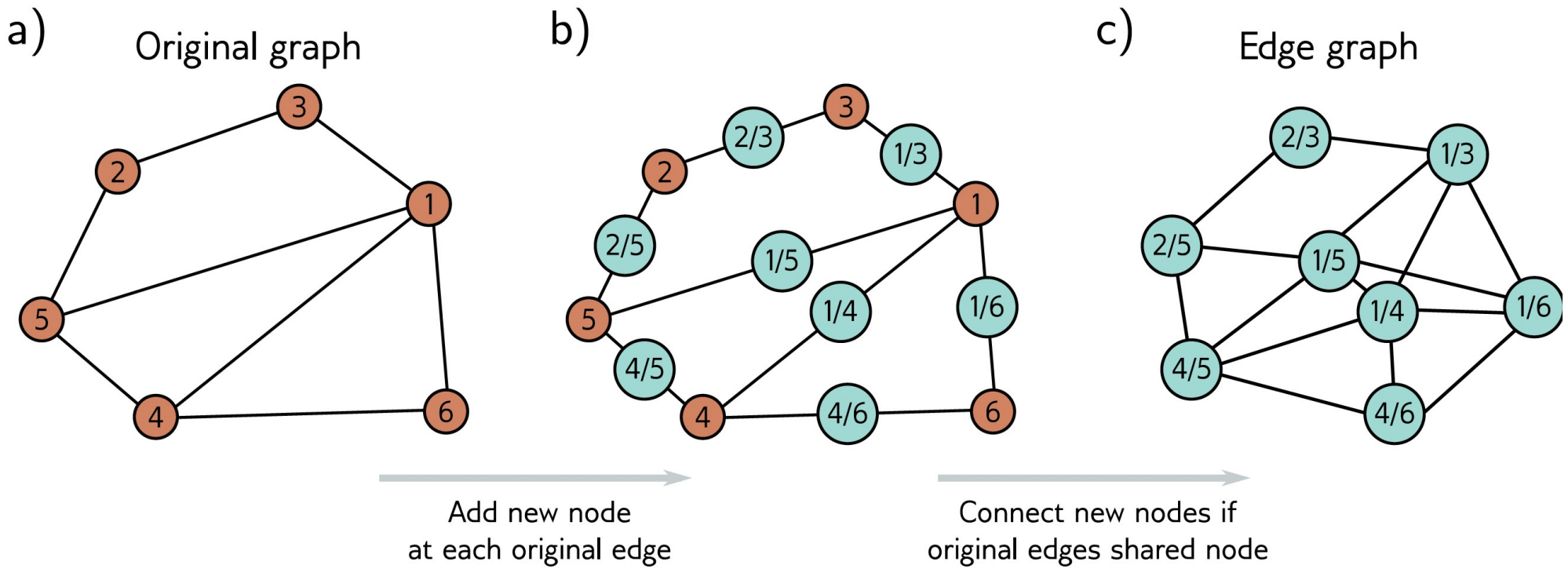


Figure 13.13 Edge graph. a) Graph with six nodes. b) To create the edge graph, we assign one node for each original edge (cyan circles), and c) connect the new nodes if the edges they represent connect to the same node in the original graph.

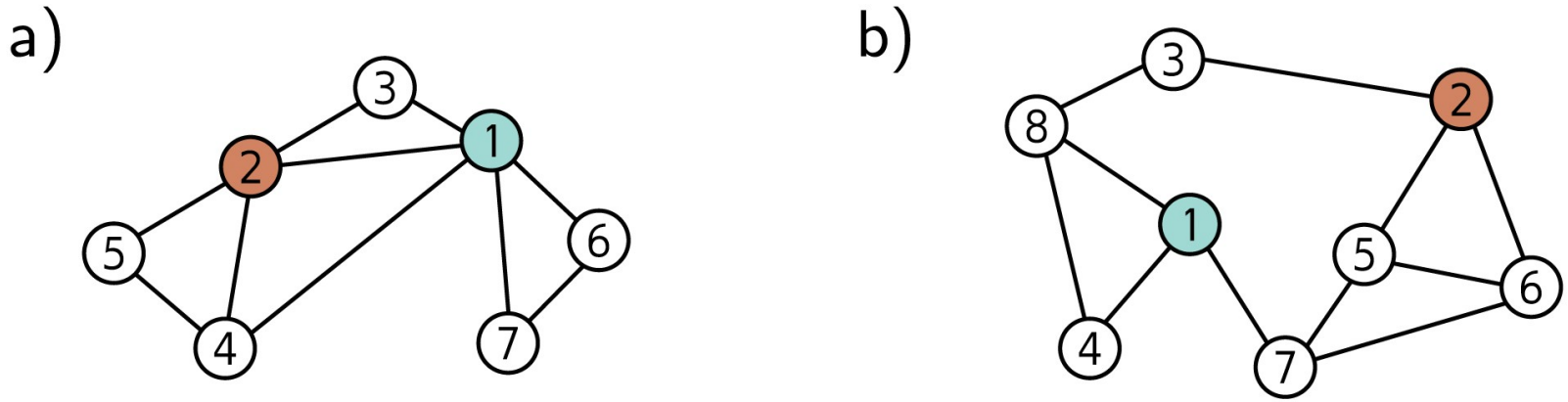


Figure 13.14 Graphs for problems 13.1, 13.3, and 13.8.

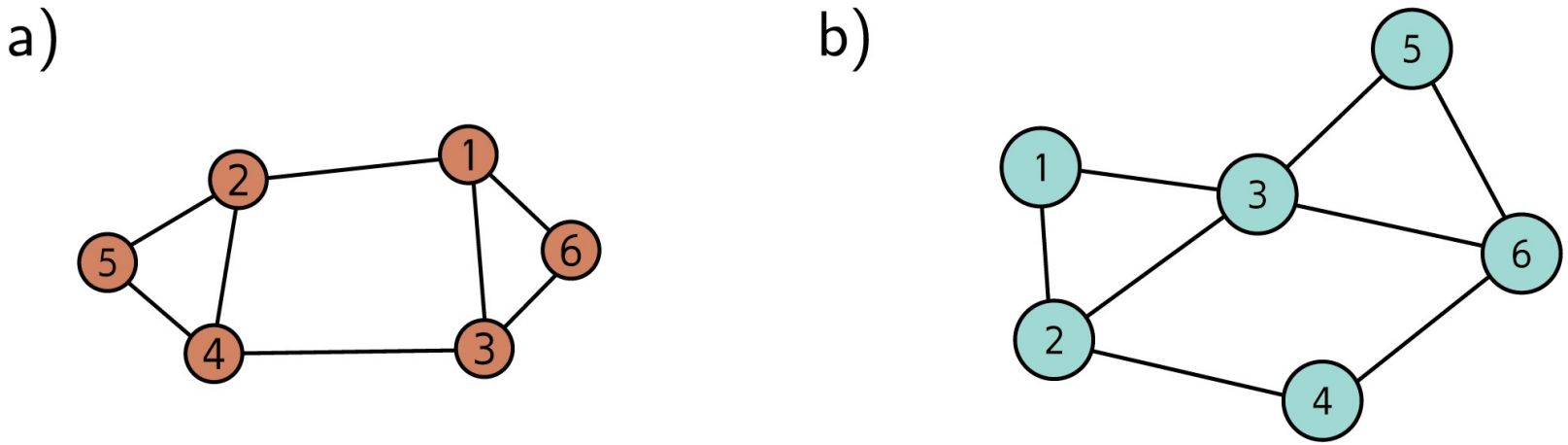


Figure 13.15 Graphs for problems 13.11–13.13.