Graph Neural Networks: Spectral Foundations and Real-World Applications

October 22, 2025

Contents

1 Graph Convolutional Networks (GCN)

Graph convolutional networks generalize convolution to irregular graph domains. For an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$ and node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, spectral graph theory provides the foundation for filtering operations. Figure **??** highlights the neighborhood aggregation process.

1.1 Spectral Formulation

Let $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denote the combinatorial Laplacian with degree matrix $\mathbf{D}$. The eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ yields graph Fourier bases. A spectral filter $g_\theta(\mathbf{\Lambda})$ applied to signal $\mathbf{X}$ is

$$g_\theta \star \mathbf{X} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{X}. \tag{1}$$

Chebyshev polynomials approximate g_θ with K-hop support:

$$g_\theta \star \mathbf{X} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}}) \mathbf{X}, \quad \tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}} \mathbf{L} - \mathbf{I}, \tag{2}$$

where T_k is the Chebyshev polynomial of order k. Kipf and Welling simplify to first-order ($K = 1$) by constraining θ:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right), \tag{3}$$

with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. This renormalization ensures numerical stability by symmetrically normalizing self-loops.

1.2 Message Passing Interpretation

The GCN layer can be viewed as message passing where nodes update via averaged neighbor features:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\tilde{d}_v \tilde{d}_u}} \mathbf{h}_u^{(l)} \mathbf{W}^{(l)} \right), \tag{4}$$

where $\tilde{d}_v$ counts neighbors with self-loop. Figure **??** depicts the flow of information across two GCN layers.

1.3 Over-smoothing and Remedies

Stacking many GCN layers causes node embeddings to converge, a phenomenon known as over-smoothing. Mitigation strategies include residual/skip connections, normalization (PairNorm, BatchNorm), and personalized propagation (APPNP) that blends teleportation:

$$\mathbf{Z} = (1-\alpha)\left(\mathbf{I} - \alpha\tilde{\mathbf{P}}\right)^{-1}\mathbf{H}^{(K)}, \tag{5}$$

with $\tilde{\mathbf{P}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$.

1.4 Training Pipeline

Listing 1: Two-layer GCN for semi-supervised node classification (PyTorch Geometric).

```python
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv

class GCN(torch.nn.Module):
    def __init__(self, in_dim, hidden_dim, out_dim, dropout=0.5):
        super().__init__()
        self.conv1 = GCNConv(in_dim, hidden_dim, normalize=True)
        self.conv2 = GCNConv(hidden_dim, out_dim, normalize=True)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

model = GCN(dataset.num_node_features, 64, dataset.num_classes)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)

for epoch in range(200):
    model.train()
    optimizer.zero_grad()
    out = model(data.x, data.edge_index)
    loss = F.nll_loss(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
```

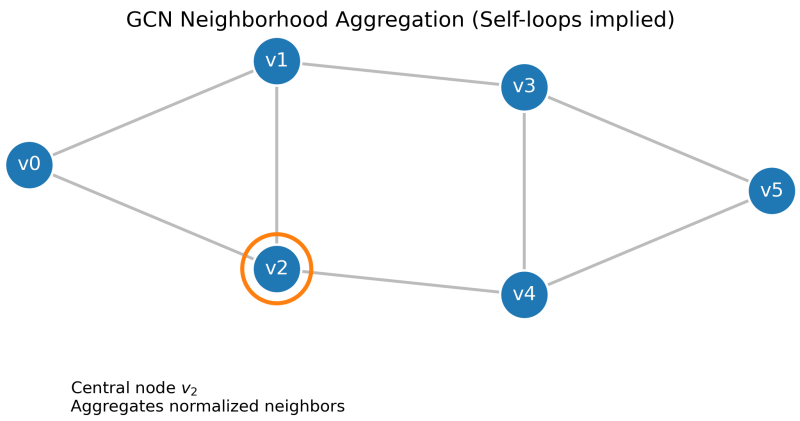

Figure 1: Graph convolution aggregates normalized neighbor information including self-loops. Degree normalization controls scale.

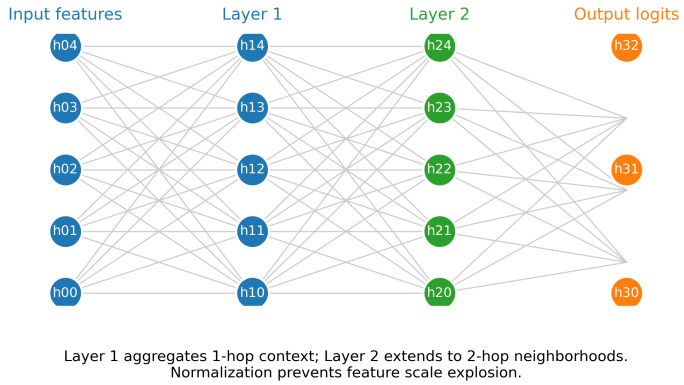

Figure 2: Two-layer GCN message passing: layer 1 captures one-hop neighborhoods, layer 2 extends to two hops.

2 Applications: Social Networks, Molecular Prediction, Recommender Systems

GCNs and related GNN architectures excel in domains where relational inductive biases matter. Figure **??** sketches representative pipelines.

2.1 Social Network Analysis

In social graphs, GCNs embed users with homophily-aware features. Tasks include community detection, influence prediction, and content recommendation. Heterogeneous graphs (users, posts, tags) demand relational GCN (R-GCN) with relation-specific weight matrices:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_r(v)}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}\right). \tag{6}$$

Graph sampling strategies (GraphSAGE, Cluster-GCN) support billion-edge training by subsampling neighborhoods.

2.2 Molecular Property Prediction

Molecules map naturally to graphs with atoms as nodes and bonds as edges. Message passing neural networks (MPNNs) generalize GCNs with edge updates:

$$\mathbf{m}_v^{(l+1)} = \sum_{u \in \mathcal{N}(v)} \phi^{(l)} \left(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}, \mathbf{e}_{uv} \right), \tag{7}$$

$$\mathbf{h}_v^{(l+1)} = \psi^{(l)} \left(\mathbf{h}_v^{(l)}, \mathbf{m}_v^{(l+1)} \right), \tag{8}$$

where $\mathbf{e}_{uv}$ encodes bond types. Global pooling ($\sum$, mean, set2set) aggregates molecular fingerprints. Combining quantum-inspired features and equivariant architectures (E(n)-GNN) yields state-of-the-art performance on QM9 and Materials Project benchmarks.

2.3 Recommender Systems

User-item interactions form bipartite graphs. LightGCN simplifies GCNs by removing nonlinearities and feature transformations:

$$\mathbf{E}^{(k+1)} = \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{E}^{(k)}, \qquad \mathbf{E} = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{E}^{(k)}. \tag{9}$$

The final embeddings produce recommendation scores via inner products, $s_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$. Real-world systems incorporate temporal dynamics, side information, and counterfactual debiasing.

2.4 Deployment Considerations

- **Scalability:** Neighbor sampling (PinSAGE), graph partitioning, and multi-GPU training handle web-scale graphs.
- **Explainability:** GNNExplainer and GraphMask identify influential substructures for decision transparency.
- **Robustness:** Adversarial perturbations on edges/nodes degrade performance; defense strategies include adversarial training and certified robustness via randomized smoothing.

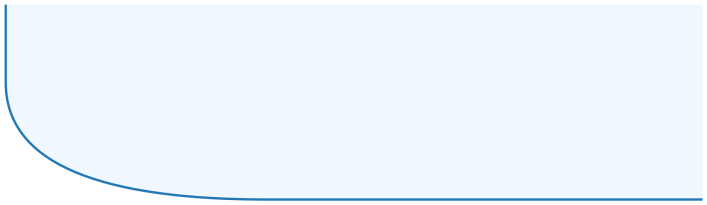

Figure 3: Applications of GNNs across social graphs, molecular chemistry, and recommender systems with domain-specific modules.

Further Reading

- Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." ICLR 2017.
- Petar Veličković et al. "Graph Attention Networks." ICLR 2018.
- Will Hamilton et al. "Inductive Representation Learning on Large Graphs." NIPS 2017.
- Keyulu Xu et al. "How Powerful are Graph Neural Networks?" ICLR 2019.
- He et al. "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation." SIGIR 2020.