Graph Neural Networks

- Slides based on snap.stanford.edu

http://snap.stanford.edu/proj/embeddings-www/

Based on material from:
Object Detection by ConvNet
Pixel is a node with a feature.
We update a feature by using features at neighbor-nodes.
Object Detection by ConvNet
Computer suggests actions by players

Klassen, Payet, Naldum going for a ball

Pogba marking Propper

Depay running for a free space

6 Pogba
10 Propper
18 Depay
Feature update
Deep learning on graphs — also known as Geometric deep learning (GDL)\(^1\), Graph representation learning (GRL), or relational inductive biases\(^2\) — has recently become one of the hottest topics in machine learning. While early works on graph learning go back at least a decade\(^3\), if not two\(^4\), it is undoubtedly the past few years' progress that has taken these methods from a niche interest into the spotlight of the ML community, complete with dedicated workshops attracting large crowds.

Graphs are mathematical abstractions of complex systems of relations and interactions. A graph represents a network of objects (called nodes or vertices) with pairwise connections (edges). Graphs are ubiquitously used in fields as diverse as biology\(^5\) \(^6\) \(^7\), quantum chemistry\(^8\), and high-energy physics\(^9\). Social networks like Twitter are examples of very large-scale complex graphs where the nodes model users and Tweets, while the edges model interactions such as replies, Retweets, or faves. Public conversations happening on our platform typically generate hundreds of millions of Tweets and Retweets every day. This makes Twitter perhaps one of the largest producers of graph-structured data in the world, second perhaps only to the Large Hadron Collider.
ConvNet (CNN)  
Fixed input dimension (3x3)

GraphNet (GNN)  
Arbitrary input dimension
Neighborhood Aggregation

- **Intuition:** Nodes aggregate information from their neighbors using neural networks.
Neighborhood Aggregation

**Intuition:** Network neighborhood defines a computation graph

Every node defines a unique computation graph!
Neighborhood Aggregation

- Nodes have embeddings at each layer.
- Model can be arbitrary depth.
- “layer-0” embedding of node $u$ is its input feature, i.e. $x_u$.
Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate information across the layers.

what’s in the box!?
Neighborhood Aggregation

- **Basic approach**: Average neighbor information and apply a neural network.

1) average messages from neighbors

2) apply neural network
The Math

- **Basic approach:** Average neighbor messages and apply a neural network.

\[ h^0_v = x_v \]

Initial “layer 0” embeddings are equal to node features

\[ h^k_v = \sigma \left( \sum_{u \in N(v)} \frac{h^{k-1}_u}{|N(v)|} + B^k h^{k-1}_v \right) \], \( \forall k > 0 \)

previous layer embedding of \( v \)

kth layer embedding of \( v \)

non-linearity (e.g., ReLU or tanh)

average of neighbor’s previous layer embeddings
Training the Model

- How do we train the model to generate “high-quality” embeddings?

Need to define a loss function on the embeddings, $L(z_u)$!
Training the Model

\[ h_v^0 = x_v \]
\[ h_v^k = \sigma \left( \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1} \right), \quad \forall k \in \{1, \ldots, K\} \]

- After K-layers of neighborhood aggregation, we get output embeddings for each node.
- We can feed these embeddings into any loss function and run stochastic gradient descent to train the aggregation parameters.
Training the Model

Human or bot?

Human or bot?

e.g., an online social network

Representation Learning on Networks, snap.stanford.edu/proj/embeddings-www, WWW 2018
Binary Cross-entropy = $- \left( p(x) \cdot \log q(x) + (1-p(x)) \cdot \log (1-q(x)) \right)$

This cancels out if the target is 0

This cancels out if the target is 1

$\mathcal{L} = \sum_{v \in V} y_v \log(\sigma(z_v^T \theta)) + (1 - y_v) \log(1 - \sigma(z_v^T \theta))$

Human or bot?

output node embedding

node class label

classification weights
Overview of Model Design

1) Define a neighborhood aggregation function.

2) Define a loss function on the embeddings, $L(z_u)$. 

INPUT GRAPH
Overview of Model Design

3) Train on a set of nodes, i.e., a batch of compute graphs
Overview of Model

4) Generate embeddings for nodes as needed

Even for nodes we never trained on!!!
Inductive Capability

Inductive node embedding generalize to entirely unseen graphs

e.g., train on protein interaction graph from model organism A and generate embeddings on newly collected data about organism B
Neighborhood Aggregation

- Key distinctions are in how different approaches aggregate messages

What else can we put in the box?
Graph Convolutional Networks

Basic Neighborhood Aggregation

$$h^k_v = \sigma \left( W_k \sum_{u \in N(v)} \frac{h^{k-1}_u}{|N(v)|} + B_k h^{k-1}_v \right)$$

VS.

GCN Neighborhood Aggregation

$$h^k_v = \sigma \left( W_k \sum_{u \in N(v) \cup v} \frac{h^{k-1}_u}{\sqrt{|N(u)||N(v)|}} \right)$$

same matrix for self and neighbor embeddings

per-neighbor normalization
Graph Convolutional Networks

- Empirically, they found this configuration to give the best results.
  - More parameter sharing.
  - Down-weights high degree neighbors.

\[
h_v^k = \sigma \left( W_k \sum_{u \in N(v) \cup v} \frac{h_{u}^{k-1}}{\sqrt{|N(u)||N(v)|}} \right)
\]

- Use the same transformation matrix for self and neighbor embeddings.
- Instead of simple average, normalization varies across neighbors.
GraphSAGE

Based on material from:
• Hamilton et al., 2017. Inductive Representation Learning on Large Graphs. NIPS.
GraphSAGE Idea

- So far we have aggregated the neighbor messages by taking their (weighted) average, can we do better?
Any differentiable function that maps set of vectors to a single vector.

\[ h^k_v = \sigma \left( [A_k \cdot \text{AGG}(\{h^{k-1}_u, \forall u \in N(v)\})], B_k h^{k-1}_v \right) \]
GraphSAGE Differences

- Simple neighborhood aggregation:

\[
h^k_v = \sigma \left( W_k \sum_{u \in N(v)} \frac{h^{k-1}_u}{|N(v)|} + B_k h^{k-1}_v \right)
\]

- GraphSAGE:

\[
h^k_v = \sigma \left( [W_k \cdot \text{AGG} (\{h^{k-1}_u, \forall u \in N(v)\})] , B_k h^{k-1}_v \right)
\]

concatenate self embedding and neighbor embedding

generalized aggregation
GraphSAGE Variants

- **Mean:**
  \[
  \text{AGG} = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}
  \]

- **Max (w/ matrix mult):**
  \[
  \text{AGG}_{\text{mean}}(\{Qh_u^{k-1}, \forall u \in N(v)\})
  \]
  \[
  \text{AGG}_{\text{max}}(\{Qh_u^{k-1}, \forall u \in N(v)\})
  \]

- **LSTM:**
  - Apply LSTM to random permutation of neighbors.
  \[
  \text{AGG} = \text{LSTM}([h_u^{k-1}, \forall u \in \pi(N(v))])
  \]
Gated Graph Neural Networks

Based on material from:
- Li et al., 2016. Gated Graph Sequence Neural Networks. ICLR.
Neighborhood Aggregation

- **Basic idea:** Nodes aggregate “messages” from their neighbors using neural networks.
Neighborhood Aggregation

- GCNs and GraphSAGE generally only 2-3 layers deep.
Neighborhood Aggregation

- But what if we want to go deeper?

10+ layers!?
Gated Graph Neural Networks

- How can we build models with many layers of neighborhood aggregation?
- **Challenges:**
  - Overfitting from too many parameters.
  - Vanishing/exploding gradients during backpropagation.
- **Idea:** Use techniques from modern recurrent neural networks!
Gated Graph Neural Networks

RNN module!

INPUT GRAPH

TARGET NODE
The Math

- **Intuition:** Neighborhood aggregation with RNN state update.

1. Get “message” from neighbors at step $k$:

   $$m^k_v = W \sum_{u \in N(v)} h^{k-1}_u$$

   The aggregation function does not depend on $k$.

2. Update node “state” using **Gated Recurrent Unit (GRU)**. New node state depends on the old state and the message from neighbors:

   $$h^k_v = \text{GRU}(h^{k-1}_v, m^k_v)$$
Gated Graph Neural Networks

- Can handle models with >20 layers.
- Most real-world networks have small diameters (e.g., less than 7).
- Allows for complex information about global graph structure to be propagated to all nodes.
Gated Graph Neural Networks

- Useful for complex networks representing:
  - Logical formulas.
  - Programs.
Summary

- **Graph convolutional networks**
  - Average neighborhood information and stack neural networks.

- **GraphSAGE**
  - Generalized neighborhood aggregation.

- **Gated Graph Neural Networks**
  - Neighborhood aggregation + RNNs
Recent advances in graph neural nets (not covered in detail here)

- Attention-based neighborhood aggregation:
  - Graph Attention Networks ([Velickovic et al., 2018](https://arxiv.org/abs/1710.10903))
  - GeniePath ([Liu et al., 2018](https://arxiv.org/abs/1801.06341))

- Generalizations based on spectral convolutions:
  - Mixture Model CNNs ([Monti et al., 2017](https://arxiv.org/abs/1606.07275))

- Speed improvements via subsampling:
  - FastGCNs ([Chen et al., 2018](https://arxiv.org/abs/1711.07553))
  - Stochastic GCNs ([Chen et al., 2017](https://arxiv.org/abs/1703.09894))