Learning the Structure of Graph Neural Networks

Mathias Niepert

NEC Laboratories Europe
Heidelberg

Heidelberg.AI Meetup
July 9th, 2019
NEC Laboratories Europe

- ~80 researchers, ~80% PhDs, 22 nationalities
- Research lab, no product development
- Main objectives:
  1. Research output at top tier conferences
  2. Stable prototypes for technology transfer
  3. Patent applications
- Located in Heidelberg (Kurfuersten-Anlage)
Main Research Themes

- **Multi-Modal Learning and Reasoning**
  - Combining different attribute types and modalities
  - **Knowledge graphs for multi-modal learning**
    (combining deep learning and logical reasoning)

- **Graph-based Machine Learning**
  - Learning graph representations
  - **Unsupervised and semi-supervised learning**

- **Systems and ML**
  - ML for Systems and Systems for ML
  - CPU/GPU/network optimizations
  - Deep learning for data networks
Technology Challenges

- ML that works with missing labels, missing features, and biased and protected data
- Verifiable and inspectable AI
- Ability to combine different data modalities (finding suitable representations for multi-model data that facilitates joint learning)
- Efficiency and support of real time predictions (network speed if required)
- Extraction and integration of multi-modal information from unstructured and structured data
- Modeling uncertainty in predictions of complex models

Tokyo (Japanese: [to-kojo] ( listen), English: /ˈtoʊki.oʊ/), officially Tokyo Metropolis, is the capital of Japan and one of its 47 prefectures.
Graph Data

Patient network

Collection of compound graphs

Protein-drug-side effect network

Schizophrenia protein interaction graph
Geometric Deep Learning

Data on a sphere

Point cloud

Tabular data

Data on a manifold
Observation: Effective representations are often composed bottom-up from local representations

- Parameter sharing
- Hierarchical features
- Model tractability (statistically and computationally)

Example: Convolutional neural networks

Question: What is a suitable notion of locality in graphs?
**Learning Representations of Structured Data**

---

**Image CNN**
- Grid graph required (spatial order)
- Works only for images

Standard CNN moves over image

---

**Graph-Based CNN**
- Arbitrary input graph
- Node attributes
- Edge attributes

What are good local features?

---

**Convolutional Network**

**Neural Network**

---

8 Learning the Structure of Graph Neural Networks
Graph-Based Machine Learning

Learn representations for entire graphs

[Diagram showing graph classification/regression problems]

Learn representations for nodes

[Diagram showing node classification/regression and link prediction problems]

Induce graph
Example Application: Drug Discovery

Learn representations for entire graphs

Graph classification/regression problems
Example Application: Neoantigen Discovery

Learn representations for nodes

Induce graph

Node classification/regression problems

Learn representations for nodes

Induce graph

Node classification/regression problems

Legend

○: observed raw data
×: missing raw data

Structure created by sequence similarity

Epitope ranking

GNN
Example Application: Neoantigen Discovery

Collaboration between NEC and Transgene
Phase 1 clinical trial for HPV-negative head and neck cancer

Tumor cell → Identification and selection of neoantigens → Integration of neoantigens in the genome of the viral vector → Individualized immunotherapy
Example Application: Patient Outcome Prediction

Learn representations for nodes

Induce graph

Node classification/regression problems

Malone et al., Learning Representations of Missing Data for Predicting Patient Outcomes
Learning the Structure of Graph Neural Networks

Example Applications: Polypharmacy Prediction

Learn representations for nodes

- Gotoh Museum
- Murasaki Shikibu
- Tokyo
- Japan
- Sensō-ji

Link prediction problem

- DrugA
- ?
- DrugB

© SNAP 2018
Outline

1. Basic Concepts

2. Three Types of Local Structure
   - Triples
   - Paths and random walks
   - Neighborhoods

3. Learning the Structure of GNNs
Basic Graph Terminology

- Node identifier
- Node label

- Node
- Entity

- Relation
- Edge

- Node features
- Node attributes
Learning Representations for Graphs

**Image CNN**
- Grid graph required (spatial order)
- Works **only** for images

**Graph NN**
- Arbitrary input graph
- Node attributes
- Edge attributes

What are good local features?

**Convolutional Network**

**Triples**
- \( f(\ ) \)

**Paths**
- \( f(\ ) \)

**Neighborhoods**
- \( f(\ ) \)
Three Types of Local Structure (I)

The multi-relational graph as a **3D tensor**

Local structure = triple: (subject, relation, object)
**Step 1:** Choose the representation (encoding) for entities and relations

Entities: \( e_i \)

Relation types: \( W_r \)

**Step 2:** Choose scoring function for **triples** \((s, r, o)\) = coordinates in the 3D tensor

\[
f(s, r, o) = e_s^T \cdot W_r \cdot e_o
\]

**Step 3:** Choose loss function

\[
\sum_{s,r,o} (T_{s,r,o} - f(s,r,o))^2
\]
Many alternative scoring functions have been proposed.

**Triple**: \((s, r, o)\)

**Scoring function**

\[ f(||, ||, ||) \]

**Embeddings**:

- Same size vectors

---

<table>
<thead>
<tr>
<th>Model</th>
<th>Scoring Function</th>
<th>Relation parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL (Nickel et al., 2011)</td>
<td>(e_s^T W_r e_o)</td>
<td>(W_r \in \mathbb{R}^{K^2})</td>
</tr>
<tr>
<td>TransE (Bordes et al., 2013b)</td>
<td>(</td>
<td></td>
</tr>
<tr>
<td>NTN (Socher et al., 2013)</td>
<td>(u_r^T f(e_s W_r^{[1..D]} e_o + V_r \begin{bmatrix} e_s \ e_o \end{bmatrix} + b_r)</td>
<td>(W_r \in \mathbb{R}^{K^2D}, b_r \in \mathbb{R}^K, V_r \in \mathbb{R}^{2KD})</td>
</tr>
<tr>
<td>DistMult (Yang et al., 2015)</td>
<td>(&lt;w_r, e_s, e_o&gt;)</td>
<td>(w_r \in \mathbb{R}^K)</td>
</tr>
<tr>
<td>HolE (Nickel et al., 2016b)</td>
<td>(w_r^T (\mathcal{F}^{-1}[\mathcal{F}[e_s] \odot \mathcal{F}[e_o]]))</td>
<td>(w_r \in \mathbb{R}^K)</td>
</tr>
<tr>
<td>ComplEx</td>
<td>(\text{Re}(&lt;w_r, e_s, e_o&gt;))</td>
<td>(w_r \in \mathbb{C}^K)</td>
</tr>
</tbody>
</table>

Trouillon et al. 2016
Three Types of Local Structures (II)

Learning from paths and random walks

**NB:** Learning from local structures can capture global properties through a recursive propagation process
Methods for Path Extraction

Perform a large number of **Random Walks**

Keep the paths most frequently encountered
Methods for Learning from Paths

- Interpret every walk as a sentence (sequence of nodes visited)
- Train word embedding method such as Word2vec

Continuous bag of nodes
DeepWalk

Results in node embeddings which can be used for other tasks

(a) Random walk generation.       (b) Representation mapping.       (c) Hierarchical Softmax.

© Perozzi

Skip-gram model
Three Types of Local Structures (III)

Learning from Local Neighborhoods

NB: Learning from local structures can capture global properties through a recursive propagation process between nodes.
How do we aggregate neighborhood information into fixed-size representations? → requirement for weight sharing

Feature transformations are applied locally for each node on its neighborhood

Requires ability to work with highly heterogeneous neighborhood structures
Graph Neural Networks

Message Passing Neural Networks

\[ f_w(\cdot): \quad x_i^{(k)} = g \left( x_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} h \left( x_j^{(k-1)} \right) \right) \]

- \( x_i^{(k)} \): Node representation
- \( \mathcal{N}(i) \): Neighbors of node \( i \)
- \( g, h \): Differentiable functions parameterized by \( w \)

\( g \) and \( h \) are differentiable functions parameterized by \( w \)
Graph Neural Networks

Message Passing Neural Networks

\[ f_w(\cdot) : x_i^{(k)} = g \left( x_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} h \left( x_j^{(k-1)} \right) \right) \]

Node representation \( x_i \)

neighbors of node \( i \)

Node representations \( x_i \)

Classification function + loss applied at nodes with labels

GCN for node classification is function \( f_w \) of features \( X \) and Laplacian \( A \) (& edge features if available), parameterized by \( w \)
Real-World Graphs

- **Noisy** graph structure inferred from experimentation
- Graph structure *induced with heuristics* (kNN graph, ...)
- Often no graph structure available at all

We want to include graph-structure learning in an “end-to-end” differentiable stochastic function
We want the connectivity structure to be \textit{discrete} and \textit{sparse}

\textbf{Commonly followed strategy:}
1. Compute k-nearest neighbor graph based on input features
2. Apply a graph-based ML method leveraging structure
   \textbf{Examples:} LLE, Isomap, k-NN graph + GCN, ...

\textbf{Problems:}
- Efficacy hinges on similarity measure and suitability of input features
- 2-step process and not “end-to-end” differentiable

\textbf{We want to include graph-structure learning in an “end-to-end” differentiable stochastic function}
Learning the Structure of Graph Neural Networks

1. Take a **graph generative model** $P_\theta$ with parameters $\theta$ (a probabilistic model from which we ought to be able to draw samples efficiently).

2. During training, we sample adjacency matrices from $P_\theta$ and use them as training data for a GNN with parameters $w$.

3. This results in a stochastic computation graph (a latent variable model with discrete variables).

4. We "backpropagate into" both $\theta$ and $w$ based on a classification loss on the nodes with class labels.
Node classification with LDS

- Given graph generative model $P_\theta$ and trained GCN $f_w$ with parameters $w$ we want to compute for input features $X$

\[ \hat{f}(X) = \mathbb{E}_{A \sim P_\theta}[f_w(X, A)] \]

- Sample $S$ graphs from $P_\theta$

- Evaluate GCN on each of the graphs and average the results to approximate of the above expectation term

\[ \hat{f}(X) \approx \sum_{i=1}^{S} f_w(X, A_i) \]
Learning with LDS (I)

With a classification loss $\mathcal{L}$ this results in the objective

$$\min_{\theta} \min_{w} \mathbb{E}_{A \sim P_\theta} [\mathcal{L}(f_w(X, A), Y)]$$

Can be written as a bilevel optimization objective

Outer objective: $\min_{\theta} \mathbb{E}_{A \sim P_\theta} [\mathcal{L}(f_{w_\theta}(X, A), Y_{val})]$ such that

Inner objective: $w_\theta = \arg\min_w \mathbb{E}_{A \sim P_\theta} [\mathcal{L}(f_w(X, A), Y_{train})]$
Learning with LDS (II)

**Inner objective:** \( \mathbf{w}_\theta = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{A} \sim P_\theta} \left[ \mathcal{L}(f_\mathbf{w}(\mathbf{X}, \mathbf{A}), \mathbf{Y}) \right] \)

- Approximate solution to the inner objective with stochastic gradient decent over the training nodes

\[
P_\theta \sim \mathbf{A} \quad \quad \mathbf{w}_{\theta}^{t+1} = \mathbf{w}_{\theta}^t - \gamma \nabla \mathcal{L}(f_{\mathbf{w}_\theta^t}(\mathbf{X}, \mathbf{A}), \mathbf{Y})
\]

- Compute gradient with respect to \( \theta \) of the **outer objective** over the validation nodes (with straight-through estimator and projected gradient decent)

\[
\nabla_\theta \mathbb{E}_{\mathbf{A} \sim P_\theta} \left[ \mathcal{L}(f_{\mathbf{w}_\theta}(\mathbf{X}, \mathbf{A}), \mathbf{Y}) \right] = \mathbb{E}_{\mathbf{A} \sim P_\theta} \left[ \nabla_\theta \mathcal{L}(f_{\mathbf{w}_\theta}(\mathbf{X}, \mathbf{A}), \mathbf{Y}) \right]
\]
Learning with LDS (III)

Initialize parameters
Graph generator: $\theta$
GCN: $W$

Sample graphs
$A_1 \sim P_\theta$
\[ \cdots \]
$A_{\tau} \sim P_\theta$

Compute gradients of and update GCN parameters
$w_{t+1} = \Phi(w_t, A_1) = w_t - \gamma \nabla L_t(w_t, A_1)$
\[ \cdots \]
$w_{t+\tau} = w_{t+\tau-1} - \gamma \nabla L_{t+\tau-1}(w_{t+\tau-1}, A_{\tau})$

Compute hypergradients and update $\theta$ of graph generator
$\nabla_\theta \mathbb{E}[F(w_{0,\tau}, \theta)]$

Validation nodes
$w_t$ \[ \cdots \] $w_{t+\tau-1}$ \[ $w_{t+\tau}$ \]}
LDS as a Framework

- **Graph generative model** should be efficient as we have to draw samples from it in the inner optimization loop.

  ![Graph generative model diagram]

  We use a simple generative model with one Bernoulli random variable per edge.

  Alternative models are possible (e.g. stochastic block model).

  As GNN we use Kipf & Welling’s GCN (but others possible).
Semi-supervised classification problem with incomplete graphs

- Classify scientific papers based on topics. Use a citations dataset as input graph where we remove 0-25-50-75% of the citations

- tau is the number of steps of SGD for inner objective

- Considerably better than Graph Convolutional Networks (GCN) on citations, even when LDS can use fewer citations as input

- The difference comes from more complete graph generated by LDS, since randomizing citations (GCN-RND) does not help GCN perform better
Learned Edge Distributions

Histogram of three nodes misclassified by vanilla GCN and correctly classified by LDS
Conclusion

- Many learning problems benefit from relational structure between data points → relational learning
- LDS is akin to “architecture search” for GNNs
- Exciting intersection with Monte-Carlo gradient estimation and the “tricks” required for backprop to work there

References

- Franceschi et al., Learning Discrete Structures of Graph Neural Networks, ICML 2019
- Garcia-Duran et al., Learning Graph Representations with Embedding Propagation, NIPS 2017
- Malone et al., Learning Representations of Missing Data for Predicting Patient Outcomes, arxiv 2018
Questions?

Social network

Knowledge graph

Brain connectivity graph

Schizophrenia protein interaction graph