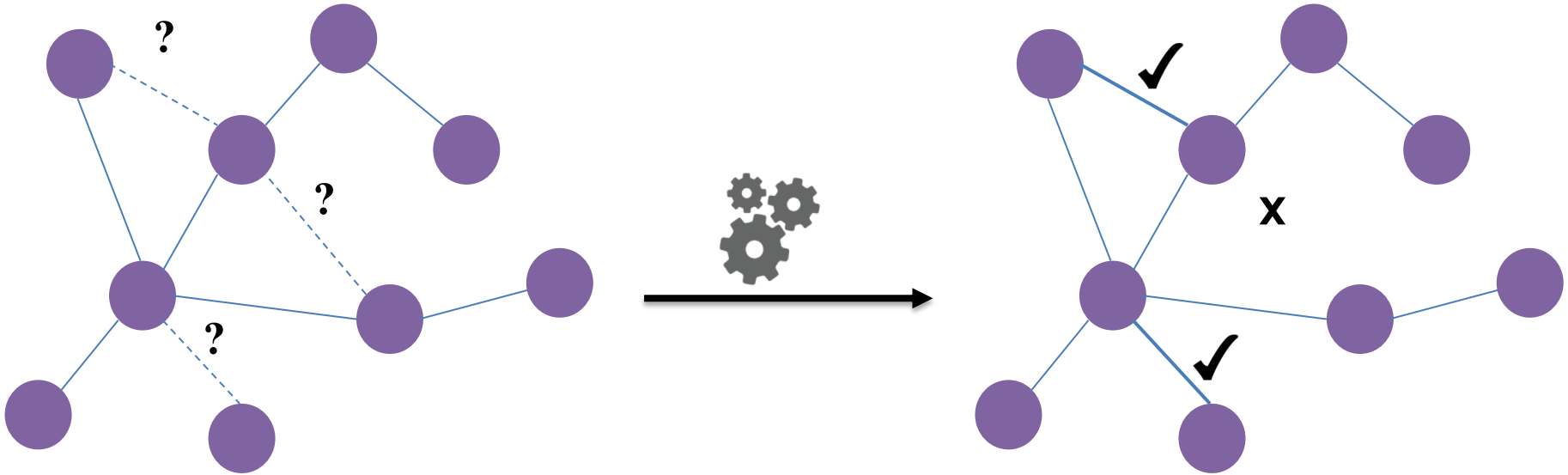
Meta Learning and Logical Induction on Graphs

William L. Hamilton

Assistant Professor at McGill University and Mila – AI Institute of Quebec

CIFAR Canada AI Chair

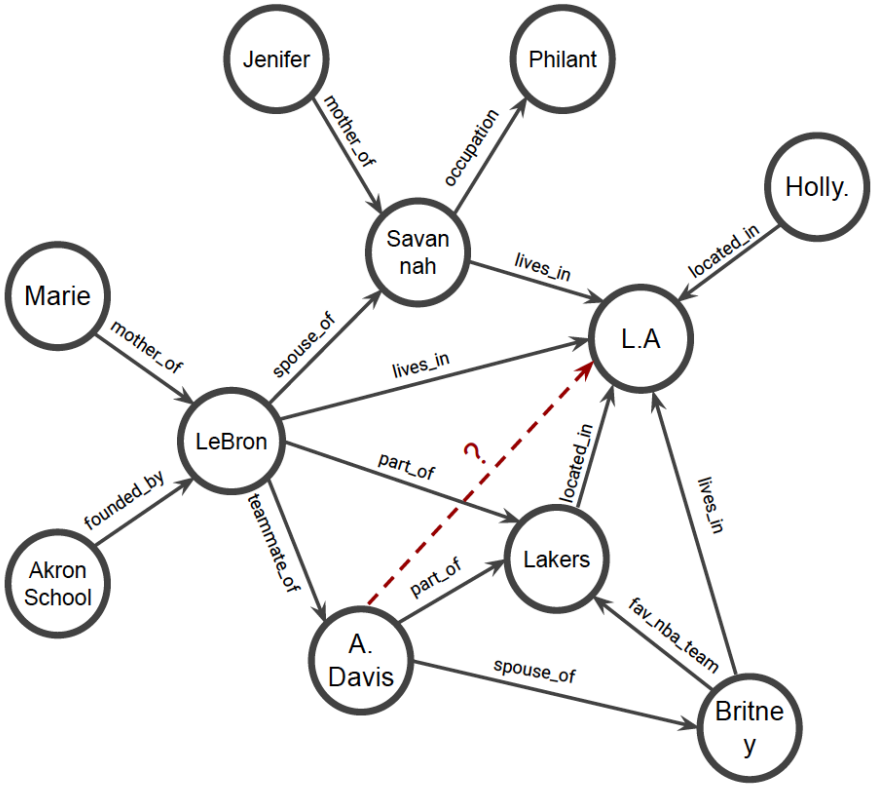
Relation prediction on graphs



Example: Knowledge graph completion

Key idea: Automatically expand and complete existing knowledge bases.

Applications in e-commerce, medicine, materials science…



Example: Computational drug design

Example tasks:

- Repurposing
- Side effect prediction
- Polypharmacy

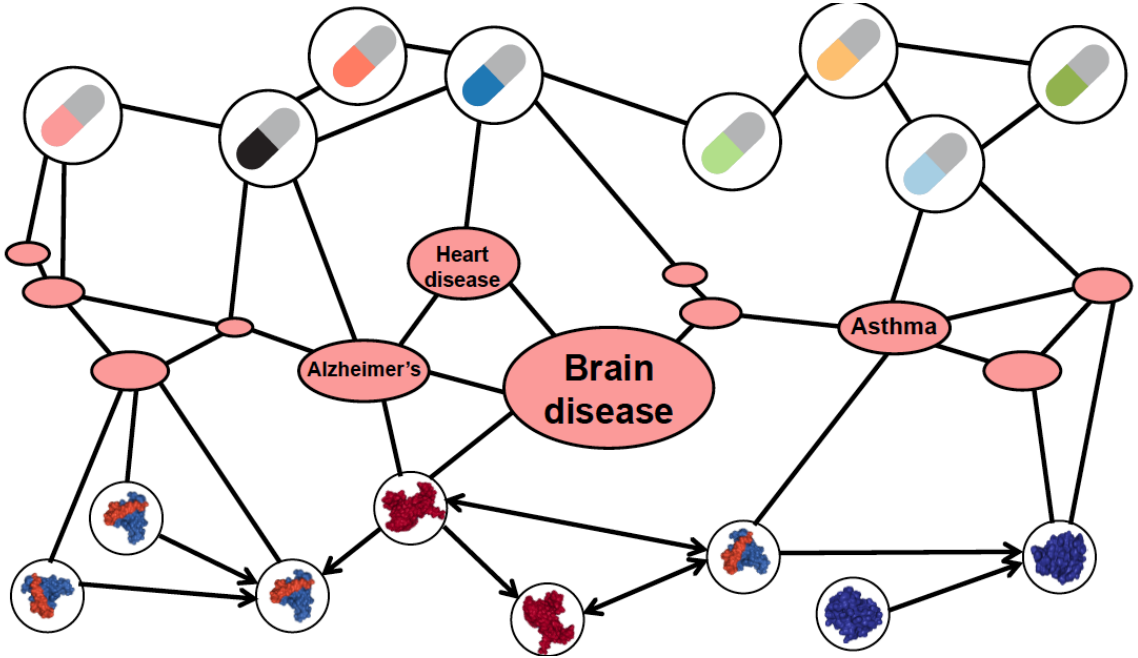
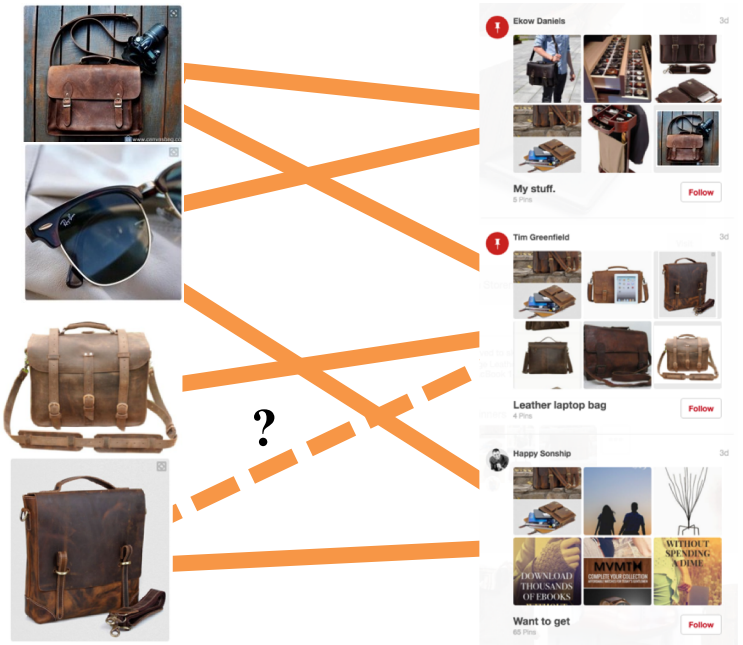


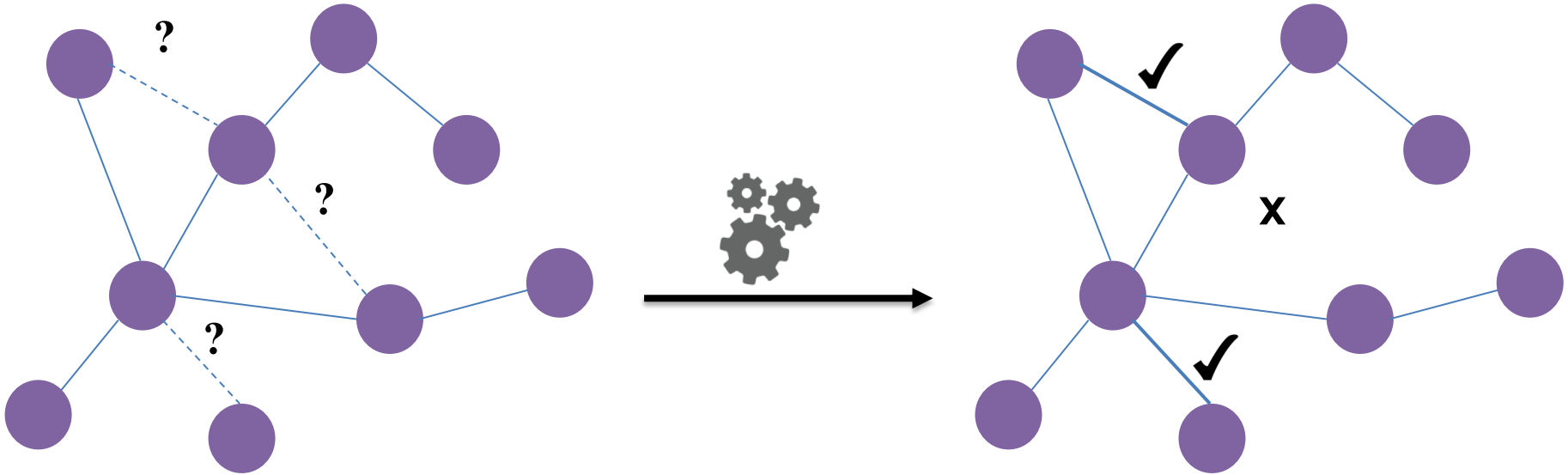
Image credit: Marinka Zitnik

Example: Content recommendation

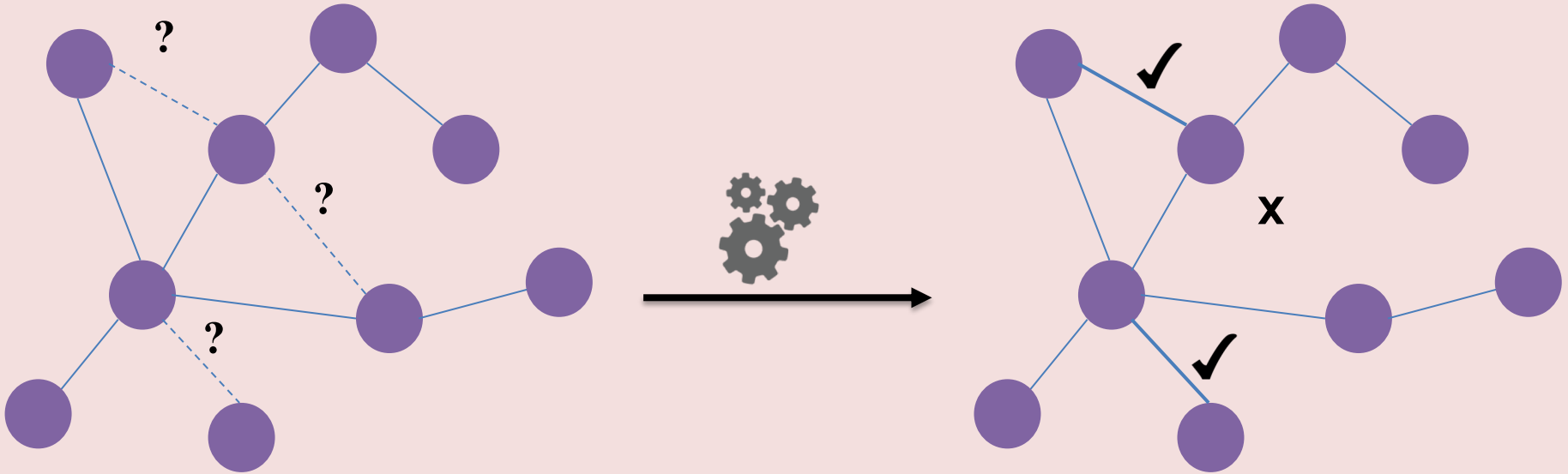
Content recommendation is link prediction!



Transductive relation prediction



Transductive relation prediction

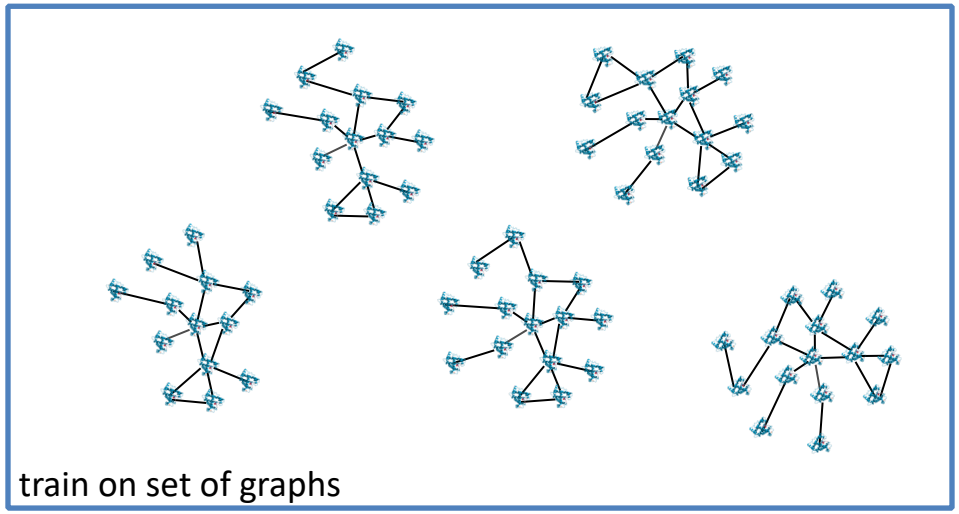


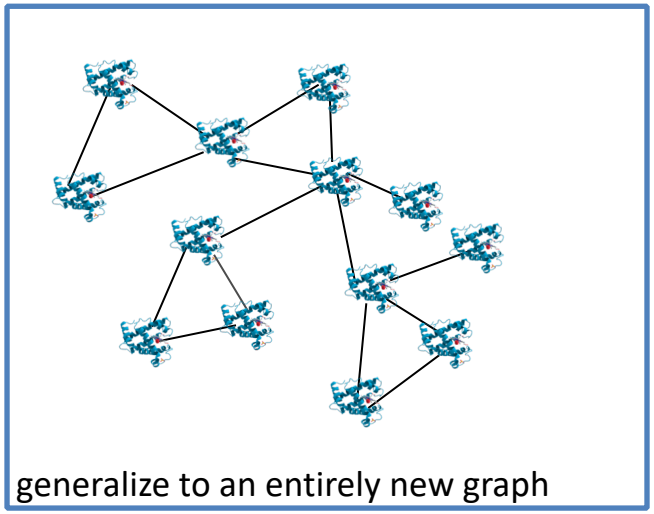
TRANSDUCTIVE

Limitations of transductive relation prediction

- Problematic for production systems
  - Need to re-train to deal with new nodes (e.g., entities, products)
  - Predictions can become stale.
- Too many parameters
  - Most transductive approaches have $O(|V|)$ space complexity.
- Biases model development
  - Focus on "embedding"-based methodologies.
  - Static and unrepresentative benchmarks

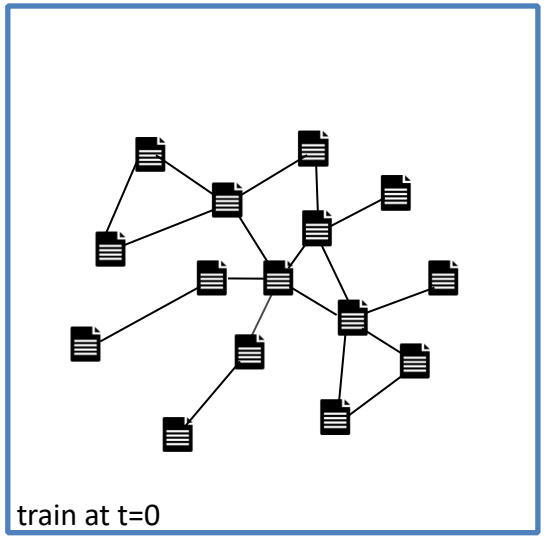
Inductive learning: multiple graphs

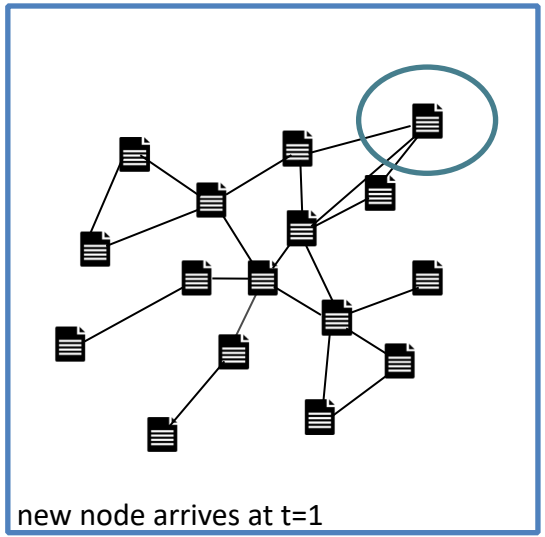


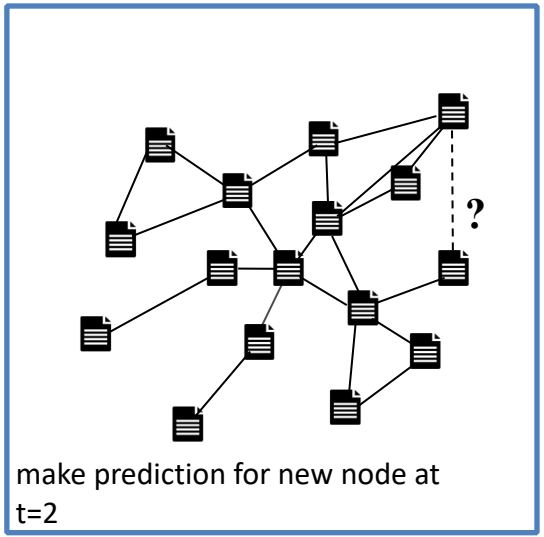


Example applications: High-throughput interactomics, recommender systems

Inductive learning: evolving data

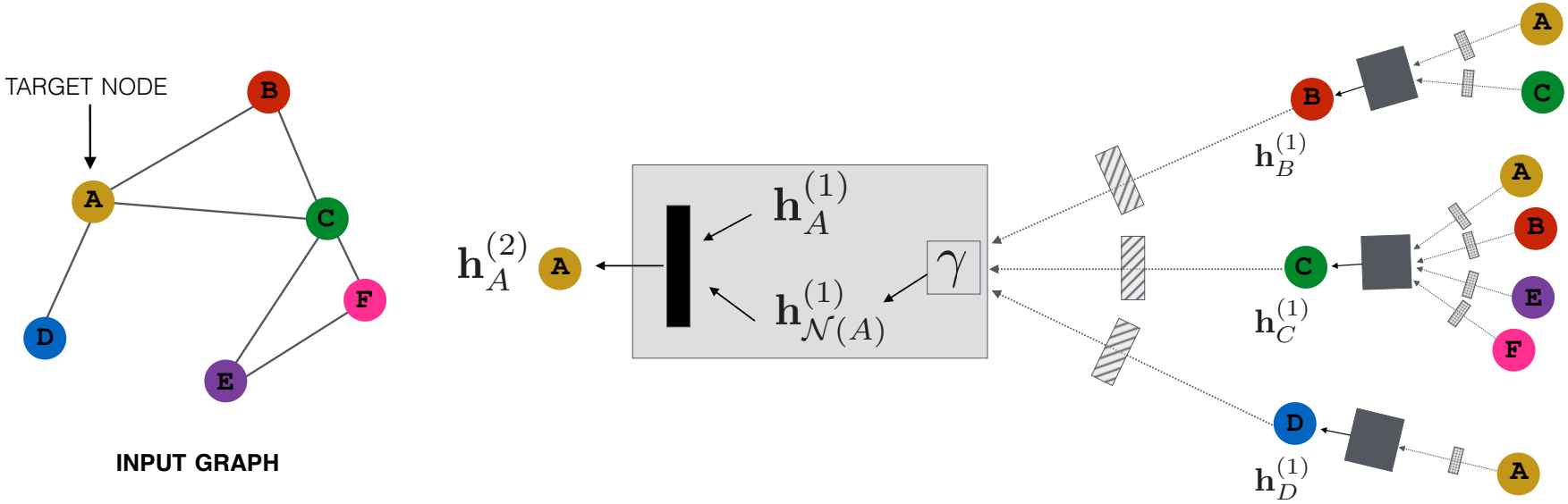






Example applications: Knowledge graph completion, recommender systems, …

Inductive learning via graph neural networks



Examples: GraphSAGE (Hamilton et al., 2017)

Significant challenges remain

- Inductive learning without node features
- Multi-relational data (i.e., knowledge graphs)
- Theoretical understanding of inductive learning
- Few shot learning

Part I

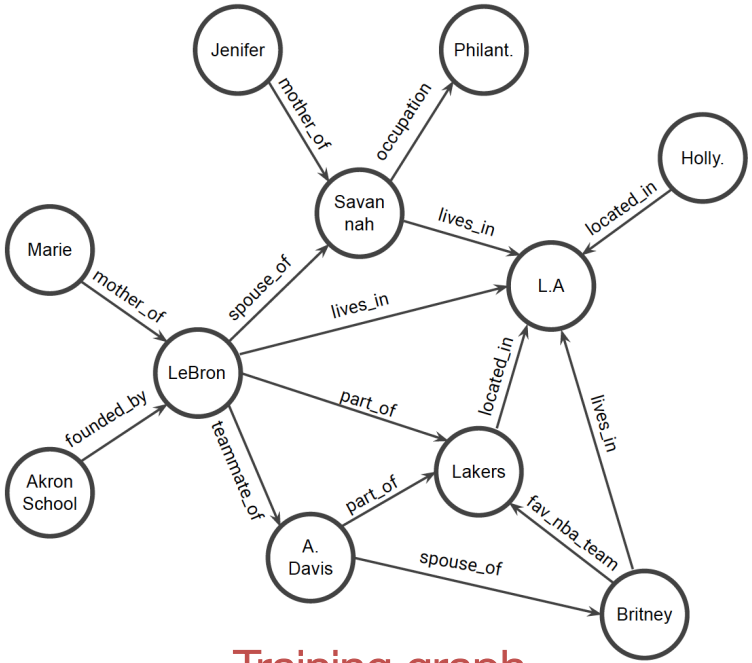
INDUCTIVE LEARNING ON KNOWLEDGE GRAPHS

Part I

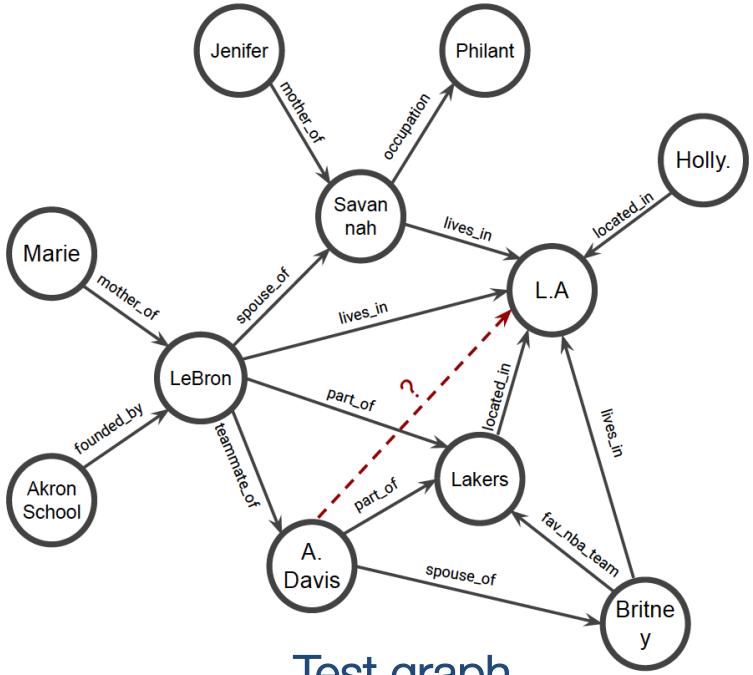
INDUCTIVE LEARNING ON KNOWLEDGE GRAPHS

Komal K. Teru, Etienne Denis, William L. Hamilton. *Inductive Relation Prediction by Subgraph Reasoning".* (Under Review). 2020.

Inductive knowledge graph completion

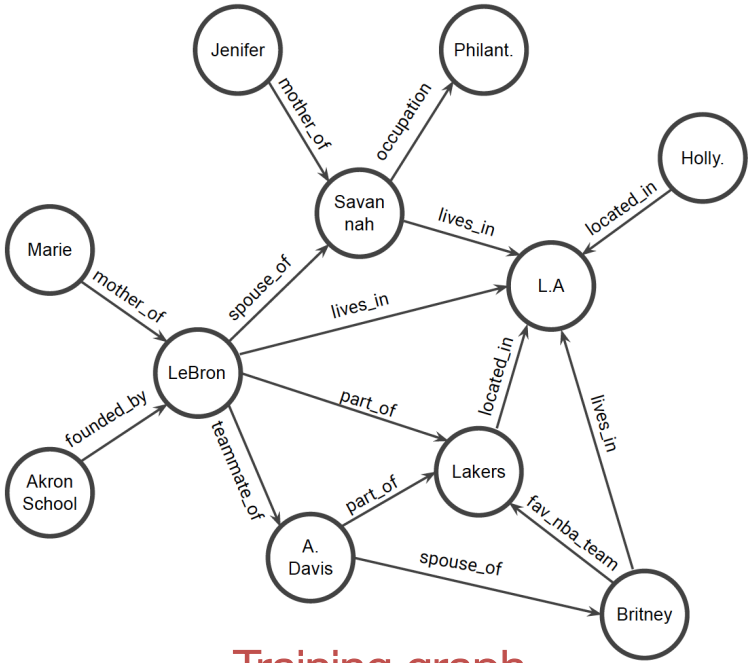


Training graph

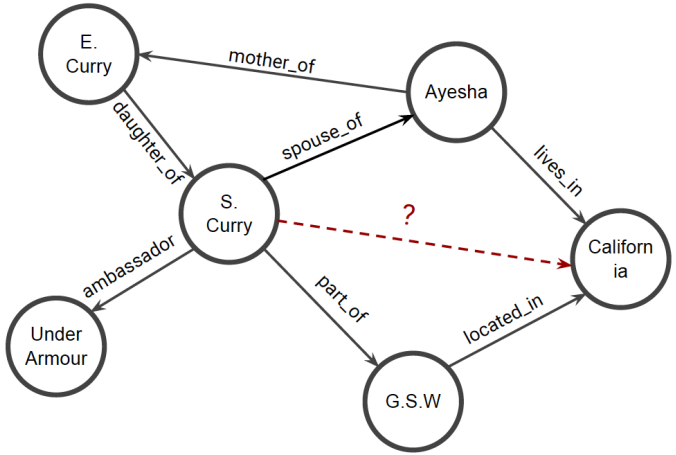


Test graph

Inductive knowledge graph completion



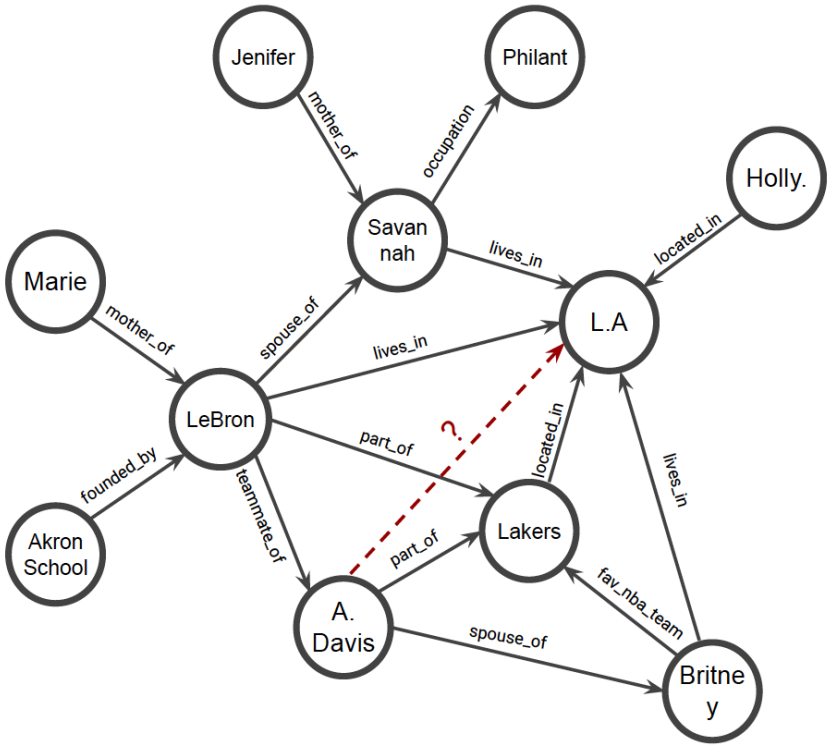
Training graph



Test graph

Embeddings vs. rules

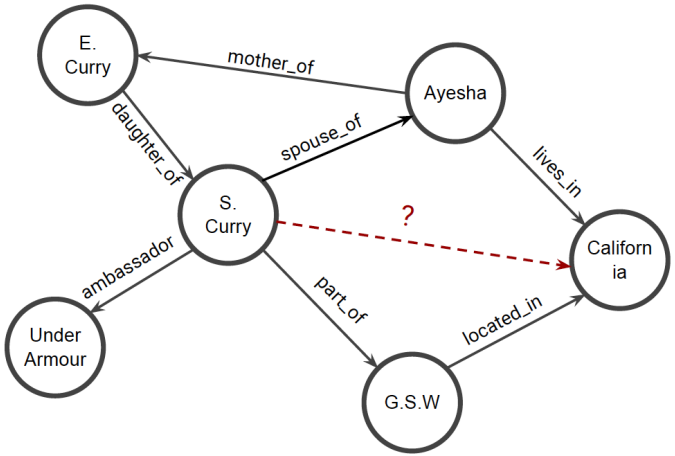
- Embeddings:
  - Neighborhood-centric
  - Exploit homophily
  - Entity-dependent



Embeddings vs. rules

- Embeddings:
  - Neighborhood-centric
  - Exploit homophily
  - Entity dependent
- Rules:
  - Entity independent
  - Exploit relational semantics

$$(X, \texttt{lives\_in}, Y) \leftarrow$$
$$\exists Z.(X, \texttt{spouse\_of}, Z) \wedge (Z, \texttt{lives\_in}, Y)$$

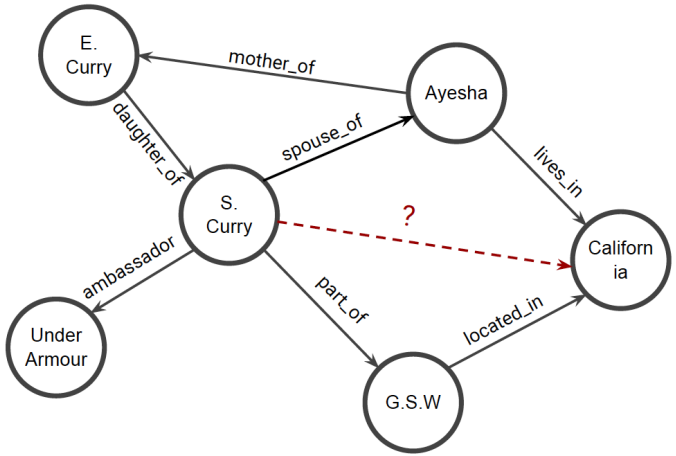


Challenges for inductive learning on KGs

- Neural rule induction methods outperformed by statistical approaches

- No features on nodes

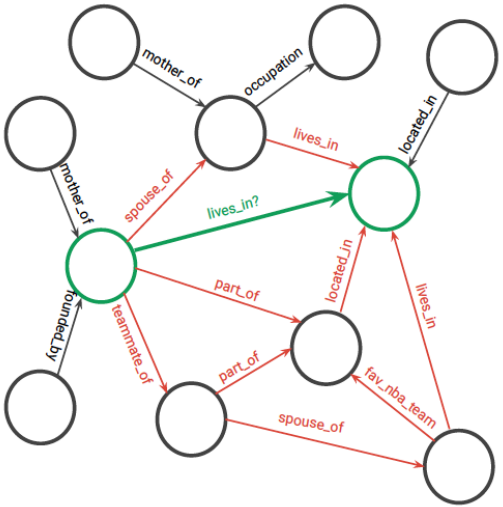
$$(X, \texttt{lives\_in}, Y) \leftarrow$$
$$\exists Z.(X, \texttt{spouse\_of}, Z) \wedge (Z, \texttt{lives\_in}, Y)$$



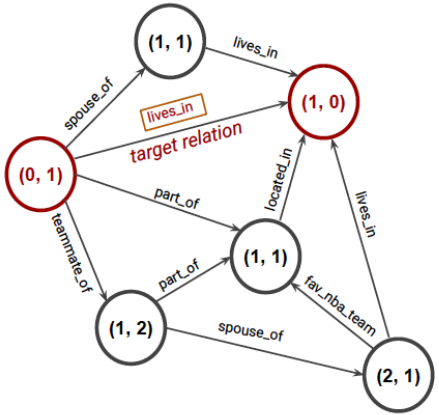
GraIL: A new approach using GNNs

- Idea 1: Apply graph neural networks (GNNs) on the subgraphs surrounding candidate relations.
- Idea 2: Avoid explicit rule induction.
- Idea 3: Ensure model is expressive enough to capture logical rules.

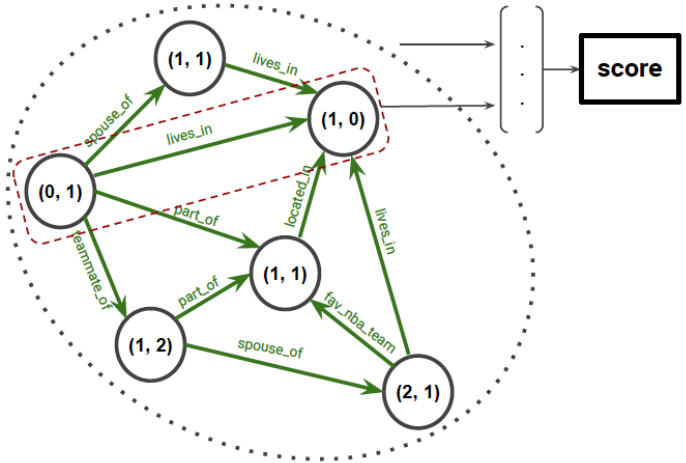
GraIL: Relation prediction via subgraph reasoning



1. Extract subgraph around candidate relation



2. Assign structural labels to nodes



3. Run GNN on the extracted subgraph

GNN architecture

Neural message-passing approach

$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right),$$

$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right)$$

GNN architecture

Neural message-passing approach

$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right),$$

$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right)$$

$$\mathbf{a}_t^k = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_t st}^k \mathbf{W}_r^k \mathbf{h}_s^{k-1}$$

Learn a relation-specific transformation matrix

Use attention to weigh information coming from different neighbors

Separately aggregate across different types of relations

GNN architecture

Neural message-passing approach

$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right),$$

$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right)$$

Information aggregated from the neighborhood

$$\mathbf{h}_t^k = \text{ReLU}\left(\mathbf{W}_{self}^k \mathbf{h}_t^{k-1} + \mathbf{a}_t^k\right)$$

Information from the nodes embedding at the previous layer

GraIL can learn logical rules

Theorem (Informally): GraIL can learn any logical rule of the form:

$$r_t(X, Y) \leftarrow \exists Z_1, ..., Z_k . r_1(X, Z_1) \wedge r_2(Z_1, Z_2) \wedge ... \wedge r_k(Z_{k-1}, Y)$$

Example of such a rule:

$$(X, \texttt{lives\_in}, Y) \leftarrow \exists Z . (X, \texttt{spouse\_of}, Z) \wedge (Z, \texttt{lives\_in}, Y)$$

These "path-based" rules are the foundation of most state-of-the-art rule induction systems.

State-of-the-art inductive performance

- Constructed inductive versions of three standard benchmarks.
- Sampled mutually exclusive subgraphs of varying sizes
- Tested four inductive datasets per each benchmark.

Table: AUC-PR results on inductive relation prediction

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	**94.32**	**94.18**	**85.80**	**92.72**	**84.69**	**90.57**	**91.68**	**94.46**	**86.05**	**92.62**	**93.34**	**87.50**

State-of-the-art inductive performance

- Compared against state-of-the-art neural rule induction methods
- Also compared against the best statistical induction approach.

Table: AUC-PR results on inductive relation prediction

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	**94.32**	**94.18**	**85.80**	**92.72**	**84.69**	**90.57**	**91.68**	**94.46**	**86.05**	**92.62**	**93.34**	**87.50**

State-of-the-art inductive performance

- Key finding: GraIL outperforms all previous approaches on all datasets (analgous results for hits@k)

Table: AUC-PR results on inductive relation prediction

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	**94.32**	**94.18**	**85.80**	**92.72**	**84.69**	**90.57**	**91.68**	**94.46**	**86.05**	**92.62**	**93.34**	**87.50**

Ensembling in the transductive setting

Each entry is a pair-wise ensemble of two methods

Table: Ensemble AUC-PR results on WN18RR

	TransE	DistMult	ComplEx	RotatE	GraIL
T	93.73	93.12	92.45	93.70	94.30
D		93.08	93.12	93.16	95.04
C			92.45	92.46	94.78
R				93.55	94.28
G					90.91

Ensembling in the transductive setting

Each entry is a pair-wise ensemble of two methods

GraIL has the lowest performance on its own

Table: Ensemble AUC-PR results on WN18RR

	TransE	DistMult	ComplEx	RotatE	GraIL
T	93.73	93.12	92.45	93.70	94.30
D		93.08	93.12	93.16	95.04
C			92.45	92.46	94.78
R				93.55	94.28
G					90.91

Ensembling in the transductive setting

Each entry is a pair-wise ensemble of two methods

GraIL has the lowest performance on its own…

But ensembling with GraIL leads to the best performance

Table: Ensemble AUC-PR results on WN18RR

	TransE	DistMult	ComplEx	RotatE	GraIL
T	93.73	93.12	92.45	93.70	94.30
D		93.08	93.12	93.16	95.04
C			92.45	92.46	94.78
R				93.55	94.28
G					90.91

Architecture details are important!

Naïve subgraph extraction causes severe overfitting

Our node labelling and attention schemes are crucial for the theory and for strong performance.

Table: Ablation study AUC-PR results

	FB (v3)	NELL (v3)
GraIL	**91.68**	**93.34**
GraIL w/o enclosing subgraph	84.25	85.89
GraIL w/o node labeling scheme	82.07	84.46
GraIL w/o attention in GNN	90.27	87.30

Part II

FEW SHOT LEARNING ON GRAPHS

Part II

FEW SHOT LEARNING ON GRAPHS

Avishek Joey Bose, Ankit Jain, Piero Molino, William L Hamilton. *Meta-Graph: Few Shot Link Prediction via Meta Learning.* (Under Review). 2020.

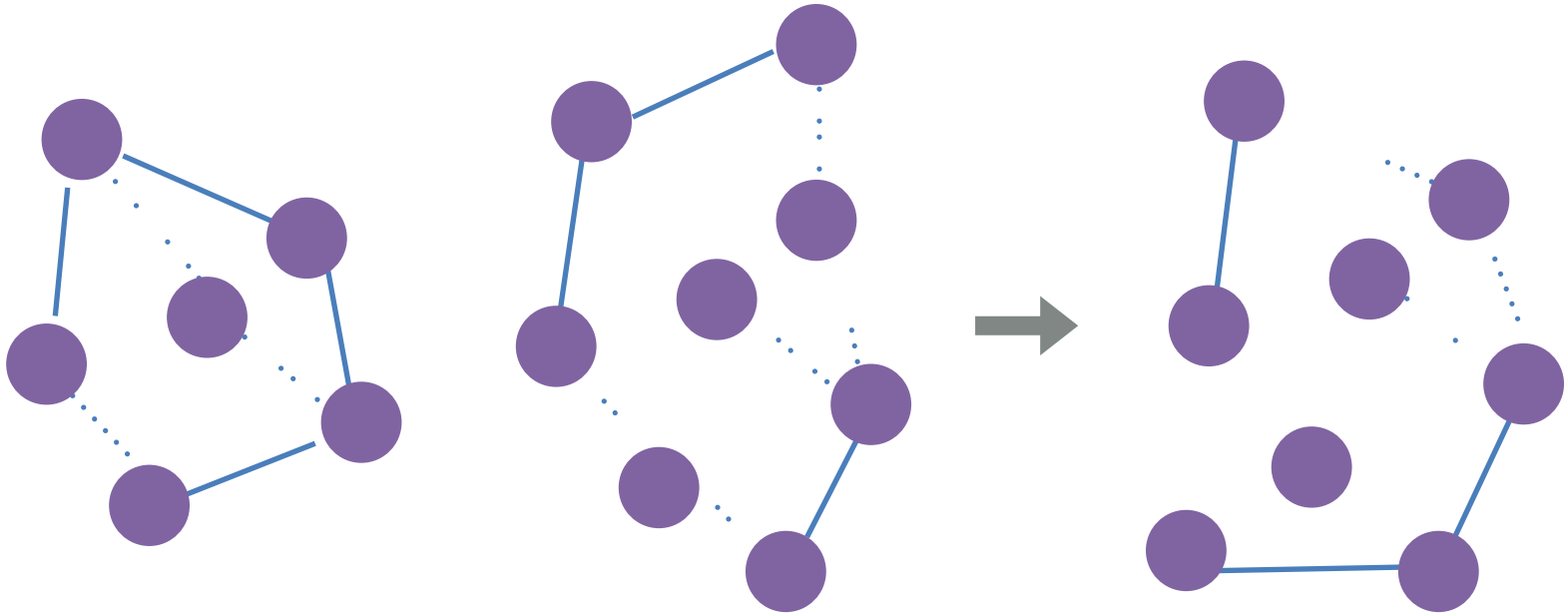
Uber Ai Labs

Few shot relation prediction

- In many applications we have:
  - Multiple graphs sampled from related domains
  - Very few training edges
  - Desire for fast adaption

- Examples:
  - Different locales for recommender systems
  - High-throughput biological applications

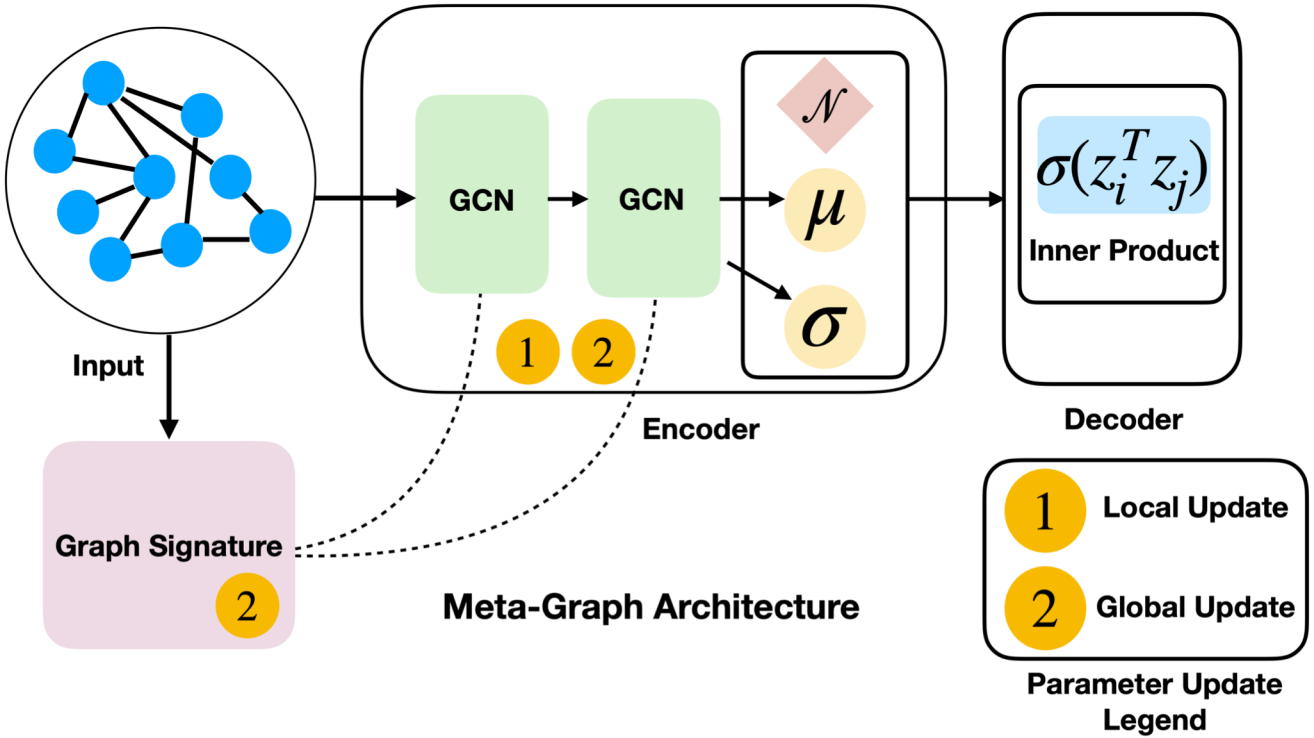
Few shot relation prediction



Meta learning on graphs

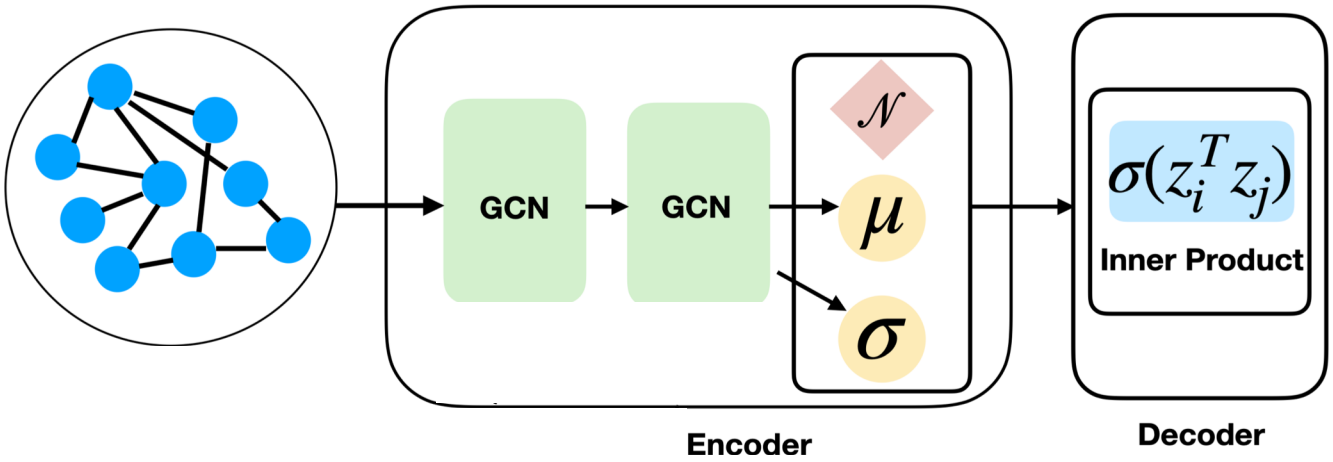
- Rather than learning on a single graph…
- We want to learn about the **"meta"** properties of graphs from a domain, so that we can
  1. Leverage information from multiple graphs
  2. Overcome small amounts of training data
  3. Achieve fast adaption

Meta-Graph Overview



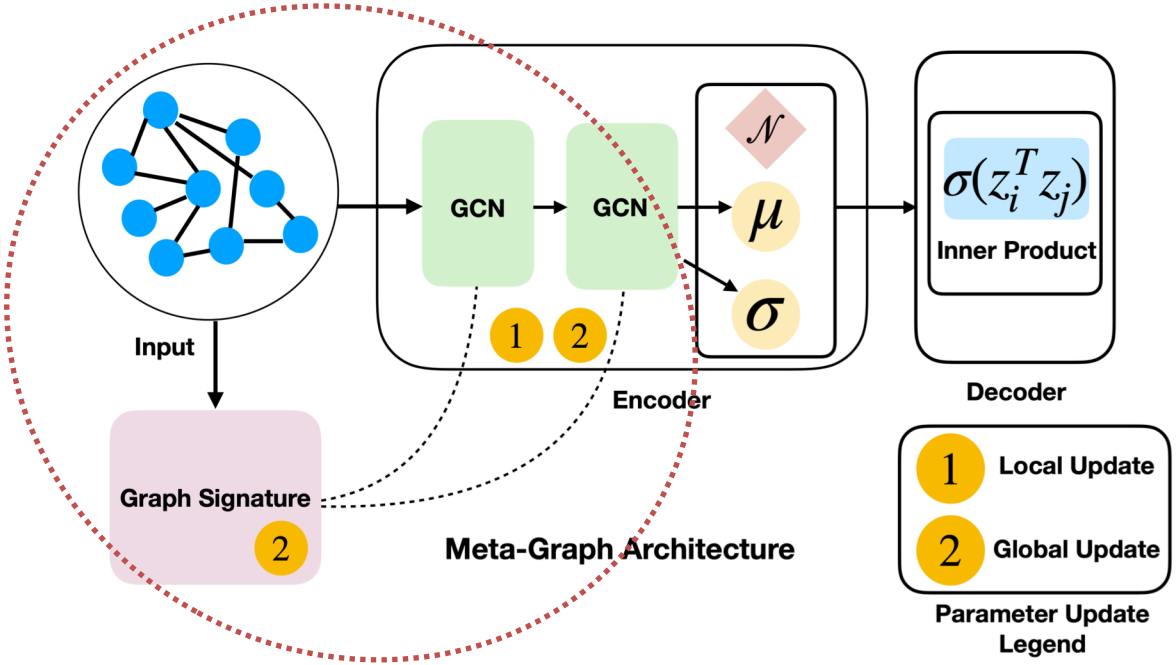
Local generative model

- Model each individual graph using a variational graph autoencoder (Kipf & Welling 2016):



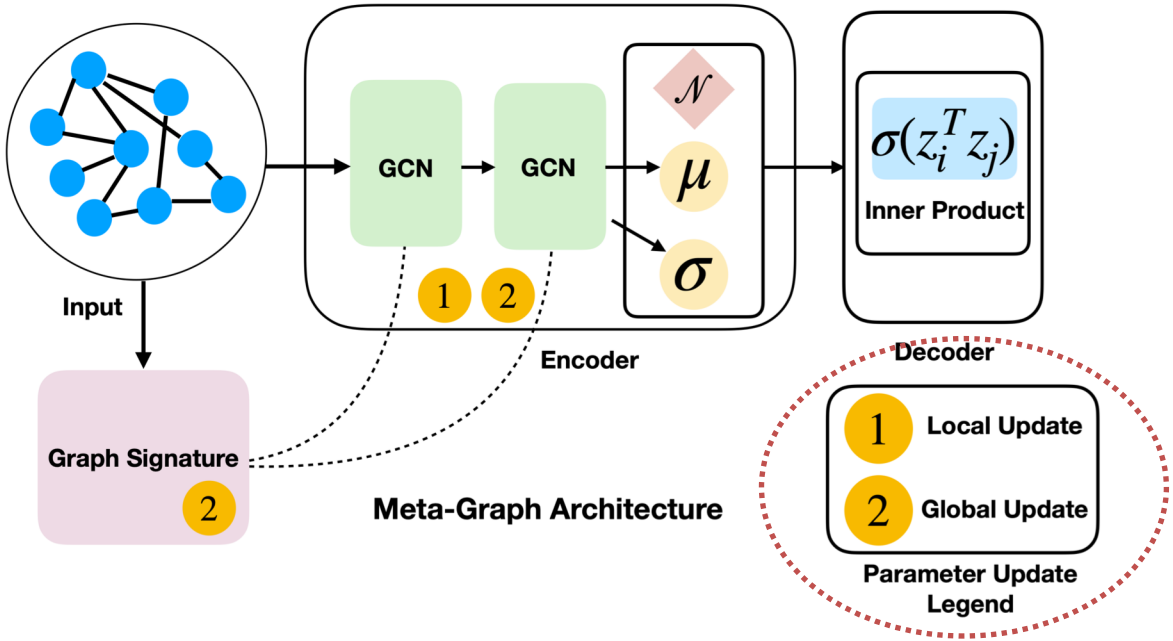
Learning a graph signature

Graph signature GNN maps the training graph to a parameter initialization for the local link prediction model



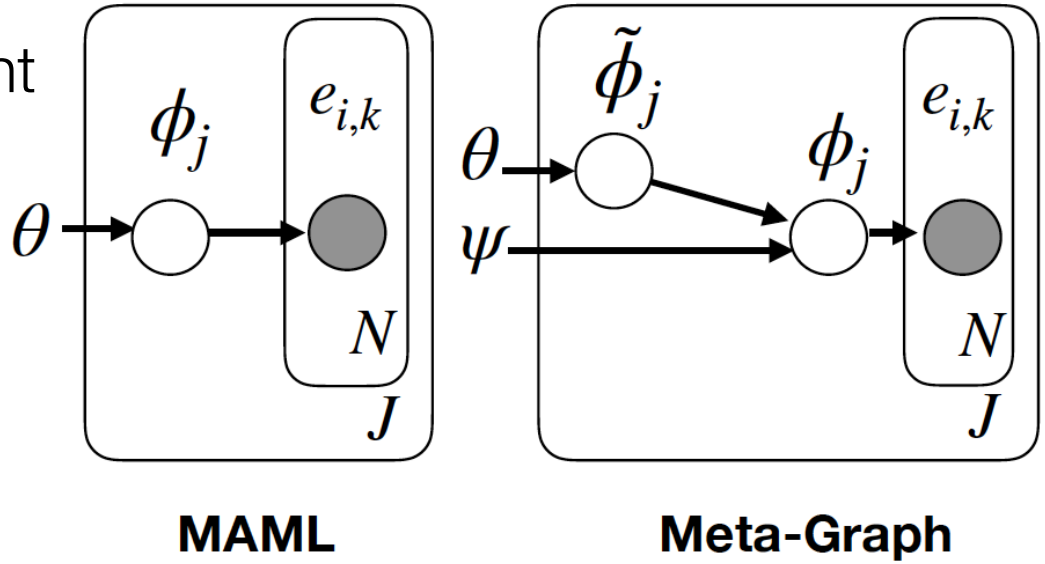
Leveraging second-order gradients

Graph signature GNN is updated via second-order gradient descent to optimize for few-shot performance



Meta-Graph and MAML

- Meta-Graph without the graph signature is equivalent to MAML on graphs.
- The graph signature allows us to condition our GNN initialization on the input.



Improved few shot learning

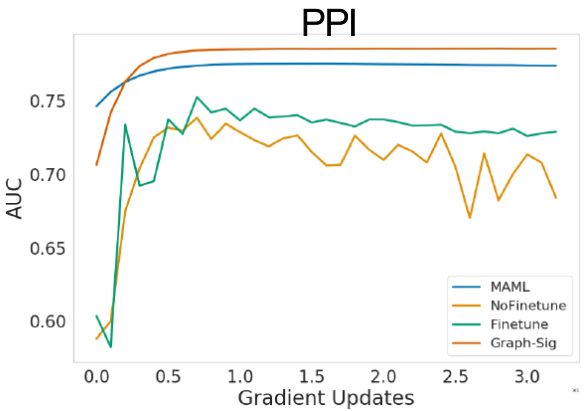
24 protein-protein interaction networks from different tissues

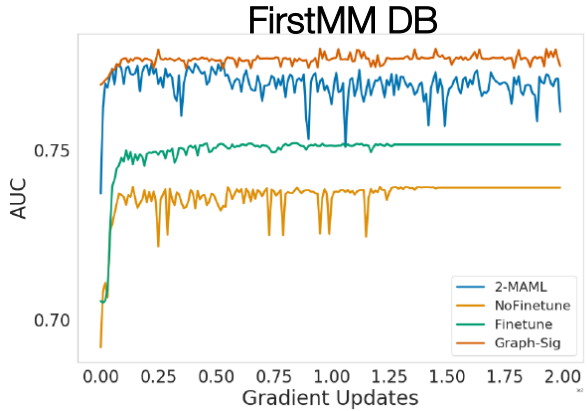
41 point-cloud graphs from different robotics experiments

72 ego citation networks

Table: AUC results on 5-shot learning on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.824**	**0.847**	**0.773**	**0.767**	0.737	**0.620**	0.585	**0.732**
MAML	0.728	0.809	0.804	0.763	0.750	**0.750**	0.500	0.504	0.500
No Finetune	0.600	0.697	0.717	0.708	0.680	0.709	0.500	0.500	0.500
Finetune	0.582	0.727	0.774	0.705	0.695	0.704	0.608	**0.675**	0.713



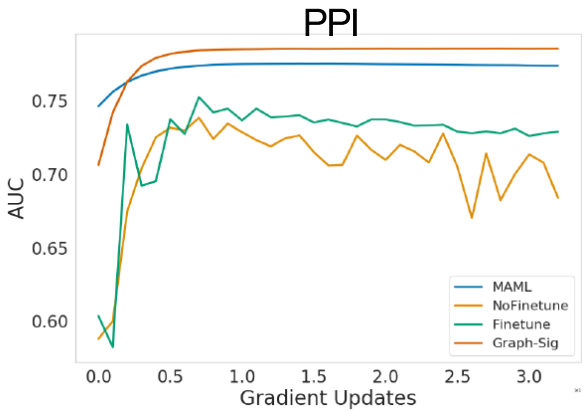


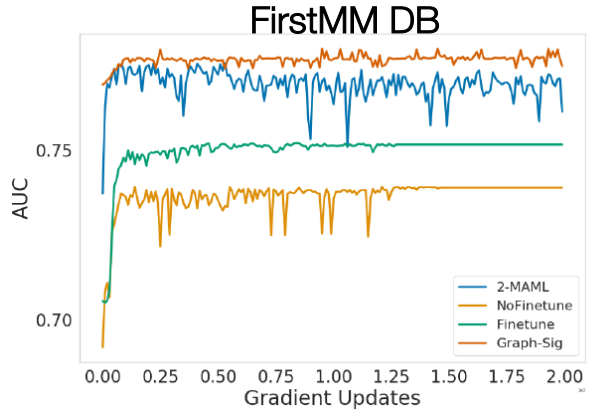
Improved few shot learning

Varied the % of edges used for training the models

Table: AUC results on 5-shot learning on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.824**	**0.847**	**0.773**	**0.767**	0.737	**0.620**	0.585	**0.732**
MAML	0.728	0.809	0.804	0.763	0.750	**0.750**	0.500	0.504	0.500
No Finetune	0.600	0.697	0.717	0.708	0.680	0.709	0.500	0.500	0.500
Finetune	0.582	0.727	0.774	0.705	0.695	0.704	0.608	**0.675**	0.713





Improved few shot learning

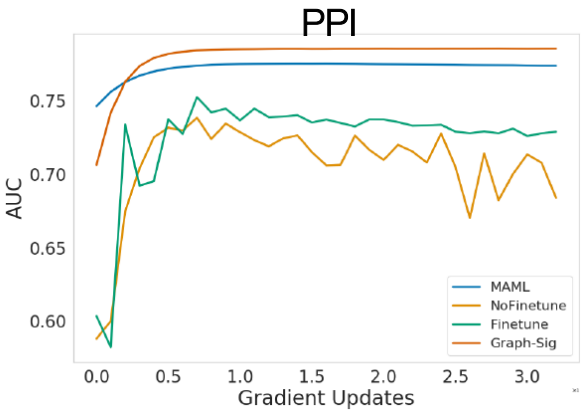
Meta-Graph with and w/o signature function.

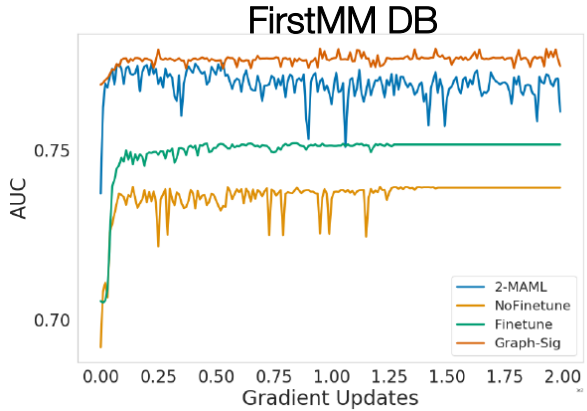
Inductive baseline with no finetuning

Naïve fine-tuning baseline.

Table: AUC results on 5-shot learning on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.824**	**0.847**	**0.773**	**0.767**	0.737	**0.620**	0.585	**0.732**
MAML	0.728	0.809	0.804	0.763	0.750	**0.750**	0.500	0.504	0.500
No Finetune	0.600	0.697	0.717	0.708	0.680	0.709	0.500	0.500	0.500
Finetune	0.582	0.727	0.774	0.705	0.695	0.704	0.608	**0.675**	0.713





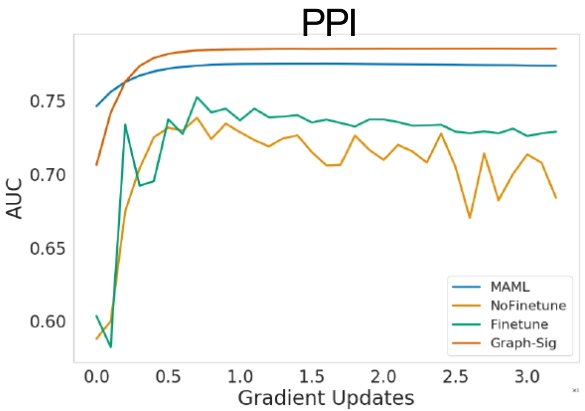
Improved few shot learning

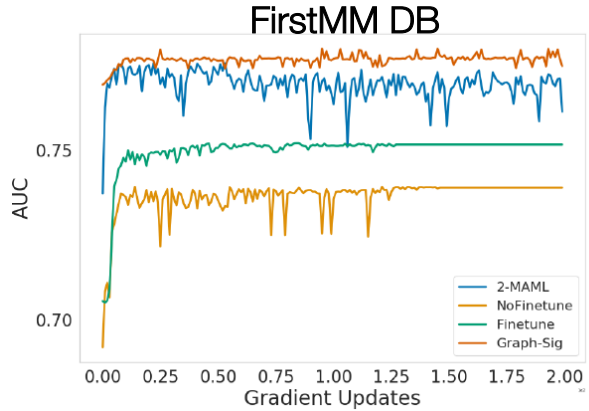
Meta-Graph consistently outperforms baselines.

Achieves very fast convergence through the signature function.

Table: AUC results on 5-shot learning on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.824**	**0.847**	**0.773**	**0.767**	0.737	**0.620**	0.585	**0.732**
MAML	0.728	0.809	0.804	0.763	0.750	**0.750**	0.500	0.504	0.500
No Finetune	0.600	0.697	0.717	0.708	0.680	0.709	0.500	0.500	0.500
Finetune	0.582	0.727	0.774	0.705	0.695	0.704	0.608	**0.675**	0.713





Improved accuracy at convergence!

Table: AUC results after training to convergence on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.833**	**0.845**	**0.782**	**0.786**	0.783	**0.626**	**0.738**	**0.786**
MAML	0.770	0.815	0.828	0.776	0.782	**0.793**	0.561	0.662	0.667
Random	0.578	0.651	0.697	0.742	0.732	0.720	0.500	0.500	0.500
No Finetune	0.738	0.786	0.801	0.740	0.710	0.734	0.548	0.621	0.673
Finetune	0.752	0.801	0.821	0.752	0.735	0.723	0.623	0.691	0.723
Adamic	0.540	0.623	0.697	0.504	0.519	0.544	0.515	0.549	0.597
Deepwalk	0.664	0.673	0.694	0.487	0.473	0.510	0.602	0.638	0.672

Improved accuracy at convergence!

Table: AUC results after training to convergence on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.833**	**0.845**	**0.782**	**0.786**	0.783	**0.626**	**0.738**	**0.786**
MAML	0.770	0.815	0.828	0.776	0.782	**0.793**	0.561	0.662	0.667
Random	0.578	0.651	0.697	0.742	0.732	0.720	0.500	0.500	0.500
No Finetune	0.738	0.786	0.801	0.740	0.710	0.734	0.548	0.621	0.673
Finetune	0.752	0.801	0.821	0.752	0.735	0.723	0.623	0.691	0.723
Adamic	0.540	0.623	0.697	0.504	0.519	0.544	0.515	0.549	0.597
Deepwalk	0.664	0.673	0.694	0.487	0.473	0.510	0.602	0.638	0.672

Additional non-GNN baselines from the literate.

Improved accuracy at convergence!

Table: AUC results after training to convergence on test graphs

	PPI			FirstMM DB			Ego-AMINER		
Edges	10%	20%	30%	10%	20%	30%	10%	20%	30%
Meta-Graph	**0.795**	**0.833**	**0.845**	**0.782**	**0.786**	0.783	**0.626**	**0.738**	**0.786**
MAML	0.770	0.815	0.828	0.776	0.782	**0.793**	0.561	0.662	0.667
Random	0.578	0.651	0.697	0.742	0.732	0.720	0.500	0.500	0.500
No Finetune	0.738	0.786	0.801	0.740	0.710	0.734	0.548	0.621	0.673
Finetune	0.752	0.801	0.821	0.752	0.735	0.723	0.623	0.691	0.723
Adamic	0.540	0.623	0.697	0.504	0.519	0.544	0.515	0.549	0.597
Deepwalk	0.664	0.673	0.694	0.487	0.473	0.510	0.602	0.638	0.672

Meta-graph again consistently improves compared to all the baselines

Part III

EVALUATING LOGICAL GENERALIZATION IN GNNS

Koustuv Sinha, Shagun Sodhani, Joelle Pinear, William L Hamilton. *Evaluating Logical Generalization in Graph Neural Networks.* (Under Review). 2020.

facebook research

When do we expect GNNs to generalize?

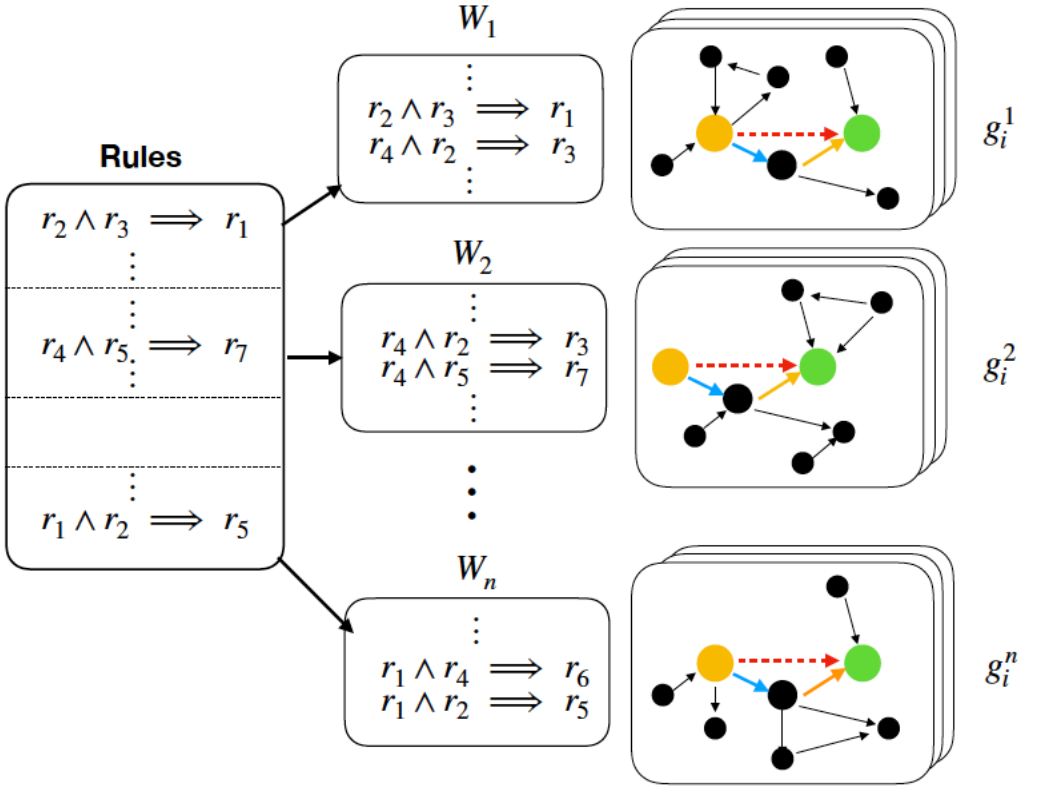
- Inductive and few-shot settings presuppose similarity between train/test graphs.

- How true is this in practice?

- How does the distribution of train/test graphs impact performance?

GraphLog: A synthetic benchmark

- **Idea:** Synthetically create KGs with different properties/overlap.
- Evaluate GNNs in multi-task, few-shot, and continual learning setups.



Some findings so far

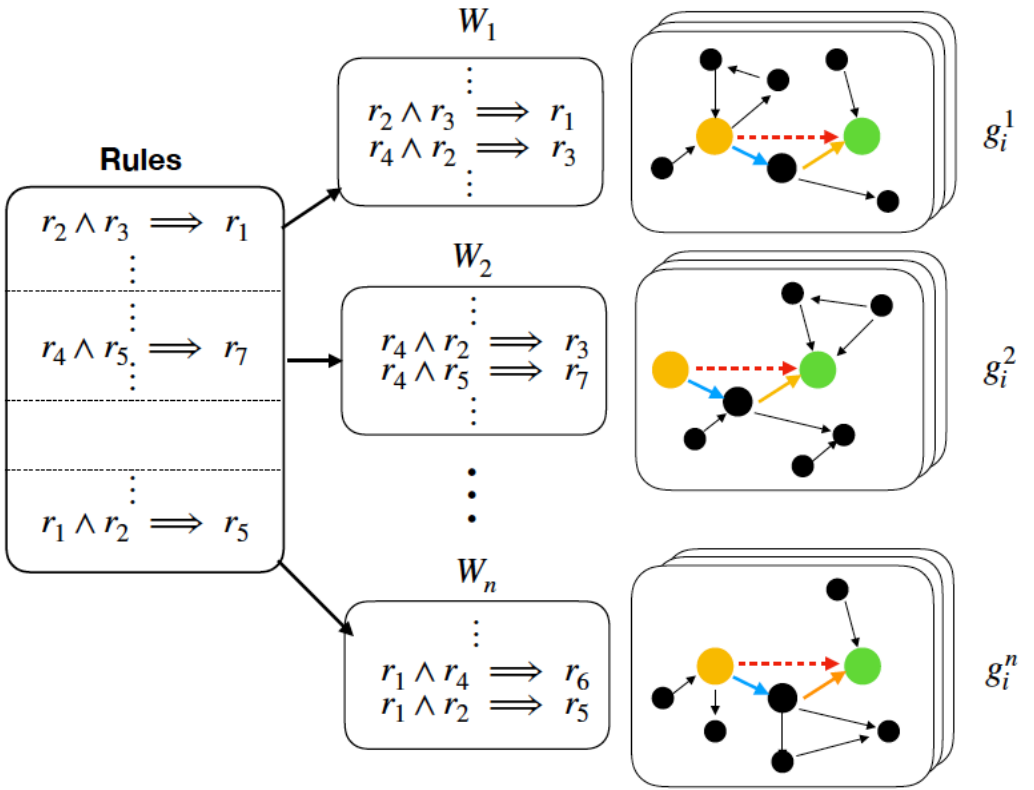
1. Enforcing modularity in GNN architectures improves generalization.

2. Diversity during training improves generalization.

3. All models exhibit catastrophic forgetting.

GraphLog: A synthetic benchmark

Provides a flexible and extendable testbed to probe the limits of GNNs for logical reasoning.



Acknowledgements

- Students
  - Joey Bose (PhD)
  - Koustuv Sinha (PhD)
  - Zichao Yan (PhD)
  - Priyesh Vijayan (PhD)
  - Devendra Singh (PhD)
  - Dora Jambor (MSc)
  - Paul Wu (MSc)
  - Komal Kumar (MSc)
  - Jin Dong (MSc)
  - Ashutosh Adhikari (Intern)
- Collaborators:
  - Joelle Pineau
  - Shagun Sodhani
  - Ankit Jain
  - Piero Molino
- Funding: