Graph Neural Networks Meet Embedding Geometry

Speaker: Rex Ying*

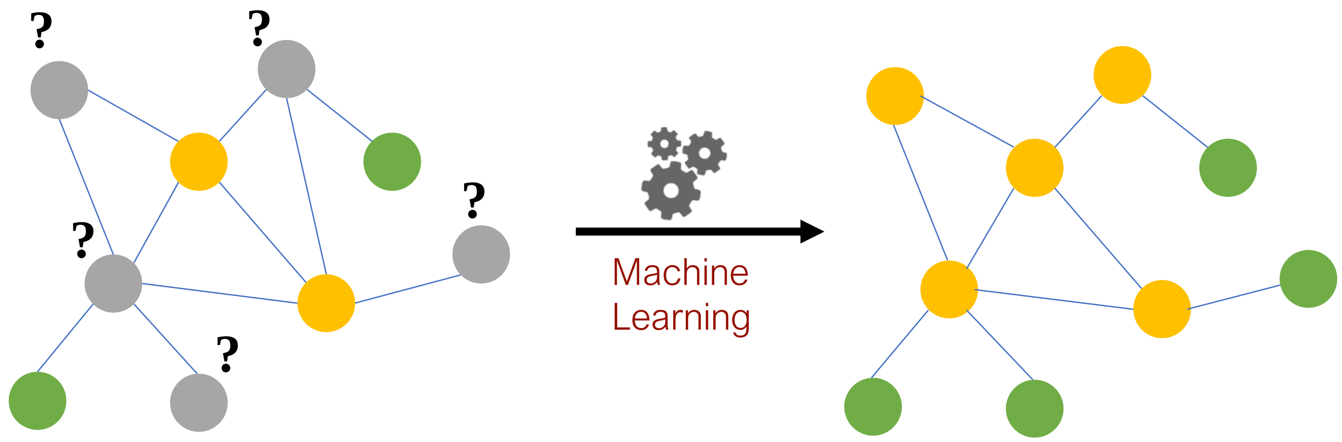
Collaborators: Andrew Wang, Ines Chami, Jiaxuan You, Zhaoyu Lou, Christopher Ré, Jure Leskovec, Siemens Inc.

SNAP, DAWN
Stanford University

Representation Learning for Graphs

- Representation Learning on graph structured data
  - Node classification; link prediction
  - Graph and subgraph-level predictions

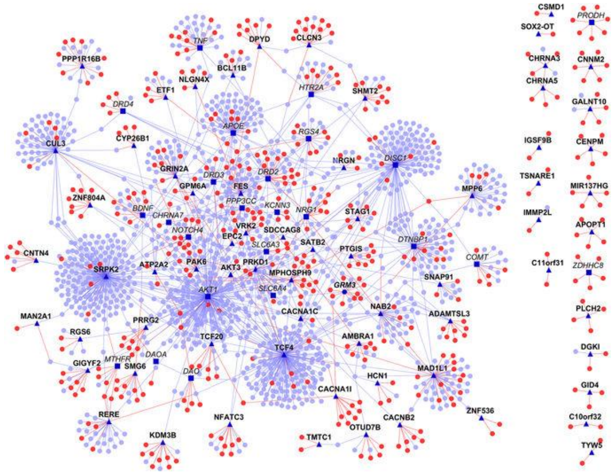
Example Node classification:



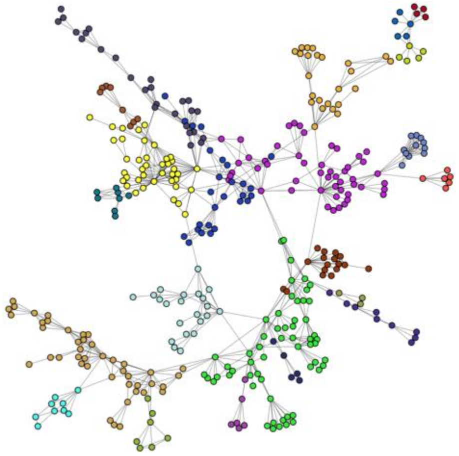
Representation Learning for Graphs

- Example applications
  - Biological networks – classify protein functions in interactome
  - Social networks – predict interactions between people
  - Molecules – predict properties of molecules and functional groups (subgraphs)

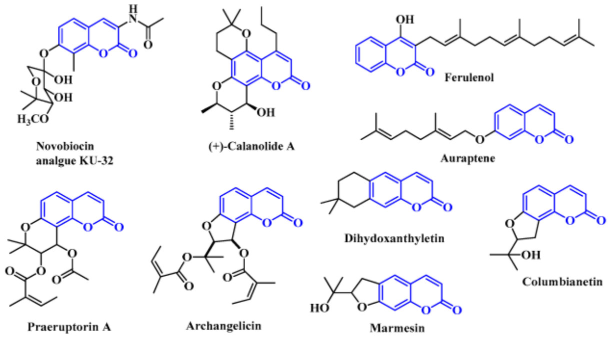
Protein (node) property prediction



Predict links in social network

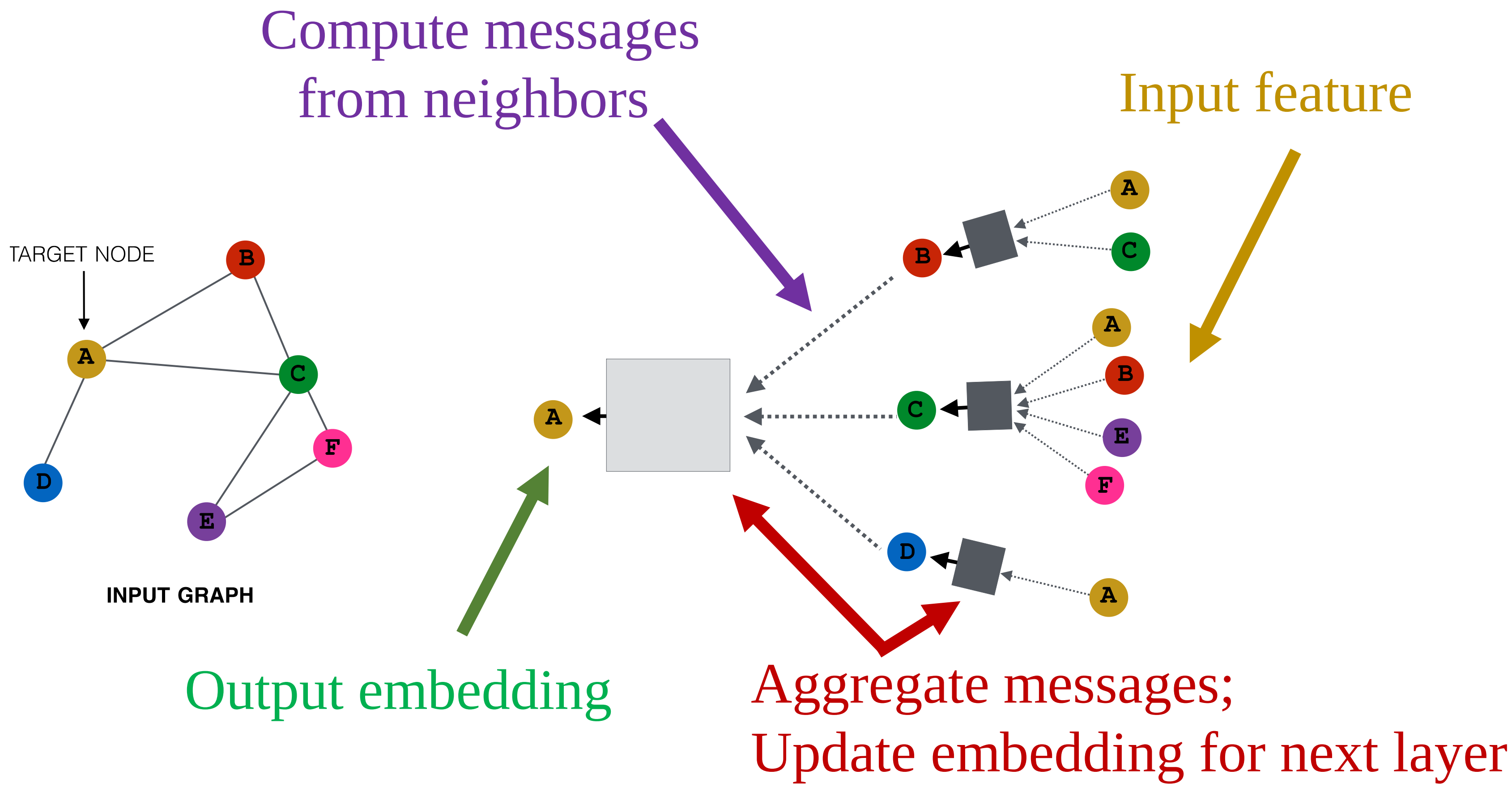


Predict molecular properties



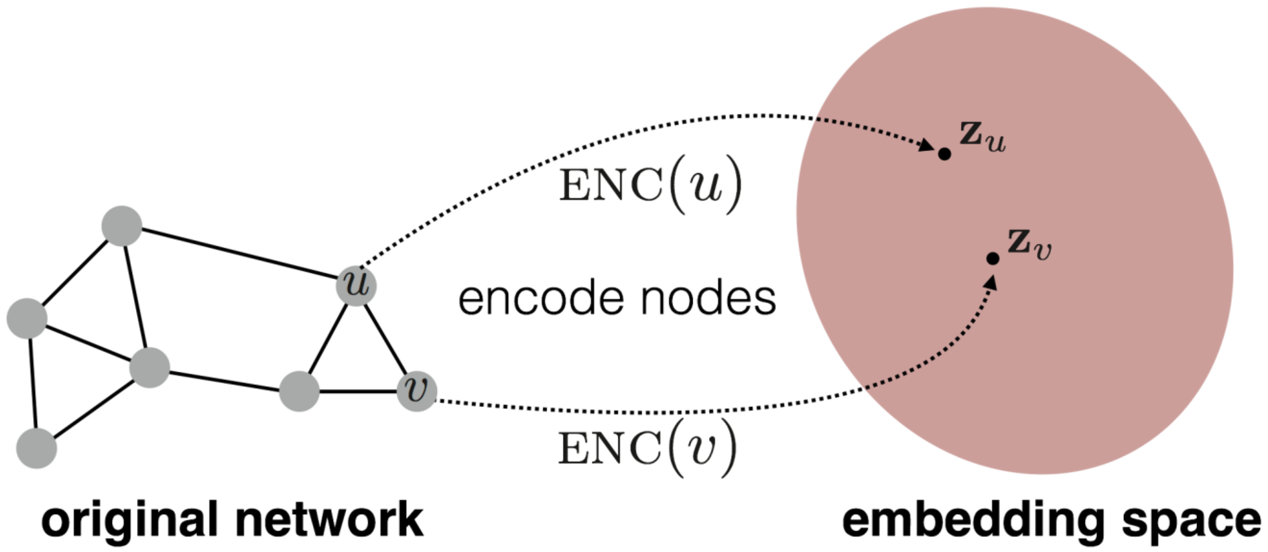
Graph Neural Networks (GNNs)

- Generate node embeddings based on local message-passing



Motivation: Leverage Embedding Geometry

- Existing **GNNs** embed nodes / links into a Euclidean vector space
- Standard classification toolkit is available for Euclidean embeddings



Motivation: Leverage Embedding Geometry

- How to learn high-quality embeddings for graphs?
  - Graph Neural Network (GNN) architecture is shown to have high expressive power (Hu *et al*. 2018).
  - **Geometry of the embedding space** is crucial in learning high-quality embeddings.

Input:
Nodes in graph

→

Learning:
Node embeddings

→

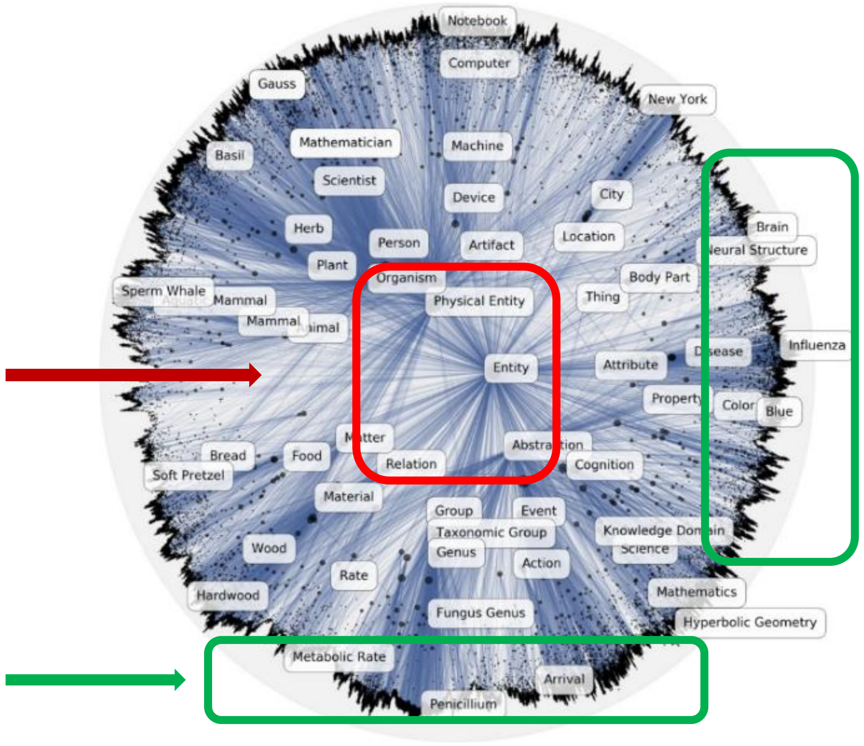
Output:
node and edge labels

Euclidean embeddings?

Outline: Embedding Geometry

- Hyperbolic Convolutional Neural Networks
  - ➢ Hyperbolic embedding geometry for hierarchy modeling

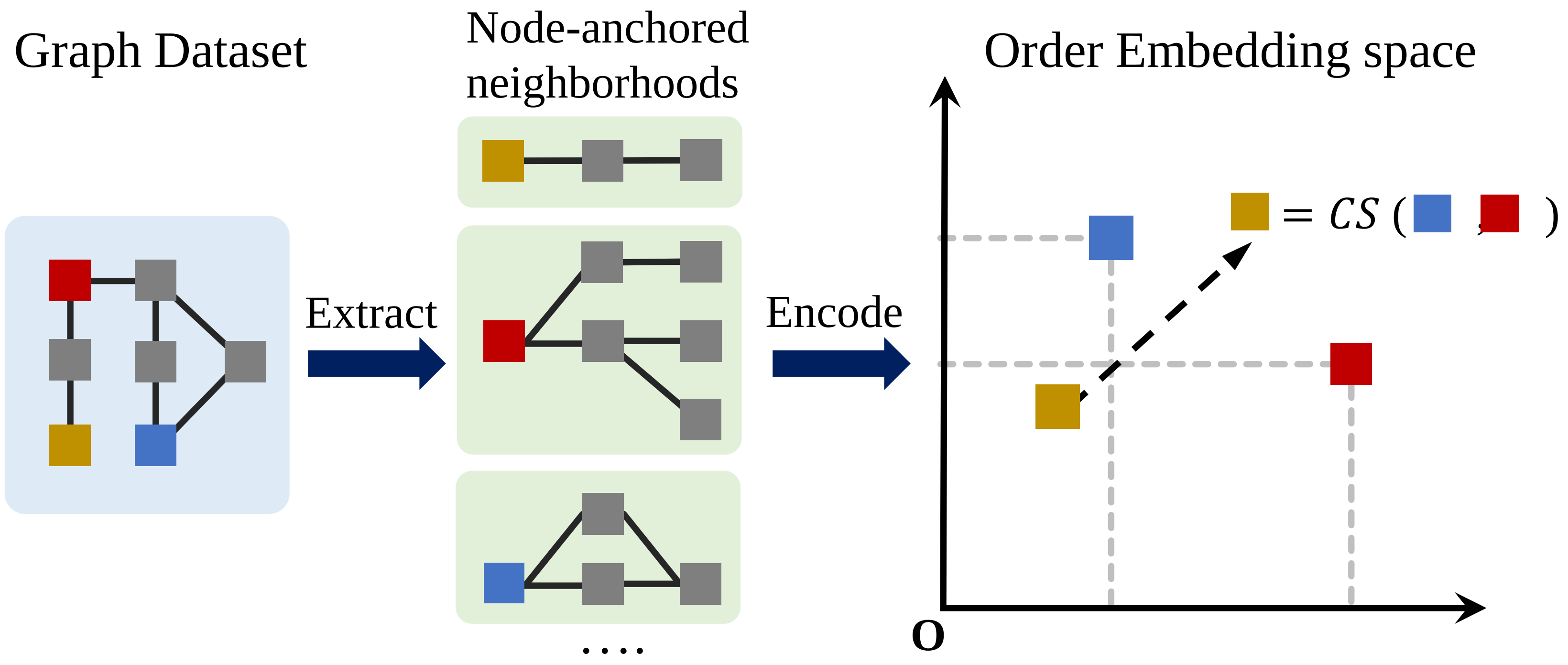
Nodes in high hierarchy



Nodes in low hierarchy

Outline: Embedding Geometry

- Neural Subgraph Matching
  - ➢ Order embedding geometry for partial ordering



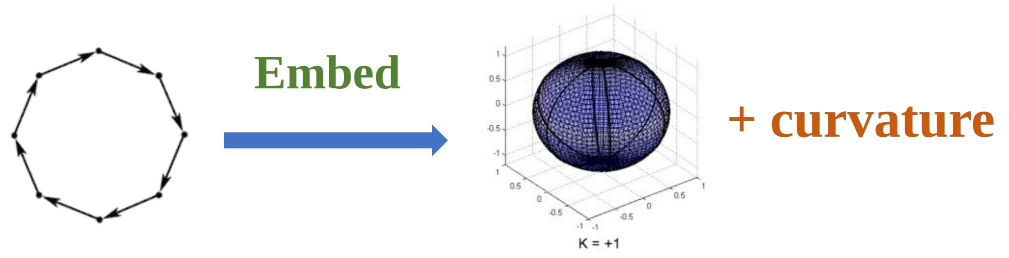
Outline: Embedding Geometry

- Hyperbolic Graph Convolutional Neural Networks
- Neural Subgraph Matching

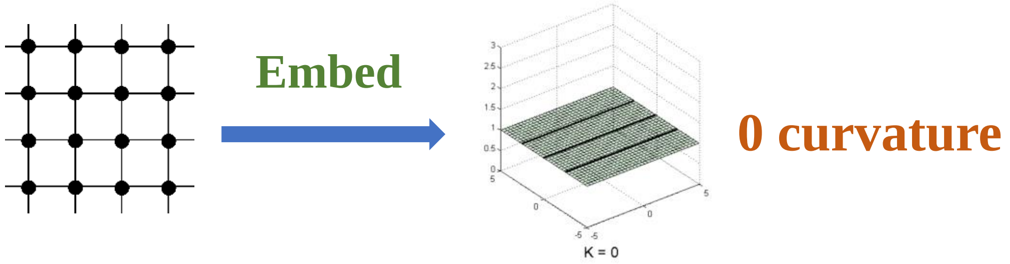
Motivation

- Space curvature: deviation from Euclidean space
  - Departure from Pythagoras theorem (positive curvature: $x^2 + y^2 < z^2$)

- Graphs with many large cycles
  - Best with **spherical** space embeddings



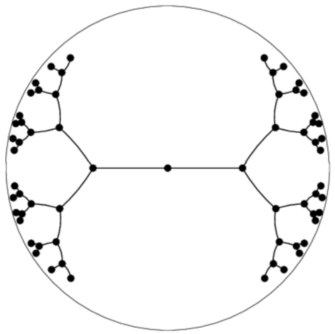
- Grid-like graphs
  - Best with **Euclidean** space embeddings



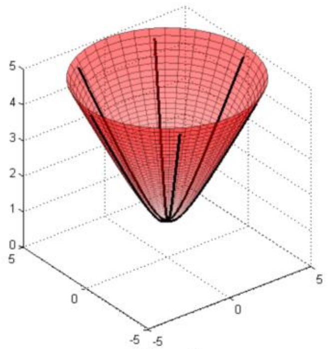
- **Hierarchical, tree-like** graphs?
  - We present **Hyperbolic Graph Convolutional Neural Networks** (NeurIPS 2019)

Motivation

- **Hierarchical, tree-like** graphs
  - Sparse graphs with low number of cycles
  - Exponential growth in number of children



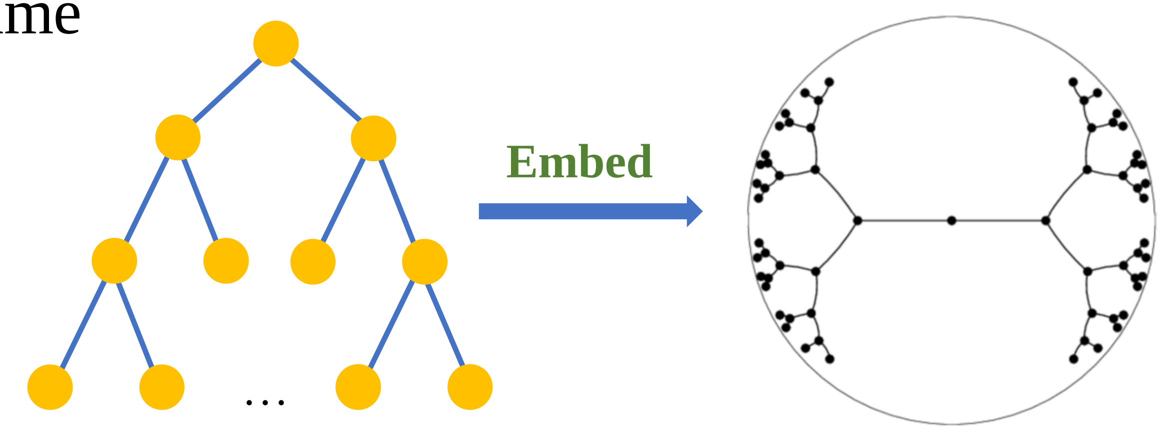
Embed



- curvature

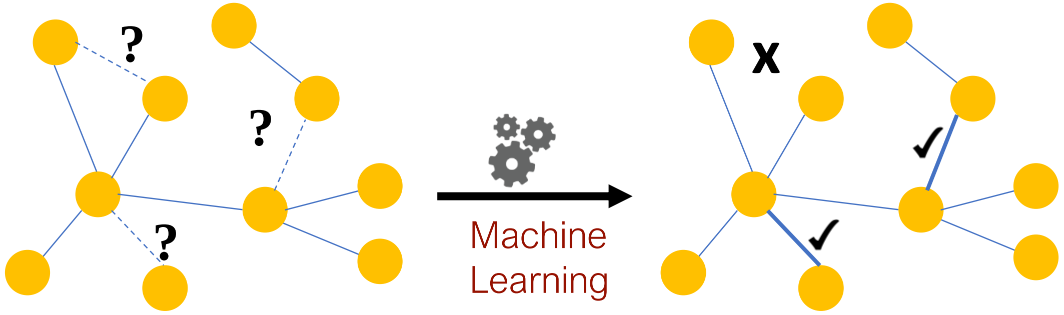
Motivation

- Suppose that we want to embed a tree
  - **Exponential** number of children
- Euclidean embedding cannot preserve shortest path metric
  - **Quadratic** growth in Euclidean volume w.r.t. radius
- Hyperbolic space
  - **Exponential** growth in hyperbolic volume
  - Problem aligns with geometry!

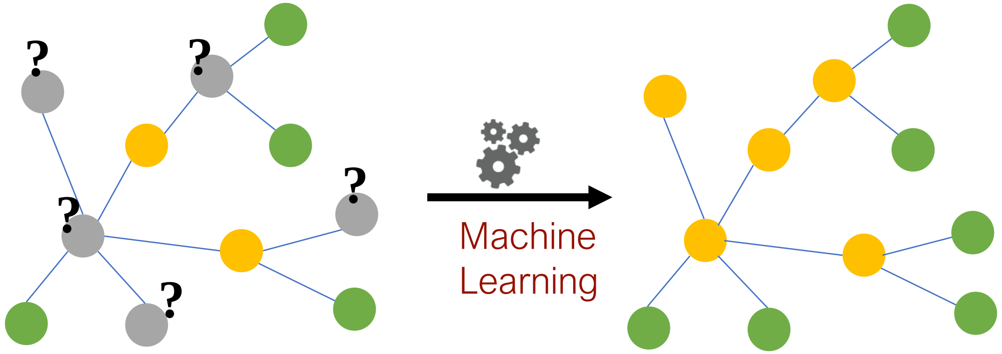


Task Setup

- Graph representation learning on **hierarchical, tree-like** graphs
  - **Link Prediction:**



  - **Node Classification:**



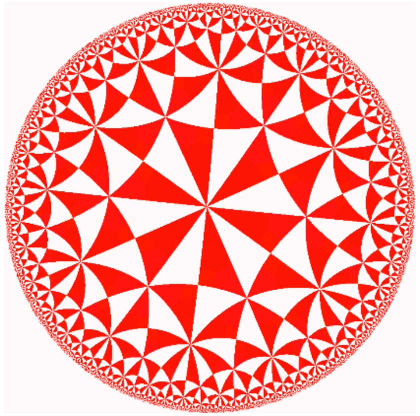
Hyperbolic GNNs

- Graph representation learning on **hierarchical** graphs
  - Link Prediction
  - Node Classification
- Graph Neural Networks (GNNs) achieve state-of-the-art in these tasks

Can we use GNN to learn hyperbolic embeddings for hierarchical graphs?

Background: Hyperbolic Space

- Poincaré Model
  - Radius proportional to $\sqrt{K}$ (inverse of curvature)
  - Open ball (exclude boundary)
  - Each triangle in the figure has the **same** area
  - Intuitive 2D visualization
- Hyperboloid Model (Lorentz Model)
  - Upper sheet of 2-sheet hyperboloid
  - Subset of Euclidean space
  - Numerically more stable, simpler formula



Poincaré Model

Hyperboloid Model

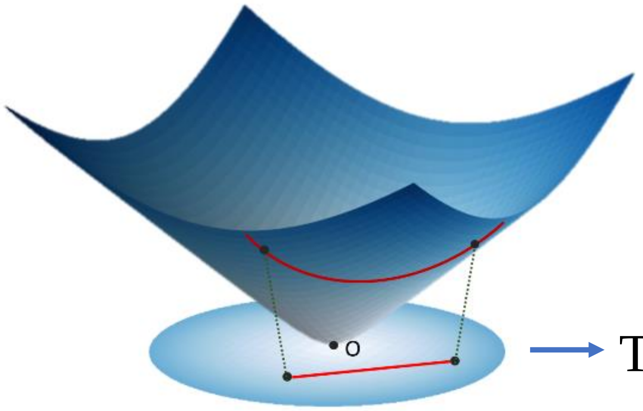
Tangent Space

- Hyperboloid model:
  - Constant Minkowski inner product $\langle .,. \rangle_{\mathcal{L}}: \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \rightarrow \mathbb{R}$

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = -x_0 y_0 + x_1 y_1 + \ldots + x_d y_d = -\frac{1}{K} \; (K > 0)$$

  - Tangent space:

$$\mathcal{T}_x \mathbb{H}^{d,K} = \{ \boldsymbol{v} \in \mathbb{R}^{d+1} : \langle \boldsymbol{v}, \boldsymbol{x} \rangle_{\mathcal{L}} = 0 \}$$

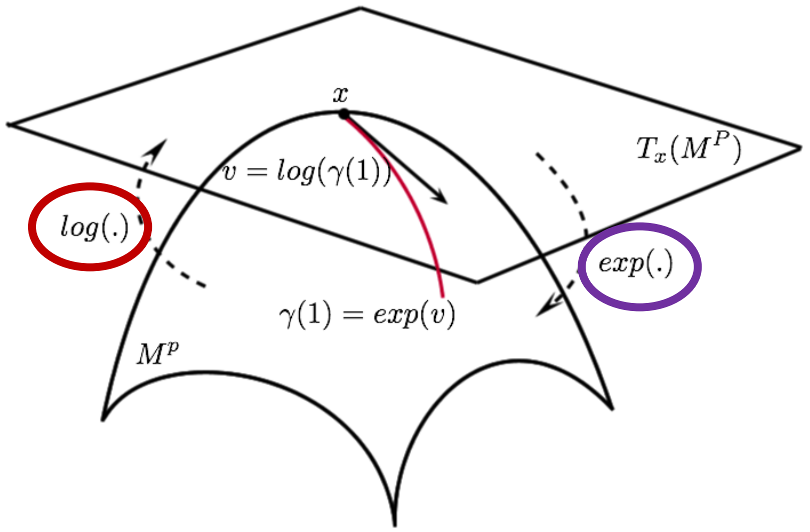


Tangent space at **north pole** $\mathcal{T}_{\mathbf{o}} \mathbb{H}^{d,K}$

d: hyperbolic space dimension
K: negative inverse of curvature

Tangent Map

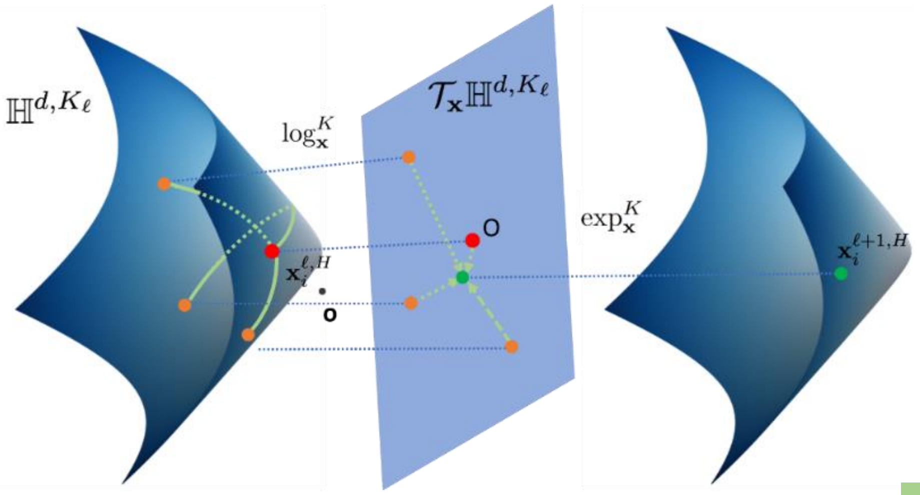
- **Exponential map**: from tangent space (Euclidean) to manifold
- **Logarithmic map**: inverse operation of exponential map
- See paper for derivation and formula



Method Overview

- Derive the core operations of GNN in the hyperbolic space
- Introduce a hyperbolic attention-based aggregation scheme that captures node hierarchies
- Use hyperbolic spaces of different trainable curvatures at different layers of the GNN

At every layer:



Transformation Aggregation Update

How is each implemented?

Hyperbolic Graph Convolutional Neural Networks, NeurIPS 2019

Input Transformation

- Input features are mapped to hyperbolic space via exp map
- Assuming the features lie on the tangent space at north pole

$$\boxed{\boldsymbol{x}^{0,H}} = \exp_o^K\left(\left(0, \boxed{\boldsymbol{x}^{0,E}}\right)\right)$$

Hyperbolic embedding (H)
at layer **0**

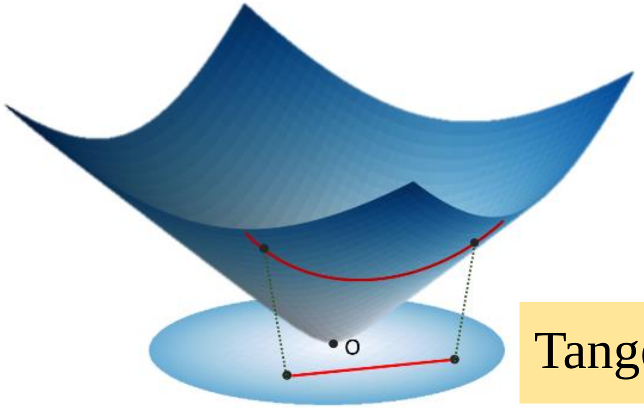
Tangent space (Euclidean)
embedding projection (E)
at layer **0**

Input Transformation

- Input features are mapped to hyperbolic space via exp map
- Assuming the features lie on tangent space at north pole

$$\boldsymbol{x}^{0,H} = \boxed{\exp_o^K}\left(\left(0, \boldsymbol{x}^{0,E}\right)\right)$$

Exponential map from
tangent space at north pole o to
hyperbolic space with curvature $-1/K$



Tangent space at **north pole**

Hyperbolic Graph Convolutional Neural Networks, NeurIPS 2019

GNN Message Computation - Transformation

- Linear transformation in tangent space at north pole (for any layer l)

$$W \boxed{\otimes^K} \boldsymbol{x}^{l,H} := \exp_o^K(W \boxed{\log_o^K}(\boldsymbol{x}^{l,H}))$$

Hyperbolic transformation notation

Tangent space projection

- W : **Euclidean** parameters
- See paper for bias addition

Message Aggregation

- Attention value computed at tangent space of north pole

$$\boxed{\omega_{ij}} = \text{Softmax}_{j\in\mathcal{N}(i)}\{\text{MLP}\left(\boxed{\log_o^K\left(\boldsymbol{x}_i^H\right) \parallel \log_o^K\left(\boldsymbol{x}_j^H\right)}\right)\}$$

Attention of node i to node j

Concat tangent space projections

- Allows model to compute attention according to the node's hierarchy in the entire graph
- Attention-based aggregation

$$AGG\left(x^H\right)_i \coloneqq \exp_{\boldsymbol{x}_i^H}^K\left(\sum_{j\in\mathcal{N}(i)} \omega_{ij} \log_{\boldsymbol{x}_i^H}^K\left(\boldsymbol{x}_j^H\right)\right)$$

Embedding Update

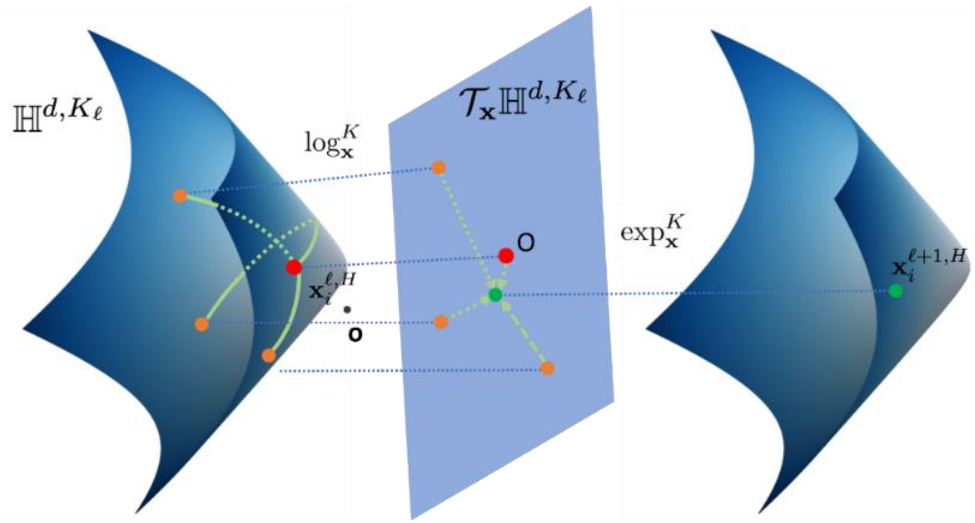
- Update $\boldsymbol{x}^{l,H}$ with messages to get next layer embedding: $\boldsymbol{x}^{l+1,H}$
- Apply non-linear activation σ to tangent space projection: $\log_{\boldsymbol{o}}^{K_{l-1}}\left(\boldsymbol{x}^{l,H}\right)$
- Use exp map to map back to hyperboloid, with different curvature K_l
- All curvatures K_l are trainable!

$$\text{Update}^{K_{l-1},K_l}\left(\boldsymbol{x}^{l,H}\right) := \exp_{\boldsymbol{o}}^{K_l}\left(\sigma\left(\log_{\boldsymbol{o}}^{K_{l-1}}\left(\boldsymbol{x}^{l,H}\right)\right)\right)$$

Model Summary

- At every layer $l = 1 \ldots L$,

  (See paper for Equations)



Transformation **Aggregation** **Update**

- Decode after computing the hyperbolic embeddings $\boldsymbol{x}_i^{L,H}$
  - Node classification: softmax + cross entropy
  - Link prediction: Fermi-Dirac decoder

Experiments

- Citation networks. CORA and PUBMED are standard benchmarks describing citation networks
- Disease networks (synthetic). We simulate the SIR disease spreading model, where the label of a node is whether the node was infected.
- PPI network. Each human tissue has a PPI network, and the dataset is a union of PPI networks for human tissues
- Airport networks. Nodes represent airports and edges represent the airline routes.
- Scale: up to 100K nodes (see paper for data statistics)

Evaluation

- We perform link prediction and node classification for each dataset
- Link prediction: **AUC ROC**
- Node classification: **F1 score**

Methods

- Proposed: hyperbolic graph convolutional networks (HGCN)
  - Use derived hyperbolic message passing on GCN model

- Baselines
  - Poincaré embedding (shallow embedding)
  - Hyperbolic neural networks (no graph information)
  - GNN variants (Euclidean)
    - GCN, GAT, GraphSAGE, SGC

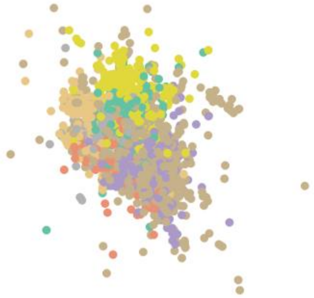
Results

Node classification

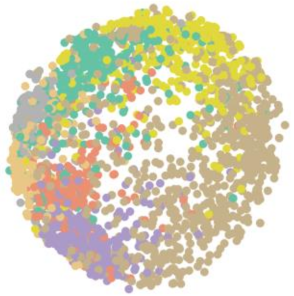
CORA: 2708 ML papers, 7 classes

HGCN performs much better in low dimensions (visualize with 2 dim)

With 2-dimensional Embedding



GCN



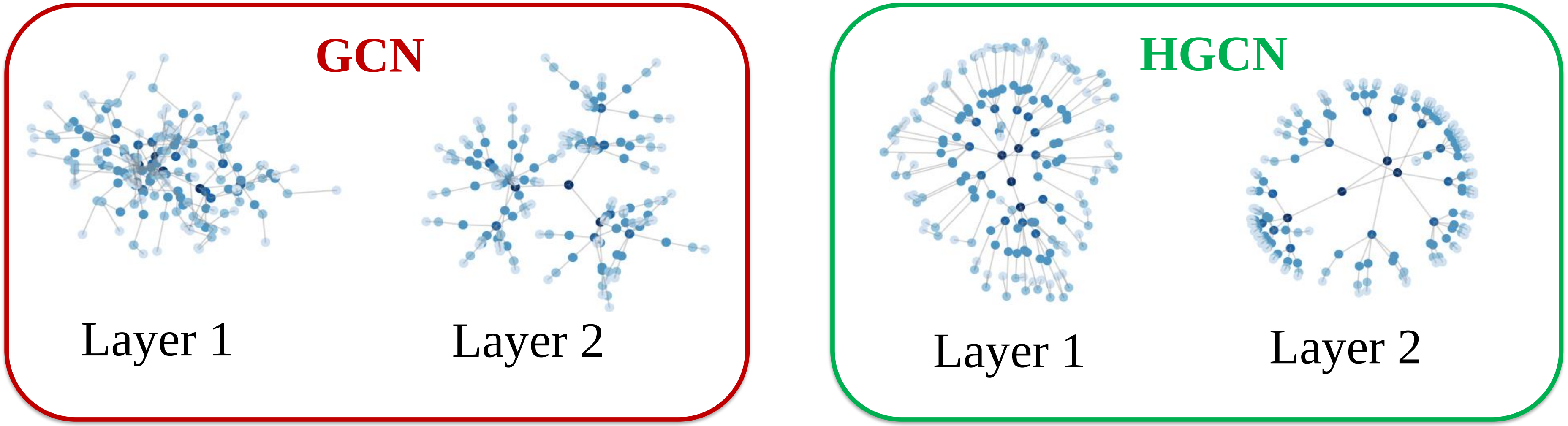
HGCN: nodes clustered by label

Results

Node Classification (tree)

Disease: 1044 nodes, 2 classes

Whether a person is infected according to FIR disease spreading model



Low distortion!

Quantitative Results

- LP: link prediction; NC: node classification

	Dataset	DISEASE		DISEASE-M		HUMAN PPI	
	Hyperbolicity δ	$\delta = 0$		$\delta = 0$		$\delta = 1$	
	Method	LP	NC	LP	NC	LP	NC
Shallow	EUC	59.8 ± 2.0	32.5 ± 1.1	-	-	-	-
	HYP [29]	63.5 ± 0.6	45.5 ± 3.3	-	-	-	-
	EUC-MIXED	49.6 ± 1.1	35.2 ± 3.4	-	-	-	-
	HYP-MIXED	55.1 ± 1.3	56.9 ± 1.5	-	-	-	-
NN	MLP	72.6 ± 0.6	28.8 ± 2.5	55.3 ± 0.5	55.9 ± 0.3	67.8 ± 0.2	55.3 ± 0.4
	HNN[10]	75.1 ± 0.3	41.0 ± 1.8	60.9 ± 0.4	56.2 ± 0.3	72.9 ± 0.3	59.3 ± 0.4
GNN	GCN[21]	64.7 ± 0.5	69.7 ± 0.4	66.0 ± 0.8	59.4 ± 3.4	77.0 ± 0.5	69.7 ± 0.3
	GAT [41]	69.8 ± 0.3	70.4 ± 0.4	69.5 ± 0.4	62.5 ± 0.7	76.8 ± 0.4	70.5 ± 0.4
	SAGE [15]	65.9 ± 0.3	69.1 ± 0.6	67.4 ± 0.5	61.3 ± 0.4	78.1 ± 0.6	69.1 ± 0.3
	SGC [44]	65.1 ± 0.2	69.5 ± 0.2	66.2 ± 0.2	60.5 ± 0.3	76.1 ± 0.2	71.3 ± 0.1
Ours	HGCN	$\mathbf{90.8} \pm 0.3$	$\mathbf{74.5} \pm 0.9$	$\mathbf{78.1} \pm 0.4$	$\mathbf{72.2} \pm 0.5$	$\mathbf{84.5} \pm 0.4$	$\mathbf{74.6} \pm 0.3$
	(%) ERR RED	-63.1%	-13.8%	-28.2%	-25.9%	-29.2%	-11.5%

Inductive tasks

Hyperbolicity measures how tree-like is a graph
Low hyperbolicity: more tree-like

Average gain:
- 9% in link prediction
- 7% in node classification

Quantitative Results

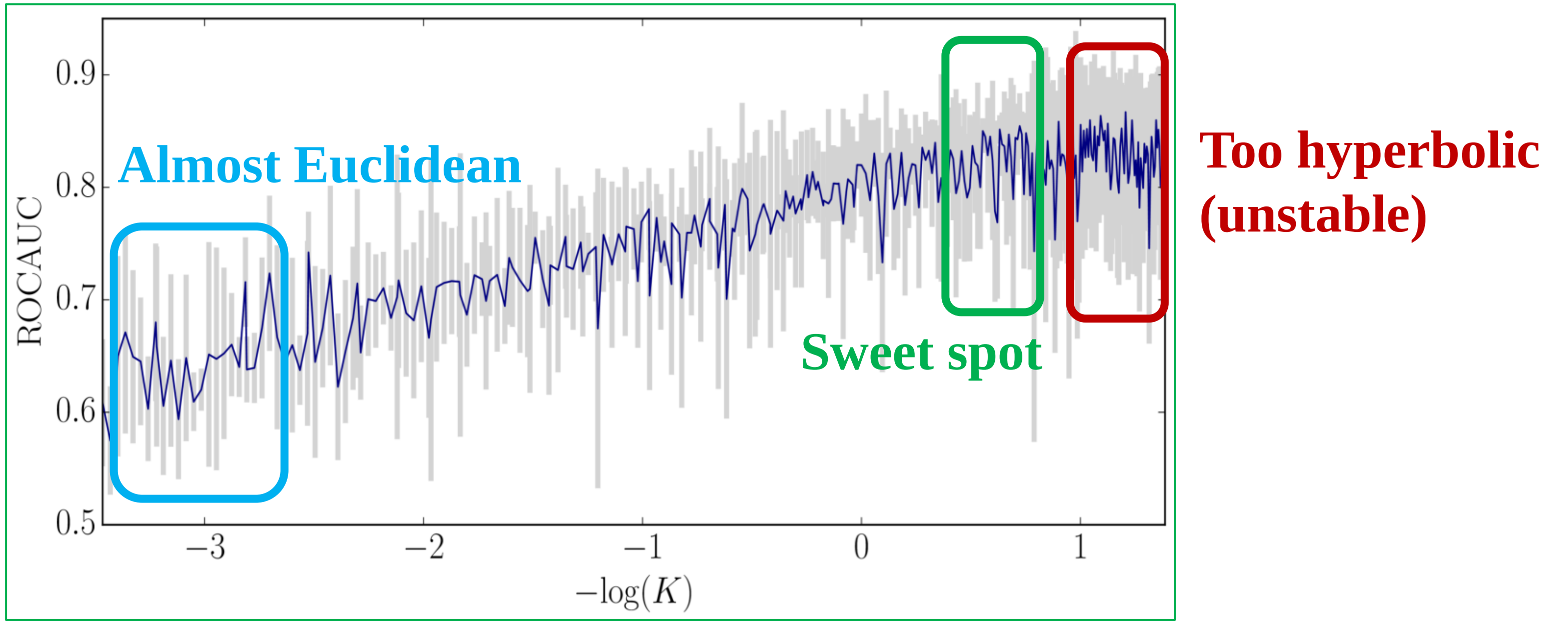
- Hyperbolicity measure correlates with HGCN performance

	Dataset **Hyperbolicity δ**	PubMed $\delta = 3.5$		Cora $\delta = 11$	
	Method	LP	NC	LP	NC
Shallow	Euc	83.3 ± 0.1	48.2 ± 0.7	82.5 ± 0.3	23.8 ± 0.7
	Hyp [29]	87.5 ± 0.1	68.5 ± 0.3	87.6 ± 0.2	22.0 ± 1.5
	Euc-Mixed	86.0 ± 1.3	63.0 ± 0.3	84.4 ± 0.2	46.1 ± 0.4
	Hyp-Mixed	83.8 ± 0.3	73.9 ± 0.2	85.6 ± 0.5	45.9 ± 0.3
NN	MLP	84.1 ± 0.9	72.4 ± 0.2	83.1 ± 0.5	51.5 ± 1.0
	HNN[10]	94.9 ± 0.1	69.8 ± 0.4	89.0 ± 0.1	54.6 ± 0.4
GNN	GCN[21]	91.1 ± 0.5	78.1 ± 0.2	90.4 ± 0.2	81.3 ± 0.3
	GAT [41]	91.2 ± 0.1	79.0 ± 0.3	$\mathbf{93.7} \pm 0.1$	$\mathbf{83.0} \pm 0.7$
	SAGE [15]	86.2 ± 1.0	77.4 ± 2.2	85.5 ± 0.6	77.9 ± 2.4
	SGC [44]	94.1 ± 0.0	78.9 ± 0.0	91.5 ± 0.1	81.0 ± 0.1
Ours	HGCN	$\mathbf{96.3} \pm 0.0$	$\mathbf{80.3} \pm 0.3$	92.9 ± 0.1	79.9 ± 0.2
	(%) Err Red	-27.5%	-6.2%	+12.7%	+18.2%

Non-hierarchical dataset (low hyperbolicity)

Effect of Curvature

- Curvature is crucial to performance
- *E.g.* Disease dataset link prediction:

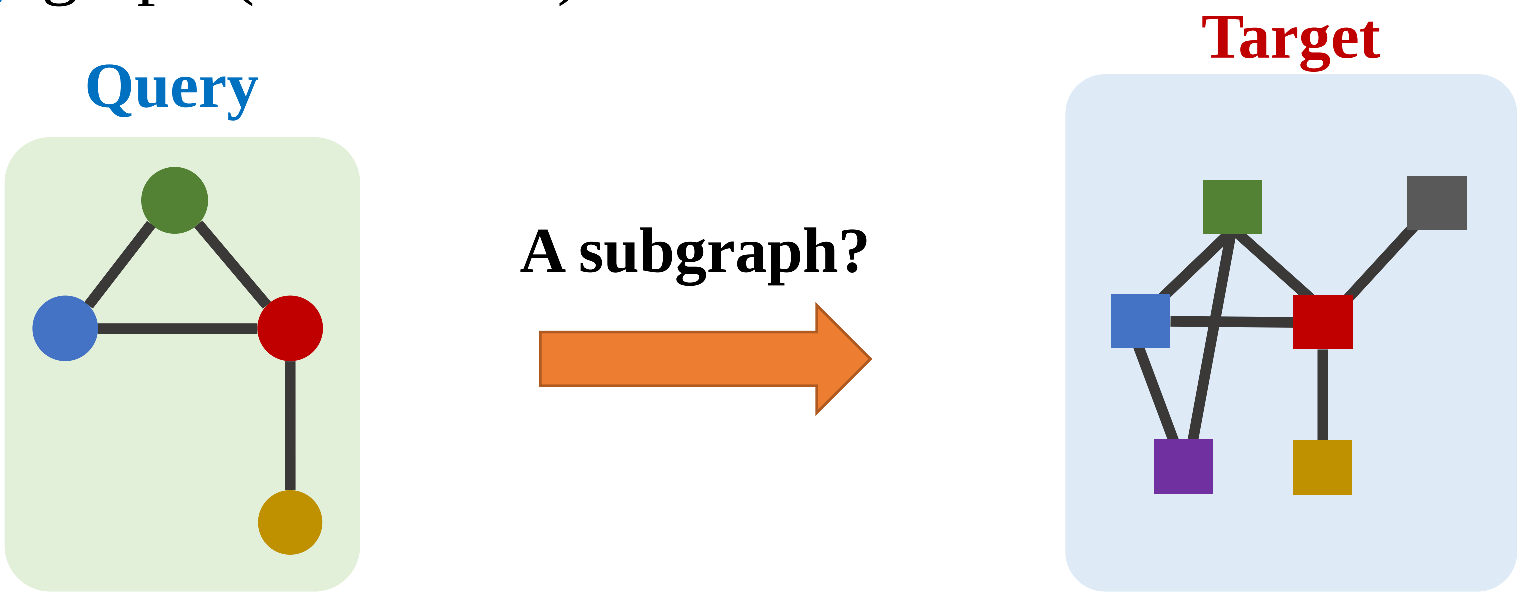


Outline: Embedding Geometry

- Hyperbolic Graph Convolutional Neural Networks
- Neural Subgraph Matching

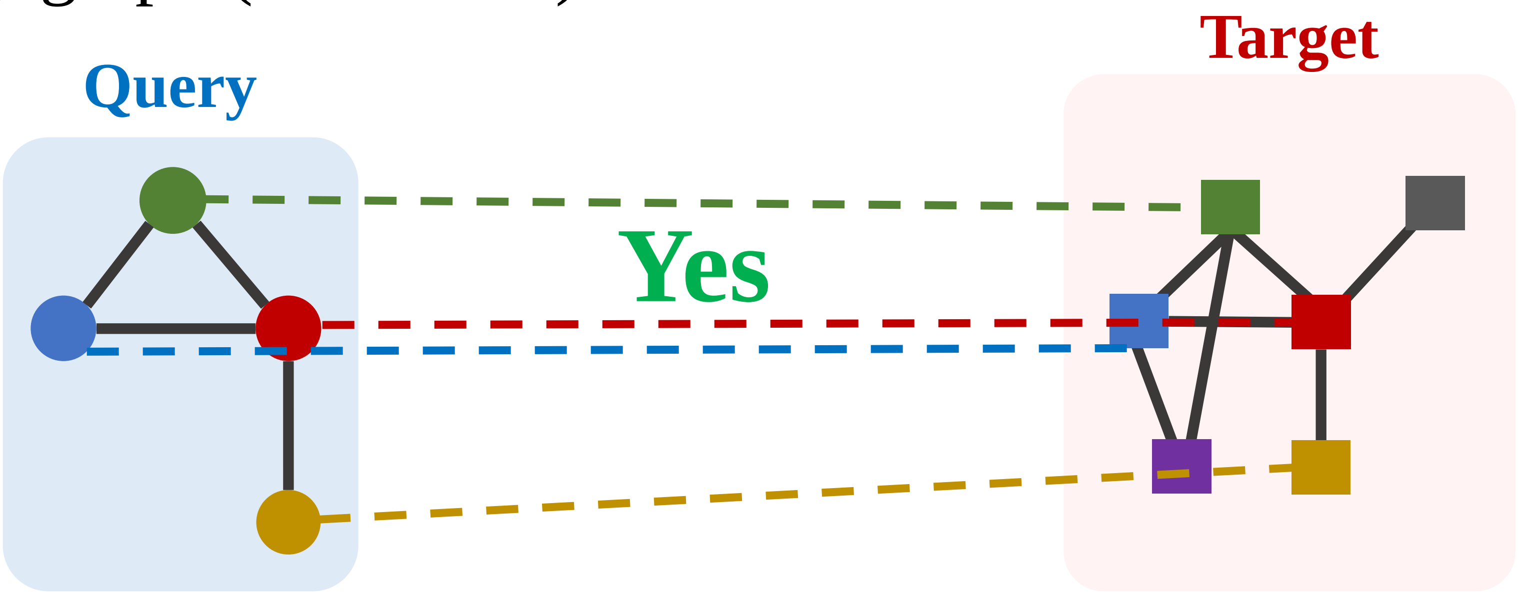
Task Setup

- Large target graph (can be disconnected)
- Query graph (connected)



Task Setup

- Large target graph (can be disconnected)
- Query graph (connected)

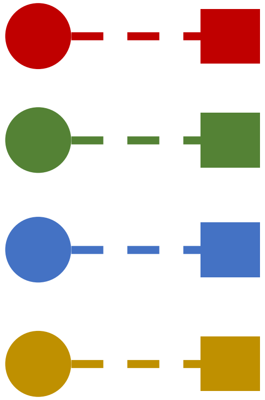


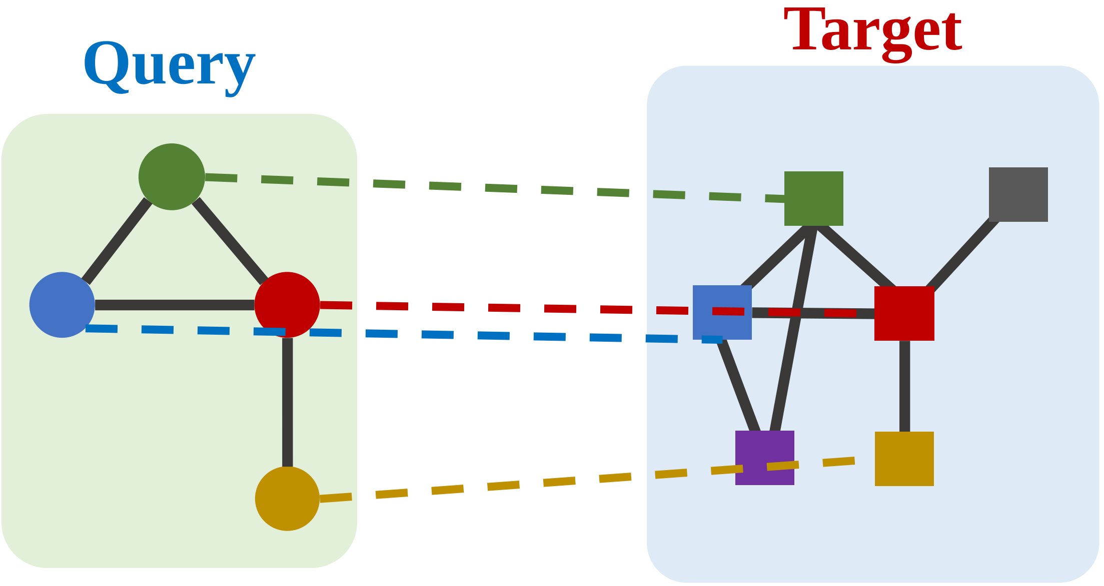
Different queries can arrive in real-time

Mostly fixed in domain application

Task Setup

- Desired output
  - Whether subgraph relation holds
  - Locate the subgraph
    (Identify the query in a neighborhood of target)
  - Find all correspondences



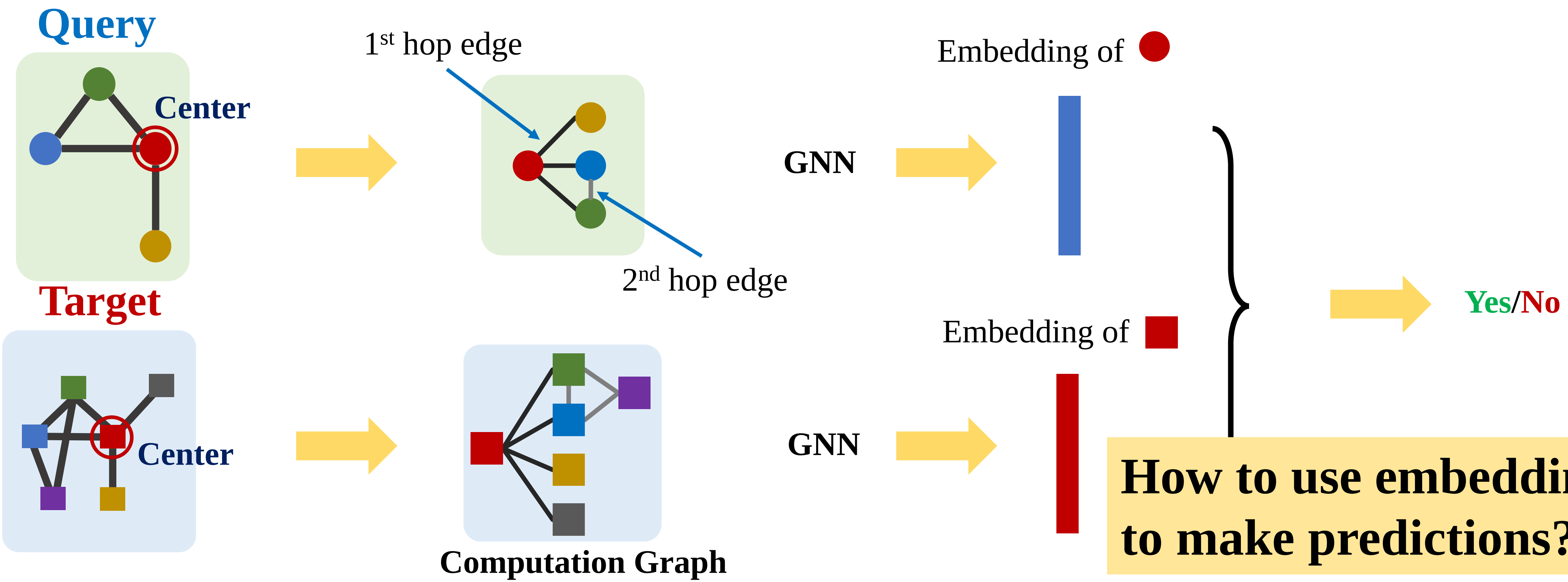


Challenges

- NP-Complete Problem
- Existing solutions
  - Heuristics-based (exponential time in worst case)
  - Approximation (domain-specific)
- **Can we use neural models to learn a subgraph matching strategy?**
  - High accuracy approximation
  - Leverage inductive bias of datasets
  - No hand-designed heuristics
  - Unexplored by previous works (the closest is neural graph isomorphism)

NeuroMatch Architecture

- Siamese Graph Neural Network Structure



How to use embeddings to make predictions?

Order Embedding Space

- Embedding space: Euclidean space (e.g. 64 dim)
- Order constraint

Embedding dimension

$$\forall_{i=1}^{D} z_q[i] \leq z_u[i] \quad \text{iff} \quad G_Q \subseteq G_U$$

trained with max-margin loss

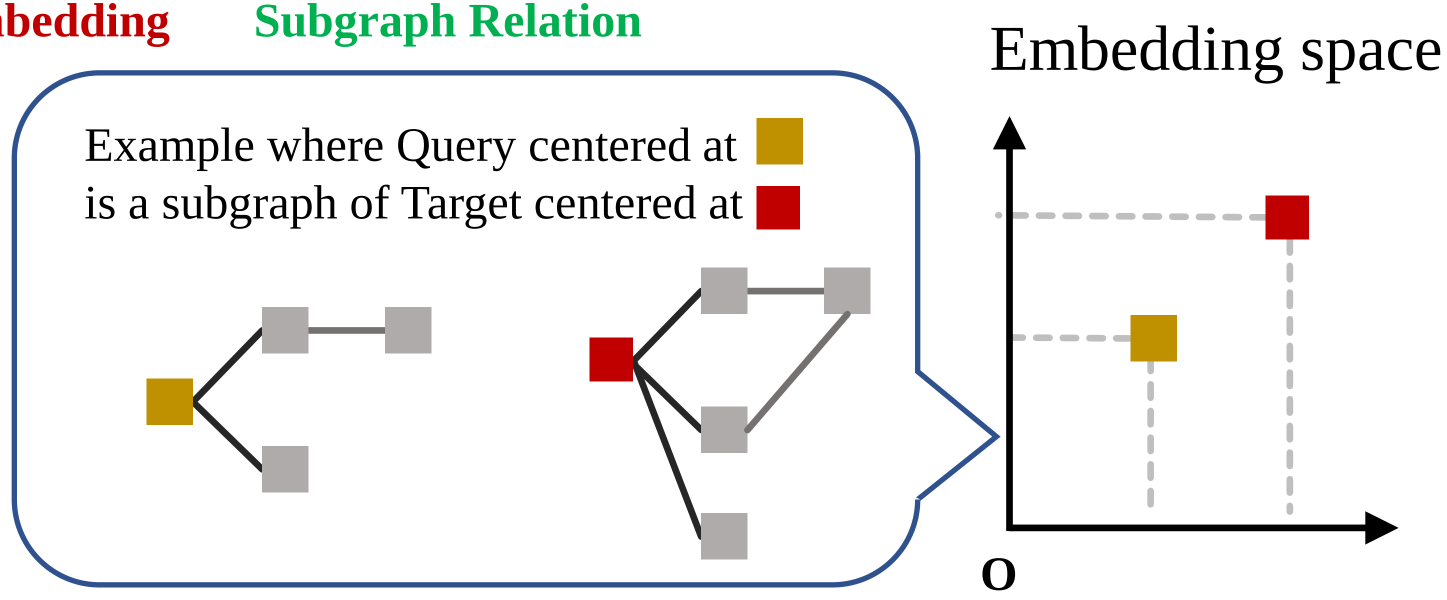
Query embedding **Target embedding** **Subgraph Relation**

✓ Transitivity

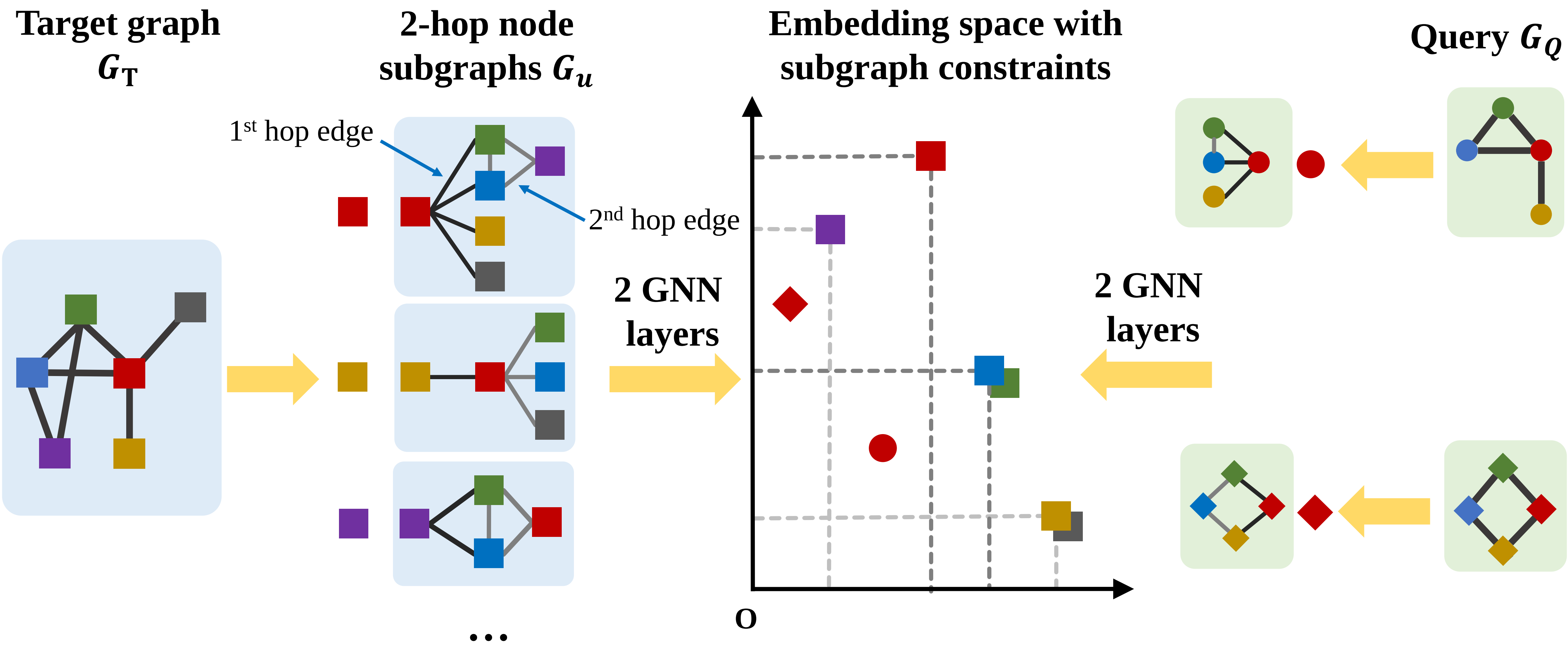
✓ Anti-symmetry

✓ Non-empty intersection

✓ Composition



Final Model Summary



Performance: Model Comparison

- Metric: AUROC
- Order embedding achieves 4% relative gains in accuracy of the binary prediction of subgraph relationship

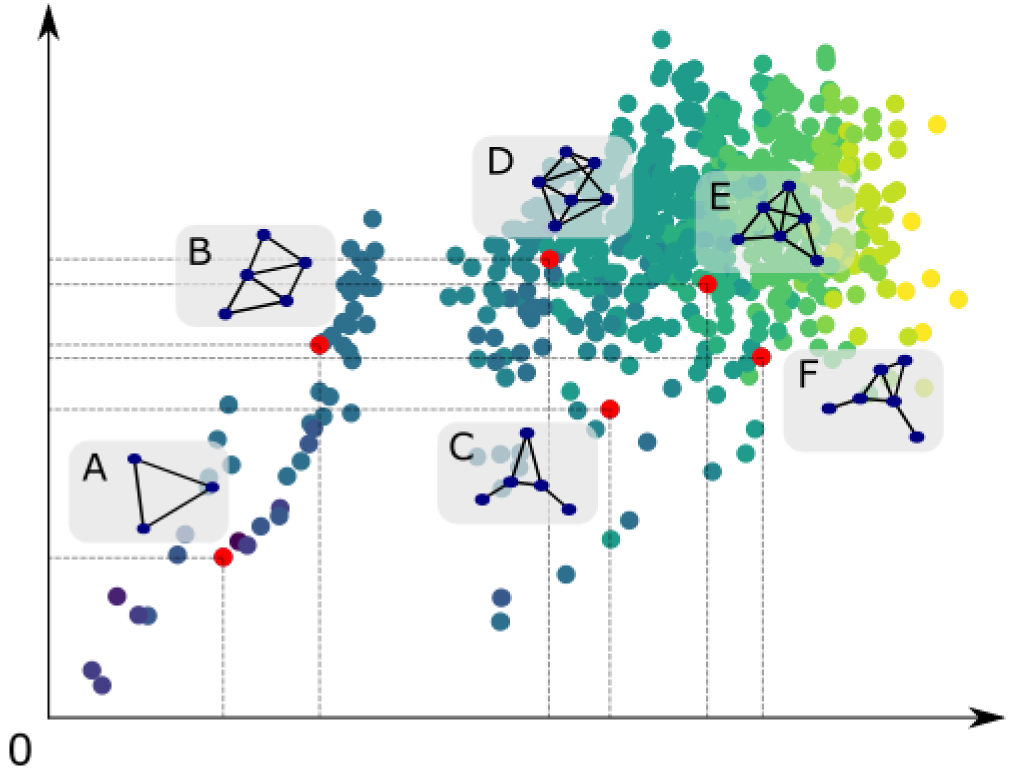
	Dataset	E-R	COX2	DD	MSRC_21	FIRSTMMDB	PPI	WORDNET18
Base	GMNN [33]	73.6 ± 1.1	75.9 ± 0.8	80.6 ± 1.5	82.5 ± 1.7	81.5 ± 2.9	72.0 ± 1.9	80.3 ± 2.0
	RDGCN [31]	79.5 ± 1.2	80.1 ± 0.4	81.3 ± 1.2	81.9 ± 1.9	82.4 ± 3.4	76.8 ± 2.2	79.6 ± 2.5
Ablation	NO CURRICULUM	82.4 ± 0.6	95.0 ± 1.6	96.7 ± 2.1	89.2 ± 2.0	87.2 ± 6.8	82.6 ± 1.7	81.4 ± 2.2
	NM-MLP	88.7 ± 0.5	95.4 ± 1.6	**98.4** ± 0.3	93.5 ± 1.0	92.9 ± 4.3	85.5 ± 1.4	87.9 ± 1.2
	NM-NTN	89.1 ± 1.9	89.3 ± 0.9	96.4 ± 1.4	94.7 ± 3.2	89.6 ± 1.1	85.7 ± 2.4	85.0 ± 1.1
	NM-BOX	84.5 ± 2.1	88.5 ± 1.2	91.4 ± 0.5	90.8 ± 1.4	93.8 ± 1.8	77.4 ± 3.1	82.7 ± 2.5
	NEUROMATCH	**93.5**± 1.1	**97.2** ± 0.4	97.5 ± 1.2	**96.1** ± 0.2	**95.5** ± 2.1	**89.9** ± 1.9	**89.3** ± 2.4

Performance: Generalization

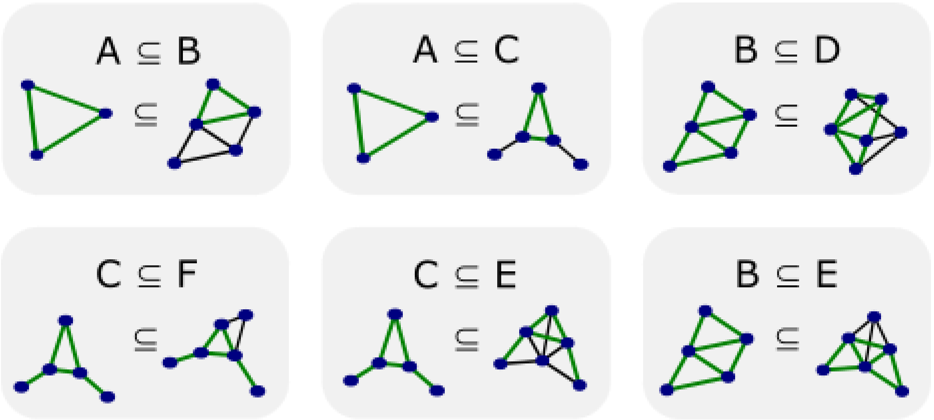
- Metric: AUROC
- Model trained only on synthetic data can generalize to real datasets from diverse domains
- Demonstrates universality of order embedding space

Dataset	ENZYMES	COX2	AIDS	PPI	IMDB-BINARY
TRANSFER	78.9	93.9	92.2	81.0	74.2
IN-DOMAIN	92.9	95.1	94.3	84.5	81.8

Visualization: Order Embedding Space



Subgraph relation preserved by embedding order relation



Conclusion

- Combine both GNN architecture and embedding geometry to achieve best performance
- Embedding geometry is crucial in learning embeddings for graphs
  - Hyperbolic embedding: Hierarchical, tree-like graphs
  - Order embedding: partial ordering structure

Thank you!