Link Prediction from Sparse Data Using Meta Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Predicting missing edges in graph-structured data is a fundamental task in graph representation learning. However, existing research focuses primarily on link prediction in dense graphs, where the majority (i.e., at least 50%) of the graph edges are observable during training. In this work, we investigate the more challenging setting, where we must train a link prediction model using only a sparse sample (i.e., less than 30%) of the edges in an otherwise dense graph. We find that the performance of state-of-the-art link prediction models degrades substantially in this setting. To address this shortcoming, we introduce a gradient-based meta-learning approach—Meta-Graph—that allows a GNN-based link prediction model to effectively leverage training data from multiple graphs. The key idea behind Meta-Graph is that we learn to conditionally generate parameter initializations for GNN link prediction models. We show that this approach is effective for overcoming the challenge of sparse graph data, outperforming standard pre-training, the MAML algorithm, and other strong baselines (e.g., DeepWalk).

1 Introduction

The goal of link prediction is to learn from a graph to infer missing or previously unknown relationships [22]. For instance, in a social network we may use link prediction to power a friendship recommendation system [2], or in the case of biological network data we link prediction might be used to infer possible relationships between drugs and diseases [43]. However, despite its popularity, previous work on link prediction generally focuses only on one particular problem setting: previous work generally assumes that link prediction is to be performed on a single dense graph, with at least 50% of the true edges observed during training [e.g., see 11, 18, 22, 24].

In this work, we consider the more challenging setting, where we must perform link prediction using only a sparse sample of a graph (i.e., with less than 30% of the true edges observed during training). While challenging, this setting is representative of many real-world scenarios. For example, social recommendation platforms often have very sparse coverage of the underlying user preferences [30]; biomedical interaction networks often contain only a small subset of the true underlying interactions, due to the cost of in vitro assays [44]; and, the constraints of sensor networks often lead to sparse samples of the underlying network data [9]. The prevalence of sparse graph data in real-world applications motivates the investigation of three key research questions:

Q1 How do state-of-the-art link prediction models perform in the face of sparse data? Do more recent methods—which are based on graph neural networks (GNNs)—perform best in this setting? Or are simpler heuristic methods more robust to sparse training data?

Q2 How can we leverage data across multiple graphs to improve performance on sparse data? Pre-training strategies have proved essential in many deep learning domains—e.g., for images [10]

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

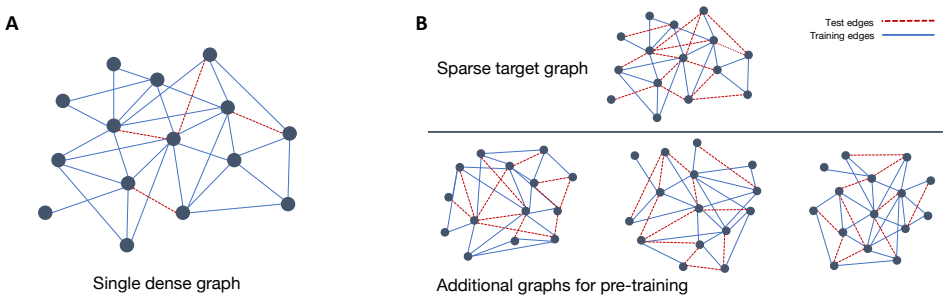

Figure 1: Comparison of the standard link prediction setting (**A**) and the sparse link prediction setting that is the focus of this work (**B**).

and text [7]—and achieved some success recently on graphs [14, 38]. However, it is unclear if pre-training is an effective strategy for overcoming sparse data in the context of link prediction.

Q3 Link prediction from sparse data bears similarities to *few shot learning*, which has received substantial attention in recent years [27, 20]. How can we leverage recent advancements in meta-learning to improve the performance of link prediction from sparse data? Finding successful ways to adapt meta learning strategies to graph representation learning is an open challenge.

In this work, we seek to shed light on these important questions. To do so, we perform experiments on three representative domains: as a representative social dataset, we include a citation network dataset extracted from AMiner [37]; as a representative biological dataset, we use human protein-protein interaction (PPI) data [43]; and, finally, as a representative sensor network dataset we use the FirstMM database, which contains graph data extracted from robot sensors [29]. These datasets were chosen due to their desirable properties for our investigations. Most importantly, the original versions of these datasets are relatively dense, allowing us to simulate the challenge of sparse data by removing edges. In addition, all these datasets contain multiple disjoint graphs from a single domain, facilitating the investigation of pre-training and meta learning approaches.

Key findings. Using these datasets, we show that existing link prediction methods [32, 1, 18]—which are based on node embeddings and neighborhood overlap statistics—exhibit substantial declines in performance as the sparsity of the training data increases. However, we find that GNN-based link prediction models are able to maintain strong performance in the sparse data setting, with and without pre-training. Lastly, we propose and investigate a novel meta-learning strategy for link prediction using GNNs. Our approach—termed Meta-Graph—uses gradient-based meta learning to conditionally generate parameter initializations for GNN link prediction models. We show that Meta-Graph achieves the strongest performance in the sparse data regime, where at most 30% of the edges are observed. In this sparse setting and across all datasets, Meta-Graph provides an average relative improvement of 5.3% in AUC compared to pre-training and an average relative improvement of 4.9% compared to the standard MAML algorithm [8]. Overall, our analysis highlights important challenges and promising new methodologies for link prediction in the sparse data regime.

2 Preliminaries

We assume that we have a distribution $p(\mathcal{G})$ over graphs, from which we can sample training graphs $\mathcal{G}_i \sim p(\mathcal{G})$, where each $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, X_i)$ is defined by a set of nodes $\mathcal{V}_i$, edges $\mathcal{E}_i$, and matrix of real-valued node attributes $X \in \mathbb{R}^{|\mathcal{V}_i| \times d}$. When convenient, we will also equivalently represent a graph as $\mathcal{G}_i = (\mathcal{V}_i, A_i, X_i)$, where $A_i \in \mathbb{Z}^{|\mathcal{V}_i| \times |\mathcal{V}_i|}$ is an adjacency matrix. We assume that each of these sampled graphs, $\mathcal{G}_i$, is a simple graph (i.e., contains a single type of relation and no self loops) and that every node $v \in \mathcal{V}_i$ in the graph is associated with a real valued attribute vector $\mathbf{x}_v \in \mathbb{R}^d$ from a common vector space. We further assume that for each graph $\mathcal{G}_i$ we have access to only a sparse subset of the true edges $\mathcal{E}_i^{\text{train}} \subset \mathcal{E}_i$ (with $|\mathcal{E}_i^{\text{train}}| << |\mathcal{E}_i|$) during training.

Our goal is to perform link prediction on a graph $\mathcal{G}_*$, which we term the *target graph*. In other words, we seek to predict the existence of edges in the set $\mathcal{E}_*^{\text{test}} = \mathcal{E}_* \setminus \mathcal{E}_*^{\text{train}}$ by training a model on the training edges $\mathcal{E}_*^{\text{train}}$ and node features X_* of graph $\mathcal{G}_*$. However, we will also allow models to *pre-train* their parameters on a sample of training graphs $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$ from the distribution $p(\mathcal{G})$.

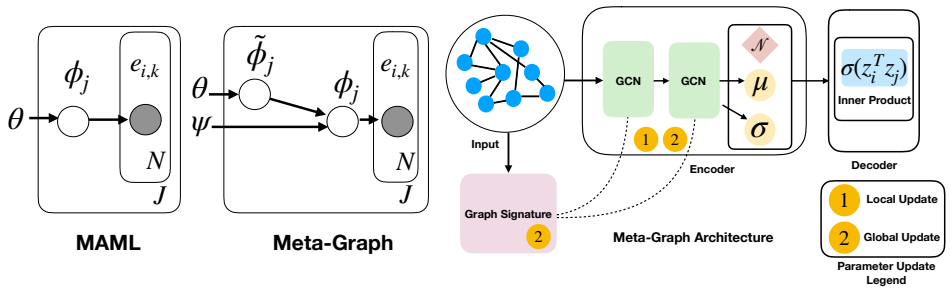

Figure 2: **Left:** Graphical model for MAML vs. Meta-Graph. **Right:** Meta-Graph architecture.

We thus distinguish between the *fine-tuning* of a link prediction model on the target graph $\mathcal{G}_*$ and the *pre-training* of a model using additional graphs $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$.

3 Models

In this section, we describe the various models that we investigate for the task of link prediction from sparse graph data. We begin with two strong baseline approaches—the Adamic-Adar heuristic [1] and DeepWalk [32]—which we use as representative heuristic and embedding-based methods, respectively. While powerful, both these methods are inherently *transductive* [13]; thus, they are only able to learn from a single graph dataset and cannot benefit from pre-training or meta-learning strategies. We then introduce the variational graph autoencoder method (VGAE) [17], which we use as our base graph neural network (GNN) model. Importantly, this method is *inductive* [13], which facilitates the application of pre-training and meta-learning approaches. Lastly, we describe how the VGAE approach can be extended using pre-training and meta-learning strategies, and we introduce in Section 3.4 our novel meta-learning approach, termed Meta-Graph.

3.1 Single-graph Baselines

Adamic-Adar Heuristic. Heuristic link prediction approaches predict missing edges by computing neighborhood overlap scores between pairs of nodes. One of the most successful heuristic approaches is the Adamic-Adar score [1]: $P(A[v_1, v_2] = 1) \propto \sum_{v' \in \mathcal{N}(v_1) \cap \mathcal{N}(v_2)} \frac{1}{\log(|\mathcal{N}(v')|)}$, where $P(A[v_1, v_2] = 1)$ denotes the likelihood of an edge occurring between nodes v_1 and v_2, and $\mathcal{N}(v)$ denotes the neighborhood of node v. Note that this approach is a statistical heuristic specific to a particular graph: there are no parameters to train and this approach cannot leverage feature information or information from multiple disjoint graphs.

DeepWalk. Node embedding methods—especially those based on random walks—have achieved impressive results in graph representation learning [12]. Here, we use the DeepWalk algorithm as a representative embedding-based approach [32]. In DeepWalk, we learn embeddings $\mathbf{z}_u \in \mathbb{R}^d$ for each node in the graph, which are optimized to encode the statistics of random walks. The likelihood of an edge $P(A[u, v])$ can then be predicted as $P(A[u, v] = 1) \propto \mathbf{z}_u^\top \mathbf{z}_v$ [12]. Note that—like the Adamic-Adar score—DeepWalk does not leverage node features, and it is inherently transductive, meaning it cannot benefit from information across multiple graphs (e.g., for pre-training) [13].

3.2 Variational Graph Autoencoder

Both the heuristic and embedding-based methods are limited in their ability to leverage multi-graph data (e.g., for pre-training). However, recent years have witnessed substantial progress in the development of link prediction models based on graph neural networks (GNNs). These GNN approaches are naturally inductive, capable of leveraging training data from multiple different graphs, and can effectively incorporate node feature data [13]. In this work, we build upon the variational graph autoencoder (VGAE) approach for GNN-based link prediction, which achieves strong performance on standard link prediction benchmarks [18].

Given a graph $\mathcal{G} = (\mathcal{V}, A, X)$, the VGAE learns an inference model, q_ϕ, that defines a distribution over node embeddings $q_\phi(Z|A, X)$, where each row $z_v \in \mathbb{R}^d$ of $Z \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a node embedding

that can be used to score the likelihood of an edge existing between pairs of nodes. The parameters of the inference model are shared across all the nodes in $\mathcal{G}$, to define the approximate posterior $q_\phi(z_v|A, X) = \mathcal{N}(z_v|\mu_v, \text{diag}(\sigma_v^2))$, where the parameters of the normal distribution are learned via GNNs:

$$\mu = \text{GNN}_\mu(A, X), \qquad \text{and} \qquad \log(\sigma) = \text{GNN}_\sigma(A, X). \tag{1}$$

The generative component of the VGAE is then defined as

$$p(A|Z) = \prod_{u=1}^{N} \prod_{v=1}^{N} p(A_{u,v}|z_u, z_v), \qquad \text{with} \qquad p(A_{u,v}|z_u, z_v) = \sigma(z_u^\top z_v), \tag{2}$$

i.e., the likelihood of an edge existing between two nodes, u and v, is proportional to the dot product of their node embeddings. Given the above components, the inference GNNs can be trained to minimize the variational lower bound on the training data:

$$\mathcal{L}_G = \mathbb{E}_{q_\phi}[\log p(A^{\text{train}}|Z)] - KL[q_\phi(Z|X, A^{\text{train}})||p(Z)], \tag{3}$$

where a Gaussian prior is used for $p(Z)$. In all our experiments, we implemented the inference GNNs based on the propagation rules proposed by Kipf et al. [17].

3.3 Pre-training and Meta Learning with VGAEs

As discussed above, a key benefit of GNN-based link prediction approaches is the fact that they are inherently inductive and can be trained on multiple graphs. Thus, in our experiments, we investigate the utility of *pre-training* the inference GNNs on a set of training graphs $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$ sampled from a particular domain. When we want to predict edges on the target graph $\mathcal{G}_*$, we can then *fine-tune* the parameters of our inference GNNs using gradient descent on the training edges $\mathcal{E}_*^{\text{train}}$ of this graph.

Moreover, we also investigate combining VGAEs with gradient-based meta learning. In particular, we experiment with applying the MAML algorithm [8] to our setting, where we use second-order gradient descent to learn a global parameter initialization for a VGAE model based on the training graphs $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$. We defer the technical details of the MAML approach for link prediction to the following section, as it is a special case of our extended Meta-Graph framework.

3.4 Meta-Graph

In the previous section, we introduced the idea of pretraining a VGAE model or optimizing it using a meta-learning approach, with the goal of improving performance when only a sparse sample of edges are available. However, simply applying pre-training or the standard MAML algorithm to the VGAE model ignores the specific details of the graph setting. In this section, we address this issue and present a novel gradient-based meta learning strategy—termed Meta-Graph—which is specialized to the graph domain. The key idea behind Meta-Graph is that we use gradient-based meta learning to optimize a shared parameter initialization θ for the inference models of a VGAE, while also learning a parametric encoding $\psi(\mathcal{G}_i)$ that modulates this parameter initialization in a graph-specific way. Specifically, given a sampled training graph $\mathcal{G}_i$, we initialize the inference model q_{ϕ_i} for a VGAE link prediction model using a combination of two learned components:

- A global initialization, θ, that is used to initialize all the parameters of the GNNs in the inference model. The global parameters θ are optimized via second-order gradient descent to provide an effective initialization point for any graph sampled from the distribution $p(\mathcal{G})$.
- A graph signature $s_{\mathcal{G}_i} = \psi(\mathcal{G}_i)$ that is used to modulate the parameters of inference model ϕ_i based on the history of observed training graphs. In particular, we assume that the inference model q_{ϕ_i} for each graph $\mathcal{G}_i$ can be conditioned on the graph signature. That is, we augment the inference model to $q_{\phi_i}(Z|A, X, s_{\mathcal{G}_i})$, where we also include the graph signature $s_{\mathcal{G}_i}$ as a conditioning input. We use a K-layer graph convolutional network (GCN) [17], with sum pooling to compute the signature:

$$s_{\mathcal{G}} = \psi(\mathcal{G}) = \text{MLP}(\sum_{v \in \mathcal{V}} z_v) \qquad \text{with} \qquad Z = \text{GCN}(A, X), \tag{4}$$

  where GCN denotes a K-layer GCN (as defined in [17]), MLP denotes a densely-connected neural network, and we are summing over the node embeddings z_v output from the GCN. As with the global parameters θ, the graph signature model ψ is optimized via second-order gradient descent.

The overall Meta-Graph architecture is detailed in Figure 2 and the core learning algorithm is summarized in the algorithm block below:

Algorithm 1: Meta-Graph

Result: Global parameters θ, Graph signature function ψ
Initialize learning rates: α, ϵ
Sample a mini-batch of graphs, $\mathcal{G}_{batch}$ from $p(\mathcal{G})$;
for *each* $\mathcal{G} \in \mathcal{G}_{batch}$ **do**
  $\mathcal{E} = \mathcal{E}^{\text{train}} \cup \mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{test}}$ // Split edges into train, val, and test
  $s_{\mathcal{G}} = \psi(\mathcal{G}, \mathcal{E}^{\text{train}})$ // Compute graph signature
  Initialize: $\phi^{(0)} \leftarrow \theta$ // Initialize local parameters via global parameters
  for k *in* $[1 : K]$ **do**
    $s_{\mathcal{G}} = \text{stopgrad}(s_{\mathcal{G}})$ // Stop Gradients to Graph Signature
    $\mathcal{L}_{train} = \mathbb{E}_q[\log p(A^{\text{train}}|Z)] - KL[q_\phi(Z|\mathcal{E}^{\text{train}}, s_{\mathcal{G}})||p(z)]$
    Update $\phi^{(k)} \leftarrow \phi^{(k-1)} - \alpha \nabla_\phi \mathcal{L}_{train}$
  end
  Initialize: $\theta \leftarrow \phi_K$
  $s_{\mathcal{G}} = \psi(\mathcal{G}, \mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{train}})$ // Compute graph signature with validation edges
  $\mathcal{L}_{val} = \mathbb{E}_q[\log p(A^{\text{val}}|Z)] - KL[q(Z|\mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{train}}, s_{\mathcal{G}})||p(Z)]$
  Update $\theta \leftarrow \theta - \epsilon \nabla_\theta \mathcal{L}_{val}$
  Update $\psi \leftarrow \psi - \epsilon \nabla_\psi \mathcal{L}_{val}$
end

MAML for link prediction as a special case. Note that a simplification of Algorithm 1, where the graph signature function is removed, can be viewed as an adaptation of model agnostic meta learning (MAML) [8] to the link prediction setting.

3.4.1 Variants of Meta-Graph

We consider several concrete instantiations of the Meta-Graph framework, which differ in terms of how the output of the graph signature function is used to modulate the parameters of the VGAE inference models. In particular, we assume that all the inference GNNs (Equation 1) are defined by stacking K neural message passing layers of the form:

$$h_v^{(k)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{m_{s_{\mathcal{G}}}\left(W^{(k)} h_u^{(k-1)}\right)}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}}\right), \tag{5}$$

where $h_v \in \mathbb{R}^d$ denotes the embedding of node v at layer k of the model, $\mathcal{N}(v) = \{u \in \mathcal{V} : e_{u,v} \in \mathcal{E}\}$ denotes the nodes in the graph neighborhood of v, and $W^{(k)} \in \mathbb{R}^{d \times d}$ is a trainable weight matrix for layer k. The key difference between Equation 5 and the standard GCN propagation rule [17] is that we add the modulation function $m_{s_{\mathcal{G}}}$, which is used to modulate the message passing based on the graph signature $s_{\mathcal{G}} = \psi(\mathcal{G})$. We describe different variations of this modulation below.

GS-Modulation. Inspired by Brockshmidt et al. [4], we experiment with basic feature-wise linear modulation [36] to define the modulation function $m_{s_{\mathcal{G}}}$:

$$\beta_k, \gamma_k, = \psi(\mathcal{G})$$
$$m_{\beta_k, \gamma_k}\left(W^{(k)} h_u^{(k-1)}\right) = \gamma_k \odot W h^{(k-1)} + \beta_k. \tag{6}$$

Here, we restrict the modulation terms β_k and γ_k output by the signature function to be in $[-1, 1]$ by applying a $\tanh$ non-linearity after Equation 4.

GS-Gating. To allow the model to adaptively learn when to apply modulation, we extend the feature-wise linear modulation using a sigmoid gating term, ρ_k (with $[0, 1]$ entries), that gates in the influence of γ and β:

$$\beta_k, \gamma_k, \rho_k = \psi(\mathcal{G})$$
$$\beta_k = \rho_k \odot \beta_k + (1 - \rho_k) \odot 1, \ \gamma_k = \rho_k \odot \gamma_k + (1 - \rho_k) \odot 1$$
$$m_{\beta_k, \gamma_k}\left(W^{(k)} h_u^{(k-1)}\right) = \gamma_k \odot W h^{(k-1)} + \beta_k.$$

Table 1: Statistics for the three datasets used to test Meta-Graph.

DATASET	#GRAPHS	AVG. NODES	AVG. EDGES	#NODE FEATS
PPI	24	2,331	64,596	50
FIRSTMM	41	1,377	6,147	5
AMINER	72	462	2245	300

GS-Weights. In the final variant of Meta-Graph, we extend the gating and modulation idea by separately aggregating graph neighborhood information with and without modulation and then merging these two signals via a convex combination:

$$\beta_k, \gamma_k, \rho_k = \psi(\mathcal{G})$$
$$h_v^{(k)} = \rho_k \odot h_v^{(k),1} + (1 - \rho_k) \odot h_v^{(k),2},$$

where $h_v^{(k),1}$ is defined based on the standard GCN propagation rule [17], $h_v^{(k),2}$ is compute according to Equation 5, and we use the basic linear modulation (Equation 6) to define $m_{s_{\beta_k}, \gamma_k}$.

4 Experiments

We now investigate the performance of the different models introduced in Section 3, with the goal of addressing the three key research questions outlined in Section 1.

4.1 Setup and Datasets

Two of our benchmarks are derived from standard multi-graph datasets from protein-protein interaction (PPI) networks [43] and 3D point cloud data (FirstMM-DB) [29]. We also create a novel multi-graph dataset based upon the AMINER citation data [37], where each node corresponds to a paper and links represent citations (Appendix A) and node embeddings are GloVe embedding [31] of the papers abstract. We preprocess all graphs in each domain such that each graph contains a minimum of 100 nodes and up to a maximum of 20000 nodes. For pre-training and meta learning, we take 80% of graphs as the training set and the remaining graphs we split evenly between validation and test. For all datasets, we perform link prediction by training on a small subset (i.e., a percentage) of the edges and then attempting to predict the unseen edges (with 20% of the held-out edges used for validation). Key dataset statistics are summarized in Table 1.

For Meta-Graph and all of these baselines we employ Bayesian optimization with Thompson sampling [16] to perform hyperparameter selection using the validation sets (with the exception of the K parameter in Algorithm 1, which was fixed to $K = 5$ based on preliminary experience training the models). We use the recommended default hyperparameters for DeepWalk. Code to replicate our experiments is included with the submission and will be made public.

4.2 Results and Discussion

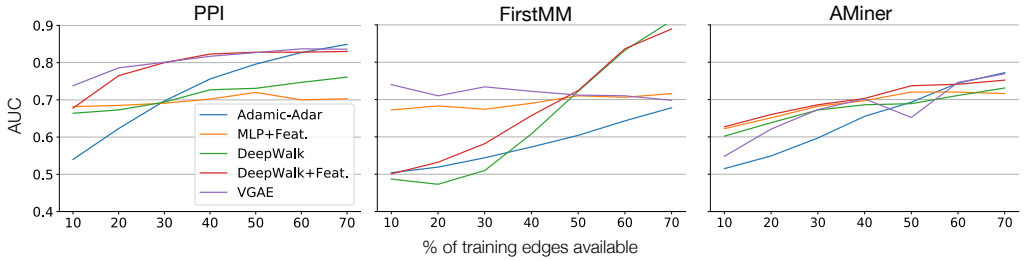

Figure 3: Link prediction results for baseline approaches as the training data sparsity is varied. (See Appendix B for full results in tabular form.)

Table 2: Performance of different pre-training and meta-learning approaches.

% Edges		10%	20%	30%	40%	50%	60%	70%
PPI	VGAE	0.738	0.786	0.801	0.817	0.827	0.837	0.836
	VGAE+pretrain	0.752	0.8010	0.821	0.832	0.818	**0.856**	0.841
	MAML	0.745	0.820	0.840	0.852	**0.854**	0.856	**0.863**
	Meta-Graph	**0.795**	**0.831**	**0.846**	**0.853**	0.848	0.853	0.855
FirstMM	VGAE	0.740	0.710	0.734	0.722	0.712	0.710	0.698
	VGAE+pretrain	0.752	0.735	0.723	0.734	0.749	0.700	0.695
	MAML	0.776	0.782	**0.793**	**0.785**	**0.791**	0.663	**0.788**
	Meta-Graph	**0.782**	**0.786**	0.783	0.781	0.760	**0.746**	0.739
AMiner	VGAE	0.548	0.621	0.673	0.702	0.652	0.7458	0.769
	VGAE+pretrain	0.623	0.691	0.723	0.764	0.767	0.792	0.781
	MAML	0.561	0.662	0.667	0.682	0.720	0.741	0.768
	Meta-Graph	**0.626**	**0.738**	**0.786**	**0.791**	**0.792**	**0.817**	**0.786**

Q1: How does sparsity impact the performance of existing methods? We first investigate the extent to which training data sparsity impacts the performance of existing approaches. Figure 3 shows the link prediction AUC performance of the Adamic-Adar heuristic, DeepWalk, and a VGAE model. All models were trained and tested on individual graphs, and the performance is averaged over the test set graphs from each domain. DeepWalk and the Adamic-Adar score exhibit substantial declines in performance as the sparsity increases. The VGAE model appears to be robust—especially on PPI and FirstMM—and achieves the best average performance in the sparse regime (i.e., $\leq 30\%$ edges). Interestingly, however, the Adamic-Adar and DeepWalk models tend to outperform the VGAE approach in the presence of very dense training data (i.e., $\geq 60\%$ of the edges observed). To investigate the role of node features in the model performance, we also considered an extension of DeepWalk that integrates node features using a multi-layer perceptron (MLP), as well as an MLP model that predicts links using node features alone (Appendix C). Both these feature-based baselines perform well in the sparse setting, but the VGAE model—which naturally integrates features—maintains the strongest performance. Overall, VGAE outperforms the next-best baseline (a combination of DeepWalk and node features) by an average of 32%, in terms of relative AUC improvement in the sparse regime (i.e., with $\leq 30\%$ of edges seen during training).

Q2: Does pre-training improve performance in the sparse setting? We next investigate the utility of standard pre-training. Table 2 shows the performance of VGAE models with and without pre-training. In both settings, the models are fine-tuned and tested on 10% of the graphs in each dataset, but in the pre-training setting the VGAE models are also pre-trained on 80% of the graphs in each dataset (with the remaining 10% used for validation purposes). Interestingly, pre-training provides only marginal benefits for the VGAE model, providing only a 0.1% relative improvement in AUC on average and even hurting performance in several settings.

Q3: Does the Meta-Graph approach provide meaningful performance gains? In Table 2, we also investigate the performance of different meta-learning approaches. Here, we can see that meta learning is a very effective strategy to overcome the challenge of sparse data and that our novel Meta-Graph approach, in particular, achieves very strong performance. Indeed, in the sparse regime where at most 30% of the edges are available for training, Meta-Graph achieves an average relative improvement of 5.3% in AUC compared to pre-training and an average relative improvement of 4.9% compared to the standard MAML algorithm. In addition to improving overall performance, we also investigate how the meta learning approaches impact the learning dynamics. Figure 4.A shows representative learning curves from the PPI data. (See Appendix B for full results on other datasets.) From these results, we can see that the Meta-Graph approach leads to fast convergence, often converging only after a few gradient updates. This result highlights that Meta-Graph could also be useful for facilitating fast convergence on newly acquired datasets.

4.2.1 Additional ablations and analysis on Meta-Graph

Section 3.4.1 introduced three variants of the Meta-Graph modulation function, which were treated as a hyperparameter in the above analysis. Figure 4.B shows representative results—again using the

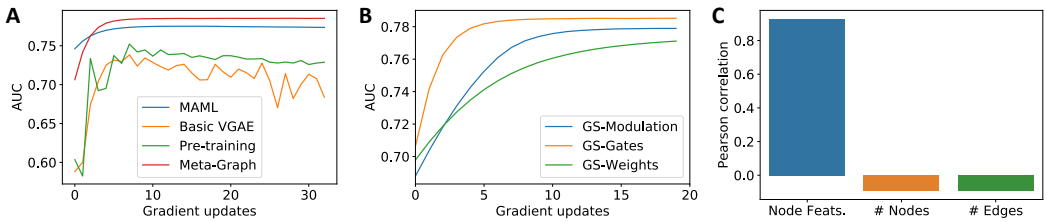

Figure 4: **A**, Validation AUC performance of different pre-training methods on the PPI datasets. **B**, Validation AUC comparison of different variants of Meta-Graph on PPI. **C**, Pearson correlation between graph properties and the Meta-Graph graph signatures on FirstMM graphs.

PPI dataset (with additional results in Appendix B)—comparing the the learning dynamics of Meta-Graph for these different modulation functions. We can see that simple modulation and GS-Gating are superior to GS-Weights after a few gradient steps, which was consistent across datasets.

Lastly, we investigated the signals that Meta-Graph uses to generate conditional parameter intializations. To investigate this, we treat the output of the signature function ($s_{\mathcal{G}} = \psi(\mathcal{G})$) as a vector and compute the cosine similarity between all pairs of graph signatures in the training set. We similarly compute three pairwise graph statistics—namely, the cosine similarity between average node features, the difference in number of nodes, and the difference in number of edges. Finally, we compute the Pearson correlation between the pairwise graph signature similarities and these other pairwise statistics. Figure 4.C shows representative Pearson correlations on the FirstMM graphs (with additional results in Appendix B). We find strong positive correlation in terms of Pearson correlation coefficient between node features and the output of the signature function for both datasets, indicating that the graph signature function is highly sensitive to node feature information.

5 Related Work

Link prediction. Historically, link prediction methods have utilized topological graph features (e.g., the Adamic-Adar score [1]). Other approaches include matrix factorization methods [25] and more recently deep learning and graph neural network based approaches [11, 40, 42]. Previous work in this space generally focuses in link prediction over a single dense graph, whereas we focus on the challenging setting of link prediction from sparse data.

Meta-learning. Meta learning has been effective in few shot learning tasks with a few notable approaches broadly classified into metric based approaches [39, 35, 19], augmented memory [34, 15, 28] and optimization based approaches [8, 21]. Recently, there are several works that lie at the intersection of meta-learning for few-shot classification and graph based learning [23, 33]. However, both these methods are restricted to the image domain and are not applicable to link prediction.

Few-shot relation prediction in knowledge graphs. Another related line of work considers the task of few shot relation prediction in knowledge graphs [5, 41]. A key distinction between the few shot relation setting and the one which we consider in this work is that we focus on link prediction in sparse simple graphs, while in the knowledge graph setting the challenge is generalizing to new types of relations within a single graph.

6 Conclusion

This work investigated the task of link prediction from sparse graph data. Overall, we found that a GNN-based method was strongest in the sparse data regime, while simpler approaches (e.g., the Adamic-Adar score) could achieve superior performance on dense graph data. We then proposed and investigated a meta-learning approach—termed Meta-Graph—which allows a GNN-based link prediction model to be effectively pre-trained using multiple graph datasets. We found that this meta-learning approach achieved the strongest results in the sparse data regime. In terms of limitations and directions for future work, one key limitation is that our approach works only on simple graphs. Extending our analysis and methodology to multi-relational data is an important open direction.

Statement of Broader Impact

This work is concerned with the development of techniques for link prediction from sparse graph data. In terms of potential positive societal impact as well as risk, it is important to consider some of the popular real-world applications of this technology.

In terms of potentially positive societal impact, one major use case for the methods discussed in this work is computational treatment design, where link prediction methods are used to recommend possible therapeutics given a sparse biomedical interaction graph [44]. Many biological datasets are sparse, so the techniques developed here could help accelerate the development of therapeutics when only sparse data is available. Of course, while generally positive, computational treatment design is a difficult and an inherently sensitive challenge. Thus, we urge and expect all researchers to abide by the academic and professional norms regarding the vetting of computational predictions by domain experts and adherence to ethical guidelines regarding the use of therapeutics in human and animal trials.

However, at the same time, the other popular application of these technologies is social recommendation systems, which are known to be fraught with societal risks due to bias, fairness, and the creation of highly polarized social dynamics [6]. Recommender systems are a popular application area in machine learning. Nonetheless, we urge researchers building off our work in this this application domain to consider the ethical aspects of recommender systems [26] and to consider using algorithmic tools that can mitigate issues, such as bias and fairness in these systems [3].

References

[1] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[2] Luca Maria Aiello, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Friendship prediction and homophily in social media. *ACM Trans. Web*, 6(2):9:1–9:33, June 2012.

[3] Avishek Joey Bose and William Hamilton. Compositional fairness constraints for graph embeddings. *arXiv preprint arXiv:1905.10674*, 2019.

[4] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. *arXiv preprint arXiv:1906.12192*, 2019.

[5] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. Meta relational learning for few-shot link prediction in knowledge graphs. *arXiv preprint arXiv:1909.01515*, 2019.

[6] Pranav Dandekar, Ashish Goel, and David T Lee. Biased assimilation, homophily, and the dynamics of polarization. *Proceedings of the National Academy of Sciences*, 110(15):5791–5796, 2013.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[9] Sorabh Gandhi, Subhash Suri, and Emo Welzl. Catching elephants with mice: sparse sampling for monitoring sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):1–27, 2010.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[12] W. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.

[13] W.L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *arXiv preprint, arXiv:1603.04467*, 2017.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[15] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[16] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[19] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[20] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[21] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. *arXiv preprint arXiv:1801.05558*, 2018.

[22] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[23] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning to propagate for graph meta-learning. In *Advances in Neural Information Processing Systems*, 2019.

[24] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[25] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 437–452, Berlin, Heidelberg, 2011. Springer-Verlag.

[26] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi. Recommender systems and their ethical challenges. *AI & SOCIETY*, pages 1–11, 2020.

[27] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

[28] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[29] Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In *Online Proceedings of the Eleventh Workshop on Mining and Learning with Graphs*, pages 0–6, 2013.

[30] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *International conference on trust management*, pages 224–239. Springer, 2005.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[33] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[34] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[35] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[36] Florian Strub, Mathieu Seurin, Ethan Perez, Harm De Vries, Jérémie Mary, Philippe Preux, and Aaron CourvilleOlivier Pietquin. Visual reasoning with multi-hop feature modulation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[37] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[38] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[39] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[40] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[41] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. *arXiv preprint arXiv:1808.09040*, 2018.

[42] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

[43] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

[44] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, 50:71–91, 2019.

7 Appendix

7.1 A: Ego-Aminer Dataset Construction

To construct the Ego-Aminer dataset we first create citation graphs from different fields of study. We then select the top 100 graphs in terms number of nodes for further pre-processing. Specifically, we take the 5-core of each graph ensuring that each node has a minimum of 5-edges. We then construct ego networks by randomly sampling a node from the 5-core graph and taking its two hop neighborhood. Finally, we remove graphs with fewer than 100 nodes and greater than 20000 nodes which leads to a total of 72 graphs as reported in Table 1.

7.2 B: Additional Results

We list out complete results when using larger sets of training edges for PPI, FIRSTMM DB and Ego-Aminer datasets. We show the results for two metrics i.e. Average AUC across all test graphs. As expected, we find that the relative gains of Meta-Graph decrease as more and more training edges are available.

PPI Convergence	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.795**	**0.831**	**0.846**	**0.853**	0.848	0.853	0.855
MAML	0.745	0.820	0.840	0.852	**0.854**	0.856	**0.863**
Random	0.578	0.651	0.697	0.729	0.756	0.778	0.795
VGAE	0.738	0.786	0.801	0.817	0.827	0.837	0.836
VGAE+pretrain	0.752	0.8010	0.821	0.832	0.818	**0.856**	0.841
Adamic	0.540	0.623	0.697	0.756	0.796	0.827	0.849
MAML-MLP	0.603	0.606	0.606	0.606	0.604	0.604	0.605
Deepwalk	0.664	0.673	0.694	0.727	0.731	0.747	0.761
DeepWalk+Feat.	0.678	0.765	0.800	0.823	0.828	0.828	0.830
MLP+Feat.	0.682	0.685	0.691	0.702	0.720	0.700	0.703

Table 3: AUC Convergence results for PPI dataset for training edge splits

PPI-5 updates	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.795**	**0.829**	**0.847**	**0.853**	0.848	**0.854**	**0.856**
MAML	0.756	0.837	0.840	0.852	**0.855**	**0.855**	**0.856**
VGAE	0.600	0.697	0.717	0.784	0.814	0.779	0.822
VGAE+pretrain	0.582	0.727	0.774	0.702	0.804	0.718	0.766
MAML-MLP	0.603	0.606	0.603	0.604	0.603	0.606	0.605

Table 4: 5-gradient update AUC results for PPI for training edge splits

FirstMM DB Convergence	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.782**	**0.786**	0.783	**0.781**	0.760	0.746	0.739
MAML	0.776	0.782	**0.793**	**0.785**	**0.791**	0.663	0.788
Random	0.742	0.732	0.720	0.714	0.705	0.698	0.695
VGAE	0.740	0.710	0.734	0.722	0.712	0.710	0.698
VGAE+pretrain	0.752	0.735	0.723	0.734	0.749	0.700	0.695
Adamic	0.504	0.519	0.544	0.573	0.604	0.643	0.678
Deepwalk	0.487	0.473	0.510	0.608	0.722	**0.832**	**0.911**
DeepWalk+Feat.	0.501	0.532	0.582	0.657	0.724	0.836	0.889
MLP+Feat.	0.672	0.683	0.674	0.690	0.710	0.706	0.716

Table 5: AUC Convergence results for FIRSTMM DB dataset for training edge splits

FirstMM DB 5 updates	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.773**	**0.767**	**0.743**	**0.759**	0.742	**0.732**	0.688
MAML	0.763	0.750	0.624	**0.776**	**0.759**	0.663	**0.738**
VGAE	0.708	0.680	0.709	0.701	0.685	0.683	0.653
VGAE+pretrain	0.705	0.695	0.704	0.704	0.696	0.658	0.670

Table 6: 5-gradient update AUC results for FIRSTMM DB for training edge splits

Ego-Aminer Convergence	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.626**	**0.738**	**0.786**	**0.791**	**0.792**	**0.817**	**0.786**
MAML	0.561	0.662	0.667	0.682	0.720	0.741	0.768
Random	0.500	0.500	0.500	0.500	0.500	0.500	0.500
VGAE	0.548	0.621	0.673	0.702	0.652	0.7458	0.769
VGAE+pretrain	0.623	0.691	0.723	0.764	0.767	0.792	0.781
Adamic	0.515	0.549	0.597	0.655	0.693	0.744	0.772
Deepwalk	0.602	0.638	0.672	0.686	0.689	0.711	0.731
DeepWalk+Feat.	**0.627**	0.660	0.686	0.703	0.737	0.741	0.752
MLP+Feat.	0.622	0.651	0.682	0.697	0.720	0.720	0.716

Table 7: AUC Convergence results for Ego-Aminer dataset for training edge splits

8 C: Node Feature based Baseline

We now provide further details on node feature based baselines. In particular, we use two baselines the first of which uses representations learned by DeepWalk and concatenates them with the output an MLP. We also investigate only employing an MLP model which simply uses the available node features as input for both training and validation edges. With this baseline we primarily seek to identify if there is any additional gains achieved by using extra edges in the validation data which is used primarily in all meta-learning methods. That is to say, can we leverage extra edges without the need to meta-learn over sparse graphs. We refer to these models as DeepWalk + Feat and MLP + Feat respectively in Tables 3-5. Both these baselines predict edges in the graph using the dot product decoder as in Eqn 2. and the overall learning objective is to maximize the log likelihood of the data. We train both the MLPs in each respective baseline for 3000 epochs with Early stopping and also use the Adam optimizer with default hyperparameters.

Ego-Aminer 5 updates	10%	20%	30%	40%	50%	60%	70%
Meta-Graph	**0.620**	0.5850	**0.732**	0.500	**0.790**	**0.733**	0.500
MAML	0.500	0.504	0.500	0.500	0.519	0.500	0.500
VGAE	0.500	0.500	0.500	0.500	0.500	0.500	0.500
VGAE+pretrain	0.608	**0.675**	0.713	**0.755**	0.744	0.706	**0.671**

Table 8: 5-gradient update AUC results for Ego-Aminer for training edge splits