Neural Graph Revealers

Harsh Shrivastava and Urszula Chajewska

Microsoft Research, Redmond, USA
{hshrivastava,urszc}@microsoft.com

Abstract. Sparse graph recovery methods work well where the data follows their assumptions, however, they are not always designed for doing downstream probabilistic queries. This limits their adoption to only identifying connections among domain variables. On the other hand, Probabilistic Graphical Models (PGMs) learn an underlying base graph together with a distribution over the variables (nodes). PGM design choices are carefully made such that the inference and sampling algorithms are efficient. This results in certain restrictions and simplifying assumptions. In this work, we propose Neural Graph Revealers (`NGRs`) which attempt to efficiently merge the sparse graph recovery methods with PGMs into a single flow. The task is to recover a sparse graph showing connections between the features and learn a probability distribution over them at the same time. `NGRs` use a neural network as a multitask learning framework. We introduce *graph-constrained path norm* that `NGRs` leverage to learn a graphical model that captures complex non-linear functional dependencies between features in the form of an undirected sparse graph. `NGRs` can handle multimodal inputs like images, text, categorical data, embeddings etc. which are not straightforward to incorporate in the existing methods. We show experimental results on data from Gaussian graphical models and a multimodal infant mortality dataset by CDC.[1]

Keywords: Sparse Graph Recovery · Probabilistic Graphical Models.

1 Introduction and Related Work

Sparse graph recovery is an important tool to gain insights from data and is a widely researched topic [8,14,17]. Graph recovery algorithms discover the feature dependencies in the form of a sparse graph, $S_G \in \mathbb{R}^{D \times D}$, where S_G is the adjacency matrix over D features. Such graphs are useful for analyzing data from various domains: obtaining gene regulatory networks from single-cell RNA sequencing data [11,20,22], stock market data and automobile sensor networks for navigation purposes [7]. Other interesting applications consist of studying brain connectivity patterns in autistic patients [12], increasing methane yield in the anaerobic digestion process [19], gaining insights from the infant-mortality data by CDC [18] and multivariate timeseries segmentation [9]. These sparse graphs can be directed, undirected or have mixed-edge types.

[1] Software: `https://github.com/harshs27/neural-graph-revealers`

Many existing approaches make a simplifying assumption about the distribution in order to achieve sparsity [2,6,11]. Some use the deep unfolding technique to use the existing optimization algorithm as an inductive bias for designing deep learning architectures [15,16,12] but they all require supervision for learning.

We present an efficient algorithm, called Neural Graph Revealers (`NGRs`), that aspires to learn the dependency graph and the distribution over features without making any simplifying assumptions. Key contributions of this work are:

- Novel use of neural networks as a multi-task learning framework to model functional dependencies jointly for all features that enables richer & complex representation as compared to the state-of-the-art methods.
- Incorporate multimodal features like images, categorical data or embeddings.
- Training is unsupervised which facilitates and adoption to new domains.
- Efficient and scalable approach that can handle large number of features.
- Once learned, the `NGR` architecture becomes an instance of a Neural Graphical Model [18] and can be used for downstream probabilistic reasoning tasks.

Related Methods. Fig. 1 is an attempt to categorize different methods to

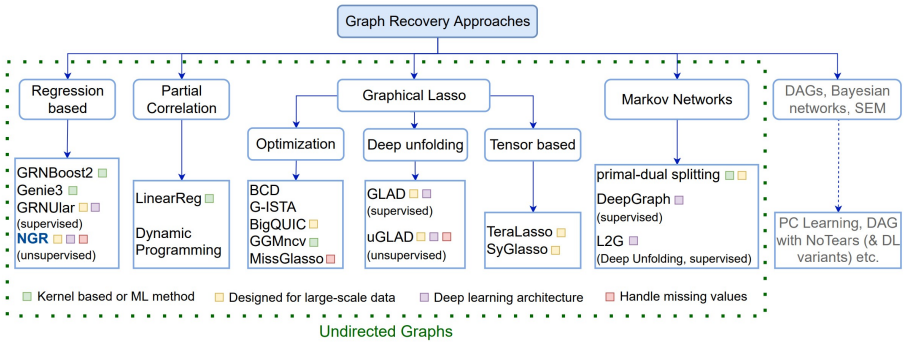

Fig. 1: **Graph Recovery approaches.** Neural Graph Revealers (`NGRs`) lie under the regression based algorithms. The algorithmsb listed here are representative of the sub-category and the list is not exhaustive and are discussed extensively in [17].

recover graphs. We primarily focus on methods developed for undirected graphs. We consider the input data $X \in \mathbb{R}^{M \times D}$ with D features and M samples. For each of the D features, the *Regression based formulation* fits a regression function with respect to all the other features, $X_d = f_d(X_{\{D\}\setminus d}) + \epsilon$, where ϵ is an additive noise term. After fitting the regression, based on the choice of functions $f_d's$, the algorithms determine the dependency of the features. These approaches have been very successful for the task of recovering Gene Regulatory Networks. For instance, GENIE3 [24] modeled each f_d to be random forest model and GRNBoost2 [11] combined random forests with gradient boosting technique to achieve superior performance among others [1]. Then, neural network based representation like GRNUlar [22] were developed, which also utilized the idea of using NNs as a multitask learning setup. This method is architecturally quite close to our

formulation, although the major difference with NGRs is that GRNUlar needs supervision for training. Most of these methods were developed for numerical input and it is not straightforward to extend them for multimodal features. NGRs on the other hand provide a flexible approach to model multimodal inputs.

2 Neural Graph Revealers

We assume we are given the input data X with D features and M samples. The task is to recover a sparse graph represented by its adjacency matrix form $S_G \in \mathbb{R}^{D\times D}$. In the recovered undirected graph obtained by any regression based approach, each feature or graph node is a function of its immediate (one-hop) neighbors, as shown in Fig. 2(right). In this section, we describe our proposed NGRs along with its potential extensions.

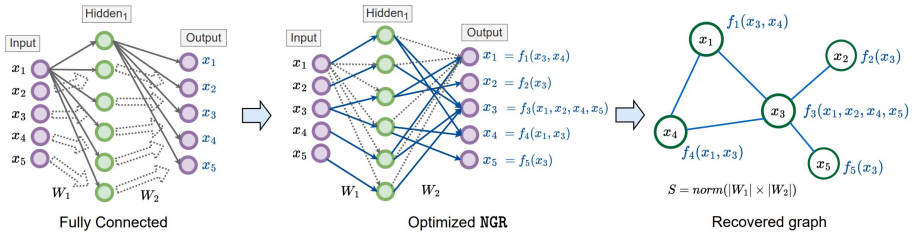

Fig. 2: *Workflow of NGRs.* (left) We start with a fully connected Neural Network (MLP here) where both the input and output are the given features $x_i's$. Viewing NN as a multitask learning framework indicates that the output features are dependent on all the input features in the initial fully connected setting. (middle) The learned NGR optimizes the network connections to fit the regression on the input data as well as satisfy the sparsity constraints, refer Eq. 2. If there is a path from the input feature to an output feature, that indicates a dependency, potentially non-linear, between them. The bigger the size of NN (number of layers, hidden unit dimensions) the richer will be the functional representation. Note that not all the weights of the MLP (those dropped during training in grey-dashed lines) are shown for the sake of clarity. (right) The sparse dependency graph between the input and output of the MLP reduces to the normalized weight matrix product $S_g = \text{norm}\,(|W_1| \times |W_2|)$.

The architecture of NGR is an MLP that takes in input features and fits a regression to get the same features as an output, shown in Fig. 2(left). We start with a fully connected network. Some edges are dropped during training. We view the trained neural network as a glass-box where a path to an output unit (or neuron) from a set of input units means that the output unit is a function of those input units.

In order to obtain a graph, for every feature X_d, we want to find the most relevant features that have a direct functional influence on X_d. This task becomes increasingly complex as we need to evaluate all possible combinations which can be computationally tedious. Fitting the regression of NGRs, refer Fig. 2(middle),

can be seen as doing *multitask learning* that simultaneously optimizes for the functional dependencies of all the features.

Main design challenges to consider while fitting the `NGR` regression are:
(A) How to avoid self-dependencies among features, eg. $X_d \rightarrow X_d, \forall d \in \{D\}$?
(B) How do we efficiently induce sparsity among the paths defined by the MLP?

2.1 Optimization

We denote a NN with L number of layers with the weights $\mathcal{W} = \{W_1, W_2, \cdots, W_L\}$ and biases $\mathcal{B} = \{b_1, b_2, \cdots, b_L\}$ as $f_{\mathcal{W},\mathcal{B}}(\cdot)$ with non-linearity not mentioned explicitly. In our implementation, we experimented with multiple non-linearities and found that ReLU fits well with our framework. Applying the NN to the input $X_{\mathcal{D}}$ evaluates the following mathematical expression, $f_{\mathcal{W},\mathcal{B}}(X_{\mathcal{D}}) = \text{ReLU}(W_L \cdot (\cdots(W_2 \cdot \text{ReLU}(W_1 \cdot X_{\mathcal{D}} + b_1) + b_2)\cdots) + b_L)$. The dimensions of the weights and biases are chosen such that the neural network input and output units are equal to $\mathcal{D}$ while the hidden layers dimension H remains a design choice. In our experiments, we found a good initial choice of $H = 2|\mathcal{D}|$, then eventually based on the loss on validation data, one can adjust the dimensions.

Our task is to design the `NGR` objective function such that it can jointly discover the feature dependency graph constraints (A) & (B) along with fitting the regression on the input data. We observe that the product of the weights of the neural networks $S_{nn} = \prod_{l=1}^{L} |W_l| = |W_1| \times |W_2| \times \cdots \times |W_L|$ gives us path dependencies between the input and the output units. We note that if $S_{nn}[x_i, x_o] = 0$ then the output unit x_o does not depend on input unit x_i.

Graph-constrained path norm. We introduce a way to map NN paths to a predefined graph structure. Consider the matrix S_{nn} that maps the paths from the input units to the output units as described above. Assume we are given a graph with adjacency matrix $S_g \in \{0,1\}^{D \times D}$. The graph-constrained path norm is defined as $\mathcal{P}_c = \left\| S_{nn} * S_g^c \right\|_1$, where S_g^c is the complement of the adjacency matrix $S_g^c = J_D - S_g$ with $J_D \in \{1\}^{D \times D}$ being an all-ones matrix. The operation $Q * V$ represents the Hadamard operator which does an element-wise matrix multiplication between the same dimension matrices Q & V. This term is used to enforce penalty to fit a particular predefined graph structure, S_g.

We use this formulation of MLPs to model the constraints along with finding the set of parameters $\{\mathcal{W}, \mathcal{B}\}$ that minimize the regression loss expressed as the Euclidean distance between $X_{\mathcal{D}}$ to $f_{\mathcal{W},\mathcal{B}}(X_{\mathcal{D}})$. Optimization becomes

$$\arg\min_{\mathcal{W},\mathcal{B}} \sum_{k=1}^{M} \left\| X_{\mathcal{D}}^k - f_{\mathcal{W},\mathcal{B}}(X_{\mathcal{D}}^k) \right\|_2^2, \qquad s.t. \quad \text{sym}(S_{nn}) * S_{\text{diag}} = 0 \tag{1}$$

where, $\text{sym}(S_{nn}) = \left(\|S_{nn}\|_2 + \|S_{nn}\|_2^T \right)/2$ converts the path norm obtained by the NN weights product, $S_{nn} = \prod_{l=1}^{L} |W_l|$, into a symmetric adjacency matrix and $S_{\text{diag}} \in \mathbb{R}^{D \times D}$ represents a matrix of zeroes except the diagonal entries that are 1. Constraint (A) is thus included as the constraint term in Eq. 1. To satisfy the constraint (B), we include an ℓ_1 norm term $\|\text{sym}(S_{nn})\|_1$ which will introduce

sparsity in the path norms. Note that this second constraint enforces sparcity of *paths*, not individual *weights*, thus affecting the entire network structure.

We model these constraints as Lagrangian terms which are scaled by a log function. The log scaling is done for computational reasons as sometimes the values of the Lagrangian terms can go very low. The constants λ, γ act as a tradeoff between fitting the regression term and their corresponding constraints. The optimization formulation to recover a valid graph structure becomes

$$\arg\min_{\mathcal{W},\mathcal{B}} \sum_{k=1}^{M} \left\| X_{\mathcal{D}}^{k} - f_{\mathcal{W},\mathcal{B}}(X_{\mathcal{D}}^{k}) \right\|_2^2 + \lambda \left\| \mathrm{sym}(S_{nn}) * S_{\mathrm{diag}} \right\|_1 + \gamma \left\| \mathrm{sym}(S_{nn}) \right\|_1 \quad (2)$$

where we can optionally add log scaling to the structure constraint terms. Essentially, we start with a fully connected graph and then the Lagrangian terms induce sparsity in the graph. Alg. 1 describes the procedure to learn the `NGR` architecture based on optimizing the Eq. 2. We note that the optimization and the graph recovered depend on the choices of the penalty constants λ, γ. Since our loss function contains multiple terms, the loss-balancing technique introduced in [13], can be utilized to get a good initial value of the constants. Then, while running optimization, based on the regression loss value on a held-out validation data, the values of penalty constants can be appropriately chosen.

2.2 Modeling multi-modal data

It is common to encounter multi-modal data in real-world datasets. For instance, ICU patient records can have information about body vitals (numerical, categorical), nurse notes (natural language) and maybe associated X-rays (images). In this section, we propose two different ways to include multi-modal input data in the `NGR` formulation.

Algorithm 1: Learning `NGRs`

Function `NGR-training(`X`):`
- $f_{\mathcal{W}^0} \leftarrow$ Fully connected MLP
- **For** $e = 1, \cdots, E$ **do**
  - Xb $\leftarrow X$ (sample a batch)
  - $\mathcal{L} = \sum_{k=1}^{M} \left\| \mathrm{Xb}_{\mathcal{D}}^{k} - f_{\mathcal{W},\mathcal{B}}(\mathrm{Xb}_{\mathcal{D}}^{k}) \right\|^2 - \lambda \left\| \mathrm{sym}(S_{nn}) * S_{\mathrm{diag}} \right\|_1 - \gamma \left\| \mathrm{sym}(S_{nn}) \right\|_1$
  - $\mathcal{W}^e, \mathcal{B}^e \leftarrow$ backprop $\mathcal{L}$ with Adam optimizer
- $\{\mathcal{W}^E\} \leftarrow f_{\mathcal{W},\mathcal{B}}^{E}$
- $\mathcal{G} \leftarrow \mathrm{sym}\left(\prod_{l=1}^{L} |W_l^E|\right)$
- **return** $\mathcal{G}, f_{\mathcal{W}^E}$

(I) Using projection modules. Fig. 3 gives a schematic view of including projection modules to the base architecture described in Fig. 2. W.l.o.g. we can consider that each of the D inputs is an embedding in $x_i \in \mathbb{R}^I$ space. For example, given an image, one way of getting a corresponding embedding can be to use the latent layer of a convolutional neural network based autoencoder. We convert all the input x_i nodes in the `NGR` architecture to hypernodes, where each hypernode contains the embedding vector. Consider a hypernode that contains an embedding vector of size E and if an edge is connected to the hypernode, then that edge is connected to all

the E units of the embedding vector. For each of these input hypernodes, we define a corresponding encoder embedding $e_i \leftarrow \text{enc}_i(x_i), \forall e_i \in \mathbb{R}^E$, which can be designed specific to that particular input embedding. Similarly, we apply the encoder modules to all the x_i hypernodes and obtain the e_i hypernodes. Same procedure is followed at the decoder end, where $x_i \leftarrow \text{dec}_i(d_i), \forall d_i \in \mathbb{R}^O$. The `NGR` graph discovery optimization reduces to discovering the connectivity pattern using the path norms between hypernodes e_i's and d_i's. A slight modification to the graph-constrained path norm is needed to account for the hypernodes. The S_{diag} term will now represent the connections between the hypernodes, so $S_{diag} \in \{0,1\}^{DE \times DO}$ with ones in the block diagonals. We can include the

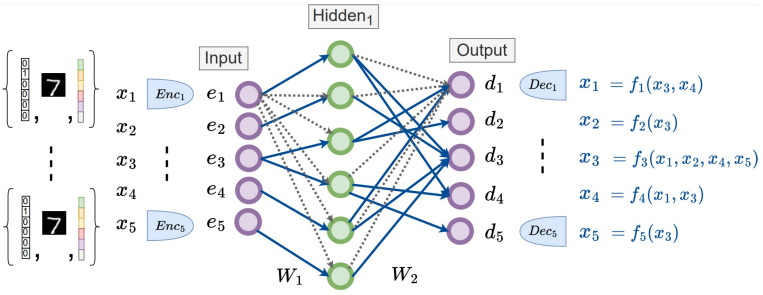

Fig. 3: *Multi-modal data handling with Projection modules.* The input $\mathbf{X}$ can be one-hot (categorical), image or in general an embedding (text, audio, speech and other data types). Projection modules (encoder + decoder) are used as a wrapper around the `NGR` base architecture. The architecture of the projection modules depends on the input data type and users' design choices. Note that the output of the encoder can be more than 1 unit (e_1 can be a hypernode) and the corresponding adjacency matrix S_{diag} of the graph-constrained path norm can be adjusted. Similarly, the decoder side decoder side of the `NGR` architecture is updated. The remaining details are similar to Fig. 2

projection modules in the regression term of the `NGR` objective, while the structure learning terms will remain intact

$$\underset{\mathcal{W},\mathcal{B},\text{proj}}{\arg\min} \sum_{k=1}^{M} \left\| X_{\mathcal{D}}^k - f_{\mathcal{W},\mathcal{B},\text{proj}}(X_{\mathcal{D}}^k) \right\|_2^2 + \lambda \left\| \text{sym}(S_{nn}) * S_{\text{diag}} \right\|_1 + \gamma \left\| \text{sym}(S_{nn}) \right\|_1 \tag{3}$$

where the proj are the parameters of the encoder and decoder projection.

(II) Using graph-constrained path norm (`GcPn`). Fig. 4 shows that we can view the connections between the D hypernodes of the input embedding $x_i \in \mathbb{R}^I$ (where I is the dimensionality of the input) to the corresponding input of the encoder layer $e_i \in \mathbb{R}^E$ (with E being the dimensionality of the embedding) as a graph. We represent each input layer to the encoder layer connections by $S_{\text{enc}} \in \{0,1\}^{DI \times DE}$, where there is a $S_{\text{enc}}[x_i, e_j] = 1$ if the (x_i, e_j) hypernodes are connected. So, if we initialize a fully connected neural network (or MLP) between the input layer and the encoder layer, we can utilize the `GcPn` penalty

function to map the paths from the input units to the encoder units to satisfy the graph structure defined by S_{enc}. Similar exercise is replicated at the decoder end to obtain S_{dec}. This extension of the `GcPn` to multi-modal data leads us to the following Lagrangian based formulation of the optimization objective

$$\begin{aligned}\underset{\mathcal{W}_{\text{enc}},\mathcal{W},\mathcal{B},\mathcal{W}_{\text{dec}}}{\arg\min} \sum_{k=1}^{M} \left\| X_{\mathcal{D}}^{k} - f_{\mathcal{W}_{\text{enc}},\mathcal{W},\mathcal{B},\mathcal{W}_{\text{dec}}}(X_{\mathcal{D}}^{k}) \right\|_2^2 + \lambda \left\| \text{sym}(S_{nn}) * S_{\text{diag}} \right\|_1 \quad (4) \\ + \gamma \left\| \text{sym}(S_{nn}) \right\|_1 + \eta \left\| \text{sym}(S_{nn}^{e}) * S_{\text{enc}} \right\|_1 + \beta \left\| \text{sym}(S_{nn}^{d}) * S_{\text{dec}} \right\|_1\end{aligned}$$

where $f_{\mathcal{W}_{\text{enc}},\mathcal{W},\mathcal{B},\mathcal{W}_{\text{dec}}}(\cdot)$ represents the entire end-to-end MLP including the encoder and decoder mappings, $S_{nn}^{e} = \prod_{l=1}^{L^e} |W_l| = |W_1| \times |W_2| \times \cdots \times |W_{L^e}|$ captures the path dependencies in the encoder MLP with L^e layers, $S_{nn}^{d} = \prod_{l=1}^{L^d} |W_l| = |W_1| \times |W_2| \times \cdots \times |W_{L^d}|$ captures the path dependencies in the decoder MLP with L^d layers. The Lagrangian constants $\lambda, \gamma, \eta, \beta$ are initialized in the same manner as outlined in Sec. 2.1. We note the advantage of using the `GcPn` penalties to enable **soft enforcing** of the path constraint requirements between the input and output units of the neural networks. We recommend the `GcPn` based approach (II) as the implementation is straightforward and it is highly scalable and can handle large embedding sizes.

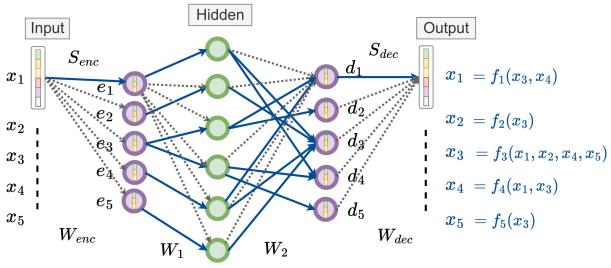

Fig. 4: *Multi-modal data handling with Graph-constrained path norm.* W.l.o.g. we consider an input $\mathbf{X}$ to be embeddings that can come from text, speech and other data types. We extend the idea of applying `GcPn` to the encoder MLP and the decoder MLP. We initilialize a fully connected MLP and then using the `GcPn` penalties, we capture the desired input to output unit path dependencies after optimizing the Eq. 4. NN nodes containing embeddings are shown as hypernodes. We use the concept of a hypernode to convey that all units of the embedding vector within the hypernode are considered a single unit when deciding the edge connections defining a graph. The encoder and decoder MLPs are used as a wrapper around the `NGR` base architecture. The remaining details are similar to the ones described in Fig. 2.

2.3 Representation as a probabilistic graphical model

Once the sparse graph is recovered, the learned architecture of `NGR` represents functional dependencies between the features. A beneficial next step for wider

adoption will be the ability to model the entire joint probability distribution of the features. This type of representation of the functional dependencies based on neural networks has been recently explored in [18] and is calle Neural Graphical Model (`NGM`). It is a type of a Probabilistic Graphical Model that utilizes neural networks and a pre-defined graph structure between features to learn complex non-linear underlying distributions. Additionally, it can model multi-modal data and have efficient inference and sampling algorithms. The inference capability can be used to estimate missing values in data. The learned `NGR` model can be viewed as an instance of an `NGM`.

3 Experiments

3.1 Learning Gaussian Graphical Models

We explored the `NGR`'s ability to model Gaussian Graphical Models (GGM). To create a GGM, we used a chain-graph structure and then defined a precision matrix over it by randomly initializing the entries $\mathcal{U}(0.5, 1)$ with random signs. The diagonal entries of the precision matrix were chosen such that it is positive semi-definite. Samples were obtained from the GGM and were used as input to the `NGR` for recovering the underlying graphical model. Fig. 5 shows the GGM and the corresponding trends discovered after fitting a `NGR`. We used a `NGR` with a single hidden layer and its dimension $H = 100$. Table 1 shows the graph recovery results by running `NGR` on varying number of samples obtained using the Gaussian graphical model. As expected, the results improve as we increase the number of samples and thereby `NGRs` are capable of representing Gaussian graphical models.

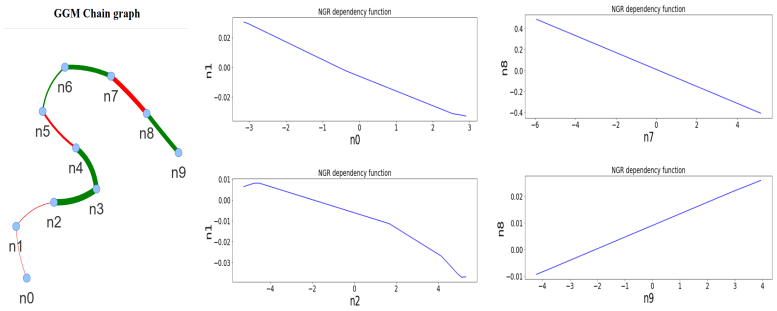

Fig. 5: *Modeling GGMs using* ***`NGRs`****.* (left) The Conditional Independence graph [17] for the chain structure used to generate the data. Positive partial correlations between the nodes are shown in green, while the negative partial correlations in red. These correlations show direct dependence or, in other words, the dependence is evaluated conditioned on all the other nodes. (middle, right) Pairwise dependence functions learned by the `NGR`. We observe that the `NGR` slopes match the trend in the GGM graph. This shows that the dependency plots learned comply with the desired behaviour as shown in the color of the partial correlation edges.

Table 1: The recovered CI graph from NGR is compared with the ground truth CI graph defined by the underlying GGMs precision matrix with $D = 10$ nodes, chain graph as shown in Fig. 5. Area under the ROC curve (AUC) and area under the precision-recall curve (AUPR) values for 5 runs are reported.

Samples	AUPR	AUC
100	0.34 ± 0.03	0.67 ± 0.05
500	0.45 ± 0.10	0.79 ± 0.03
1000	0.63 ± 0.11	0.90 ± 0.03

3.2 Infant Mortality data analysis

The infant mortality dataset we used is based on CDC Birth Cohort Linked Birth – Infant Death Data Files [23]. It describes pregnancy and birth variables for all live births in the U.S. together with an indication of an infant's death (and its cause) before the first birthday. We used the data for 2015 (latest available), which includes information about 3,988,733 live births.

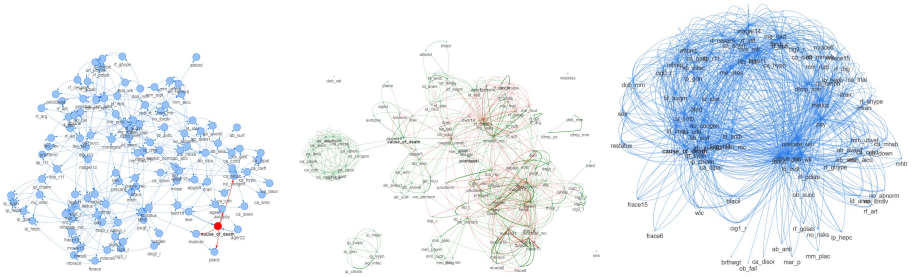

Fig. 6: *Graphs recovered for the Infant Mortality 2015 data.* (left) The Bayesian network graph learned using score-based method, (middle) the CI graph recovered by uGLAD and (right) the NGR graph. For NGR, we applied a threshold to retain top relevant edges.

Recovered graphs. We recovered the graph strucure of the dataset using NGR, uGLAD [19] and Bayesian network package bnlearn [14] with Tabu search and AIC score. The graphs are shown in Fig. 6. All variables were converted to categorical for bnlearn structure learning and inference as it does not support networks containing both continuous and discrete variables. In contrast, uGLAD and NGRs are both equipped to work with mixed types of variables and were trained on the dataset prior to conversion. It is interesting to observe that although there are some common clusters in all three graphs, each graph has a different extent of inter-cluster connections. The three different graph recovery methods are based on different distribution assumptions, training methodology, way of handling multimodal data, which leads to different connectivity patterns. This dataset and corresponding BN and uGLAD graph analysis are discussed in [18].

NGR architecture details. Since we have mixed input data types, real and categorical data, we utilize the NGR multimodal architecture's neural view given in Fig. 4. We used a 2-layer neural view with $H = 1000$. The categorical input was converted to its one-hot vector representation and added to the real features

Methods	Gestational age (ordinal, weeks)		Birthweight (continuous, grams)	
	MAE	RMSE	MAE	RMSE
Logistic Regression	1.512 ± 0.005	3.295 ± 0.043	N/A	N/A
Bayesian network	$\mathbf{1.040 \pm 0.003}$	2.656 ± 0.027	N/A	N/A
EBM	1.313 ± 0.002	2.376 ± 0.021	$\mathbf{345.21 \pm 1.47}$	$\mathbf{451.59 \pm 2.38}$
`NGM` w/full graph	1.560 ± 0.067	2.681 ± 0.047	394.90 ± 11.25	517.24 ± 11.51
`NGM` w/BN graph	1.364 ± 0.025	2.452 ± 0.026	370.20 ± 1.44	484.82 ± 1.88
`NGM` w/`uGLAD` graph	1.295 ± 0.010	$\mathbf{2.370 \pm 0.025}$	371.27 ± 1.78	485.39 ± 1.86
`NGR`	1.448 ± 0.133	2.493 ± 0.100	369.68 ± 1.14	483.96 ± 1.56

Table 2: Comparison of predictive accuracy for gestational age and birthweight.

Methods	Survival (binary)		Cause of death (multivalued, majority class frequency 0.9948)			
			micro-averaged		macro-averaged	
	AUC	AUPR	Precision	Recall	Precision	Recall
Logistic Regression	0.633 ± 0.004	0.182 ± 0.008	0.995 ± 7.102e-05	0.995 ± 7.102e-05	0.136 ± 0.011	0.130 ± 0.002
Bayesian network	0.655 ± 0.004	0.252 ± 0.007	0.995 ± 7.370e-05	0.995 ± 7.370e-05	0.191 ± 0.008	0.158 ± 0.002
EBM	0.680 ± 0.003	$\mathbf{0.299 \pm 0.007}$	0.995 ± 5.371e-05	0.995 ± 5.371e-05	0.228 ± 0.014	0.166 ± 0.002
`NGM` w/full graph	0.721 ± 0.024	0.197 ± 0.014	0.994 ± 1.400e-05	0.994 ± 1.400e-05	$\mathbf{0.497 \pm 7.011}$e-06	$\mathbf{0.500 \pm 1.000}$e-06
`NGM` w/BN graph	0.752 ± 0.012	0.295 ± 0.010	0.995 ± 4.416e-05	0.995 ± 4.416e-05	$\mathbf{0.497 \pm 2.208}$e-05	$\mathbf{0.500 \pm 1.000}$e-06
`NGM` w/`uGLAD` graph	0.726 ± 0.020	0.269 ± 0.018	0.995 ± 9.735e-05	0.995 ± 9.735e-05	$\mathbf{0.497 \pm 4.868}$e-05	$\mathbf{0.500 \pm 1.000}$e-06
`NGR`	$\mathbf{0.770 \pm 0.009}$	0.269 ± 0.030	$\mathbf{0.995 \pm 3.357}$e-05	$\mathbf{0.995 \pm 3.357}$e-05	$\mathbf{0.497 \pm 1.678}$e-05	$\mathbf{0.500 \pm 1.000}$e-06

Table 3: Comparison of predictive accuracy for 1-year survival and cause of death. Note: recall set to zero when there are no labels of a given class, and precision set to zero when there are no predictions of a given class.

which gave us roughly ~ 500 features as input. `NGR` was trained on the 4 million data points with $D = 500$ using 64 CPUs within 4 hours.

Inference accuracy comparison. Infant mortality dataset is particularly challenging due to the data skewness. For instance, the cases of infant death during the first year of life are rare compared to cases of surviving infants. Getting good performance on imbalanced data is a challenging problem and multiple techniques have been developed to assist existing learning algorithms [5,21,3]. We do not report results on applying these techniques as it would be out of scope for this work. Since `NGR` becomes an instance of a Neural Graphical Model, we also include comparisons of `NGMs` that use base graphs obtained from Bayesian networks, CI graph from `uGLAD` and also show the results on using a fully connected graph which basically avoids using any internal graph structure. We compared prediction for four variables of various types: gestational age (ordinal, expressed in weeks), birthweight (continuous, specified in grams), survival till 1st birthday (binary) and cause of death ('alive', 10 most common causes of death with less common grouped in category 'other' with 'alive' indicated for 99.48% of infants). We compared with other prediction methods like logistic regression, Bayesian networks, Explainable Boosting Machines (EBM) [4,10] and report 5-fold cross validation results.

Tables 2 and 3 demonstrate that `NGR` models are more accurate than logistic regression, Bayesian Networks and on par with EBM models for categorical and ordinal variables. They performance is similar to the `NGM` models with different input base graphs highlighting that learning a `NGR` graph is can help us gain new insights. We note an additional advantage of `NGRs` and `NGMs` in general: we just need to train a single model and their inference capability can be leveraged to

output predictions for multiple tasks listed here. For the case of EBM and LR models, we had to train a separate model for each outcome variable evaluated.

4 Conclusions

We address the important problem of doing sparse graph recovery and querying the corresponding graphical model. The existing graph recovery algorithms make simplifying assumptions about the feature dependency functions in order to achieve sparsity. Some deep learning based algorithms achieve non-linear functional dependencies but their architecture demands too many learnable parameters. Neural Graph Revealers leverage neural networks as a multitask learning framework to represent complex distributions and also introduces the concept of graph-constrained path norm to learn a sparse graphical model. The trained model can be used for probabilistic inference. Our experiments on infant mortality dataset demonstrate usefulness of `NGRs` to model complex multimodal input real-world problems.

References

1. Aluru, M., Shrivastava, H., Chockalingam, S.P., Shivakumar, S., Aluru, S.: EnGRaiN: a supervised ensemble learning method for recovery of large-scale gene regulatory networks. Bioinformatics (2021)
2. Belilovsky, E., Kastner, K., Varoquaux, G., Blaschko, M.B.: Learning to discover sparse graphical models. In: International Conference on Machine Learning. pp. 440–448. PMLR (2017)
3. Bhattacharya, S., Rajan, V., Shrivastava, H.: ICU mortality prediction: a classification algorithm for imbalanced datasets. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
4. Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1721–1730. ACM (2015)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002)
6. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**(3), 432–441 (2008)
7. Hallac, D., Park, Y., Boyd, S., Leskovec, J.: Network inference via the time-varying graphical lasso. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 205–213 (2017)
8. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine learning **20**(3), 197–243 (1995)
9. Imani, S., Shrivastava, H.: Are uGLAD? Time will tell! arXiv preprint arXiv:2303.11647 (2023)
10. Lou, Y., Caruana, R., Gehrke, J., Hooker, G.: Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 623–631. ACM (2013)

11. Moerman, T., Aibar Santos, S., Bravo González-Blas, C., Simm, J., Moreau, Y., Aerts, J., Aerts, S.: GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. Bioinformatics **35**(12), 2159–2161 (2019)
12. Pu, X., Cao, T., Zhang, X., Dong, X., Chen, S.: Learning to learn graph topologies. Advances in Neural Information Processing Systems **34** (2021)
13. Rajbhandari, S., Shrivastava, H., He, Y.: AntMan: Sparse low-rank compression to accelerate RNN inference. arXiv preprint arXiv:1910.01740 (2019)
14. Scutari, M.: Learning Bayesian networks with the bnlearn R package. Journal of Statistical Software **35**(3), 1–22 (2010). https://doi.org/10.18637/jss.v035.i03
15. Shrivastava, H.: On Using Inductive Biases for Designing Deep Learning Architectures. Ph.D. thesis, Georgia Institute of Technology (2020)
16. Shrivastava, H., Bart, E., Price, B., Dai, H., Dai, B., Aluru, S.: Cooperative neural networks (CoNN): Exploiting prior independence structure for improved classification. arXiv preprint arXiv:1906.00291 (2019)
17. Shrivastava, H., Chajewska, U.: Methods for recovering Conditional Independence graphs: A survey. arXiv preprint arXiv:2211.06829 (2022)
18. Shrivastava, H., Chajewska, U.: Neural Graphical Models. In: Proceedings of the 17th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), to appear (2023), `https://doi.org/10.48550/arXiv.2210.00453`
19. Shrivastava, H., Chajewska, U., Abraham, R., Chen, X.: A deep learning approach to recover conditional independence graphs. In: NeurIPS 2022 Workshop: New Frontiers in Graph Learning (2022), `https://openreview.net/forum?id=kEwzoI3Am4c`
20. Shrivastava, H., Chen, X., Chen, B., Lan, G., Aluru, S., Liu, H., Song, L.: `GLAD`: Learning sparse graph recovery. In: International Conference on Learning Representations (2020), `https://openreview.net/forum?id=BkxpMTEtPB`
21. Shrivastava, H., Huddar, V., Bhattacharya, S., Rajan, V.: Classification with imbalance: A similarity-based method for predicting respiratory failure. In: 2015 IEEE international conference on bioinformatics and biomedicine (BIBM). pp. 707–714. IEEE (2015)
22. Shrivastava, H., Zhang, X., Song, L., Aluru, S.: GRNUlar: A deep learning framework for recovering single-cell gene regulatory networks. Journal of Computational Biology **29**(1), 27–44 (2022)
23. United States Department of Health and Human Services (US DHHS), Centers of Disease Control and Prevention (CDC), National Center for Health Statistics (NCHS), Division of Vital Statistics (DVS): Birth Cohort Linked Birth – Infant Death Data Files, 2004-2015, compiled from data provided by the 57 vital statistics jurisdictions through the Vital Statistics Cooperative Program, on CDC WONDER On-line Database. Accessed at `https://www.cdc.gov/nchs/data_access/vitalstatsonline.htm`
24. Vân Anh Huynh-Thu, A.I., Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. PloS one **5**(9) (2010)

A Ethical Concerns

Our method does not introduce any new ethical issues, however, we should take care when applying to sensitive data. Anonymized infant mortality data is meant to generate insights into risk factors, primarily for use by doctors. It is not intended to offer medical advice to expecting parents.