Learning to Discover Probabilistic Graphical Model Structures

Eugene Belilovsky
INRIA Galen
University of Paris-Saclay
Chatenay-Malabry, France
eugene.belilovsky@inria.fr

Kyle Kastner
MILA
University of Montreal
Montreal, Canada
kyle.kastner@umontreal.ca

Gael Varoquaux
INRIA Parietal
Saclay,France
gael.varoquaux@normalesup.org

Matthew Blaschko
KU Leuven
Leuven, Belgium
matthew.blaschko@esat.kuleuven.be

Abstract

In this work we consider structure discovery of undirected graphical models from observational data. Inferring likely structures from few examples is a complex task often requiring formulating priors and sophisticated inference procedures. In the setting of Gaussian Graphical Models (GGMs) a popular approach to formulating an estimator is with a penalized maximum likelihood objective on the precision matrix. This objective is often difficult to design to specifically fit ones priors and the graph structure recovery is often not explicitly possible to embed in the objective, moreover incorporating any additional assumptions often requires a great deal of research effort. By contrast, it may be easier to generate samples of data that are arise from graphs with the desired properties. We propose here to leverage this latter source of information in order to learn a function that maps from empirical covariance matrices to estimated graph structures. This learned function brings two benefits: it implicitly models the desired structure or sparsity properties to form suitable priors, and it can more directly be tailored to the specific problem of edge structure discovery. We apply this framework to several critical real world problems in structure discovery and show that it can be competitive to standard approaches such as graphical lasso, at a fraction of the execution speed. We use deep neural networks to parametrize our estimators. Experimentally, our learn able graph discovery method trained on synthetic data generalizes well to different data: identifying relevant edges in real data, completely unknown at training time. We find that on genetics, brain imaging, and simulation data we obtain competitive (and often superior) performance, compared with analytical methods.

1 Introduction

Probabilistic graphical models provide a powerful framework for describing the dependencies between a set of variables. Many applications require inferring the structure of a probabilistic graphical model from data to elucidate the relationships between variables. These relationships are often represented by an undirected graphical model also known as a Markov Random Field (MRF). When there are few samples and the data is high dimensional, characterizing the distribution of possible structures is very challenging, even when assuming a straightforward parametric distribution on the underlying data.

We focus on a common MRF model arising from the multivariate gaussian, the Gaussian graphical models (GGMs). GGMs are used in structure-discovery settings for rich data such as neuroimaging, genetics, or finance [36, 32]. Despite their Gaussian assumption, determining likely structures from few examples is a complex task requiring strong priors, typically a sparsity assumption, or other restrictions on the structure of the graph.

A standard approach to estimating sparse Gaussian graphical models in high dimensions is based on the classic result that the zeros of a precision matrix correspond to zero partial correlation, a necessary and sufficient condition for conditional independence [27, 21]. Assuming only a few conditional dependencies corresponds to a sparsity constraint on the entries of the partial correlation or precision matrix. Since this leads to a combinatorial problem the ℓ_0-norm is often relaxed to a convex ℓ_1-norm regularizer.

Many popular approaches to learning GGMs can be seen as leveraging the ℓ_1-norm to create convex surrogates to this problem. [28] use nodewise ℓ_1 penalized regressions. Other estimators penalize the precision matrix directly. These methods include the CLIME estimator [5] and the very popular graphical lasso [12, 3, 35]

$$f_{glasso}(\hat{\mathbf{\Sigma}}) = \arg\min_{\mathbf{\Theta}\succ 0} -\log|\mathbf{\Theta}| + \mathrm{Tr}\,(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) + \lambda\|\mathbf{\Theta}\|_1 \tag{1}$$

which can be seen as a penalized maximum likelihood estimator. Here $\mathbf{\Theta}$ and $\hat{\mathbf{\Sigma}}$ are the precision and sample covariance matrices, respectively. A large variety of alternative regularization penalties, extend the priors of the graphical lasso [8, 32, 19, 36, 40].

Several problems exists in this approach. Constructing novel surrogates for structured sparsity assumptions on MRF structures is rather complex. A prior is formulated and incorporated into a penalized maximum likelihood objective. However, to efficiently optimize such an objective sophisticated algorithms must be developed, often requiring a separate research effort. Furthermore, model selection in a penalized maximum likelihood setting is rather challenging since the regularization parameters are often unintuitive.

We propose instead of designing an estimator, to frame the problem of finding a graph estimation procedure as selecting a function from a large flexible function class by way of risk minimization. This allows us to construct a loss function that explicitly attempts to recover the edge structure. We can often sample from a distribution of graphs and empirical covariances with our desired properties, even though this distribution may not be analytically tractable. This allows us to perform empirical risk minimization to select an appropriate function to use for edge estimation. This framework allows us to more easily control the assumed level of sparsity (as opposed to graph lasso) or modify various other aspects of the sampling to shape the expected distribution, while optimizing a desired performance metric.

Motivated by theoretical results showing effectiveness in learning smooth functions [7] and impressive empirical results on approximating solutions to combinatorial problems [42], we propose to use a deep neural network as the function class. We train it by sampling from small sample data, generated from graphs with the prescribed properties, with a primary focus on sparse (Gaussian) graphical models. Small sample ($n < p$) covariance matrices are estimated and given to the neural network architecture (Figure 1) as input training data, which are to be mapped to indicators of present and absent edges in the underlying GGM. The learned network can then be employed in various real-world structure discovery problems.

In Section 1.1 we review the the related work. In Section 2 we formulate the risk minimization view of graph structure inference and describe how it can apply to sparse GGMs. Section 2.3 we describe and motivate the deep learning architecture we chose to use for the sparse GGMs problem in this work. In Section 3 we describe the details of how we train the deep network to obtain an edge estimator for sparse GGMs. We then evaluate its properties extensively on simulation data. Finally, we show that this edge estimator trained only on generated data can obtain state of the art performance at inference time on real neuroimaging and genetics problems, while being much faster to execute than other methods.

1.1 Related Work

Related to our work, [26] propose and analyze learning based methods to discover structure of directed graphical models in causal inference. They frame the problem as learning on probability measures and make use of estimates of kernel-mean embeddings as a representation of the distribution.

As in our work they demonstrate the use of simulations for training while testing on real data. This work primarily focuses on finding the causal direction in two node graphs with lots of observations. The other major difference in our work is that we focus on undirected graphs and learning functions which specifically take as input few observations. Moreover, we use GGMs for which allows us to use the covariance as a representation of the distribution.

Our choice of learning architecture is motivated by the recent literature on deep networks. In [42, 41] it was shown that neural networks can learn to provide approximate solutions to NP-hard combinatorial problems, and the problem of optimal edge recovery in a probabilistic graphical model can be seen as a combinatorial optimization problem. Several works have been proposed which deal with neural architectures for input data arising from a graph. [17, 11, 24, 37]. These are often based on multi layer convolutional networks as in this work or multi-step recurrent neural network models. The input in our approach can be viewed as a complete graph, while the ouput a sparse graph, thus none of these models directly apply, but we use them as inspiration. Another related use of deep networks to approximate a posterior distribution can be found in [1].

Bayesian approaches to structure learning often rely on priors on the graph combined with sampling techniques to estimate the posterior of the graph structure. Some approaches make assumptions on the decomposability of the graph [29]. The G-Wishart distribution is a popular distribution which forms part of a framework for structure inference, and advances have been recently made in efficient sampling [30]. However, as seen in some timing and performance evaluations done as part of this work these methods can still be rather slow compared to competing methods, and in the setting of $p > n$ we find they are less powerful.

2 Methods

In this Section we describe our formulation of edge estimation in MRF as a learnable function, we specialize this to the case of sparse GGMs, and then describe a deep learning architecture to represent to represent a function class we can use find efficient edge estimators.

2.1 Learning an Approximate Edge Estimation Procedure

Let $\boldsymbol{X} \in R^{n \times p}$ be a matrix whose n columns are i.i.d. samples $x \sim P(x)$ of dimension p. Let $G = (V, E)$ be an undirected and unweighted graph associated with the set of variables in x. Let $\mathcal{L} = \{0, 1\}$ and $N_e = \frac{p(p-1)}{2}$ the maximum possible edges in E. Let $Y \in \mathcal{L}^{N_e}$ indicate the presence or absence of edges in the edge set E of G, namely

$$Y^{ij} = \begin{cases} 0 & x_i \perp x_j | x_{V \setminus i,j} \\ 1 & x_i \not\perp x_j | x_{V \setminus i,j} \end{cases} \tag{2}$$

We define an approximate structure discovery method $g_w(\boldsymbol{X})$, which produces a prediction of the edge structure, $\hat{Y} = g_w(\boldsymbol{X})$, given a set of data $\boldsymbol{X}$. In the rest of this paper we focus on $\boldsymbol{X}$ arising from Gaussian variables for which it can be more straightforward to consider $\hat{\boldsymbol{\Sigma}}$, the empirical covariance matrix corresponding to $\boldsymbol{X}$. In this case we will denote $g_w(\boldsymbol{X}) := f_w(\hat{\boldsymbol{\Sigma}})$. f_w is parametrized by w and belongs to the function class $\mathcal{F}$. We note that the graphical lasso in Equation (1) is an f_w for an appropriate choice of $\mathcal{F}$.

This view on the edge estimator now allows us to bring the selection of f_w from the domain of human design to the domain of empirical risk minimization over $\mathcal{F}$. Defining a distribution $\mathbb{P}$ on $R^{n \times p} \times \mathcal{L}^{N_e}$ such that $(\hat{\boldsymbol{\Sigma}}, Y) \sim \mathbb{P}$, we would like our estimator, f_w, to minimize the expected risk

$$R(f) = \mathbb{E}_{(\hat{\boldsymbol{\Sigma}}, Y) \sim \mathbb{P}}[l(f(\hat{\boldsymbol{\Sigma}}), Y)] \tag{3}$$

Here $l : R^{n \times p} \times \mathcal{L}^{N_e} \rightarrow \mathbb{R}^+$ is the loss function. For graphical model selection the $0/1$ loss function, $l = \mathbb{I}(Y \neq \hat{Y})$ (where $\mathbb{I}$ is the indicator function), the natural error metric to consider [43]. In this setting it has been shown that (1) achieves the information theoretic optimal recovery rate up to a constant for certain $\mathbb{P}$ corresponding to classes of uniformly sparse graphs with a maximum degree, when the optimal λ is used [43, 35].

The design of the estimator in Equation (1) is not explicitly minimizing this risk functional. Thus modifying the estimator to fit a different class of graphs (e.g small-world networks) while minimizing

$R(f)$ is not obvious. Furthermore in practical settings the optimal λ is unknown. We would prefer to directly minimize the risk functional. The desired structural assumptions on samples from $\mathbb{P}$, such as sparsity, on the underlying graph mean the distribution is not tractable for analytic solutions. Meanwhile, we can often sample from such a distribution allowing us to select an appropriate function via empirical risk minimization.

In practice we need to approximate the $0/1$ loss function with a convex surrogate. We use a typical convex surrogate, the cross-entropy loss:

$$l(f_w(\hat{\mathbf{\Sigma}}), Y) = \sum_{i \neq j} \left(Y^{ij} \log(f_w^{ij}(\hat{\mathbf{\Sigma}})) + (1 - Y^{ij}) \log(1 - f_w^{ij}(\hat{\mathbf{\Sigma}})) \right) \tag{4}$$

Unlike expressions such as the graphical lasso, the estimator with minimum risk is generally not possible to compute a closed form expression for most interesting choices of $\mathbb{P}$, such as those arising from sparse graphs. However, we can often devise a sampling procedure for $\mathbb{P}$ as will be discussed in subsequent sections. Thus it is sufficient to define a rich enough $\mathcal{F}$ over which we can minimize the empirical risk over the samples generated. Giving us a learning objective over N samples $\{Y_k, \mathbf{\Sigma}_k\}_{k=1}^N$ drawn from $\mathbb{P}$:

$$\min_w \frac{1}{n} \sum_{k=1,..,N} l(f_w(\hat{\mathbf{\Sigma}}_k), Y_k) \tag{5}$$

We now need to select a sufficiently rich function class for f_w and a method to produce appropriate $(Y, \hat{\mathbf{\Sigma}})$ which model our desired data priors. This will allow us to learn a f_w that explicitly attempts to minimize errors in edge discovery.

2.2 Discovering Sparse Gaussian Graphical Model and Beyond

We specialize this approach to Gaussian graphical models (GGMs) which are used in many modalities [19, 40, 4]. A typical assumption when applying these models in genetics, neuroimaging, social networking, and other data is that the number of edges is sparse. A convenient property of these GGMs is that the precision matrix has a zero value in the (i, j)th entry precisely when variables i and j are independent conditioned on all other variables. Additionally, the precision matrix and partial correlation matrix have the same sparsity pattern, while the partial correlation matrix has normalized entries.

We propose to simulate our *a priori* assumptions of sparsity and Gaussianity and attempt to learn $f_w(\hat{\mathbf{\Sigma}})$, which can then produce predictions of edges from the input data. The empirical covariance $\hat{\mathbf{\Sigma}}$ is a natural input for this task as, for Gaussian distributions, the empirical covariance is a sufficient statistic for the population covariance and hence the conditional independence structure. We model $P(x|G)$ as arising from a sparse prior on the graph G and correspondingly the entries of the precision matrix $\mathbf{\Theta}$. To obtain a single sample of $\boldsymbol{X}$ corresponds to n i.i.d. samples from $\mathcal{N}(0, \mathbf{\Theta}^{-1})$. We can now train $f_w(\hat{\mathbf{\Sigma}})$ by generating sample pairs $(\hat{\mathbf{\Sigma}}, Y)$. At execution time we standardize the input data and compute the covariance matrix before evaluating $f_w(\hat{\mathbf{\Sigma}})$. The process of learning f_w for the sparse GGM is given in Algorithm 1. A rather generic sparsity prior is one where each edge is equally likely with small probability, verusus structured sparsity where edges have specific configurations. For obtaining the training samples $(\hat{\mathbf{\Sigma}}, Y)$ in this case we would like to create a sparse precision matrix, $\mathbf{\Theta}$, with the desired number of zero entries distributed uniformly. One strategy to do this and assure the precision matrices lie in the positive definite cone is to first construct an upper triangular sparse matrix and then multiply it by its transpose. This process is described in detail in the experimental section. Alternatively an MCMC based G-Wishart distribution sampler can be potentially employed if specific strucutres of the graph are desired [23]. This sampling process however is more time consuming.

The sparsity patterns in probabilistic graphical models are often not uniformly distributed. Many real world networks are known to form small-world networks: graphs that are sparse and yet have a comparatively short average distance between nodes. These transport properties often hinge on a small number of hubs: high-degree nodes. Such structural patterns often require sophisticated adaptation of the learning objectives and reformulations of convex optimization procedures. Indeed, high-degree nodes break the small-sample, sparse-recovery properties of ℓ_1-penalized estimators [35]. In our framework such structure assumption appears as a prior that can be learned offline during training of the neural network. Similarly priors on other distributions such as general exponential families can be more easily integrated if a sampling procedure is readily available. As the structure discovery model can be trained offline even a slow sampling procedure can suffice.

2.3 Deep Graph Network

For our $f_w(\hat{\mathbf{\Sigma}})$ we have chosen to use a neural network due to their ability to efficiently approximate arbitrary functions. In this section we discuss the network design for $f_w(\hat{\mathbf{\Sigma}})$. There are many architectural and design decisions one could make when applying a neural network, especially on a positive semidefinite matrix as done here. We use the empirical covariance as the input to the network (as opposed e.g. to a flattened upper triangular matrix). This input representation, combined with convolutional layers, allows the network to better leverage the 2D structure of the input.

Algorithm 1 Process of training a GGM edge estimator

```
Select Function Class $\mathcal{F}$ (e.g. CNN)
$N$ samples, $n$ iid samples
for i $\in \{1, .., N\}$ do
    Sample $G_i \sim P(G)$
    Sample $\mathbf{\Sigma}_i \sim P(\mathbf{\Sigma}|G = G_i)$
    $\boldsymbol{X}_i \leftarrow \{x_j \sim N(0, \mathbf{\Sigma})\}_{j=1}^{n}$
    Construct $(Y_i, \hat{\mathbf{\Sigma}}_i)$ pair from $(G_i, \boldsymbol{X_i})$
end for
Optimization: $\min_{f \in \mathcal{F}} \sum L_i(Y_i, f(\hat{\mathbf{\Sigma}}_i))$
```

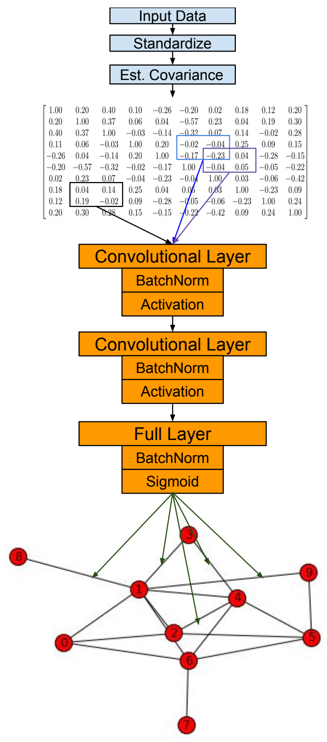

Figure 1: Diagram of the DeepGraph structure discovery architecture used in this work. The input is first standardized and then the sample covariance matrix is estimated. A neural network consisting of two convolutional layers and one fully connected layer is used to predict edges corresponding to non-zero entries in the precision matrix.

Compositionality in neural networks, which exploits structural patterns in the data, can often improve performance. For example, one of several driving points in the improvement of deep learning methods has been the use of compositional structures such as convolutional and recurrent layers. These structures allow the learning of parameters that exploit locality (convolution) or repetition (recurrence). Unlike image, speech, or text data, the existence of a compositional structure in the posed problem is not obvious. However, it is known that in the case of Gaussians as well as other families of distributions [25] the zeros of the inverse covariance matrix correspond to the conditional independence structure of the data. A well known formula from linear algebra for the (i, j)th entry of the inverse of an invertible matrix, A, is

$$A^{-1}_{i,j} = \frac{(-1)^{i+j}\det(\tilde{A}_{i,j})}{\det(A)},$$

where $\det(\tilde{A}_{i,j})$ refers to the (i,j)th minor of A. The determinant has a compositional structure, thus for the limiting case of a very large sample size in Gaussian data we may want our network to (approximately) compute determinants. This motivates the application of a convolutional layer directly to the covariance matrix. Convolutional layers allow the network to easily learn computations similar to determinants and share them amongst output variables. We found that this substantially improved generalization performance compared to using deep and shallow fully connected layers. We highlight also that the linear algebra formula shows the inverse entries are rational functions of the original matrix and thus the inverse operation is a smooth function. Theoretical analysis of deep learning algorithms, recently for the case of convolutional networks in particular[7], has shown they are very effective at learning smooth functions.

Thus the form of $f_w(\hat{\mathbf{\Sigma}})$ used in the experiments is as follows

$$h_1 = g(\mathrm{conv}(\hat{\mathbf{\Sigma}}, W_1)) \tag{6}$$

$$h_2 = g(\mathrm{conv}(h_1, W_2)) \tag{7}$$

$$h_3 = g(W_3 h_2) \tag{8}$$

$$f_w(\hat{\mathbf{\Sigma}}) = \sigma(W_4 h_3) \tag{9}$$

Here $g(\cdot)$ is a generic activation function, $\mathrm{conv}(\cdot,\cdot)$ indicates a convolutional layer, and $\sigma(\cdot)$ is the sigmoid function. We use 2×2 convolutional kernels at the first and second layers. Figure 1 gives a diagram of the network.

The optimal graph structure should be independent of the order of the input dimensions. This property is inherently present in existing structure recovery methods, however the network must learn this invariance property from the data. This means if we permute the data we can obtain another estimate of the output. In our experiments we find that averaging results over several permutations of the input data can significantly improve performance.

3 Experiments

In our experiments we explore how well networks trained purely on parametric samples generalize, both to unseen synthetic data and to several real world problems. In order to highlight the generality of the learned networks, we apply the same network to multiple domains. We train two networks, DeepGraph-39 which takes in 39×39 covariance matrices, and DeepGraph-50 which is trained to output structures for 50 node networks. The sizes are chosen based on the real data we consider in subsequent sections. In both networks, we have 100 feature maps of 2×2 kernels in the first convolutional layer, and 50 feature maps of 2×2 kernels in the 2nd layer. The fully connected layer has 3000 hidden nodes. For DeepGraph-39 we use PReLU activation and ReLU in DeepGraph-50. The last layer is a sigmoid outputing a value between 0 and 1 for each edge.

Each network was trained continously with new samples generated until the validation error saturated. For a given precision matrix we generated 5 possible $\boldsymbol{X}$ samples to be used as training data. A total of approximately 1M training samples were used for each network. The networks were optimized using the ADAM optimizer [20] coupled with cross-entropy loss as the objective function (cf. Sec. 2.1). We additionally use batch normalization at each layer. In order to encourage the networks to converge to sparser solutions we add a small regularization term $\lambda \| f_w(\hat{\mathbf{\Sigma}}) \|_2$, with $\lambda = 0.01$. We found this improved convergence for our expectedly low outputs. Additionally, we found that using the absolute value of the true partial correlations as labels, instead of hard binary labels, improved our logloss and AUC on the validation set.

We use the following procedure for sampling $P(X|G)$ with a sparse prior on $P(G)$.

- Construct lower diagonal matrix L where each entry has α probability of being zero. Non-zero entries are set uniformly betwen -0.9 and 0.9
- Multiply L by its transpose to obtain a sparse positive definite precision matrix, Ω. This gives us our $P(\Omega|G)$ with a sparse prior on $P(G)$
- Sample from the Gaussian $\mathcal{N}(0, \Omega^{-1})$ to obtain samples of $\boldsymbol{X}$

Here α corresponds approximately to a specific sparsity level in the final precision matrix. In our experiments we set α to produce matrices sparse in the range of $92 - 96\%$.

Our experiments focus on high dimensional settings where the number of samples are relatively small compared to the feature dimensionality. A typical caveat [6] of recovery methods for GGMs is that when $n \gg p$ estimation methods which are consistent will converge to the true precision matrix and corresponding edge structure, as such we focus on the case of $p < n$ where the covariance is singular and estimation of the underlying structure is challenging. We begin by considering synthetic data generated from a different sampling process and then evaluate real data from the genetics and neuroimaging domains.

3.1 Synthetic Data Evaluation

To understand the properties of our learned networks, we evaluated them on different synthetic data than the ones they were trained on. More specifically, we used a completely different third party sampler so as to avoid any contamination. For all experiments DeepGraph-39 was used. The same trained network is utilized in the subsequent neuroimaging evaluations as well.

We used the `BDGraph` R-package to produce sparse precision matrices based on the G-Wishart distribution [31]. Additionally we utilized the R-package `rags2ridges` [34] to generate data from small-world networks corresponding to the Watts–Strogatz model. We compared our network against the `scikit-learn` [33] implementation of Graphical Lasso with regularization chosen by cross-validation as well as the Birth-Death Rate MCMC method from [30] implemented in `BDGraph`. As discussed in Section 2.3 we can permute the input to obtain another estimate from our network for the same data. We average the result of 20 permutations of DeepGraph-39 and compare it to the other methods as well as a single application of DeepGraph-39. In Table 1 we show various evaluation metrics for edge recovery in different scenarios. For each scenario we repeat the experiment for 20 different graphs and small sample observations showing the final average result. We evaluate logloss, area under the ROC curve (AUC), precision@k, average precision (AP), and calibration error (CE) [30]. For graphical lasso we use the partial correlations to indicate confidence in edges; `BDGraph` automatically returns posterior probabilities as does our method. For all our experiments, data with graph sparsity of approximately 5% is used. Experiments with variable sparsity are considered in the supplementary material. We find that for very sparse graphs, the networks remain robust in performance, while for increased density performance degrades but is still competitive in several categories.

Experimental Setup	Method	logloss	AUC	Prec@2.5%	Prec@5%	AP	CE
Gaussian Random Graphs (n=35)	GLasso	1.306	0.709	**0.73**	**0.53**	**0.451**	0.07
	Bdgraph	1.020	0.716	0.54	0.48	0.354	0.19
	DeepGraph-39	0.266	0.752	0.63	0.5	0.374	0.07
	DeepGraph-39+Perm	***0.260***	***0.772***	*0.67*	*0.52*	*0.403*	*0.07*
Laplace Random Graphs (n=35)	GLasso	1.254	0.698	**0.65**	**0.46**	**0.386**	0.07
	Bdgraph	2.897	0.666	0.29	0.27	0.287	0.28
	DeepGraph-39	0.263	0.708	0.56	0.42	0.325	0.08
	DeepGraph-39+Perm	***0.255***	***0.729***	*0.62*	*0.44*	*0.347*	*0.08*
Gaussian Random Graphs (n=15)	GLasso	1.625	0.642	0.46	0.34	**0.281**	**0.07**
	Bdgraph	3.397	0.608	0.17	0.16	0.212	0.38
	DeepGraph-39	0.280	0.688	0.44	0.32	0.246	0.09
	DeepGraph-39+Perm	***0.274***	***0.695***	***0.49***	***0.37***	*0.270*	*0.09*
Gaussian Random Graphs (n=100)	GLasso	0.995	0.757	**0.85**	**0.6**	**0.557**	0.06
	Bdgraph	0.450	0.777	0.82	0.62	0.538	0.09
	DeepGraph-39	0.251	0.787	*0.7*	0.53	0.421	0.06
	DeepGraph-39+Perm	***0.249***	***0.81***	0.69	*0.55*	*0.436*	*0.06*
Gaussian Hub Graphs (n=35)	GLasso	0.81	0.733	0.4	0.31	0.276	0.06
	Bdgraph	1.226	0.716	0.31	0.28	0.262	0.2
	DeepGraph-39	0.192	***0.801***	0.43	0.33	0.256	0.06
	DeepGraph-39+Perm	***0.19***	0.781	***0.48***	***0.34***	***0.309***	*0.06*
Gaussian Small-World Graphs (n=35)	Glasso	2.971	0.582	0.45	0.34	**0.309**	0.11
	Bdgraph	1.041	0.695	0.51	0.45	0.311	0.17
	DeepGraph-39	0.483	0.687	0.52	0.43	0.294	0.11
	DeepGraph-39+Perm	***0.473***	***0.709***	***0.50***	*0.45*	*0.305*	*0.11*
	DeepGraph-39+Update	0.303	0.761	0.62	0.54	0.367	0.13
	DeepGraph-39+Update+Perm	***0.285***	***0.779***	***0.68***	***0.59***	***0.419***	0.14

Table 1: For each scenario we generate 20 graphs with 39 nodes, and the corresponding data matrix is sampled from distributions with those underlying graphs. We see that DeepGraph outperforms other methods in terms of logloss and AUC. In terms of precision at expected sparsity levels DeepGraph has better performance for graphs with high-degree nodes such as small world and hub graphs, demonstrated in Figure 2. The number of samples is indicated by n. Further experiments in other scenarios are considered in the appendix.

For the case of random Gaussian graphs with number of samples at 35 (our training distribution) we have superior performance in terms of AUC, logloss, and calibration error. Graphical Lasso performs better in terms of precision at the relevant levels. Our method is superior to the Birth-Death Rate MCMC based `bdgraph` in all categories.

Next we apply the method to a less straightforward synthetic data, with distributions typical of many applications. We found that, compared to baseline methods, our network performs particularly well in with high-degree nodes. In particular our method performs well on the relevant metrics with small-world networks, a very common family of graphs in real-world data, obtaining superior precision at the primary levels of interest. Figure 2 shows examples of random, Watts and Strogatz small-world graphs, and concentrated hub graphs used in these experiments.

Training a new network for each number of samples can pose difficulties with our proposed method. Thus we evaluted how robust the network DeepGraph-39 is to input covariances obtained from fewer or more samples. We find that overall the performance is quite good even when lowering the number of samples to $n = 15$, we obtain superior performance to the other approaches in a number of sample sizes and metrics (Table 1).

We generated data from a multivariate generalization of the Laplace distribution as described in [14]. As in other experiments precision matrices were sampled from the G-Wishart at a sparsity of 5%. [14, Proposition 3.1] was applied to produce samples using these precision matrices. We find that our method still performs competitively, despite the discrepancy between train and test distributions. Finally we note that our optimization criteria in training is based on log loss. However, this can be adjusted to fit a desired metric by modifying the objective, a rather challenging task in standard approaches.

Using the small-world network data generator [34], we demonstrate that we can update the generic sparse prior to a structured one. We re-train DeepGraph-39 using only 1000 examples from mixed with 1000 examples from the original uniform sparsity model. We perform just one epoch of training and observe markedly improved performance on this test case as seen in the last row of Table 1.

We compute the average execution time of our method compared to Graph Lasso and BDGraph on a CPU in Table 2. We note that we use a production quality version of graph lasso [33], whereas we have not optimized the network execution, for which known strategies may be applied [9].

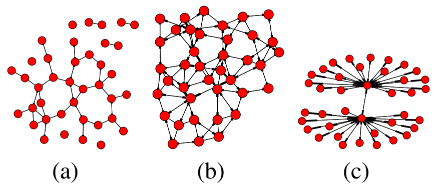

Figure 2: Example of (a) random (b) small world (c) hub networks used in synthetic experiments

	average execution time (s)
`sklearn` GraphLassoCV	4.81
`BDgraph`	42.13
DeepGraph-50	***0.21***

Table 2: Average execution time over 10 trials for 50 node problem on a CPU for Graph Lasso, Birth Death rate MCMC, and DeepGraph-50

3.2 Evaluation using Cancer Genome Data

We perform experiments on a gene expression dataset described in [18]. The data come from a cancer genome atlas from 2360 subjects for various types of cancer. We used the first 50 genes from Appendix C.2 of [18] of commonly regulated genes in cancer. We evaluated on two groups of subjects, one with breast invasive carcinoma (BRCA) consisting of 590 subject samples and the other colon adenocarcinoma (CODA) consisting of 174 samples.

As there is no ground truth for this dataset, evaluating edge selection is challenging. We use the following methodology: for each method we select the top-k ranked edges, and then fix all other edges, recomputing the maximum likelihood precision matrix with support given by the corresponding edge selection method. We then evaluate the likelihood on a held-out set of data. We repeat this procedure for a range of k. We rely on Algorithm 0 in [16] to compute the maximum likelihood precision given a support. The experiment is repeated for each of CODA and BRCA subject groups 50 times, results are shown in Figure 3. In all cases we use 40 samples for edge selection and precision estimation. We compare with graphical lasso as well as the Ledoit-Wolfe shrinkage based estimator [22]. We additionally consider the Bayesian birth-death rate MCMC approach described in previous sections. For graphical lasso and Ledoit-Wolfe, edge selection is based on thresholding partial correlation [2].

Additionally we evaluate the stability of the solutions provided by the various methods. This is important in several applications where one wants a low variance estimate of the edge set. We chose to use Spearman correlations between pairs of solutions, as it is a measure of a monotone link between two variables. The results are shown in Table 3. We see that DeepGraph has better stability by a wide margin in all cases in this dataset.

	Gene BRCA	Gene COAD	ABIDE Control	ABIDE Autistic
Graph Lasso	$0.24 \pm .003$	0.32 ± 0.004	$0.22 \pm .003$	$0.25 \pm .003$
Ledoit-Wolfe	0.12 ± 0.002	0.15 ± 0.003	$0.13 \pm .003$	$0.14 \pm .003$
Bdgraph	0.06 ± 0.002	0.08 ± 0.003	N/A	N/A
DeepGraph	$\mathbf{0.74 \pm 0.004}$	$\mathbf{0.71 \pm 0.005}$	$\mathbf{0.29 \pm .004}$	$\mathbf{0.31 \pm .004}$
DeepGraph +Permute	0.60 ± 0.003	0.58 ± 0.0043	$0.22 \pm .004$	$0.22 \pm .004$

Table 3: Average Spearman correlation results. Stability of solution amongst 50 trials for each dataset

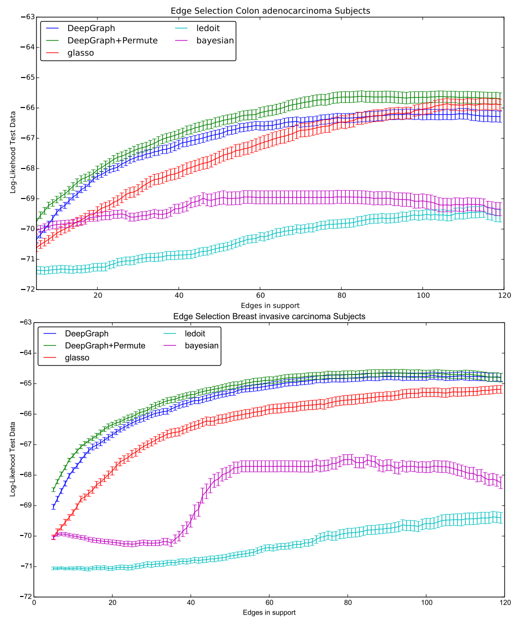

Figure 3: Average test log likelihood for COAD and BRCA subject groups using different settings of the number of selected edges. Each experiment is repeated 50 times. DeepGraph with averaged permutation dominates in all cases.

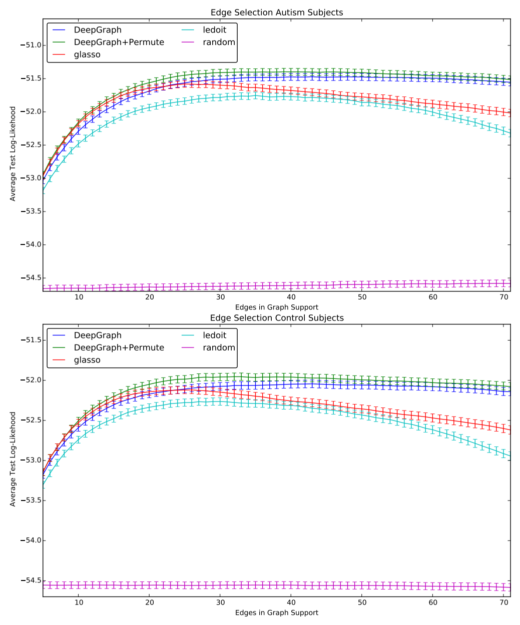

Figure 4: Average test log likelihood for Autism and Control groups in ABIDE data. DeepGraph+Permuation is superior or equal to competing methods in each case. The test is repeated 1000 times to obtain significant results due high variance in the data.

3.3 Resting State Functional Connectivity

We evaluate our graph discovery method for the purpose of identifying brain functional connectivity in resting state fMRI data. Correlations in brain activity measured via fMRI reveal functional interactions between remote brain regions. These are an important measure to study psychiatric diseases that have no known anatomical support. A typical approach in connectome analysis is to describe each subject or group by a Gaussian graphical model measuring functional connectivity between a set of regions [38, 4].

We chose to use the ABIDE (Autism Brain Imaging Data Exchange) dataset [10], a large scale resting state fMRI dataset. It gathers brain scans from 1,112 subjects across 16 sites, with 539 individuals suffering from autism spectrum disorder and 573 typical controls. Preprocessed data is available online.[1] For our experiments we use a multi-subject atlas of functional regions derived from resting-state fMRI using multi-subject dictionary-learning (MSDL) [39] which produces 39 regions of interest.

[1] `http://preprocessed-connectomes-project.github.io/abide/`

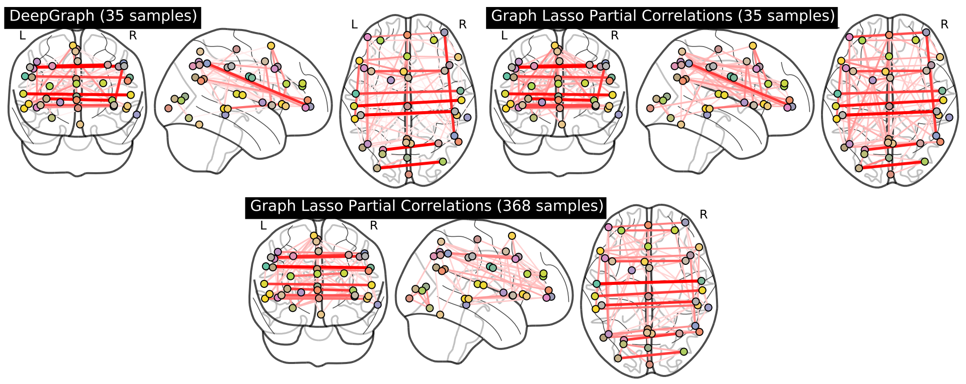
DeepGraph (35 samples)
Graph Lasso Partial Correlations (35 samples)
Graph Lasso Partial Correlations (368 samples)
L
R

Figure 5: Example solution from DeepGraph and Graph Lasso in the small sample regime on the same 35 samples, along with a larger sample solution of Graph Lasso for reference

We use the network DeepGraph-39, the same network from synthetic experiments, using the same evaluation protocol as described in Section 3.2. For both control and autism patients we use time series from 35 random subjects to estimate edges and corresponding precision matrices. We find that for both the Autism and Control group we can obtain edge selection comparable to graph lasso for very few selected edges. When the number of selected edges is in the range above 25 we begin to perform significantly better in edge selection as seen in Fig. 4. We evaluated stability of the results as shown in Tab. 3. DeepGraph outperformed the other methods, however the method using permutations produces a statistical tie for the Control group and underperforms graph lasso in the Autistic subject group.

ABIDE has high variability across sites and subjects. To obtain low variance evaluation metrics we needed to perform 1000 folds to obtain well separated error bars, many more trials than were needed to see distinctions between approaches in the genetics experiment. We found that the birth-death MCMC method took very long to converge on this data, moreover the need for many folds to obtain significant results amongst the methods made this approach prohibitively slow to evaluate.

We show the edges returned by Graph Lasso and DeepGraph for a sample from 35 subjects (Fig. 5) in the control group. We also show the result of a large sample result based on 368 subjects from graphical lasso. In visual evaluation of the edges returned by DeepGraph we find that they closely align with expected results from a large sample estimation procedure. Furthermore we can see several edges in the subsample which were particularly strongly activated in both methods.

4 Discussion and Future Work

Our method was competitive with very strong baselines in many of the synthetic experiments. Particularly in the case of high-degree nodes and higher order cliques we found that neural networks performed better.

We have seen that the networks can adapt to variations in the number of samples and the sparsity level even when not explicitly trained on these parameters. This suggests a potential binning strategy where functions could be trained on certain ranges of data matrix samples or sparsity levels. At execution time one can call the appropriate method based on prior assumptions or use a model selection approach to select amongst these methods. The fast execution time would make approaches such as cross-validation feasible.

A disadvantage of our method compared to approaches such as penalized maximum likelihood is that providing a theoretical guarantee is more difficult. However, guarantees in the latter method often make restrictive assumptions on the form of the distribution and the design matrix and often do not consider the regularization path. In our method, one could additionally obtain bounds under the prescribed data distribution. Furthermore at execution time the speed of the approach can allow for re-sampling based uncertainty estimates that are infeasible for slower methods [13].

A difficulty of the proposed architecture is that the dimension of the input data must be specified for each trained network. Our experiments weakly suggest the network may be learning generic computations for determining the graph structure. Transfer learning, where a network for one size of data is used as a starting point to learn a higher dimensional network, is therefore an interesting approach. For very high dimensional settings, with $p \gg n$, it may make sense to input the data matrix directly and utilize a variable length architecture such as the one described in [41].

5 Conclusions

We have introduced a method of learning an estimator for determining the structure of an undirected graphical model. We proposed a network architecture and sampling procedure for learning such an estimator for the case of sparse Gaussian graphical models. We obtained competitive results on synthetic data with various underlying distributions, as well as on challenging real-world data. We found that our method works particularly well compared to other approaches for small-world networks, an important class of graphs occurring frequently in real-world domains. We have shown that neural networks can obtain improved results over various statistical methods on real datasets, despite being trained with samples from parametric distributions. Our approach can allow for straightforward specifications of new priors and open new directions in efficient graphical structure discovery from few examples.

References

[1] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3420–3428, 2015.

[2] Samuel Balmand Samuel Balmand and Arnak S Dalalyan. On estimation of the diagonal elements of a sparse precision matrix. *arXiv preprint arXiv:1504.04696*, 2015.

[3] Onureena Banerjee, Laurent El Ghaoui, Alexandre d'Aspremont, and Georges Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd international conference on Machine learning*, pages 89–96. ACM, 2006.

[4] Eugene Belilovsky, Gaël Varoquaux, and Matthew B Blaschko. Hypothesis testing for differences in Gaussian graphical models: Applications to brain connectivity. *arXiv preprint arXiv:1512.08643*, 2015.

[5] Tony Cai, Weidong Liu, and Xi Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[6] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[7] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: a tensor analysis. *arXiv preprint arXiv:1509.05009*, 2015.

[8] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(2):373–397, 2014.

[9] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[10] Adriana Di Martino, Chao-Gan Yan, Qingyang Li, Erin Denio, Francisco X Castellanos, Kaat Alaerts, Jeffrey S Anderson, Michal Assaf, Susan Y Bookheimer, et al. The autism brain imaging data exchange: Towards a large-scale evaluation of the intrinsic brain architecture in autism. *Molecular psychiatry*, 19:659, 2014.

[11] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2215–2223, 2015.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

[14] E Gómez, MA Gomez-Viilegas, and JM Marin. A multivariate generalization of the power exponential family of distributions. *Communications in Statistics-Theory and Methods*, 27(3):589–600, 1998.

[15] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[16] Hisayuki Hara and Akimichi Takemura. A localization approach to improve iterative proportional scaling in Gaussian graphical models. *Communications in Statistics–Theory and Methods*, 39(8-9):1643–1654, 2010.

[17] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[18] Jean Honorio, Tommi Jaakkola, and Dimitris Samaras. On the statistical efficiency of $\ell_{1,p}$ multi-task learning of Gaussian graphical models. *arXiv preprint arXiv:1207.4255*, 2012.

[19] Jean Honorio and Dimitris Samaras. Multi-task learning of Gaussian graphical models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 447–454, 2010.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Steffen L Lauritzen. *Graphical models*. Oxford University Press, 1996.

[22] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[23] Alex Lenkoski. A direct sampler for g-wishart variates. *Stat*, 2(1):119–128, 2013.

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[25] Po-Ling Loh and Martin J. Wainwright. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *The Annals of Statistics*, 41(6):3022–3049, 2013.

[26] David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf, and Ilya Tolstikhin. Towards a learning theory of cause-effect inference. In *Proceedings of the 32nd International Conference on Machine Learning, JMLR: W&CP, Lille, France*, 2015.

[27] George Marsaglia. Conditional means and covariances of normal variables with singular covariance matrix. *Journal of the American Statistical Association*, 59(308):1203–1204, 1964.

[28] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, pages 1436–1462, 2006.

[29] Baback Moghaddam, Emtiyaz Khan, Kevin P Murphy, and Benjamin M Marlin. Accelerating Bayesian structural inference for non-decomposable Gaussian graphical models. In *Advances in Neural Information Processing Systems*, pages 1285–1293, 2009.

[30] Abdolreza Mohammadi and Ernst C. Wit. Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.

[31] Abdolreza Mohammadi and Ernst C Wit. BDgraph: An R package for Bayesian structure learning in graphical models. *arXiv preprint arXiv:1501.05108*, 2015.

[32] Karthik Mohan, Mike Chung, Seungyeop Han, Daniela Witten, Su-In Lee, and Maryam Fazel. Structured learning of Gaussian graphical models. In *Advances in Neural Information Processing Systems*, pages 620–628, 2012.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] C.F.W. Peeters, A.E. Bilgrau, and W.N. van Wieringen. *rags2ridges: Ridge Estimation of Precision Matrices from High-Dimensional Data*, 2015.

[35] Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

[36] Srikanth Ryali, Tianwen Chen, Kaustubh Supekar, and Vinod Menon. Estimation of functional connectivity in fMRI data using stability selection-based sparse partial correlation with elastic net penalty. *NeuroImage*, 59(4):3852–3861, 2012.

[37] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Neural Networks, IEEE Transactions on*, 20(1):61–80, 2009.

[38] Gaël Varoquaux and R Cameron Craddock. Learning and comparing functional connectomes across subjects. *NeuroImage*, 80:405–415, 2013.

[39] Gaël Varoquaux, Alexandre Gramfort, Fabian Pedregosa, Vincent Michel, and Bertrand Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *Information Processing in Medical Imaging*, page 562, 2011.

[40] Gaël Varoquaux, Alexandre Gramfort, Jean-Baptiste Poline, and Bertrand Thirion. Brain covariance selection: Better individual functional connectivity models using population prior. In *Advances in Neural Information Processing Systems*, pages 2334–2342, 2010.

[41] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[42] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682, 2015.

[43] Wei Wang, Martin J Wainwright, and Kannan Ramchandran. Information-theoretic bounds on model selection for gaussian markov random fields. In *ISIT*, pages 1373–1377. Citeseer, 2010.

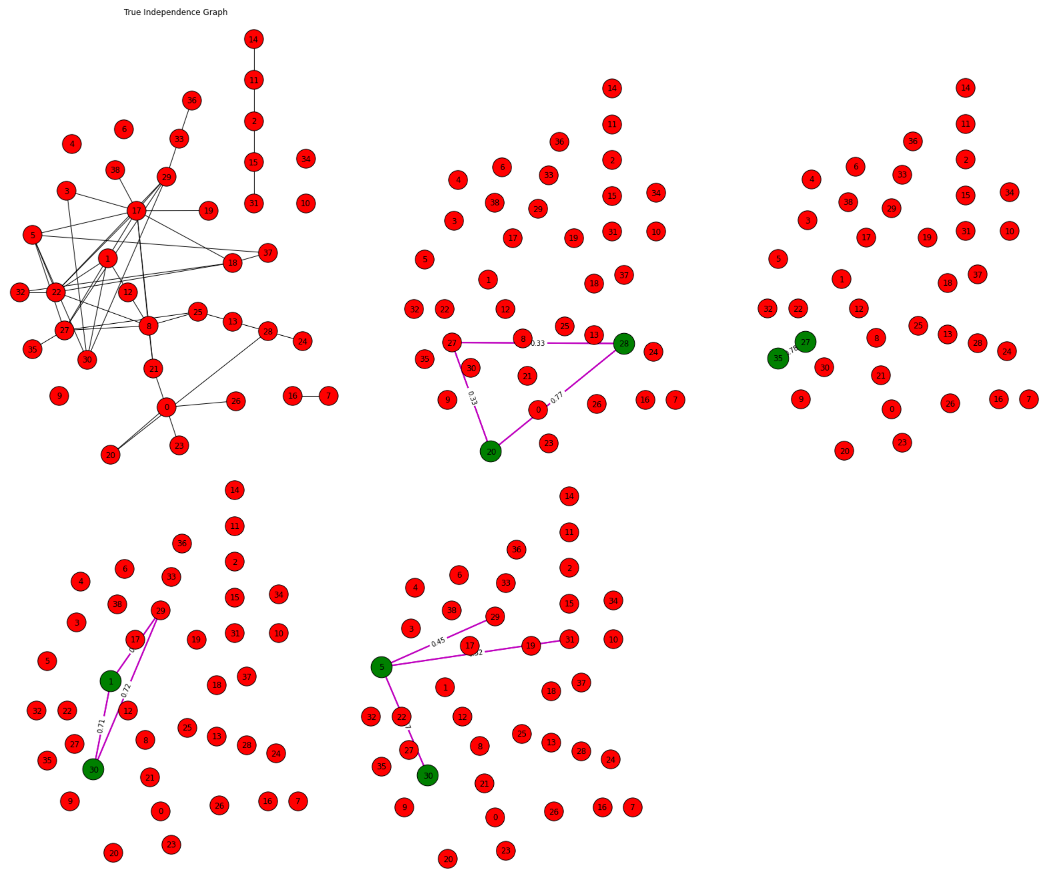

Figure 6: True graph structure and top activated partials corresponding to the covariance input for top activated outputs of the network. The two green nodes indicate the connection being evaluated, the magenta edges show the top partials corresponding to the input. We see the network is learning to associate outputs and inputs (not specified in any way) and potentially explore correlated nodes amongst the considered ones

A Supplementary Experiments

A.1 Analyzing the Network Jacobian

In [15] a method for loosely obtaining the attention of a trained neural network is used. The jacobian matrix of partial derivatives can be used for a given input to get an idea of which entries in the input matrix most affect the given outputs. We use this approach to begin to analyze the behavior of our DeepGraph-39 network. We gave one empirical covariance matrix to the network which was generated from the underlying graph specified in 6. We first note the jacobian is concentrated in just a few values, the top activated edges for the input closely correspond to the higher value partials in the overall jacobian matrix.

For the most activated output edges we visualize the entries or edges in the input (which can also be thought of as entries in a weighted adjacency matrix) that are most activated. We do this by taking a softmax of all partials for a given output edge and only keeping outliers (2 standard deviations). In 6 we show the top partials for 4 of the top output edges.

The relevant covariance entry corresponding to the edge at the output is almost always amongst the top partial values. We note that there is no explicit association of input relations outputs specified before or during training of the network so it has learned to associate the input and outputs. When there are other covariance entries pointed to they are usually connected to the relevant nodes either directly or indirectly. This may indicate the network is learning an algorithm to more succintly represent the connections of the two nodes.

	mean $\|\hat{\Sigma} - \Sigma\|_2^2$	mean $\|\hat{\Sigma} - \Sigma\|_\infty$
Empirical	0.0267	0.543
Graph Lasso	0.0223	0.680
DeepGraph	0.0232	0.673

Table 4: Covariance prediction of ABIDE data. Averaged over 50 trials of 35 samples from the ABIDE Control data

A.2 Predicting Covariance Matrices

Using our framework it is possible to attempt to directly predict an accurate covariance matrix given a noisy one constructed from few observations. This is a more challenging task than predicting the edges. In this section we show preliminay experiments which given an empirical covariance matrix from few observations attempts to predict a more accurate covariance matrix that takes into account underlying sparse data dependency structure.

One challenge is that outputs of our covariance predictor must be on the positive semidefinite cone, thus we choose to instead predict on the cholesky decompositions, which allows us to always produce positive definite covariances. We train the same DeepGraph-39 structure modifying the last layer to remove the non-linear sigmoid so that we can output covariance entry values, we train to predict on the cholesky decomposition of the true covariance matrices generated by our model with a squared loss.

We evaluate this network using the ABIDE dataset described in Section 3.3. The ABIDE data has a large number of samples allowing us to obtain a large sample estimate of the covariance and compare it to our estimator as well as graphical lasso and empirical covariance estimators. Using the large sample ABIDE empirical covariance matrix. We find that we can obtain competitive ℓ_2 and ℓ_∞ norm using few samples. We use 403 subjects from the ABIDE Control group each with a recording of $150 - 200$ samples to construct covariance matrix using a total of 77330 (some correlated) this acts as our very approximate estimate of the population Σ. We then evaluate covariance estimation on 35 samples using the empirical covariance estimator, graphical lasso, and DeepGraph trained to output covariance matrices. We repeat the experiment for 50 different subsamples of the data. We see in 4 that the prediction approach can obtain competitive results. In terms of ℓ_2 graphical lasso performs better, however our estimate is better than empirical covariance estimation and much faster then graphical lasso. In some applications such as robust estimation a fast estimate of the covariance matrix (automatically embedding sparsity assumptions) can be of great use. For ℓ_∞ error we see the empirical covariance estimation outperforms graphical lasso and DeepGraph for this dataset, while DeepGraph performs better in terms of this metric.

We note these results are preliminary, as the covariance predicting networks were not heavily optimized, moreover the ABIDE dataset is very noisy even when pre-processed and thus even the large sample covariance estimate may not be accurate. We believe this is an interesting alternate application of our paper.

A.3 Additional Synthetic Results on Sparsity

We investigate the affect of sparsity on DeepGraph-39 which has been trained with input that is between $4\% - 7\%$ sparse. We find that DeepGraph performs well relativel at the 2% sparsity level despite not seeing this at training time. At the same time performance begins to degrade for 15% but is still competitive in several categories. The results are shown in Table 5

Future investigation can consider how alternate variation of sparsity at training time will affect these results.

Experimental Setup	Method	logloss	AUC	Prec@2.5%	Prec@5%	AP	CE
Gaussian Random Graph (n=35, sparsity=2%)	GLasso	0.519	0.701	0.41	0.41	0.421	0.02
	Bdgraph	0.742	0.727	0.39	0.42	0.294	0.15
	DeepGraph-39-G	0.098	0.744	0.42	0.43	0.368	0.03
	DeepGraph-39-G+Perm	0.098	0.752	0.42	0.44	0.376	0.03
Gaussian Random Graph (n=35, sparsity=15%)	GLasso	8.038	0.598	0.91	0.84	0.537	0.34
	Bdgraph	2.289	0.601	0.76	0.74	0.504	0.36
	DeepGraph-39-G	1.448	0.617	0.76	0.7	0.493	0.35
	DeepGraph-39-G+Perm	1.401	0.626	0.85	0.76	0.511	0.35

Table 5: For each scenario we generate 20 graphs with 39 nodes, and corresponding data matrix sampled from distributions with those underlying graphs. The number of samples is indicated by n.