Relate and Predict: Structure-Aware Prediction with Jointly Optimized Neural Dependency Graph

Arshdeep Sekhon [1] **Zhe Wang** [1] **Yanjun Qi** [1]

Abstract

Understanding relationships between feature variables is one important way humans use to make decisions. However, state-of-the-art deep learning studies either focus on task-agnostic statistical dependency learning or do not model explicit feature dependencies during prediction. We propose a deep neural network framework, dGAP , to learn neural dependency Graph and optimize structure-Aware target Prediction simultaneously. dGAP trains towards a structure self-supervision loss and a target prediction loss jointly. Our method leads to an interpretable model that can disentangle sparse feature relationships, informing the user how relevant dependencies impact the target task. We empirically evaluate dGAP on multiple simulated and real datasets. dGAP is not only accurate, but can also recover the correct dependency structure.

1. Introduction

Cognitive psychologists have identified relational structure as one primary component humans rely on to tackle unstructured problems (Halford et al., 2010). Relational representations are the foundation in higher cognition. One primary relational thinking describes complex systems as compositions of entities and their interaction graphs. In this paper, we borrow such an idea and design graph-oriented relational representation learning into state-of-the-art deep neural networks. Learning such structured representations from data can provide semantic clarity, ease of reasoning for generating new knowledge, and possibly causal interpretation.

Existing deep learning literature has proposed effective ways, like using graph neural networks, to represent data when relational graphs are known apriori (Zhou et al., 2018). However, little attention has been paid to address cases when the underlying relation graph is unknown. We, therefore, ask a question: is it possible to learn graph-based relational knowledge from data, for both knowledge communication and prediction using knowledge at the same time.

Our method, we name dGAP , jointly learns neural dependency Graph and exploits the inferred graph for structure-Aware target Prediction. dGAP is trained end-to-end by optimizing the task-specific loss plus a structure self-supervision loss as regularization. Our method extracts knowledge of a target task at a macro level, and at the same time using the learned knowledge to conduct reasoning. In summary, we make the following contributions:

- dGAP exploits an explicit neural dependency module to model and learn variable interactions relevant to a task we care. Our design is in a direction towards making deep learning more human-like, since our neural dependency network-oriented design is consistent with the way humans organize knowledge for higher cognition.
- Our method leads to an interpretable model that can disentangle sparse feature relationships, informing the user how relevant dependencies impact the target task. Having a structure not only helps understand the problem at a macro level, it also helps to pinpoint areas that require deeper understanding.
- We empirically evaluate dGAP on multiple simulated and real-world datasets. dGAP predicts accurately and can discover task-oriented knowledge. On simulated cases we empirically prove that the discovered graphs match well with the true dependency networks, outperforming state-of-the-art baselines with a significant gain.

Like a tourist needs a map, human thinking requires structure. We borrow such an intuition in the design of dGAP . To the authors' best knowledge, dGAP is the first deep learning architecture that extracts knowledge in the form of neural dependency graph and conducts knowledge based predictions at the same time.

2. Method

Notations: We denote the input data matrix by $\boldsymbol{X} \in \mathbb{R}^{n \times p}$

[1]Department of Computer Science, University of Virginia. Correspondence to: Yanjun Qi <yanjun@virginia.edu>.

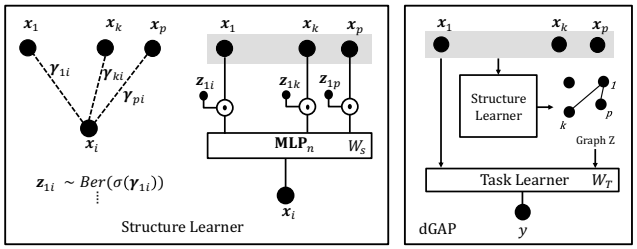


Figure 1. A schematic diagram representing dGAP .

where n represents the number of samples, and p represents the number of input nodes or variables. We represent the corresponding labels $\boldsymbol{Y} \in \mathbb{R}^{n \times C}$, where C indicates the number of classes. We denote the discrete binary graph between the variables by $\boldsymbol{Z} \in \mathbb{R}^{p \times p}$. We denote one individual sample and its corresponding label by $\boldsymbol{x}; \boldsymbol{y}$.

Our main objective is to learn undirected dependencies between input variables jointly with a target supervised task. Accordingly, dGAP has two modules: a Structure Learner $\mathbb{S}$ and a Task Learner $\mathbb{T}$. In the following subsections, we describe in detail these two modules.

2.1. Structure Learner ($\mathbb{S}$)

The goal of this module is to learn graph-oriented structure representation about the input variables. We choose to learn generic undirected dependency graphs via a neural network module.

To learn these structures, we introduce binary variables $\boldsymbol{Z} = \{\boldsymbol{Z}_{i,j}\}$, that represent the dependency structure between the relevant variables (aka a neural dependency graph $\boldsymbol{Z}$). $\boldsymbol{Z}_{i,j}$ specifies the presence or absence of a dependency between variable i and j. Thus, the state of without a dependency can be encoded by multiplying the corresponding parent node by 0. The neighbors of each input variable i is given by $\boldsymbol{Z}_{:,i}$. Finding the optimal dependency structure corresponds to a search over all configurations of $\boldsymbol{Z}$. Besides we need to optimize the other learnable model parameters for each possible dependency structure in the search space. To avoid this intractable procedure, we parametrize $\boldsymbol{Z}$ by placing a distribution over $\boldsymbol{Z}$. This enables us to directly optimize the parameters of this distribution to arrive at a data-driven optimal dependency structure. In detail, $p\ (\boldsymbol{x}; \boldsymbol{Z}) = p\ (\boldsymbol{x}|\boldsymbol{Z})p(\boldsymbol{Z})\ _{\boldsymbol{Z},\boldsymbol{Z}'}$ where $_{\boldsymbol{Z},\boldsymbol{Z}'}$ is the Kronecker delta, which effectively selects a single structure. Connecting from Eq. 7, to optimize the pseudo log likelihood of the data distribution using dependency structure $\boldsymbol{Z}$:

$$\ell_p(\theta) = \sum_{s=1} \log(p\ (\boldsymbol{x}_s|\boldsymbol{Z})) \approx \sum_{s=1} \sum_{i=1} \log p\ (\boldsymbol{x}_{si}|\boldsymbol{x}_{s,-i}; \boldsymbol{Z}_{:,i}) \tag{1}$$

Now, we treat each $\boldsymbol{Z}_{i,j}$ as an independent random variable, sampled from a Bernoulli distribution with mean $_{i,j}$, i.e. $\boldsymbol{Z}_{i,j} \sim p(\boldsymbol{Z}_{i,j}) = \text{Ber}(\boldsymbol{Z}_{i,j}\ )$. We denote the mean parameter of this Bernoulli distribution for all edges as a matrix γ. In detail, we introduce a learnable parameter matrix $\gamma \in \mathbb{R}^{p \times p}$. The sigmoid of the ij^{th} entry in this matrix, (γ_{ij}), indicates the Bernoulli probability of an interaction edge between the input i^{th} and j^{th} node. $\boldsymbol{Z} \sim (Ber(\ (\gamma)))$ defines binary graphs, representing dependencies between the input features.

In summary, the graph $\boldsymbol{Z}$ acts as a mask to reconstruct the input in a self supervision setting. We reconstruct $\boldsymbol{x}_i$ from $[\boldsymbol{Z}_{j,i} \odot x_j]\{j = \{1;\cdots;p\};\ j \neq i\}$. In other words, for feature i, we take all other features as input features masked by $\boldsymbol{Z}_{:,i}$ to predict feature i via a function (MLP in our case). Mathematically, $\boldsymbol{x}_i^1 = [\boldsymbol{Z}_{1,i}x_1;\cdots;\boldsymbol{Z}_{p,i}x_p]; \forall i \in \{1;\cdots;p\}; i \neq j$.

Self-supervision loss: We use $\boldsymbol{x}_i^1$, obtained using the input masked by the neighbors learnt by dependency graph $\boldsymbol{Z}_{:,i}$, as input to an MLP to reconstruct $\boldsymbol{x}_i'$: $\boldsymbol{x}_i = MLP_i(\boldsymbol{x}_i^1)$. We use reconstruction loss $\ell_{struct} = \sum_{i=1}^{p} ReconstructionLoss(\boldsymbol{x}_i; \boldsymbol{x}_i')$. If $\boldsymbol{x}_i \in \mathbb{R}$, we use Mean Squared Error Loss for this self-supervision based reconstruction. When on categorical input variables $\boldsymbol{X}_i \in \mathbb{R}^L$, where L is the number of categories in the categorical variable $\boldsymbol{X}_i$, we use Negative Log Likelihood Loss as Reconstruction loss.

M-dGAP : $\boldsymbol{Z}$ may differ depending on data samples. While some interactions may be present in all the samples irrespective of the target class it belongs to, some interactions may be specific to a cluster of samples. We denote this variation as MultiGraph-dGAP (in short, M-dGAP) and explain it in detail in Appendix C.

2.2. Task Learner ($\mathbb{T}$)

We want to explicitly incorporate the dependency structures into our target objective. We want to encourage discovery of interactions that are specifically relevant to the end task. For this purpose, we use Graph Attention Networks(GAT) as an interface to jointly represent dependency structures and input features.

For each specific sample, we represent each of its i-th feature node $\boldsymbol{x}_i$ as a position-specific embedding vector $\boldsymbol{h}_i^0$: $\boldsymbol{h}_i^0 = [\boldsymbol{x}_i||\boldsymbol{W}_i^{pos}]$, where $\boldsymbol{W}^{pos} \in \mathbb{R}^{p \times d_{pos}}$. Here, $[\odot||\odot]$ denotes concatenation. After these node specific embeddings, we use graph attention to combine the discrete graph $\boldsymbol{Z}$ and input embeddings $\boldsymbol{h}_i^0$ $i \in \{1;\cdots;p\}$. In detail, we learn a graph attention(Veličković et al., 2017) using the following equation for each edge $\boldsymbol{Z}_{i,j}$:

$$\alpha_{ij,k}^l = \frac{exp(LeakyReLU(\boldsymbol{a}_e^T[\boldsymbol{W}_k^l\boldsymbol{h}_i^{l-1}||\boldsymbol{W}_k^l\boldsymbol{h}_j^{l-1}]))}{\sum_{j=1}^{p} exp(LeakyReLU(\boldsymbol{a}_e^T[\boldsymbol{W}_k^l\boldsymbol{h}_i^{l-1}||\boldsymbol{W}_k^l\boldsymbol{h}_j^{l-1}]))} \tag{2}$$

$$\boldsymbol{h}_i^l = (\sum_{j \in N_i} \alpha_{ij,k}^l \boldsymbol{W}_k^l \boldsymbol{h}_j{}^{l-1}) \tag{3}$$

Via the above equation, the embedding of node i is updated by an attention weighted sum of its neighbors' embedding. More specifically, we use a multi head attention based formulation: $\boldsymbol{h}_i^l = ||_{k=1}^{K} (\sum_{j \in N_i} \boldsymbol{\alpha}_{ij,k}^l \boldsymbol{W}_k^l \boldsymbol{h}_j^{l-1})$ Here, the neighbors for a node i are obtained using $\boldsymbol{Z}_{:,i}$, k denotes the k-th attention head out of a total of K heads, and $||$ denotes concatenation including all K heads based embedding. l represents the index of graph attention layers (steps). Eq. 2 are repeated for L layers. This gives us the final representation $\boldsymbol{h}_L \in \boldsymbol{R}^{p \times d}$. To represent all the nodes together for a sample $\boldsymbol{x}$, we use a CLS node based pooling(Devlin et al., 2018). That is, we add an additional node that is connected to all other nodes in the graph. The final representation of this node, followed by a single layer MLP and a log softmax, makes the final graph level prediction $\hat{\boldsymbol{y}}$. The loss from the end task $\ell_{task} = \ell_{pred}(\boldsymbol{y}, \hat{\boldsymbol{y}})$. Additional details regarding classification with multiple graphs for variation M-dGAP are covered in Appendix Section C.

2.3. Training and Loss

We additionally use a sparsity constraint $\ell_{sparse} = \sum_{i=1}^{p} \sum_{j=1}^{p} (\quad_{ij})$ as a regularization to encourage learning of a sparse $\boldsymbol{Z}$. To sample discrete graphs, we use a gumbel softmax (Jang et al., 2016; Maddison et al., 2016) trick to sample from Bernoulli distribution parametrized by (γ).

$$\boldsymbol{Z}_{i,j} = \frac{exp(\log(\quad_{ij}) + \quad_1)/\quad)}{exp(\log(\quad_{ij} + \quad_1)/\quad) + exp(\log(1 - \quad_{ij} + \quad_2)/\quad)} \tag{4}$$

where $\quad_1$ and $\quad_2$ are i.i.d samples drawn from a Gumbel(0, 1) distribution and $\quad$ is a temperature parameter. The Gumbel-Softmax distribution is differentiable for $\quad > 0$. This allows us to sample discrete graphs for the graph attention network, while being able to propagate gradients to the learnt parameter γ both from the structure learner ($\boldsymbol{W}_S$) as well as the task learner ($\boldsymbol{W}_T$). To optimize the model, we minimize $\ell = \ell_{task} + \quad_{struct} \ell_{struct} + \quad_{sparse} \ell_{sparse}$ using Adam optimizer to train all components: γ, $\boldsymbol{W}_S$ and $\boldsymbol{W}_T$ together.

2.4. Connecting to related studies

We group the related work[1] into (1) Classical statistical methods to learn dependency relationships, probabilistic graphical models (PGM)(Lachapelle et al., 2019; Zheng et al., 2018; 2019; Magliacane et al., 2018; Ke et al., 2019). (2) DNN based studies that aimed to estimate relationships among input variables, but their relations are not about dependency structure(Kipf et al., 2018; Franceschi et al., 2019). (3) Methods that tried to disentangle variable interactions learnt by deep neural models from a post-hoc interpretation perspective(Tsang et al., 2017; 2020; Friedman et al., 2008; Sorokina et al., 2008; Lundberg et al., 2018; Cui et al., 2019; Janizek et al., 2020). These methods are different from dGAP , because our method trains end-to-end to discover and utilize the interactions for the task at hand jointly. We have covered these in detail in Appendix B.

3. Experiments

Data, Baselines and Evaluation Metrics: We group our experiments into simulation and real world datasets. The prediction tasks include both regression and classification. We show dGAP can achieve state-of-the-art prediction performance, and also discover meaningful dependencies in the data. For evaluation, we use Root Mean Squared Error (RMSE) for regression and Area under Curve (AUC) for classification. For the evaluation of estimated dependency structure on simulated cases, we report average AUC. Our baselines include Quadratic Discriminant Analysis (QDA) and MLP. We also compare against a GAT without structure learning (GAT-FC). Here, we use a fully connected graph as input to GAT. For evaluating graph recovery, we compare against Neural Interaction Detection(Tsang et al., 2017)(NID). [2]

3.1. Simulated: Gaussian-based Classification Datasets

We consider a simple binary classification task. For each class c, we use multivariate normal distribution $\mathbb{N}(0; \quad_c^{-1})$ to generate simulated samples. The precision matrix (inverse of covariance matrix) $\quad_c$ serves as the ground truth dependency graph . We consider three cases: $\{p = 5; 10; 20\}$. We show the $p = 5$ case in Figure 2, along with convergence of γ during training. Table 1 shows the classification and graph recovery performance. We have included more details about data generation in Appendix Section D .

3.2. Real World Datasets

Cal Housing Regression Dataset(Pace & Barry, 1997): For this regression dataset, we learn one graph. Figure 3(a) shows the graphs learnt by dGAP and Table 2 shows the RMSE results. **Qualitative Interpretation:** Our method is able to successfully learn a relationship between latitude and longitude which is important for house value prediction, as shown in Figure 3(a). We also show the case where we use solely the task loss to learn a graph in Figure 8(b). While some interactions are similar to Figure 3(a), without the self-supervision loss, we cannot recover a strong relationship between latitude and longitude.

[1] We have covered the detailed related work and background about dependency structure learning in detail in the Appendix Section A and B.

[2] Owing to space considerations, we have moved details explaining our choices of baselines, ablations, hyperparameters and additional datasets in Appendix.

Models, Baselines, Ablations and Variations	AUC (p = 5)	AUC (p = 10)	AUC (p = 20)	Graph-AUC (p = 5)	Graph-AUC (p = 10)	Graph-AUC (p = 20)
dGAP	0.7132	0.7145	**0.8491**	**1.0**	0.9979	0.9957
dGAP-NSS w/o sparsity	0.7139	0.7162	0.8479	0.4667	0.4713	0.4462
dGAP-NSS	0.7144	**0.7192**	0.8471	0.4521	0.4602	0.5297
dGAP-GCN	0.6324	0.6191	0.6635	**1.0**	0.9997	0.9973
M-dGAP	**0.7147**	0.7142	0.8467	**1.0**	**1.0**	**1.0**
M-dGAP-NSS w/o sparsity	0.7135	0.7153	0.8443	0.3396	0.5680	0.5912
M-dGAP-NSS	0.7139	0.7165	0.8453	0.6021	0.4567	0.4671
M-dGAP-GCN	0.6479	0.6141	0.6614	**1.0**	**1.0**	**1.0**
NID	NA	NA	NA	0.6250	0.6526	0.6300
GAT-FC	0.7145	0.7051	0.8489	NA	NA	NA
MLP	0.7161	0.7193	**0.8548**	NA	NA	NA
QDA	**0.7178**	**0.7252**	0.8215	NA	NA	NA

Table 1. Classification Results Area under Curve (AUC) on test data and Evaluation on Graph Estimations for simulation datasets averaged across 5 random seeds.

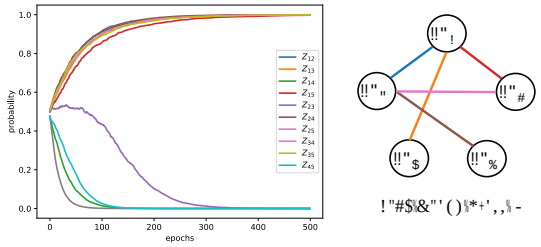
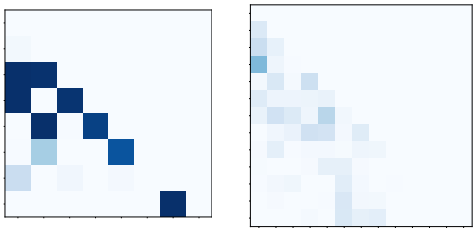


Figure 2. On Simulation Data p = 5: We show convergence during training of the Graph parameters ($_{ij}$) indicating the probability of the edge in the graph structure. We show the True Graph Class A adjacent to the convergence graphs. The true edges converge to probability 1:0 and the true non-edges converge to probability 0:0.

Heart Disease Prediction(Detrano et al., 1986): For this binary classification dataset, we show the graphs learnt by our method in Figure 3(b) and AUC(Area Under Curve) results obtained in Table 2. Our model is able to achieve similar AUC performance along with successfully disentangling meaningful dependencies.**Qualitative Interpretation:** Figure 3 and Figure 9(Appendix Section D) show the graphs learnt by our dGAP . We show a stronger link between Maximum Heart Rate (thalach) and Age in healthy class (Figure 9(b)), whereas a weaker link in heart disease patients (Figure 9(c)). We show this differential case explicitly in Figure 3(b).

Models, Baselines, Ablations and Variations	Housing (RMSE)	Heart (AUC)	Letter (AUC)	Text (AUC)
dGAP	**0.4967**	0.8616	0.9985	0.9010
dGAP-NSS	0.5040	0.8804	0.9981	0.9076
dGAP-NSS w/o sparsity	0.5061	0.8643	0.9984	**0.9078**
dGAP-GCN	0.9155	0.8966	0.7233	0.9045
M-dGAP	NA	0.8482	0.9983	0.8534
M-dGAP-NSS	NA	0.8481	0.9983	0.8496
M-dGAP-NSS w/o sparsity	NA	0.8607	**0.9988**	0.8533
M-dGAP-GCN	NA	**0.9016**	0.7274	0.8446
GAT-FC	0.5043	0.8692	**0.9985**	**0.9075**
MLP	**0.5028**	**0.8884**	0.9981	0.8296

Table 2. Real Data Results. For Housing Dataset, we report Root Mean Squared Error (RMSE), For Heart, Letter and Text datasets, we report Area Under Curve (AUC).



(a) (b)

Figure 3. On housing and heart disease data: Heatmaps of pairwise interaction strengths learnt by dGAP and variations. (a) The dependency graph learnt by dGAP for housing dataset. Our method is able to successfully learn a relationship between latitude and longitude; (b) Differential between Class specific graphs learnt by M-dGAP for heart disease prediction dataset. We show a stronger link between Maximum Heart Rate (thalach) and Age in healthy case, whereas a weaker link in disease case. We show this differential graph explicitly in (b).

Sentiment Classification Text Dataset(Socher et al., 2013) : We also evaluate dGAP on Stanford Sentiment Treebank (SST-2) binary sentiment classification task. We use pretrained GloVe (Pennington et al., 2014) embeddings both for the structure learner as well as the task learner. We show the graph learnt by dGAP in Figure 11(Appendix Section D) and classification AUC in Table 2. **Qualitative Interpretation:** As shown in Figure 11, local neighbors depend on each other(indicated by the thick orange diagonal). The sentences are variable length accounting for the diagonal Y shape.

4. Conclusion

This paper proposes an end-to-end deep learning framework, dGAP , to learn graph-structured representations that help in predictions. We can view dGAP in two different ways: (1) An approach for discovering task-oriented structured representations that are critical components contributing towards human-like cognitive data modeling. (2) We can also view dGAP as being “modular" interpretable (Murdoch et al., 2019), that is, a model in which a meaningful portion of its prediction-making process can be interpreted independently.

References

Besag, J. Efficiency of pseudolikelihood estimation for simple gaussian fields. *Biometrika*, pp. 616–618, 1977.

Cui, T., Marttinen, P., and Kaski, S. Recovering pairwise interactions using neural networks. *arXiv preprint arXiv:1901.08361*, 2019.

Detrano, R., Salcedo, E. E., Hobbs, R. E., and Yiannikas, J. Cardiac cinefluoroscopy as an inexpensive aid in the diagnosis of coronary artery disease. *American Journal of Cardiology*, 57(13):1041–1046, 1986.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. *arXiv preprint arXiv:1903.11960*, 2019.

Friedman, J. H., Popescu, B. E., et al. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2 (3):916–954, 2008.

Halford, G. S., Wilson, W. H., and Phillips, S. Relational knowledge: the foundation of higher cognition. *Trends in cognitive sciences*, 14(11):497–505, 2010.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Janizek, J. D., Sturmfels, P., and Lee, S.-I. Explaining explanations: Axiomatic feature interactions for deep networks, 2020.

Ke, N. R., Bilaniuk, O., Goyal, A., Bauer, S., Larochelle, H., Pal, C., and Bengio, Y. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Magliacane, S., van Ommen, T., Claassen, T., Bongers, S., Versteeg, P., and Mooij, J. M. Domain adaptation by using causal inference to predict invariant conditional distributions. In *Advances in Neural Information Processing Systems*, pp. 10846–10856, 2018.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

Pace, R. K. and Barry, R. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sorokina, D., Caruana, R., Riedewald, M., and Fink, D. Detecting statistical interactions with additive groves of trees. In *Proceedings of the 25th international conference on Machine learning*, pp. 1000–1007, 2008.

Staiger, D., Kaulen, H., and Schell, J. A cacgtg motif of the antirrhinum majus chalcone synthase promoter is recognized by an evolutionarily conserved nuclear protein. *Proceedings of the National Academy of Sciences*, 86(18): 6930–6934, 1989.

Tsang, M., Cheng, D., and Liu, Y. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

Tsang, M., Cheng, D., Liu, H., Feng, X., Zhou, E., and Liu, Y. Feature interaction interpretability: A case for explaining ad-recommendation systems via neural interaction detection. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkgnhTEtDS.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pp. 9472–9483, 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. P. Learning sparse nonparametric dags. *arXiv preprint arXiv:1909.13189*, 2019.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434, 2018.

A. Background: Dependency Structure Learning

Learning the dependency structure between input variables is an important task in machine learning. Traditionally, statistical Graphical Models intuitively represent relationships between random variables as conditional dependence graphs. In detail, the variables of interest are represented by a set of nodes V, with relationships as edge set E in graph $G = (V; E)$. The edges E represent conditional dependencies between the variables. Essentially, learning these relationships is equivalent to learning a factorization of the joint probability distribution. One notable method that uses representation of undirected dependency graphs for a joint probability distribution is the Markov Random Field(MRF).

An MRF models a set of random variables to have a joint probability stipulated by an undirected graph $G = (V; E)$. These models satisfy the markov properties: (1) Pairwise Markov Property: Any two non-adjacent variables are conditionally independent given all other variables: (2) Local Markov Property : A variable is conditionally independent of all other variables given its neighbors. (3) Any two subsets of variables are conditionally independent given a separating subset. Discovering the MRF dependency structure G is a reconstruction of the graphical structure of a Markov Random Field from independent and identically distributed samples.

Assuming a dataset $D = (x^1; : : : ; x^n)$, where $x^s \in R^p$, that is sampled i.i.d. from the parametric distribution p, a maximum likelihood estimator is de ned as $_n$:

$$\ell_n(\theta_n; D) = \sum_{s=1}^{n} \log p_\theta(x^s) \tag{5}$$

To make the learning process more tractable, (Besag, 1977) proposed to use pseudo likelihood as an approximation. Pseudo-log likelihood(Besag, 1977) offers a tractable replacement for the likelihood, de ned as $_p$:

$$\ell_p(\theta_p; D) = \sum_{s=1}^{n} \sum_{i=1}^{p} \log p_\theta(x_i^s | x_{\setminus i}^s) \tag{6}$$

In case of a Markov Random Field, this reduces to:

$$\ell_p(\theta_p; D) = \sum_{s=1}^{n} \sum_{i=1}^{p} \log p_\theta(x_i^s | x_{N(i)}^s) \tag{7}$$

Here $N(i)$ stands for neighbors of node i in the dependency structure G.

B. Connecting to related studies

One classic statistical way to learn dependency relationships from data was from the family of probabilistic graphical models (PGM). These models represent the random variables as a node set V and the relationships between variables as an edge set E. Edges denote conditional dependencies among variables. Learning PGM is equivalent to learning a factorization of the joint probability distribution. PGM roughly fall to two groups: (1) undirected PGM, for which a non exhaustive list includes Gaussian Graphical Models, Ising Models, etc; (2) Directed PGM that tries to model and learn directed conditional dependency structure, normally plus acyclic graph constraints. A few recent studies (Lachapelle et al., 2019; Zheng et al., 2018; 2019) learned directed dependencies via deep neural networks (DNN). In many cases of PGM learning, sparsity regularization is used to encourage a final sparse graph. Besides, few recent literature aims to learn causal Bayesian networks (Magliacane et al., 2018; Ke et al., 2019) via DNN based formulations. In the absence of intervention distributions, these causal methods can only learn a Markov equivalence class of the true networks. All studies in this category perform unsupervised estimation, that is they don't jointly optimize a task-specific loss when learning the dependency structures from data.

A few recent DNN studies also aimed to estimate relationships among input variables, but their relations are not about dependency structure. For instance, Neural Relational Inference (NRI) (Kipf et al., 2018), is a neural network based interaction learning framework designed for modeling dynamical interacting systems. NRI learns sample-specific pairwise interactions between input units using a physics perspective. NRI represented the edges as latent variables in a Variational Autoencoder(VAE) framework. Relational graphs in NRI are sample-conditioned and are learned via unsupervised alone, that is the learnt interactions are not informed by a target task objective. Differently, another recent study, LDS from (Franceschi et al., 2019) proposed to jointly learn the parameters of graph convolutional networks (GCNs) and the graph structure (as unknown hyperparameter of GCN) by approximately solving a bilevel program. The whole learning is guided by a downstream node classification objective. Comparing to ours, the graph LDS has no dependency semantics and the whole formulation only applies to GCN.

Our method also connects to an array of methods that tried to disentangle variable interactions from deep neural models from a post-hoc interpretation perspective. For instance, (Tsang et al., 2017) detects interactions learnt by multilayer perceptron networks (MLP) by decomposing the weights of the MLP. Similarly, (Cui et al., 2019) estimates global pairwise interaction effects of a MLP model using Bayesian parameter analysis. Another closely related work (Tsang

et al., 2017; 2020) focused on discovering interactions between sparse features and then explicitly encoding them for explaining MLP based recommendation neural models in a two stage approach. All of these methods are post-hoc interpretation methods, and most are restricted to specific DNN architectures. Loosely related, another set of feature interaction based interpretation studies have been proposed in the literature to explain decision-tree based learning methods (Friedman et al., 2008; Sorokina et al., 2008; Lundberg et al., 2018). Notably, two recent studies from the post-hoc interpretation category stack the discovered interactions (Tsang et al., 2017; 2020) into a second stage of prediction task in order to improve prediction performance. These methods are different from dGAP , because our method trains end-to-end to discover and utilize the interactions for the task at hand jointly.

Besides, most of the post-hoc interaction-based interpretation methods focused on learning feature interactions for each sample , like (Janizek et al., 2020). dGAP , instead, provides global dependency explanations. Global explanations are desirable for interpretable data and model summarization(Tsang et al., 2017).

C. dGAP variation: M-dGAP

Z may differ depending on data samples. While some interactions may be present in all the samples irrespective of the target class it belongs to, some interactions may be specific to a cluster of samples. To incorporate one kind of this variability, we use the training data labels to construct class specific graphs. For example, in the case of classification, if $y \in \{1, \ldots, C\}$, we introduce a total of C learnt dependency graphs Z_c, where $c \in \{1, \ldots, C\}$, where C is the number of classes. During the training step, we use the sum of the class specific losses: $\ell_{struct} = \sum_{c=1}^{C} 1_{y=c} \ell_{struct}(x, S_c(x))$ We denote this variation as MultiGraph-dGAP (short as M-dGAP).

For incorporating these multiple graphs,we use Graph Attention Networks one for each of $Z_c \forall c \in \{1, \ldots, C\}$. This gives us output representations $h^L_{CLS_c}$, one for each class. For the final classification, we add another attention layer on the$c$ representations. In detail, we use a context vector$v \in \mathbb{R}^d$. We obtain attention scores $\alpha^c = Softmax_{1,\ldots,C}(v \cdot h^L_{CLS_c})$. Finally, we use$h_{out} = \sum_{c=1}^{C} \alpha_c h^L_{CLS_c}$ as input to the MLP layer for classification.

Additional dGAP Details : We want to point out that modeling a binary graph with prior matrix allows us to have a probabilistic interpretation of each edge in the graphs we learn. Additionally, the combination of a global (population level) dependency graph along with sample level attention (on each edge learnt by GAT) allows us to model the global binary dependency structure, then refined with local (sample level) fine-tuning regarding which edges were most important for a specific prediction. An easy analogy: learning without a structure is like a tourist without a map. The graph we learn is the map for everyone and how to use the map depends on each traveler's choice (being modelled as graph attention for each sample).

D. Experiments

Data: We group our experiments into simulation and real world datasets. The prediction tasks include both regression and classification. On 3 simulated and 5 real-world datasets, we show dGAP can achieve state-of-the-art prediction performance, and also discover meaningful dependencies in the data. The real-world datasets covered our experiments include both tabular inputs and text inputs.

Evaluation Metrics: When evaluating prediction performance for a regression task, we use Root Mean Squared Error (RMSE) on test set. When evaluating prediction for a classification task, we report area under curve (AUC) on test set. For the evaluation of estimated dependency structure on simulated cases (in which we know the true dependency graphs), we report average AUC to compare estimated graph probability (or interaction score from baselines) with the true binary graphs.

Baselines: (1) QDA: We use QDA as one baseline on simulation cases as it is close to ground truth for classifying Gaussian Datasets with distinct covariance matrices. (2) MLP: we compare against multi-layer perceptron networks (MLP). (3) GAT-FC: We compare against a GAT without graph training (GAT-FC). Here we use a Fully Connected graph to GAT. For Graph Recovery, we compare against Neural Interaction Detection(Tsang et al., 2017)(NID).

Ablation Variations: To understand how each component impact dGAP , we explore the following:

dGAP : Basic version with one graph.
dGAP-GCN : In this variation, dGAP is trained with GCN model for prediction.
dGAP-NSS : dGAP is trained without self-supervision loss. Concretely, $\alpha_{struct} = 0$. We only use the task loss with sparsity to train the graph.
dGAP-NSS w/o sparsity : In this variation, dGAP is trained without self-supervision loss and sparsity regularization is also removed. Concretely $\alpha_{struct} = 0$, $\alpha_{sparse} = 0$.
M-dGAP : Multiple graphs variation of dGAP .
M-dGAP-GCN : In this variation, M-dGAP is trained with GCN model for prediction.
M-dGAP-NSS : M-dGAP is trained without self-

supervision loss. .i.e., $_{struct} = 0$.

M-dGAP-NSS w/o sparsity : M-dGAP is trained without self-supervision loss. and sparsity regularization is also removed in M-dGAP-NSS .i.e., $_{struct} = 0$, $_{sparse} = 0$.

D.1. Simulated 2D Classi cation Datasets

We rst evaluate our model on a simple 2D dataset. We consider the binary classi cation of the dataset shown in Figure 4(a). The task is to classify Black Points (denoted as Class A) from the Green Points (Class B). We generate Gaussian dataset samples from Class A using X_A $N(0; _A^{-1})$ and X_B $N(0; _B^{-1})$. Here, $_A = \begin{pmatrix} 1:0 & 0:99 \\ 0:99 & 1:0 \end{pmatrix}$ and $_B = \begin{pmatrix} 1:0 & 0:99 \\ 0:99 & 1:0 \end{pmatrix}$. In this case, both x_1 and x_2 have identical marginal distributions. The model needs to incorporate the interactions between x_1 and x_2 i.e. both the input variables to classify correctly. Note that this data has two types of interactions, in Class A the two features interact positively, while the points from Class B interact negatively. As shown in Figure 4, we are successfully able to discover the correct graphs. We compare the class speci c graphs from M-dGAP in Figure 4(c)(Class A) and Figure 4(d)(Class B) learnt using the dGAP in Figure 4(b). If we use a single shared graph, we are unable to recover a true representative set of interactions. Instead, our M-dGAP variation allows us to incorporate this class speci c variability. In terms of classi cation performance, dGAP is able to achieve a classi cation AUC of $0:9956$, M-dGAP of $0:9956$ and a 2 layer MLP is able to achieve $0:9957$.

Hyper parameters: We use hidden size $task_H = 8$, layers $task_L = 2$, heads $K = 2$, structure learner's hidden size $func_H = 8$ and $d_{pos} = 4$. We compare to an MLP with 2 hidden layers and hidden size 16. We use $`_{sparse} = 0:0$ and $`_{struct} = 10:0$. We train the models for 250 epochs. To optimize the model, we use Adam optimizer with learning rate $0:001$. We report the average performance 3 random seeds.

D.2. Simulated: Gaussian-based Classi cation Datasets

We consider a simple binary classi cation task. For each class, we use a multivariate normal distribution to generate simulated samples. For a multivariate normal distribution, the precision matrix , inverse of the covariance matrix, is representative of the conditional dependency graph. The values indicate partial correlation between two variables, while the zero entries represent conditional independence given all other variables (therefore no edge in the graph). Specifically, $_{ij} = 0$ if and only if X_i and X_j are conditionally independent given all other coordinates of X.

In detail, we simulate two dependency graphs being sampled as Erdos Renyi Graphs: matrices and $R_l \in R^{p \times p}$, with probability p_d and p_i respectively. Then we generate data from two classes A and B using the following equations: $_A = + R_l + _d I$, and $_B = R_l + _d I$. $_c$ and $_d$ are selected large enough to guarantee positive de niteness. We generate Gaussian samples using X_A $N(0; _A^{-1})$ and X_B $N(0; _B^{-1})$.

We consider three cases $p = 5$, $p = 10$ and $p = 20$. For $p = 5$, we consider the graphs shown in Figure 5. For $p = 10$, we consider $p_d = 0:3$ and $p_i = 0:2$. For $p = 20$, we consider $p_d = 0:3$ and $p_i = 0:1$. We use training samples $n = 40000$, and validation and test samples, each as 8000. In Table 3, we present the classi cation results of dGAP and variations. For the task learner, we use $task_H = 32$, $K = 4$, $task_L = 2$, $d_{pos} = 16$. We compare to Quadratic Discriminant Analysis (QDA) and a two layer MLP with hidden size 128 with similar number of parameters(20000). We use $struct_H = 64$ and $struct_L = 1$. We pretrain the structure learner for 50 epochs using a fully connected graph with a dropout of $0:3$, to encourage the structure learner to use all input nodes for reconstruction. We report the results averaged across 5 random seeds. Table 3 shows our classi cation results(AUC) as well as our graph recovery results(AUC wrt the true graph) on Gaussian Datasets for all settings. We use bold text to represent the best variation from our model and from the baselines. We report the standard deviation across the seeds in the Appendix. Figure 5 shows that the Bernoulli parameters ( ) converge, with edges converging to probability $1:0$ and non edges to probability $0:0$.

Figure 5. On Simulation Data: We show convergence during training of the Graph parameters ($_{ij}$) indicating the probability of the edge in the graph structure. We show the ground truth graphs True Graph Class A and True Graph Class B adjacent to the convergence graphs. The true edges converge to probability $1:0$ and the true non-edges converge to probability $0:0$.

Figure 4.On Simulation: Learnt Graphs using dGAP : (a) 2D Gaussian Data (b) dGAP ; M-dGAP (c) Graph from Class A and (d) Graph from Class B.

Models, Baselines, Ablations and Variations	AUC (p = 5)	AUC (p = 10)	AUC (p = 20)
dGAP	0.7132(0.006)	0.7145(0.026)	0.8490(0.019)
dGAP-NSS w/o sparsity	0.7139(0.006)	0.7162(0.029)	0.8479(0.019)
dGAP-NSS	0.7144(0.006)	0.719(0.029)	0.8471(0.020)
dGAP-GCN	0.6324(0.028)	0.6191(0.030)	0.6635(0.032)
M-dGAP	0.7147(0.005)	0.7142(0.033)	0.8467(0.018)
M-dGAP-NSS w/o sparsity	0.7135(0.006)	0.7153(0:029)	0.8443(0.018)
M-dGAP-NSS	0.7139(0.005)	0.7165(0.029)	0.8453(0.020)
M-dGAP-GCN	0.6479(0.012)	0.6141(0.038)	0.6614(0.071)
GAT-FC	0.7137(0.006)	0.7051(0.014)	0.8489(0.020)
MLP	0.7161(0.006)	0.7193(0.029)	0.8548(0.017)
QDA	0.7178(0.004)	0.7252(0.026)	0.8215(0.021)

Table 3.Classi cation Results Area under Curve (AUC) on test data and Evaluation on Graph Estimations for simulation datasets averaged across5 random seeds.

Visualization:Figure 6 shows the different graphs learnt by ablations and variations of dGAP . In Figure 6(b), We show that the learnt graph for the single graph case dGAP corresponds to the shared graph entries in the ground truth graph. Figure 6(f) indicates the importance of the self-supervision loss as without this loss even with the sparsity assumption, we are unable to recover any correct edges at all. Similarly, Figure 7 shows the discovered graphs for multi graph variation M-dGAP forp = 20.

D.3. Real World Dataset: California housing Dataset

This is a regression data where the target variable is the median house value for California districts. The total number of training points are20640 The number of input features p = 8 . [3] We scale the features to zero mean and unit variance. We split the data from (Pace & Barry, 1997) into train, valid and test using 80/10/10 splits. As this is a regression dataset, we learn one graph. We show the graphs learnt by our method in Figure 8 and RMSE (Root Mean Squared Error) results in Table 2.

Hyperparameters: We use the validation data to select the best hyperparameters. We use an initial position embedding d_{pos} = 16 and$struct_H$ = 16 with a single layer in the structure learnerS. We vary$task_H$ in the set off 16; 32; 64g, $task_L$ in f 2; 4g and$n$heads in f 4; 8g. We use dropout of f 0:3g for the pretraining step in the structure learner. We use a learning rate of0:001 for the Adam Optimizer. We vary$`_{sparse}$ = f 0:001; 0:005; 0:01g. For the MLP baseline, we vary$task_L$ 2 f 2; 4; 6g and$task_H$ 2 f 16; 32; 64g. We choose the reported based on validation results.

Qualitative Interpretation: Figure 8(a) shows that our method is able to successfully learn a relationship between latitude and longitude which is important for house value prediction. We also show the case where we use solely the task loss to learn a graph in Figure 8(b). While some interactions are similar to (a), without structure loss we cannot recover a strong relationship between latitude and longitude.

[3]The relevant features are median income in block (MedInc), median house age in block (HouseAge), average number of rooms (AveRooms), average number of bedrooms (AveBedrms), block Population, average house occupancy (AveOccup), house block latitude(Latitude) and house block longitude (Longitude).

Models, Baselines, Ablations and Variations	Graph-AUC (p = 5)	Graph-AUC (p = 10)	Graph-AUC (p = 20)
dGAP	1.0(0.000)	0.9979(0.004)	0.9957(0.004)
dGAP-NSS w/o sparsity	0.4667(0.155)	0.4713(0.164)	0.4462(0.095)
dGAP-NSS	0.4521 (0.160)	0.4602(0.074)	0.5297(0.047)
dGAP-GCN	1.0(0.000)	0.9997(0.0014)	0.9973(0.0053)
M-dGAP	1.0(0.000)	1.0(0.000)	1.0(0.000)
M-dGAP-NSS w/o sparsity	0.3396(0.228)	0.5680(0.119)	0.5912(0.080)
M-dGAP-NSS	0.6021(0.177)	0.4567(0.059)	0.467(0.065)
M-dGAP-GCN	1.0(0.000)	1.0(0.000)	1.0(0.000)
NID	0.6250(0.088)	0.6526(0.026)	0.6300(0.026)

Table 4. Evaluation on Graph Estimations for simulation datasets averaged across 5 random seeds.

D.4. Real World Dataset: Heart Disease Prediction Dataset

We use the Cleveland heart disease dataset from (Detrano et al., 1986). The task is to predict whether or not a patient has coronary artery disease given demographic information like age, gender and clinical measurements. After preprocessing, the dataset contains 298 patients with 13 associated features. We split the data into training, validation and test randomly using 70/10/20 splits.[4] We show the graphs learnt by our method in Figure 9 and AUC(Area Under Curve) results obtained in Table 2. Our model is able to achieve similar AUC performance along with successfully disentangling dependencies.

Hyperparameters: To select hyperparameters, we use validation AUC for each random seed, and report the test AUC. We use an initial position embedding $d_{pos} = 16$ and $\text{struct}_H = 64$ with a single layer in the structure learner. We vary task_H in the set of $\{8, 16, 32\}$, task_L in $\{2, 4\}$ and nheads in $\{2, 4\}$. We use dropout of $\{0.3\}$ for the pretraining step in the structure learner. We use a learning rate of 0.001 for the Adam Optimizer. We use $\ell_{\text{sparse}} = 0.01$ and $\ell_{\text{struct}} = 1.0$. For the GCN variation, we vary task_H in the set of $\{16, 32, 64, 128\}$, and task_L in $\{2, 4\}$. For the MLP baseline, we vary $\text{task}_L \in \{2, 4, 6\}$ and $\text{task}_H \in \{16, 32, 64\}$. We vary task dropout in $\{0.0, 0.1, 0.3\}$. We report average results for two random seeds.

[4] The list of features (Detrano et al., 1986) are: real variables include: age of patient (age), resting systolic blood pressure (trestbps), cholesterol (chol), maximum heart rate achieved exercise (thalach) and exercise-induced ST-segment depression (oldpeak). Categorical variables include: type of pain a patient experienced if any (cp), slope of peak exercise ST segment (slope), classification of resting electrocardiogram (restecg), whether or not a patient had thallium defects (thal) as revealed by scintigraphy, number of major vessels appearing to contain calcium as revealed by cine uoroscopy (ca), gender (gender),whether or not a patient's fasting blood sugar was above 120 mg/dl (fbs), and whether or not a patient has exercise-induced angina (exang).

Qualitative Interpretation: Figure 9 shows the graphs learnt by our dGAP . We show a stronger link between Maximum Heart Rate (thalach) and Age in healthy class Figure 9(b), whereas a weaker link in heart disease patients Figure 9(c). We show this explicitly in the differential case Figure 9(d).

D.5. DNA MYC Binding Data

We use the MYC binding DNA data[5] about DNA binding specificity of a transcription factor MYC. We use the bound sequences as positive class and the unbound sequences as negative class. We split the data into 80/10/10. Figure 10 shows the learnt dependency structure for bound sequences. We observe a densely connected subgraph for positions 16 to 22. We show the corresponding sequence logo for the test data bound sequences. We observe the motif CACGTG which which forms a canonical DNA sequence(Staiger et al., 1989), also shown by (Tsang et al., 2020). We also observe that in general the adjacent sequential positions depend on each other.

Hyperparameters: We use $\text{struct}_H = 128$, $l_{\text{sparse}} = 0.001$, $\text{task}_H = 64$, $\text{init}_H = 32$, $\text{task}_L = 2$, nheads $= 4$ and a learning rate of 0.001. We use a batch size of 256.

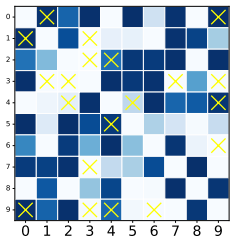
D.6. Real World Dataset: Stanford Sentiment Treebank-2 Dataset

We also evaluate dGAP on Stanford Sentiment Treebank (SST-2) binary sentiment classification task. We pad all sentences to a fixed length of 56. Instead of one hot feature representation, we use pretrained GloVe(Pennington et al., 2014) embeddings both for the structure learner as well as the task learner.

Hyperparameters: For this dataset, we use a shared structure learner MLP with two layers with hidden size

[5] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE47026

(a) (b) (c) (d)



(e) (f) (g) (h)

Figure 6. On Simulation: Heat maps of pairwise interaction strengths learnt by dGAP framework on for simulation datasets $p = 10$. Cross-marks indicate ground truth interactions. Red cross marks indicate Class A and Yellow cross marks indicate Class B. (a), Graph learnt by dGAP (b) Graph learnt by dGAP in comparison to the graph edges common to both classes. Here, we treat the common edges as ground truth indicated by red cross marks. (c) and (d) Graph learnt by M-dGAP . (c) shows the graph learnt for Class A and (d) shows the graph learnt for Class B. In the second row, (e) shows the graphs learnt by dGAP-NSS w/o sparsity , (f) shows the graphs learnt by dGAP-NSS and $_{sparse} = 0.005$, (g) and (h) show the graphs learnt by M-dGAP-NSS w/o sparsity : (h) graph corresponding to Class A and (h) graph corresponding to Class B.

$\{1024; 512\}$, $l_{sparse} = 0.01$, $task_H = 128$, $init_H = 100$, $task_L = 4$, $nheads = 4$ structure learner dropout of 0.3, task learner dropout 0.2 and a learning rate of 0.0001. We use a batch size of 128. For the M-dGAP variation, we use $task_H = 64$. For GCN and MLP variations, we use $task_H = 1024$. We show the learnt graph in Figure 11 and AUC in Table 2. Here, the color indicates belief in the presence or absence of an edge. Red indicates probability of an edge; blue indicates belief in the absence of an edge and yellow indicates maximum uncertainty. Local neighbors depend on each other(indicated by the thick orange diagonal). The sentences are variable length accounting for the diagonal Y shape in the learnt dependency graph.

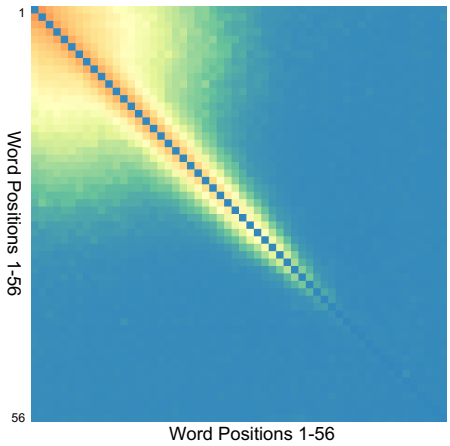


Figure 11. Graph learnt by dGAP on the sentiment classification SST-2 dataset. Red indicates belief in the presence of an edge; Blue indicates belief in the absence of an edge. In the learnt graph by dGAP , local neighbors depend on each other(indicated by the thick orange diagonal). The sentences are variable length accounting for the diagonal Y shape.

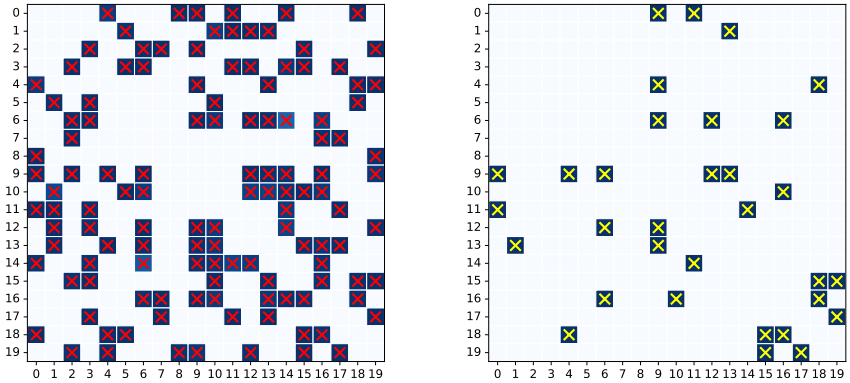
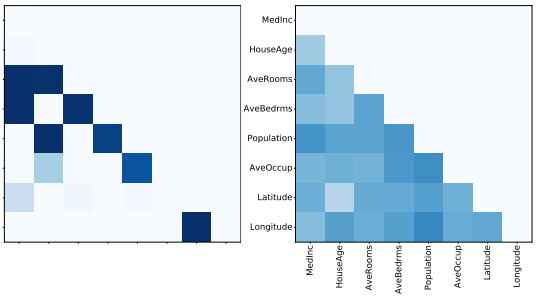
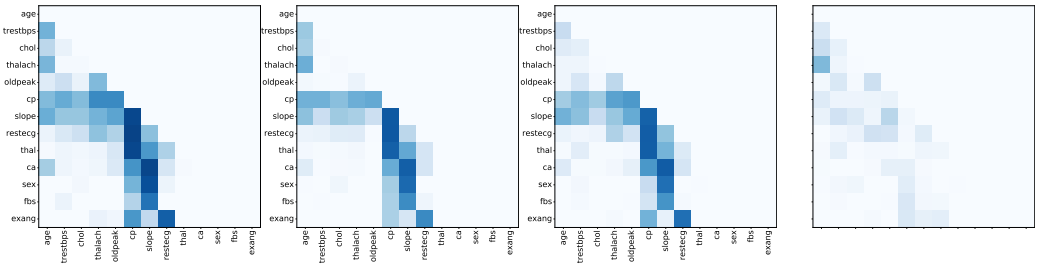


Figure 7. On Simulation: Learnt graphs for $p = 20$, (LEFT) Heat Map for Graph learnt for Class A by M-dGAP , Red Cross Marks show the ground truth graphs, and (RIGHT) Heat Map for Graphs learnt by Class B by M-dGAP , Yellow Cross Marks show the ground truth graphs.



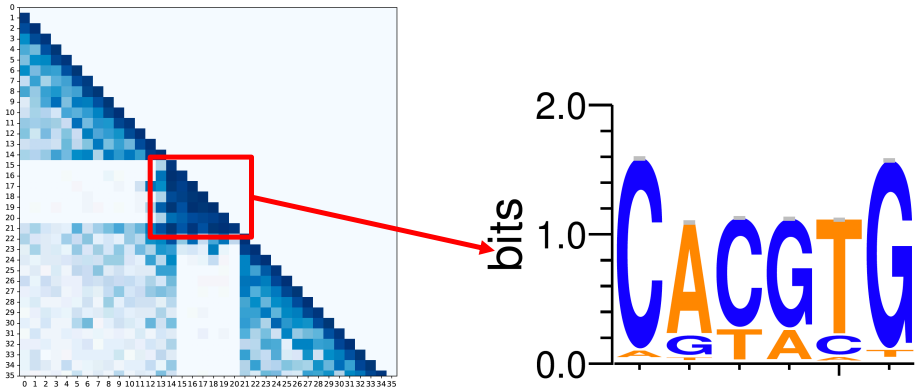
(a) (b)

Figure 8. Heatmaps of pairwise interaction strengths learnt by dGAP on the Cal-Housing data: (a) Graph learnt by dGAP . Our method is able to successfully learn a relationship between latitude and longitude; (b) Graph learnt by dGAP-NSS w/o sparsity . While some interactions are shared with (a), without structure loss we cannot recover a strong relationship between latitude and longitude.



(a) (b) (c) (d)

Figure 9. On heart disease data: Heatmaps of pairwise interaction strengths learnt by dGAP and variations. (a) The dependency graph learnt by dGAP . (b) and (c) Class specific graphs learnt by M-dGAP . We show a stronger link between Maximum Heart Rate (thalach) and Age in healthy case (b), whereas a weaker link in disease case (c). We show this differential graph explicitly in (d).



(a) (b)

Figure 10. Dependency Graph for MYC bound DNA sequences learnt by M-dGAP on MYC dataset. We observe the motif "CACGTG" in the densely connected positions 16 to 22 which forms a canonical DNA sequence(Staiger et al., 1989).