Probabilistic Neural Networks

Dissertation Defense

Evan Ott
November 28, 2022

The University of Texas at Austin
Department of Statistics and Data Sciences
College of Natural Sciences

Committee Members
Sinead Williamson, Supervisor
José Miguel Hernández-Lobato
James Scott
Mingyaun Zhou

Overview

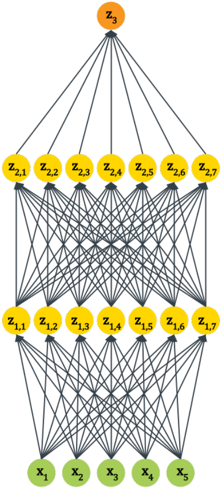
Overview

Neural networks are highly flexible models for complicated data relationships

Typically, parameters are estimated by MLE (or minimizing a loss)

Typically used in deterministic settings

Inspired by Bayesian methodology, this work explores ways of incorporating uncertainty in neural network-based models



Overview

Explore methods of approximate Bayesian inference in neural networks (Ott and Williamson, 2022b)

Approximate a Bayesian-inspired posterior in normalizing flow models (Ott and Williamson, 2022a)

Construct a generative graph neural network model inspired by results in Bayesian nonparametrics

Spike-and-Slab Probabilistic Backpropagation

This work will appear as a paper and poster, "Spike-and-Slab Probabilistic Backpropagation: When Smarter Approximations Make No Difference," (Ott and Williamson, 2022b), at the I Can't Believe It's Not Better workshop at NeurIPS 2022 (this Saturday)

Bayesian Neural Networks

Consider a feed-forward neural network with weights and biases $\mathcal{W} = [\{W_\ell\}_{\ell=1}^L, \{\mathbf{b}_\ell\}_{\ell=1}^L]$:

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{x} \\ \mathbf{z}_\ell &= \sigma_l \left(W_\ell \mathbf{z}_{\ell-1} + \mathbf{b}_\ell \right), \end{aligned}$$

Consider a likelihood, e.g., $p(y|\mathcal{W}) = N(y|\mathbf{z}_L, \gamma^{-1})$

Construct a prior $p(\mathcal{W})$, typically Gaussian

Posterior $p(\mathcal{W}|Y)$ is intractable

Bayesian Neural Networks

MCMC methods are slow (Neal, 1995)

Laplace approximations are sensitive (MacKay, 1992)

Variational inference approaches require sampling (Graves, 2011; Blundell et al., 2015)

Other work: $p(\mathcal{W}|Y) \to q(\mathcal{W})$ and "messages" $q(\mathbf{z}_\ell)$ (e.g., Hernández-Lobato and Adams, 2015; Wu et al., 2018)

Probabilistic Backpropagation

In our work, we follow probabilistic backpropagation (PBP, Hernández-Lobato and Adams, 2015)

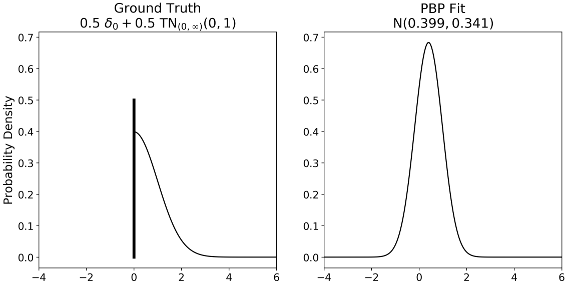
Assume a mean-field Gaussian approximate posterior for $q(\mathcal{W})$

Gaussian messages $q(\mathbf{z}_\ell)$ in linear and ReLU layers, using moment-matching

Probabilistic Backpropagation

PBP models messages with Gaussians

Consider $z_1 = \mathrm{ReLU}(W_1\mathbf{x} + b_1)$, the "true" distribution for $(z_1)_i$ induced is $(1-\rho)\delta_0 + \rho\mathrm{TN}_{(0,\infty)}(m, v)$



Can we do better?

Spike-and-Slab Approximation

We propose a spike-and-slab approximation for message distributions $(1-\rho)\delta_0 + \rho N(m, v)$

We derive optimal parameters by minimizing $KL(p\|q)$, giving:

$$\rho = \mathbb{P}_{X\sim p}[X \neq 0], \qquad m = \frac{1}{\rho}\mathbb{E}_{X\sim p}[X],$$

$$v = \frac{1}{\rho}\left(\mathbb{V}_{X\sim p}[X] - \rho(1-\rho)m^2\right).$$

Constructed drop-in replacements for PBP equations for linear and ReLU layers

Visual Comparison

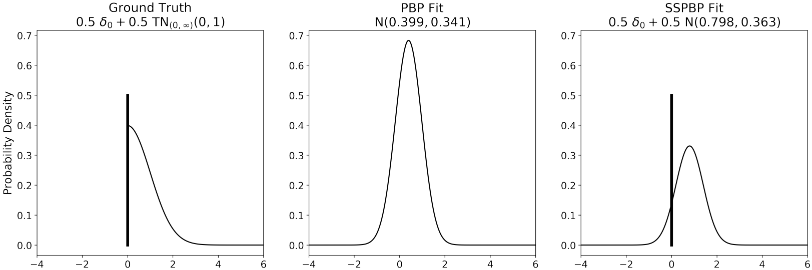


Figure: Comparison of Gaussian and spike-and-slab approximations of the $0.5\delta_0 + 0.5\mathrm{TN}_{0,\infty}(0,1)$.

Simulation Study

Consider samples of $Z = \mathrm{ReLU}(XW) = \max(XW, 0)$:

p_X	p_W	% Saturated	PBP	SSPBP
N$(0, 1)$	N$(0, 1)$	49.86%	0.066	**0.020**
N$(1, 1)$	N$(3, 1)$	16.24%	0.031	**0.015**
N$(1, 1)$	N$(-3, 1)$	84.44%	0.21	**0.0038**
N$(3, 1)$	N$(3, 1)$	0.22%	**0.0043**	0.0051
N$(3, 1)$	N$(-3, 1)$	99.72%	0.017	**0.00024**

Table: Simulation study of how well SSPBP approximates the true distribution, reporting the MMD between a ground truth sample and approximations obtained using either PBP or SSPBP.

Method Comparison

Dataset	2 Layers 10 Nodes	
	PBP	SSPBP
Boston Housing	3.097±0.147	**2.997±0.165**
Combined Cycle Power Plant	**4.088±0.067**	4.096±0.066
Concrete Compression Strength	6.031±0.161	**5.921±0.158**
Energy Efficiency	**1.477±0.043**	1.660±0.112
Kin8nm	0.111±0.004	**0.109±0.002**
Naval Propulsion	**0.006±0.000**	**0.006±0.000**
Wine Quality Red	0.653±0.012	**0.652±0.008**
Yacht Hydrodynamics	**1.064±0.072**	1.131±0.063

Table: Mean and standard error of average test set RMSE of PBP and SSPBP, on eight datasets.

Method Equality

Intuition: the bias term forces the "true" distribution's spike probability to be 0

Proved: methods produce *identical* approximations following a ReLU and linear layer

Considered a bias-free version of PBP and SSPBP to compare approximations that are different

Bias-Free Method Comparison

	2 Layers 10 Nodes	
Dataset	**PBP**	**SSPBP**
Boston Housing	**3.809±0.295**	3.865±0.322
Combined Cycle Power Plant	4.190±0.017	**4.188±0.023**
Concrete Compression Strength	6.823±0.214	**6.668±0.237**
Energy Efficiency	1.699±0.041	**1.617±0.019**
Kin8nm	0.126±0.001	**0.125±0.001**†
Naval Propulsion	**0.005±0.000**	0.006±0.000
Wine Quality Red	0.635±0.011	**0.633±0.014**
Yacht Hydrodynamics	**3.898±0.245**	4.276±0.390

Table: Mean and standard error of average test set RMSE for the bias-free versions of PBP and SSPBP, on eight datasets.
† Due to numerical issues, the trials for this model were repeated with a different random seed.

I Can't Believe It's Not Better

Turns out, Gaussians are effective

Suspect spike-and-slab approximate posterior for $q(\mathcal{W})$ may not differ strongly from PBP with wide layers

Could explore alternative approximations for messages, e.g., collapsing near-zero values to the spike

Nonparametric Posterior Normalizing Flows

Some work presented here is currently under review as part of (Ott and Williamson, 2022a) for AISTATS 2023.

Normalizing Flows

Normalizing flows (Tabak and Turner, 2013) apply latent h and a change-of-variables via invertible and differentiable "flow" g_ϕ:

$$U \sim h \qquad\qquad x = g_\phi(u)$$

$$p_\phi(x) = h(g_\phi^{-1}(x)) \left| \det \frac{dg_\phi^{-1}(X)}{dX} \right|_{X=x}$$

Can model complex high-dimensional distributions

Normalizing Flows

Trained by maximum likelihood, ignoring model uncertainty

As with Bayesian neural networks, intractable Bayesian posterior

Prior on flow parameters is not intuitive (true for BNNs as well)

Can we incorporate model uncertainty and a more intuitive prior?

Nonparametric Learning

Consider $X \sim F_0$ and parameter of interest θ_0

Thought experiment: if we had access to $x_{1:\infty}$ or F_0, how to estimate θ_0?

Nonparametric Learning

Consider $X \sim F_0$ and parameter of interest θ_0

Thought experiment: if we had access to $x_{1:\infty}$ or F_0, how to estimate θ_0?

MLE totally appropriate

$$\theta_0(F_0) = \arg\min_{\theta} \int \ell(x, \theta) dF_0(x)$$

Problem: only have $x_{1:n}$, represent uncertainty about $x_{(n+1):\infty}$ or F_0

Nonparametric Learning

Nonparametric learning (Lyddon et al., 2018; Fong et al., 2019) is an Bayesian-inspired approach to inference

Key idea: express our uncertainty about F_0 and construct $F_0|x_{1:n}$, then

$$\tilde{\pi}(\theta|x_{1:n}) = \int \pi(\theta|F_0) d\pi(F_0|x_{1:n})$$

Can sample this "nonparametric posterior" via MC

Nonparametric Learning

We'll use $F_0 \sim \mathrm{DP}(\alpha, F_\pi)$, following Fong et al. (2019), with

$$F_0 | x_{1:n} \sim \mathrm{DP}\left(\alpha + n, \frac{1}{\alpha + n}\left(\sum_i \delta_{x_i} + \alpha F_\pi\right)\right)$$

Can obtain bootstrap samples of $\tilde{\pi}(\theta | x_{1:n})$

Nonparametric Learning

Algorithm NPL posterior sampling (Fong et al., 2019)

Require: Observations $x_{1:n}$, number of samples B, base measure F_π, concentration parameter α

for $b = 1, \ldots, B$ **do**

$F^{(b)} = \sum_{i=1}^{\infty} w_i \delta_{\psi_i} \sim \mathrm{DP}\left(\alpha + n, \frac{\sum_i \delta_{x_i} + \alpha F_\pi}{\alpha + n}\right)$

$\theta^{(b)} = \arg\min_\theta \sum_{i=1}^{\infty} w_i \ell(\psi_i, \theta)$

end for

return $\{\theta^{(b)}\}_{b=1}^{B}$

Our Method

Naïve NPL approach: learn B independent NFs (expensive)

Idea: reparameterize NFs with latent θ and shared ϕ

$$U \sim \mathbf{h}_\theta = \mathbf{N}(\mu, \mathbf{I}) \qquad\qquad x = g_\phi(u)$$

Learn ϕ globally, θ is analytically tractable to optimize

Simulations for GMM

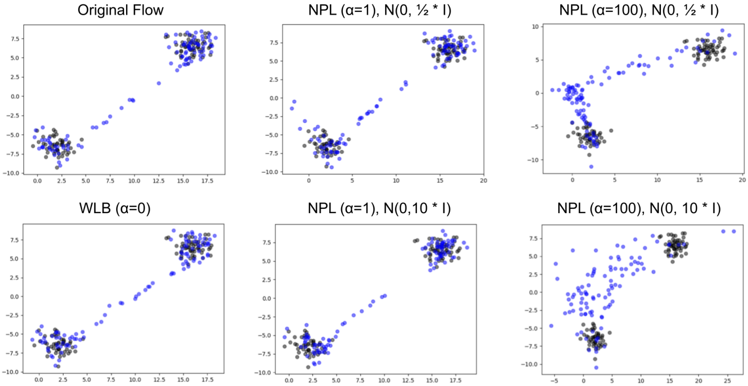


Figure: Samples (blue) from MAF model trained on $N = 100$ data points (black), in the original MAF model, and in the posterior normalizing flow version with $\alpha \in \{0, 1, 100\}$ and two spherical Gaussian priors centered at $(0, 0)^\top$.

Empirical Results

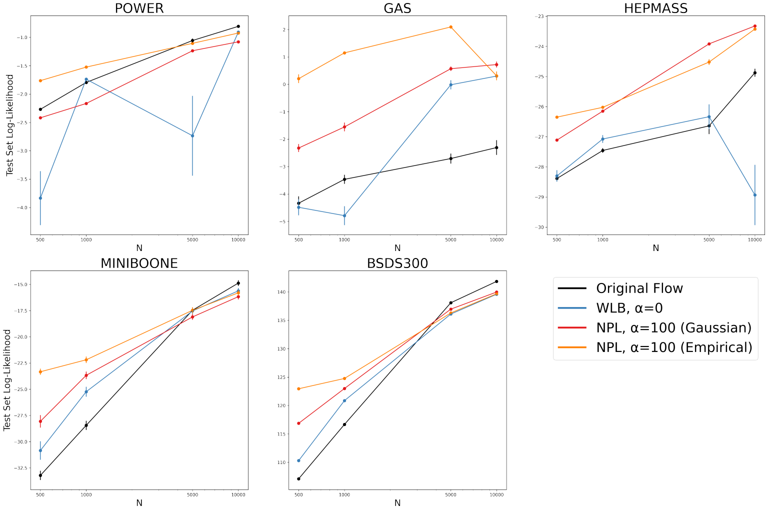


Figure: Results on datasets, with covariate dimension increasing, comparing priors.

Comparison with Naïve NPL Approach

Method	Average Test LL
Maximum Likelihood, $\mu = 0$	-28.441 ± 0.405
Naïve WLB, $\mu = 0$	-31.975 ± 0.453
Ours, WLB, $\theta = \mu$	-25.234 ± 0.438
Ours, NPL $\alpha = 100$ (Gaussian), $\theta = \mu$	-23.671 ± 0.327
Ours, NPL $\alpha = 100$ (Empirical), $\theta = \mu$	-22.188 ± 0.278

Table: Comparison of standard NPL approach to our version on the MiniBooNE dataset ($N = 1000$).

Discussion and Future Work

Choice of flow architecture is important – explored as joint work in Ott and Williamson (2022a)

Explored simple and best-case priors, can explore historical data or transfer learning approaches

Edge-Based Generative Graph Neural Networks

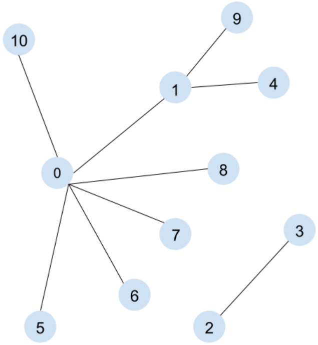
This represents my contributions to a larger collaboration with Curtis Carter, Elahe Ghalebi, and Sinead Williamson.

Graph Neural Networks

Graph neural networks (GNNs, Scarselli et al., 2008) create "node embeddings" h_v for graph $G = (V, E)$ by message passing

$$m_v = \sum_{(u,v)\in E} f_{\text{message}}(x_v, x_u, x_{(u,v)}, h_u)$$

$$h'_v = f_{\text{update}}(h_v, m_v)$$



Generative GNNs

GNNs great for node or graph classification

Generative GNN models construct a distribution on graphs (e.g., DeepGMG, Li et al., 2018)

Example: DeepGMG constructs G node-by-node

Add new node based on graph embedding; add edges based on node embeddings

Generally, not focused on graph properties

Generative GNNs

Sparsity is an important property of graphs:

$$\mathrm{sparsity}(G) = 1 - \mathrm{density}(G) = 1 - \frac{2|E|}{|V|(|V| - 1)}$$

Real-world graphs are often sparse, e.g., social networks

Many generative GNN models are graphon-like, which produce dense graphs (Orbanz and Roy, 2014)

Edge-Based Graph Models

Certain edge-based methods can produce sparse graphs (Crane and Dempsey, 2016; Cai et al., 2016)

Consider "binary Hollywood process" (BHP, Crane and Dempsey, 2016), with new edge $e_{n+1} = (s, r)$ for graph $G_n = (V_n, E_n)$:

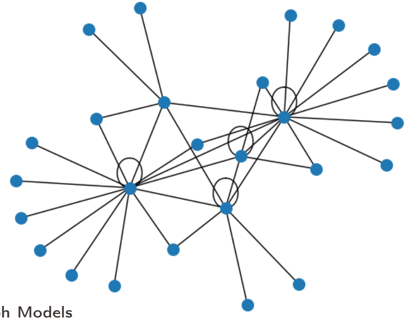
$$p(S = v) \propto \begin{cases} \text{degree}(v) - \sigma & v \in V_n, \\ \alpha + \sigma |V_n| & v = |V_n| + 1, \end{cases}$$

$$p(R = v | S = s) \propto \begin{cases} \text{degree}(v) - \sigma & v \in V_n, \\ 1 - \sigma & v = s = |V_n| + 1, \\ \alpha + \sigma |V_n \cup \{s\}| & v = |V_n \cup \{s\}| + 1 \end{cases}$$

Edge-Based Graph Models

$$p(S = v) \propto \begin{cases} \text{degree}(v) - \sigma & v \in V_n, \\ \alpha + \sigma |V_n| & v = |V_n| + 1, \end{cases}$$

$$p(R = v | S = s) \propto \begin{cases} \text{degree}(v) - \sigma & v \in V_n, \\ 1 - \sigma & v = s = |V_n| + 1, \\ \alpha + \sigma |V_n \cup \{s\}| & v = |V_n \cup \{s\}| + 1 \end{cases}$$



Edge-Based Generative GNN

Select new edge $e_{n+1} = (s, r)$ for $G_n = (V_n, E_n)$

$$p(S = v) \propto \begin{cases} f_{\text{sender}}(h_v; \theta) & v \in V_n, \\ f_{\text{new sender}}(h_G; \theta) & v = |V_n| + 1, \end{cases}$$

$$p(R = v | S = s) \propto \begin{cases} f_{\text{recipient}}(h_v, h_s; \theta) & v \in V_n \cup \{s\}, \\ f_{\text{new recipient}}(h_G, h_s; \theta) & v = |V_n \cup \{s\}| + 1 \end{cases}$$

Add edge to graph and update embeddings, optionally weighting messages by time

Comparison on BHP Graphs

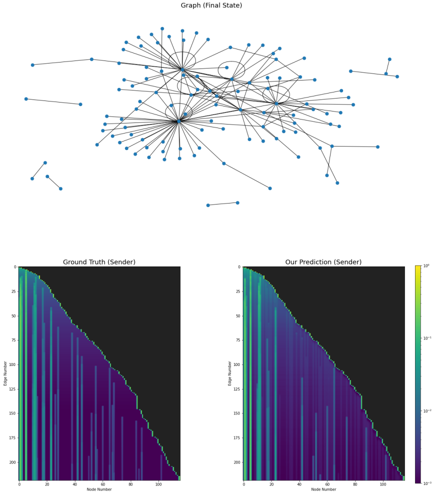


Figure: *Top*, a BHP graph from the test set. *Left*, ground truth sender probabilities under the BHP model. *Right*, Predicted sender probabilities within our model.

Sparsity for BHP Graphs

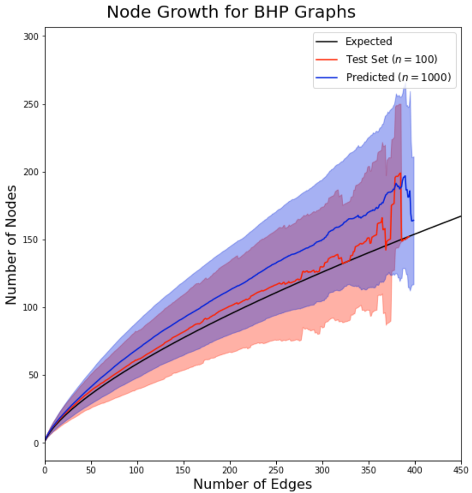


Figure: Expected and empirical node growth for binary Hollywood process graph sequences and generated graph sequences.

Visual Comparison on Synthetic Graphs

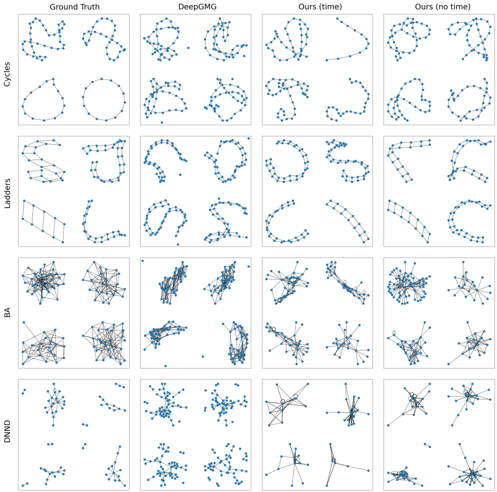


Figure: Graphs generated from DeepGMG and our model for synthetic graph datasets.

Quantitative Comparison on Synthetic Graphs

Dataset + Metric	Ours (time)	Ours (no time)	DeepGMG
BA	$n = 4$	$n = 6$	$n = 4$
$\mathrm{MMD}_{\mathrm{degree}}$	0.35268 ± 0.00943	0.38978 ± 0.01611	$\mathbf{0.22016 \pm 0.01586}$
$\mathrm{MMD}_{\mathrm{cluster}}$	$\mathbf{0.10822 \pm 0.00065}$	0.11020 ± 0.00117	0.14204 ± 0.00011
$\mathrm{MMD}_{\mathrm{spectral}}$	$\mathbf{0.19653 \pm 0.00923}$	0.23202 ± 0.01701	0.21706 ± 0.00332
$\mathrm{KS}_{\mathrm{density}}$	$\mathbf{0.54475 \pm 0.02572}$	0.67117 ± 0.02720	0.89250 ± 0.02161
Cycles	$n = 4$	$n = 6$	$n = 5$
$\mathrm{MMD}_{\mathrm{degree}}$	0.08389 ± 0.00018	$\mathbf{0.08350 \pm 0.00025}$	0.12930 ± 0.01692
$\mathrm{MMD}_{\mathrm{cluster}}$	$\mathbf{0.00000 \pm 0.00000}$	0.00531 ± 0.00232	0.18182 ± 0.06401
$\mathrm{MMD}_{\mathrm{spectral}}$	0.14965 ± 0.00069	$\mathbf{0.14872 \pm 0.00146}$	0.29722 ± 0.00620
$\mathrm{KS}_{\mathrm{density}}$	$\mathbf{0.09850 \pm 0.00043}$	0.09983 ± 0.00146	0.81000 ± 0.04195
DNND	$n = 4$	$n = 5$	$n = 4$
$\mathrm{MMD}_{\mathrm{degree}}$	0.37253 ± 0.00360	0.33253 ± 0.01476	$\mathbf{0.09297 \pm 0.00482}$
$\mathrm{MMD}_{\mathrm{cluster}}$	0.39874 ± 0.00749	$\mathbf{0.39486 \pm 0.00724}$	0.50634 ± 0.01349
$\mathrm{MMD}_{\mathrm{spectral}}$	0.25171 ± 0.00781	0.22512 ± 0.01368	$\mathbf{0.17333 \pm 0.00211}$
$\mathrm{KS}_{\mathrm{density}}$	0.50975 ± 0.04510	$\mathbf{0.41720 \pm 0.03059}$	0.99250 ± 0.00217
Ladders	$n = 4$	$n = 6$	$n = 5$
$\mathrm{MMD}_{\mathrm{degree}}$	0.00397 ± 0.00043	$\mathbf{0.00375 \pm 0.00013}$	0.10733 ± 0.00009
$\mathrm{MMD}_{\mathrm{cluster}}$	$\mathbf{0.00000 \pm 0.00000}$	0.00000 ± 0.00000	0.00143 ± 0.00128
$\mathrm{MMD}_{\mathrm{spectral}}$	0.05223 ± 0.00027	$\mathbf{0.05133 \pm 0.00030}$	0.50509 ± 0.00086
$\mathrm{KS}_{\mathrm{density}}$	0.08125 ± 0.00134	$\mathbf{0.08033 \pm 0.00122}$	0.96000 ± 0.00000

Table: Synthetic dataset results, showing the mean and standard error of each metric on four trials unless otherwise noted.

Quantitative Comparison on Real Graphs

Dataset + Metric	Ours (time)	Ours (no time)	DeepGMG
DBLP	$n = 4$	$n = 6$	$n = 4$
$\mathrm{MMD}_{\mathrm{degree}}$	0.38128 ± 0.00871	0.41075 ± 0.00616	$\mathbf{0.06716 \pm 0.00766}$
$\mathrm{MMD}_{\mathrm{cluster}}$	0.23977 ± 0.00006	$\mathbf{0.23971 \pm 0.00001}$	0.26312 ± 0.01346
$\mathrm{MMD}_{\mathrm{spectral}}$	0.43224 ± 0.00550	0.43172 ± 0.00813	$\mathbf{0.15566 \pm 0.00464}$
$\mathrm{KS}_{\mathrm{density}}$	0.96535 ± 0.01185	0.96659 ± 0.00614	$\mathbf{0.47656 \pm 0.07952}$
Highschool	$n = 4$	$n = 4$	$n = 3$
$\mathrm{MMD}_{\mathrm{degree}}$	0.31491 ± 0.00859	0.39204 ± 0.01191	$\mathbf{0.19257 \pm 0.01318}$
$\mathrm{MMD}_{\mathrm{cluster}}$	$\mathbf{0.21266 \pm 0.00165}$	0.21684 ± 0.00106	0.21780 ± 0.00306
$\mathrm{MMD}_{\mathrm{spectral}}$	$\mathbf{0.25285 \pm 0.01048}$	0.34389 ± 0.01671	0.29707 ± 0.02010
$\mathrm{KS}_{\mathrm{density}}$	0.84394 ± 0.01469	0.76664 ± 0.06818	$\mathbf{0.23444 \pm 0.04132}$
MIT	$n = 0$	$n = 2$	$n = 2$
$\mathrm{MMD}_{\mathrm{degree}}$	–	0.64142 ± 0.01921	$\mathbf{0.09543 \pm 0.00096}$
$\mathrm{MMD}_{\mathrm{cluster}}$	–	0.80641 ± 0.00130	$\mathbf{0.63332 \pm 0.00997}$
$\mathrm{MMD}_{\mathrm{spectral}}$	–	0.47489 ± 0.04091	$\mathbf{0.09277 \pm 0.00473}$
$\mathrm{KS}_{\mathrm{density}}$	–	$\mathbf{0.82450 \pm 0.08733}$	0.86000 ± 0.02121
Tumblr	$n = 4$	$n = 5$	$n = 2$
$\mathrm{MMD}_{\mathrm{degree}}$	0.21777 ± 0.03324	0.33572 ± 0.01402	$\mathbf{0.06708 \pm 0.00416}$
$\mathrm{MMD}_{\mathrm{cluster}}$	0.54088 ± 0.01591	0.57516 ± 0.00149	$\mathbf{0.35256 \pm 0.01699}$
$\mathrm{MMD}_{\mathrm{spectral}}$	0.18212 ± 0.02233	0.27061 ± 0.01587	$\mathbf{0.14842 \pm 0.00133}$
$\mathrm{KS}_{\mathrm{density}}$	$\mathbf{0.74173 \pm 0.05625}$	0.86281 ± 0.02615	0.98324 ± 0.01185

Table: Real-world dataset results, showing the mean and standard error of each metric on four trials unless otherwise noted.

Discussion and Future Work

Works well on smaller synthetic graphs

Suspect that real-world graph performance was hindered by over-smoothing of node embeddings

Reducing number of messages (e.g., window)

Concluding Remarks

Explored incorporating stochasticity in neural networks

Proposed spike-and-slab approximation for PBP

Applied novel inference method to normalizing flows

Created generative GNN capable of producing sparse graphs

Thanks

Questions?

References I

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Diana Cai, Trevor Campbell, and Tamara Broderick. Edge-exchangeable graphs and sparsity. *Advances in Neural Information Processing Systems*, 29, 2016.

Harry Crane and Walter Dempsey. Edge exchangeable models for network data. *arXiv preprint arXiv:1603.04571*, 2016.

Edwin Fong, Simon Lyddon, and Chris Holmes. Scalable nonparametric sampling from multimodal posteriors with the posterior bootstrap. In *International Conference on Machine Learning*, pages 1952–1962. PMLR, 2019.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Simon Lyddon, Stephen Walker, and Chris C Holmes. Nonparametric learning from Bayesian models with randomized objective functions. In *Advances in Neural Information Processing Systems*, 2018.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

Peter Orbanz and Daniel M Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2014.

Evan Ott and Sinead Williamson. Nonparametric Posterior Normalizing Flows. submitted, 2022a.

Evan Ott and Sinead Williamson. Spike-and-Slab Probabilistic Backpropagation: When Smarter Approximations Make No Difference. In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification*, 2022b. URL https://openreview.net/forum?id=iYAdBHSA_Pt.

References II

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

Coupling NF Layers

Transformations can include non-invertible functions f_ϕ, e.g.,

$$\begin{aligned} x_{1:m} &= u_{1:m} \\ x_{(m+1):2m} &= u_{(m+1):2m} + f_\phi(u_{1:m}) \end{aligned}$$

Posterior Bootstrap Sampling

Algorithm Posterior bootstrap sampling (Fong et al., 2019)

Require: Observations $y_1, \ldots, y_n$, base measure F_π, number of samples B, concentration parameter α

for $b = 1, \ldots, B$ **do**

Sample m pseudo-observations $y^*_{1:m} \sim F_\pi$

Sample weights $W := (w_{1:n}, w^*_{1:m}) \sim \mathrm{Dir}\left(1, \cdots, 1, \frac{\alpha}{m}, \cdots, \frac{\alpha}{m}\right)$

$F^{(b)} = \sum_{i=1}^{n} w_i \delta_{y_i} + \sum_{i=1}^{m} w^*_i \delta_{y^*_i}$

$\theta^{(b)} = \arg\min_\theta \sum_{i=1}^{n} w_i \ell(y_i, \theta) + \sum_{i=1}^{m} w^*_i \ell(y^*_i, \theta)$

end for

return $\{\theta^{(b)}\}_{b=1}^{B}$

Posterior Bootstrap with Shared Parameters

Algorithm Posterior Bootstrap with Shared Parameters

Require: Observations $y_1, \ldots, y_n$, base measure F_π, initial shared parameter value ϕ_0, number of samples B, concentration parameter α, learning rate τ

$\phi \leftarrow \phi_0$

while not converged **do**

Sample m pseudo-observations $y^*_{1:m} \sim F_\pi$

Sample weights $W := (w_{1:n}, w^*_{1:m}) \sim \text{Dir}\left(1, \cdots, 1, \frac{\alpha}{m}, \cdots, \frac{\alpha}{m}\right)$

$\tilde{F} = \sum_{i=1}^n w_i \delta_{y_i} + \sum_{i=1}^m w_i^* \delta_{y_i^*}$

$\theta = \arg\min_\theta \int \ell(y, \phi, \theta) d\tilde{F}(y)$

$\phi \leftarrow \phi + \tau \nabla \left(\sum_{i=1}^n w_i \ell(y_i, \phi, \theta) + \sum_{i=1}^m w_i^* \ell(y_i^*, \phi, \theta)\right)$

end while

$\hat{\phi} \leftarrow \phi$