Inference Networks for Graphical Models

Brooks Paige Frank Wood

Summary

We introduce a new approach to **amortizing inference** in **directed graphical models**. Inference in graphical models entails characterizing the joint distribution of latent variables conditioned on some observed data.

We learn a structured **neural network** to represent an **inverse factorization** of the graphical model. This conditional density estimator takes particular values of observed random variables as input, and returns an approximation to the posterior distribution.

The recognition model can be **learned offline**, independent of any particular dataset, before inference is performed. The learned representations **compile away the runtime costs of inference**, critical for applications that require fast inference when encountering new data.

Approach

Generative models with latent variables **x** and observed variables **y** define a distribution $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$:

$$p(\mathbf{x},\mathbf{y}) \triangleq \prod_{i=1}^{N} p\left(x_i|\mathrm{PA}(x_i)\right) \prod_{j=1}^{M} p\left(y_j|\mathrm{PA}(y_j)\right)$$

We are interested in characterizing the posterior distribution $\pi(\mathbf{x}) \equiv p(\mathbf{x}|\mathbf{y})$.

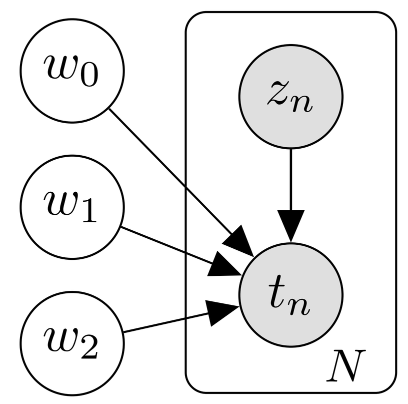
To do this we construct an **inverse factorization** of the graphical model $\tilde{p}(\mathbf{x},\mathbf{y}) = \tilde{p}(\mathbf{y})\tilde{p}(\mathbf{x}|\mathbf{y})$. The inverse model has the **same joint distribution** as the generative model, but a different factorization [5].

Unfortunately, the conditional densities $\tilde{p}(\mathbf{x}_i|\widetilde{\mathrm{PA}}(\mathbf{x}_i))$ in the inverse model have forms we do not know how to normalize or sample from in general.

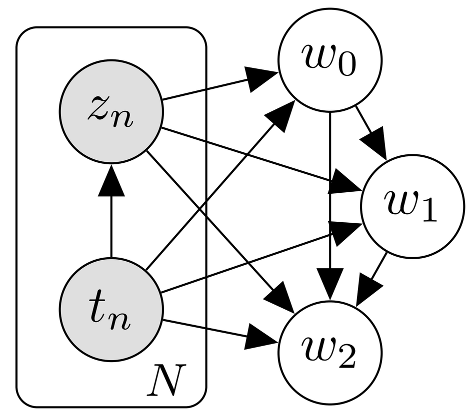
Our approach is to employ neural density estimators to **learn tractable approximations** to these conditional densities in the inverse model. These can be learned offline, in the sense that no real data is required.

Examples of inverse models

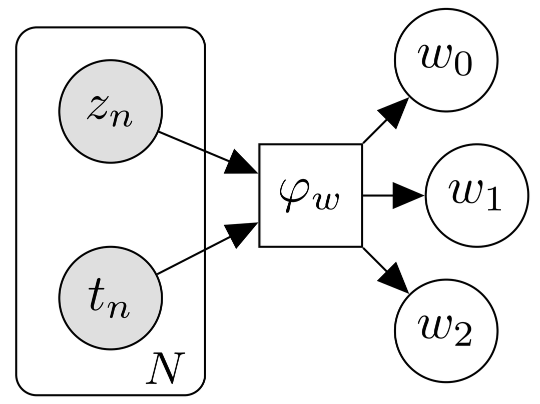
A generative model generates the data; the inverse model generates the latent parameters. This inverse model is fully connected due to "explaining away".



A regression model (generates data)

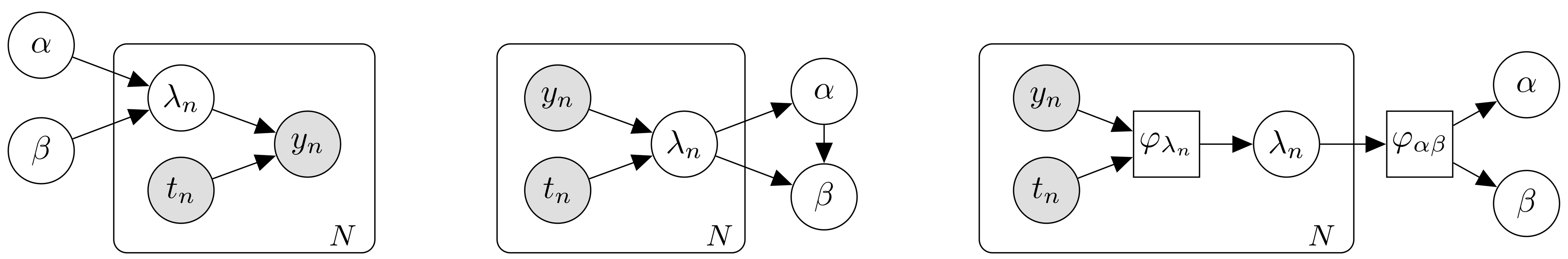


An inverse model (generates latents)



Learn a mapping from data to latents

In multilevel models (e.g. hierarchical Bayesian models) we can take leverage any factorization in the inverse model to run an SMC algorithm for graphical models [4], sweeping through successively larger sets of latent variables.



Learning a family of importance sampling proposals

Importance sampling and SMC approximate the posterior as weighted samples:

$$\hat{p}(\mathbf{x}|\mathbf{y}) = \sum_{k=1}^{K} W_k \delta_{\mathbf{x}_k}(\mathbf{x}) \qquad w(\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{x}|\lambda)} \qquad W_k = \frac{w(\mathbf{x}_k)}{\sum_{j=1}^{K} w(\mathbf{x}_j)}$$

Performance depends crucially on the quality of the proposal $q(\mathbf{x}|\lambda)$. A standard approach for learning these proposals [1,2] is to fix a parametric family, and then minimize the reverse KL divergence:

$$D_{KL}(\pi || q_\lambda) = \int \pi(\mathbf{x}) \log\left[\frac{\pi(\mathbf{x})}{q(\mathbf{x}|\lambda)}\right] \mathrm{d}\mathbf{x}$$

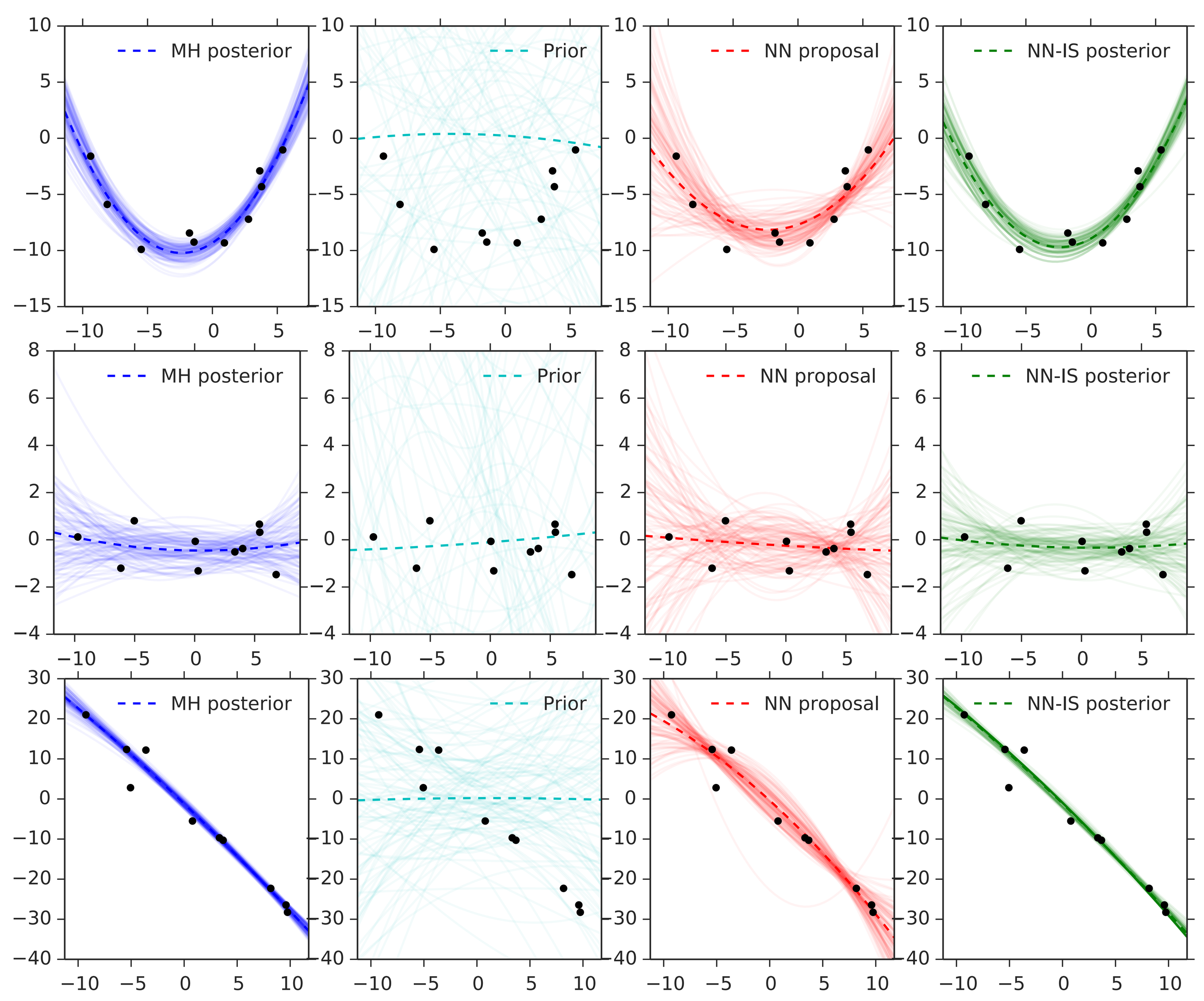
This is typically performed in the context of a single dataset, minimizing with respect to λ. In the amortized inference case we instead want to **average over all possible datasets**. To do so we introduce a function $\lambda = \varphi(\eta, \mathbf{y})$ which maps from some (new) dataset to a parameter setting, and learn hyperparameters η.

This suggests the objective function shown at right. Note the expectation is with respect to the **tractable** joint distribution, so we can train using purely synthetic data, via SGD on $\nabla_\eta \mathcal{J}(\eta) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[-\nabla_\eta \log q(\mathbf{x}|\varphi(\eta,\mathbf{y}))\right]$.

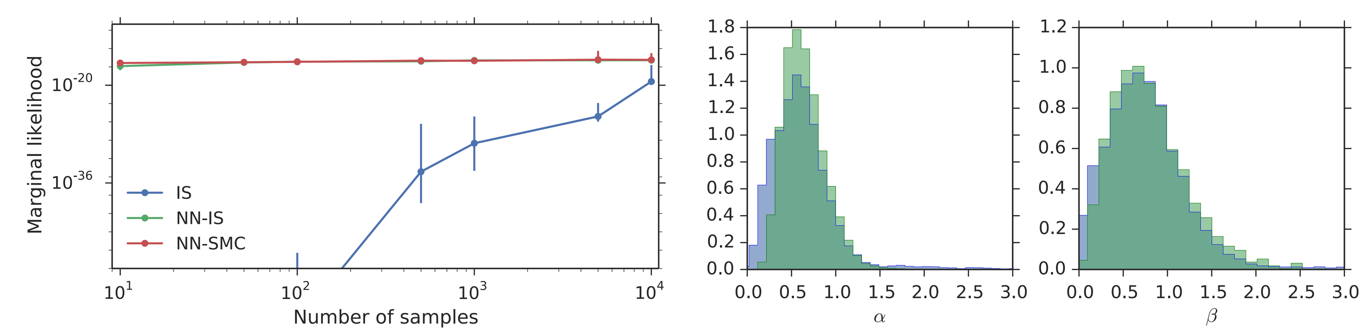
$$\begin{aligned}\mathcal{J}(\eta) &= \int D_{KL}(\pi || q_\lambda) p(\mathbf{y}) \mathrm{d}\mathbf{y} \\ &= \int p(\mathbf{y}) \int p(\mathbf{x}|\mathbf{y}) \log\left[\frac{p(\mathbf{x}|\mathbf{y})}{q(\mathbf{x}|\varphi(\eta,\mathbf{y}))}\right] \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\ &= \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[-\log q(\mathbf{x}|\varphi(\eta,\mathbf{y}))\right] + const.\end{aligned}$$

Representative results

Visualization in non-conjugate robust regression



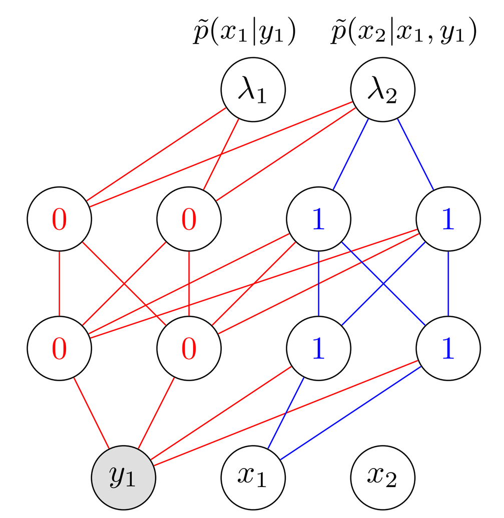
Fast convergence in a hierarchical Poisson model



Conditional density estimation

We use a mixture of Gaussians as our parametric family, and define $\varphi(\eta, \mathbf{y})$ to be a multilayer neural network.

For high dimensional outputs we use a masking scheme based on MADE [3].



References

[1] Olivier Cappé, Randal Douc, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 2008.

[2] Julien Cornebise, Éric Moulines, and Jimmy Olsson. Adaptive methods for sequential importance sampling with application to state space models. *Statistics and Computing*, 2008

[3] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. *ICML*, 2015.

[4] Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential Monte Carlo for Graphical Models. *NIPS*, 2014.

[5] Andreas Stuhlmüller, Jessica Taylor, and Noah Goodman. Learning Stochastic Inverses. *NIPS*, 2013.