On Priors for Bayesian Neural Networks

Eric Nalisnick

PhD Defense
University of California, Irvine
# First-Author Publications and Preprints

## Working Papers / Preprints


## Publications During PHD


## Pre-UCI

# First-Author Publications and Preprints

## Working Papers / Preprints

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Year(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. Nalisnick and P. Smyth.</td>
<td>Averaging and Combining Variational Models with Stein Particle Descent.</td>
<td>2018</td>
</tr>
<tr>
<td>E. Nalisnick, A. Anandkumar, and P. Smyth.</td>
<td>A Scale Mixture Perspective of Multiplicative Noise in Neural Networks.</td>
<td>2015 / 2018</td>
</tr>
</tbody>
</table>

## Publications During PhD

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. Nalisnick and P. Smyth.</td>
<td>Learning Priors for Invariance.</td>
<td>AISTATS 2018</td>
</tr>
<tr>
<td>E. Nalisnick and P. Smyth.</td>
<td>Learning Approximately Objective Priors.</td>
<td>UAI 2017</td>
</tr>
</tbody>
</table>

## Pre-UCI

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. Nalisnick and H. Baird.</td>
<td>Extracting Sentiment Networks from Shakespeare’s Plays.</td>
<td>ICDAR 2013</td>
</tr>
</tbody>
</table>
First-Author Publications and Preprints

WORKING PAPERS / PREPRINTS


PUBLICATIONS DURING PHD


PRE-UCI

# Outline

1. Motivation
2. Bayesian Neural Networks and Their Priors
3. Approximating Objective Priors
4. Nonparametric Density Networks
5. Future Directions and Conclusions
BACKGROUND

Motivation for Bayesian Deep Learning
Deep Learning Results

Computer Vision: Results on ImageNet object classification dataset.
Deep Learning Results

Computer Vision: Results on ImageNet object classification dataset.

Reinforcement Learning: Results in playing Go.

DEEP NEURAL NETWORKS
Neural Networks

\[ \mathbb{E}[y_i | x_i] \]
Neural Networks

\[ h_i^l = f_l(h_i^{l-1}W_l + b_l), \quad h_i^0 = x_i \]
Neural Networks

\[ h_i^l = f_l(h_i^{l-1} W_l + b_l), \quad h_i^0 = x_i \]

\[ \mathbb{E}[y_i | x_i] = g^{-1}(h_i^L w_{L+1} + b_{L+1}) \]
Neural Networks

\[ h^l_i = f_l(h^{l-1}_i W_l + b_l), \quad h^0_i = x_i \]

**ACTIVATION FUNCTION**

**WEIGHT MATRIX**

(PARAMETERS)

**HIDDEN UNIT / NEURON**

**HIDDEN LAYER**

\[ E[y_i | x_i] \]
Neural Networks

\[ \mathbf{X}_i \rightarrow h_i^1 \rightarrow h_i^2 \rightarrow \cdots \rightarrow h_i^L \rightarrow \mathbb{E}[y_i | x_i] \]
Neural Networks

\[ y_i \mid x_i = f_L(h_{iL}) \]

\[ E[y_i \mid x_i] \]
Bayesian Neural Networks

STEP #1: 

\[ p(\{W_l\}_{l=1}^{L+1}) = \prod_{l=1}^{L+1} p(W_l) \]

PRIOR

\[ \mathbb{E}[y_i | x_i] \]
### Bayesian Neural Networks

**STEP #2:**

\[
p(y|X, \{W_l\}_{l=1}^{L+1}) \prod_{l=1}^{L+1} p(W_l)
\]

**LIKELIHOOD**

\[p(y|X, \{W_l\}_{l=1}^{L+1})\]

\[
\mathbb{E}[y_i|x_i]
\]
Bayesian Neural Networks

STEP #3: \[ p(\{W_l\}_{l=1}^{L+1} | y, X) \]

POSTERIOR

\[ E[y_i | x_i] \]
Benefits of Bayesian Neural Networks
Benefits of Bayesian Neural Networks

Model Uncertainty for Safe Automation.

Benefits of Bayesian Neural Networks

Model Uncertainty for Safe Automation.


Posterior Predictive Distribution
Benefits of Bayesian Neural Networks

Model Uncertainty for Safe Automation.

\[ p(y^* | x^*, y, X) = \int_{W} p(y^* | x^*, W)p(W | y, X) \, dW \]

Benefits of Bayesian Neural Networks

Coherent Framework for Online / Sequential Learning.
Benefits of Bayesian Neural Networks

Coherent Framework for Online / Sequential Learning.

BAYESIAN UPDATING

\[
p(W|\{y_t, y_{t+1}\}, \{X_t, X_{t+1}\})
\propto p(y_{t+1}|X_{t+1}, W)p(W|y_t, X_t)
\]

LIKELIHOOD OF NEW DATA

POSTERIOR

SERVES AS PRIOR
Benefits of Bayesian Neural Networks

Coherent Framework for Online / Sequential Learning.

BAYESIAN UPDATING

\[
p(W | \{y_t, y_{t+1}\}, \{X_t, X_{t+1}\}) \\
\propto p(y_{t+1} | X_{t+1}, W)p(W | y_t, X_t)
\]

LIKELIHOOD OF NEW DATA

POSTERIOR

SERVES AS PRIOR

THOMPSON SAMPLING FOR EXPLORE/EXPLOIT

\[
\argmax_{\text{action}} \mathbb{E}[\text{reward}_{T+1} | \text{action}_{T+1}, \hat{W}]
\]

\[
\hat{W} \sim p(W | \{(\text{reward}_t, \text{action}_t)\}_{t=0}^T)
\]

Benefits of Bayesian Neural Networks

Regularization and Model Selection to Ease NN’s Data Hunger.
Benefits of Bayesian Neural Networks

Regularization and Model Selection to Ease NN’s Data Hunger.

SPARSITY-INDUCING PRIORS
Benefits of Bayesian Neural Networks

Regularization and Model Selection to Ease NN’s Data Hunger.

SPARSITY-INDUCING PRIORS

![Probability Density Plot]

BAYESIAN EVIDENCE FRAMEWORK

\[
P(D | \mathcal{H}_i) \approx \frac{P(D | w_{\text{MP}}, \mathcal{H}_i)}{P(w_{\text{MP}} | \mathcal{H}_i) \Delta w}
\]

Evidence \approx \frac{\text{Best fit likelihood}}{\text{Occam factor}}

[MacKay, 1991]
Great...so why aren’t Bayesian neural networks more prevalent?
Great…so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.
Great...so why aren’t Bayesian neural networks more prevalent?

Computational and Memory Cost: Obtaining distributions instead of point estimates is intrinsically more costly.

Computation: A fully Bayesian treatment requires solving high-dimensional integration problems, which are especially intractable to neural networks.
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.
  - **Computation:** A fully Bayesian treatment requires solving high-dimensional integration problems, which are especially intractable to neural networks.
  - **Memory:** Even coarse posterior approximations (e.g., fully-factorized Gaussian) usually require at least 2x the number of parameters.
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost**: Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations**: Usually variational inference is needed to approximate the posterior.
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations:** Usually variational inference is needed to approximate the posterior.

  - **Aggressive Factorization:** For the sake of tractability, need to factorize the posterior across layers (and usually further).

  
  
  $$p(\{W_l\}_{l=1}^{L+1} | y, X) \approx \prod_{l=1}^{L+1} q(W_l)$$
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations:** Usually variational inference is needed to approximate the posterior.

  - **Aggressive Factorization:** For the sake of tractability, need to factorize the posterior across layers (and usually further).

\[
p(\{W_l\}_{l=1}^{L+1} | y, X) \approx \prod_{l=1}^{L+1} q(W_l) \\
\approx \prod_{l=1}^{L+1} \prod_{j=1}^{R_l} \prod_{k=1}^{C_l} q(w_{l,j,k})
\]

[Blei et al., 2017]
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations:** Usually variational inference is needed to approximate the posterior.

- **Arbitrary and Difficult to Specify Priors:** NN weights have no intuitive interpretation (in comparison to, say, a Bernoulli model).
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost:** Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations:** Usually variational inference is needed to approximate the posterior.

- **Arbitrary and Difficult to Specify Priors:** NN weights have no intuitive interpretation (in comparison to, say, a Bernoulli model).

  - **Unintended Effects:** Priors on weights can have unintended effects on the distribution over output functions [Neal, 1994].
Great...so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost**: Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations**: Usually variational inference is needed to approximate the posterior.

- **Arbitrary and Difficult to Specify Priors**: NN weights have no intuitive interpretation (in comparison to, say, a Bernoulli model).

  - **Unintended Effects**: Priors on weights can have unintended effects on the distribution over output functions [Neal, 1994].

  

  Garbage in: arbitrary priors
  Garbage out: uncontrollable error bars

  ~ Michael Jordan on Bayesian Deep Learning, MLSS 2017
Great…so why aren’t Bayesian neural networks more prevalent?

- **Computational and Memory Cost**: Obtaining distributions instead of point estimates is intrinsically more costly.

- **Coarse Posterior Approximations**: Usually variational inference is needed to approximate the posterior.

- **Arbitrary and Difficult to Specify Priors**: NN weights have no intuitive interpretation (in comparison to, say, a Bernoulli model).
  - **Unintended Effects**: Priors on weights can have unintended effects on the distribution over output functions [Neal, 1994].

  - *Garbage in: arbitrary priors*
  - *Garbage out: uncontrollable error bars*

  ~ Michael Jordan on Bayesian Deep Learning, MLSS 2017
BACKGROUND

Bayesian Neural Networks and Their Priors
Bayesian Neural Networks

CONDITIONAL MODEL (SUPERVISED)
Bayesian Neural Networks

CONDITIONAL MODEL (SUPERVISED)

\[
p\left(\{W_l\}_{l=1}^{L+1} \mid y, X\right) = \frac{p(y \mid X, \{W_l\}_{l=1}^{L+1}) \prod_{l=1}^{L+1} p(W_l)}{p(y \mid X)}
\]

\[
p(y \mid X) = \int_{W} p(y \mid X, \{W_l\}_{l=1}^{L+1}) \prod_{l=1}^{L+1} p(W_l) \, dW
\]
Bayesian Neural Networks

DENSITY NETWORK (UNSUPERVISED)
[Amemiya, 1993; MacKay, 1999]
Bayesian Neural Networks

DENSITY NETWORK (UNSUPERVISED)
[Amemiya, 1993; MacKay, 1999]
Bayesian Neural Networks

**DENSITY NETWORK (UNSUPERVISED)**
[Amemiya, 1993; MacKay, 1999]

\[
p\left(\{z_i\}_{i=1}^N | X; \{W_l\}_{l=1}^{L+1}\right) = \frac{\prod_{i=1}^N p(x_i | z_i; \{W_l\}_{l=1}^{L+1})p(z_i)}{p(X; \{W_l\}_{l=1}^{L+1})}
\]

\[
p(X; \{W_l\}_{l=1}^{L+1}) = \prod_{i=1}^N \int_{z_i} p(x_i | z_i; \{W_l\}_{l=1}^{L+1})p(z_i) \, dz_i
\]
Bayesian Neural Networks

CONDITIONAL MODEL

\[ x \rightarrow h_1 \rightarrow h_2 \rightarrow E[y|x] \]

\[ w_1 \sim p(w_1) \]
\[ w_2 \sim p(w_2) \]
\[ w_3 \sim p(w_3) \]

DENSITY NETWORK

\[ p(z) \sim z \rightarrow h_1 \rightarrow h_2 \rightarrow E[x|z] \]

\[ \hat{w}_1 \]
\[ \hat{w}_2 \]
\[ \hat{w}_3 \]
Bayesian Neural Networks

CONDITIONAL MODEL

- $x \rightarrow h_1 \rightarrow h_2 \rightarrow E[y|x]$
- $w_1 \sim p(w_1)$
- $w_2 \sim p(w_2)$
- $w_3 \sim p(w_3)$

DENSITY NETWORK

- $p(z) \sim z$
- $\hat{w}_1$
- $\hat{w}_2$
- $\hat{w}_3$
- $\hat{E}[x|z]$
Priors for Neural Networks

A SPECTRUM OF PRIORS

**NONPARAMETRIC**

Infinite dimensional priors that give the model adaptive capacity. Examples: Dirichlet process, Indian Buffet process.

**OBJECTIVE**

Priors that ‘follow the data,’ perhaps having frequentist properties. Examples: diffuse, Jeffreys, reference, empirical Bayes.

**INDUCTIVE BIAS**

Priors that give the model some inductive bias that improves performance. Example: sparsity.

**SUBJECTIVE**

Priors determined by previous experiments or expert knowledge.
Priors for Neural Networks

CONDITIONAL MODEL: $p(W)$

**NONPARAMETRIC**
- GP Equivalence [Neal, 1994]
- ARD [Neal, 1994; MacKay, 1994]

**INDUCTIVE BIAS**
- Gaussian [Buntine+, 1991; MacKay, 1991]
- Student-t [Neal, 1994]
- Laplace [Williams, 1995]
- Horseshoe [Ghosh+, 2017]

**OBJECTIVE**
- Jeffreys* [Lee, 2005]
- Reference* [Lee, 2005]
  *only for output weights

**SUBJECTIVE**

**OBJECTIVE BY PROXY**
- Init. NNs from Trees [Arunava, 1997]
Priors for Neural Networks

**DENSITY NETWORK:** $p(z)$

**NONPARAMETRIC**
- ?

**OBJECTIVE**
- Parametrized [Chen+, 2017; Makhzani+, 2015]
- Empirical/Parametrized [Tomczak+, 2018; Higgins+, 2017]

**SUBJECTIVE**
- Context Free Grammar [Kushner+, 2017]

**INDUCTIVE BIAS**
- Gaussian [MacKay, 1999; Kingma+, 2014; Rezende+, 2014]
- Laplace [Kingma, unpublished]
- Scale Mixture [Kingma, unpublished]
- Discrete [Neal, 1992]
- Structured [Johnson+, 2016]
Priors for Neural Networks

**DENSITY NETWORK: \( p(z) \)**

**OBJECTIVE**

- **PARAMETRIC**
  - Jeffreys* [Lee, 2005]
  - Reference* [Lee, 2005]
  - *only for output weights

**SUBJECTIVE**

- **PARAMETRIC**
  - Context Free Grammar [Kushner+, 2017]

**NONPARAMETRIC**

- ?

**CONDITIONAL MODEL: \( p(W) \)**

**OBJECTIVE**

- **PARAMETRIC**
  - GP Equivalence [Neal, 1994]
  - ARD [Neal, 1994; MacKay, 1994]

- **NONPARAMETRIC**
  - GP Equivalence [Neal, 1994]
  - ARD [Neal, 1994; MacKay, 1994]

**INDUCTIVE BIAS**

- **PARAMETRIC**
  - Gaussian [Buntine+, 1991; MacKay, 1991]
  - Student-t [Neal, 1994]
  - Laplace [Williams, 1995]
  - Horseshoe [Ghosh+, 2017]

- **NONPARAMETRIC**
  - GP Equivalence [Neal, 1994]
  - ARD [Neal, 1994; MacKay, 1994]

**SUBJECTIVE BY PROXY**

- **PARAMETRIC**
  - Init. NNs from Trees [Arunava, 1997]

- **NONPARAMETRIC**
  - GP Equivalence [Neal, 1994]
  - ARD [Neal, 1994; MacKay, 1994]
Priors that ‘follow the data,’ perhaps having frequentist properties. Examples: diffuse, Jeffreys, reference, empirical Bayes.
**Objective Priors**

Reference Priors [Bernardo, 1979] are objective priors defined as:

\[ p^*(\theta) = \arg\max_{p(\theta)} I[\theta, X] \]
Objective Priors

Reference Priors [Bernardo, 1979] are objective priors defined as:

$$ p^*(\theta) = \arg\max_{p(\theta)} I[\theta, X] $$

$$ = \arg\max_{p(\theta)} \int_X p(X) \text{ KLD}[p(\theta|X)||p(\theta)] \, dX $$
Reference Priors [Bernardo, 1979] are objective priors defined as:

$$p^*(\theta) = \arg\max_{p(\theta)} \mathbb{I}[\theta, X]$$

$$= \arg\max_{p(\theta)} \int_X p(X) \text{KLD}[p(\theta|X)||p(\theta)] \, dX$$

Independent of any dataset for generative models.
Objective Priors

Reference Priors [Bernardo, 1979] are objective priors defined as:

\[
p^*(\theta) = \arg\max_{p(\theta)} \mathbb{I}[\theta, X]
\]

\[
= \arg\max_{p(\theta)} \int_X p(X) \text{KLD}[p(\theta|X)||p(\theta)] \, dX
\]

- **Independent** of any dataset for generative models.
- Need posterior in closed-form and therefore difficult to derive.
Objective Priors

Reference Priors [Bernardo, 1979] are objective priors defined as:

\[ p^* (\theta) = \arg \max_{p(\theta)} \mathbb{E}[\theta, X] \]

\[ = \arg \max_{p(\theta)} \int_X p(X) \text{KLD}[p(\theta|X) || p(\theta)] \, dX \]

- **Independent** of any dataset for generative models.

- Need posterior in closed-form and therefore difficult to derive.

- Termed *reference* since they serve as a reference against which to compare other priors.
Reference Priors [Bernardo, 1979] are objective priors defined as:

$$p^*(\theta) = \arg\max_{p(\theta)} I[\theta, X]$$

$$= \arg\max_{p(\theta)} \int_X p(X) \text{KLD}[p(\theta|X)||p(\theta)] \, dX$$

Bernoulli Priors:
Objective Priors

Reference Priors [Bernardo, 1979] are objective priors defined as:

\[ p^*(\theta) = \arg\max_{p(\theta)} \mathbb{I}[\theta, X] \]

\[ = \arg\max_{p(\theta)} \int_X p(X) \text{KLD}[p(\theta|X)||p(\theta)] \, dX \]

Bernoulli Priors:

![Bernoulli Priors Diagram](image)
Markov Chain Monte Carlo [Lafferty & Wasserman, 2001]: Given a likelihood function, use MCMC to draw samples from the reference prior.

\[ \{ p(x|\theta) \} \rightarrow \{ \hat{\theta}_1, \ldots, \hat{\theta}_S \} \]
Markov Chain Monte Carlo [Lafferty & Wasserman, 2001]: Given a likelihood function, use MCMC to draw samples from the reference prior.

**Drawback**: Have to perform a second step of modeling in order to evaluate the prior’s density.

\[
\{ p(x|\theta) \} \longrightarrow \{ \hat{\theta}_1, \ldots, \hat{\theta}_S \}
\]
Previous Work

- **Markov Chain Monte Carlo** [Lafferty & Wasserman, 2001]: Given a likelihood function, use MCMC to draw samples from the reference prior.
  
  |
  | **Drawback**: Have to perform a second step of modeling in order to evaluate the prior’s density.
  |
  |
  \[
  \{p(x|\theta)\} \rightarrow \{\hat{\theta}_1, \ldots, \hat{\theta}_S\}
  \]

- **Numerical Algorithm** [Berger et al., 2009]: Given a user-specified parameter value and likelihood function, it can compute the reference prior density at that point.

  \[
  \{p(x|\theta), \theta_0\} \rightarrow p^*(\theta_0)
  \]
Previous Work

Markov Chain Monte Carlo [Lafferty & Wasserman, 2001]: Given a likelihood function, use MCMC to draw samples from the reference prior.

**Drawback**: Have to perform a second step of modeling in order to evaluate the prior’s density.

\[
\{p(x|\theta)\} \rightarrow \{\hat{\theta}_1, \ldots, \hat{\theta}_S\}
\]

Numerical Algorithm [Berger et al., 2009]: Given a user-specified parameter value and likelihood function, it can compute the reference prior density at that point.

**Drawback**: Have to choose the evaluation points yourself, which is very hard to do in high dimensions.

\[
\{p(x|\theta), \theta_0\} \rightarrow p^*(\theta_0)
\]
Proposed Method

- **Parametric Approximation:** Posit a parametric family and optimize the parameters to find an approximation.

\[ p^*(\theta) \approx p(\theta; \lambda^*) \]
**Proposed Method**

- **Parametric Approximation:** Posit a parametric family and optimize the parameters to find an approximation.

\[
p^*(\theta) \approx p(\theta; \lambda^*)
\]

- **Lower Bound Optimization:** We derive a lower bound on the mutual information and optimize it w.r.t. \(\lambda\).

\[
\mathbb{I}[\theta, X] \geq \mathbb{E}_{\theta, \lambda, X} \left[ \log p(X|\theta) - \frac{1}{1 + \alpha} \log \mathbb{E}_{\theta, \lambda} \left[ p(X|\theta)^{1+\alpha} \right] \right]
\]

for \(\alpha > 0\)
Proposed Method

- **Parametric Approximation**: Posit a parametric family and optimize the parameters to find an approximation.

\[ p^*(\theta) \approx p(\theta; \lambda^*) \]

- **Lower Bound Optimization**: We derive a lower bound on the mutual information and optimize it w.r.t. \( \lambda \).

\[
\mathbb{I}[\theta, X] \geq \mathbb{E}_{\theta, X, \lambda} \left[ \log p(X|\theta) - \frac{1}{1 + \alpha} \log \mathbb{E}_{\theta, \lambda} \left[ p(X|\theta)^{1+\alpha} \right] \right]
\]

for \( \alpha > 0 \)

\[
\approx \mathbb{E}_{\theta, X} \left[ \log p(X|\theta) - \max_s \log p(X|\hat{\theta}_s) \right]
\]

[Li & Turner, 2016]
Proposed Method

**Lower Bound Optimization:** We derive a lower bound on the mutual information and optimize it w.r.t. $\lambda$.

$$\{p(x|\theta), p(\theta; \lambda)\} \rightarrow p(\theta; \lambda^*)$$
Proposed Method

**Lower Bound Optimization:** We derive a lower bound on the mutual information and optimize it w.r.t. $\lambda$.

$$\{p(x|\theta), p(\theta; \lambda)\} \rightarrow p(\theta; \lambda^*)$$

**BENEFITS**

**Gaussian mean:** $p^*(\mu) \propto 1$
**Proposed Method**

**Lower Bound Optimization:** We derive a lower bound on the mutual information and optimize it w.r.t. $\lambda$.

$$\{p(x|\theta), p(\theta; \lambda)\} \rightarrow p(\theta; \lambda^*)$$

**BENEFITS**

**Gaussian mean:** $p^*(\mu) \propto 1$

**Conjugate Reference Priors:** Can pick a prior family that is conjugate and then find the member that is nearest to a reference prior.
**Proposed Method**

**Lower Bound Optimization:** We derive a lower bound on the mutual information and optimize it w.r.t. $\lambda$.

$$\{p(x|\theta), p(\theta; \lambda)\} \longrightarrow p(\theta; \lambda^*)$$

**BENEFITS**

**Gaussian mean:** $p^*(\mu) \propto 1$

- **Conjugate Reference Priors:** Can pick a prior family that is conjugate and then find the member that is nearest to a reference prior.

- **Proper Reference Priors:** If concerned b/c a reference prior is improper, can use an approximation that is proper.
Proposed Method

Lower Bound Optimization: We derive a lower bound on the mutual information and optimize it w.r.t. $\lambda$.

$$\{p(x|\theta), p(\theta; \lambda)\} \longrightarrow p(\theta; \lambda^*)$$

**BENEFITS**

- **Comparison to Lafferty & Wasserman’s [2001] MCMC:** No need to perform a second modeling step to evaluate the prior.

- **Comparison to Berger et al.’s [2009] Algorithm:** No need to specify evaluation points.
Proposed Method

**Drawback?**: Need to specify the variational family.
**Proposed Method**

**Drawback?:** Need to specify the variational family.

Notice that the objective doesn’t require evaluation of the prior, only expectations / samples:

\[
\mathbb{I}[\theta, X] \geq \mathbb{E}_{\theta, X} \left[ \log p(X|\theta) - \frac{1}{1 + \alpha} \log \mathbb{E}_{\theta, X} [p(X|\theta)^{1+\alpha}] \right]
\]
**Drawback?**: Need to specify the variational family.

Notice that the objective doesn’t require evaluation of the prior, only expectations / samples:

\[
\mathbb{I}[\theta, X] \geq \mathbb{E}_{\theta, \lambda, X} \left[ \log p(X|\theta) - \frac{1}{1 + \alpha} \log \mathbb{E}_{\theta, \lambda} \left[ p(X|\theta)^{1+\alpha} \right] \right]
\]

Thus, we can use an **implicit prior**:

\[
\hat{\theta} = f(\hat{\epsilon}; \lambda), \quad \hat{\epsilon} \sim p_0(\epsilon)
\]

This results in \(f\) being a functional sampler, similar to the MCMC approach.
Recovering Jeffreys Priors

See paper / dissertation for two-sample tests comparing approximation methods.
“…to improve our variational bounds we should improve our priors and not just the encoder and decoder….perhaps we should investigate multimodal priors…”

Finding the Density Net’s Reference Prior

\[
p(z; \lambda) = p(W)\]

\[
\text{DENSITY NET}
\]

ALGORITHM
Finding the Density Net’s Reference Prior

ALGORITHM

STEP #1: sample $\hat{z}_0 \sim p(z; \lambda)$

$p(z; \lambda) \ p(W)$

DENSITY NET
Finding the Density Net’s Reference Prior

**ALGORITHM**

STEP #1: sample $\hat{z}_0 \sim p(z; \lambda)$

STEP #2: sample $\hat{W} \sim p(W)$

$p(z; \lambda) \ p(W)$
Finding the Density Net’s Reference Prior

**ALGORITHM**

**STEP #1:** sample $\hat{z}_0 \sim p(z; \lambda)$

**STEP #2:** sample $\hat{W} \sim p(W)$

**STEP #3:** sample $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_N\} \sim p(x|\hat{z}_0, \hat{W})$
Finding the Density Net’s Reference Prior

**ALGORITHM**

1. Sample \( \hat{z}_0 \sim p(z; \lambda) \)
2. Sample \( \hat{W} \sim p(W) \)
3. Sample \( \hat{X} = \{\hat{x}_1, \ldots, \hat{x}_N\} \sim p(x|\hat{z}_0, \hat{W}) \)
4. Sample \( \{\hat{z}_1, \ldots, \hat{z}_S\} \sim p(z; \lambda) \)

**DENSITY NET**

\( p(z; \lambda) \ p(W) \)
**Finding the Density Net’s Reference Prior**

**ALGORITHM**

**STEP #1:** sample $\hat{z}_0 \sim p(z; \lambda)$

**STEP #2:** sample $\hat{W} \sim p(W)$

**STEP #3:** sample $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_N\} \sim p(x|\hat{z}_0, \hat{W})$

**STEP #4:** sample $\{\hat{z}_1, \ldots, \hat{z}_S\} \sim p(z; \lambda)$

**STEP #5:** maximize w.r.t. $\lambda$

$$J(\lambda) = \log p(\hat{X}|\hat{z}_0, \hat{W}) - \max_s \log p(\hat{X}|\hat{z}_s, \hat{W})$$
Finding the Density Net’s Reference Prior

ALGORITHM

STEP #1: sample $\hat{z}_0 \sim p(z; \lambda)$

STEP #2: sample $\hat{W} \sim p(W)$

STEP #3: sample $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_N\} \sim p(x|\hat{z}_0, \hat{W})$

STEP #4: sample $\{\hat{z}_1, \ldots, \hat{z}_S\} \sim p(z; \lambda)$

STEP #5: maximize w.r.t. $\lambda$

$$J(\lambda) = \log p(\hat{X}|\hat{z}_0, \hat{W}) - \max_s \log p(\hat{X}|\hat{z}_s, \hat{W})$$
Finding the Density Net’s Reference Prior

ALGORITHM

STEP #1: sample $\hat{z}_0 \sim p(z; \lambda)$

STEP #2: sample $\hat{W} \sim p(W)$

STEP #3: sample $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_N\} \sim p(x|\hat{z}_0, \hat{W})$

STEP #4: sample $\{\hat{z}_1, \ldots, \hat{z}_S\} \sim p(z; \lambda)$

STEP #5: maximize w.r.t. $\lambda$

$J(\lambda) = \log p(\hat{X}|\hat{z}_0, \hat{W}) - \max_s \log p(\hat{X}|\hat{z}_s, \hat{W})$

STEP #6: repeat
Finding the Density Net’s Reference Prior

Improves performance in low dimensions but no gains in high dimensions (25+).
Priors for Neural Networks

**DENSITY NETWORK:** \( p(z) \)

**OBJECTIVE**

- Parametrized [Chen+, 2017; Makhzani+, 2015]
- Empirical/Parametrized [Tomczak+, 2018; Higgins+, 2017]
- Reference Approx. [Nalisnick & Smyth, 2017]

**NONPARAMETRIC**

**INDUCTIVE BIAS**

- Jeffreys [Lee, 2005]
- Reference [Lee, 2005]
- Reference Approx.

**SUBJECTIVE**

- SUBJECTIVE BY PROXY

**CONDITIONAL MODEL: \( p(W) \)**
CONTRIBUTION

Nonparametric Priors

CHAPTER #7 [Nalisnick & Smyth, ICLR 2017]

NONPARAMETRIC

Infinite dimensional priors that give the model adaptive capacity. Examples: Dirichlet process, Indian Buffet process.
Nonparametric Priors for Density Nets

FINITE DIMENSIONAL
Nonparametric Priors for Density Nets

**Finite Dimensional**
- Gaussian Prior
- Griffiths-Engen-McCloskey (GEM) Prior

**Infinite Dimensional**
- Reference Prior
- Dirichlet Process Mixture Prior
What about the IBP, the natural nonparametric prior for latent feature models?
What about the IBP, the natural nonparametric prior for latent feature models?

Structured Variational Autoencoders for the Beta-Bernoulli Process

Rachit Singh* Jeffrey Ling* Finale Doshi-Velez
Harvard University
{rachitsingh@college, jling@college, finale@seas}.harvard.edu

Abstract

Beta-Bernoulli processes, also known as Indian buffet processes, are nonparametric priors that allow generative models to automatically infer the number of features in datasets. However, inference for these models proves to be challenging, often relying on specific forms of the likelihood for computational tractability. We propose to amortize inference using a variational autoencoder trained via gradient descent, allowing for arbitrary likelihood models. Our model extends previously considered mean field variational methods with a structured posterior and new developments in the training of discrete variable VAEs. We experimentally demonstrate a Beta-Bernoulli process VAE that learns decomposable latent features and allows for scalable inference of arbitrary likelihoods on large datasets.

Training

Two methods to compute backpropagation gradients:

- Black box variational inference (BBVI): directly backpropagate with score estimator
- Replace non-reparameterizable variables in variational approximation:
  - Bernoulli → Gumbel softmax (Maddison et al. 2017, Jang et al. 2017)
  - Beta → Kumaraswamy (Nalisnick & Smyth 2017)
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

CONCENTRATION

BASE MEASURE
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

BASE MEASURE

\[ \mu_k \sim H(\mu) \]
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

**BASE MEASURE**

\[ \mu_k \sim H(\mu) \]

**WEIGHTS**

\[ \{\pi_1, \ldots, \pi_k, \ldots\} = \text{GEM}(\alpha) \]

\[ \pi_k = \nu_k \prod_{j=1}^{k} (1 - \nu_j) \]

\[ \nu_k \sim \text{Beta}(1, \alpha) \]
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

BASE MEASURE

\[ \mu_k \sim H(\mu) \]

WEIGHTS

\[ \{\pi_1, \ldots, \pi_k, \ldots\} = \text{GEM}(\alpha) \]

\[ \pi_k = \nu_k \prod_{j=1}^{k} (1 - \nu_j) \]

\[ \nu_k \sim \text{Beta}(1, \alpha) \]

DISTRIBUTION

\[ G(\mu) = \sum_{k=1}^{\infty} \pi_k \delta[\mu_k] \]
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

\( \mu \)

CONCENTRATION

BASE MEASURE
The Dirichlet Process

\[ G \sim DP(\alpha, H) \]

\[ \hat{\mu} \sim H(\mu) \]
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

\[ \hat{\mu} \sim H(\mu) \]
The Dirichlet Process

\[ G \sim DP(\alpha, H) \]

CONCENTRATION \quad BASE MEASURE

\[ \hat{\mu} \sim H(\mu) \]
The Dirichlet Process

\[ G \sim DP(\alpha, H) \]

CONCENTRATION

BASE MEASURE
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

\[ \{\hat{\pi}_1, \ldots, \hat{\pi}_3, \ldots\} \sim \text{GEM}(\alpha) \]

CONCENTRATION

BASE MEASURE
The Dirichlet Process

\[ G \sim \text{DP}(\alpha, H) \]

\[ \{\hat{\pi}_1, \ldots, \hat{\pi}_3, \ldots\} \sim \text{GEM}(\alpha) \]
The Dirichlet Process

\[ G \sim DP(\alpha, H) \]

\( \hat{\mu} \sim G(\mu) \)
The Dirichlet Process

\[ G \sim DP(\alpha, H) \]

\[ \hat{\mu} \sim G(\mu) \]

CONCENTRATION

BASE MEASURE

\[ \mu \]

\[ \mu_2 \]

\[ \mu_1 \]

\[ \mu_3 \]
The Dirichlet Process Mixture Model

\[ G \sim \text{DP}(\alpha, H) \]
The Dirichlet Process Mixture Model

\[ G \sim \text{DP}(\alpha, H) \quad z \mid G \sim \int p(z \mid \mu) G(\mu) \]
The Dirichlet Process Mixture Model

\[ G \sim \text{DP}(\alpha, H) \]

\[ z \mid G \sim \int p(z \mid \mu)G(\mu) \]

Collapsing out \( G \), we have...

\[ z \mid \{\mu_k\}, \{\pi_k\} \sim \sum_{k=1}^{\infty} \pi_k p(z \mid \mu_k) \]

\[ \{\pi_k\} \sim \text{GEM}(\alpha) \]

\[ \{\mu_k\} \sim H(\mu) \]
The Dirichlet Process Mixture Model

\[ G \sim \text{DP}(\alpha, H) \quad z | G \sim \int p(z | \mu) G(\mu) \]

COLLAPSING OUT G, WE HAVE...

\[ z \sim \text{DPMM}(p, \alpha, H) \quad z | \{\mu_k\}, \{\pi_k\} \sim \sum_{k=1}^{\infty} \pi_k p(z | \mu_k) \]

\[ \{\pi_k\} \sim \text{GEM}(\alpha) \]

\[ \{\mu_k\} \sim H(\mu) \]
The Dirichlet Process Mixture Model

\[ z \sim \text{DPMM}(p, \alpha, H) \]
The Dirichlet Process Mixture Model

\[ z \sim DPMM(p, \alpha, H) \]
The Dirichlet Process Mixture Model

\[ z \sim \text{DPMM}(p, \alpha, H) \]
The Dirichlet Process Mixture Model

\[ z \sim \text{DPMM}(p, \alpha, H) \]
The Dirichlet Process Mixture Model

\[ z \sim \text{DPMM}(\rho, \alpha, H) \]
Nonparametric Density Networks
Nonparametric Density Networks

**STICK-BREAKING DENSITY NETWORK**

\[ \pi_i \sim \text{GEM}(\alpha) \]

\[ x_i \sim p(x_i | \pi_i; \{W\}_{l=1}^{L+1}) \]

**Adaptive Width:** Due to order bias, can think of the network as having adaptive width, using enough latent variables as the data needs.
## Nonparametric Density Networks

<table>
<thead>
<tr>
<th>STICK-BREAKING DENSITY NETWORK</th>
<th>LATENT DIRICHLET PROCESS DENSITY NETWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_i \sim \text{GEM}(\alpha) )</td>
<td>( z_i \sim \text{DPMM}(N(\mu, \Sigma), \alpha, N(\mu)) )</td>
</tr>
<tr>
<td>( x_i \sim p(x_i</td>
<td>\pi_i; {W}_{l=1}^{L+1}) )</td>
</tr>
</tbody>
</table>

**Adaptive Width:** Due to order bias, can think of the network as having adaptive width, using enough latent variables as the data needs.

**Adaptive Complexity:** Can draw the latent representation from as complex a distribution as needed.
Gradient-Based Inference: Want to use an inference model to predict the latent variables and thus need to take gradients through samples. The Beta distribution does not allow this so we use the Beta-like Kumaraswamy distribution as the posterior on stick segments.
Amortized Variational Inference

Gradient-Based Inference: Want to use an inference model to predict the latent variables and thus need to take gradients through samples. The Beta distribution does not allow this so we use the Beta-like Kumaraswamy distribution as the posterior on stick segments.

Poondi Kumaraswamy (1930-1988)

STICK-BREAKING VAE

Kumaraswamy Samples

Truncated posterior; not necessary but learns faster
Gradient-Based Inference: Want to use an inference model to predict the latent variables and thus need to take gradients through samples. The Beta distribution does not allow this so we use the Beta-like Kumaraswamy distribution as the posterior on stick segments.

Truncated posterior; not necessary but learns faster.
Results: MNIST Samples

STICK-BREAKING VAE

DPMM VAE
Results: MNIST Samples

Truncation level of 50. Beta(1,5) prior.

DIMENSIONALITY

STICK-BREAKING VAE

DPMM VAE
Results: MNIST Samples

Truncation level of 50. Beta(1,5) prior.

DIMENSIONALITY

2
3
4
5

50

STICK-BREAKING VAE

DPMM VAE

Samples from Component #1

Samples from Component #5
Results: MNIST Latent Space

Gaussian Latent Space (t-SNE)
Results: MNIST Latent Space

Gaussian Latent Space (t-SNE)  GEM Latent Space (t-SNE)
Results: MNIST Latent Space

Gaussian Latent Space (t-SNE)  
GEM Latent Space (t-SNE)  
DPMM Latent Space (t-SNE)
## Results: MNIST Latent Space

<table>
<thead>
<tr>
<th></th>
<th>k=3</th>
<th>k=5</th>
<th>k=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB-VAE</td>
<td>9.34</td>
<td>8.65</td>
<td>8.90</td>
</tr>
<tr>
<td>DLGMM</td>
<td>9.14</td>
<td>8.38</td>
<td>8.42</td>
</tr>
<tr>
<td>Gauss VAE</td>
<td>28.4</td>
<td>20.96</td>
<td>15.33</td>
</tr>
<tr>
<td>GMVAE$^6$</td>
<td>—</td>
<td>8.96</td>
<td>—</td>
</tr>
</tbody>
</table>

**Error Rate** of k-Nearest Neighbor Classifier on Latent Space
# Results: MNIST Latent Space

<table>
<thead>
<tr>
<th>Latent Space</th>
<th>k=3</th>
<th>k=5</th>
<th>k=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB-VAE</td>
<td>9.34</td>
<td>8.65</td>
<td>8.90</td>
</tr>
<tr>
<td>DLGMM</td>
<td>9.14</td>
<td>8.38</td>
<td>8.42</td>
</tr>
<tr>
<td>Gauss VAE</td>
<td>28.4</td>
<td>20.96</td>
<td>15.33</td>
</tr>
<tr>
<td>GMVAE$^6$</td>
<td>—</td>
<td>8.96</td>
<td>—</td>
</tr>
</tbody>
</table>

**Error Rate** of k-Nearest Neighbor Classifier on Latent Space
Results: Likelihood and Semi-Supervised

HELD-OUT LIKELIHOOD

SEMI-SUPERVISED CLASSIFICATION
## Results: Likelihood and Semi-Supervised

<table>
<thead>
<tr>
<th>HELD-OUT LIKELIHOOD</th>
<th>$-\log p_\theta(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNIST</td>
</tr>
<tr>
<td>DLGMM (500d-3x25s)</td>
<td>96.50</td>
</tr>
<tr>
<td>DLDPMM (500d-17tx25s)</td>
<td>96.91</td>
</tr>
<tr>
<td>Gauss-VAE (500d-25s)</td>
<td>96.80</td>
</tr>
<tr>
<td>SB-VAE (500d-25t)</td>
<td>98.01</td>
</tr>
</tbody>
</table>

**Negative log likelihood** on test set (importance sampled)

---

## Semi-Supervised Classification
### Results: Likelihood and Semi-Supervised

#### HELD-OUT LIKELIHOOD

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST</th>
<th>Omniglot</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLGMM (500d-3x25s)</td>
<td>96.50</td>
<td>123.50</td>
</tr>
<tr>
<td>DLDPM (500d-17tx25s)</td>
<td>96.91</td>
<td>123.76</td>
</tr>
<tr>
<td>Gauss-VAE (500d-25s)</td>
<td>96.80</td>
<td>119.18</td>
</tr>
<tr>
<td>SB-VAE (500d-25t)</td>
<td>98.01</td>
<td>–</td>
</tr>
</tbody>
</table>

*Negative log likelihood* on test set (importance sampled)

#### SEMI-SUPERVISED CLASSIFICATION

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST (N=45,000)</th>
<th>SVHN (N=65,000)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
<td>5%</td>
</tr>
<tr>
<td>SB-DGM</td>
<td>4.86±.14</td>
<td>5.29±.39</td>
</tr>
<tr>
<td>Gauss-DGM</td>
<td>3.95±.15</td>
<td>4.74±.43</td>
</tr>
<tr>
<td>kNN</td>
<td>6.13±.13</td>
<td>7.66±.10</td>
</tr>
</tbody>
</table>

*Error rate* on test set when training with X% labeled.

Nonparametric version of (Kingma et al., NIPS 2014)’s M2 model
Priors for Neural Networks

CONDITIONAL MODEL: $p(W)$

**NONPARAMETRIC**
- GP Equivalence [Neal, 1994]
- ARD [Neal, 1994; MacKay, 1994]

**INDUCTIVE BIAS**

**SUBJECTIVE BY PROXY**

**OBJECTIVE**

**SUBJECTIVE**

**NONPARAMETRIC**
- Stick-Breaking [Nalisnick & Smyth, 2017]
- DP-Mixture [Nalisnick et al., 2016]

DENSITY NETWORK: $p(z)$
Future Directions and Summary
Future Directions

Structured Priors / Models: [Johnson et al., 2016] demonstrated how to combine density networks and latent graphical models.

From M. Johnson’s DL summer school slides
‘Neural Nonparametrics’: Recent deep learning innovations such as differentiable memory [Graves et al., 2014] and adaptive computation [Graves, 2016] have the character of nonparametrics. I think they can be formalized with traditional BNP priors.
Bayesian neural networks have the potential to quantify uncertainty, learn in sequential settings, and ease NN’s hunger for data.

Choice of prior is important: They endow the model with characteristics such as the ability to follow the data and grow in complexity.

Contributions presented: Extended and explored reference priors and nonparametric priors for density networks.
Many Thanks to...

MY COMMITTEE

MY RESEARCH GROUP
Appendix
Can re-write the Reference prior objective as:

\[ p^*(\theta) = \arg \max_{p(\theta)} \int_{\mathcal{D}} p(\mathcal{D}) \int_{\theta} p(\theta|\mathcal{D}) \log \frac{p(\theta|\mathcal{D})}{p(\theta)} d\theta d\mathcal{D} \]

\[ = \arg \max_{p(\theta)} \int_{\theta} p(\theta) \int_{\mathcal{D}} p(\mathcal{D}|\theta) \log \frac{p(\mathcal{D}|\theta)}{p(\mathcal{D})} d\mathcal{D} d\theta \]

\[ = \arg \max_{p(\theta)} \mathbb{E}_{p(\theta)} \text{KLD}[p(\mathcal{D}|\theta) \| p(\mathcal{D})]. \]
MODEL #2: Dirichlet Process Variational Autoencoder

(Nalisnick et al., 2016)

\[
\mathcal{L}_{SGVB} = \sum_k \mu_{\pi_k} \left[ \frac{1}{S} \sum_s \log p_{\theta}(x_i | \hat{z}_{i,k,s}) + \mathbb{E}_{q_k} [\log p(z_i)] \right] \\
- \text{KLD}[q(\pi_k | x_i) || p(\pi_k)] - \frac{1}{S} \sum_s \log \sum_k \hat{\pi}_{i,k,s} q(\hat{z}_{i,k,s}, \phi_k)
\]