Implicit Variational Inference with Kernel Density Ratio Fitting

Jiaxin Shi [*1] **Shengyang Sun** [*2] **Jun Zhu** [1]

Abstract

Recent progress in variational inference has paid much attention to the flexibility of variational posteriors. Work has been done to use implicit distributions, i.e., distributions without tractable likelihoods as the variational posterior. However, existing methods on implicit posteriors still face challenges of noisy estimation and can hardly scale to high-dimensional latent variable models. In this paper, we present an implicit variational inference approach with kernel density ratio fitting that addresses these challenges. As far as we know, for the first time implicit variational inference is successfully applied to Bayesian neural networks, which shows promising results on both regression and classification tasks.

1. Introduction

Bayesian methods have been playing vital roles in machine learning by providing a principled approach for uncertainty modeling, posterior inference and preventing overfitting (Ghahramani, 2015). As it becomes a common practice to build large and deep models that have many parameters (LeCun et al., 2015), it is even more important to have a sophisticated Bayesian formulation to protect these models. For example, Bayesian Neural Networks (Neal, 2012) have shown promise in dealing with model uncertainty and learning with few labeled data.

Besides a few simple examples, Bayesian inference is typically challenging, for which variational inference has been a standard workhorse to approximate the true posterior. Traditional variational inference focuses on mean-field variational posteriors to get analytical variational updates. While recent progress in this field drives variational inference into stochastic, differentiable and amortized (Hoffman et al., 2013; Paisley et al., 2012; Mnih & Gregor, 2014; Kingma & Welling, 2013), which does not rely on analytical updates anymore, mean-field posteriors are still commonly used as the variational family. This greatly restricts the flexibility of the variational posterior, especially in high-dimensional spaces, which often leads to biased inference. There has been some works that try to improve the flexibility of variational posteriors, borrowing ideas from invertible transformation of probability distributions to get tractable density (Rezende & Mohamed, 2015; Kingma et al., 2016).

Although invertible transformation is a promising direction to increase the expressiveness of the variational posterior, we argue that a more flexible variational family can be constructed by using general deterministic or stochastic transformations, which is not necessarily invertible. As a common result, the variational posterior does not admit a tractable likelihood, despite there is a way to sample from it. This kind of distribution is called implicit distributions, and for variational methods that use an implicit variational posterior (also known as variational programs (Ranganath et al., 2016), wild variational approximations (Liu & Feng, 2016)), we refer to them as *Implicit Variational Inference* (Implicit VI).

In this paper we present an implicit variational inference approach that uses kernel density ratio fitting, which can scale to high-dimensional latent variable models. As far as we know, for the first time implicit variational inference is successfully applied to Bayesian neural networks.

2. Implicit Variational Inference

Consider a generative model $p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, where $\mathbf{z}$ represents latent variables. In variational inference, a variational distribution $q_\phi(\mathbf{z})$ in some parametric family is chosen to approximate the true posterior $p(\mathbf{z}|\mathbf{x})$. The objective to optimize is called the *evidence lower bound* (ELBO):

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z})\|p(\mathbf{z})). \qquad (1)$$

We can see that the challenge of using an implicit q_ϕ is that computing $\mathrm{KL}(q_\phi(\mathbf{z})\|p(\mathbf{z}))$ requires to evaluate the density of q_ϕ, which is intractable for an implicit distribution.

[*]Equal contribution [1]Department of Computer Science & Technology, TNList Lab, Tsinghua University, Beijing, 100084 [2]Department of Electronic Engineering, Tsinghua University, Beijing, 100084. Correspondence to: Jiaxin Shi <shijx15@mails.tsinghua.edu.cn>, Shengyang Sun <ssy13@tsinghua.org.cn>, Jun Zhu <dcszj@tsinghua.edu.cn>.

ICML 2017 Workshop on Implicit Models. Copyright 2017 by the author(s).

Recently, inspired by the probabilistic interpretation of Generative Adversarial Networks (Goodfellow et al., 2014; Mohamed & Lakshminarayanan, 2016), there has been some works that extend the adversarial learning approach to the posterior inference of latent variable models (Mescheder et al., 2017; Huszár, 2017; Tran et al., 2017). These methods all use an implicit variational family and thus can be categorized into Implicit VI methods. One of the key observations in them is that the density ratio $q_\phi(\mathbf{z})/p(\mathbf{z})$ can be estimated from samples of the two distributions by using a probabilistic classifier. They first assign class labels (y) to q and p: Let samples from $q_\phi(\mathbf{z})$ be of class $y = 1$, and samples from $p(\mathbf{z})$ be of class $y = 0$. Given equal class priors, the density ratio at a given point can be calculated as $q_\phi(\mathbf{z})/p(\mathbf{z}) = p(\mathbf{z}|y = 1)/p(\mathbf{z}|y = 0) = p(y = 1|\mathbf{z})/p(y = 0|\mathbf{z})$, which is the ratio between the class probabilities given the data point. To estimate this, a probabilistic classifier D is trained to classify between the two classes, with a logistic loss:

$$\max_D \mathbb{E}_{q_\phi(\mathbf{z})} \log\left(D(\mathbf{z})\right) + \mathbb{E}_{p(\mathbf{z})} \log\left(1 - D(\mathbf{z})\right), \tag{2}$$

where $D(\mathbf{z})$ is a classifier that outputs the probability of $\mathbf{z}$'s being from class $y = 1$. Given D is flexible enough, the optimal solution of problem (2) is $D(\mathbf{z}) = q_\phi(\mathbf{z})/(q_\phi(\mathbf{z}) + p(\mathbf{z}))$. Therefore, the KL divergence term in the ELBO of Eq. (1) can be approximated as $\mathrm{KL}(q_\phi\|p) \approx \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log D(\mathbf{z}) - \log(1 - D(\mathbf{z}))\right]$. This is called prior-contrastive forms of VI in (Huszár, 2017). Note that the ratio approximation doesn't change the gradients once the approximation is accurate (Huszár, 2017). Gradients are computed using the reparameterization trick (Kingma & Welling, 2013). Though incorporating the discriminative power in a probabilistic model has shown great success in GANs, this approach still suffers from several challenging problems when applied to variational inference:

- **Noisy density ratio estimation** In variational inference, the variational posterior gets updated in each iteration. To produce accurate estimation, the classifier should be trained to optimum after each iteration. However, in practice the classifier is only trained for one or several iterations for each variational update. The resulting unstable loss often produces noisy gradients and leads to unsatisfying results. Besides, even if the classifier quickly achieves the optimum, there is still another issue. Notice that the training loss in problem (2) is with expectations. But in practice we are using samples from the two distributions to approximate it. When the support of the distributions is high-dimensional, given the limited number of samples we use, the variance of this estimate is considerable, i.e., the classifier may overfit the samples. The phenomenon is that the classifier achieves a state where samples are easily distinguished and the probabilities given by the classifier are near 0 or 1, which is commonly observed in experiments (Mescheder et al., 2017).
- **High dimensional latent variables** As the density ratio is estimated by a classifier, the samples of latent variables from the two distributions should be fed into it. However, the typically used neural network classifier cannot scale towards very high-dimensional inputs (e.g., moderate-size Bayesian neural networks).

3. Kernel Density Ratio Fitting

As previously mentioned, the key problem of the noisy estimation is partly due to crude truncation of the inner loop after each variational update. Based on the intuition, we apply a kernel density ratio fitting (KDRF) method similar to the unconstrained Least Square Importance Fitting (uLSIF) (Kanamori et al., 2009). Specifically, let the true density ratio be $r(\mathbf{z}) = q(\mathbf{z})/p(\mathbf{z})$. Consider modeling the density ratio by the linear model: $\hat{r}(\mathbf{z}) = \boldsymbol{\alpha}^T\boldsymbol{\psi}(\mathbf{z}) = \sum_{k=1}^K \alpha_k\psi_k(\mathbf{z})$, where $\psi_k, k \in \{1, 2, \ldots, K\}$ are the basis functions and α_k are the combination weights. A common choice is to set the basis functions as RBF kernels around the training data points. In practice, to keep the algorithm efficient, we randomly select a subset of data points from the samples of q as kernel bases. To estimate the true ratio, the following squared distance is minimized

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\int (\hat{r}(\mathbf{z}) - r(\mathbf{z}))^2 p(\mathbf{z}) d\mathbf{z} = \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha} - \mathbf{h}^T\boldsymbol{\alpha} + C \tag{3}$$

where $H = \mathbb{E}_p\left[\boldsymbol{\psi}(\mathbf{z})\boldsymbol{\psi}(\mathbf{z})^T\right]$, and $\mathbf{h} = \mathbb{E}_q\left[\boldsymbol{\psi}(\mathbf{z})\right]$. The expectation can be estimated directly by Monte Carlo integration. We denote them as $\hat{H}, \hat{h}$. Adding a quadratic regularization term $\frac{\lambda}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}$ to the above loss and removing the constant, we get the final objective

$$\min_\alpha \frac{1}{2}\,\boldsymbol{\alpha}^T\hat{H}\boldsymbol{\alpha} + \hat{\mathbf{h}}^T\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}. \tag{4}$$

We can get the closed-form solution as: $\boldsymbol{\alpha}^* = (\hat{H} + \lambda I)^{-1}\hat{\mathbf{h}}$. Note that there should be a constraint that the estimated density ratio should be non-negative. However, we do not involve it in the optimization objective in order to get a closed-form solution.

There are two key differences between the method we use and the original uLSIF. The first is on how we deal with the non-negative constraint of density ratios. Because no explicit constraint is included in the loss, some post-processing must be done to ensure the estimated ratios are non-negative. In uLSIF, all αs are clipped to be not less than zero. However, we argue that reserving some negative coefficients are essential to obtain smooth approximations, and the clipping on coefficients often leads to dramatical increase in some high density regions, resulting in over-estimates of the density ratio. We solve this issue by replacing the clipping on α with clipping on the estimated density ratio. The clipping values are searched from $\{10^{-8}, 10^{-16}, 10^{-32}\}$. The second difference is on the way of determining the kernel bandwidth. The original uLSIF uses cross-validation while we apply a heuristic approach that the kernel bandwidth is chosen to be the median of all distances between the training data points and the chosen basis, which turns out to work very well in practice.

Another trick is essential to get an accurate estimate of the KL divergence. We observe that when applying the above methods to estimate $\frac{q}{p}$, the optimization objective in Eq. 3 puts more weights into places where the probability mass of p is high. However, $\mathrm{KL}(q\|p)$ puts more weights into places where the probability mass of q is high. Unless p and q match very well in where they put most probabilities, the ratio estimation objective does not fit well with the KL-divergence targets. The solution is by a simple trick. Instead of estimating $\frac{q}{p}$, we choose to estimate $\frac{p}{q}$ and compute $\mathrm{KL}(q\|p)$ as $-\mathbb{E}_q \log \frac{p}{q}$. This trick is found very essential in experiments to make the estimation accurate enough for variational inference.

Here we explain how this method addresses the two challenges stated in Section 2. First, the ratio estimates are given in closed-forms, thus not having the problem of not catching up. Second, the bias/variance trade-off of the estimation can be controlled by the regularization coefficient (λ). When λ is set smaller, the estimation is more aggressive to match the samples. When λ is set larger, the estimated ratio function is smoother. Choosing the appropriate λ, the variance of estimation can be controlled while maintaining a reasonably good fit, compared to the extreme ratio estimates given by classifiers when their probabilities are near 0 or 1.

4. Implicit Variational Bayesian Neural Networks

In a Bayesian neural network, given input $\mathbf{x}$, the output y is modeled with

$$\mathbf{W} \sim \mathcal{N}(\mathbf{W}|\mathbf{0},\mathbf{I})\,; \hat{y} = f_{NN}(\mathbf{x},\mathbf{W})\,; y \sim \mathcal{P}(\hat{y};\theta) \quad (5)$$

That is, $\hat{y}$ is the output of the feed forward network. And the final output y is of a distribution $\mathcal{P}$ parameterized by $\hat{y}$ and θ. For regression, $\mathcal{P}$ is usually a Gaussian with $\hat{y}$ as the mean. For classification, $\mathcal{P}$ is usually a discrete distribution with $\hat{y}$ as the unnormalized log probabilities.

We use variational inference to approximate the true posterior. The variational posterior is usually set to be factorized by layer: $q\left(\{\mathbf{W}_l\}_{l=1}^L\right) = \prod_{l=1}^L q_l(\mathbf{W}_l)$. Previous methods use a variational posterior with limited capacity, including mean-field Gaussian (Hernandez-Lobato & Adams, 2015), matrix variate Gaussian (Louizos & Welling, 2016; Sun et al., 2017), normalizing flows (Louizos & Welling, 2017). Enabled to learn implicit variational posteriors, we propose to adopt a general distribution without explicit likelihood.

In a neural network, each layer's parameter $\mathbf{W}$ is very high dimensional. Therefore, we often cannot directly use a fully connected neural network to model the distribution of $\mathbf{W}$. Drawing inspiration from low-rank matrix factorization (Koren et al., 2009), we propose a new kind of network called *Matrix Multiplication Neural Network* (MMNN) to model the implicit distribution of a 2-D matrix, as shown in Alg. 1. In each layer of a MMNN, for input matrix ($M_{in} \times N_{in}$), we left multiply a parameter matrix ($M_{out} \times M_{in}$) and add a bias matrix, then we right multiply a parameter matrix ($N_{in} \times N_{out}$) and add a bias matrix. Finally it is passed through a nonlinear activation function such as Relu. We call such a layer as Matrix Multiplication Layer of size $[M_{out}, N_{out}]$.

Algorithm 1 Matrix Multiplication Neural Network

```
Require: Input matrix $X_0$
Require: network parameters $\{W_i^l\}_{i=1}^L$, $\{B_i^l\}_{i=1}^L$,
    $\{W_i^r\}_{i=1}^L$, $\{B_i^r\}_{i=1}^L$
1: $i = 1$
2: while $i \leq L$ do
3:   left multiplication: $X_i = W_i^l X_{i-1} + B_i^l$
4:   right multiplication: $X_i = X_i W_i^r + B_i^r$
5:   if $i \leq L - 1$ then
6:     $X_i = Relu\,(X_i)$
7:   end if
8: end while
9: Output $X_L$
```

When modeling a matrix, MMNN has significant computational advantages over fully connected networks. Due to its low-rank property, MMNN easily scales with matrix size. To model a $M \times N$ matrix, consider a one-layer network whose input shape is $[M_0, N_0]$. For fully connected structure, the parameter matrix's shape is $[M_0 N_0, MN]$. While for MMNN structure, we only need two matrices of shape $[M, M_0]$ and two matrices of shape $[N_0, N]$.

To model the implicit distribution of $\mathbf{W}_l$, we only need to randomly sample a matrix $\mathbf{W}_l^0$ of smaller size $[M_0, N_0]$, and feed it forward through the MMNN to get the output matrix samples.

$$\mathbf{W}_l^0 \sim N(\mathbf{0},\mathbf{I})\,, \quad q(\mathbf{W}_l) = \mathcal{MMNN}_{\phi_l}.\left(\mathbf{W}_l^0\right) \quad (6)$$

Thus, we can use a MMNN to model the variational posterior of each layer's parameter $\mathbf{W}_l$ for large scale networks. However, in tasks with small networks, we still use the fully connected network to model the variational posterior.

5. Experiments

Below we present the experiments of Bayesian neural nets. We leave results on synthetic datasets to the appendix (A).

Table 1: Average test set RMSE, predictive log-likelihood for the regression datasets.

	Avg. Test RMSE			Avg. Test LL		
Dataset	SVGD	Dropout	Implicit VI (KDRF)	SVGD	Dropout	Implicit VI (KDRF)
Boston	2.957 ± 0.099	2.97 ±0.19	**2.875±0.133**	-2.504 ± 0.029	**-2.46 ±0.06**	-2.598±0.086
Concrete	5.324 ± 0.104	5.23 ±0.12	**4.881±0.124**	-3.082 ± 0.018	**-3.04 ±0.02**	-3.116±0.047
Energy	1.374 ± 0.045	1.66 ±0.04	**0.583±0.019**	-1.767 ± 0.024	-1.99 ±0.02	**-1.339±0.006**
Kin8nm	0.090 ± 0.001	0.10 ±0.00	**0.075±0.001**	0.984 ± 0.008	0.95 ±0.01	**1.162±0.008**
Naval	0.004 ± 0.000	0.01 ±0.00	**0.001±0.000**	4.089 ± 0.012	3.80 ±0.01	**5.434±0.167**
Combined	4.033 ± 0.033	4.02 ±0.04	**4.010±0.035**	-2.815 ± 0.008	**-2.80 ±0.01**	**-2.800±0.008**
Protein	4.606 ± 0.013	4.36 ±0.01	**4.320±0.012**	-2.947 ± 0.003	-2.89 ±0.00	**-2.883±0.003**
Wine	**0.609 ± 0.010**	0.62 ±0.01	0.637±0.007	**-0.925 ± 0.014**	-0.93 ±0.01	-0.962±0.011
Yacht	0.864 ± 0.052	1.11 ±0.09	**0.654±0.058**	**-1.225 ± 0.042**	-1.55 ±0.03	-2.084±0.006
Year	**8.684 ± NA**	8.849 ±NA	9.280±NA	**-3.580 ± NA**	-3.588 ± NA	-3.648±NA

5.1. Multivariate Regression

To illustrate the predicative ability, we compare the multivariate regression results for several public datasets with stein variational gradient descent (SVGD) (Liu & Wang, 2016) and dropout uncertainty (Gal & Ghahramani, 2016). We follow the settings in probabilistic backpropagation (Hernandez-Lobato & Adams, 2015). For information about our experiment and variational posterior network, see D.1.

The results are shown in Table 1. For each dataset, the best result is shown as bold. We can see that Implicit VI (KDRF) consistently outperforms SVGD and dropout in both RMSE and test log likelihood for most datasets. Especially in RMSE, Implicit VI (KDRF) has obvious improvements compared to SVGD and dropout except on *Wine* and *Year Predication MSD*. That is mainly attributed to that Implicit VI (KDRF) captures more about the complex parameter correlations. Note that matrix variate Gaussian prior (Louizos & Welling, 2016; Sun et al., 2017) is different from our factorized prior; the former also additionally used variational dropout, thus their results are not comparable to ours.

Recently, normalizing flows are shown to have good performance on Bayesian neural networks (Louizos & Welling, 2017). Being interested in this, we also experiment with directly applying normalizing flows on this task. See Appendix B for details.

5.2. MNIST Classification

Table 2: MNIST Classification error rate

hidden	Bayes by Backprop, Gaussian	Implicit VI (KDRF)
400	1.82%	**1.68%**
800	1.99%	**1.56%**
1200	2.04%	**1.73%**

In this part we compare the classification results on MNIST datasets, which has been worked on by many methods. However, many of them use a different prior, such as structured prior (Louizos & Welling, 2016), scale mixture prior (Blundell et al., 2015). Therefore we only compare the classification results with Bayes by Backprop (Gaussian) (Blundell et al., 2015). We use neural network with 2 hidden layers. For detailed experiment setting, see D.2. The results are shown in 2. As seen, with the more expressive capacity of our implicit variational posterior, Implicit VI (KDRF) show superior classification results compared to the mean field variational posterior.

To compare with prior-contrastive methods, we also observe the training lower bound with time going. As the posterior is extremely high-dimensional, only simple logistic regression is possible for being a discriminator. So we use logistic regression as the discriminator for prior-contrastive methods. Lower bounds of the two methods increase in the same pace at first, then prior-contrastive fails to converge with lower bound explosion while Implicit VI (KDRF) improves consistently. The explosion is mainly because the input to the discriminator is of hundreds of thousands of dimensions for this Bayesian neural network. Plain discriminator cannot handle with such high-dimensional inputs. We also experiment with prior-contrastive for layer size 800 and 1200. They both fail to converge at end.

ACKNOWLEDGMENTS

The work was supported by the National Basic Research Program (973 Program) of China (No. 2013CB329403), NSFC Projects (Nos. 61620106010, 61621136008), and the Youth Top-notch Talent Support Program. All implementations are based on ZhuSuan `https://github.com/thu-ml/zhusuan`.

References

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

Dai, Bo, He, Niao, Dai, Hanjun, and Song, Le. Provable bayesian inference via particle mirror descent. *arXiv preprint arXiv:1506.03101*, 2015.

Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Dumoulin, Vincent, Belghazi, Ishmael, Poole, Ben, Lamb, Alex, Arjovsky, Martin, Mastropietro, Olivier, and Courville, Aaron. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Ghahramani, Zoubin. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Hernandez-Lobato, Jose Miguel and Adams, Ryan. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John William. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Huszár, Ferenc. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Kanamori, Takafumi, Hido, Shohei, and Sugiyama, Masashi. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, Diederik P, Salimans, Tim, Jozefowicz, Rafal, Chen, Xi, Sutskever, Ilya, and Welling, Max. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Koren, Yehuda, Bell, Robert, and Volinsky, Chris. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Liu, Qiang and Feng, Yihao. Two methods for wild variational inference. *arXiv preprint arXiv:1612.00081*, 2016.

Liu, Qiang and Wang, Dilin. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2370–2378, 2016.

Louizos, Christos and Welling, Max. Structured and efficient variational deep learning with matrix gaussian posteriors. *arXiv preprint arXiv:1603.04733*, 2016.

Louizos, Christos and Welling, Max. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

Mescheder, Lars, Nowozin, Sebastian, and Geiger, Andreas. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722*, 2017.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Mohamed, Shakir and Lakshminarayanan, Balaji. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Neal, Radford M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Paisley, John, Blei, David, and Jordan, Michael. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Ranganath, Rajesh, Tran, Dustin, Altosaar, Jaan, and Blei, David. Operator variational inference. In *Advances in Neural Information Processing Systems*, pp. 496–504, 2016.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Sun, Shengyang, Chen, Changyou, and Carin, Lawrence. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

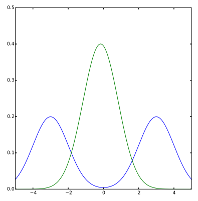
Tomczak, Jakub M and Welling, Max. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

Tran, Dustin, Ranganath, Rajesh, and Blei, David M. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*, 2017.

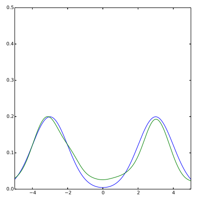
Uehara, Masatoshi, Sato, Issei, Suzuki, Masahiro, Nakayama, Kotaro, and Matsuo, Yutaka. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.

A. Experiments on Synthetic Data

A.1. Toy 1-D Gaussian Mixtures



(a) Mean-field VI



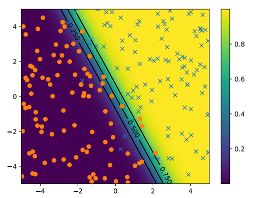
(b) Implicit VI (KDRF)

Figure 1: Fitting Gaussian Mixture distribution

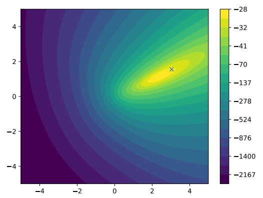
We firstly conduct a toy experiment to approximate a 1-D Gaussian mixture distribution with variational inference. The Gaussian mixture distribution has two equally distributed unit-variance components whose means are -3 and 3. We compare the results of Implicit VI (KDRF) with mean-field VI in Fig. 1. For Implicit VI (KDRF), we forward random samples from a standard normal distribution through a two-layer fully connected neural network with 50 hidden units and one output unit.

As shown in Fig. 1a, the mean-field posterior converges to the middle of the two modes, where probability mass is small. In contrast, Implicit VI (KDRF) can accurately approximate the Gaussian mixture distribution with expressive variational posterior.

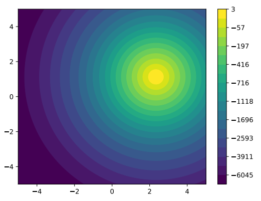
A.2. 2-D Bayesian Logistic Regression



(a) Training data



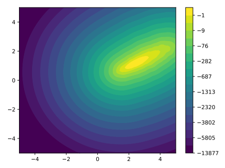
(b) Unnormalized true posterior



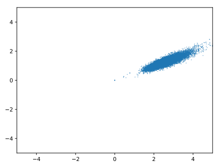
(c) Mean-field VI

Figure 2: 2-D Bayesian logistic regression

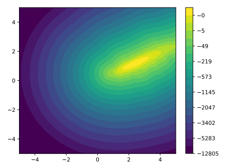
We also conduct experiments on a 2-D Bayesian logistic regression example, which has an intractable posterior. The



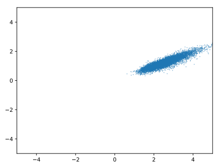
(a) HMC (KDE)



(b) HMC (samples)



(c) implicit (KDE)



(d) implicit (samples)

Figure 3: HMC vs. Implicit VI in 2-D Bayesian logistic regression

model is

$$\mathbf{w} \sim N(\mathbf{0}, \mathbf{I}), \quad y_i \sim \text{Bernoulli}(\sigma(\mathbf{w}^T \mathbf{x}_i)), \quad i = 1, \ldots, N \tag{7}$$

where $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^2$; σ is the sigmoid function. We generate $N = 200$ data points ($\{(x_i, y_i)\}_{i=1}^{200}$) from the true model as the training data (Fig. 2a). The unnormalized true posterior is plotted in Fig. 2b. As a baseline, we first run mean-field variational inference to do posterior inference. The result is shown in Fig. 2c. It can be clearly seen that the mean-field VI can capture the position and the scale of the posterior. But due to its independence property across dimensions, it cannot fit well to the shape of the true posterior.

We then apply our Implicit VI method. The implicit posterior we use is a simple stochastic neural network. To see how good the result is, we also run Hamiltonian Monte Carlo (HMC) to get posterior samples. The results are plotted in Fig. 3. We can see that the implicit posterior are learned to capture the strong correlation between the two dimensions and can produce posterior samples that have a similar shape with samples drawn by HMC.

B. Comparison with Normalizing Flows

The results are reported in Table 3. We apply 10 planar normalizing flows (Rezende & Mohamed, 2015) on the weights to match the computation time of our methods. However, the results of normalizing flows do not improve over mean-field VI in this task.

C. Related Work

There are three lines of works closely related.

Table 3: Test RMSE, log-likelihood for the regression datasets. VI and NF represent mean-field VI and normalizing flow.

RMSE	VI	NF	Implicit VI (KDRF)
boston	3.42±0.19	3.43±0.19	**2.88±0.13**
concrete	6.00±0.10	6.04±0.10	**4.88±0.12**
energy	2.42±0.06	2.48±0.09	**0.58±0.02**
kin8nm	0.09±0.00	0.09±0.00	**0.08±0.00**
naval	0.01±0.00	0.01±0.00	**0.00±0.00**
LL	VI	NF	Implicit VI (KDRF)
boston	-2.66±0.04	-2.66±0.04	**-2.60±0.09**
concrete	-3.22±0.06	-3.24±0.06	**-3.12±0.05**
energy	-2.34±0.02	-2.36±0.03	**-1.34±0.01**
kin8nm	0.96±0.01	1.01±0.01	**1.16±0.01**
naval	4.00±0.11	4.04±0.12	**5.43±0.17**

Implicit generative models Implicit generative models (generative models that define implicit distributions) have drawn much attention these days due to the popularity of Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). General learning algorithms of implicit models have been surveyed in (Mohamed & Lakshminarayanan, 2016), of which density ratio estimation plays the central role. The connection between density ratio estimation and GANs is also discussed in (Uehara et al., 2016).

Variational inference Our work builds upon the recent developments of variational inference, including stochastic optimization by mini-batches (Hoffman et al., 2013), direct gradient optimization of variational lower bounds (Paisley et al., 2012; Mnih & Gregor, 2014), and the reparametrization trick for continuous latent variable models (Kingma & Welling, 2013). Following the success of learning with implicit generative models, implicit distributions are applied to variational inference, which are surveyed in (Huszár, 2017). This paper divides implicit variational inference into two categories: prior-contrastive and joint-contrastive. Classifiers in prior-contrastive methods distinguish between samples from the prior and the variational posterior, while in joint-contrastive methods it distinguish between the model joint distribution and the joint distribution composed of data distribution and variational posteriors. Concurrent with (Huszár, 2017), authors of (Mescheder et al., 2017) propose Adversarial Variational Bayes, which is an amortized version of prior-contrastive methods for training local latent-variable models like VAEs (Kingma & Welling, 2013). Prior to (Huszár, 2017), similar ideas with joint-contrastive methods have been proposed in ALI and Bi-GAN (Dumoulin et al., 2016; Donahue et al., 2016). Nonparametric methods for variational inference (Liu & Wang, 2016; Dai et al., 2015) that adapt a set of particles towards the true posterior are also closely related to implicit variational inference. They share the similar advantage of flexible approximations. As previously mentioned, there is also another line of works on flows (Rezende & Mohamed, 2015; Kingma et al., 2016; Tomczak & Welling, 2016) that tries to improve the expressiveness of the variational posterior with invertible transformations.

Bayesian neural networks There are many recent advances in inference algorithms for Bayesian neural networks. Probabilistic back-propagation (Hernandez-Lobato & Adams, 2015), structured uncertainty(Sun et al., 2017) is based on Assumed Density Filtering. Bayes by backprop (Blundell et al., 2015), dropout uncertainty (Gal & Ghahramani, 2016), and variational matrix Gaussian (Louizos & Welling, 2016) build upon variational inference.

D. Experiment Settings and Networks for Modeling Variational Posteriors

D.1. Multivariate Regression

Following the settings in probabilistic backpropagation (Hernandez-Lobato & Adams, 2015), we randomly select 90% of the whole dataset for training and use the rest for testing and use a Bayesian neural network of one hidden layer. For the two big datasets, i.e., *Protein Structure* and *Year Predication MSD*, the hidden layer is with 100 units. For the rest datasets, the hidden layer is with 50 units. We also put a Gamma prior on the output precision. We run 20 times and report the mean and std errors of test performances, except 5 times for *Protein Structure* and only 1 time for *Year Predication MSD*.

As the regression datasets have small feature dimensions (usually less than 15, except 90 for *year*), using Bayesian neural networks of one hidden layer (50 units) doesn't produce very high-dimensional weights. Therefore we still use fully connected neural networks in the variational posterior. For the first layer parameter ($[50, n_{in}+1]$), we use two hidden layers of size 1024 and 50 $(n_{in}+1)$ with Relu activations. The input to it is random samples of length 100 from a standard normal distribution. For the second layer parameter ($[1, 51]$), we use two hidden layers of size 100, 51. The input to it is random samples of length 50 from a standard normal distribution.

D.2. MNIST Classification

We use feed forward neural network with two Relu hidden layers and experiment on layer size 400, 800, 1200. We use randomly set learning rate 0.001. In both training and testing, we use 10 random samples, and batch size 100 (For layer size 1200, because of memory problem we use batch size 40). We randomly select 10000 data points in training set as validation set. Training with the left 50000 data points, we select the model with best validation result for final testing.

MNIST classification needs a larger scale network than the one used in multivariate regression. Therefore we use a Matrix Multiplication Neural Network for modeling the variational posterior. We experimented with hidden layer size 400, 800 and 1200, denoted by L. Below we set $N = 500$ when $L = 400$, otherwise $N = 800$. In the variational posterior network, we use two matrix multiplication hidden layers.

For the first layer parameter ($[785, L]$), the two hidden matrix multiplication layers are of size $[N, N]$ and $[N, N]$ with Relu activations. The final output layer is of size $[L, 785]$ with linear activations. The input matrix is random samples of size $[30, 30]$ from a standard normal distribution. To be noted, each MM layer represents a left multiply, a right multiply and two sums.

For the second layer parameter ($[L, L + 1]$), the two hidden matrix multiplication layers are of size $[N, N]$ and $[N, N]$ with Relu activations. The final output layer is of size $[L, L + 1]$ with linear activations. The input matrix is random samples of size $[30, 30]$ from a standard normal distribution.

For the third layer parameter ($[10, L + 1]$), the two hidden matrix multiplication layers are of size $[30, N]$ and $[30, N]$ with Relu activations. The final output layer is of size $[10, L + 1]$ with linear activations. The input matrix is random samples of size $[30, 30]$ from a standard normal distribution.

E. Lower bound with Gamma-Prior Precision

In the multivariate regression task, the output is sampled from a normal distribution with $\hat{y}(x, \mathbf{W})$ as mean and a parameter as variance. The variance controls the likelihood of the model, therefore, choosing an appropriate variance is essential. Therefore we place a Gamma prior $Ga(6, 6)$ on its reciprocal (i.e., the precision of the Normal distribution). The variational posterior we used is $q(\mathbf{W}, \lambda) = q(\mathbf{W})\, q(\lambda)$. Then the ELBO can be computed as

$$\begin{aligned}
\mathcal{L} =& \mathrm{E}_{q(\mathbf{W})}\mathrm{E}_{q(\lambda)} \log p(y|\mathbf{x}, \mathbf{W}, \lambda) \\
&- \mathrm{KL}(q(\mathbf{W}) \parallel p(\mathbf{W})) - \mathrm{KL}(q(\lambda) \parallel p(\lambda)) \\
=& \frac{1}{2}\mathrm{E}_{q(\mathbf{W})}\mathrm{E}_{q(\lambda)} \left[\log \lambda - \lambda (y - \hat{y}(\mathbf{x}, \mathbf{W}))^2 - \log 2\pi\right] \\
&- \mathrm{KL}(q(\mathbf{W}) \parallel p(\mathbf{W})) - \mathrm{KL}(q(\lambda) \parallel p(\lambda)) \\
=& \frac{1}{2}\mathrm{E}_{q(\mathbf{W})} \left[\psi(\alpha) - \log \beta - \frac{\alpha}{\beta}(y - \hat{y}(\mathbf{x}, \mathbf{W}))^2 - \log 2\pi\right] \\
&- \mathrm{KL}(q(\mathbf{W}) \parallel p(\mathbf{W})) - \mathrm{KL}(q(\lambda) \parallel p(\lambda)) .
\end{aligned}$$

Where $\psi(x)$ is the digamma function and the KL divergences be calculated in closed-form.