KERNEL IMPLICIT VARIATIONAL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent progress in variational inference has paid much attention to the flexibility of variational posteriors. One promising direction is to use implicit distributions, i.e., distributions without tractable densities 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 over-fitting (Ghahramani, 2015). As it becomes a common practice to build deep models that have many parameters (LeCun et al., 2015), it is even more important to have a Bayesian formulation to protect these models. For example, Bayesian Neural Networks (BNNs) (Neal, 2012; Blundell et al., 2015) have shown promise in dealing with model uncertainty and learning with few labeled data. Another recent trend is to incorporate deep neural networks as a powerful function mapping between random variables in a Bayesian network, such as deep generative models like variational autoencoders (VAE) (Kingma & Welling, 2013).

Except a few simple examples, Bayesian inference is typically challenging, for which variational inference (VI) has been a standard workhorse to approximate the true posterior (Zhu et al., 2017). Traditional VI focuses on factorized variational posteriors to get analytical updates. While recent progress in this field drives VI 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, factorized 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 (Rezende & Mohamed, 2015; Kingma et al., 2016; Tomczak & Welling, 2016). In their works, it is important for the transformation to be invertible to ensure that the transformed distribution has a tractable density and can be used as the variational posterior.

Although utilizing 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 we get in this way 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 approach named *Kernel Implicit Variational Inference* (KIVI), which addresses the noisy estimation and scalability problems of implicit VI. KIVI is applicable to both global and local latent variable models, which is demonstrated by experiments on BNNs and VAEs. As far as we know, this is the first time that implicit VI is successfully applied to BNNs, which shows promising results on both regression and classification tasks.

2 BACKGROUND

Consider a generative model $p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, where $\mathbf{x}$ and $\mathbf{z}$ denotes observed and latent variables, respectively. In VI, a variational distribution $q_\phi(\mathbf{z})$ in some parametric family is chosen to approximate the true posterior $p(\mathbf{z}|\mathbf{x})$ by optimizing the *evidence lower bound* (ELBO):

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

where KL denotes the Kullback-Leibler divergence $\mathrm{KL}(q\|p) = \mathbb{E}_q \left[\log \frac{q}{p}\right]$. This objective is a lower bound of the log-likelihood $\log p(\mathbf{x})$ since it can be written as $\mathcal{L}(\mathbf{x}; \phi) = \log p(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x}))$. The maximum of this objective is achieved when $q_\phi(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$. From Eq. (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.

Recently, inspired by the probabilistic interpretation of Generative Adversarial Networks (GAN) (Goodfellow et al., 2014; Mohamed & Lakshminarayanan, 2016), there has been some works that extend the GAN approach to the posterior inference of latent variable models (LVMs) (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 their key observations is that the density ratio $\frac{q_\phi(\mathbf{z})}{p(\mathbf{z})}$ 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:

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

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). Though incorporating the discriminative power in a probabilistic model has shown great success in GANs, this method still suffers from challenging problems when applied to VI:

- **Noisy density ratio estimation (DRE)** In VI, 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. It is hard for the classifier to catch up with the variational posterior. This unstable loss often produces noisy gradients and leads to unsatisfying results. Besides, even if the classifier quickly achieves the optimum in a small number of iterations, 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 from the two distributions of latent variables should be fed into it. However, the typically used neural network classifier cannot scale towards very high-dimensional inputs (e.g., parameters in a moderate-size Bayesian neural networks).

3 KERNEL IMPLICIT VARIATIONAL INFERENCE

To address the above challenges for Implicit VI, we propose to replace the probabilistic classifier with a kernel method for DRE. When this approach is applied to Bayesian neural networks, we further design a highly correlated implicit variational family that can scale to very large networks. Specifically, the key problem of the noisy estimation is partly due to the crude truncation of the inner

loop after each variational update. Then if a method for estimating the KL term can give closed-form solutions while maintaining reasonable accuracy, it should enable an algorithm without inner loops, which is more stable and potentially leads to better performance. Based on the intuition, we apply a kernel regression approach to estimate the density ratio function. It has a closed-form solution that is guaranteed to be globally optimal.

3.1 ESTIMATING THE KL TERM

Specifically, let $\mathbf{z} \in \mathbb{R}^d$ be the latent variable, and the true density ratio is $r(\mathbf{z}) = q(\mathbf{z})/p(\mathbf{z})$. Consider modeling it with a function $\hat{r} \in \mathcal{H}$, where $\mathcal{H}$ is a *Reproducing Kernel Hilbert Space* (RKHS) induced by a positive definite kernel $k(\mathbf{z}, \mathbf{z}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Similar to kernel ridge regression, we use an objective composed of a square loss for regression plus a penalty for the complexity of the function, where the complexity is measured by the RKHS norm ($\|\hat{r}\|^2_{\mathcal{H}}$):

$$\min_{\hat{r} \in \mathcal{H}} \ \mathcal{L}(\hat{r}) + \frac{\lambda}{2}\|\hat{r}\|^2_{\mathcal{H}}. \tag{3}$$

Here λ is the regularization coefficient. For the squared loss we choose the form used by the unconstrained Least Square Importance Fitting (uLSIF) (Kanamori et al., 2009):

$$\mathcal{J}(\hat{r}) = \frac{1}{2}\int (\hat{r}(\mathbf{z}) - r(\mathbf{z}))^2 p(\mathbf{z})\ d\mathbf{z} = \frac{1}{2}\ \mathbb{E}_p \hat{r}(\mathbf{z})^2 - \mathbb{E}_q \hat{r}(\mathbf{z}) + C, \tag{4}$$

where C is a constant. Then $\mathcal{L}(\hat{r})$ is defined by the Monte Carlo estimate of $\mathcal{J}(\hat{r})$, using samples from p and q: $\mathcal{L}(\hat{r}) = \hat{\mathcal{J}}(\hat{r}) = \frac{1}{2n_p}\sum_{i=1}^{n_p} \hat{r}(\mathbf{z}_i^p)^2 - \frac{1}{n_q}\sum_{i=1}^{n_q} \hat{r}(\mathbf{z}_j^q) + C,\ \mathbf{z}_i^p \sim p(\mathbf{z}),\ \mathbf{z}_j^q \sim q(\mathbf{z})$.

Proposition 1. *The optimal solution of Eq. (3) lies in the linear subspace spanned by the kernel functions with the samples* $(\{\mathbf{z}_i^p\}_{i=1}^{n_p}, \{\mathbf{z}_j^q\}_{j=1}^{n_q})$ *as bases, i.e.,* $\hat{r}$ *has the form:*

$$\hat{r} = \sum_{i=1}^{n_p} \alpha_i k(\mathbf{z}_i^p, \cdot) + \sum_{j=1}^{n_q} \beta_j k(\mathbf{z}_j^q, \cdot). \tag{5}$$

Proof. This can be seen as the generalization of the representer theorem (Schölkopf et al., 2001) to the density ratio problem. So the proof follows the same procedure. See Appendix A. □

Substitute Eq. (5) into Eq. (3) and let the derivatives w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be zeros, we get the optimal solution (a detailed derivation is given in Appendix B):

$$\boldsymbol{\beta} = \frac{1}{\lambda n_q}\mathbf{1}, \quad \boldsymbol{\alpha} = -\frac{1}{\lambda n_p n_q}\left(\frac{1}{n_p}\mathbf{K}_p + \lambda \mathbf{I}\right)^{-1}\mathbf{K}_{pq}\mathbf{1}, \tag{6}$$

where $(\mathbf{K}_p)_{i,j} = k(\mathbf{z}_i^p, \mathbf{z}_j^p)$, $(\mathbf{K}_{pq})_{i,j} = k(\mathbf{z}_i^p, \mathbf{z}_j^q)$, and $(\mathbf{K}_q)_{i,j} = k(\mathbf{z}_i^q, \mathbf{z}_j^q)$. We use RBF kernels $k(\mathbf{z}, \mathbf{z}') = \exp\left(-\|\mathbf{z} - \mathbf{z}'\|_2^2 / 2\sigma^2\right)$. σ is the kernel bandwidth, which is determined by the commonly used median heuristic (the median of pairwise distances of the data points).

Now that we have the approximate density ratio function $\hat{r}$, a Monte Carlo estimate of the KL divergence term can be constructed by $\frac{1}{n_q}\sum_{i=1}^{n_q} \log \hat{r}(\mathbf{z}_i^q)$. Note that there is 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, which indicates that some post-processing should be done to ensure this property. We solve this issue by clipping the estimated density ratio. The clipping values are searched from $\{10^{-8}, 10^{-16}, 10^{-32}\}$. In experiments we found that the algorithm is not sensitive to the clipping value. This is due to the accurate estimation guaranteed by the global optimal in the RKHS, which is a universal family when RBF kernels are used (Carmeli et al., 2010).

The reverse ratio trick Another technique is essential to get an accurate estimate of the KL divergence, which we call the reverse ratio trick. The key observation is that when the above method is applied, the optimization objective in Eq. (3) puts more weights into regions where the probability mass of p is high. However, the KL term $\mathrm{KL}(q\|p)$ to estimate chooses q as the base measure. 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 the KL term as $-\mathbb{E}_q \log \frac{p}{q}$. We denote the estimated reverse

density ratio as $\hat{r}_{pq}$, then the corresponding KL estimate is $-\mathbb{E}_q \log \hat{r}_{pq}$. We find in experiments that this trick is very essential to make the estimation accurate enough for VI.

Gradient computation We now consider how to estimate the gradient of the KL term w.r.t. variational parameters ϕ. First it's easy to prove as in Huszár (2017) that

$$\nabla_\phi \mathrm{KL}(q_\phi \| p) = -\nabla_\phi \mathbb{E}_{q_\phi} \log \frac{p}{q_\phi} = -\nabla_\phi \mathbb{E}_{q_\phi} \log \frac{p}{q}. \tag{7}$$

We also include it in Appendix C. Eq.(7) indicates that we can use any approximation of the density ratio, and the gradients w.r.t. ϕ won't change as long as the approximation is accurate. We now replace the density ratio on the right side with $\hat{r}_{pq}$: $\nabla_\phi \mathrm{KL}(q_\phi \| p) \approx -\nabla_\phi \mathbb{E}_{q_\phi} \log \hat{r}_{pq}$. Then, the reparameterization trick (Kingma & Welling, 2013) can be used:

$$-\nabla_\phi \mathbb{E}_{q_\phi} \log \hat{r}_{pq} = -\mathbb{E}_{\epsilon \sim N(0,I)} \nabla \log \hat{r}_{pq}(\mathbf{z}^q(\epsilon; \phi)). \tag{8}$$

3.2 The Algorithm

We have constructed a closed-form estimate for the KL term and show its gradients can be estimated by the reparameterization trick. Note that the reparameterization trick can also be used to compute the gradients of the reconstruction term in Eq.(1) and thus can be applied to the ELBO. See Algo. 1 for the complete algorithm. Note that the number of samples (M) used in the reconstruction term is independent with those required for the KL estimation, which can be reduced when the model is expensive (e.g., we use only $M = 1$ in the experiments of VAEs). Thus the computational cost added compared to normal reparameterized VI is mainly the matrix inversion of size n_p in Eq.(6). In experiments we found tens or a hundred samples are enough to obtain an accurate KL estimate, so the added cost is not high.

KIVI addresses the two challenges stated in Sec. 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. Moreover, KIVI is directly applicable to both global and local latent variable models (LVMs), which is an advantage over nonparametric VI methods like particle mirror descent (Dai et al., 2015) and stein variational gradient descent (Liu & Wang, 2016). For the task of training local LVMs like VAEs, we we additionally use the adaptive contrast (AC) technique (Mescheder et al., 2017), whose details are summarized in Appendix D.

Algorithm 1 Kernel Implicit Variational Inference (KIVI)

Require: Observed data $\mathbf{x}$, model $p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$
Require: Implicit variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$, n_p, n_q, M.

1: **repeat**
2: Sample from prior: $\mathbf{z}_i^p \sim p(\mathbf{z}),\ i = 1, \dots, n_p$
3: Sample from variational: $\mathbf{z}_j^q \sim q(\mathbf{z}|\mathbf{x}),\ j = 1, \dots, n_q$
4: Compute the density ratio $\hat{r}_{pq}$ by Eq. (5), (6) and clip $\hat{r}_{pq}$ to be positive at $\mathbf{z}^q$s.
5: Compute $\hat{\mathrm{KL}} = -\frac{1}{n_q} \sum_{j=1}^{n_q} \log \hat{r}_{pq}(\mathbf{z}_j^q)$
6: Compute $\mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} p(\mathbf{x}|\mathbf{z}_m^q) - \hat{\mathrm{KL}}$ and the gradient $\nabla_\phi \mathcal{L}$ with the reparameterization trick.
7: Do gradient descent with $\nabla_\phi \mathcal{L}$
8: (Optional) For parameter learning, do gradient descent with $\nabla_\theta \mathcal{L}$
9: **until** Convergence

4 Example: Implicit Variational Bayesian Neural Networks

Now we present an example for using KIVI in Bayesian neural networks (BNNs), which has received increasing attention due to their ability to model uncertainty, an important factor in many tasks such as adversarial defense and reinforcement learning. However, it is nontrivial to apply implicit VI to BNNs because of the high-dimensional latent variables (weights). The recently developed implicit posteriors (Mescheder et al., 2017; Song et al., 2017) are not applicable. We present *Matrix Multiplication Neural Network* (MMNN), an efficient framework for sampling large matrices. Deploying MMNN, KIVI can easily scale up to large BNNs.

In BNNs, a prior is specified over the neural network parameters $\mathbf{W} = \{\mathbf{W}_l\}_{l=1}^L$, where $\mathbf{W}_l$ indicates weights in the l-th layer. Given input $\mathbf{x}$, the output y is modeled with

$$\mathbf{W} \sim N(\mathbf{0}, \mathbf{I}), \quad \hat{y} = f_{\mathrm{NN}}(\mathbf{x}, \mathbf{W}), \quad y \sim \mathcal{P}(\hat{y}; \theta), \tag{9}$$

where $\hat{y}$ is the output of the feed-forward network f_{NN}. And 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.

The true posterior of $\mathbf{W}$ in BNNs is intractable. Thus we turn to VI and use a variational posterior q to approximate it. Denoting the dataset with $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N, \mathbf{Y} = \{y_i\}_{i=1}^N$, we have the ELBO:

$$\mathcal{L}(\mathbf{Y}, \mathbf{X}; \phi) = \mathbb{E}_{q_\phi(\mathbf{W})} \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}) - \mathrm{KL}(q_\phi(\mathbf{W}) \| p(\mathbf{W})).$$

The variational posterior is usually set to be factorized by layer: $q_\phi(\mathbf{W}) = \prod_{l=1}^L q_{\phi_l}(\mathbf{W}_l)$. However, previous methods used variational posteriors with a limited capacity for each $q_{\phi_l}(\mathbf{W}_l)$, including factorized Gaussian (Hernandez-Lobato & Adams, 2015), matrix variate Gaussian (Louizos & Welling, 2016; Sun et al., 2017) and normalizing flows (Louizos & Welling, 2017). Enabled to learn implicit variational posterior, we propose to adopt a general distribution without an explicit density function, which has a form of

$$\mathbf{W}_l^0 \sim N(\mathbf{0}, \mathbf{I}), \quad q_{\phi_l}(\mathbf{W}_l) = g_{\phi_l}(\mathbf{W}_l^0). \tag{10}$$

Here g is a transform parameterized by ϕ_l. The key challenge here is that $\mathbf{W}_l$ are very high dimensional for moderate size neural networks. Thus, we often cannot use a fully connected neural network (MLP) as g here. Inspired by low-rank matrix factorization (Koren et al., 2009), we propose a new kind of network called *Matrix Multiplication Neural Network* (MMNN) to serve as the g function here, as shown in Alg. 2. In each layer of a MMNN, for input matrix $\mathbf{X}_0$ ($M_{in} \times N_{in}$), we left multiply a parameter matrix $\mathbf{W}_i^l$ ($M_{out} \times M_{in}$) and add a bias matrix $\mathbf{B}_i^l$ ($M_{out} \times N_{in}$), then we right multiply a parameter matrix ($N_{in} \times N_{out}$) and add a bias matrix ($M_{out} \times N_{out}$). Finally it is passed through a nonlinear activation function such as ReLU. We call such a layer as Matrix Multiplication Layer of size $M_{out} \times N_{out}$.

Algorithm 2 MMNN

Require: Input matrix $\mathbf{X}_0$
Require: network parameters $\{\mathbf{W}_i^l, \mathbf{B}_i^l, \mathbf{W}_i^r, \mathbf{B}_i^r\}_{i=1}^L$
for $i = 1 : L$ **do**
  left multiplication: $\mathbf{X}_i = \mathbf{W}_i^l \mathbf{X}_{i-1} + \mathbf{B}_i^l$
  right multiplication: $\mathbf{X}_i = \mathbf{X}_i \mathbf{W}_i^r + \mathbf{B}_i^r$
  if $i \leq L - 1$ **then**
    $\mathbf{X}_i = \mathrm{Relu}\,(\mathbf{X}_i)$
  end if
end for
Output $\mathbf{X}_L$

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

To model the implicit posterior of $\mathbf{W}_l$, we only need to randomly sample a matrix $\mathbf{W}_l^0$ of smaller size $M_0 \times N_0$, and feed it forward through the MMNN to get the output variational samples ($\mathbf{W}_l^q$):

$$\mathbf{W}_l^0 \sim N(\mathbf{0}, \mathbf{I}), \quad \mathbf{W}_l^q \sim \mathrm{MMNN}_{\phi_l}(\mathbf{W}_l^0). \tag{11}$$

Thus, we can use a MMNN to model the variational posterior of each layer's parameter W_l for large scale networks. However, in tasks with small networks, we still use MLPs to model q.

5 Related Work

Our work closely relates to implicit generative models (IGMs) and variational inference (VI). IGMs (generative models that define implicit distributions) have drawn much attention due to the popularity of GAN. 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).

Our work also builds upon the recent VI methods, including stochastic optimization by mini-batches (Hoffman et al., 2013), direct gradient optimization of variational lower bounds (Paisley et al.,

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

Dataset	**Avg. Test RMSE**			**Avg. Test LL**		
	SVGD	Dropout	KIVI	SVGD	Dropout	KIVI
Boston	2.957±0.099	2.97±0.19	**2.798±0.173**	-2.504±0.029	**-2.46±0.06**	-2.527±0.102
Concrete	5.324±0.104	5.23±0.12	**4.702±0.116**	-3.082±0.018	**-3.04±0.02**	-3.054±0.043
Energy	1.374±0.045	1.66±0.04	**0.467±0.015**	-1.767±0.024	-1.99±0.02	**-1.298±0.005**
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	**5.501±0.121**
Combined	4.033±0.033	4.02±0.04	**3.976±0.037**	-2.815±0.008	-2.80±0.01	**-2.794±0.009**
Protein	4.606±0.013	4.36±0.01	**4.255±0.019**	-2.947±0.003	-2.89±0.00	**-2.868±0.005**
Wine	**0.609±0.010**	0.62±0.01	0.629±0.008	**-0.925±0.014**	-0.93±0.01	-0.958±0.015
Yacht	0.864±0.052	1.11±0.09	**0.737±0.068**	**-1.225±0.042**	-1.55±0.03	-2.123±0.010
Year	**8.684±NA**	8.849±NA	8.950±NA	**-3.580±NA**	-3.588±NA	-3.615±NA

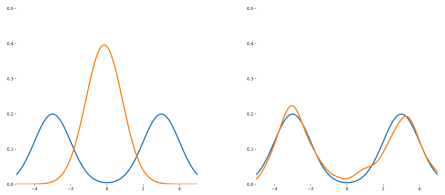
2012; Mnih & Gregor, 2014), and the reparametrization trick for continuous LVMs (Kingma & Welling, 2013). Following the success of learning with IGMs, implicit distributions are applied to VI. Huszár (2017) divide implicit VI using a discriminator into two categories: prior-contrastive and joint-contrastive. Classifiers in prior-contrastive methods distinguish between samples from the prior and the variational posterior, while in joint-contrastive methods they distinguish between the model joint distribution and the joint distribution composed of data distribution and variational posteriors. Concurrent with Huszár (2017), Mescheder et al. (2017) proposed Adversarial Variational Bayes (AVB), which is an amortized version of prior-contrastive methods for training local LVMs 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 VI (Liu & Wang, 2016; Dai et al., 2015) that adapt a set of particles towards the true posterior are also closely related to implicit VI. They share the similar advantage of flexible approximations.

6 Experiments

We present empirical results on both synthetic and real datasets to demonstrate the benefits of KIVI.

6.1 Toy 1-D Gaussian Mixtures

We firstly conduct a toy experiment to approximate a 1-D Gaussian mixture distribution with VI. The Gaussian mixture distribution has two equally distributed unit-variance components whose means are -3 and 3. We compare KIVI with VI using a single Gaussian posterior (Fig. 1). The variational distribution used by KIVI generates samples by forwarding a standard normal distribution through a two-layer MLP with 10 hidden units each and one output unit. As shown, the Gaussian posterior converges to the middle of the two modes, where the probability mass is small. In contrast, KIVI can accurately approximate the Gaussian mixture distribution with an expressive variational posterior.



(a) VI (normal posterior) (b) KIVI

Figure 1: Fitting Gaussian Mixture distribution

6.2 Bayesian Neural Networks (BNNs)

As stated in Sec. 4, the latent variables in BNNs are global to all data points and are usually very high-dimensional, for which a flexible variational family is essential. We compare KIVI with state-of-the-art VI methods by doing regression and classification on standard benchmarks.

6.2.1 Regression

To quantitatively measure the predictive ability of BNNs with KIVI as the inference method, we use standard multivariate regression benchmarks from recent works (Table 1), such as probabilistic backprogation (PBP) (Hernandez-Lobato & Adams, 2015). We compare with state-of-the-art methods: the Bayesian interpretation of dropout (Gal & Ghahramani, 2016) and stein variational gradient

Method	# Hidden	# Weights	Test err.
SGD (Simard et al., 2003)	800	1.3m	1.6%
Dropout (Srivastava et al., 2014)			$\approx 1.3\%$
Dropconnect (Wan et al., 2013)	800	1.3m	$\mathbf{1.2}\%^\star$
Bayes B. (Blundell et al., 2015), with Gaussian posterior	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes B. (Blundell et al., 2015), with scale mixture prior	400	500k	$1.36\%^\star$
	800	1.3m	$1.34\%^\star$
	1200	2.4m	$1.32\%^\star$
KIVI	400	500k	$\mathbf{1.29}\%$
	800	1.3m	$\mathbf{1.22}\%$
	1200	2.4m	$\mathbf{1.27}\%$

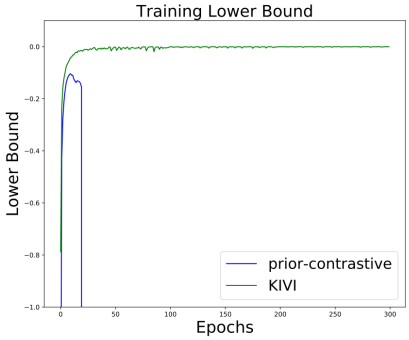


Figure 2: Results for MNIST classification. The left table shows the test error rates. $\star$ indicates results that are not directly comparable to ours: Wan et al. (2013) used an ensemble of 5 networks, and the second part of Blundell et al. (2015) changed the prior to a scale mixture. The plot on the right shows training lower bound in MNIST classification with prior-contrastive and KIVI.

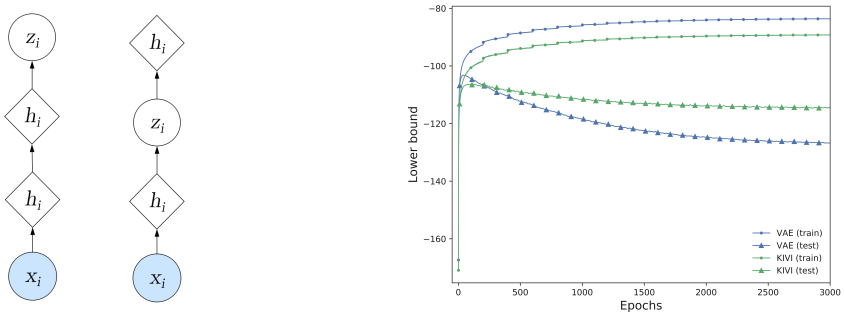
descent (SVGD) (Liu & Wang, 2016) [1]. Following the setup in PBP, we use BNNs with one 50-unit hidden layer except in the two large datasets, i.e., *Protein Structure* and *Year Predication MSD*, where 100 units are used. We randomly select 90% of the whole dataset for training and the rest for testing. We also put a Gamma prior on the observation noise precision to adaptively learn it (see Appendix G). For all datasets, we use batch size 100, $n_p = n_q = M = 100, \lambda = 0.001$, learning rate 1e-3. The model is trained for 3000 epochs for small datasets with less than 1000 data points, and 500 epochs for others. We report the mean errors and standard deviations averaged over 20 runs, except 5 runs for *Protein Structure* and 1 run for *Year Predication MSD*. As networks used in these tasks are of small scale, we use MLPs with one hidden layer in the implicit variational posterior (see Appendix F.3.1 for details).

Table 1 shows the results with the best ones marked in bold. Results of SVGD and dropout are copied from their papers, which have the same setting with us. We can see that KIVI consistently outperforms SVGD and dropout on both RMSE and test-LL for most datasets. Especially on RMSE, KIVI has significant improvements over them except on *Wine* and *Year Predication MSD*. It suggests that KIVI enables the implicit variational posterior to capture the structured uncertainty in network parameters, which is hard to be fully described by a mixture of two delta distributions (dropout) and a fixed set of particles (SVGD). We emphasize that although the nonparametric nature of SVGD has also made the approximation more flexible, it uses the same set of particles throughout the inference produre, while each iteration of KIVI generates a new set of particles. Thus the implicit posterior learned by KIVI is smoothed by the parametric model. Recently, normalizing flows have shown good performance on BNNs (Louizos & Welling, 2017). So we also experiment with directly applying normalizing flows on this task. The results are reported in Appendix E.2.

6.2.2 Classification

We finally present the classification results on MNIST, which consists of 60k training images and 10k test images of handwriting digits. Compared to the regression problems, MNIST has a much higher feature dimension, introducing millions of parameters even in moderate-size networks. Thus it brings big challenges for BNNs. As a standard benchmark, the performance on MNIST can be improved by many other techniques: convolution, generative pretraining, data augmentation, etc. To ensure fair comparison, we follow the settings from Bayes-By-Backprop (Blundell et al., 2015) and focus on improving the performance of ordinary MLPs without using any of these techniques. The network structures used are three MLPs, all with two ReLU hidden layers, and the layer sizes are 400, 800 and 1200, respectively. For KIVI, we used MMNNs with two hidden matrix multiplication layers in the implicit posterior (see Appendix F.3.2 for details). We set $n_p = n_q = M = 10, \lambda = 0.001$, batch size 100 and trained for 300 epochs. The initial learning rate is 0.001 and is annealed every 100

[1] Note that VMG (Louizos & Welling, 2016) and PBP-MV (Sun et al., 2017) both use adaptive weight priors, which are different from the common setting of standard normal priors; the former also additionally used variational dropout, thus their results are not comparable to the works discussed here as well as ours.



(a) (b)

Figure 3: Variational Autoencoders: (a) Gaussian posterior vs. implicit posterior; (b) Training and evaluation curves of the lower bounds on statically binarized MNIST.

epochs by ratio 0.5. We used the last 10k samples of the training set as the validation set for model selection.

Fig. 2 summarizes the results. We can see that KIVI achieves better accuracy compared to plain SGD (Simard et al., 2003), dropout (Srivastava et al., 2014) and Bayes-By-Backprop (Blundell et al., 2015) on all three types of MLPs. KIVI even performs better than Bayes-By-Backprop with a changed prior (scale mixture), which makes the model iterself more flexible than ours. When the layer size is 800, our result is comparable to an ensemble of 5 networks with dropconnect (Wan et al., 2013), which demonstrates that the implicit posterior has been learned to account for most model uncertainty.

We also conduct experiments with the prior-contrastive (PC) method (the corresponding counterpart of AVB for global latent variables, see Sec. 2). The key challenge for applying PC here is that posterior samples are extremely high-dimensional, and if fed into discriminators like neural networks, they will cause unaffordable computation cost. To get around this, we use logistic regression as the discriminator in PC. The experiment settings of PC are reported in Appendix F.3.2. The training lower bounds of the two methods are plotted in Fig. 2. We can see that at first they increase at the same pace, then PC fails to converge with lower bound explosion while KIVI improves consistently. The explosion is mainly because the input to the discriminator is of hundreds of thousands of dimensions, and plain logistic regression cannot produce reliable density ratio estimates. We also experiment with PC for layer size 800 and 1200. They both fail to converge in the end.

6.3 Variational Autoencoders

As stated in Sec. 3.2, KIVI is applicable to both global and local LVMs, and the latter can be learned using an amortized scheme (i.e., use $q_\phi(\mathbf{z}|\mathbf{x})$ instead of $q_\phi(\mathbf{z})$). Here we present an application on training variational autoencoders (VAE) with implicit posteriors to demonstrate this. The generative process of VAEs proceeds as

$$\mathbf{z} \sim N(\mathbf{0}, \mathbf{I}), \quad \mathbf{x} \sim \mathcal{P}(f_{NN}(\mathbf{z})), \tag{12}$$

where $\mathbf{z}$ are latent features, $\mathbf{x}$ are observations, $\mathcal{P}$ is the output distribution for modeling the observations, which takes the outputs of the neural network ($f_{\text{NN}}(\mathbf{z})$) as parameters. We conduct experiments on two widely used datasets for generative modeling: binarized MNIST and CelebA (Liu et al., 2015). $\mathcal{P}$ is Bernoulli for binarized MNIST, while a normal distribution for CelebA. For VAE the variational lower bound is the same form as Eq. (1), except $q_\phi(\mathbf{z})$ is replaced by $q_\phi(\mathbf{z}|\mathbf{x})$. The original VAE parameterizes $q_\phi(\mathbf{z}|\mathbf{x})$ as illustrated in Fig. 3a (left). To form an implicit posterior, a direct choice is to move the stochastic noise (denoted as z_i in the figure) one layer down the network and add a fully-connected layer on top of it to produce samples, as illustrated by Fig. 3a (right).

Before applying KIVI, a crucial question is what we expect to get by using implicit posteriors for training VAEs. One target could be that we may get tighter lower bounds of the data log-likelihood (LL) because the algorithm searches in a larger variational family for the optimal lower bound. This suggests, however in a very weak way, that optimizing in the larger space will lead to better test LL, given the optimization always arrives at local optima. Previously Adversarial Variational Bayes (AVB) has shown some results on MNIST, by comparing the test LL of models trained by plain VAE and AVB, using golden truths estimated by annealed importance sampling (AIS) (Wu et al., 2016). However, for the results reported in AVB, the model architectures used by plain VAE and AVB are

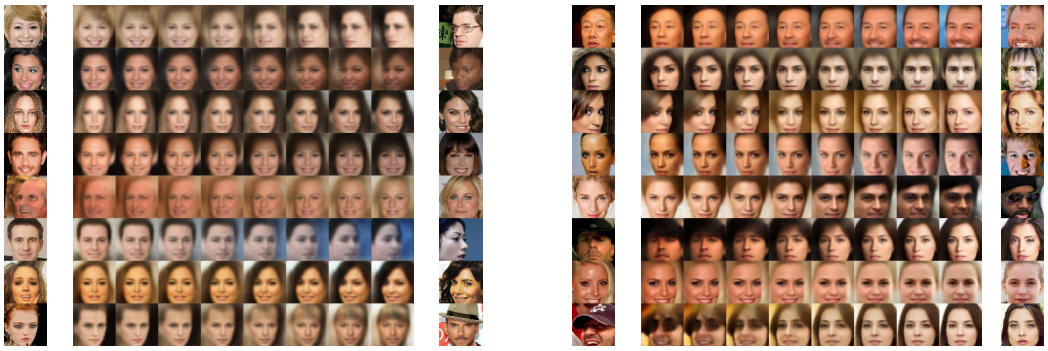


Figure 4: Interpolation experiments for celebA: AVB (left); KIVI (right).

very different, which leads to concerns about which part of the change contributes to the improved likelihoods.

Here we adopt another setting to better demonstrate the gain from implicit posteriors. We observe the key improvements of implicit posteriors is that objectives with them average over a much broader range of posterior configurations. This effect not only contributes to a larger search space that contains tighter lower bound values, but also keeps the VAE model better prevent overfitting. To verify that KIVI keeps this property, we conduct experiments on a statically binarized MNIST dataset and use models with no prior knowledge of the problem (MLP instead of convnets), which is a typical setting that leads to overfitting. The latent dimension of the used VAE model is 8, more details can be found in Appendix F.2. Fig 3b shows the training curves of VAEs with or without KIVI. We can see that the plain VAEs quickly overfit the training set while the KIVI trained VAEs are more robust against overfitting. After 3k epochs we evaluate the test LL on 2048 test images using AIS and get **-97.3** for plain VAEs, while **-94.8** for KIVI trained VAEs.

To demonstrate that KIVI scales to larger models, we trained a VAE with the same network structure used by DCGAN (Radford et al., 2015) on CelebA using KIVI, the latent dimimension in this case are 32. The implicit posterior is constructed in the way shown in Fig. 3a, with the bottom hidden layers symmetric to the decoder network (See Appendix F.2 for details). To visually check the latent space learned by KIVI, we show the reconstruction results of linearly interpolating between the latent $\mathbf{z}$ vector of two real images in Fig. 4 after 25 epochs, when the model converges. Compared to the latent space learned by AVB after the same epochs (We use the public code from AVB and set the same decoder structure), we find ours are smoother, and the interpolated images are much sharper. In fact many efforts are required to make AVB successfully train a model for CelebA to produce results shown in the figure. As reported in Mescheder et al. (2017), the log prior $\log p(\mathbf{z})$ is explicitly added to the discriminator ($T(\mathbf{x}, \mathbf{z})$), while KIVI doesn't need much tuning (no need to carefully design the discriminator, and the only two hyper-parameters (i.e, λ and the clipping value) have clear meanings).

7 Conclusions

We present an implicit variational inference method named Kernel Implicit Variational Inference (KIVI), which addresses the existing challenges of implicit variational inference, including noisy estimation and scalability with high-dimensional latent variable models. We successfully apply this approach to Bayesian neural networks and achieve superior performance on both regression and classification tasks. We also demonstrate that KIVI can be applied to learn local latent variable models like VAEs with implicit posteriors successfully. Future work may include applying this method on neural networks with larger scale and improving the kernel estimator further.

References

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

Claudio Carmeli, Ernesto De Vito, Alessandro Toigo, and Veronica Umanitá. Vector valued reproducing kernel hilbert spaces and universality. *Analysis and Applications*, 8(01):19–61, 2010.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

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

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

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

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

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

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

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

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

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

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

Bernhard Schölkopf, Ralf Herbrich, and Alex Smola. A generalized representer theorem. In *Computational learning theory*, pp. 416–426. Springer, 2001.

P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pp. 958–963, Aug 2003. doi: 10.1109/ICDAR.2003.1227801.

Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. *arXiv preprint arXiv:1706.07561*, 2017.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

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

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

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

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

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1058–1066, 2013.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

Jun Zhu, Jianfei Chen, Wenbo Hu, and Bo Zhang. Big learning with bayesian methods. *National Science Review (NSR)*, 2017.

A PROOF OF PROPOSITION 1

Proof. Let V denote the span of the representers of the sample points:

$$V = \text{span}(\{k(\mathbf{z}_i, \cdot) : i = 1, \ldots, n\} = \left\{ \sum_{i=1}^{n_p} \alpha_i k(z_i^p, \cdot) + \sum_{j=1}^{n} \beta_j k(z_j^q, \cdot) \right\}, \tag{13}$$

and define its orthogonal complement to be

$$V_\perp = \{f \in \mathcal{H} : \langle f, g \rangle = 0, \forall g \in V\} \tag{14}$$

Because $\hat{r} \in \mathcal{H}$, it can be decomposed into two parts:

$$\hat{r} = \hat{r}_\parallel + \hat{r}_\perp \tag{15}$$

where $\hat{r}_\parallel \in V$, and $\hat{r}_\perp \in V_\perp$. Note that the squared loss is not changed by the orthogonal component $\hat{r}_\perp$:

$$\hat{r}(\mathbf{z}_i) = (\hat{r}_\parallel + \hat{r}_\perp)(\mathbf{z}_i) = \hat{r}_\parallel(\mathbf{z}) + \langle \hat{r}_\perp, k(\mathbf{z}_i, \cdot) \rangle = \hat{r}_\parallel(\mathbf{z}_i) \tag{16}$$

Meanwhile the regularization term can be decomposed into $\|\hat{r}_\parallel\|_{\mathcal{H}}^2 + \|\hat{r}_\perp\|_{\mathcal{H}}^2$. So the optimal solution will have $\hat{r}_\perp^* = 0$, which indicates that $\hat{r}^* \in V$. □

B DERIVATION OF THE OPTIMAL DENSITY RATIO FUNCTION

Substituting Eq. (5) into Eq. (3) and removing the constant, we get

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) =& \frac{1}{2n_p} \left(\|\mathbf{K}_p \boldsymbol{\alpha}\|_2^2 + \|\mathbf{K}_{pq} \boldsymbol{\beta}\|_2^2 + 2\boldsymbol{\alpha}^\top \mathbf{K}_p \mathbf{K}_{pq} \boldsymbol{\beta} \right) - \frac{1}{n_q} \left(\boldsymbol{\alpha}^T \mathbf{K}_{pq} \mathbf{1} + \boldsymbol{\beta}^\top \mathbf{K}_q \mathbf{1} \right) \\ &+ \frac{\lambda}{2} (\boldsymbol{\alpha}^\top \mathbf{K}_p \boldsymbol{\alpha} + \boldsymbol{\beta}^\top \mathbf{K}_q \boldsymbol{\beta} + 2\boldsymbol{\alpha}^\top \mathbf{K}_{pq} \boldsymbol{\beta}), \end{aligned} \tag{17}$$

Taking derivatives w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and setting them to be zeros:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} = \frac{1}{n_p} \mathbf{K}_p (\mathbf{K}_p \boldsymbol{\alpha} + \mathbf{K}_{pq} \boldsymbol{\beta}) - \frac{1}{n_q} \mathbf{K}_{pq} \mathbf{1} + \lambda \mathbf{K}_p \boldsymbol{\alpha} + \lambda \mathbf{K}_{pq} \boldsymbol{\beta} = \mathbf{0}, \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \frac{1}{n_p} \mathbf{K}_{pq}^\top (\mathbf{K}_{pq} \boldsymbol{\beta} + \mathbf{K}_p \boldsymbol{\alpha}) - \frac{1}{n_q} \mathbf{K}_q \mathbf{1} + \lambda \mathbf{K}_q \boldsymbol{\beta} + \lambda \mathbf{K}_{pq}^T \boldsymbol{\alpha} = \mathbf{0}. \tag{19}$$

Rearranging the terms, we have

$$\left(\frac{1}{n_p} \mathbf{K}_p \mathbf{K}_{pq} + \lambda \mathbf{K}_{pq} \right) \boldsymbol{\beta} + \mathbf{K}_p \left(\frac{1}{n_p} \mathbf{K}_p + \lambda \mathbf{I} \right) \boldsymbol{\alpha} = \frac{1}{n_q} \mathbf{K}_{pq} \mathbf{1}, \tag{20}$$

$$\left(\frac{1}{n_p} \mathbf{K}_{pq}^\top \mathbf{K}_{pq} + \lambda \mathbf{K}_q \right) \boldsymbol{\beta} + \mathbf{K}_{pq}^\top \left(\frac{1}{n_p} \mathbf{K}_p + \lambda \mathbf{I} \right) \boldsymbol{\alpha} = \frac{1}{n_q} \mathbf{K}_q \mathbf{1}. \tag{21}$$

Left multiplying Eq. (20) with $\mathbf{K}_{pq}^\top \mathbf{K}_p^{-1}$ and then subtracting Eq. (21) yields

$$\boldsymbol{\beta} = \frac{1}{\lambda n_q} \mathbf{1}. \tag{22}$$

Substituting the optimal $\boldsymbol{\beta}$ into Eq.(20), we get

$$\boldsymbol{\alpha} = -\frac{1}{\lambda n_p n_q} (\frac{1}{n_p} \mathbf{K}_p + \lambda \mathbf{I})^{-1} \mathbf{K}_{pq} \mathbf{1} \tag{23}$$

Note that although we use $\mathbf{K}_p^{-1}$ in the derivation, the forms of the solution do not require $\mathbf{K}_p$ to be invertible. In fact, if $\mathbf{K}_p$ has zero eigenvalues (not invertible), the objective $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is not bounded below since one can always choose a $\boldsymbol{\alpha}$ in the null space of $\mathbf{K}_p$ to decrease it. In other words, the form of the solution is well defined in any condition and is optimal when the objective is well defined.

C GRADIENTS OF THE KL TERM

$$\begin{aligned}\nabla_\phi \mathrm{KL}(q_\phi||p) &= \nabla_\phi \mathrm{E}_{q_\phi} \log \frac{q_\phi}{p} = \int \nabla_\phi q_\phi(z) \log \frac{q_\phi}{p} + q_\phi(z) \nabla_\phi \log \frac{q_\phi}{p} \mathrm{d}z \\ &= \nabla_\phi \mathrm{E}_{q_\phi} \log \frac{q}{p} + \int \nabla_\phi q_\phi(z) \mathrm{d}z = \nabla_\phi \mathrm{E}_{q_\phi} \log \frac{q}{p}\end{aligned} \tag{24}$$

The formula above shows that a good density ratio estimator gives accurate gradient estimation of ϕ.

D KIVI WITH ADAPTIVE CONTRAST

Although KIVI gives rather good estimation of the density ratio, the estimation accuracy still degrades with larger discrepancy between p and q. The problem is very critical in training local latent variable models like VAEs because the same variational model is required to infer posteriors of all local latent variables. In order to mitigate that, AVB (Mescheder et al., 2017) adopted a technique called Adaptive Contrast (AC), which can easily be integrated with KIVI. AC introduces an auxiliary explicit distribution $r_\alpha(\mathbf{z}|\mathbf{x})$ that resembles q. With $r_\alpha(\mathbf{z}|\mathbf{x})$, the ELBO can be rewritten as

$$\mathcal{L}(\mathbf{x};\phi) = \mathrm{E}_{q_\phi}(-\log r_\alpha(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x},\mathbf{z})) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||r_\alpha(\mathbf{z}|\mathbf{x})) \tag{25}$$

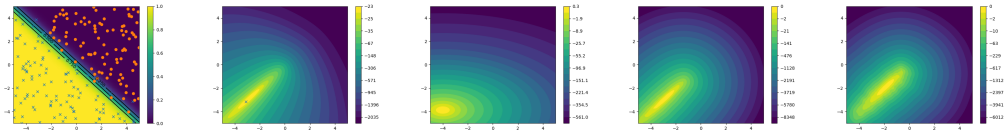
For explicit $r_\alpha(\mathbf{z}|\mathbf{x})$, gradients of the first term towards θ and ϕ can be easily computed, and KL can be estimated using KIVI. Compared to prior, adaptive contrast estimates KL better if $r_\alpha(\mathbf{z}|\mathbf{x})$ approximates q well. To make sure it is explicit, a commonly used adaptive distribution is Gaussian distribution whose mean $\mu(\mathbf{x})$ and standard derivation $\sigma(\mathbf{x})$ matches with q. According to KL divergence's invariance under reparamterization, we have

$$\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||r_\alpha(\mathbf{z}|\mathbf{x})) = \mathrm{KL}(\hat{q}_\phi(\hat{\mathbf{z}}|\mathbf{x})||\hat{r}_0(\hat{\mathbf{z}}))$$

Where $\hat{q}_\phi(\mathbf{z}|\mathbf{x})$ denotes the distribution of $\hat{\mathbf{z}} = \frac{\mathbf{z}-\mu x}{\sigma x}$ and $\hat{r}_0(\hat{\mathbf{z}})$ denotes standard normal distribution. Under this reparameterization, we only need to estimate the density ratio between two distributions with 0 mean and 1 variance. In addition, the mean $\mu(x)$ and standard derivation $\sigma(x)$ can be estimated with Monte Carlo from samples of q.

E ADDITIONAL EXPERIMENTS

E.1 2-D BAYESIAN LOGISTIC REGRESSION



(a) Training data (b) True posterior (c) VI (factorized) (d) HMC (e) KIVI

Figure 5: 2-D Bayesian logistic regression

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

$$\mathbf{w} \sim N(\mathbf{0}, \mathbf{I}), \quad y_i \sim \mathrm{Bernoulli}(\sigma(\mathbf{w}^T \mathbf{x}_i)), \quad i = 1, \ldots, N \tag{26}$$

where $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^2$; σ is the sigmoid function. $N = 200$ data points ($\{(x_i, y_i)\}_{i=1}^{200}$) are generated from the true model as the training data (Fig. 5a). The unnormalized true posterior is plotted in Fig. 5b. As a baseline, we first run VI with factorized normal distributions. The result is shown in Fig. 5c. It can be clearly seen that the factorized normal can capture the position and the scale of the true posterior but cannot fit well to the shape due to its independence property across dimensions.

We then apply KIVI. The implicit posterior we use is a simple stochastic neural network (see Appendix F.1). To demonstrate how good the result is, we also run Hamiltonian Monte Carlo (HMC) to get posterior samples. The results are plotted in Fig. 5d and 5e. We can see that the implicit posterior is 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.

E.2 COMPARISON WITH NORMALIZING FLOW

For comparing the results of KIVI with factorial variance inference and normalizing flow, we use 100 samples, batch size 10 except for *kin8nm* and *naval* we use batch size 100. We set the learning rate as 0.01 and run 500 epochs for 20 times. In NF, we apply 10 planar normalizing flows on the weights to match the computation time of our implicit posterior. The results are shown in Table 2. However, in the regression task normalizing flows do not have improvements over VI with factorized normal posteriors. We suspect that this is due to the limited form of normalizing flows has led to optimization challenges, thus the inference procedure doesn't take much benefit from more flexibility introduced by the flow.

Table 2: Test RMSE, log-likelihood for the regression datasets. Factorized and NF represent VI with factorized normal posteriors and normalizing flow, respectively.

RMSE	**Factorized**	**NF**	**KIVI**
boston	3.42±0.19	3.43±0.19	**2.80±0.17**
concrete	6.00±0.10	6.04±0.10	**4.70±0.12**
energy	2.42±0.06	2.48±0.09	**0.47±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	**Factorized**	**NF**	**KIVI**
boston	-2.66±0.04	-2.66±0.04	**-2.53±0.10**
concrete	-3.22±0.06	-3.24±0.06	**-3.05±0.04**
energy	-2.34±0.02	-2.36±0.03	**-1.30±0.01**
kin8nm	0.96±0.01	1.01±0.01	**1.16±0.01**
naval	4.00±0.11	4.04±0.12	**5.50±0.12**

F DETAILS OF EXPERIMENTS

F.1 TOY EXPERIMENTS

1-D Gaussian Mixture We generate samples by forwarding a standard normal distribution through a two-layer MLP with 10 hidden units and one output unit. We set regularization coefficient λ as 0.003 and density ratio clipping threshold 1e-8.

2-D Bayesian Logistic Regression We compare factorized VI, KIVI and HMC in 2-D Bayesian logistic regression with $\mathcal{N}(0, 1)$ prior. The training data $\mathbf{X}$ are randomly sampled 200 points from $\mathrm{U}[-5, 5] \times \mathrm{U}[-5, 5]$, output $\mathbf{Y}$ are prediction of $\mathbf{X}$ with randomly sampled weights from prior. For HMC, we run 100 chains, 200 iterations each, 10 leapfrog steps, and adaptive step size beginning from 1e-3. We discard first 100 samples generated. For factorized variational inference, we run for 100 epochs using full batch and 100 samples. Along the training, we anneal the learning rate linearly according to $lr = \frac{100}{100+epoch-1}$. We follow the same setting as factorized variational inference for KIVI, except that we use 1000 samples. The regularization coefficient λ and minimal density ratio interval are set to 0.1, $1\mathrm{e}{-8}$, respectively. For modelling the implicit variational posterior in KIVI, we forward 2-D random normal samples through 2 20-neuron fully connected layers getting $\mathbf{h}$ (we don't use activation function for the last layer), then plus $\mathbf{h}$ with another random normal noise with adaptive variance getting $\mathbf{z}$. Finally we forward $\mathbf{z}$ through a 20-neuron layer and then 2-neuron linear layer getting the variational samples.

F.2 VARIATIONAL AUTOENCODERS

The decoders used in the MNIST experiment are MLPs with two hidden ReLU layers. The latent dimension is 8. Each hidden layer is of size 500. The implicit posterior is also a MLP with two hidden ReLU layers, with Gaussian noises of 500 dimension, zero mean and 500 dimensional adaptive variances added to the first hidden layer.

The decoders used in the CelebA experiment are exactly the same structure with the one used for 64-size images in the DCGAN paper. The latent dimension is 32. The implicit posterior is a deep convolutional neural network with symmetric structure to the decoder, the output of the last convolutional layer is flattened and is added a Gaussian noise of the same shape as the last dimension. The noise is zero mean and has adaptive variances. Then the last hidden layer is of 500 hidden RelU units. For AVB we use the same decoder.

F.3 Bayesian Neural Networks

F.3.1 Regression

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 for the variational posterior, where the samples of variational posterior is output of forwarding random normal distribution through a MLP. For all dataset, we use Relu as activation functions, we use MLP with one hidden layer except for *yacht* we use 2 hidden layers.

We list the details in Table 3, which consists of 10 datasets and 2 weight matrices each. Take Layer1 for boston as an example, (20, 30, N1) represents 20 random normal noise was generated and forwarded through a MLP with hidden neurons 30 and output neurons N1$= 14 \times 50$, thus N1 corresponding to the number of weights in the first layer of the Bayesian neural network.

Table 3: Implicit posteriors details for regression experiments

	Boston	**Concrete**	**Energy**	**Kin8nm**	**Naval**
Layer 1	(20, 30, N1)	(30, 50, N1)	(100, 500, N1)	(100, 500, N1)	(100, 500, N1)
Layer 2	(20, 30, N2)	(30, 50, N2)	(50, 100, N2)	(50, 100, N2)	(50, 100, N2)
	Combined	**Protein**	**Wine**	**Yacht**	**Year**
Layer 1	(100, 500, N1)	(100, 500, N1)	(20, 10, N1)	(100, 800, N1, N1)	(100, 500, N1)
Layer 2	(100, 500, N2)	(100, 500, N2)	(5, 20, N2)	(50, 200, 51, N2)	(100, 500, N2)

F.3.2 Classification

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.

With the variational posterior settings above, we perform experiments with KIVI and PC. We use logistic regression as discriminator for PC. We set regularization coefficient λ as 0.001 and minimal density ratio threshold as $1\mathrm{e}{-8}$ for KIVI. Both use 10 samples, batch size 100 and learning rate 0.001 for training. However, PC fails to converge because of the large scale latent space and KIVI shows promising results.

G LOWER BOUND WITH GAMMA-PRIOR PRECISION

In the multivariate regression task, the output is sampled from a normal distribution with $\hat{y}\left(x, \mathbf{W}\right)$ 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 $\mathrm{Gamma}\left(6, 6\right)$ on its reciprocal (i.e., the precision of the Normal distribution). The variational posterior we used is $q\left(\mathbf{W}, \lambda\right) = q\left(\mathbf{W}\right) q\left(\lambda\right), q\left(\lambda\right) \sim \mathrm{Gamma}\left(\alpha, \beta\right)$. Then the ELBO can be computed as

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

where $\psi\left(x\right)$ is the digamma function and $\mathrm{KL}\left(q(\lambda) \| p(\lambda)\right)$ can be calculated in closed-form.