Variational Variance: Simple and Reliable Predictive Variance Parameterization

Andrew Stirn, David A Knowles*
Department of Computer Science
Columbia University
New York, NY 10027
{andrew.stirn,daknowles}@cs.columbia.edu

Abstract

An often overlooked sleight of hand performed with variational autoencoders (VAEs), which has proliferated the literature, is to misrepresent the posterior predictive (decoder) distribution’s expectation as a sample from that distribution. Jointly modeling the mean and variance for a normal predictive distribution can result in fragile optimization where the ultimately learned parameters can be ineffective at generating realistic samples. The two most common principled methods to avoid this problem are to either fix the variance or use the single-parameter Bernoulli distribution—both have drawbacks, however. Unfortunately, the problem of jointly optimizing mean and variance networks affects not only unsupervised modeling of continuous data (a taxonomy for many VAE applications) but also regression tasks. To date, only a handful of papers have attempted to resolve these difficulties. In this article, we propose an alternative and attractively simple solution: treat predictive variance variationally. Our approach synergizes with existing VAE-specific theoretical results and, being probabilistically principled, provides access to Empirical Bayes and other such techniques that utilize the observed data to construct well-informed priors. We extend the VAMP prior, which assumes a uniform mixture, by inferring mixture proportions and assignments. This extension amplifies our ability to accurately capture heteroscedastic variance. Notably, our methods experimentally outperform existing techniques on supervised and unsupervised modeling of continuous data.

1 Introduction

Employing neural networks to map a conditioning random variable onto the parameter space for a distribution describing some other dependent random variable is commonplace nowadays. For example, we typically map covariates onto the simplex with neural networks to parameterize a categorical distribution over observed labels during supervised classification. Using neural networks to parameterize the mean and variance for the normal distribution is also prevalent (Nix and Weigend, 1994; Kingma and Welling, 2013; Rezende et al., 2014) but problematic. In particular, if our conditional mean network predicts nearly perfectly (i.e. \( \mu(x_i) \approx y_i \forall i \in [N] \)), then maximizing the log likelihood will push the variance network \( \sigma^2(x_i) \) towards a pathological 0. This tendency coupled with the fact that \( \sigma^{-2} \) appears as a multiplicative factor in the gradient of the normal log likelihood w.r.t. \( \mu \), underlies why jointly optimizing mean and variance networks can be unstable. As the mean estimates \( \mu(x_i) \) improve, the log likelihood encourages \( \sigma^{-2}(x_i) \to \infty \) such that minuscule errors by the mean network can produce inappropriately large parameter updates. The variance network \( \sigma^2(x_i) \) effectively controls the learning rate of the mean network \( \mu(x_i) \)--increasing it as the mean network

*jointly affiliated with Columbia University’s Data Science Institute and the New York Genome Center

Preprint. Under review.
improves—in direct opposition to the stochastic gradient descent convergence criteria of Robbins and Monro (1951). This leaves one wondering if fragile optimization simply should be expected when jointly learning parameterizations for mean and variance. While there are examples of good optima being found when these criteria are not met, instability has been observed when optimizing mean and variance networks in regression (Detlefsen et al., 2019) as well as when using separate mean and variance networks for normal decoder likelihoods in VAEs (Takahashi et al., 2018). The theoretical preference of an optimal decoder for zero variance has also been examined (Dai and Wipf, 2019).

Learning reliable parameterizations for the mean and variance has numerous applications in today’s machine learning arena. For regression, accurate variance estimates enable Bayesian active learning (Cohn et al., 1996) and reinforcement learning (Ghavamzadeh et al., 2016) regimes where new data is requested or exploration carried out based on predicted variance. Realistic sample generation is critical to adversarial learning and data imputation. We inspected the code of many state-of-the-art VAE methods and found that the generated ‘samples’ were rarely sampled from the decoder distribution. Instead, these methods ancestrally resample latent variables and report the decoder’s expected value, which preserves uncertainty on the latent space but obfuscates it for the learned distribution(s) over the observed data. VAE papers claiming improvements to sample quality (van den Oord et al., 2017; Razavi et al., 2019) and imputation (Nazabal et al., 2018; Mattei and Frellsen, 2018b) do not sample the decoder distribution despite sometimes optimizing its variance (Dai and Wipf, 2019). Posterior predictive checks (PPC) (Ranganath and Blei, 2019) are Bayesian model criticism techniques that utilize the posterior predictive distribution. Generating realistic samples from the decoder distribution constitutes one such test, for which we find that continuous VAEs can fail (figs. 1, 7 and 8).

In this article, we advocate a Bayesian treatment of the predictive variance (or rather precision for computational convenience). Treating variance variationally—conditioned on the covariates for regression and on latent codes for VAEs—fortifies the joint optimization of the mean and variance networks that parameterize the likelihood of observed data. This fortification arises from the resulting Kullback–Leibler (KL) divergence, which, for an appropriate prior, will induce gradients that prohibit variance from approaching the aforementioned zero pathology (somewhat analogous to logarithmic barriers to enforce constraints in convex optimization). Thus, variational variance not only stabilizes optimization but also effectively upper-bounds the predictive likelihood, which in turn alleviates the theoretical concern that maximum likelihood estimation of continuous VAEs is ill-posed (Mattei and Frellsen, 2018a). In section 2, we review relevant variational inference concepts. In section 3, we formalize our proposed methods for regression and introduce our novel prior families. Therein, we also compare our methods to Detlefsen et al. (2019) on a variety of regression tasks with accompanying active learning experiments. In section 4, we compare our posterior predictive sample
quality to that of Takahashi et al. (2018). We emphasize that our proposals broadly apply to both regression and continuous VAEs and notably outperform other methods specific to each context.  

2 Variational Inference: Normal Likelihoods and Precision Priors

Variational inference (VI) (Blei et al., 2017) posits a family of tractable distributions \( q(\Theta; \nu) \) to approximate the true posterior \( p(\Theta | D) \) over latent variables \( \Theta \) conditioned on observed data \( D \). We assume i.i.d. data such that \( p(D | \Theta) = \prod_{d \in D} p(d | \Theta) \). Often, and as in the case of amortized inference (Kingma and Welling, 2013), we use a neural network with shared learnable parameters \( \phi \) to map data \( d \) onto the variational parameters \( \nu \) (i.e. \( f_\phi : d \rightarrow \nu \)). Amortized VI minimizes \( D_{KL}(q(\Theta | D) || p(\Theta | D)) \) by maximizing the evidence lower bound (ELBO)

\[
L = \sum_{d \in D} \mathbb{E}_{q(\Theta | f_\phi(d))} \left[ \log p(d | \Theta) \right] - D_{KL}(q(\Theta | f_\phi(d)) || p(\Theta))
\]  

since the summation of these dual objectives is constant and KL divergence is non-negative. We focus on Gaussian likelihoods with mean \( \mu \) and precision \( \lambda \). If we treat \( \lambda \) variationally—consider it a latent variable (i.e. \( \lambda \in \Theta \)), specify a prior \( p(\lambda) \) to describe its generative process, and employ a variational family \( q(\lambda | \cdot) \) to approximate the posterior—then the KL divergence in eq. (1) will contain \( D_{KL}(q(\lambda | \cdot) || p(\lambda)) \). This term fortifies optimization as discussed in section 1 and, with well-informed priors, ideally will find distributions over \( \lambda \) that accurately reflect the local predictive ability of mean network \( \mu(\cdot) \). We now give a general treatment and later explicate the model for regression in eq. (2) and fig. 4 and for VAEs in eqs. (4) to (6).

3 Variational Variance for Regression

Existing Methods. Parameterizing the conditional mean and variance of an observed, normally distributed response (target) variable with neural networks operating on covariates (features) is captured in fig. 2 and yields a likelihood \( p(y_i | x_i) \triangleq \mathcal{N}(y_i | \mu(x_i), \sigma_2(x_i)) \). Only recently (NeurIPS 2019) has the reliability of the variance network \( \sigma_2(x_i) \) been critically examined and improved (Detlefsen et al., 2019). As well as optimization instabilities, the authors observe that simultaneously learning neural mean and variance parameterizations will underestimate the true variance, especially in areas of covariate space with scarce data. They propose four tricks to ameliorate these issues: locality sampling, isolated mean-variance training, using a Student’s \( t \) likelihood (as a Gamma-Normal compound distribution, fig. 3), and variance extrapolation (see appendix section 5.1 for a summary of their methods). Detlefsen et al. (2019) perform ablation and find that their methods are generally complementary with the locality sampler and Student’s \( t \) distribution individually providing the most benefit. Their top method, which employs all four of their proposals, generally outperforms their chosen baselines: Gaussian processes (Williams and Rasmussen, 2006; Snelson and Ghahramani, 2006; Damianou and Lawrence, 2013), unmodified neural-network parameterizations

†Shaded and transparent circles are respectively observed and latent RVs. Diamonds are deterministic nodes (i.e. output of a neural net). Solid arrows denote generative process. Dashed arrows define the variational family.
We derive eq. (3) in section 5.2, where we further decompose it (eq. (7)) to provide insight on $y$.

Analytic evaluation is preserved under a diagonal covariance assumption across dimensions of $y$.

Proposed Methods. In contrast to Detlefsen et al. (2019), we propose a single, simple modification to the naive approach of fig. 2 which is to treat precision variationally (fig. 4). Our variational objective is then

$$\mathcal{L} = \sum_{(x,y) \in \mathcal{D}} \mathbb{E}_{q(\lambda(x), \beta(x))} \left[ \log \mathcal{N}(y|\mu(x), \lambda) - D_{KL}(q(\lambda(x), \beta(x)) \| p(\lambda)) \right],$$

for which we employ black-box VI (Ranganath et al. 2014) with reparameterization gradients (Salimans et al. 2013; Kingma and Welling 2013; Rezende et al. 2014; Figurnov et al. 2018) to optimize $\mathcal{L}$ w.r.t. the networks $\mu(\cdot)$, $\alpha(\cdot)$, and $\beta(\cdot)$. We consider $q(\lambda|\alpha(x), \beta(x)) \in \{\text{Gamma}(\lambda|x), \beta(x)), \text{LogNormal}(\lambda|x), \beta(x))\}$, which both admit closed-formed univariate expected log likelihoods. We use $\alpha(x)$ and $\beta(x)$ to represent respectively the mean and variance parameters for the LogNormal to keep notation consistent across variational families.

$$\mathbb{E}_q[\log \mathcal{N}(y|\mu(x), \lambda)] = \frac{1}{2} \left( \mathbb{E}_q[\log \lambda] - \log(2\pi) - \mathbb{E}_q[\lambda] - \mu(x))^2 \right)$$

$$\mathbb{E}_q[\lambda] = \begin{cases} \frac{\alpha(x)}{\sigma(x)^2}, & q = \text{Gamma} \\ \exp(\alpha(x) + \beta(x) / 2), & q = \text{LogNormal} \end{cases}$$

$$\mathbb{E}_q[\log \lambda] = \begin{cases} \psi(\alpha(x)) - \log(\beta(x)), & q = \text{Gamma} \text{ (}$\psi$ is the Digamma) \\ \alpha(x), & q = \text{LogNormal} \end{cases}$$

Analytic evaluation is preserved under a diagonal covariance assumption across dimensions of $y$. $\alpha(x)$ and $\beta(x)$ employ softplus activations to ensure they give valid (positive) parameter values.

For regression, we consider five classes of precision priors. We always set $q(\lambda|\alpha(x), \beta(x))$ to be of the same family as $p(\lambda)$ or its mixture components. First, we use standard priors, $p(\lambda) \triangleq \text{Gamma}(\lambda; a, b)$ and $p(\lambda) \triangleq \text{LogNormal}(\lambda; a, b)$, where the former is conjugate and the latter is not but has the correct support (conjugacy is not required for black-box VI). Second, we use what we call an MLE prior. Formally, this prior independently sets $p(\lambda_i) \triangleq q(\lambda|\alpha(x_i), \beta(x_i))$ for each data point such that the KL divergence penalty in eq. (2) vanishes. This ‘prior’ serves as a control in assessing our claims that the KL divergence is useful in fortifying the simultaneous optimization of mean and variance parameter networks. Third, we utilize a novel modification of the VAMP prior (Tomczak and Welling 2017). Typically, one sets the prior in advance. The original VAMP prior, however, assumes an Empirical Bayes perspective by finding an optimal prior that maximizes the ELBO. This prior turns out to be the aggregate posterior $p^*(\lambda) = N^{-1} \sum_{i=1}^N q(\lambda|\alpha(x_i), \beta(x_i))$ taken over the $N$ training points. The original VAMP prior proposes using $K$ randomly selected (without replacement) training points (pseudo-inputs) instead of all $N$. They denote the $i$'th pseudo-input as $u_i$. In our initial investigations, we found that the original VAMP prior was poor at capturing heteroscedastic variance (falsely predicting homoscedasticity) due to the assumed uniform mixture of variational posteriors (see appendix section 5.4 for details). We resolved this problem by introducing latent mixture probabilities with prior $p(c) \triangleq \text{Uniform}[K]$ such that the joint and marginal priors are

$$p(\lambda, c) \triangleq q(\lambda|\alpha(u_c), \beta(u_c)) \cdot p(c) \Rightarrow p(\lambda) = \sum_{i=1}^K q(\lambda|\alpha(u_i), \beta(u_i)) \cdot p(c = i).$$

Since $c$ is latent, we may treat it variationally with a discrete $q(c|\pi(x))$, where $\pi(x)$ is a neural network that maps $x$ onto the simplex. This augmentation requires replacing the KL divergence of eq. (2) with

$$D_{KL}(q(c|\pi(x)) \| p(c)) + \mathbb{E}_{q(c|\pi(x))} \left[ D_{KL}(q(\lambda|\alpha(x), \beta(x)) \| q(\lambda|\alpha(u_c), \beta(u_c))) \right].$$

We derive eq. (3) in section 5.2 where we further decompose it (eq. (7)) to provide insight on its computational evaluation. To summarize, we evaluate the LHS of eq. (3) analytically. For the
Tomczak and Welling (2017) also introduce the concept of treating pseudo-inputs \{u_1, \ldots, u_K\} as trainable parameters. We too consider trainable pseudo-inputs for our modified VAMP prior, which we denote as VAMP*.

For our fifth and final prior, we define the joint and marginal as

\[
p(\lambda, c) \triangleq p(\lambda|a_c, b_c) \cdot p(c) \Rightarrow p(\lambda) = \sum_{c=1}^{K} p(\lambda|a_c, b_c) \cdot p(c = i),
\]

where \(p(\lambda|a_c, b_c) \in \{\text{Gamma}(\lambda|a_c, b_c), \text{LogNormal}(\lambda|a_c, b_c)\}\) and \(\{a_1, b_1, \ldots, a_K, b_K\}\) are trainable prior parameters, which we restrict to the positive reals via the softplus function (see section 5.3.1). We denote this prior as VBEM, which stands for Variational Bayes Expectation Maximization, since optimizing the prior parameters during VI is analogous to performing M steps. The VBEM KL divergence is identical to eq. 3 except that we replace \(q(\lambda|\alpha(u_c), \beta(u_c))\) with \(p(\lambda|a_c, b_c)\).

**Toy Data.** Detlefsen et al. (2019) propose simulating data with heteroscedastic variance. We modify their process slightly to ensure non-negative variance. The process is defined as \(y|x \sim \lambda \cdot x \cdot \sin(x) + \epsilon\) where \(\epsilon|x \sim \mathcal{N}(0, [0.3 \cdot (1 + x)]^2)\). We train on covariates \(x\) sampled uniformly from \([0, 10]\) and test over equally spaced points in \([-4, 14]\). We noticed that Detlefsen et al. (2019)’s code’s normal log likelihood’s log variance term was missing a \(\frac{1}{2}\) (this bug only affected this particular experiment). We ran their code with and without our correction to assess its impact. We mimicked the implementation specifics of Detlefsen et al. (2019) (see section 5.3.1). All methods do well at estimating the true mean on the interval of observed data \([0, 10]\) (top two rows of subplots, fig. 5). Fixing their bug significantly improves their ability to learn the true variance on \([0, 10]\) (first subplot of bottom row, fig. 5). Our MLE method, for both Gamma and LogNormal precisions, are equally accurate on this interval (second subplot of bottom row, fig. 5). The VAMP and VAMP* priors behave similarly and seemingly sacrifice accuracy near the closure of \([0, 10]\), with the Gamma-distributed precision edging out the LogNormal (fourth and fifth subplots of bottom row, fig. 5). Interestingly, the standard Gamma prior (third subplot of bottom row, fig. 5) while overestimating the true variance on parts of \([0, 10]\), is the only method to consistently overestimate the variance outside of \([0, 10]\) (a desiderata of Detlefsen et al. (2019)).
Table 1: UCI regression log likelihoods reported as mean ± std. We bold the top performer as well as any others who are statistically indistinguishable (ρ ≥ 0.05). Tuples below experiment names are of the form (Nobservations, dim(x), dim(y)).

| Algorithm  | Prior          | boston       | carbon       | concrete     | energy       | naval        |
|------------|----------------|--------------|--------------|--------------|--------------|--------------|
| Detlefsen  | N/A            | -4.53 ± 0.12 | 9.549 ± 0.101| -3.287 ± 0.031| -4.246 ± 0.037| 10.64 ± 0.209|
| Gamma-Normal | MLE           | -2.829 ± 0.337| 13.83 ± 2.879| -3.215 ± 0.2  | -0.836 ± 0.953| 14.396 ± 0.397|
|            | Standard       | -3.216 ± 0.047| 1.099 ± 0.003| -3.779 ± 0.017| -6.4 ± 0.013| 6.851 ± 0.003|
|            | VAMP           | -2.779 ± 0.309| 15.47 ± 2.748| -3.123 ± 0.209| -0.608 ± 1.032| 14.451 ± 0.452|
|            | VAMP*          | -2.785 ± 0.33 | 15.42 ± 2.92 | -3.126 ± 0.21 | -0.703 ± 0.735| 14.467 ± 0.494|
|            | VBEM           | -2.63 ± 0.276 | 13.587 ± 2.682| -3.097 ± 0.187| 0.453 ± 0.091| 16.967 ± 0.301|

Table 1: UCI regression results–we omit results for our LogNormal priors as they performed

UCI Data. We consider many of the same UCI data sets as [Detlefsen et al., 2019] and process them similarly. To summarize, we independently whiten all features and targets to enforce zero mean and unit variance, while also reporting performance metrics for the original target scalings. We collected all reported metrics from 20 randomly held-out test sets that each constituted 10% of the data. Again, we matched the implementation specifics of Detlefsen et al. (2019) (see section 5.3.2). We consider the active learning regime from Detlefsen et al. (2019). We split this process ten times for each experiment and repeat each experiment ten times per data set (see section 5.3.2). We bold the top performer as well as any others who are statistically indistinguishable (ρ ≥ 0.05). Tuples below experiment names are of the form (Nobservations, dim(x), dim(y)).

Active Learning. We consider the active learning regime from [Detlefsen et al., 2019]. We split each data set into 20% train, 60% reserve, and 20% test. The first active learning step utilizes the 20% training split. Thereafter, we move the n points from the reserve pool with highest predicted variance to the training set. We define n to be 5% of the original size of the reserve pool. We repeat this process ten times for each experiment and repeat each experiment ten times per data set (see

Figure 6: Log likelihoods across active learning steps for UCI data sets. The x axis is the ratio of utilized training data to the available. Darker lines are means. Areas cover ± 1 standard deviation.
We plot the log likelihoods on the held out test set across active learning steps in fig. 6. In section 5.7, we include identical plots but for MAE and RMSE as well tables that integrate these curves. From fig. 6, we find that VBEM is our best method, which makes sense given its previous top performances on these data sets. Interestingly, we find cases for all methods, but on differing data sets, where additional training data does not improve test-set performance.

4 Variational Variance for VAEs

The variational autoencoder (VAE) (Kingma and Welling, 2013) is a deep latent variable model (DLVM) that provides computationally efficient VI to describe a generative process from a low-dimensional latent variable \( z_i \) to high-dimensional data \( x_i \). As is typically done, we place a \( p(z) \triangleq \mathcal{N}(0, I) \) prior on the latent variables and perform inference by setting \( q(z|x) \triangleq \mathcal{N}(z|\mu_z(x), \sigma_z^2(x)) \), where \( \mu_z(x) \) and \( \sigma_z^2(x) \) are bifurcated outputs of the same neural network. Post-bifurcation, a softplus is applied to the variances to ensure positivity. Our goal is to improve decoder sample quality when the data \( x_i \) is modeled as Gaussian with diagonal covariance. The ELBO of the standard VAE is

\[
\mathcal{L} = \sum_{x \in D} \mathbb{E}_{q(z|x)} \left[ \log \mathcal{N}(x|\mu_x(z), \sigma_x^2(z)) \right] - D_{KL}(q(z|x) \parallel p(z)),
\]

where we define \( \mu_x(z) \) and \( \sigma_x^2(z) \) as separate neural networks to maintain a notion of consistency with the other methods we consider. We call this method VAE-Split. Takahashi et al. (2018) propose using a Student’s \( t \) likelihood to alleviate optimization instabilities. Their resulting ELBO is

\[
\mathcal{L} = \sum_{x \in D} \mathbb{E}_{q(z|x)} \left[ \log \text{Student}(x|\mu_x(z), \lambda_x(z), \nu_x(z)) \right] - D_{KL}(q(z|x) \parallel p(z)),
\]

where \( \mu_x(z), \lambda_x(z) \) and \( \nu_x(z) \) are the mean, precision, and the degrees-of-freedom networks respectively. Since the Student’s \( t \) variance is undefined for \( \nu_x(z) \in (0, 1] \) and infinite for \( \nu_x(z) \in (1, 2] \), we restrict \( \nu_x(z) > 2 \) using a shifted softplus. Allowing the posterior predictive to attain infinite variance would surely and severely deteriorate sample quality. We refer to their method as VAE-Student. Our method, V3AE (variational variance VAE) posits a prior \( p(\lambda) \) and treats precision variationally. We set \( q(\lambda|z) \triangleq \text{Gamma}(\lambda|\alpha(z), \beta(z)) \) such that our resulting ELBO is

\[
\sum_{x \in D} \mathbb{E}_{q(z|x)} \left[ \mathbb{E}_{q(\lambda|z)} \left[ \log \mathcal{N}(x|\mu_z(z), \lambda) \right] - D_{KL}(q(\lambda|\alpha(z), \beta(z)) \parallel p(\lambda)) \right] - D_{KL}(q(z|x) \parallel p(z)),
\]

which introduces a KL divergence that avoids pathological variances for reasons discussed in section 5.3.3 (Takahashi et al. 2018) additionally propose MAP-VAE where the precision is absorbed into the likelihood: \( p(\lambda|z) \triangleq \text{Gamma}(\lambda|a, b) \) for pre-defined constants \( a = 1 \) and \( b = 0.001 \) (for comparability, we use these same parameters for our standard prior). The MAP-VAE’s ELBO is identical to eq. (1) except for the additional log likelihood term and replacing \( \sigma_x^2(z) \) with a network that outputs precision. See section 5.3.4 for additional details. For V3AE, we consider only the MLE and standard priors that we discussed in section 5. We found that applying our VAMP and VBEM priors did not improve decoder sample quality beyond usage of the standard prior. Additionally, we found that Detlefsen et al. (2019)’s modified VAE produces very unappealing samples due to its tendency to output seemingly constant large decoder variances. See section 5.4 for examples of both.

Examining table 2, we see that our V3AE-Gamma (eq. (2) with a standard Gamma prior) generates the crispest samples in terms of RMSE but sacrifices log likelihood to do so. Appreciating that our precision prior effectively lower and upper bounds the predictive variance explains the log likelihood sacrifice and the sample quality improvement respectively. Our sample quality is confirmed qualitatively when comparing the fourth row of subplots in figs. 1, 7 and 8 (each subplot corresponds to a model’s best RMSE performance run). For MNIST (fig. 1) and Fashion MNIST (fig. 7), the V3AE learns a structured and highly interpretable predictive variance: it places low variance everywhere except near and around an object’s edges. Conversely, the standard VAE achieves or statistically ties the best log likelihood in each experiment, yet fails to generate appealing samples. Without an effective upper bound, its predictive variance lacks structure and places large variances across objects.
Table 2: VAE results. We report, as mean ± standard deviation, the test-set log likelihoods and the RMSE (original test data versus a true sample of the decoder) for each method across five trials.

| Data          | Method     | VAE-Split | MAP-VAE   | Student-VAE | V3AE-MLE | V3AE-Gamma |
|---------------|------------|-----------|-----------|-------------|----------|-----------|
| fashion mnist | LL         | 2250.1±94.63 | 1139.45±4.46 | 1994.53±44.75 | 1756.85±44.57 | 1048.88±22.49 |
|               | RMSE       | 0.447±0.008 | 0.156±0.001 | 39.466±8.075 | 0.367±0.005 | 0.136±0.008 |
| mnist         | LL         | 2290.14±69.42 | 1275.18±1.17 | 2353.57±10.29 | 2113.81±146.63 | 1316.19±6.84 |
|               | RMSE       | 0.528±0.089 | 0.2±0.0    | 399.929±159.227 | 0.339±0.007 | 0.166±0.004 |
| svhn cropped  | LL         | 6517.25±11.95 | 5459.7±40.17 | 6522.44±4.83 | 6401.3±58.05 | 5493.46±14.97 |
|               | RMSE       | 0.063±0.001 | 0.073±0.001 | 9.701±1.841 | 0.083±0.04 | 0.054±0.001 |

from MNIST and Fashion MNIST, which inhibits its ability to generate realistic samples. We also find the standard VAE’s predictive mean to be unappealing for MNIST and Fashion MNIST, which we suspect is an artifact of the variance network affecting the learning of the mean network. The MAP-VAE baseline performs second to ours in terms of sample quality. The Student-VAE has a very bad sample RMSE because it was still able to attain excessively large predictive variances for a subset of pixels despite our efforts to avoid such pathologies. Our plotting routine clamps pixels to [0, 1], which obfuscates the Student-VAE’s predictive sampling limitations. The sample quality of our control, V3AE-MLE, was notably worse than V3AE-Gamma, which confirms the benefit of the KL divergence. We used three dense layers for Fashion MNIST and MNIST, while employing convolution layers for SVHN. Interestingly, the difference in sample quality between the considered methods is lower for SVHN and the convolution architecture.

Figure 7: The standard VAE fails posterior predictive checks for Fashion MNIST. The rows within a subplot from top to bottom are randomly selected test data followed by the posterior predictive mean and variance and a sample from it.

Broader Impact

In this article, we propose a probabilistically principled method for improving neural-network-based variance parameterizations that broadly applies to both regression tasks as well continuous VAEs, a subset of deep latent variable models. For regression, our methods and proposed priors can accurately capture the true variance of the target and learn to output large variances for data outside the training distribution for simulated data with heteroscedastic variance. Our methods outperform the current state-of-the-art [Detlefsen et al., 2019] on their proposed regression and active learning experiments involving real-world data. Given the simplicity of our method, we believe it will cost less energy to train, be more accessible to practitioners, and thus contribute to the democratization of AI. Reliable uncertainty measures are important for many real-world machine learning systems. When we rely on such systems to intervene in economic, medical, or societal systems, we should require that the model be aware of its own predictive uncertainties. However, we strongly caution relying on our
methods without confirming they experimentally perform in the domain to which they are targeted for deployment. Also, as some of our methods require defining priors, care must be taken when designing and validating appropriate priors for the target application.

Generating realistic samples from deep latent variable models has important applications in adversarial training and data imputation. Current VAE-based methods rarely truly sample from the posterior predictive (decoder) distribution and instead report its expectation as a sample since it was generated by ancestrally sampling the latent variable that supposedly generated it. However, this sampling strategy, while it captures uncertainty on the latent variable, may fail to do so for the observed data and its posited likelihood. For continuous VAEs, our methods provide access to—for the first time to the best of our knowledge—realistic samples that are truly sampled from the posterior predictive. Alongside this ability, we find that our methods imbue appealing and interpretable structure to the posterior predictive variance. Our method could be of use in generating adversarial training examples and/or generating realistic samples for an under-represented subset of the population. If so, our methods could improve both algorithmic brittleness and fairness. Machine learning applications in medicine and computational biology, often rely on data imputation. To date, imputation with VAEs has been done with the posterior predictive distribution’s mean. Unfortunately, this decision hamstrings the diversity of imputed values. When expectations are used, imputed values lose any notion of uncertainty, which could lead to unintended and potentially deleterious side effects in the event that the downstream system relies too heavily on the imputed values. To this end, our VAE methods offer an improvement both in sample diversity and the ability to capture predictive uncertainty for continuous data. Again, we caution relying on our methods for real-world systems without careful consideration of the prior and validation of the complete system.

References

Bishop, C. M. (1994). Mixture density networks.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

Dai, B. and Wipf, D. (2019). Diagnosing and enhancing VAE models. In *International Conference on Learning Representations*.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.
Detlefsen, N., Jørgensen, M., and Hauberg, S. (2019). Reliable training and estimation of variance networks. In *Advances in Neural Information Processing Systems*, pages 6323–6333.

Figurnov, M., Mohamed, S., and Mnih, A. (2018). Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pages 441–452.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2016). Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Horvitz, D. G. and Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Loader, C. (2006). *Local regression and likelihood*. Springer Science & Business Media.

MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

Mattei, P.-A. and Frellsen, J. (2018a). Leveraging the exact likelihood of deep latent variable models. In *Advances in Neural Information Processing Systems*, pages 3855–3866.

Mattei, P.-A. and Frellsen, J. (2018b). Miwae: Deep generative modelling and imputation of incomplete data. *arXiv preprint arXiv:1812.02633*.

Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2018). Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN’94)*, volume 1, pages 55–60. IEEE.

Ranganath, R. and Blei, D. M. (2019). Population predictive checks. *arXiv preprint arXiv:1908.00882*.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822.

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Salimans, T., Knowles, D. A., et al. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264.

Takahashi, H., Iwata, T., Yamanaka, Y., Yamada, M., and Yagi, S. (2018). Student-t variational autoencoder for robust density estimation. In *IJCAI*, pages 2696–2702.
Tibshirani, R. and Hastie, T. (1987). Local likelihood estimation. *Journal of the American Statistical Association*, 82(398):559–567.

Tomczak, J. M. and Welling, M. (2017). Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

van den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.
5 Appendix

5.1 Summary of Detlefsen et al. (2019) Methods

This section summarizes the four proposed tricks of Detlefsen et al. (2019): locality sampling, isolated mean-variance training, using a Student’s $t$ likelihood, and variance extrapolation. The motivation for their locality sampler is local likelihood estimation (Tibshirani and Hastie, 1987; Loader, 2006) where for some $(x_i, y_i)$ (not necessarily in the training data $\mathcal{D}$),

$$
\log p(y_i | x_i) = \sum_{j=1}^{N} w_j(x_i) \log p(y_i | \mu(x_j), \sigma^2(x_j)),
$$

where $w_j(x_i) \propto ||x_i - x_j||^2$ to make $p(y_i | x_i)$ depend primarily on $x_j$ near $x_i$. Detlefsen et al. (2019) argue that a batch containing $x_i$, but lacking other nearby data, while sufficient for updating the mean, is insufficient for updating the variance (unless the mean is already known). They therefore propose a locality sampler that ensures any batch sample $(x_i, y_i)$ is accompanied by its $K$ nearest neighbors (w.r.t. $x_i$), which are found in pre-training. Unfortunately, nearest neighbor algorithms can produce meaningless relationships for high dimensional data (e.g. natural images). To account for the neighbors’ inclusions, they rescale the log likelihood according to Horvitz and Thompson (1952) using the mini-batch inclusion probabilities. Their separation of mean and variance optimization attempts to resolve their ill-defined variance proposition but in a simpler manner. Namely, in the first half of allotted training epochs, they fit only the mean network (with fixed variance) to ensure that, during the latter half of training, variance estimation is feasible since the mean network is presumably now reasonable. The second half of training alternates according to some pre-defined modulo of mini-batches between optimizing the mean and variance networks. Their Student’s $t$ likelihood arises from the Gamma-Normal compound parameterization $\text{Student}(y|x) \triangleq \int_0^\infty \mathcal{N}(y|\mu(x), \lambda) \text{Gamma}(\lambda|\alpha(x), \beta(x))d\lambda$ (fig. 5), which highlights that the Student’s $t$ distribution is a scaled mixture of Gaussians. In lieu of the closed-form parameterization used by Takahashi et al. (2018), their code employs Monte-Carlo integration to evaluate the resulting single-point log-likelihood objective

$$
\lambda_1, \ldots, \lambda_m \overset{iid}{\sim} \text{Gamma}(\alpha(x_i), \beta(x_i)) \rightarrow \log p(y_i | x_i) \approx \log \frac{1}{m} \sum_{j=1}^{m} \mathcal{N}(y_i | \mu(x_j), \lambda_j),
$$

which is comparatively sample inefficient. Lastly, they extrapolate the variance as a learnable convex combination between the estimated variance (inverted samples from the parameterized Gamma) and some pre-defined constant variance.

5.2 Non-Uniform VAMP Prior KL-Divergence Derivation

Here, we derive eq. (3), which assumes that $q(\lambda, c | x) \equiv q(\lambda | \alpha(x), \beta(x))q(c | \pi(x))$ and $p(\lambda, c) \triangleq p(c)q(\lambda | \alpha(u_c), \beta(u_c))$. Thus, $D_{KL}(q(\lambda, c | x) \parallel p(\lambda, c))$

$$
= \mathbb{E}_{q(\lambda, c | x)} \left[ \log q(\lambda, c | x) - \log p(\lambda, c) \right]
= \mathbb{E}_{q(\lambda, c | x)} \left[ \log q(\lambda | \alpha(x), \beta(x)) + \log q(c | \pi(x)) - \log p(c) - \log q(\lambda | \alpha(u_c), \beta(u_c)) \right]
= \left[ \log \frac{q(c | \pi(x))}{p(c)} \right] + \mathbb{E}_{q(\lambda, c | x)} \left[ \log \frac{q(\lambda | \alpha(x), \beta(x))}{q(\lambda | \alpha(u_c), \beta(u_c))} \right]
= D_{KL}(q(c | \pi(x)) \parallel p(c)) + \mathbb{E}_{q(\lambda, c | x)} \left[ D_{KL}(q(\lambda | \alpha(x), \beta(x)) \parallel q(\lambda | \alpha(u_c), \beta(u_c))) \right]
= D_{KL}(q(c | \pi(x)) \parallel p(c)) + \sum_{j=1}^{K} \pi_j(x) \mathbb{E}_{q(\lambda | x)} \left[ \log q(\lambda | \alpha(u_j), \beta(u_j)) \right]
$$

(7)

5.3 Implementation Details

Our code can be found at https://github.com/astirn/VariationalVariance As noted in the README, we use CUDA acceleration for our experiments. Our models are relatively small, and
we found that any modern laptop with an Nvidia GPU should suffice. In addition to our personal
computers, we parallelized our many experiments both on our laptops as well as on an Nvidia P4
GPU cloud instance. We note that laptops actually had higher CUDA compute capabilities than
the utilized cloud infrastructure.

5.3.1 Toy Regression Implementation Details

Here, we explicate the implementation details used in the toy regression experiments. For all methods,
we use neural networks with a single hidden layer of 50 sigmoid neurons to match [Detlefsen et al.
(2019)]. Knowing the true mean a-priori allows us to learn an MLE distribution over the inverted
squared errors. We use the resulting MLE parameters from the appropriate distribution to set \( a \) and
\( b \) for our standard priors. For our VAMP, VAMP* and VBEM priors, we set \( K = 20 \). For VAMP
and VAMP*, we sample pseudo-inputs \( u_c \sim \text{Uniform}([-4, 14]) \). To ensure valid and continuously
differentiable parameters for VBEM, we randomly initialize \( u_c, v_c \sim \text{Uniform}([-3, 3]) \) and define
\( a_c \triangleq \text{softplus}(u_c) \) and \( b_c \triangleq \text{softplus}(v_c) \). Like [Detlefsen et al. (2019)], we use ADAM (Kingma
and Ba, 2014) for optimization. As mentioned in section 5.3.1, Detlefsen et al. (2019) employ separate
optimizers for the mean and variance networks that respectively use 1e-2 and 1e-3 as learning rates.
We employ a single ADAM instance with a learning rate of 1e-2. We run all algorithms for 6e3
epochs without mini-batching (i.e. the single batch contains all 500 training points).

5.3.2 UCI Regression Implementation Details

Here, we detail the experimental specifics used for the supervised regression experiments on the UCI
data sets. Again, we match the [Detlefsen et al. (2019)] network architecture, which now uses a single
hidden layer with 50 ReLU neurons. The baseline code (Detlefsen et al. 2019) allows training to run
for some number of mini-batch iterations, whereas our code uses the notion of an epoch, which
encompasses the number of mini-batches required to see each example in the training set exactly once.
To keep things equal, we allow each algorithm to run for 20e4 mini-batch iterations with a batch
size of 512, which we convert to epochs \( \left( \frac{\text{iterations}}{\text{batch size}} \right) \) for our methods. We originally considered 10e3
iterations (as used by [Detlefsen et al. (2019)]), but found our algorithms and theirs were not reaching
convergence, which we detail further in section 5.6. Additionally, we observed that our proposed
priors could tolerate different learning rates, which we independently tuned for speed. We use 1e-4
for our MLE, VAMP, and VAMP* priors. We use 5e-4 for VBEM priors and 1e-3 for standard priors.
For UCI data, [Detlefsen et al. (2019)] use 1e-2 and 1e-4 as learning rates for the mean and variance
networks respectively. We set the standard prior to ensure unit-moments for the first two moments
\( (a = 1 \text{ and } b = 1 \) for the Gamma prior and \( a = \frac{\log 2}{2} \) and \( b = \sqrt{\log 2} \) for the LogNormal prior). For the VAMP and VAMP*
priors, we sample \( K = 100 \) pseudo-inputs uniformly from the training set without replacement. We initialize VBEM
priors in the same manner discussed in section 5.3.1 but now with \( K = 100 \). We employ early stopping on test-set log likelihood with a patience of 500
epochs. We implemented an equivalent early stopping mechanism in the baseline code (Detlefsen
et al. 2019), in which we also introduced support for multivariate target variables.

5.3.3 UCI Active Learning Implementation Details

For the active learning regression experiments, we preserve the implementations details from section
5.3.2. The only difference is that we grow \( K \) proportional to the ratio of utilized training data
to total available. Specifically, we multiply \( K = 100 \) by this ratio at each active learning step to set
the number of mixture components. We identically scale the maximum allowed 20e4 mini-batch
iterations at each active learning step.

5.3.4 VAE Implementation Details

For the VAE experiments, we use ADAM with a learning rate of 1e-4 and a batch size of 250. We
allow all algorithms to train for 1e3 epochs, but employ early stopping on test-set log likelihood with
a patience of 50 epochs. For MNIST and Fashion MNIST, we use an encoder architecture with hidden
layers of sizes 512, 256, and 128, each of which applies an ELU activation. The decoder architecture
is the transpose of the encoder. For SVHN, we employ convolution layers. The encoder sequentially
employs a convolution kernel of size \( 32 \times 5 \times 5 \), a max pool kernel of size \( 3 \times 3 \) with stride length 2,
a convolution kernel of size \( 64 \times 3 \times 3 \), a max pool kernel of size \( 3 \times 3 \) with stride length 2, and
a dense layer with 128 outputs. We employ ELU activations after the max pool and dense layers.
The convolution-based decoder is the transpose of the encoder with bi-linear up-sampling. The dimensions of the latent variable \( \dim(z) \) are 10, 25, and 50 respectively for the MNIST, Fashion MNIST, and SVHN data sets.

5.4 Things that Failed

Uniform Variational Mixture of Posteriors Prior. In section 2, we mentioned that the original VAMP prior, which assumes a uniform mixture of variational posteriors, fails to accurately capture the true variance for heteroscedastic processes. We first noticed this shortcoming when fitting our proposed models to the simulated heteroscedastic process of Detlefsen et al. (2019), which we detail in section 3. Figures 9 and 10 show the variational distributions failing to capture heteroscedasticity when respectively using Gamma and LogNormal variational families for precision. Instead, the variational distributions falsely predict homoscedasticity by learning a constant \( \text{std}(y|x) \approx 2 \), which is almost exactly equal to the expected \( \text{std}(y|x) \triangleq |0.3 \cdot (1 + x)| \) over the training data range of \( x \in [0, 10] \) given that \( x \sim \text{Uniform}(0, 10) \).

\[ \text{Figure 9: Variance estimation using Gamma distributed precision with a uniform VAMP prior.} \]

\[ \text{Figure 10: Variance estimation using LogNormal distributed precision with a uniform VAMP prior.} \]

VAE Methods with Unappealing Decoder Samples. In fig 11, we see examples of the VAE proposed by Detlefsen et al. (2019) as well as our V3AE with a VBEM prior producing unappealing decoder samples. The Detlefsen et al. (2019) VAE produced variances > 1, such that decoder sample quality for images with pixels in \( [0, 1] \) is abysmal. Our V3AE with a VBEM prior is comparatively better, but was not competitive with our V3AE-Gamma (standard prior). Comparing fig. 11 with fig. 1, one will find that variance for VBEM lacks the structure achieved by the standard prior (V3AE-Gamma).

\[ \text{Figure 11: MNIST PPC for the Detlefsen et al. (2019) VAE and our V3AE with a VBEM prior.} \]

5.5 Additional UCI Regression Results

Here, we provide additional results for the supervised regression task outlined in section 3. Table 3 contains results for our log LogNormal treatment of precision, which is supplemental to table 1 in the main body of our manuscript. In this section, we also report RMSE and MAE values for these same experiments in tables 4 and 5. Note that we employed early stopping and report test-set log likelihoods, RMSEs, and MAEs from the epoch with highest test-set log likelihood.
Table 3: UCI regression log likelihoods reported as mean±std. We bold the top performer as well as any others who are statistically indistinguishable (p ≥ 0.05). Tuples below experiment names are of the form (Nobservations, dim(x), dim(y)).

| Algorithm       | Prior | power plant | superconductivity | wine-red | wine-white | yacht |
|-----------------|-------|-------------|-------------------|----------|------------|-------|
| N/A             |       | (9568, 4, 1) | (21263, 81, 1)    | (1599, 11, 1) | (4898, 11, 1) | (308, 6, 1) |
| Gamma-Normal    | MLE   | 2.344±0.008 | 5.027±0.014       | -0.91±0.07 | -1.056±0.05 | -2.925±0.025 |
| VAMP            |       | -2.786±0.043 | -3.64±0.123       | -0.98±0.11 | -1.055±0.066 | -2.066±0.049 |
| VAMP*           |       | -2.777±0.037 | -3.61±0.081       | -0.976±0.081 | -1.052±0.059 | -1.927±0.035 |
| LogNormal-Normal | MLE   | -2.777±0.037 | -3.61±0.081       | -0.976±0.081 | -1.052±0.059 | -1.927±0.035 |
| VAMP            |       | -2.777±0.037 | -3.61±0.081       | -0.976±0.081 | -1.052±0.059 | -1.927±0.035 |
| VAMP*           |       | -2.777±0.037 | -3.61±0.081       | -0.976±0.081 | -1.052±0.059 | -1.927±0.035 |

Table 4: UCI regression RMSE reported identically as above table.

| Algorithm       | Prior | power plant | superconductivity | wine-red | wine-white | yacht |
|-----------------|-------|-------------|-------------------|----------|------------|-------|
| N/A             |       | (9568, 4, 1) | (21263, 81, 1)    | (1599, 11, 1) | (4898, 11, 1) | (308, 6, 1) |
| Gamma-Normal    | MLE   | 2.792±0.65  | 0.014±0.006       | 4.987±0.529 | 1.634±0.189 | 0.005±0.0 |
| VAMP            |       | 3.014±0.132 | 0.013±0.007       | 6.404±0.712 | 2.008±0.275 | 0.004±0.001 |
| VAMP*           |       | 3.014±0.132 | 0.013±0.007       | 6.404±0.712 | 2.008±0.275 | 0.004±0.001 |
| LogNormal-Normal | MLE   | 3.014±0.132 | 0.013±0.007       | 6.404±0.712 | 2.008±0.275 | 0.004±0.001 |
| VAMP            |       | 3.014±0.132 | 0.013±0.007       | 6.404±0.712 | 2.008±0.275 | 0.004±0.001 |
| VAMP*           |       | 3.014±0.132 | 0.013±0.007       | 6.404±0.712 | 2.008±0.275 | 0.004±0.001 |

15
we collected UCI regression results using 10k iterations for all considered methods, we include them. The baseline code from Detlefsen et al. (2019) uses only 10e3 mini-batch iterations. We found this both iteration settings employed early stopping and we report test-set log likelihoods, RMSEs, and number of iterations too short for convergence not only for our models but also theirs. However, since

| Algorithm          | Prior                  | boston (596, 4, 1) | carbon (10721, 5, 3) | concrete (1030, 8, 1) | energy (768, 8, 2) | naval (11934, 16, 2) |
|--------------------|------------------------|--------------------|----------------------|-----------------------|--------------------|----------------------|
| Detlefsen          | N/A                    | 0.977 ± 0.331      | 0.007 ± 0.001        | 3.719 ± 0.321         | 1.17 ± 0.156       | 0.003 ± 0.001        |
| Gamma-Normal       | MLE                    | 2.596 ± 0.633      | 0.004 ± 0.003        | 4.714 ± 0.526         | 1.376 ± 0.166      | 0.003 ± 0.001        |
| LogNormal-Normal   | MLE                    | 2.022 ± 0.259      | 0.004 ± 0.001        | 3.244 ± 0.404         | 1.016 ± 0.249      | 0.00 ± 0.00          |
| VAMP               | 2.438 ± 0.562          | 0.003 ± 0.002      | 4.123 ± 0.589        | 1.29 ± 0.187          | 0.003 ± 0.001      
| VAMP*              | 2.432 ± 0.622          | 0.003 ± 0.002      | 4.156 ± 0.673         | 1.303 ± 0.144       | 0.003 ± 0.001      
| VBM                | 2.183 ± 0.421          | 0.003 ± 0.001      | 3.811 ± 0.531         | 1.151 ± 0.162        | 0.00 ± 0.00         
| VBEM               | 2.662 ± 0.681          | 0.004 ± 0.003      | 4.716 ± 0.422         | 1.34 ± 0.163        | 0.003 ± 0.001      |

5.6 10k Iteration UCI Regression Results

The baseline code from Detlefsen et al. (2019) uses only 10e3 mini-batch iterations. We found this number of iterations too short for convergence not only for our models but also theirs. However, since we collected UCI regression results using 10k iterations for all considered methods, we include them here for reference. Respectively comparing the log likelihood table for 10e3 mini-batch iterations (table[4]) to the same table but for 20e4 mini-batch iteration (table[5]) shows that convergence was not achieved for some data sets, especially those with more training examples. We note however that both iteration settings employed early stopping and we report test-set log likelihoods, RMSEs, and MAEs from the epoch with highest test-set log likelihood.

Table 5: UCI regression MAE reported identically as above table.

| Algorithm          | Prior                  | power plant (9568, 4, 1) | superconductivity (10263, 81, 1) | wine-red (1599, 11, 1) | wine-white (4898, 11, 1) | yacht (308, 6, 1) |
|--------------------|------------------------|--------------------------|----------------------------------|------------------------|--------------------------|-------------------|
| Detlefsen          | N/A                    | 3.196 ± 0.078            | 11.917 ± 0.352                   | 0.469 ± 0.035          | 0.543 ± 0.028           | 0.388 ± 0.119     |
| Gamma-Normal       | MLE                    | 3.028 ± 0.084            | 10.105 ± 0.449                   | 0.483 ± 0.035          | 0.556 ± 0.029           | 0.297 ± 0.707     |
| LogNormal-Normal   | MLE                    | 3.021 ± 0.067            | 8.199 ± 0.309                    | 0.469 ± 0.036          | 0.534 ± 0.033           | 0.441 ± 0.111     |
| VAMP               | 3.005 ± 0.068          | 9.443 ± 0.388            | 0.478 ± 0.031                    | 0.535 ± 0.028          | 2.42 ± 0.656            |
| VBM                | 2.967 ± 0.069          | 8.807 ± 0.338            | 0.471 ± 0.034                    | 0.528 ± 0.027          | 0.418 ± 0.118          |
| VBP                | 3.024 ± 0.086          | 9.954 ± 0.359            | 0.481 ± 0.036                    | 0.542 ± 0.029          | 2.548 ± 0.783          |
| VBM                | 3.005 ± 0.051          | 8.137 ± 0.319            | 0.473 ± 0.034                    | 0.532 ± 0.027          | 0.487 ± 0.127          |

Table 6: UCI regression LL for 10,000 batch iterations. This table can be compared to table[5]
In this section we include additional metrics for our active learning experiments. Figures 12 and 13 respectively plot RMSE and MAE for 10,000 batch iterations and are complimentary to fig. 9 (from the main body of this manuscript), which plots log likelihood. Tables 9 to 11 respectively report the integrated curves from figs. 6, 12 and 13 in an effort to distill active learning performance into a single scalar.
Figure 12: RMSE across active learning steps for UCI data sets. The $x$ axis is the ratio of utilized training data to the available. Darker lines are means. Areas cover ±1 standard deviation.

Figure 13: MAE across active learning steps for UCI data sets. The $x$ axis is the ratio of utilized training data to the available. Darker lines are means. Areas cover ±1 standard deviation.

Table 9: UCI cumulative sum of log likelihood across active learning steps reported as mean±std. We bold the top performer as well as any others who are statistically indistinguishable ($p \geq 0.05$). Tuples below experiment names are of the form $(N_{\text{observations}}, \dim(x), \dim(y))$.

| Algorithm | Prior | power plant | superconductivity | concrete | energy | naval |
|-----------|-------|-------------|-------------------|----------|--------|-------|
| Detlefsen | N/A   | -32.249±0.052 | -50.584±0.142    | -9.095±0.355 | -10.811±0.284 | -30.8±0.767 |
| Gamma-Normal | MLE   | -27.978±0.249 | -38.673±0.455    | -10.657±0.852 | -11.199±0.392 | -30.483±3.939 |
| Standard | -38.029±0.086 | -45.485±0.063 | -10.679±0.296    | -12.222±0.156 | -35.967±0.833 |
| VAMP     | -28.022±0.174 | -38.070±0.436 | -9.695±0.699     | -10.8±0.401  | -28.104±3.819 |
| VAMP*    | -28.021±0.184 | -37.993±0.416 | -9.819±0.681     | -10.786±0.38  | -28.618±3.992 |
| VBEM     | -27.999±0.172 | -38.012±0.383 | -9.812±0.609     | -11.081±0.338 | -19.683±3.08 |

- 0.3 0.4 0.5 0.6
- 0.64 0.66 0.68 0.70 0.72 0.74 0.76
- 0.3 0.4 0.5 0.6
- 0.184 -37.993 ± 0.691 -33.745 ± 1.001 -21.242 ± 0.852 -12.222 ± 0.296 -35.967 ± 0.833 -28.618 ± 3.992 -19.683 ± 3.08
Table 10: UCI cumulative sum of RMSE across active learning steps reported as mean±std.

| Algorithm | Prior | boston (506, 13, 1) | carbon (10721, 5, 3) | concrete (1030, 8, 1) | energy (768, 2, 7) | naval (11934, 16, 2) |
|-----------|-------|---------------------|----------------------|------------------------|-------------------|---------------------|
| Detlefsen | N/A   | 26.254±4.149        | 0.164±0.037          | 46.54±1.427            | 16.31±1.331       | 0.048±0.001         |
| Gamma-Normal | MLE   | 44.598±10.374       | 0.162±0.049          | 72.524±5.225           | 21.051±1.165      | 0.063±0.001         |
| Standard  | 29.499±5.642      | 0.159±0.037          | 48.724±2.102         | 16.874±0.969          | 0.006±0.0         |
| VAMP      | 38.774±11.191     | 0.156±0.04           | 61.18±3.74           | 20.202±1.264         | 0.063±0.002       |
| VAMP*     | 37.623±9.804      | 0.156±0.041          | 61.766±3.754         | 20.489±1.477         | 0.063±0.002       |
| VBEM      | 32.428±6.864      | 0.153±0.04           | 53.518±4.423         | 16.027±1.554         | 0.006±0.001       |

Table 11: UCI cumulative sum of MAE across active learning steps reported as mean±std.

| Algorithm | Prior | power plant (1956, 4, 1) | superconductivity (21263, 81, 1) | wine-red (1599, 11, 1) | wine-white (4898, 11, 1) | yacht (308, 6, 1) |
|-----------|-------|--------------------------|----------------------------------|------------------------|--------------------------|------------------|
| Detlefsen | N/A   | 40.522±0.749             | 162.57±2.019                     | 6.151±0.23             | 7.078±0.168              | 6.911±1.296      |
| Gamma-Normal | MLE   | 39.778±0.778             | 151.197±2.373                    | 6.096±0.183            | 7.001±0.216              | 69.532±57.812   |
| Standard  | 39.471±0.7          | 122.045±2.17            | 5.63±0.217                      | 6.733±0.155            | 13.0±2.955             |
| VAMP      | 39.816±0.736        | 143.691±2.715           | 5.885±0.219                     | 6.88±0.206             | 51.342±15.081         |
| VAMP*     | 39.82±0.76          | 143.695±2.638           | 5.931±0.226                     | 6.868±0.198            | 53.253±16.576        |
| VBEM      | 39.1±0.668         | 134.821±2.558           | 5.857±0.22                     | 6.823±0.162            | 18.353±5.575        |

| Algorithm | Prior | power plant (9568, 4, 1) | superconductivity (21263, 81, 1) | wine-red (1599, 11, 1) | wine-white (4898, 11, 1) | yacht (308, 6, 1) |
|-----------|-------|--------------------------|----------------------------------|------------------------|--------------------------|------------------|
| Detlefsen | N/A   | 16.849±2.171             | 0.065±0.005                      | 34.535±1.031           | 11.559±0.941             | 0.035±0.001     |
| Gamma-Normal | MLE   | 29.53±5.44               | 0.056±0.027                      | 55.147±4.991           | 15.081±1.148             | 0.053±0.001     |
| Standard  | 18.32±1.781        | 0.053±0.004               | 33.783±1.132                    | 11.034±0.936           | 0.004±0.0        |
| VAMP      | 25.605±5.548       | 0.04±0.011                | 45.901±2.382                    | 14.432±1.211           | 0.053±0.002      |
| VAMP*     | 24.989±5.0         | 0.043±0.013               | 46.437±2.818                    | 14.645±1.499           | 0.053±0.002      |
| VBEM      | 20.922±3.302       | 0.031±0.004               | 39.006±2.491                    | 10.105±1.038           | 0.004±0.0        |

| Algorithm | Prior | power plant (9568, 4, 1) | superconductivity (21263, 81, 1) | wine-red (1599, 11, 1) | wine-white (4898, 11, 1) | yacht (308, 6, 1) |
|-----------|-------|--------------------------|----------------------------------|------------------------|--------------------------|------------------|
| Detlefsen | N/A   | 31.678±0.504             | 115.973±1.778                    | 4.455±0.146            | 5.428±0.125              | 4.187±0.619     |
| Gamma-Normal | MLE   | 30.779±0.429             | 103.861±1.818                    | 4.724±0.13             | 5.422±0.169              | 42.414±10.5     |
| Standard  | 30.494±0.309        | 85.046±1.55               | 4.191±0.144                     | 5.173±0.123            | 6.813±1.176             |
| VAMP      | 30.847±0.335        | 97.27±2.134               | 4.531±0.143                     | 5.312±0.15             | 30.883±8.464           |
| VAMP*     | 30.86±0.368         | 97.061±2.002              | 4.569±0.146                     | 5.314±0.148            | 32.489±9.674         |
| VBEM      | 30.048±0.286        | 90.867±1.875              | 4.512±0.152                     | 5.288±0.129            | 9.054±2.47         |