Batch, match, and patch: low-rank approximations for score-based variational inference

Chirag Modi[*]
Center for Computational Mathematics, Flatiron Institute
Center for Cosmology and Particle Physics, New York University

Diana Cai[*]
Center for Computational Mathematics, Flatiron Institute

Lawrence K. Saul
Center for Computational Mathematics, Flatiron Institute

Abstract

Black-box variational inference (BBVI) scales poorly to high-dimensional problems when it is used to estimate a multivariate Gaussian approximation with a full covariance matrix. In this paper, we extend the *batch-and-match* (BaM) framework for score-based BBVI to problems where it is prohibitively expensive to store such covariance matrices, let alone to estimate them. Unlike classical algorithms for BBVI, which use stochastic gradient descent to minimize the reverse Kullback-Leibler divergence, BaM uses more specialized updates to match the scores of the target density and its Gaussian approximation. We extend the updates for BaM by integrating them with a more compact parameterization of full covariance matrices. In particular, borrowing ideas from factor analysis, we add an extra step to each iteration of BaM—a *patch*—that projects each newly updated covariance matrix into a more efficiently parameterized family of diagonal plus low rank matrices. We evaluate this approach on a variety of synthetic target distributions and real-world problems in high-dimensional inference.

1 Introduction

Many statistical analyses can be formulated as problems in Bayesian inference. In this formulation, the goal is to compute the posterior distribution over latent random variables from observed ones. But difficulties arise when the latent space is of very high dimensionality, D. In this case, even if the underlying models are simple—e.g., Gaussian—it can be prohibitively expensive to store let alone estimate a $D \times D$ covariance matrix.

These difficulties are further compounded when the underlying models are not Gaussian. In this case, though, a multivariate Gaussian can sometimes be used to approximate a posterior distribution that is otherwise intractable. This idea is the basis of variational inference (VI) (Jordan et al., 1999; Wainwright et al., 2008; Blei et al., 2017), which in this setting seeks the Gaussian approximation with the lowest divergence to the posterior. There are well-developed software packages to compute these approximations that make very few assumptions about the form of the posterior distribution; hence this approach is known as black-box variational inference (BBVI) (Ranganath et al., 2014; Kingma and Welling, 2014; Titsias and Lázaro-Gredilla, 2014; Kucukelbir et al., 2017). BBVI, however, also scales poorly to high-dimensional problems when it is used to estimate a multivariate Gaussian approximation with a full covariance matrix (Ko et al., 2024).

One hopeful approach is to parameterize a large covariance matrix as the sum of a diagonal matrix and a low-rank matrix. This idea is used in maximum likelihood models for factor analysis, where the diagonal and low-rank matrices can be estimated by an Expectation-Maximization (EM) algorithm (Rubin and Thayer, 1982; Ghahramani and Hinton, 1996; Saul and Rahim, 2000). This parameterization can also be incorporated into BBVI algorithms that use stochastic gradient methods to minimize the reverse Kullback-Leibler (KL) divergence; in this case, the required gradients can be computed automatically by the chain rule (Miller et al., 2017; Ong et al., 2018). This approach overcomes the poor scaling with respect to the dimensionality of the latent space, but it still relies on the heuristics and hyperparameters of stochastic

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. [*]Equal contribution

gradient methods. In particular, the optimization itself does not leverage in any special way the structure of factored covariance matrices.

In this paper, we extend the *batch-and-match* (BaM) framework for score-based BBVI (Cai et al., 2024b) to these problems. Unlike classical algorithms for BBVI, BaM does not use gradient descent to minimize a KL divergence; it is based on more specialized updates to match the scores of the target density and its Gaussian approximation. We augment these original updates for BaM with an additional step—a *patch*—that projects each newly updated covariance matrix into a more efficiently parameterized family of diagonal plus low rank matrices. This patch is closely based on the EM updates for maximum likelihood factor analysis, and the resulting algorithm scales much better to problems of high dimensionality. We refer to this variant of the algorithm as *patched batch-and-match* (pBaM)[1].

The organization of this paper is as follows. In Section 2, we review the basic ideas of BBVI and BaM. In Section 3, we derive the patch for BaM with structured covariance matrices. In Section 4, we summarize related work on these ideas. In Section 5, we evaluate the pBaM algorithm on a variety of synthetic target distributions and real-world problems in high-dimensional inference. Finally, in Section 6, we conclude and discuss directions for future work.

2 Score-based variational inference

In this section, we provide a brief review of BBVI, score-based VI, and the BaM algorithm for score-based BBVI with multivariate Gaussian approximations.

2.1 Black-box variational inference

The goal of VI is to approximate an intractable target density p by its closest match q in a variational family $\mathcal{Q}$. The goal of BBVI is to compute this approximation in a general way that makes very few assumptions about the target p. Typically, BBVI assumes only that it is possible to evaluate the *score* of p (i.e., $\nabla_z \log p(z)$) at any point z in the latent space.

As originally formulated, BBVI attempts to find the best approximation $q \in \mathcal{Q}$ by minimizing the reverse KL divergence, $\mathrm{KL}(q||p)$, or equivalently by maximizing the evidence lower bound (ELBO). One popular approach for ELBO maximization is the automatic differentiation framework for variational inference (ADVI), which is based on stochastic gradient-based optimization. ADVI (Kucukelbir et al., 2017), which uses the reparameterization trick to cast the gradient of the ELBO as an expectation, constructs a stochastic estimate of this gradient. The main strength of this approach is its generality: it can be used with any variational family that lends itself to reparameterization. But this generality also comes at a cost: the approach may converge very slowly for richer families of variational approximations—including, in particular, the family of multivariate Gaussian densities with full (dense) covariance matrices.

2.2 Score-based approaches to BBVI

Recently, several algorithms have been proposed for BBVI that aim to match the scores of the target density and its variational approximation (Yang et al., 2019; Zhang et al., 2018; Modi et al., 2023; Cai et al., 2024b,a). Some of these score-based approaches are specially tailored to Gaussian variational families (Modi et al., 2023; Cai et al., 2024b), and they exploit the particular structure of Gaussian distributions to derive more specialized updates for score-matching.

These approaches find the best Gaussian approximation by minimizing a weighted score-based divergence:

$$\mathscr{D}(q;p) := \int \|\nabla \log q(z) - \nabla \log p(z)\|^2_{\mathrm{Cov}(q)} \; q(z)\, dz, \tag{2.1}$$

where $\|x\|_\Sigma := \sqrt{x^\top \Sigma x}$ and q is assumed to belong to the variational family $\mathcal{Q} := \{\mathcal{N}(\mu, \Sigma) : \mu \in \mathbb{R}^D, \Sigma \in \mathbb{S}^D_{++}\}$ of multivariate Gaussian distributions.

One of these approaches is the batch-and-match (BaM) algorithm for score-based VI. BaM uses a KL-regularized proximal point update to optimize the above divergence. BaM avoids certain heuristics of gradient-based optimization because its proximal point update can be solved in closed form. The BaM updates reduce as a special case to an earlier algorithm for Gaussian score matching (Modi et al., 2023); the latter is equivalent to BaM with a batch size of one and an infinite learning rate. Notably, both of these approaches have been empirically observed to converge faster than ADVI with a full covariance. The "patch" step in this paper can be applied both to Gaussian score matching and BaM, but we focus on the latter because it contains the former as a special case.

2.3 Batch-and-match variational inference

We now review the BaM framework for BBVI in more detail. BaM iteratively updates its variational parameters by minimizing an objective with two terms: one is a score-based divergence between q and p, estimated from a batch of B samples, while the other is a regularizer

[1] A Python implementation of pBaM is available at https://github.com/modichirag/GSM-VI/.

that penalizes overly aggressive updates. Specifically, the update is given by

$$q_{t+1} = \arg\min_{q\in\mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;p) + \tfrac{2}{\lambda_t}\mathrm{KL}(q_t;q), \tag{2.2}$$

where $\widehat{\mathscr{D}}_{q_t}(q;p)$ is a stochastic estimate of the score-based divergence in Eq. 2.1 based on B samples $\{z_b \sim q_t\}_{b=1}^B$. We compute this stochastic estimate as

$$\widehat{\mathscr{D}}_{q_t}(q;p) := \sum_{b=1}^{B} \big\|\nabla \log q(z_b) - \nabla \log p(z_b)\big\|^2_{\mathrm{Cov}(q)}. \tag{2.3}$$

Note also that the prefactor of the KL regularizer in Eq. 2.2 contains a *learning rate* parameter $\lambda_t > 0$.

The optimization in Eq. 2.2 has a closed-form solution, and this solution forms the basis for BaM. Each iteration of BaM alternates between a *batch step* and a *match step*. The *batch step* collects B samples $\{z_b \sim q_t\}_{b=1}^B$, evaluates their scores $g_b := \nabla \log p(z_b)$, and then computes the following statistics:

$$\overline{z} = \tfrac{1}{B}\sum_{b=1}^{B} z_b, \qquad C = \tfrac{1}{B}\sum_{b=1}^{B}(z_b - \overline{z})(z_b - \overline{z})^\top \tag{2.4}$$

$$\overline{g} = \tfrac{1}{B}\sum_{b=1}^{B} g_b, \qquad \Gamma = \tfrac{1}{B}\sum_{b=1}^{B}(g_b - \overline{g})(g_b - \overline{g})^\top. \tag{2.5}$$

Finally, the batch step uses these statistics to compute the empirical score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;p)$ in Eq. 2.3. We note for future reference that the matrices C and Γ in these statistics are constructed as the sum of B rank-1 matrices; thus *the ranks of C and Γ are less than or equal to the batch size B.*

The *match step* of BAM computes updated variational parameters Σ_{t+1} and μ_{t+1} that minimize the score-matching objective in Eq. 2.3. In what follows we present the updates for $B < D$, which is the relevant regime for high-dimensional problems. We begin by introducing two intermediate matrices U and V, defined as

$$U := \lambda_t\Gamma + \tfrac{\lambda_t}{1+\lambda_t}\,\overline{g}\,\overline{g}^\top \tag{2.6}$$

$$V := \Sigma_t + \lambda_t C + \tfrac{\lambda_t}{1+\lambda_t}(\mu_t - \overline{z})(\mu_t - \overline{z})^\top. \tag{2.7}$$

In fact, it is not necessary to store the dense $D \times D$ matrix U; it suffices to work with any tall and thin matrix $Q \in \mathbb{R}^{D\times(B+1)}$ satisfying

$$U = QQ^\top. \tag{2.8}$$

One solution for Q satisfying the above can be immediately deduced from the expressions for Γ and U in Eqs. 2.5 and 2.6; in particular, it is obtained by concatenating the mean and centered scores of the batch:

$$Q = \left[\sqrt{\tfrac{1}{B}}(g_1 - \overline{g}),\ \ldots,\ \sqrt{\tfrac{1}{B}}(g_B - \overline{g}),\ \sqrt{\tfrac{\lambda_t}{1+\lambda_t}}\,\overline{g}\right]. \tag{2.9}$$

Finally, in terms of these matrices, BaM updates the covariance as

$$\Sigma_{t+1} = V - V^\top Q\big[\tfrac{1}{2}I + \big(Q^\top V Q + \tfrac{1}{4}I\big)^{\frac{1}{2}}\big]^{-2} Q^\top V, \tag{2.10}$$

where the thinness of Q helps to perform this update more efficiently. After this covariance update, BaM updates the mean as

$$\mu_{t+1} = \tfrac{1}{1+\lambda_t}\mu_t + \tfrac{\lambda_t}{1+\lambda_t}\left(\Sigma_{t+1}\,\overline{g} + \overline{z}\right), \tag{2.11}$$

where Σ_{t+1} is the update in Eq. 2.10. The new parameters Σ_{t+1} and μ_{t+1} from these updates in turn define the new variational approximation q_{t+1}.

BaM differs from previous approaches to BBVI that optimize a KL-based objective using gradient descent. Gradient-based approaches rely at each iteration on an implicit linearization of the objective function that they are trying to optimize. By contrast, each BaM iteration constructs a proximal-point objective function whose global optimum can be computed in closed form and without resorting to an implicit linearization.

Finally, we note that the BaM update in Eq. 2.10 takes $\mathcal{O}(D^2B + B^3)$ computation for dense covariance matrices Σ_t. Since this update scales quadratically in D, it may be too expensive to compute for very high dimensional problems. (In fact, it may even be too expensive to store the covariance matrix Σ_t.) In the next section, we introduce a *patch* at each iteration of BaM to avoid this quadratic scaling.

3 The patch: a low-rank plus diagonal covariance matrix

In this section, we consider how BaM can be adapted to a Gaussian variational family whose covariance matrices Σ are parameterized as the sum of a low-rank matrix and a diagonal matrix: namely,

$$\Sigma = \Lambda\Lambda^\top + \Psi. \tag{3.1}$$

In this parameterization, Λ is a $D \times K$ matrix with rank $K \ll D$, and Ψ is a $D \times D$ diagonal matrix. For Gaussian distributions whose covariance matrices are of this form, we can evaluate the log-density with $\mathcal{O}(D)$ computation using the Woodbury matrix identity. We can also generate samples efficiently as follows (Miller et al., 2017; Ong et al., 2018):

$$\varepsilon \sim \mathcal{N}(0, I_D), \quad \zeta \sim \mathcal{N}(0, I_K), \quad z = \mu + \Lambda\zeta + \sqrt{\Psi}\varepsilon.$$

Algorithm 1 Batch, match, and patch BBVI

Input: Iterations T, batch size B, inverse regularization $\lambda_t > 0$, target score function $s : \mathbb{R}^D \to \mathbb{R}^D$, initial variational mean $\mu_0 \in \mathbb{R}^D$ and covariance matrix $\Sigma_0 = \Psi_0 + \Lambda_0\Lambda_0^\top$.
for $t = 0, \ldots, T-1$ **do**
  Sample batch $z_b \sim \mathcal{N}(\mu_t, \Psi_t + \Lambda_t\Lambda_t)$ for $b = 1, \ldots, B$
  Evaluate scores $g_b = s(z_b)$ for $b = 1, \ldots, B$
  Compute the batch mean statistics $\overline{z}, \overline{g} \in \mathbb{R}^D$ given by

$$\overline{z} = \frac{1}{B}\sum_{b=1}^{B} z_b \quad \text{and} \quad \overline{g} = \frac{1}{B}\sum_{b=1}^{B} g_b.$$

  Compute the intermediate matrices

$$Q = \left[\sqrt{\tfrac{1}{B}}(g_1 - \overline{g}),\ \ldots, \sqrt{\tfrac{1}{B}}(g_B - \overline{g}),\ \sqrt{\tfrac{\lambda_t}{1+\lambda_t}}\,\overline{g}\right],$$

$$R = \left[\sqrt{\tfrac{\lambda_t}{B}}(z_1 - \overline{z}), ..., \sqrt{\tfrac{\lambda_t}{B}}(z_B - \overline{z}), \sqrt{\tfrac{\lambda_t}{1+\lambda_t}}(\mu_t - \overline{z}), \Lambda_t\right],$$

$$V = \Psi_t + RR^\top.$$

  Update variational parameters ("match" and "patch")

$$\Sigma_{t+\frac{1}{2}} = V - V^\top Q\left[\tfrac{1}{2}I + \left(Q^\top V Q + \tfrac{1}{4}I\right)^{\frac{1}{2}}\right]^{-2} Q^\top V,$$

$$\Sigma_{t+1} = \texttt{Patch}\left[\Sigma_{t+\frac{1}{2}}\right]$$

$$\mu_{t+1} = \tfrac{1}{1+\lambda_t}\mu_t + \tfrac{\lambda_t}{1+\lambda_t}\left(\Sigma_{t+1}\,\overline{g} + \overline{z}\right)$$

end for
Output: mean μ_T and covariance parameters Ψ_T, Λ_T

Algorithm 2 `Patch` step

Input: Dense covariance $\Sigma_{t+\frac{1}{2}}$ (represented implicitly by Ψ_t, R and Q), initial parameters Λ_0, Ψ_0, momentum η, tolerance ε, maximum number of steps N
$\mathrm{KL}_0 = \log(|\Lambda_0\Lambda_0^\top + \Psi_0|) + \mathrm{tr}\left((\Lambda_0\Lambda_0^\top + \Psi_0)^{-1}\Sigma_{t+\frac{1}{2}}\right)$
for $\tau = 0, \ldots, N-1$ **do**
  Compute intermediate matrix of size $K \times D$

$$\beta_\tau = \Lambda_\tau^\top \Psi_\tau^{-1}[I - \Lambda_\tau(I + \Lambda_\tau^\top \Psi_\tau^{-1}\Lambda_\tau)^{-1}\Lambda_\tau^\top \Psi_\tau^{-1}]$$

  Update low rank and diagonal parameters

$$\Lambda_{\tau+1} = \Sigma_{t+\frac{1}{2}}\beta_\tau^\top(\beta_\tau\Sigma_{t+\frac{1}{2}}\beta_\tau^\top + I - \beta_\tau\Lambda_\tau)^{-1}$$

$$\Psi_{\tau+1} = \mathrm{diag}((I - \Lambda_{\tau+1}\beta_\tau)\Sigma_{t+\frac{1}{2}})$$

  Update parameters with momentum η:

$$\Lambda_{\tau+1} = (1-\eta)\Lambda_\tau + \eta\Lambda_{\tau+1}$$

$$\Psi_{\tau+1} = (1-\eta)\Psi_\tau + \eta\Psi_{\tau+1}$$

  Compute

$$\mathrm{KL}_{\tau+1} = \log(|\Lambda_{\tau+1}\Lambda_{\tau+1}^\top + \Psi_{\tau+1}|) + \mathrm{tr}\left((\Lambda_{\tau+1}\Lambda_{\tau+1}^\top + \Psi_{\tau+1})^{-1}\Sigma_{t+\frac{1}{2}}\right)$$

  if $(\mathrm{KL}_{\tau+1}/\mathrm{KL}_\tau < 1+\varepsilon)$ **then**
    break # Early stopping
  end if
end for
Output: Σ_{t+1} (represented implicitly by $\Lambda_{\tau+1}, \Psi_{\tau+1}$)

Our goal is to derive an algorithm for score-based VI with this variational family whose computational cost and memory also scale linearly in D. To do so, we introduce a *patch step* that projects the covariance update in Eq. 2.10 into one that has the form of Eq. 3.1.

The new algorithm is shown in Algorithm 1. Intuitively, at each iteration, we use $\Sigma_{t+\frac{1}{2}}$ to denote the intermediate update of the *unconstrained* covariance matrix computed by Eq. 2.10, and then we apply a patch that projects $\Sigma_{t+\frac{1}{2}}$ to a low-rank plus diagonal matrix,

$$\Sigma_{t+1} = \texttt{Patch}\left[\Sigma_{t+\frac{1}{2}}\right] \tag{3.2}$$

such that the updated covariance matrix is still of the form in Eq. 3.1, i.e. $\Sigma_{t+1} = \Lambda_{t+1}\Lambda_{t+1}^\top + \Psi_{t+1}$. Note that we perform this patch without constructing the Σ_{t+1} matrix explicitly, but only by estimating the parameters Λ_{t+1} and Ψ_{t+1}. We compute this projection via an Expectation-Maximization (EM) algorithm that minimizes the KL divergence, $\mathrm{KL}(q_{t+\frac{1}{2}}||q_{t+1})$, with respect to the parameters $(\Lambda_{t+1}, \Psi_{t+1})$. Here, $q_{t+\frac{1}{2}}$ and q_{t+1} are the Gaussian distributions with shared mean μ_{t+1} and covariance matrices $\Sigma_{t+\frac{1}{2}}$ and Σ_{t+1}.

The next sections show why a patch is needed at each iteration to perform this projection, describe how the projection is performed via an EM algorithm, and discuss the overall scaling of this new algorithm.

3.1 Why add a "patch step"?

In this section, we investigate the update of Eq. 2.10 in detail and show why a patch step is need to scale linearly in the number of dimensions, D. Recall that when $B \ll D$, the matrix U in Eq. 2.6 of rank at most $B+1$ because it is constructed from a sum of $B+1$ rank-one matrices. Here we note the following: when Σ_t is the sum of a diagonal plus low-rank matrix, as in Eq. 3.1, then the matrix V in Eq. 2.7 can also be written as the sum of a diagonal plus low-rank matrix. In particular, we can write

$$V = \Psi_t + RR^\top, \tag{3.3}$$

where the matrix $R \in \mathbb{R}^{D \times (B+1+K)}$ is constructed by concatenating the column vectors

$$R = \left[\sqrt{\tfrac{\lambda_t}{B}}(z_1 - \overline{z}), ..., \sqrt{\tfrac{\lambda_t}{B}}(z_B - \overline{z}), \sqrt{\tfrac{\lambda_t}{1+\lambda_t}}(\mu_t - \overline{z}), \Lambda_t\right]. \tag{3.4}$$

The construction of the tall but thin matrix R in Eq. 3.4 is analogous to the construction of the matrix Q in Eq. 2.9.

With these definitions, we can now realize certain gains in efficiency when the number of dimensions, D, is large. First, note that the covariance update of Eq. 2.10 can

be written explicitly in terms of R and Q as

$$\Sigma_{t+\frac{1}{2}} = \Psi_t + RR^\top - (\Psi_t + RR^\top)^\top Q M Q^\top (\Psi_t + RR^\top), \tag{3.5}$$

where we have introduced M as shorthand for the bracketed $(B+1)\times(B+1)$ matrix $\left[\frac{1}{2}I + \left(Q^\top V Q + \frac{1}{4}I\right)^{\frac{1}{2}}\right]^{-2}$ that appears in Eq. 2.10. Hence the updated covariance matrix $\Sigma_{t+\frac{1}{2}}$ can be implicitly represented by the matrices Ψ_t, Q, R, and M, which are respectively of size D (along the diagonal), $D\times(B+1)$, $D\times(K+B+1)$, and $(B+1)\times(B+1)$; i.e., the memory cost is $\mathcal{O}(D)$. Second, we note that none of the matrix products in Eq. 3.5 are between dense high-rank matrices. Hence, we can also evaluate Eq. 3.5 with a computational cost that is $\mathcal{O}(D)$; this is shown more fully in Appendix A.3.

3.2 `Patch`: projection via an EM algorithm

To perform the patch in Eq. 3.2, we aim to minimize the KL divergence $\mathrm{KL}(q_{t+\frac{1}{2}}||q_{t+1})$, with respect to the parameters (Λ, Ψ), where

$$q_{t+\frac{1}{2}} = \mathcal{N}(\mu_{t+1}, \Sigma_{t+\frac{1}{2}}), \tag{3.6}$$
$$q_{t+1} = \mathcal{N}(\mu_{t+1}, \Sigma_{t+1}), \tag{3.7}$$

and $\Sigma_{t+1} = \Lambda\Lambda^\top + \Psi$. This KL divergence is given by

$$\mathrm{KL}(q_{t+\frac{1}{2}}||q_{t+1}) \tag{3.8}$$
$$= \frac{1}{2}\left[\log\left(\frac{|\Lambda\Lambda^\top+\Psi|}{|\Sigma_{t+\frac{1}{2}}|}\right) - D + \mathrm{tr}\left\{(\Lambda\Lambda^\top+\Psi)^{-1}\Sigma_{t+\frac{1}{2}}\right\}\right].$$

Note that the KL divergence is independent of the mean μ_{t+1}, which is the same for both distributions. Hence, for simplicity, and without loss of generality, we assume that the mean is zero in the following discussion.

To minimize this objective, we adapt a *factor analysis model*, as it allows us to minimize this KL divergence by defining and minimizing a simpler auxiliary function. Indeed, the update we consider is the infinite data limit of the EM algorithm for factor analysis (Rubin and Thayer, 1982; Ghahramani and Hinton, 1996; Saul and Rahim, 2000).

Let $\tilde{q}_{t+\frac{1}{2}}(z) = \mathcal{N}(z\,|\,0, \Sigma_{t+\frac{1}{2}})$. Consider the following factor analysis model:

$$\zeta \sim \mathcal{N}_K(0, I), \qquad z\,|\,\zeta \sim \mathcal{N}_D(\Lambda\zeta, \Psi), \tag{3.9}$$

where $\zeta \in \mathbb{R}^K$ is a hidden variable and $K \ll D$, and the subscript on the Gaussian $\mathcal{N}(\cdot)$ denotes the dimension of the multivariate variable. Then the marginal distribution $q_\theta(z)$, where $\theta = (\Lambda, \Psi)$, is Gaussian with the structured covariance matrix $\Sigma = \Lambda\Lambda^\top + \Psi$:

$$q_\theta(z) = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi). \tag{3.10}$$

We estimate a factored representation for $\Sigma_{t+\frac{1}{2}}$ by minimizing the KL divergence between $\tilde{q}_{t+\frac{1}{2}}$ and q_θ with respect to θ (which is equivalent to minimizing Eq. 3.8 with respect to Λ and Ψ).

As is standard in the EM algorithm, we minimize an auxiliary function that upper bounds the KL divergence. This auxiliary function is minimized by simple updates that also monotonically decrease the KL divergence. The auxiliary function is given by

$$\mathcal{A}(\theta_\tau, \theta) := \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[-\int \log(q(\zeta, z|\theta))\, q(\zeta|z, \theta_\tau)\, d\zeta\right]. \tag{3.11}$$

We show in Appendix A.2 how to minimize this function with respect to θ, i.e.,

$$\theta_{\tau+1} = \arg\min_{\theta\in\Theta} \mathcal{A}(\theta_\tau, \theta), \tag{3.12}$$

and we also show that this update decreases $\mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}||q_\theta)$ at every iteration.

The EM algorithm takes the following form. In the E-step, we compute the following statistics (expectations) of the conditional distribution of the hidden variable ζ given z:

$$\mathbb{E}_q[\zeta] = \beta_\tau z, \qquad \mathbb{E}_q[\zeta\zeta^\top] = \beta_\tau z z^\top \beta_\tau^\top + I - \beta_\tau\Lambda_\tau,$$

where $\beta_\tau := \Lambda_\tau^\top(\Lambda_\tau\Lambda_\tau^\top + \Psi_\tau)^{-1}$ is a $K\times D$ matrix that can be computed efficiently via the Woodbury matrix identity. In the M-step, we optimize the objective in Eq. 3.11, which becomes

$$\mathcal{L} := \tfrac{1}{2}\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[z^\top\Psi^{-1}z - 2z^\top\Psi^{-1}\Lambda\mathbb{E}_q[\zeta] + \mathrm{tr}(\Lambda^\top\Psi^{-1}\Lambda\mathbb{E}_q[\zeta\zeta^\top])\right] + \tfrac{\log|\Psi|}{2} + \text{const.} \tag{3.13}$$

The EM updates are derived by setting $\nabla_\Lambda\mathcal{L} = 0$ and $\nabla_\Psi\mathcal{L} = 0$. In this way, we derive the updates

$$\Lambda_{\tau+1} = \Sigma_{t+\frac{1}{2}}\beta_\tau^\top(\beta_\tau\Sigma_{t+\frac{1}{2}}\beta_\tau^\top + I - \beta_\tau\Lambda_\tau)^{-1}, \tag{3.14}$$
$$\Psi_{\tau+1} = \mathrm{diag}((I - \Lambda_{\tau+1}\beta_\tau)\Sigma_{t+\frac{1}{2}}). \tag{3.15}$$

These steps are summarized in Algorithm 2, and a complete derivation is provided in Appendix A.1.

3.3 Convergence of the EM updates

To minimize the number of EM steps, Algorithm 2 adds a momentum hyperparameter η to the EM updates (lines 7–8). We also implement early stopping wherein we monitor the KL divergence (Eq. 3.8) for every update, and if the relative difference in KL over successive iterations is less than a threshold value, we terminate the patch step. We fix this threshold value to 10^{-4} for all experiments in this paper. At every

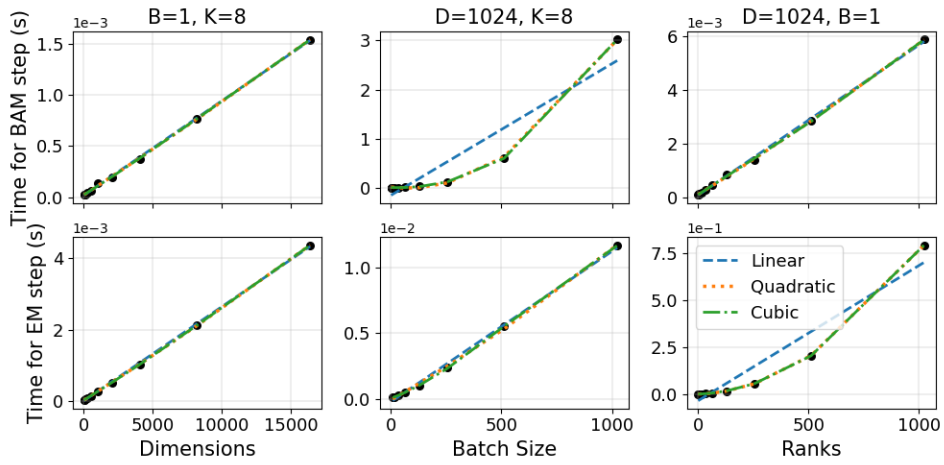

Figure 3.1: Scaling of pBaM algorithm with dimensions, batch size and rank. The top row shows the timing for BaM step (without evaluating the scores of the target) and the bottom row shows timing for a single EM step. All timings are in seconds. We also show linear, quadratic and cubic fits to the data-points.

iteration of pBaM, we initialize Ψ and Λ for the patch step with their current values. As a result, the convergence is quite fast. Roughly speaking, we find that convergence requires hundreds of EM updates in the first couple iterations of pBaM (when the variational approximation is rapidly evolving), but fewer than 10 EM updates after only a few iterations of BaM. The average number of EM updates over 10,000 iterations was fewer than 5 in most of our experiments.

3.4 Scaling of pBaM

Before proceeding to numerical experiments, we study the scaling of our algorithm in terms of the dimensionality D, batch size B, and rank K. Every iteration of our approach consists of two steps—a BaM update and the patch step, which is implemented as a series of EM updates. In Appendix A.3, we show that the BaM update scales as $\mathcal{O}(DB^2 + B^3 + KBD)$, while the EM update scales as $\mathcal{O}(DK^2 + K^3 + KBD)$. As desired, the scaling of both steps is linear in D. We validate these scalings empirically in Figure 3.1, where we show the time taken for the BaM update and a single EM step for increasing dimensionality, batch size, and rank, varying one of these factors while keeping the others fixed. For the BaM step, we use a dummy model in which it takes a negligible time to evaluate scores.

Note that while the EM updates scale similarly as each BaM step, the former are independent of the target distribution. On the other hand, for each BaM step, it is necessary to re-evaluate the scores (i.e., gradients) of the target distribution. For any physical model of reasonable complexity, evaluating scores is likely to be far more expensive than a single iteration of EM.

4 Related work

Structured covariances have been considered in several variational inference algorithms. In particular, Miller et al. (2017) and Ong et al. (2018) propose using the Gaussian family with low rank plus diagonal structured covariances as a variational family in a BBVI algorithm based on reparameterization of the ELBO objective. Tomczak et al. (2020) study the use of low-rank plus diagonal Gaussians for variational inference in Bayesian neural networks.

Bhatia et al. (2022) study the problem of fitting a low-rank plus diagonal Gaussian to a target distribution; they show that when the target itself is Gaussian, gradient-based optimization of the (reverse) KL divergence results in a power iteration. However, their approach restricts the diagonal to be constant.

Alternative families for structured VI have also been proposed by many researchers (Saul and Jordan, 1995; Hoffman and Blei, 2015; Tan and Nott, 2018; Ko et al., 2024). For example, Tan and Nott (2018) use sparse precision matrices to model conditional independence, and Ko et al. (2024) use covariance matrices with block diagonal structure.

5 Experiments

The patched BaM algorithm (pBaM) is implemented in JAX (Bradbury et al., 2018), and publicly available on Github. We compare pBaM against factorized and low rank plus diagonal ADVI (ADVI-D and ADVI-LR respectively) (Miller et al., 2017; Ong et al., 2018), both of which can scale to high dimensions. For smaller dimensions, we will also compare against ADVI with

full covariance (ADVI-F) and Batch-and-Match (BaM) with full covariance. For higher dimensional experiments, we omit the full covariance methods due to their large computational cost. Unless otherwise mentioned, all experiments are done with batch size $B = 32$.

pBaM Setup: We set the parameters of the patch step to default values of $\eta = 1.2$ and tolerance $t = 10^{-4}$. We experimented with different values and found the algorithm to be robust in the broad parameter ranges of $1 < \eta < 1.5$ and $t < 10^{-3}$. We initialize the learning rate parameter as $\lambda_0 = 1$ and decay its value at each iteration according to the schedule $\lambda_t = \lambda_0/(1+t)$.

ADVI Implementation: To implement ADVI we used automatic differentiation in JAX and the ADAM optimizer (Kingma and Ba, 2015) to minimize the reverse KL divergence. In all experiments, we evaluate performance over a grid of learning rates and only show the best results. We scheduled the learning rate to decrease linearly over the course of iterations to a minimum value of 10^{-5}. We also experimented with other forms of scheduling, e.g., cosine scheduling, but did not find significant differences in performance.

We implemented factorized and multivariate normal distributions in NumPyro. We implemented a more efficient alternative for Gaussian distributions with low-rank plus diagonal covariance matrices. This implementation generates samples via the factor analysis model in Eq. 3.9 and evaluates the log-probability using the Woodbury matrix identity; the full covariance matrix is never stored in memory.

5.1 Synthetic Gaussian targets

We begin by considering Gaussian target distributions with a low-rank plus diagonal covariance:

$$p(z) = \mathcal{N}(z \,|\, \mu, \Psi + \Lambda\Lambda^\top),$$
$$\mu \sim \mathcal{N}(0,1), \quad \Psi \sim \mathcal{U}(0,1), \quad \Lambda \sim \mathcal{N}(0,1), \tag{5.1}$$

where $\mu, \Psi \in \mathbb{R}^D$ and $\Lambda \in \mathbb{R}^{D\times K}$. Note that this distribution has strong off-diagonal components and the condition number (κ) of this covariance matrix increases with increasing dimensions, e.g., $\kappa > 10^6$ for $D > 2048$ and $K = 32$. We study the performance of pBaM on this target distribution as a function of the number of dimensions (D) and rank (K).

5.1.1 Performance with increasing dimensions

We begin by considering the setting where the variational distribution and the target distribution are in the same family. In this setting, we show that despite the patch step, the algorithm converges to match the target, even in high dimensions.

Figure 5.1 shows the results on Gaussian targets with increasing numbers of dimensions but fixed rank $K = 32$. We show results for pBaM and ADVI-LR that fit this target distribution with variational approximations of the same rank, $K = 32$. We monitor the reverse KL divergence as a function of gradient evaluations. Both algorithms are able to fit the target distribution, but pBaM converges much faster. We also show results for the best two learning rates for ADVI and two different values of λ_t for pBaM. While ADVI-LR converges faster initially with a large learning rate, it gets stuck closer to the solution even when the learning rate decays with a linear schedule. On the other hand, BaM performs well across different learning rates, though though $\lambda_t = 1$ is more stable than $\lambda_t = 10$ for $D = 8192$.

5.1.2 Performance with increasing ranks

Next we study the performance of pBaM with increasing ranks for the variational approximation. The results are shown in Figure 5.2. Here we have fixed the target to be Gaussian (Eq. 5.1) with $D = 512$ and $K = 256$, and we fit it with variational approximations of ranks $K = 16, 64, 128$ and 256. In the last setting, the variational approximation and the target again belong to the same family, and we expect an exact fit.

In the left panel of Figure 5.2, we monitor the reverse KL divergence while in the other panels we show marginal histograms for the first dimension. Again, we find that while both ADVI-LR and pBaM converge to the same quality of solution, pBaM converges much faster. When the rank of the variational distribution is less than the target, the final approximation is more concentrated (smaller marginal variance). With increasing rank, the final approximation becomes a better fit, both in terms of minimizing the reverse KL and capturing the variance of the target distribution.

5.2 Gaussian process inference

Next we study pBAM for variational inference in Gaussian processes (GPs). Consider a generative model of the form

$$f \sim \mathrm{GP}(m, \tilde{\kappa}), \quad y_{1:N} | f \overset{iid}{\sim} P_f, \tag{5.2}$$

where m is a mean function, $\tilde{\kappa}$ is a kernel function, and P_f is a distribution that characterizes the likelihood of a data point. This structure describes many popular models with intractable posteriors, such as Gaussian process regression (with e.g., Poisson or Negative Binomial likelihoods), Gaussian process classification, and many spatial statistics models (e.g., the log-Gaussian Cox process).

The function f is typically represented as a vector of function evaluations on a set of D points, and thus

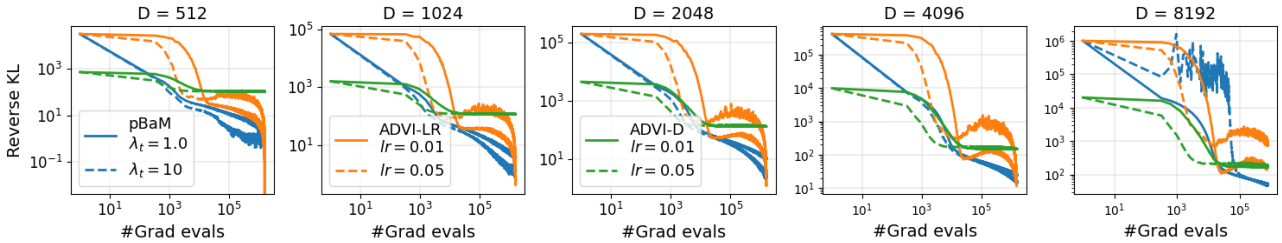

Figure 5.1: Performance with increasing dimensions for Gaussian with low-rank ($K = 32$) plus diagonal covariance as target. We monitor reverse KL divergence for pBaM (blue) and ADVI-LR (orange) with $K = 32$, and ADVI-D (green) for two different λ_t and learning rate respectively (solid vs dashed).

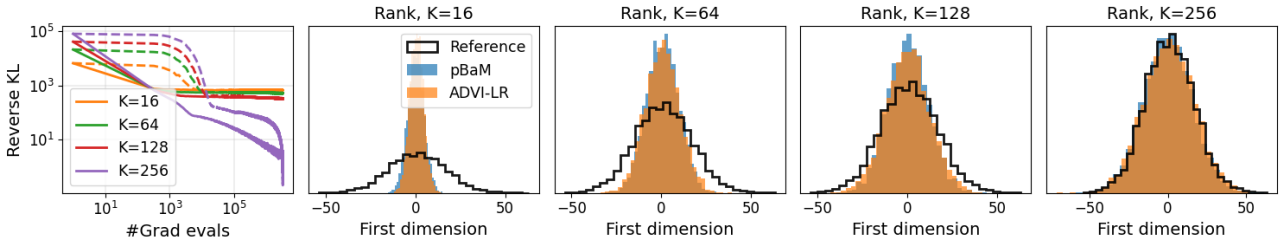

Figure 5.2: Performance with increasing ranks (K) for variational family. Target is 512 dimension Gaussian with low-rank ($K = 256$) plus diagonal. We monitor reverse KL divergence for pBaM (solid) and ADVI-LR (dashed), and show marginal histograms for first dimension.

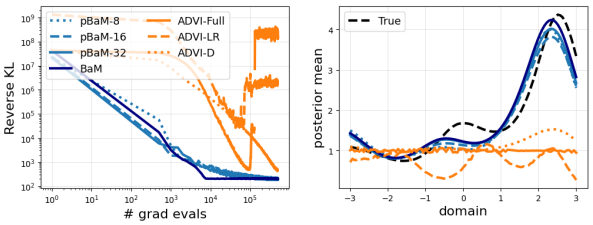

Figure 5.3: Synthetic GP-Poisson regression, $D = 100$. We show the reverse KL (left) and the estimated rate using the posterior means (right).

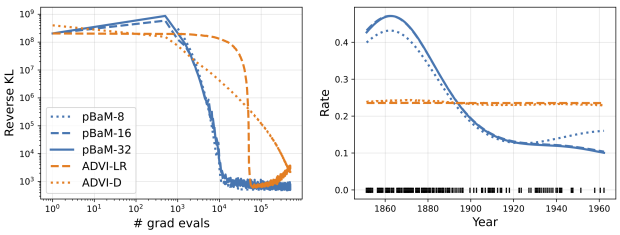

Figure 5.4: Log Gaussian Cox Process, $D = 811$. We plot the reverse KL (left) and the estimated rate (right).

can be itself high-dimensional (even if the data are low dimensional). In all experiments, each method was run for a fixed number of iterations, and so not all methods converged using the same computational budget.

Poisson regression. We consider a Poisson regression problem in $D = 100$ dimensions, where

$$p(y_1, \ldots, y_N; f) = \prod_{n=1}^{N} \text{Pois}(y_n; \lambda_n), \quad \lambda_n = \exp(f_n).$$

Data were generated from this model using a GP with zero-valued mean function and an RBF kernel with unit length-scale.

In Figure 5.3, we compare pBaM with ranks $K = 8, 16, 32$ against the BaM algorithm and ADVI (with full, rank 32, and diagonal covariances). In the left plot, we report the reverse KL vs the number of gradient evaluations. Here we observe that the low-rank BaM algorithms converge almost as quickly as the full-rank BaM algorithm. We found that even with extensive tuning of the learning rate, the ADVI algorithms tended to perform poorly for this model; the full covariance ADVI algorithm diverges. In the right plot, we show the true rate function along with the inferred rate functions (we report the posterior mean from the VI algorithms). For the BaM/pBaM methods, we found that increasing the rank results in a better fit.

Log-Gaussian Cox process. Now we consider the log-Gaussian Cox process (LGCP) (Møller et al., 1998), which is used to model point process data. One approach to approximate the log-Gaussian Cox process is to bin the points and then to consider a Poisson likelihood for the counts in each bin. Specifically, we consider $p(y_1, \ldots, y_N; f) = \prod_{n=1}^{N} \text{Pois}(y_n; \exp(f(x_n) + m)$, where m represents a mean offset.

We apply this model to a coal mining disasters data set (Carlin et al., 1992), which contains 191 events of mining explosions in Britain between March 15, 1851 and March 22, 1962. Following Murray et al. (2010), we bin the points into 811 bins, and so the latent space is $D = 811$ dimensional; in addition, we use an RBF kernel with a length-scale of 37 and set the mean to $m = \log(191/811)$.

The left panel of Figure 5.4 shows the results of ADVI-LR, ADVI-D, and pBaM on ranks $8, 16, 32$. Here we report the reverse KL divergence, and we find that pBaM converges much faster than the ADVI methods. Notably, the varying ranks for pBaM perform similarly. The right panel of Figure 5.4 shows the estimated rate functions using the posterior mean of the variational inference algorithms; the event data are visualized in the black rug plot points. The estimated intensity from pBaM is also much more accurate than ADVI solutions that use MCMC sampling for this problem (e.g., Nguyen and Bonilla (2014, Figure 4)).

5.3 IRT model

Given I students and J questions, item response theory (IRT) models how students answer questions on a test depending on student ability (α_i), question difficulty (β_j), and the discriminative power of the questions (θ_j) (Gelman and Hill, 2006). The observations to be modeled are the binary values $\{y_{ij}\}$, where y_{ij} indicates the correctness of the i^{th} student's answer to question j. Both the mean-centered ability of the student α_i, and the difficulty of questions β_j have hierarchical priors. These terms are combined into a variant of logistic regression with a multiplicative discrimination parameter.

We show the results of variational inference for this model in Figure 5.5. Here $I = 20$ and $J = 100$, leading to dimensionality $D = 143$. ADVI-LR and ADVI-F diverged on this model even with tuned learning rates. Full-rank BAM converges to a better reverse KL divergence than the low-rank pBaM or the factorized approximation of ADVI-D. However, the two-dimensional marginals for pBaM look closer to those of BAM than to those of the factorized approximation.

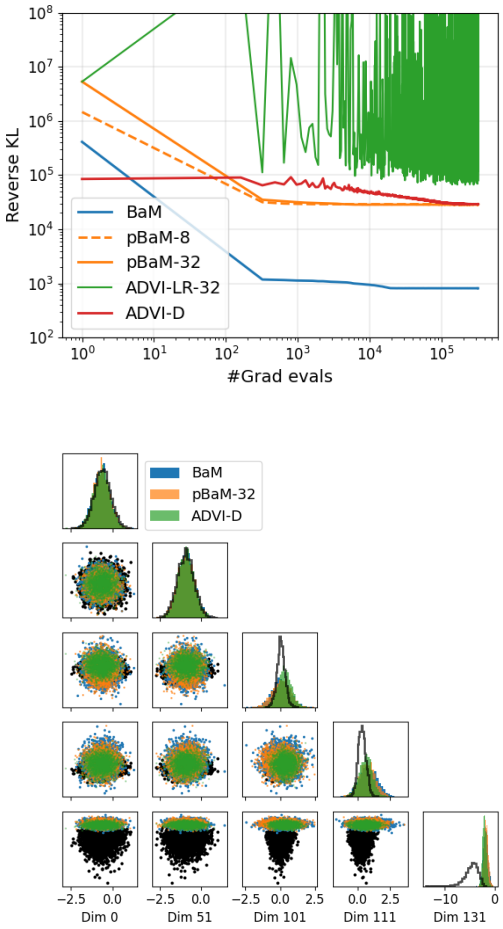

Figure 5.5: We show the reverse KL as well as 2-D marginals for five randomly selected dimensions for fitting IRT model with BaM, pBaM for ranks K=8 and 32, and ADVI-D. ADVI-LR and ADVI-F did not converge for this model.

6 Discussion and future work

We have developed a score-based BBVI algorithm for high-dimensional problems where the covariance of the Gaussian variational family is represented as a sum of a low-rank and diagonal (LR+D) component. Our method extends the batch-and-match algorithm by adding a patch step that projects an unconstrained covariance matrix to one that is structured in this way. The computational and storage costs of this algorithm scale linearly in the dimension D. Empirically, we find that our algorithm converges faster than ADVI with LR+D and is more stable on real-world examples. We provide a Python implementation of pBaM at https://github.com/modichirag/GSM-VI/.

A number of future directions remain. One is to study pBaM for other high dimensional problems such as arise (for instance) in Bayesian neural networks. Another is to successively increase the rank of variational approximation in pBaM after an initial low-rank fit. Finally, it would also be of interest to consider other types of structured covariance matrices (e.g., exploiting sparsity) for scaling score-based VI to high dimensional distributions.

Acknowledgments

We thank Robert Gower for helpful discussions.

References

K. Bhatia, N. L. Kuang, Y.-A. Ma, and Y. Wang. Statistical and computational trade-offs in variational inference: A case study in inferential model selection. *arXiv eprint 2207.11208*, 2022.

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

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

D. Cai, C. Modi, C. Margossian, R. Gower, D. Blei, and L. Saul. EigenVI: score-based variational inference with orthogonal function expansions. In *Advances in Neural Information Processing Systems*, 2024a.

D. Cai, C. Modi, L. Pillaud-Vivien, C. Margossian, R. Gower, D. Blei, and L. Saul. Batch and match: black-box variational inference with a score-based divergence. In *International Conference on Machine Learning*, 2024b.

B. P. Carlin, A. E. Gelfand, and A. F. Smith. Hierarchical Bayesian analysis of changepoint problems. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(2):389–405, 1992.

A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2006.

Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

M. D. Hoffman and D. M. Blei. Structured stochastic variational inference. In *Artificial Intelligence and Statistics*, pages 361–369, 2015.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

J. Ko, K. Kim, W. C. Kim, and J. R. Gardner. Provably scalable black-box variational inference with structured variational families. In *International Conference on Machine Learning*, 2024.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2017.

A. C. Miller, N. J. Foti, and R. P. Adams. Variational boosting: Iteratively refining posterior approximations. In *International Conference on Machine Learning*, pages 2420–2429. PMLR, 2017.

C. Modi, C. Margossian, Y. Yao, R. Gower, D. Blei, and L. Saul. Variational inference with Gaussian score matching. *Advances in Neural Information Processing Systems*, 36, 2023.

J. Møller, A. R. Syversveen, and R. P. Waagepetersen. Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.

I. Murray, R. Adams, and D. MacKay. Elliptical slice sampling. In *Artificial Intelligence and Statistics*, volume 9, pages 541–548, 2010.

T. V. Nguyen and E. V. Bonilla. Automated variational inference for Gaussian process models. *Advances in Neural Information Processing Systems*, 27, 2014.

V. M.-H. Ong, D. J. Nott, and M. S. Smith. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3):465–478, 2018.

K. B. Petersen and M. S. Pedersen. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

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

D. B. Rubin and D. T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47:69–76, 1982.

L. Saul and M. Jordan. Exploiting tractable substructures in intractable networks. *Advances in Neural Information Processing Systems*, 8, 1995.

L. K. Saul and M. G. Rahim. Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(2):115–125, 2000.

L. S. Tan and D. J. Nott. Gaussian variational approximation with sparse precision matrices. *Statistics and Computing*, 28:259–275, 2018.

M. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*. PMLR, 2014.

M. Tomczak, S. Swaroop, and R. Turner. Efficient low rank Gaussian variational inference for neural networks. *Advances in Neural Information Processing Systems*, 33:4610–4622, 2020.

M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1 (1–2):1–305, 2008.

Y. Yang, R. Martin, and H. Bondell. Variational approximations using Fisher divergence. *arXiv preprint arXiv:1905.05284*, 2019.

C. Zhang, B. Shahbaba, and H. Zhao. Variational Hamiltonian Monte Carlo via score matching. *Bayesian Analysis*, 13(2):485, 2018.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No] Our code is available on github, and we will include a link to it in the published version.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A The patch step: additional details

A.1 Derivation of the patch step update

In this section, we derive the infinite-data-limit EM algorithm for factor analysis in the context of pBaM. In particular, we consider the model in Eq. 3.9 and the objective in Eq. 3.12.

Recall that $\tilde{q}_{t+\frac{1}{2}}(z) = \mathcal{N}(z \,|\, 0, \Sigma_{t+\frac{1}{2}})$, where we assume we are given $\Sigma_{t+\frac{1}{2}}$. As we discuss in Section 3.1, we do not need to form this matrix explicitly.)

First note that the joint distribution under the model in Eq. 3.9 is

$$\begin{bmatrix} \zeta \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} I & \Lambda^\top \\ \Lambda & \Lambda\Lambda^\top + \Psi \end{bmatrix} \right). \tag{A.1}$$

Defining $\beta := \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}$, the conditional distribution of $\zeta | z, \Psi, \Lambda$ is Gaussian with mean and covariance

$$\mathbb{E}_q[\zeta \,|\, z] = \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1} z = \beta z \tag{A.2}$$

$$\mathrm{Cov}_q[\zeta \,|\, z] = I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda = I - \beta\Lambda. \tag{A.3}$$

In what follows, we denote the conditional expectation with respect to $q(\zeta \,|\, z, \Lambda_\tau, \Psi_\tau)$ as $\mathbb{E}_q[\cdot]$, where τ represents the current EM iteration number. In the E-step, we compute the following statistics of the conditional distribution (Eq. A.2 and Eq. A.3) of the hidden variable ζ given z and the current parameters Λ_τ, Ψ_τ:

$$\mathbb{E}_q[\zeta] = \beta_\tau z \tag{A.4}$$

$$\mathbb{E}_q[\zeta\zeta^\top] = \mathbb{E}_q[\zeta]\mathbb{E}_q[\zeta]^\top + \mathrm{Cov}_q[\zeta] = \beta_\tau z z^\top \beta_\tau^\top + I - \beta_\tau\Lambda_\tau, \tag{A.5}$$

where $\beta_\tau := \Lambda_\tau^\top(\Lambda_\tau\Lambda_\tau^\top + \Psi_\tau)^{-1}$.

In the M-step, we maximize the objective in Eq. 3.12. First, we rewrite the objective in terms of the optimization variables (Λ, Ψ) and constants. The log joint distribution is

$$\log q(\zeta, z) = -\tfrac{1}{2}(z - \Lambda\zeta)^\top\Psi^{-1}(z - \Lambda\zeta) - \tfrac{\log|\Psi|}{2} + \text{const.},$$

and so the objective is

$$\mathcal{L} := -\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[\mathbb{E}_q[\log(q(\zeta, z \,|\, \Lambda, \Psi))]] + \text{const.}, \tag{A.6}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[\mathbb{E}_q\left[z^\top\Psi^{-1}z - 2z^\top\Psi^{-1}\Lambda\zeta + \zeta^\top\Lambda^\top\Psi^{-1}\Lambda\zeta\right]\right] + \tfrac{\log|\Psi|}{2} + \text{const.}, \tag{A.7}$$

$$= \tfrac{1}{2}\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[z^\top\Psi^{-1}z - 2z^\top\Psi^{-1}\Lambda\mathbb{E}_q[\zeta] + \mathrm{tr}(\Lambda^\top\Psi^{-1}\Lambda\mathbb{E}_q[\zeta\zeta^\top])\right] + \tfrac{\log|\Psi|}{2} + \text{const.}, \tag{A.8}$$

Assuming we can exchange limits (and thus derivatives) and integration, below, we compute the derivatives with respect to the optimization parameters. We use the following identities (Petersen and Pedersen, 2008):

$$\nabla_{\mathcal{C}} a^\top\mathcal{C}b = ab^\top \tag{A.9}$$

$$\nabla_{\mathcal{C}}\mathrm{tr}(\mathcal{C}A\mathcal{C}^\top) = 2\mathcal{C}A \tag{A.10}$$

$$\nabla_{\mathcal{C}}\log(|\mathcal{C}|) = \mathcal{C}^{-1} \tag{A.11}$$

$$\nabla_{\mathcal{C}}\mathrm{tr}\{\mathcal{C}B\} = B^\top, \tag{A.12}$$

where we assume above that $A = A^\top$.

Derivative w.r.t. Λ. Setting the derivative of Eq. A.8 with respect to Λ to 0, i.e.,

$$\nabla_\Lambda\mathcal{L} = \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[\Psi^{-1}z\mathbb{E}_q[\zeta]^\top - \Psi^{-1}\Lambda\mathbb{E}[\zeta\zeta^\top]\right] = 0, \tag{A.13}$$

implies that

$$\Lambda^* = \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[z\mathbb{E}_q[\zeta]^\top]\left(\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\mathbb{E}_q[\zeta\zeta^\top]\right)^{-1} \tag{A.14}$$

$$= \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[zz^\top]\beta_\tau^\top(\beta_\tau\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[zz^\top]\beta_\tau^\top + I - \beta_\tau\Lambda_\tau)^{-1} \tag{A.15}$$

$$= \Sigma_{t+\frac{1}{2}}\beta_\tau^\top(\beta_\tau\Sigma_{t+\frac{1}{2}}\beta_\tau^\top + I - \beta_\tau\Lambda_\tau)^{-1}, \tag{A.16}$$

where in the second line we plugged in the statistics from Eq. A.4 (note that the statistics are evaluated with respect to the current parameters Λ_τ, Ψ_τ), and in the third line we used $\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[zz^\top] = \Sigma_{t+\frac{1}{2}}$.

Derivative w.r.t. Ψ. Taking the derivative w.r.t. Ψ^{-1} and setting $\nabla_{\Psi^{-1}}\mathcal{L} = 0$, i.e.,

$$\tfrac{1}{2}\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[zz^\top - 2\Lambda\mathbb{E}_q[\zeta]z^\top + \Lambda\mathbb{E}_q[\zeta\zeta^\top]\Lambda^\top\right] - \tfrac{1}{2}\Psi = 0 \tag{A.17}$$

results in

$$\Psi^* = \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[zz^\top - 2\Lambda\mathbb{E}_q[\zeta]z^\top + \Lambda\mathbb{E}_q[\zeta\zeta^\top]\Lambda^\top\right] \tag{A.18}$$

$$= \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[zz^\top] - \Lambda\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}[\mathbb{E}_q[\zeta]z^\top] \tag{A.19}$$

$$= \Sigma_{t+\frac{1}{2}} - \Lambda\beta_\tau\Sigma_{t+\frac{1}{2}} \tag{A.20}$$

$$= (I - \Lambda\beta_\tau)\Sigma_{t+\frac{1}{2}}, \tag{A.21}$$

where the second line follows from substituting the Λ^* update into the third term of Eq. A.18.

The final update results from taking the diagonal part of Ψ^* above. Note that this update depends on the newly updated Λ^* term in Eq. A.18 (as opposed to the current Λ_τ).

A.2 The patch step updates monotonically improve KL divergence

Suppose we have an auxiliary function that has the following properties:

Property 1: $\mathcal{A}(\theta, \theta) = \mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_\theta)$
Property 2: $\mathcal{A}(\theta, \theta') \geq \mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_{\theta'})$ for all $\theta' \in \Theta$.

Then the update

$$\theta_{\tau+1} = \arg\min_{\theta\in\Theta} \mathcal{A}(\theta_\tau, \theta) \tag{A.22}$$

monotonically decreases $\mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}; q)$ at each iteration. The property that the KL divergence monotonically decreases holds because

$$\mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_{\theta_\tau}) = \mathcal{A}(\theta_\tau, \theta_\tau) \tag{A.23}$$

$$\geq \mathcal{A}(\theta_\tau, \theta_{\tau+1}) \tag{A.24}$$

$$\geq \mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_{\theta_{\tau+1}}), \tag{A.25}$$

where Eq. A.23 is due to Property 1, Eq. A.24 is by construction from the update, and Eq. A.25 is due to Property 2.

We now show that an auxiliary function satisfying Property 1 and 2 above exists. In particular, consider

$$\mathcal{A}(\theta, \theta') = \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[\log\tilde{q}_{t+\frac{1}{2}}(z) - \int\log\left(\tfrac{q(\zeta, z\,|\,\theta')}{q(\zeta\,|\,z,\theta)}\right)q(\zeta\,|\,z,\theta)d\zeta\right], \tag{A.26}$$

where $q(\zeta\,|\,z,\theta) \propto q(\zeta)q(z\,|\,\zeta,\theta)$ is the conditional distribution of the hidden variable induced by Eq. 3.9.

If $\theta' = \theta$, then Property 1 is satisfied: since $\log\frac{q(\zeta,z\,|\,\theta)}{q(\zeta\,|\,z,\theta)} = \log q_\theta(z)$ and $\int q(\zeta\,|\,z)dz = 1$, this results in $\mathcal{A}(\theta, \theta) = \mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_\theta)$.

Property 2 can be verified using an application of Jensen's inequality to the second term of Eq. A.26:

$$\begin{aligned}\mathcal{A}(\theta,\theta') &\geq \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[\log \tilde{q}_{t+\frac{1}{2}}(z) - \log\int\left(\tfrac{q(\zeta,z\,|\,\theta')}{q(\zeta\,|\,z,\theta)}\right)q(\zeta|z,\theta)d\zeta\right]\\ &= \mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[\log \tilde{q}_{t+\frac{1}{2}}(z) - \log\int q(\zeta,z\,|\,\theta')d\zeta\right]\\ &= \mathrm{KL}(\tilde{q}_{t+\frac{1}{2}}, q_{\theta'}).\end{aligned}$$

Thus, minimizing the function in Eq. A.26 monotonically decreases the KL divergence.

Summary of objective and updates. Combining Eq. A.22 and Eq. A.26 results in the iteration in Eq. 3.12: i.e.,

$$\theta_{\tau+1} = \arg\min_{\theta\in\Theta}\mathbb{E}_{\tilde{q}_{t+\frac{1}{2}}}\left[-\int\log(q(\zeta,z\,|\,\theta))q(\zeta\,|\,z,\theta_\tau)d\zeta\right] + \text{const.}, \tag{A.27}$$

where the constant represents factors that are independent of θ.

A.3 Computational cost of BaM update and the patch step

We first show that the computational cost of the BaM update with a structured covariance is linear in the dimension D. Next we discuss the computational cost of the patch step, which in combination with the BaM update, results in a computational cost linear in D.

Recall from Eq. 3.5 that the covariance can be written as

$$\Sigma_{t+\frac{1}{2}} = \Psi_t + RR^\top - (\Psi_t + RR^\top)^\top QMQ^\top(\Psi_t + RR^\top),$$

where $Q \in \mathbb{R}^{D\times(B+1)}$ and $R \in \mathbb{R}^{D\times(K+B+1)}$, and Ψ_t is a D-dimensional diagonal matrix.

First consider $H := (\Psi_t + RR^\top)^\top Q$, where $H \in \mathbb{R}^{D\times(B+1)}$; this matrix can be constructed with $\mathcal{O}((K+B)BD)$ computation.

Then we store $\Sigma_{t+\frac{1}{2}}$ implicitly by storing the matrices Ψ_t, R, H and M:

$$\Sigma_{t+\frac{1}{2}} = \Psi_t + RR^\top - HMH^\top, \tag{A.28}$$

Note that evaluating $RR^\top$ can be a $\mathcal{O}(D^2)$ operation, but since we store R, we do not need to explicitly evaluate it now, and we will be able to contract it with other elements in the patch update to maintain $\mathcal{O}(D)$ computation.

The matrix $M \in \mathbb{R}^{(B+1)\times(B+1)}$ defined in the update is:

$$M := \left[\tfrac{1}{2}I_{B+1} + \left(Q^\top(\Psi_t + RR^\top)Q + \tfrac{1}{4}I_{B+1}\right)^{\frac{1}{2}}\right]^{-2} = \left[\tfrac{1}{2}I_{B+1} + \left(H^\top Q + \tfrac{1}{4}I_{B+1}\right)^{\frac{1}{2}}\right]^{-2}.$$

Here evaluating the matrix in the square-root takes $\mathcal{O}(DB^2)$ computation, and it takes $\mathcal{O}(B^3)$ to compute the square root and inverse. Thus, overall, the total computational cost of the BaM update is $\mathcal{O}(DB^2 + B^3 + KBD)$; that is, it is linear in D.

Next, we turn to the computational cost of the patch step. Every iteration of patch step requires us to compute:

$$\begin{aligned}\beta_\tau &= \Lambda_\tau^\top\Psi_\tau^{-1}[I_K - \Lambda_\tau(I + \Lambda_\tau^\top\Psi_\tau^{-1}\Lambda_\tau)^{-1}\Lambda_\tau^\top\Psi_\tau^{-1}] \quad \text{(via Woodbury inverse identity)}\\ \Lambda_{\tau+1} &= \Sigma_{t+\frac{1}{2}}\beta_\tau^\top(\beta_\tau\Sigma_{t+\frac{1}{2}}\beta_\tau^\top + I_K - \beta_\tau\Lambda_\tau)^{-1},\\ \Psi_{\tau+1} &= \mathrm{diag}((I_D - \Lambda_{\tau+1}\beta_\tau)\Sigma_{t+\frac{1}{2}})\end{aligned}$$

Here $\beta_\tau \in \mathbb{R}^{K\times D}$, and evaluating β_τ requires forming matrix products that take $\mathcal{O}(K^2D)$ computation and inverting a $K\times K$ matrix, which in total takes $\mathcal{O}(K^2D + K^3)$ computation.

Using Eq. A.28, $\Sigma_{t+\frac{1}{2}}\beta_\tau^\top$ can be evaluated in $\mathcal{O}(DK(K+B))$, and hence the Λ_τ update can be evaluated in $\mathcal{O}(DK(K+B) + K^3)$ computation, with a similar complexity for the Ψ-update.

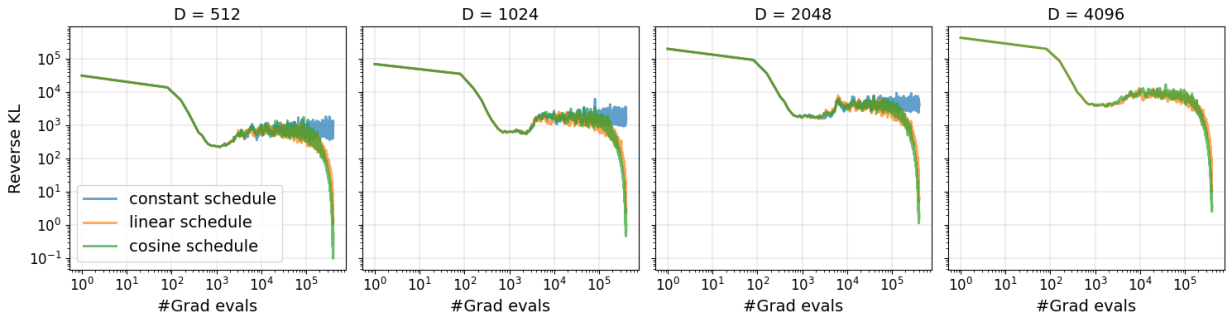

Figure B.1: Impact of scheduling the learning rate for ADVI-LR.

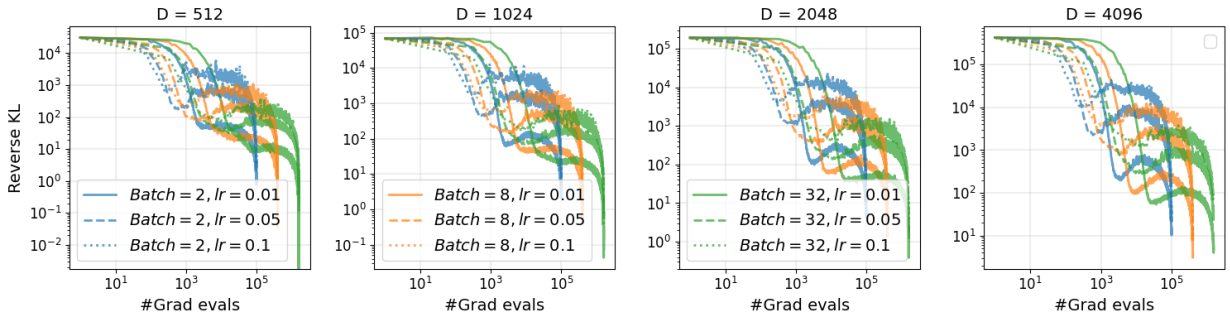

Figure B.2: Impact of changing the batch size and the learning rate for ADVI-LR.

B Additional experiments

In Table B.1, we provide an overview of the problems studied in this paper.

Table B.1: Summary of problems

Problem	**Target**	D
Synthetic	Gaussian	512
Synthetic	Gaussian	1024
Synthetic	Gaussian	2048
Synthetic	Gaussian	4096
Synthetic	Gaussian	8192
Poisson regression	Non-Gaussian	100
Log Gaussian Cox process	Non-Gaussian	811
IRT model	Non-Gaussian	143

B.1 Impact of hyperparameters on Gaussian example

Here we show the impact of varying choices of parameters like scheduling, learning rate, and batch size for ADVI-LR and pBaM on the synthetic Gaussian example. We always work in the setting where the rank of the low-rank component of the target and the variational distribution is the same ($K = 32$).

Figure B.1 shows the impact of scheduling the learning rate. We consider no scheduling (constant learning rate), linear and cosine scheduling. We fix the inital and final learning rate to be 0.1 and 10^{-5} respectively, and batch size $B = 8$. All three schedules converge at the same rate initially, however the constant learning rate asymptotes at a higher loss (KL divergence) once it reaches closer to the solution. Both linear and cosine scheduling give similar performance. Hence we use linear scheduling for all experiments.

Next in Figure B.2, we study the impact of learning rate and batch size. Broadly, we find that the higher learning

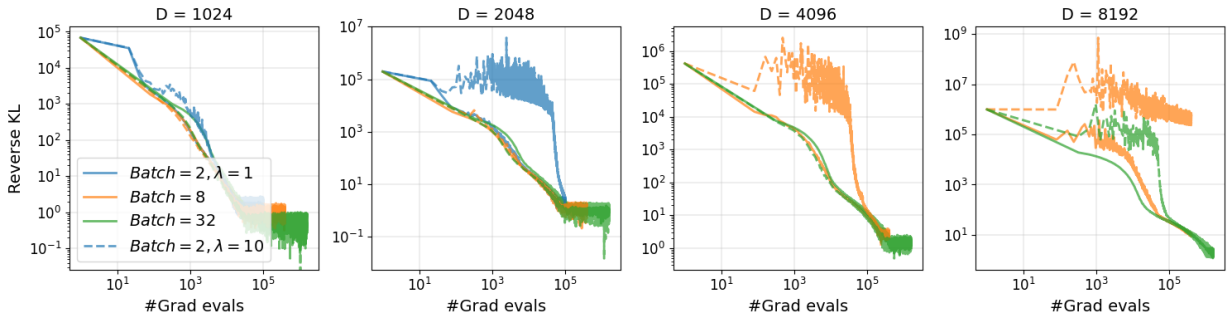

Figure B.3: Impact of changing the batch size and the learning rate parameter λ_t for pBaM.

rate converges more quickly initially, but plateaus to a larger loss before decreasing sharply again. Smaller batch sizes show faster convergence initially, while they plateau at a lower KL divergence. Hence we use batch size $B = 32$ for all the experiments. However, the conclusion of this figure along with Figure B.1 suggests that the relationship with learning rate is more complicated. Given that pBaM seems to outperform all these learning rates considered nevertheless, we show results for two best learning rates in the main text.

Finally we study the impact of batch size and learning rate parameter λ_t for pBaM in Figure B.3. Overall, the algorithm seems to get increasingly more stable for larger batch size, and smaller λ_t for a given batch size. However the convergence rate for any stable run seems to broadly be insensitive to these choices.

B.2 Gaussian process inference

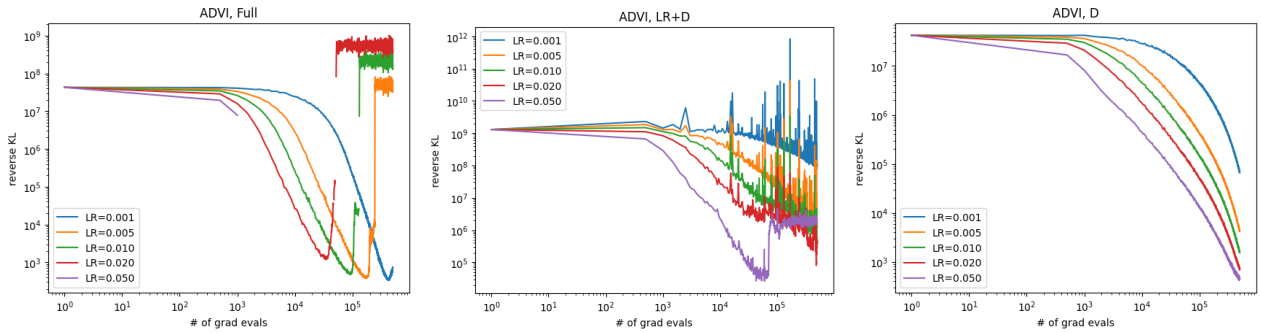

Figure B.4: Poisson regression $D = 100$ with varying learning rates for ADVI

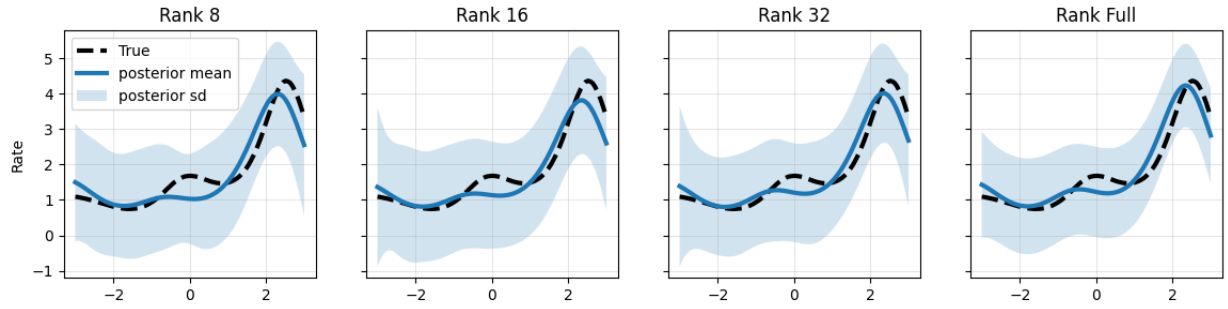

Figure B.5: Poisson regression $D = 100$. BaM/pBaM with varying ranks.

B.2.1 Poisson regression

In Section 5, we studied a $D = 100$ Poisson regression problem. First, we show in Figure B.4 the results of the different ADVI methods on a range of learning rates: $0.001, 0.005, 0.01, 0.02, 0.05$. For the final plots, the learning

rate was set to 0.02. For the full covariance family, we also tried a linear schedule and found similar results. We also found larger batch sizes to be more stable, and so in all plots for this example, we set $B = 50$. In Figure B.5 we visualize the results of BaM/pBaM for varying ranks; the estimated mean rates in the right panel of Figure 5.3, but here we additionally show the uncertainties estimated from the posterior standard deviations.

B.2.2 Log Gaussian Cox process

In this example, the dimension was 811, and so we only ran the variational inference methods with diagonal and low-rank plus diagonal families. We show in Figure B.6 ADVI for these families, varying the learning rate on a grid. In Figure B.7, we show the BaM/pBaM estimated mean rate and uncertainty computed using the variational posterior mean and standard deviation. Overall, the differences in this set of ranks were small.

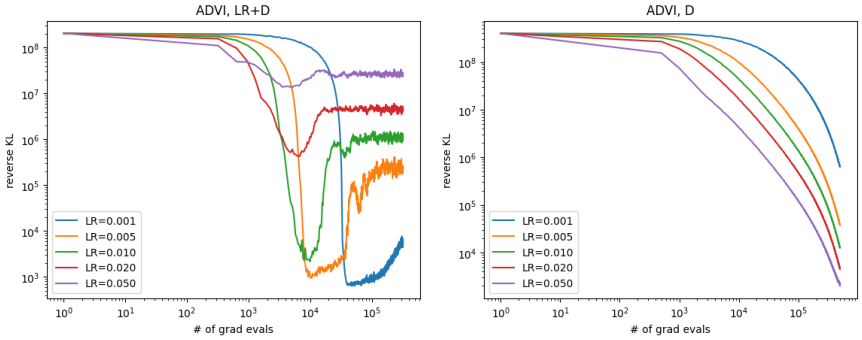

Figure B.6: LGCP, $D = 811$. ADVI with varying learning rate.

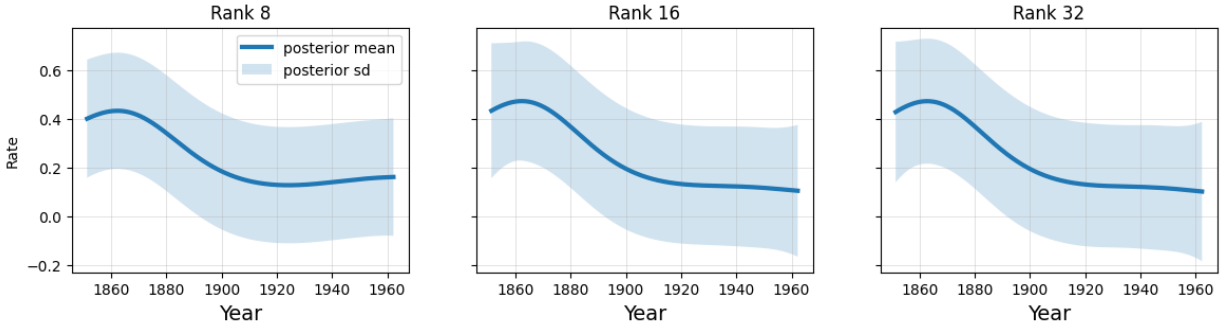

Figure B.7: LGCP, $D = 811$. BaM/pBaM with varying ranks.