Journal of Computational Physics 432 (2021) 110134

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Bayesian sparse learning with preconditioned stochastic gradient MCMC and its applications

Yating Wang [a], Wei Deng [a], Guang Lin [b,*]

[a] Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA
[b] Department of Mathematics, School of Mechanical Engineering, Department of Statistics (Courtesy), Department of Earth, Atmospheric, and Planetary Sciences (Courtesy), Purdue University, West Lafayette, IN 47907, USA

ARTICLE INFO

Article history:
Available online 26 January 2021

Keywords:
Bayesian sparse learning
Preconditioned stochastic gradient MCMC
Deep learning
Deep neural network
Adaptive hierarchical posterior
Stochastic approximation

ABSTRACT

Deep neural networks have been successfully employed in an extensive variety of research areas, including solving partial differential equations. Despite its significant success, there are some challenges in effectively training DNN, such as avoiding overfitting in over-parameterized DNNs and accelerating the optimization in DNNs with pathological curvature. In this work, we propose a Bayesian type sparse deep learning algorithm. The algorithm utilizes a set of spike-and-slab priors for the parameters in the deep neural network. The hierarchical Bayesian mixture will be trained using an adaptive empirical method. That is, one will alternatively sample from the posterior using preconditioned stochastic gradient Langevin Dynamics (PSGLD), and optimize the latent variables via stochastic approximation. The sparsity of the network is achieved while optimizing the hyperparameters with adaptive searching and penalizing. A popular SG-MCMC approach is Stochastic gradient Langevin dynamics (SGLD). However, considering the complex geometry in the model parameter space in nonconvex learning, updating parameters using a universal step size in each component as in SGLD may cause slow mixing. To address this issue, we apply a computationally manageable preconditioner in the updating rule, which provides a step-size parameter to adapt to local geometric properties. Moreover, by smoothly optimizing the hyperparameter in the preconditioning matrix, our proposed algorithm ensures a decreasing bias, which is introduced by ignoring the correction term in the preconditioned SGLD. According to the existing theoretical framework, we show that the proposed algorithm can asymptotically converge to the correct distribution with a controllable bias under mild conditions. Numerical tests are performed on both synthetic regression problems and learning solutions of elliptic PDE, which demonstrate the accuracy and efficiency of the present work.

1. Introduction

Deep neural networks have attracted extensive attention in recent times. Due to their strong potential in approximating high-dimensional nonlinear maps and the universal approximation property to represent a rich class of functions, DNNs have been successfully employed in problems from various research areas. However, effectively training DNN is still challenging

[*] Corresponding author.
E-mail address: guanglin@purdue.edu (G. Lin).

https://doi.org/10.1016/j.jcp.2021.110134
0021-9991/© 2021 Elsevier Inc. All rights reserved.

due to the difficulty of escaping local minima in nonconvex optimization and avoiding overfitting in over-parameterized networks.

Bayesian learning is appealing because of its ability to capture uncertainties in the model parameters, and MCMC sampling helps to address the overfitting issue. There has been extensive work bringing the Bayesian methods to the context of DNN optimization. The stochastic gradient Langevin dynamics (SGLD) [37] is first proposed and becomes a popular approach in the family of stochastic gradient MCMC algorithms [7,23,21]. SGLD is the first-order Euler discretization of Langevin diffusion with stationary distribution on Euclidean space. It can be viewed as adding some noise to a standard stochastic gradient optimization algorithm. Since it resembles SGD, SGLD inherits the advantage of SGD where the gradients are stochastically approximated using mini-batches. This makes MCMC scalable and provides a seamless transition between stochastic optimization and posterior sampling. It was shown that samples from SGLD will converge to samples from the true posterior distribution with annealed step size [5,37].

In DNN, the underlying models may have complicated geometric properties and possess non-isotropic target density functions [12,21,7]. When the components of parameters have different curvature, generating samples using a universal step size for every model parameter may cause slow mixing and can be inefficient. In the optimization literature, there are many approaches to accelerate the gradient descent, such as preconditioning and Newton's method [11,39,4,3]. However, naively borrowing this idea and using a preconditioning matrix in SGLD fails to produce a proper MCMC scheme, the Markov chain does not target the underlying posterior except for a few cases [21,30]. Considering that a Langevin diffusion with invariant measure can be directly defined on a Riemannian manifold, and the expected Fisher information is one typical choice for the Riemannian metric tensor [17], SGRLD is proposed [25]. Built-up from Riemannian Langevin dynamics, SGRLD is a discretization of the Riemannian Langevin dynamics and the gradients are approximated stochastically. It incorporates local curvature information in the parameter updating scheme, such that a constant step size is adequate along with all directions. However, the full expected Fisher information is usually intractable. A more computationally efficient preconditioner is needed to approximate second-order Hessian information. Preconditioned SGLD adopts the same preconditioner as introduced in RMSprop [31] as discussed in [21] which reduces the computational and storage cost. One can update the preconditioner sequentially taking into account the current gradient and the previous preconditioning matrix. The preconditioner is in a diagonal form and can handle scale differences in the target density. However, the algorithm in [21] introduces a permanent bias on the MSE due to ignoring a correction term in the updating equation.

On the other hand, DNN models are usually over-parameterized and require extensive storage capacity as well as a lot of computational power. The over specified models may also lead to bad generalization and large prediction variance. Enforcing sparsity in the network is necessary. In [13], the authors propose an adaptive empirical Bayesian method for sparse learning. The idea is to incorporate an adaptive empirical Bayesian model selection technique with SG-MCMC sampling algorithm (SGLD-SA). In SGLD-SA algorithm [13], one adopts a spike-and-slab prior and obtains a Bayesian mixture DNN model. The model parameters are sampled from the adaptive hierarchical posterior using SG-MCMC, and the hyperparameters in the priors are optimized via stochastic approximation adaptively. The algorithm automatically searches and penalizes the low probability parameters and identifies promising sparse high posterior probability models [29]. One can also apply a pruning strategy to cut off model parameters with small magnitudes to further enforce sparsity in the network [24,22]. The performance of the sparse approach is demonstrated with numerous examples, and the method is also shown to be robust to adversarial attacks. Theoretically, the authors show that the proposed algorithm can asymptotically converge to the correct distribution.

In support of the advantages and considering the issues of the above-mentioned methods, we incorporate the preconditioned SGLD methods with sparse learning. We will apply the proposed method to learn solutions of partial differential equations with heterogeneous coefficients. Numerous approaches have been proposed to numerically solve ODEs and PDEs with deep neural networks, for example, parametric PDEs [19], ODE systems driven by data [6,26], time-dependent multiscale problems [34,33] and physical informed DNN ([27,28,38,40]). Moreover, various types of network architectures are constructed to achieve efficient learning based on existing fast numerical solvers. These approaches include designing multigrid neural networks [15,18], constructing multiscale models [32,34,35], learning surrogate reduced-order models by deep convolution networks [40,36,8] and so on.

This work attempts to design an efficient sparse deep learning algorithm and apply it to learn the solution of elliptic PDE with heterogeneous coefficients. Numerical simulations for these problems are challenging since it naturally contains heterogeneity at various scales as well as uncertainties. Based on the model reduction idea, for example, generalized multiscale finite element method (GMsFEM) [14,9,10], the authors [35] design an appropriate sparse DNN structure to learn the map from the heterogeneous permeability to velocity fields in Darcy's flow. The idea is to apply locally connected/convolutional layers which can be an analogy to the upscaling and downscaling procedures in multiscale methods. However, the network is still over-parameterized. In particular, the last decoding step joins neurons representing features on the coarser level to the neurons representing the fine-scale solutions and is realized by a fully connected layer. Due to the large degrees of freedom in the fine grid solution, the number of parameters in the network will be very large and result in inefficient training. Our main contribution is to bring together preconditioned SGLD and stochastic approximation to achieve efficient and sparse learning. We propose an adaptive empirical Bayesian algorithm, where the neural network parameters are sampled from a Bayesian mixture model using PSGLD method, and the latent variables are smoothly optimized during stochastic approximation. PSGLD incorporates local curvature information in the parameter updating scheme, thus it is suitable to deal with our problem which possesses multiscale nature. More importantly, we will sequentially update the preconditioning matrix under the framework of stochastic approximation, such that the bias formed by ignoring the correction term in the sampling approaches to zero asymptotically. We theoretically show the convergence of the proposed algorithm and demonstrate its performance in several numerical experiments.

The paper is organized as follows. In Section 2, we review some basic ideas in SGLD, SGRLD. In Section 3, the sparse adaptive empirical Bayesian approach is reviewed. Our main algorithm which combines preconditioned SGLD with sparse learning is explored in Section 4. Its convergence is discussed in Section 5. Applying the proposed method to a large-p-small-n regression problem, and to learn solutions of elliptic problems with heterogeneous coefficients, its performances are presented in Section 6. A conclusion is made in the last Section 7.

2. Stochastic gradient Langevin dynamics (SGLD) and stochastic gradient Riemann Langevin dynamics (SGRLD)

Throughout the paper, we denote by $\boldsymbol{\beta}$ the model parameters with $p(\boldsymbol{\beta})$ as a prior distribution, and $D = \{d_i\}_{i=1}^N$ the entire dataset, where $d_i = (x_i, y_i)$ is an input-output pair for the model. Let $p(d|\boldsymbol{\beta})$ be the likelihood, the posterior is then $p(\boldsymbol{\beta}|D) \propto p(\boldsymbol{\beta})\prod_{i=1}^N p(d_i|\boldsymbol{\beta})$. SGLD combines the idea of stochastic gradient algorithms and posterior Bayesian sampling using Langevin dynamics. The loss gradient is approximated efficiently using mini-batches of data in SGLD, and the uncertainties in the model parameters can be captured through Bayesian learning to avoid overfitting. Let ϵ_k be the learning rate at epoch k and $\tau > 0$ be the inverse temperature, the model parameters update as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \epsilon_k \nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) + \mathcal{N}(0, 2\epsilon_k \tau^{-1})$$

where for a subset of n data points $d_k = \{d_{k1}, \cdots, d_{kn}\}$

$$\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}} \log p(\boldsymbol{\beta}) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\beta}} \log p(d_{ki}|\boldsymbol{\beta})$$

is the stochastic gradient computed using a minibatch, which is used to approximate the true gradient $\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta})$.

Complicated posterior distributions often exhibit pathological curvatures with different scales of model parameters, and any uniform learning rate may maximize the efficiency in one direction but fails entirely in exploring other regions with large curvature and small scales. Stochastic Gradient Riemann Langevin Dynamics (SGRLD) [25] is a generalization of SGLD on a Riemannian manifold. In this case, consider the probability model on a Riemann manifold with some metric tensor $G^{-1}(\boldsymbol{\beta})$, the parameter updates can be guided by the geometric information of this manifold as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \epsilon_k \Big[G(\boldsymbol{\beta}_k)\nabla_{\boldsymbol{\beta}} \tilde{L}(\boldsymbol{\beta}_k) + \Gamma(\boldsymbol{\beta}_k) \Big] + \mathcal{N}(0, 2\epsilon_k \tau^{-1} G(\boldsymbol{\beta}_k)) \tag{1}$$

where $\Gamma(\boldsymbol{\beta}_k)$ is an additional drift term and $\Gamma_i(\boldsymbol{\beta}_k) = \sum_j \frac{\partial G_{ij}(\boldsymbol{\beta}_k)}{\partial \beta_j}$. The expected Fisher information can be used as a natural metric tensor, however it is intractable in many cases. One can choose a more practical metric tensor and use it as a preconditioning matrix.

3. SGLD with stochastic approximation (SGLD-SA)

To achieve sparse learning in DNN, in [13], the authors propose an adaptive empirical Bayesian method. It assumes that the weight parameter β^{lj}, the j-th neuron in the l-th layer, follows spike-and-slab Gaussian Laplace prior

$$\pi(\beta^{lj}|\sigma^2, \gamma_{lj}) = (1-\gamma_{lj})\mathcal{L}_p(0, \sigma v_0) + \gamma_{lj}\mathcal{N}(0, \sigma^2 v_1)$$

where $\gamma_{lj} \in \{0, 1\}$ are the latent binary variable selection indicators, $\mathcal{L}_p$ is the Laplace distribution, and $\mathcal{N}$ is the Normal distribution. The variance parameters v_0 and v_1 are constants which can control the variance of Laplace and Gaussian distribution, resulting in spike-and-slab priors. The error variance σ^2 follows an inverse gamma prior $\pi(\sigma^2) = IG(\nu/2, \nu\lambda_{\boldsymbol{\gamma}}/2)$, where $\lambda_{\boldsymbol{\gamma}}$ is usually set to be constant in practice [16]. The prior for $\boldsymbol{\gamma}$ follows a Bernoulli distribution, $\pi(\gamma_l|\delta^l) = (\delta^l)^{|\gamma_l|}(1-\delta^l)^{p_l - |\gamma_l|}$, which incorporate uncertainty regarding which variables β_{lj} need be included in the model. Here, p_l is the number of model parameters in the l-th sparse layer. $|\gamma_l| = \sum_j \gamma_{lj}$, and δ^l follows $\pi(\delta^l) = \delta_l^{a-1}(1-\delta^l)^{b-1}$ where a, b are some positive constants.

Let d^m be the m-th minibatch of the dataset. The likelihood for a regression problem can be rewritten as

$$\pi(d^m|\boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\Big\{ - \frac{\sum\limits_{x_i^m \in d^m} (y_i^m - \mathcal{F}(x_i^m; \boldsymbol{\beta}))^2}{2\sigma^2} \Big\}$$

where $\mathcal{F}$ denotes a map describing the input-output relationship from x_i^m to y_i^m.

Then, the posterior follows

$$\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}, \boldsymbol{\gamma}|d^m) \propto \pi(d^m|\boldsymbol{\beta}, \sigma^2)^{\frac{N}{n}} \pi(\boldsymbol{\beta}|\sigma^2, \boldsymbol{\gamma})\pi(\sigma^2|\boldsymbol{\gamma})\pi(\boldsymbol{\gamma}|\boldsymbol{\delta})\pi(\boldsymbol{\delta}) \tag{2}$$

Algorithm 1 SGLD-SA.

```
INPUT: Initialize $\boldsymbol{\beta}_1, \boldsymbol{\rho}_1, \boldsymbol{\kappa}_1, \boldsymbol{\delta}_1, \sigma_1$. Given target sparse rate $s$, step size $\omega_k$
for all $k \leftarrow 1 : \#iterations$ do
    $\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \epsilon_k \nabla_{\boldsymbol{\beta}} Q(\cdot|d_k) + \mathcal{N}(0, 2\epsilon_k \tau^{-1})$
    $a_{lj} \leftarrow \pi(\boldsymbol{\beta}_k^{lj}|\gamma_{lj} = 1)\delta_k^l$, $b_{lj} \leftarrow \pi(\boldsymbol{\beta}_k^{lj}|\gamma_{lj} = 0)(1 - \delta_k^l)$
    $\rho_{k+1}^{lj} \leftarrow (1 - \omega_{k+1})\rho_k^{lj} + \omega_{k+1} \frac{a_{lj}}{a_{lj} + b_{lj}}$
    $\kappa_{k+1,0}^{lj} \leftarrow (1 - \omega_{k+1})\kappa_{k,0}^{lj} + \omega_{k+1} \frac{1 - \rho_{k+1}^{lj}}{v_0}$
    $\kappa_{k+1,1}^{lj} \leftarrow (1 - \omega_{k+1})\kappa_{k,1}^{lj} + \omega_{k+1} \frac{\rho_{k+1}^{lj}}{v_1}$
    $\sigma_{k+1} \leftarrow (1 - \omega_{k+1})\sigma_k + \omega_{k+1} R$
    $\delta_{k+1} \leftarrow (1 - \omega_{k+1})\delta_k^l + \omega_{k+1} \frac{\sum_j \rho_{k+1}^{jl} + a_{lj} - 1}{a_{lj} + b_{lj} + p - 2}$
    if Pruning then
        Prune the last $s$% weights with smallest magnitude
        Increase the sparse rate
```

Since there are $P = \sum_{l \in L_D} p_l$ number of model parameters, this results in 2^P choices of $\boldsymbol{\gamma}$. For large P, the binary variable $\boldsymbol{\gamma}$ is hard to optimize directly. We adopt the idea of expectation and maximization approach as introduced in [29] to build a continuous approximation from probabilistic considerations. Here the latent variables $\boldsymbol{\gamma}$ are unknown parameters and will be treated as missing data. At every iteration, the expected conditional probability for $\boldsymbol{\gamma}$ is computed given $\mathcal{D}$, current parameters $\boldsymbol{\beta}$ and other latent variables. At iteration k, instead of sampling from true posterior with respect to the whole dataset $\mathcal{D}$, one needs to sample from Q with respect to a mini-batch $\mathcal{B}$

$$Q(\boldsymbol{\beta}, \sigma, \boldsymbol{\delta}|\boldsymbol{\beta}_k, \sigma_k, \boldsymbol{\delta}_k) = \mathbb{E}_{\mathcal{B}}\left[\mathbb{E}_{\boldsymbol{\gamma}|\mathcal{D}}[\log \pi(\boldsymbol{\beta}, \sigma, \boldsymbol{\delta}, \boldsymbol{\gamma})|\mathcal{B}]\right]$$

and it can be separated as

$$Q(\boldsymbol{\beta}, \sigma, \boldsymbol{\delta}|\boldsymbol{\beta}_k, \sigma_k, \boldsymbol{\delta}_k) = Q_1(\boldsymbol{\beta}, \sigma|\boldsymbol{\beta}_k, \sigma_k, \boldsymbol{\delta}_k) + Q_2(\boldsymbol{\delta}|\boldsymbol{\beta}_k, \sigma_k, \boldsymbol{\delta}_k) + C$$

where

$$\begin{aligned} Q_1(\boldsymbol{\beta}, \sigma|\boldsymbol{\beta}_k, \sigma_k, \boldsymbol{\delta}_k) = &\frac{N}{n} \log \pi(d^m|\boldsymbol{\beta}) - \sum_{l \in L_D} \sum_{j \in p_l} \frac{(\beta^{lj})^2}{2\sigma_0^2} - \frac{p + \nu + 2}{2} \log(\sigma^2) - \\ &\sum_{l \in L_S} \sum_{j \in p_l} \Big\{ \frac{|\beta^{lj}|}{\sigma} E[\frac{1}{v_0(1 - \gamma_{lj})}] + \frac{\beta_{lj}^2}{2\sigma^2} E[\frac{1}{v_1 \gamma_{lj}}] \Big\} - \frac{\nu\lambda}{2\sigma^2} \\ Q_2(\boldsymbol{\delta}|\boldsymbol{\beta}_k, \sigma_k, \delta_k) = &\sum_{l \in L_S} \sum_{j \in p_l} \log(\frac{\delta_l}{1 - \delta_l}) E[\gamma_{lj}] + (a - 1)\log(\delta_l) + (p_l + b - 1)\log(1 - \delta_l) \end{aligned}$$

Here, we partition all model parameters $\boldsymbol{\beta}$ into two groups, one group are from dense layers L_D, another group are from sparse layers L_S. For the model parameters in sparse layers, we assume they have spike and slab Laplace-Gaussian priors as mentioned before. For the model parameters in dense layers, we assume they simply have Gaussian prior with zero mean and standard deviation $\sigma_0 = 1$. We remark that the full Bayesian hierarchical model is theoretically appealing, but is computationally slow. To speed up the inference, the priors are inferred from the empirical data without considering uncertainty.

The adaptive empirical Bayesian algorithm samples $\boldsymbol{\beta}$ from Q and iteratively optimize Q with respect to $\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\delta}$ via stochastic approximation as in Algorithm 1. We note that the update formulas of latent variables $\boldsymbol{\rho}, \boldsymbol{\kappa}, \boldsymbol{\delta}, \sigma$ are motivated by EM approach to Bayesian variable selection (EMVS) [29] In Algorithm 1, $\rho^{lj} = E[\gamma_{lj}]$, ω_k is the step size in updating latent variables, $\kappa_{k,0}^{lj} = E[\frac{1}{v_0(1-\gamma_{lj})}]$ and $\kappa_{k,1}^{lj} = E[\frac{1}{v_1 \gamma_{lj}}]$, R is the positive root to the following quadratic formula:

$$\begin{aligned} &\{N + \sum_{l \in L_s} p_l + \nu\}\sigma^2 + \{||\sum_{l \in L_s} \kappa_{k,0}^l \circ \beta_{k+1}^l||_1\}\sigma \\ &+ \{\frac{N}{n} \sum_{x_i^m \in d^m} (y_i^m - \mathcal{F}(x_i^m; \boldsymbol{\beta}))^2 + ||\sum_{l \in L_s} \kappa_{k,1}^l \circ \beta_{k+1}^l||_2^2 + \nu\lambda\} = 0 \end{aligned}$$

where $\circ$ denotes the point-wise product, $||\cdot||_1$ and $||\cdot||_2$ are the vector l_1 and l_2 norm correspondingly.

4. Preconditioned SGLD with stochastic approximation (PSGLD-SA)

As seen in Section 2, all model parameters $\boldsymbol{\beta}$ are updated using the same learning rate ϵ_k in SGLD. If the loss function has very different scales in different directions, the sampling procedure will take larger steps in directions where the model parameter has small variance, and smaller steps in directions of large variance during sampling. This causes slow mixing and slows down the convergence of sampling. However, a small enough learning rate is required to avoid divergence in the largest positive curvature direction.

Here, we will introduce a preconditioning matrix $G(\boldsymbol{\beta})$ to guide the updating directions during sampling. In gradient descent algorithms, the optimization can be improved using the second-order information, i.e., the inverse of the Hessian matrix, as the preconditioning matrix. However, it is too computationally expensive to store and invert the full Hessian during the training. An efficient approximation is to use the same preconditioner as in RMSprop [31]. The idea is to scale the gradient using a moving average of its recent norm in each iteration, so that one can adapt the step size separately for each weight. By keeping a moving average for each weight parameter from the previous step, one can control the changes between adjacent mini batches. We propose a sequentially updated preconditioner using the stochastic approximation idea as follows

$$G(\boldsymbol{\beta}_k) = diag^{-1}(\eta + \sqrt{V(\boldsymbol{\beta}_k)}) \tag{3}$$

$$V(\boldsymbol{\beta}_k) = \alpha_k V(\boldsymbol{\beta}_{k-1}) + (1-\alpha_k) g(\boldsymbol{\beta}_k) \circ g(\boldsymbol{\beta}_k) \tag{4}$$

where η is a regularization constant, and $\alpha_k = (1-\omega_k)$, $g(\boldsymbol{\beta}_k) = \nabla_{\boldsymbol{\beta}} Q$. Here the operation $diag^{-1}(\boldsymbol{v})$ means taking element-wise reciprocal of the vector $\boldsymbol{v}$ and forming a diagonal matrix. Importantly, we note that the weight parameter α_k is a sequence approaching 1 as the time step k increases, which is different from the constant α in [21]. The change in the parameters will then be

$$\triangle\boldsymbol{\beta}_k = \epsilon_k\big(G(\boldsymbol{\beta}_k)g(\boldsymbol{\beta}_k) + \Gamma(\boldsymbol{\beta}_k)\big) + \mathcal{N}(0, 2\epsilon_k \tau^{-1} G(\boldsymbol{\beta}_k)) \tag{5}$$

where $\Gamma_i(\boldsymbol{\beta}_k) = \sum_j \frac{\partial G_{ij}(\boldsymbol{\beta}_k)}{\partial \beta_j}$.

We note that in [21], $\Gamma(\boldsymbol{\beta}_k)$ is ignored in practice, and α is a constant. This produces a permanent bias $\mathcal{O}\left(\frac{(1-\alpha)^2}{\alpha^3}\right)$ on the MSE. To address this issue, we let α_k gradually approach 1 during the adaptive optimization of the latent variables, then the bias mentioned before will decrease. To be specific, for the i-th main diagonal entries, we have

$$\begin{aligned}
\left|\sum_{k=1}^{K} \Gamma_i(\boldsymbol{\beta}_k)\right| &= \left|\sum_{k=1}^{K} (1-\alpha_k) V_i^{-\frac{3}{2}}(\boldsymbol{\beta}_k) g_i(\boldsymbol{\beta}_k) \frac{\partial g_i(\boldsymbol{\beta}_k)}{\partial \beta_i}\right| \\
&= \left|\sum_{k=1}^{K} (1-\alpha_k) g_i(\boldsymbol{\beta}_k) \big[\alpha_{k-1} V_i(\boldsymbol{\beta}_{k-1}) + (1-\alpha_{k-1}) g_i(\boldsymbol{\beta}_{k-1}) g_i(\boldsymbol{\beta}_{k-1})\big]^{-\frac{3}{2}} \frac{\partial g_i(\boldsymbol{\beta}_k)}{\partial \beta_i}\right| \\
&\leq \left|\sum_{k=1}^{K} (1-\alpha_k) g_i(\boldsymbol{\beta}_k) / \left(\alpha_{k-1}^{\frac{3}{2}} V_i^{\frac{3}{2}}(\boldsymbol{\beta}_{k-1})\right) \frac{\partial g_i(\boldsymbol{\beta}_k)}{\partial \beta_i}\right| \\
&\leq \left|\sum_{k=1}^{K} (1-\alpha_k) g_i(\boldsymbol{\beta}_k) / \left(\alpha_1^{\frac{3}{2}} V_i^{\frac{3}{2}}(\boldsymbol{\beta}_{k-1})\right) \frac{\partial g_i(\boldsymbol{\beta}_k)}{\partial \beta_i}\right|
\end{aligned}$$

Then we have

$$\left|\sum_{k=1}^{K} \Gamma_i(\boldsymbol{\beta}_k)\right| \leq M \left|\sum_{k=1}^{K} \frac{(1-\alpha_k)}{\alpha_1^{\frac{3}{2}}}\right| \tag{6}$$

due to the assumption that the derivatives of the gradients are bounded, $|V_i^{-\frac{3}{2}}(\boldsymbol{\beta}_{k-1}) g_i(\boldsymbol{\beta}_k) \frac{\partial g_i(\boldsymbol{\beta}_k)}{\partial \boldsymbol{\beta}}| \leq M$ for some constant $M > 0$.

Typically, let α_k be in the form of $\alpha_k = 1 - c_1(c_2 + k)^{-\zeta}$ for some $\zeta \in (0.5, 1]$, and constants c_1, c_1, we can see that the bias introduced $\sum_{k=1}^{K} \frac{(1-\alpha_k)^2}{\alpha_1^3}$ on the MSE will approach 0 as $K \to \infty$.

Thus, our proposed adaptive preconditioned SGLD samples $\boldsymbol{\beta}$ and optimizes σ^2, γ, δ as in Algorithm 2.

Algorithm 2 PSGLD-SA.

```
INPUT: Initialize $\boldsymbol{\beta}_1, \boldsymbol{\rho}_1, \boldsymbol{\kappa}_1, \boldsymbol{\delta}_1, V_1$, let $\alpha_1 = 0.9, \eta = 10^{-3}$
for all  $k \leftarrow 1 : \#iterations$ do
    $g(\boldsymbol{\beta}_k) \leftarrow \nabla_{\boldsymbol{\beta}} Q(\cdot|\boldsymbol{d}_k)$
    if $k == 1$ then
        $V(\boldsymbol{\beta}_k) \leftarrow g(\boldsymbol{\beta}_k) \circ g(\boldsymbol{\beta}_k)$
    else
        $V(\boldsymbol{\beta}_k) \leftarrow (1-\alpha_k)V(\boldsymbol{\beta}_{k-1}) + \alpha_k g(\boldsymbol{\beta}_k) \circ g(\boldsymbol{\beta}_k)$
    $G(\boldsymbol{\beta}_k) \leftarrow diag^{-1}(\eta + \sqrt{V(\boldsymbol{\beta}_k)})$
    $\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \epsilon_k\big(G(\boldsymbol{\beta}_k)g(\boldsymbol{\beta}_k)\big) + G^{\frac{1}{2}}(\boldsymbol{\beta}_k)\mathcal{N}(0, 2\epsilon_k\tau^{-1})$
    Updating hyperparameters by running steps 3-11 in Algorithm 1
```

5. Convergence results

Now, we will discuss the weak convergence of our proposed algorithm PSGLD-SA. First, we will take a look at the hyperparameters. Denote by $\boldsymbol{\theta}$ all the hyperparameters $(\rho, \kappa, \sigma, \delta)$. The stochastic approximation attempts to get the optimal $\boldsymbol{\theta}_*$ based on the asymptotically target distribution $\pi(\boldsymbol{\beta}, \boldsymbol{\theta}_*)$. Define $H(\boldsymbol{\theta}, \boldsymbol{\beta}) = f_{\boldsymbol{\theta}}(\boldsymbol{\beta}) - \boldsymbol{\theta}$, where $f_{\boldsymbol{\theta}}(\boldsymbol{\beta})$ represents a function to obtain optimal $\boldsymbol{\theta}$ given current model parameters $\boldsymbol{\beta}$. Denote by its mean field function $h(\boldsymbol{\theta}) = \mathbb{E}[H(\boldsymbol{\theta}, \boldsymbol{\beta})]$. SA aims to solve the fixed point equation $\int f_{\boldsymbol{\theta}}(\boldsymbol{\beta})\pi(\boldsymbol{\beta}, \boldsymbol{\theta})d\boldsymbol{\beta} = \boldsymbol{\theta}$, which is to find the root $\boldsymbol{\theta}_*$ of the equation $h(\boldsymbol{\theta}) = 0$. As described in Algorithm 2, in each iteration, we first sample $\boldsymbol{\beta}_{k+1}$ using reconditioned SGLD based on $\boldsymbol{\theta}_k$, then update the latent variables using

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \omega_{k+1} H(\boldsymbol{\theta}_k, \boldsymbol{\beta}_{\boldsymbol{k+1}}),$$

where the map $f_{\boldsymbol{\theta}}$ is motivated by EMVS. However, we only use a small set of data of n samples instead of the full set in the computation of obtaining optimal latent variables. This will result the bias $\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})$ at each step. That is, we actually use $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \omega_{k+1}\tilde{H}(\boldsymbol{\theta}_k, \boldsymbol{\beta}_{\boldsymbol{k+1}})$ with

$$\tilde{H}(\boldsymbol{\beta}, \boldsymbol{\theta}) = H(\boldsymbol{\beta}, \boldsymbol{\theta}) + \Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1}), \tag{7}$$

and we assume $\mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||^2 \leq C^2$ for some constant C.

Following a similar proof in [13], under suitable assumptions, the adaptive empirical Bayesian method for sparse approximation algorithm has the following convergence results. The details of the proof are in Appendix A.

Theorem 1. *For a sufficiently large k_0, there exists a constant λ such that*

$$\mathbb{E}\left[||\boldsymbol{\theta}_k - \boldsymbol{\theta}^*||^2\right] = \mathcal{O}(\lambda\omega_k + \sup_{i \geq k_0} \mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||).$$

Next, we present a weak convergence result of the model parameters.

Corollary 1. *Under Assumption 2 in [5], the bias and MSE of PSGLD-SA for K steps with decreasing step size ϵ_k is bounded, the distribution of $\boldsymbol{\beta}_k$ converges weakly to the target posterior with a controllable bias, as $\epsilon_k \to 0$ and $k \to \infty$.*

Proof. With geometric information for probability models, the Langevin diffusion on the manifold is described by

$$d\boldsymbol{\beta}(\boldsymbol{t}) = G(\boldsymbol{\beta}(t))\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}(t), \boldsymbol{\theta}^*) + \Gamma(\boldsymbol{\beta}(t)) + G^{\frac{1}{2}}(\boldsymbol{\beta}(t))d\mathcal{B}_t \tag{8}$$

where $\mathcal{B}_t$ is the Brownian motion.

Denote by $\mathcal{L}$ the generator for (8), then

$$\mathcal{L} = \left[G(\boldsymbol{\beta_k})\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta_k}, \boldsymbol{\theta}^*) + \Gamma(\boldsymbol{\beta_k})\right] \cdot \nabla_{\boldsymbol{\beta}} + 2G^{\frac{1}{2}}(\boldsymbol{\beta})G^{\frac{1}{2}}(\boldsymbol{\beta_k})^T : \nabla_{\boldsymbol{\beta}_k}\nabla_{\boldsymbol{\beta}}^T \tag{9}$$

The generator $\mathcal{L}$ is associated with the backward Kolmogorov equation

$$\mathbb{E}[\phi(\boldsymbol{\beta_k})] = e^{t\mathcal{L}}\phi(\boldsymbol{\beta}_0)$$

In PSGLD-SA, one will sample from the adaptive hierarchical posterior using (3) (5), and gradually optimize the latent variables through stochastic approximation.

Write the local generator of our proposed algorithm as

$$\tilde{\mathcal{L}}_k = \left[G(\boldsymbol{\beta_k})\tilde{g}_k\right] \cdot \nabla_{\boldsymbol{\beta}} + 2G^{\frac{1}{2}}(\boldsymbol{\beta})G^{\frac{1}{2}}(\boldsymbol{\beta_k})^T : \nabla_{\boldsymbol{\beta}_k}\nabla_{\boldsymbol{\beta}}^T \tag{10}$$

where $\tilde{\mathcal{L}}_k = \mathcal{L} + \Delta V_k$, and

$$\Delta V_k = \left[G(\boldsymbol{\beta_k})\left(\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta_k}, \boldsymbol{\theta}^*) - \tilde{g}_k\right) + \Gamma(\boldsymbol{\beta_k})\right] \cdot \nabla_{\boldsymbol{\beta}}.$$

Thus

$$\tilde{g}_k = \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_k) + \xi_k + \mathcal{O}(k^{-\zeta} + \sup_{i\geq k_0} \mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||)$$

where ξ_k is a random vector denoting the difference between the true gradient and stochastic gradient, and $\mathcal{O}(k^{-\gamma} + \sup_{i\geq k_0} \mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||)$ is the bias term generated by SA.

Given a test function ϕ of interest, let $\bar{\phi}$ be the posterior average of ϕ under the invariant measure of the SDE (8). Let $\boldsymbol{\beta}_k$ be the numerical samples, and define $\hat{\phi} = \sum_{k=1}^K \frac{\epsilon_k}{S_K}\phi(\boldsymbol{\beta}_k)$, where $S_K = \sum_{k=1}^K \epsilon_k$. Let ψ be a functional which solves the Poisson equation

$$\mathcal{L}\psi(\boldsymbol{\beta}_k) = \phi(\boldsymbol{\beta}_k) - \bar{\phi}.$$

Following a similar proof as in [5], one can obtain the following results. The bias of PSGLD-SA is

$$|\mathbb{E}\hat{\phi} - \bar{\phi}| \leq \frac{1}{S_K}|\mathbb{E}\psi(\boldsymbol{\beta}_K) - \psi(\boldsymbol{\beta}_0)| + \sum_{k=1}^K \frac{\epsilon_k}{S_K}\mathbb{E}||\Delta V_k \psi(\boldsymbol{\beta}_{k-1})|| + C\sum_{k=1}^K \epsilon_k^2$$

Formally, we note that in the above bound for the bias, the term $\sum_{k=1}^K \frac{\epsilon_k}{S_K}\mathbb{E}||\Delta V_k \psi(\boldsymbol{\beta}_{k-1})||$ is important. It is related to the bias introduced by stochastic approximation and ignoring $\Gamma(\boldsymbol{\beta}_k)$. By Assumption 2 in [5] on the smoothness and boundedness of the functional ψ, and the boundedness of the preconditioner, it is easy to see that the bias introduced by stochastic approximation can be decomposed into (1) the term $\sum_{k=1}^K \frac{\epsilon_k k^{-\zeta}}{S_K}$ in the bias, which approaches 0 as $K \to \infty$, and (2) $\sum_{k=1}^K \frac{\epsilon_k}{S_K} \sup_{i\geq k_0}\mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})|| = \sup_{i\geq k_0}\mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||$ which is a controllable bias. The bias introduced by ignoring $\Gamma(\boldsymbol{\beta}(t))$ can be bounded by $\sum_{k=1}^K (1-\alpha_k)\alpha_1^{-\frac{3}{2}} = \mathcal{O}(\sum_{k=1}^K k^{-\zeta})$ according (6), which goes to 0 as $K \to \infty$.

The MSE of PSGLD-SA can be bounded by

$$\mathbb{E}(\hat{\phi} - \bar{\phi})^2 \leq C\left(\sum_{k=1}^K \frac{1}{S_K^2} + \sum_{k=1}^K \frac{\epsilon_k^2}{S_K^2}\mathbb{E}||\Delta V_k \psi(\boldsymbol{\beta}_{k-1})||^2 + \frac{(\sum_{k=1}^K \epsilon_k)^2}{S_K^2}\right)$$

which converges as long as $\sup_k \mathbb{E}||\Delta V_k \psi(\boldsymbol{\beta}_{k-1})||^2$ is bounded.

Thus, we conclude that, as $\epsilon_k \to 0$ and $k \to \infty$, the distribution of $\boldsymbol{\beta}_k$ converges weakly to the target posterior with a controllable bias. The bias is expected to decrease if we enlarge the minibatch size to approximate the gradient. □

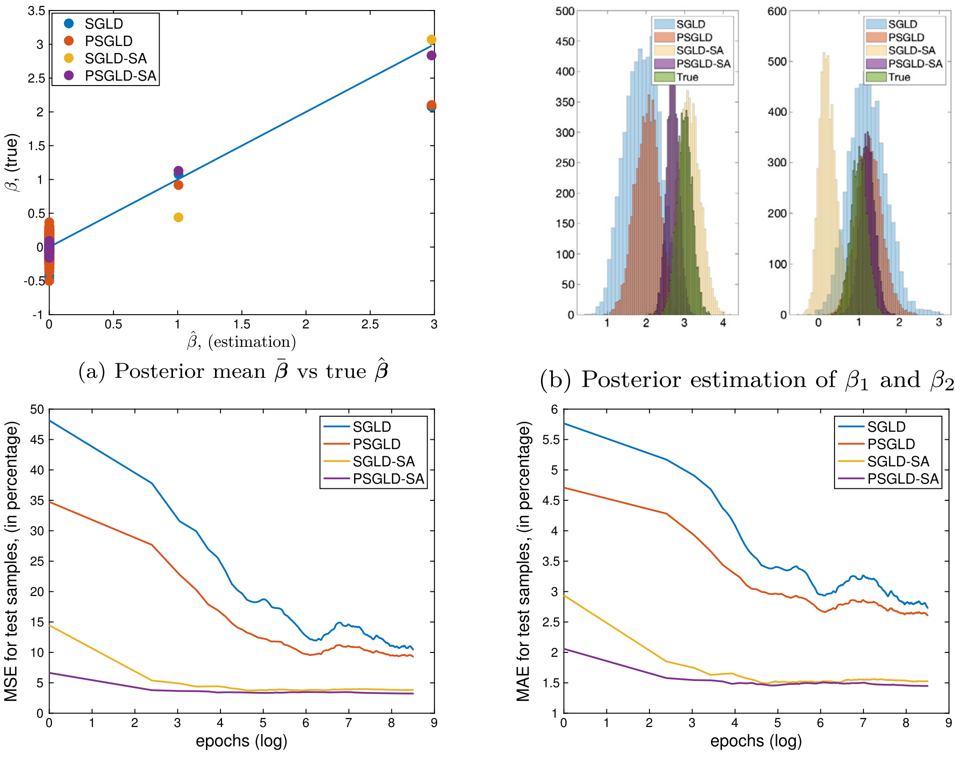
6. Numerical example

6.1. Small n large p problem

We first test on a linear regression problem, where the model parameters $\boldsymbol{\beta} \in \mathbb{R}^p$, and predictors $X \in \mathbb{R}^{n\times p}$. We take a dataset with $n = 100$ observations and $p = 200$ predictors.

For the first test (section 6.1 test 1), we use $\mathcal{N}_p(0, \Sigma)$ with $\Sigma_{ij} = 0.6^{|i-j|}$ to simulate predictor values X. The responses $y = X\boldsymbol{\beta} + \epsilon$, and $\epsilon \sim \mathcal{N}_n(0, 3I_n)$. $\boldsymbol{\beta}_1 \sim \mathcal{N}(3, 0.2)$, $\boldsymbol{\beta}_2 \sim \mathcal{N}(1, 0.2)$, and $\beta_j = 0$, for $j = 1, \cdots, p$. The hyperparameters used for SGLD-SA and PSGLD-SA are both: $v_0 = 0.05$, $v_1 = 5$, $\delta = 0.5$, $b = p$, $a = 1$, $\lambda = 1$, $\nu = 1$. For the preconditioner used for PSGLD-SA, we choose $\alpha = 0.999$. The learning rate for SGLD is $\epsilon_k = 0.001$, for the other three methods is $\epsilon_k = 0.01$. The step size to update latent variables is $\omega_k = 10 \times (k + 100)^{-0.7}$. The performances of SGLD, SGLD-SA, PSGLD and PSGLD-SA are compared and presented in Fig. 1. It shows that both SGLD-SA and PSGLD-SA work better compared with SGLD and PSGLD in terms of detecting the sparsity in model parameters. The true variances in the model parameters $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are both 0.2, and they can be captured correctly in both SGLD-SA and PSGLD-SA. However, PSGLD-SA gives the best approximation for the mean of each component of $\boldsymbol{\beta}$. Moreover, Fig. 1 (c)-(d) shows the testing error curves. It can be seen that PSGLD converges faster compared to vanilla SGLD, and PSGLD-SA gives better results compared to SGLD-SA.

In the second test (section 6.1 test 2), we use $\mathcal{N}_p(0, \Sigma)$ with $\Sigma_{ij} = 0.8^{1/2|i-j|}$ to simulate predictor values X. The model parameters are now $\beta_1 \sim \mathcal{N}(3, 0.2)$, $\beta_2 \sim \mathcal{N}(1, 0.8)$ with different scales in variance. The hyperparameters used for SGLD-SA and PSGLD-SA are both: $v_0 = 0.1$, $v_1 = 100$, $\delta = 0.5$, $b = p$, $a = 1$, $\lambda = 1$, $\nu = 1$, $\alpha = 0.999$. The learning rates for SGLD-SA



(a) Posterior mean $\bar{\boldsymbol{\beta}}$ vs true $\hat{\boldsymbol{\beta}}$

(b) Posterior estimation of β_1 and β_2

(c) Testing MSE error history

(d) Testing MAE error history

Fig. 1. Section 6.1 test 1. Large p small n regression for parameters with uniform scale. In sub-figures (c)-(d), the x-axes are in log scale. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

and PSGLD-SA are $\epsilon_k = 0.01$, $\epsilon_k = 0.05$ correspondingly. The response values y and the true regression coefficients are set to be similar as before. In this case. We also compare the performance of SGLD, SGLD-SA, PSGLD, and PSGLD-SA and present them in Fig. 2. In this example, we see that PSGLD-SA outperforms other approaches clearly in quantifying the uncertainties of $\boldsymbol{\beta}$. However, SGLD, and SGLD-SA cannot capture the uncertainties properly. Preconditioned methods provide better results according to the testing error.

6.2. Elliptic problem with heterogeneous coefficients

Next, we apply the proposed approach to solve the elliptic problem with heterogeneous coefficients. The mixed formulation of the elliptic problem reads:

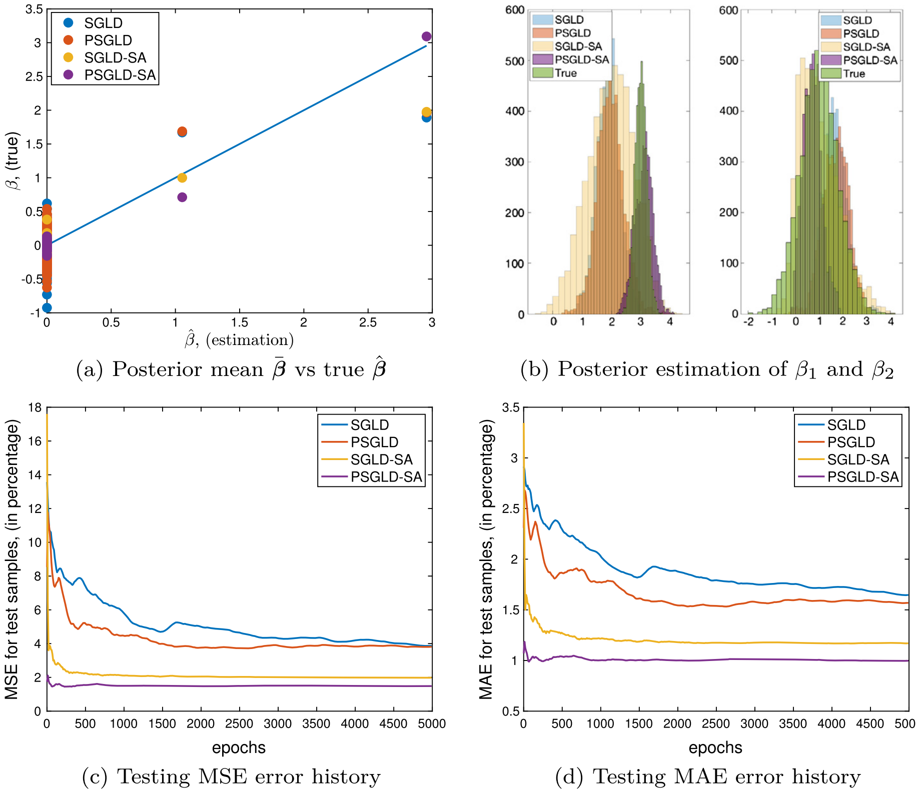
$$\begin{aligned} \kappa^{-1}u + \nabla p = 0 \quad & \text{in} \quad \Omega \\ \text{div}(u) = f \quad & \text{in} \quad \Omega \\ u \cdot n = u_N \quad & \text{on} \quad \Gamma_N \\ p = p_D \quad & \text{on} \quad \Gamma_D \end{aligned}$$

where κ represents the heterogeneous permeability field which can be generated using Karhunen-Loeve expansion. $f = 1$ is a constant source term, Ω is a squared computational domain $[0,1] \times [0,1]$, and $\Gamma_N \cup \Gamma_D = \partial\Omega$. The boundary conditions are $u_N = 0$ at $[0,1] \times \{0\}$ and $[0,1] \times \{1\}$, $p_D = 1$ at $\{0\} \times [0,1]$, and $p_D = 0$ at $\{1\} \times [0,1]$.

Specifically, the permeability field $\kappa(x;\mu)$ can be constructed as follows:

$$\kappa^H(x;\mu) = \kappa_0 + \sum_{j=1}^{p} \mu_j \sqrt{\xi_j}\, \Phi_j(x)$$

where κ_0 is a constant permeability, denotes the mean of the random field. $\sum_{j=1}^{p} \mu_j \sqrt{\xi_j}\, \Phi_j(x)$ corresponds to a random contribution obtained from Karhunen-Loeve expansion, and describes the uncertainty in the permeability field. μ_j are random numbers drawn from i.i.d. $N(0,1)$. $(\sqrt{\xi_j}, \Phi_j(x))$ are the eigen-pairs obtained from a Gaussian covariance kernel:



(a) Posterior mean $\bar{\beta}$ vs true $\hat{\beta}$

(b) Posterior estimation of β_1 and β_2

(c) Testing MSE error history

(d) Testing MAE error history

Fig. 2. Section 6.1 test 2. Large p small n regression for parameters with different scale.

$$\mathrm{Cov}(x_i, y_i; x_j, y_j) = \sigma \exp\left(\frac{-|x_i - x_j|^2}{l_x^2} - \frac{|y_i - y_j|^2}{l_y^2}\right)$$

where we choose $[l_x, l_y] = [0.2, 0.3]$, $\sigma = 2$ and $p = 32, 64, 128$ in our example.

In the discretized system, we use RT_0 element for the velocity space V_h, and piecewise constant element P_0 for the pressure solution space Q_h.

$$\begin{aligned} a(u, v) + b(v, p) &= \int\limits_{\Gamma_\Omega} p_D v \cdot n ds && \text{for all } v \in V_h \\ b(u, q) &= -(f, q) && \text{for all } q \in Q_h \end{aligned}$$

where $a(u, v) = \int_\Omega \kappa^{-1} u \cdot v$, and $b(v, p) = -\int_\Omega p \, \mathrm{div} v$.

The discrete system has the following matrix formulation

$$\begin{bmatrix} A_h(\kappa) & B_h^T \\ B_h & 0 \end{bmatrix} \begin{bmatrix} u_h \\ p_h \end{bmatrix} = \begin{bmatrix} G_D \\ -F \end{bmatrix} \tag{11}$$

However, due to the multiscale nature of κ, a sufficiently fine mesh is required to resolve all scale properties. Thus, the fine scale matrix $\begin{bmatrix} A_h(\kappa) & B_h^T \\ B_h & 0 \end{bmatrix}$ has a large size, leading to some difficulties in solving the linear system. To overcome these, one can develop a reduced order model as a surrogate. Numerous mixed multiscale methods have been explored [9,2,1]. For example, in [9], one aims to construct velocity multiscale basis in each local coarse region, and use the piecewise constant on coarse grid to approximate the pressure. Typically, let N_u^H be the dimension of the multiscale velocity space, and denote by R_u the matrix assembled using multiscale velocity basis in every row, then R_u maps from $\mathbb{R}^{N_u^h}$ to $\mathbb{R}^{N_u^H}$, where N_u^h is the fine degrees of freedom for the velocity. Similarly, denote by R_p the matrix containing coarse grid piecewise constant basis for pressure which maps from $\mathbb{R}^{N_p^h}$ to $\mathbb{R}^{N_p^H}$. Then one can rewrite the system (11) in the following form

$$\begin{bmatrix} A_H & B_H^T \\ B_H & 0 \end{bmatrix} \begin{bmatrix} u_H \\ p_H \end{bmatrix} = \begin{bmatrix} R_u & 0 \\ 0 & R_p \end{bmatrix} \begin{bmatrix} A_h(\kappa) & B_h^T \\ B_h & 0 \end{bmatrix} \begin{bmatrix} R_u^T & 0 \\ 0 & R_p^T \end{bmatrix} \begin{bmatrix} u_H \\ p_H \end{bmatrix} = \begin{bmatrix} 0 \\ -F_H \end{bmatrix} \tag{12}$$

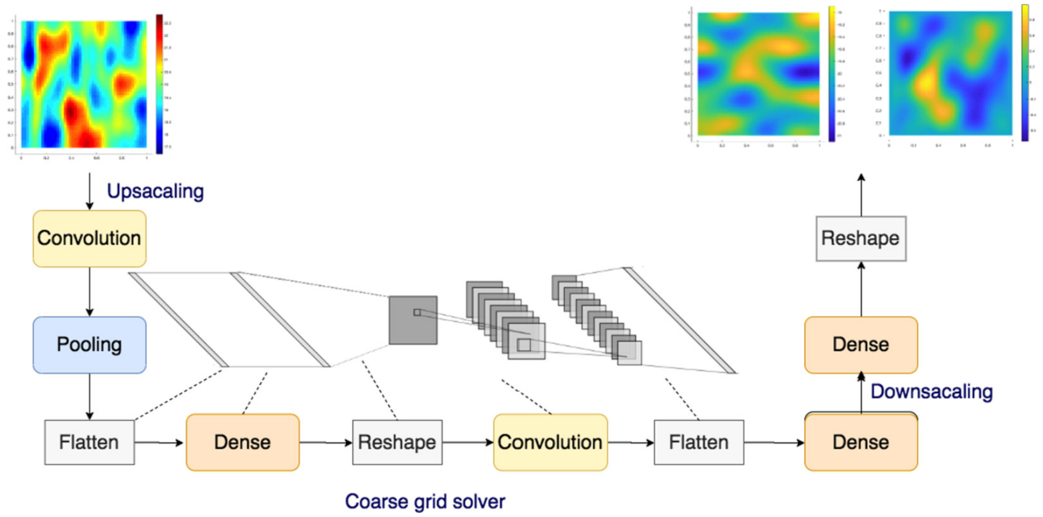


Fig. 3. An illustration of the network architecture for flow approximation.

where $\begin{bmatrix} R_u & 0 \\ 0 & R_p \end{bmatrix}$ can be viewed as an encoder which maps from fine grid to coarse grid (upscaling), and $\begin{bmatrix} R_u^T & 0 \\ 0 & R_p^T \end{bmatrix}$ acts as an decoder which maps from coarse grid to fine grid (downscaling).

The coarse grid solver reveals its efficiency when we need to solve flow problems with varying source or boundary conditions, while with a fixed permeability field. However, in a lot of applications, it is more interesting to solve for the velocity u given different κ. When the permeability fields vary, one needs to reconstruct the multiscale basis (reconstruct the matrix R_u) in the above-mentioned multiscale method framework, which is not practical.

As discussed in [35], we will construct an encoding-decoding type of network to approximate the relationship between the permeability fields κ and fine grid velocity solution u_h. That is, $u = \mathcal{N}(\kappa; \theta)$. The proposed network structure is in analogy to the coarse-scale solver but will take permeability fields as input without constructing a set of multiscale bases for each case.

The idea is to first apply a few convolution layers to extract features from the input permeability with size $\sqrt{N_p^h} \times \sqrt{N_p^h}$, and then project the extracted features on a coarser mesh by employing an average pooling layer. The intermediate output is then flattened and is linked to N_p^H neurons with a fully connected layer. This procedure is in analogy to upscaling. We will then reshape the hidden coarse grid features to an image with size $\sqrt{N_p^H} \times \sqrt{N_p^H}$. A few locally connected layers or convolution layers are followed to mimic the coarse grid solver.

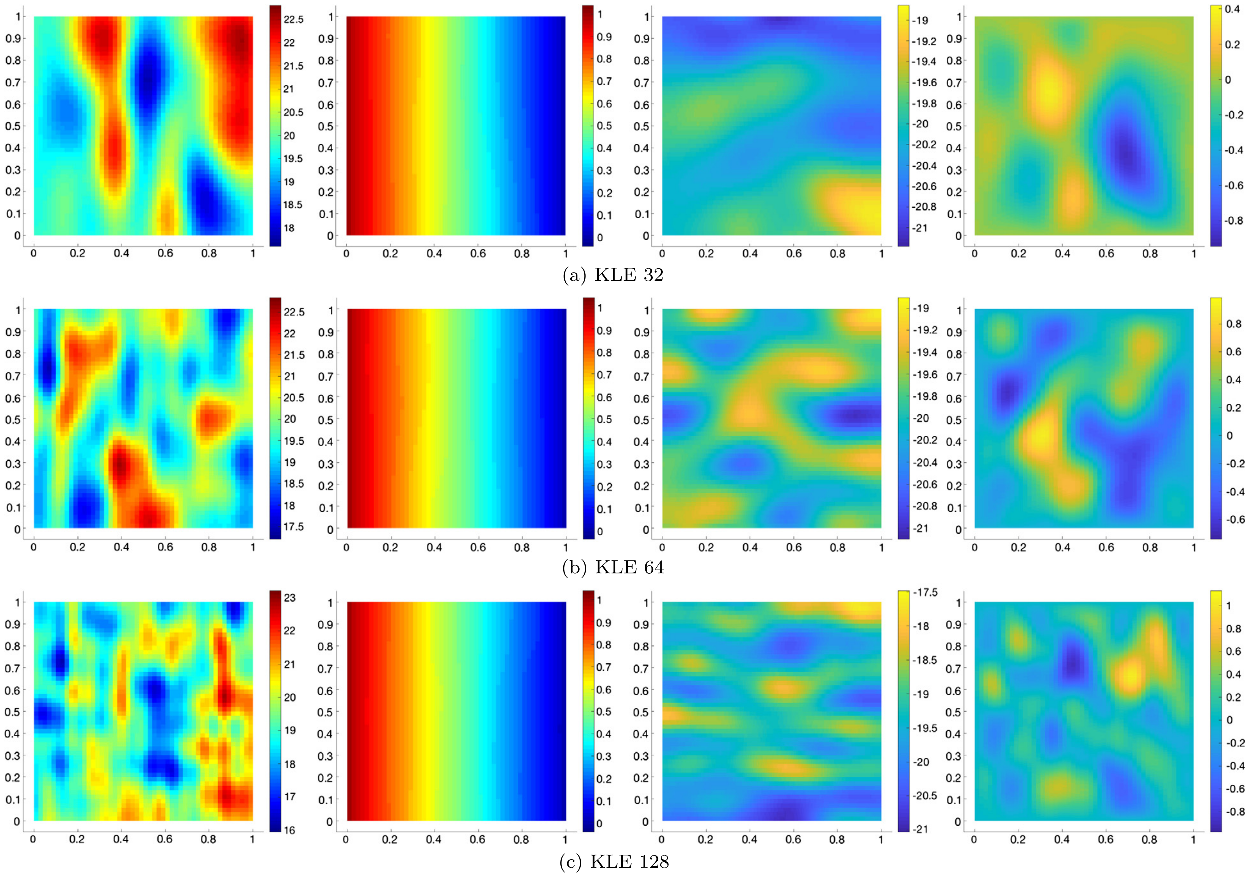
After that, the resulting hidden features are flattened again and are fully connected to N_u^H neurons in the next layer, where N_u^H is the degrees of freedom of the multiscale velocity space. It is natural to represent the coarse grid velocity using a vector since the degrees of freedom are not located at coarse grid centers, which makes it not obvious to reshape it as a square image. Finally, we decode the coarse level features using a densely connected layer, and we obtain a fine grid velocity output with dimension N_u^h. The network architecture is illustrated in Fig. 3.

However, the last downscaling layer is still fully connected. Due to the large degrees of freedom for the velocity solution, the last fully connected layer contributes a very large number of trainable weights. Here, we would like to use our proposed sparse learning method to tackle this difficulty.

The training and testing data can be generated by solving the equations with a mixed finite element method on the fine grid for various permeability fields. An illustration of the permeability fields for $p = 32, 64, 128$ and corresponding solutions are presented in 4. We can see that when p becomes larger, the velocity solutions exhibit much more scale features.

We generate 1500 samples pairs (κ_i, u_h^i), and randomly pick 1300 samples to train the network, and take the rest of the samples for validation. The size of an input permeability is 50×50, an output velocity solution vector is $5,100$. The network first uses 2 convolution layers with window size 3×3, and 64 and 32 channels, respectively. Then, an average pooling layer with pool size 2×2 is followed by a flatten layer and then a fully connected layer with 100 nodes. This part of the network can be viewed as an encoder. Then, a reshaping layer, two convolution layers, a flatten layer, and a dense layer with 800 neurons are used to mimic the coarse grid solver. Finally, a dense layer is used as a decoder. The total number of parameters is $8,252,320$ in the entire network, and the layers we choose to perform sparse learning contain $6,110,624$ parameters.

The numerical results using SGLD, PSGLD, SGLD-SA and PSGLD-SA are presented in Table 1. Denote by $e_1 = \frac{\|u_{\text{pred}} - u_{\text{true}}\|_{L^2}}{\|u_{\text{true}}\|_{L^2}}$ and $e_2 = \frac{\|u_{\text{pred}} - u_{\text{true}}\|_{L^2_\kappa}}{\|u_{\text{true}}\|_{L^2_\kappa}}$ where $||u||_{L^2_\kappa} = \int_\Omega \kappa^{-1}|u|^2$. The mean relative errors e_1 and e_2 among 300 testing samples are shown. We can see that the results using PSGLD-SA outperform SGLD-SA in all three cases when $p = 32, 64, 128$. Some

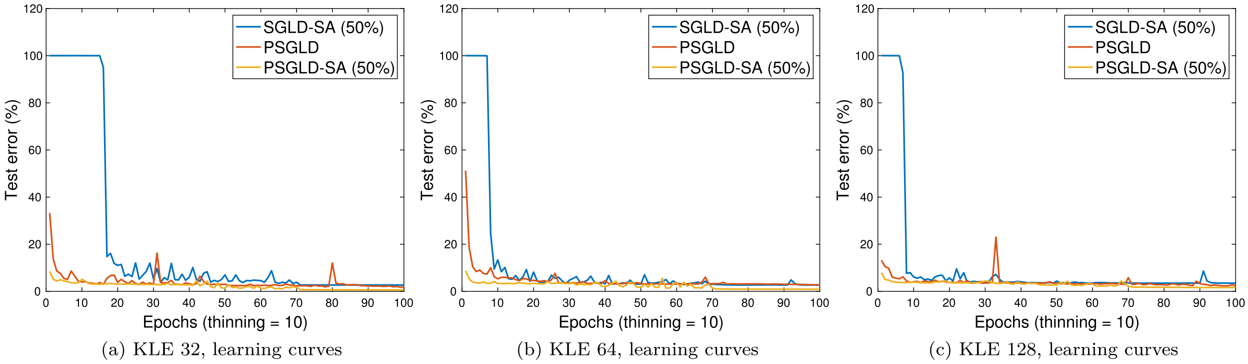


(a) KLE 32

(b) KLE 64

(c) KLE 128

Fig. 4. Illustrations of the permeability fields when using different number of terms in KLE expansion and corresponding solutions. In each subplot, permeability (upper left), pressure solution (upper right), horizontal velocity magnitude (lower left) and vertical velocity magnitude (lower right).



(a) KLE 32, learning curves

(b) KLE 64, learning curves

(c) KLE 128, learning curves

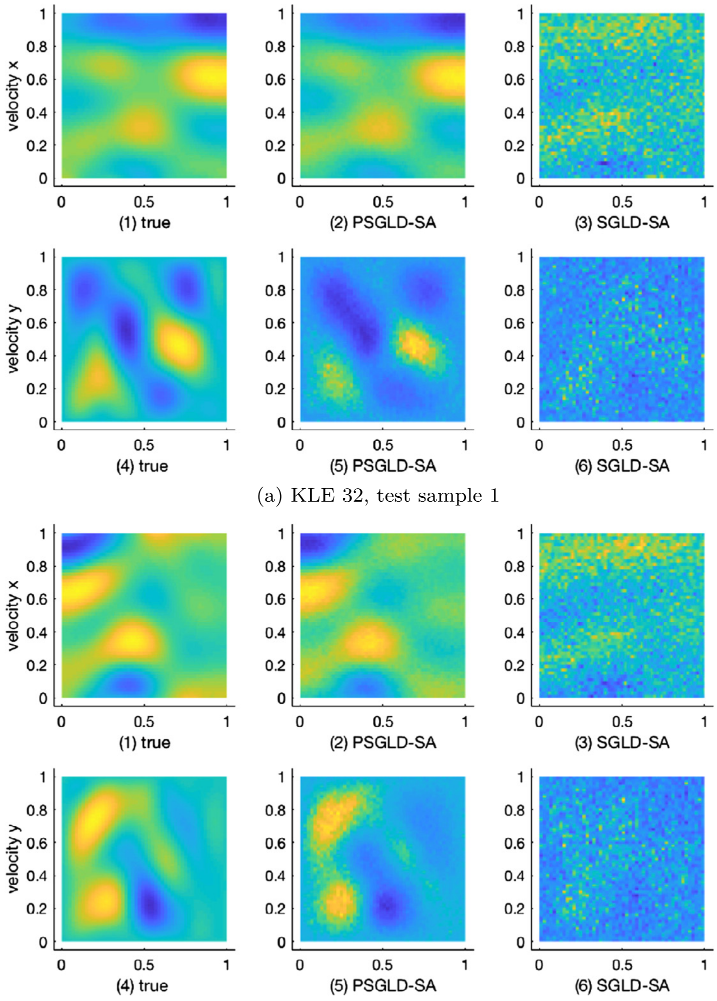
Fig. 5. Learning curves. In each sub-figure, there are comparison among test errors SGLD with sparse approximation, vanilla PSGLD, and PSGLD with sparse approximation.

predicted results for different values of p in KLE are presented in Figs. 6, 7 and 8. We can actually see that the predictions using SGLD-SA are very poor, but the results using PSGLD-SA are very similar to the ground truth.

In this example, we choose the sparse rate to be 50% and 70%. By choosing appropriate hyperparameters, we can achieve similar accuracy for the dense network and sparse network as shown in Table 1. This indicates that enforcing sparsity using our method can maintain accuracy while reducing storage/computational cost. However, if the sparse rate is too large, we find it is hard to get comparable results since over sparse network may not be sufficient to represent the properties of the target map of interest. On the other hand, comparing PSGLD and SGLD, we notice that applying preconditioners can provide better results. The learning curves are presented in 5. It shows that PSGLD-SA converges faster than SGLD-SA or vanilla PSGLD in all three cases.

Table 1
Mean errors between the true and predicted velocity solutions using SGLD, PSGLD, SGLD-SA, and proposed PSGLD-SA. Mean errors of 300 testing cases.

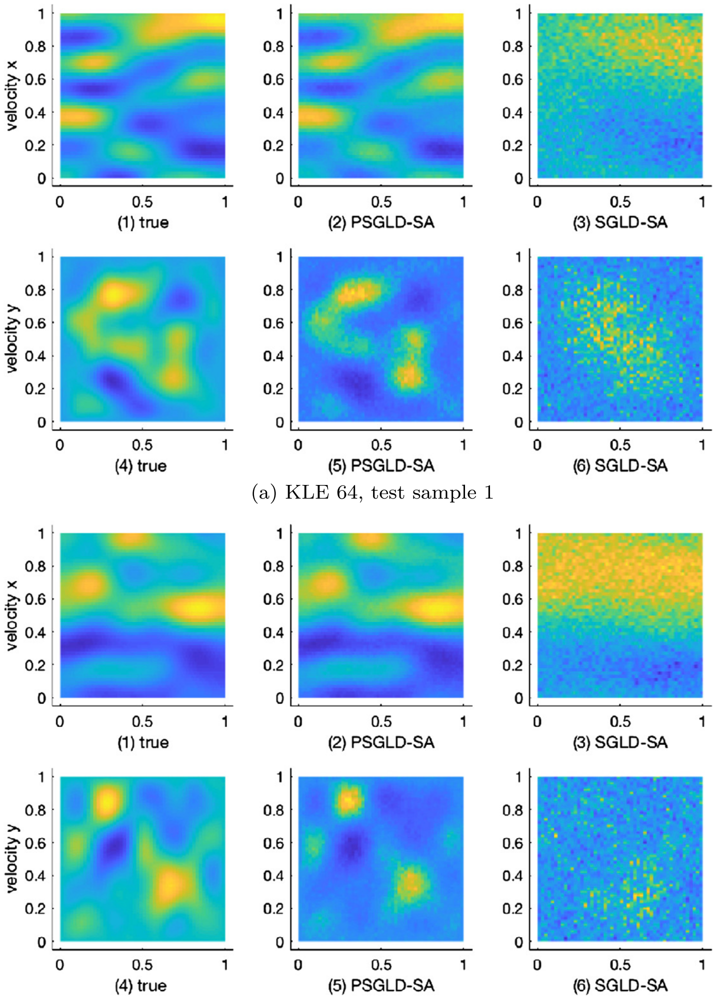
Dense		
	PSGLD (e_1/e_2 %)	SGLD(e_1/e_2 %)
KLE32	0.75/0.57	2.37/2.17
KLE64	0.82/0.63	2.38/2.25
KLE128	2.13/1.93	2.90/2.60
	PSGLD-SA (e_1/e_2 %)	SGLD-SA (e_1/e_2 %)
Sparse rate 50%		
KLE32	0.59/0.56	2.67/2.35
KLE64	0.78/0.58	2.68/2.41
KLE128	1.60/1.31	3.47/3.00
Sparse rate 70%		
KLE32	0.58/0.51	2.28/2.10
KLE64	0.76/0.61	2.40/2.97
KLE128	1.79/1.60	3.51/3.02



(a) KLE 32, test sample 1

(b) KLE 32, test sample 2

Fig. 6. KLE 32. True and prediction solutions. In each sub-figure, the first row represents horizontal velocity solution magnitude, the second row represents vertical velocity solution magnitude. From left to right: true, PSGLD-SA prediction, SGLD-SA prediction.



(a) KLE 64, test sample 1

(b) KLE 64, test sample 2

Fig. 7. KLE 64. True and prediction solutions. In each sub-figure, the first row represents horizontal velocity solution magnitude, the second row represents vertical velocity solution magnitude. From left to right: true, PSGLD-SA prediction, SGLD-SA prediction.

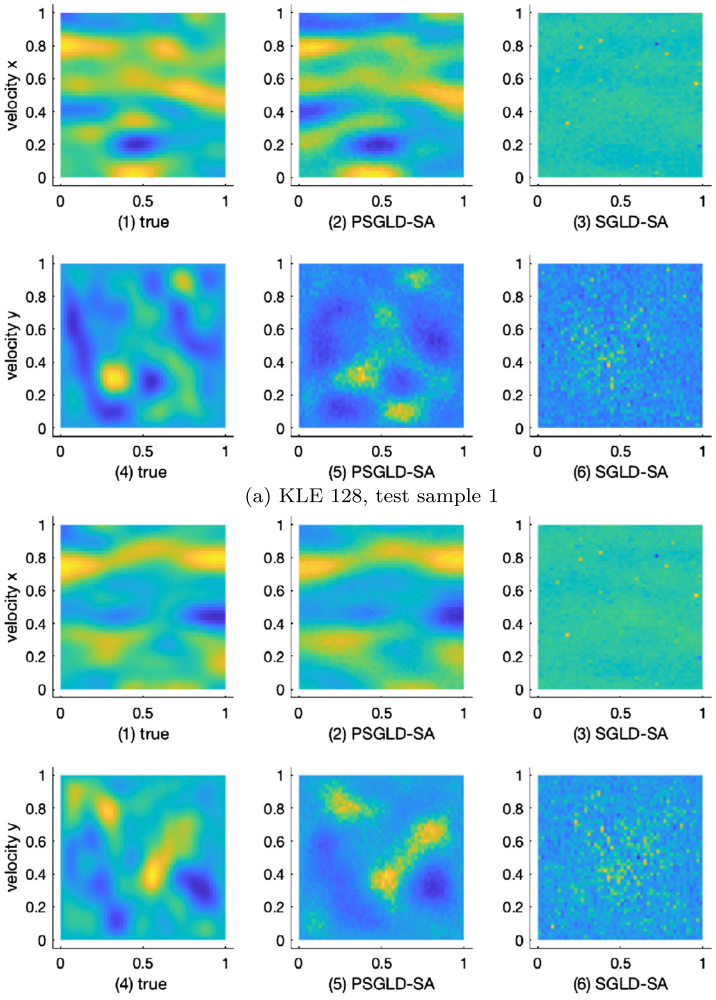
6.3. *Elliptic problem with channelized media*

Last, we employ the proposed algorithm to predict the solution of an elliptic problem with channelized media. The problem setup is the same as in section 6.2. However, the background permeability fields are images of channelized media. The image size for our problem is 50×50, which are patches cropped from the channelized media in [20]. An illustration of the permeability data and corresponding solutions are presented in Fig. 9. We generate 3000 data pairs and randomly split them into 80% and 20% for training and testing purposes, respectively.

We set the sparsity rate to be 30%, 50%, and 70%. The performance of our proposed method PSGLD-SA compared with vanilla SGLD, vanilla PSGLD, and SGLD-SA is shown in Table 2. We see that PSGLD-SA outperforms SGLD-SA in all three sparse cases. PSGLD-SA also results in more accurate results compared to vanilla PSGLD. The test learning curves are presented in Fig. 10. It shows that preconditioning helps to improve the convergence speed. With stochastic approximation, PSGLD-SA provides better results compared to PSGLD. Some samples are presented in Fig. 11. It is clear that the predicted velocity solutions using PSGLD-SA capture the heterogeneity in the underlying problem and look very close to the true solutions. However, the predicted solutions obtained from SGLD-SA are not reliable.

7. Conclusion

We propose a Bayesian sparse learning algorithm, where the model parameters are adaptively trained using a Bayesian mixture deep neural network, and the latent variables are smoothly learned through optimization. The Bayesian hierarchical model adopts SSGL priors, and samples are generated from the posterior using preconditioned Stochastic gradient descent Markov Chain Monte Carlo (PSGLD). PSGLD incorporates local curvature information in the parameter updating scheme,



(a) KLE 128, test sample 1

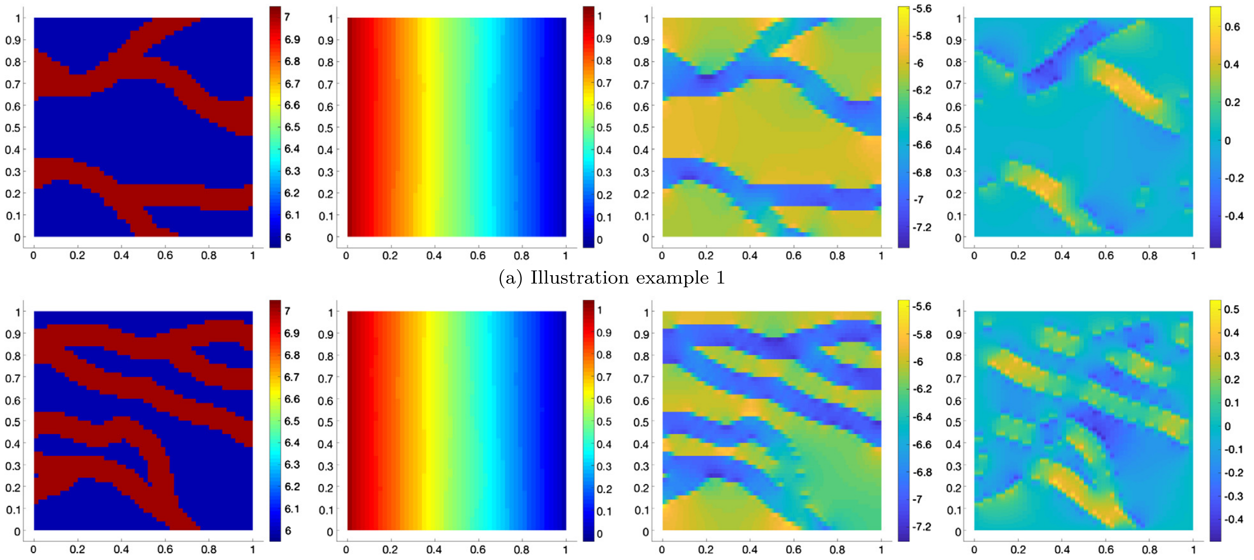
(b) KLE 128, test sample 2

Fig. 8. KLE 128. True and prediction solutions. In each sub-figure, first row represents horizontal velocity solution magnitude, second row represents vertical velocity solution magnitude. From left to right: true, PSGLD-SA prediction, SGLD-SA prediction.

Table 2
Channelized permeability fields. Mean errors between the true and predicted velocity using proposed SGLD, PSGLD, SGLD-SA, and PSGLD-SA. Mean errors among 600 testing samples.

	PSGLD (e_1/e_2 %)	SGLD (e_1/e_2 %)
Dense	3.31/2.73	6.39/4.94
	PSGLD-SA (e_1/e_2 %)	SGLD-SA (e_1/e_2 %)
Sparse rate 30%	2.73/2.13	3.94/3.12
Sparse rate 50%	2.75/2.16	3.91/3.05
Sparse rate 70%	2.71/1.95	4.46/3.57

such that a constant step size is adequate and slow mixing can be avoided. Due to the diagonal form of the preconditioning matrix, PSGLD needs less computational and storage cost compared to SGRLD. Moreover, we apply stochastic approximation techniques in the sequentially updated preconditioning matrix, the bias on the MSE introduced due to ignoring a correction term will approach zero. The convergence of the proposed algorithm is discussed. Numerical simulations are performed to learn the solutions of elliptic PDE with heterogeneous coefficients. Sparse learning with preconditioned SGLD sampling algorithm is helpful to accelerate the learning process and the trained sparse models, which can be used as computational efficient surrogates for solving the underlying PDE. The algorithm can also be extended to solve other heterogeneous problems and applied to the multi-fidelity framework. Moreover, we may construct an appropriate network structure and enforce sparsity according to physical information, such that we can interpret the sparse network obtained physically.



(a) Illustration example 1

(b) Illustration example 2

Fig. 9. Illustrations of channelized permeability fields and corresponding solutions. In each subplot, permeability (upper left), pressure solution (upper right), horizontal velocity magnitude (lower left) and vertical velocity magnitude (lower right).

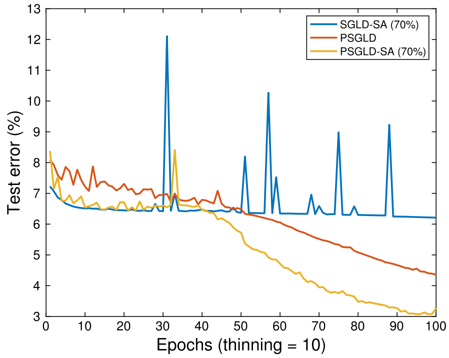


Fig. 10. Channelized media, learning curves. Comparison among test errors SGLD with sparse approximation, vanilla PSGLD, and PSGLD with sparse approximation.

CRediT authorship contribution statement

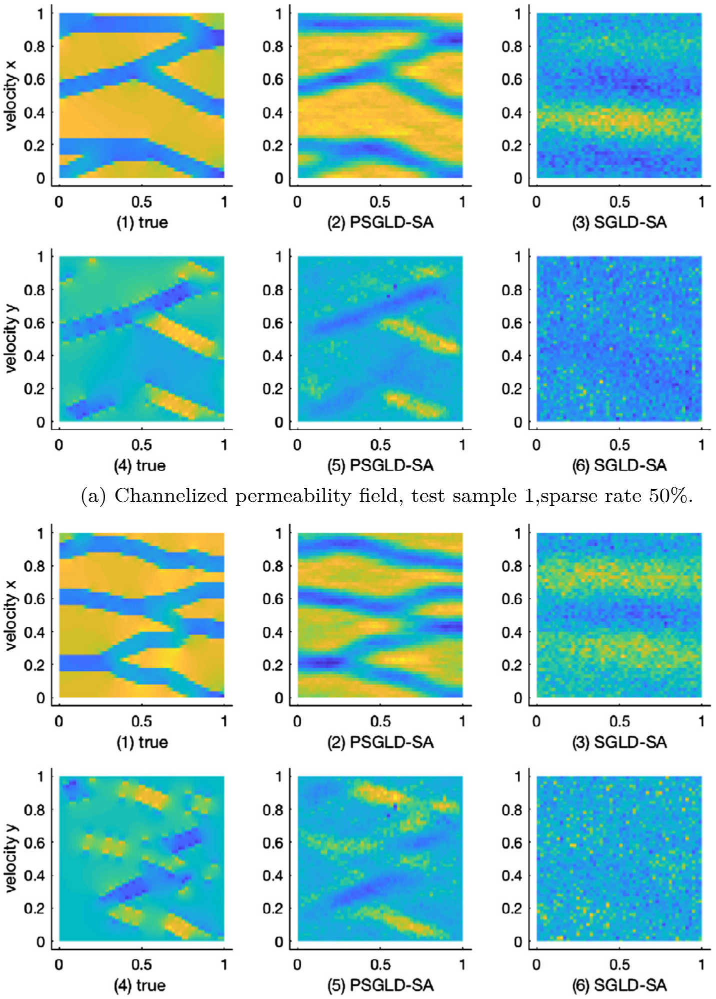
Yating Wang and Wei Deng conceived the machine learning models, implemented the methods, designed the numerical experiments, interpreted the results, and wrote the manuscript text. **Guang Lin** supported this study and edited the final manuscript. **All authors** provided final approval for publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We gratefully acknowledge the support from The National Science Foundation (DMS-1555072, DMS-1736364, CMMI-1634832, and CMMI-1560834), and Brookhaven National Laboratory Subcontract 382247, ARO/MURI grant W911NF-15-1-0562, and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0021142. The authors would also like to acknowledge the support from NVIDIA Corporation for the donation of the Titan Xp GPU used for this research.



(a) Channelized permeability field, test sample 1,sparse rate 50%.

(b) Channelized permeability field, test sample 2, sparse rate 70%.

Fig. 11. Channelized permeability field. True and prediction solutions. In each sub-figure, first row represents horizontal velocity solution magnitude, second row represents vertical velocity solution magnitude.

Appendix A. Convergence of latent variables

Assumption 1. The step size $\{\omega_k\}$ in the update formula for latent variables satisfies $\sum_{k=1}^{\infty}\omega_k = +\infty$, $\sum_{k=1}^{\infty}\omega_k^2 < +\infty$, moreover,

$$\liminf_{k\to\infty} 2\delta\frac{\omega_k}{\omega_{k+1}} + \frac{\omega_{k+1}-\omega_k}{\omega_{k+1}^2} > 0$$

In practice, one can choose $\omega_k = c_1(k+c_2)^{-\zeta}$ for $\zeta \in (0, 1])$ and constants c_1, c_2.

Assumption 2. For all $\boldsymbol{\theta} \in \Theta$, there exists a function $\mu_{\boldsymbol{\theta}}(\boldsymbol{\beta})$ that solves the Poisson equation $\mu_{\boldsymbol{\theta}}(\boldsymbol{\beta}) - \Pi_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{\beta}) = H(\boldsymbol{\theta}, \boldsymbol{\beta}) - h(\boldsymbol{\theta})$. There exists a constant C such that

$$\mathbb{E}||\Pi_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{\beta})|| \le C$$
$$\mathbb{E}||\Pi_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{\beta}) - \Pi_{\boldsymbol{\theta}'}\mu_{\boldsymbol{\theta}'}(\boldsymbol{\beta})|| \le C||\boldsymbol{\theta} - \boldsymbol{\theta}'||$$

Lemma 1. *There exists* λ_0 *and* k_0 *such that* $\forall\lambda \ge \lambda_0$ *and* $\forall k \ge k_0$, *the sequence* $\{\psi_k\}_{k=1}^{\infty}$ *with* $\psi_k = \lambda\omega_k + 2C_2/\delta \sup_{i\ge k_0} \triangle_i$ *satisfies*

$$\psi_{k+1} \ge (1 - 2\delta\omega_{k+1} + C_1\omega_{k+1}^2)\psi_k + C_1\omega_{k+1}^2 + 2C_2\triangle_k\omega_{k+1} \tag{13}$$

Proof. Plug in $\psi_k = \lambda\omega_k + 2C_2/\delta \sup_{i\geq k_0} \triangle_i$ in equation (13),

$$(\lambda\omega_{k+1} + 2C_2/\delta \sup_{i\geq k_0} \triangle_i) \geq (1 - 2\delta\omega_{k+1} + C_1\omega_{k+1}^2)(\lambda\omega_k + 2C_2/\delta \sup_{i\geq k_0} \triangle_i) + C_1\omega_{k+1}^2 + 2C_2\triangle_k\omega_{k+1}$$

Rearranging terms, we need to show

$$\lambda(\omega_{k+1} - \omega_k + 2\delta\omega_k\omega_{k+1} - C_1\omega_k\omega_{k+1}^2) \geq (-2\delta\omega_{k+1} + C_1\omega_{k+1}^2)2C_2/\delta \sup_{i\geq k_0} \triangle_i + C_1\omega_{k+1}^2 + 2C_2\triangle_k\omega_{k+1}$$

That is,

$$\lambda(2\delta\frac{\omega_k}{\omega_{k+1}} + \frac{\omega_{k+1} - \omega_k}{\omega_{k+1}^2} - C_1\omega_k)\omega_{k+1}^2 \geq \omega_{k+1}^2(C_1 + 2C_1C_2/\delta \sup_{i\geq k_0} \triangle_i) - (\sup_{i\geq k_0} \triangle_i - \triangle_k)2C_2\omega_{k+1} \tag{14}$$

Let $M_1 = \liminf_{k\to\infty} 2\frac{\omega_k}{\omega_{k+1}} + \frac{\omega_{k+1} - \omega_k}{\omega_{k+1}^2}$, we see the left hand side of (14) is greater than equal to $\lambda(M_1 - C_1\omega_k)\omega_{k+1}^2$. And use the fact that $\sup_{i\geq k_0} \triangle_i - \triangle_k \geq 0$, we have the right hand side of (14) is less than equal to $\omega_{k+1}^2(C_1 + 2C_1C_2/\delta \sup_{i\geq k_0} \triangle_i)$. Now it suffices to show that

$$\lambda(M_1 - C_1\omega_k)\omega_{k+1}^2 \geq \omega_{k+1}^2(C_1 + 2C_1C_2 \sup_{i\geq k_0} \triangle_i) \tag{15}$$

By choosing λ_0 and k_0 such that $\omega_{k_0} \leq \frac{M_1}{2C_1}$, and $\lambda_0 = \frac{4C_1C_2 \sup_{i\geq k_0} \triangle_i + 2C_1}{M_1}$, (14) holds, thus the desired inequality (13) holds. □

Theorem 2. *Suppose Assumptions 1-2 hold, with assumptions and Lemmas 1-2, Propositions 1-3 in [13], we have the sequence $\{\boldsymbol{\theta}_k\}_{k=1}^{\infty}$ converge to $\boldsymbol{\theta}_*$, and there exists a sufficiently large k_0 such that*

$$\mathbb{E}||\boldsymbol{\theta}_k - \boldsymbol{\theta}_*||^2 = \mathcal{O}(\lambda\omega_k + \sup_{i\geq k_0} \mathbb{E}||\Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})||)$$

Proof. Denote by $E_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_*$, we have

$$||E_{k+1}||^2 = ||E_k||^2 + \omega_{k+1}^2||\tilde{H}(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k)||^2 + 2\omega_{k+1}\mathbb{E}\langle E_k, \tilde{H}(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k)\rangle \tag{16}$$

For the third term in (16), we have

$$\begin{aligned}&\langle E_k, \tilde{H}(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k)\rangle = \langle E_k, H(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k) + \Delta(n, \boldsymbol{\theta}_i, \boldsymbol{\beta}_{i+1})\rangle \\ \leq& -||E_k||^2 + \langle E_k, \mu_{\boldsymbol{\theta}_k}(\boldsymbol{\beta}_{k+1}) - \Pi_{\boldsymbol{\theta}_k}\mu_{\boldsymbol{\theta}_k}(\boldsymbol{\beta}_k)\rangle + \langle E_k, \Pi_{\boldsymbol{\theta}_k}\mu_{\boldsymbol{\theta}_k}(\boldsymbol{\beta}_k) - \Pi_{\boldsymbol{\theta}_{k-1}}\mu_{\boldsymbol{\theta}_{k-1}}(\boldsymbol{\beta}_k)\rangle \\ &+ \langle E_k, \Pi_{\boldsymbol{\theta}_{k-1}}\mu_{\boldsymbol{\theta}_{k-1}}(\boldsymbol{\beta}_k) - \Pi_{\boldsymbol{\theta}_k}\mu_{\boldsymbol{\theta}_k}(\boldsymbol{\beta}_{k+1})\rangle + ||E_k||\triangle_k\end{aligned}$$

where $||\Delta(n, \boldsymbol{\theta}_k, \boldsymbol{\beta}_{k+1})|| = \triangle_k$. Following a similar proof as in [13], we have

$$2\omega_{k+1}\mathbb{E}\langle E_k, \tilde{H}(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k)\rangle C_2\omega_{k+1}$$

Thus,

$$\mathbb{E}||E_{k+1}||^2 \leq (1 - 2\delta\omega_{k+1} + C_1\omega_{k+1}^2)\mathbb{E}||E_k||^2 + C_1\omega_{k+1}^2 + 2C_2\triangle_k\omega_{k+1} + 2\omega_{k+1}\mathbb{E}[z_k - z_{k+1}]\rangle$$

where we use the fact that $||\tilde{H}(\boldsymbol{\beta_{k+1}}, \boldsymbol{\theta}_k)||^2 \leq C_1(1 + ||E_k||^2)$.

According to Lemma 1, there exists λ_0, k_0 such that

$$\mathbb{E}||E_{k_0}||^2 \leq \psi_{k_0} = \lambda_0\omega_{k_0} + 2C_2/\delta \sup_{i\geq k_0} \triangle_i$$

Thus,

$$\mathbb{E}||E_k||^2 \leq \psi_k + \mathbb{E}[\sum_{j=k_0+1}^{k} \Lambda_j^k(z_{j+1} - z_j)] \tag{17}$$

From Assumption 2 and that $\boldsymbol{\theta}$ is uniformly bounded, there exists $C_3 > 0$

$$\mathbb{E}[|z_k|] = \mathbb{E}\left[\left|\langle E_k, \Pi_{\boldsymbol{\theta}_{k-1}}\mu_{\boldsymbol{\theta}_{k-1}}(\boldsymbol{\beta}_k)\rangle\right|\right] \leq \mathbb{E}||E_k||\mathbb{E}\left[\left|\Pi_{\boldsymbol{\theta}_{k-1}}\mu_{\boldsymbol{\theta}_{k-1}}(\boldsymbol{\beta}_k)\right|\right] \leq C_3$$

Moreover, due to the fact that k_0 is an integer satisfying

$$\inf_{k\geq k_0}\frac{\omega_{k+1}-\omega_k}{\omega_k\omega_{k+1}}+2\delta-C_1\omega_{k+1}>0.$$

Then $\forall k\geq k_0$, the sequence $\{\Lambda_k^K\}_{k=k_0}^K$ is increasing, where

$$\Lambda_k^K=\begin{cases}2\omega_k\prod_{j=k}^{K-1}(1-2\omega_{k+1}\delta+C_1\omega_{k+1}^2), & \text{if } k<K\\ 2\omega_k, & \text{if } k=K\end{cases}$$

Thus,

$$\begin{aligned}\mathbb{E}\left[\left|\sum_{j=k_0+1}^{k}\Lambda_j^k(z_{j+1}-z_j)\right|\right]&=\mathbb{E}\left[\left|\sum_{j=k_0+1}^{k-1}(\Lambda_{j+1}^k-\Lambda_j^k)z_j+\Lambda_{k_0+1}^k z_{k_0}-\Lambda_k^k z_k\right|\right]\\&\leq(\Lambda_k^k-\Lambda_{k_0+1}^k)C_3+\Lambda_{k_0+1}^kC_3+\Lambda_k^kC_3\\&=2\Lambda_k^kC_3\leq 4C_3\omega_k\end{aligned}$$

Then the inequality (17) can be further bounded as

$$\begin{aligned}\mathbb{E}||E_k||^2&\leq\lambda_0\omega_k+C_2\sup_{i\geq k_0}\triangle_i+4C_3\omega_k\\&=\lambda\omega_k+C_2\sup_{i\geq k_0}\triangle_i\end{aligned}$$

where $\lambda=\lambda_0+4C_3$. □

References

[1] J. Aarnes, Y. Efendiev, Mixed multiscale finite element for stochastic porous media flows, SIAM J. Sci. Comput. 30 (5) (2008) 2319–2339.

[2] T. Arbogast, Homogenization-based mixed multiscale finite elements for problems with anisotropy, Multiscale Model. Simul. 9 (2011) 624–653.

[3] A. Bordes, L. Bottou, P. Gallinari, SGD-QN: careful quasi-Newton stochastic gradient descent, J. Mach. Learn. Res. 10 (2009) 1737–1754.

[4] R.H. Byrd, S.L. Hansen, J. Nocedal, Y. Singer, A stochastic quasi-Newton method for large-scale optimization, SIAM J. Optim. 26 (2016) 1008–1031.

[5] C. Chen, N. Ding, L. Carin, On the convergence of stochastic gradient MCMC algorithms with high-order integrators, in: Advances in Neural Information Processing Systems, 2015, pp. 2278–2286.

[6] R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, preprint, arXiv:1806.07366, 2018.

[7] T. Chen, E. Fox, C. Guestrin, Stochastic gradient Hamiltonian Monte Carlo, in: International Conference on Machine Learning, 2014, pp. 1683–1691.

[8] S.W. Cheung, E.T. Chung, Y. Efendiev, E. Gildin, Y. Wang, J. Zhang, Deep global model reduction learning in porous media flow simulation, Comput. Geosci. 24 (2020) 261–274.

[9] E. Chung, Y. Efendiev, C. Lee, Mixed generalized multiscale finite element methods and applications, Multiscale Model. Simul. 13 (2014) 338–366.

[10] E. Chung, Y. Efendiev, W.T. Leung, M. Vasilyeva, Y. Wang, Online adaptive local multiscale model reduction for heterogeneous problems in perforated domains, Appl. Anal. 96 (2017) 2002–2031.

[11] Y. Dauphin, H. De Vries, Y. Bengio, Equilibrated adaptive learning rates for non-convex optimization, in: Advances in Neural Information Processing Systems, 2015, pp. 1504–1512.

[12] Y.N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in: Advances in Neural Information Processing Systems, 2014, pp. 2933–2941.

[13] W. Deng, X. Zhang, F. Liang, G. Lin, An adaptive empirical Bayesian method for sparse deep learning, in: Advances in Neural Information Processing Systems, 2019, pp. 5564–5574.

[14] Y. Efendiev, J. Galvis, T. Hou, Generalized multiscale finite element methods (GMsFEM), J. Comput. Phys. 251 (2013) 116–135.

[15] Y. Fan, L. Lin, L. Ying, L. Zepeda-Núnez, A multiscale neural network based on hierarchical matrices, Multiscale Model. Simul. 17 (2019) 1189–1213.

[16] E.I. George, R.E. McCulloch, Approaches for Bayesian variable selection, Stat. Sin. (1997) 339–373.

[17] M. Girolami, B. Calderhead, Riemann manifold Langevin and Hamiltonian Monte Carlo methods, J. R. Stat. Soc., Ser. B, Stat. Methodol. 73 (2011) 123–214.

[18] J. He, J. Xu, MgNet: a unified framework of multigrid and convolutional neural network, Sci. China Math. 62 (2019) 1331–1354.

[19] Y. Khoo, J. Lu, L. Ying, Solving parametric PDE problems with artificial neural networks, arXiv:1707.03351, 2017.

[20] E. Laloy, R. Hérault, D. Jacques, N. Linde, Training-image based geostatistical inversion using a spatial generative adversarial neural network, Water Resour. Res. 54 (2018) 381–406.

[21] C. Li, C. Chen, D. Carlson, L. Carin, Preconditioned stochastic gradient Langevin dynamics for deep neural networks, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[22] J. Lin, Y. Rao, J. Lu, J. Zhou, Runtime neural pruning, in: Advances in Neural Information Processing Systems, 2017, pp. 2181–2191.

[23] Y.-A. Ma, T. Chen, E. Fox, A complete recipe for stochastic gradient MCMC, in: Advances in Neural Information Processing Systems, 2015, pp. 2917–2925.

[24] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, preprint, arXiv:1611.06440, 2016.

[25] S. Patterson, Y.W. Teh, Stochastic gradient Riemannian Langevin dynamics on the probability simplex, in: Advances in Neural Information Processing Systems, 2013, pp. 3102–3110.

[26] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, preprint, arXiv:1811.05537, 2018.

[27] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, preprint, arXiv:1711.10561, 2017.
[28] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations, preprint, arXiv:1711.10566, 2017.
[29] V. Ročková, E.I. George, EMVS: the EM approach to Bayesian variable selection, J. Am. Stat. Assoc. 109 (2014) 828–846.
[30] U. Simsekli, R. Badeau, T. Cemgil, G. Richard, Stochastic Quasi-Newton Langevin Monte Carlo, 2016.
[31] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude, COURSERA, Neural Netw. Mach. Learn. 4 (2012) 26–31.
[32] M. Wang, S.W. Cheung, E.T. Chung, Y. Efendiev, W.T. Leung, Y. Wang, Prediction of discretization of gmsfem using deep learning, preprint, arXiv:1810.12245, 2018.
[33] M. Wang, S.W. Cheung, W.T. Leung, E.T. Chung, Y. Efendiev, M. Wheeler, Reduced-order deep learning for flow dynamics: the interplay between deep learning and model reduction, J. Comput. Phys. 401 (2020) 108939.
[34] Y. Wang, S.W. Cheung, E.T. Chung, Y. Efendiev, M. Wang, Deep multiscale model learning, J. Comput. Phys. 406 (2020) 109071.
[35] Y. Wang, G. Lin, Efficient deep learning techniques for multiphase flow simulation in heterogeneous porous media, J. Comput. Phys. 401 (2020) 108968.
[36] E. Weinan, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[37] M. Welling, Y.W. Teh, Bayesian learning via stochastic gradient Langevin dynamics, in: Proceedings of the 28th International Conference on Machine Learning, ICML-11, 2011, pp. 681–688.
[38] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850.
[39] Y. Zhang, C.A. Sutton, Quasi-Newton methods for Markov chain Monte Carlo, in: Advances in Neural Information Processing Systems, 2011, pp. 2393–2401.
[40] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.