Stein Variational Gradient Descent with Matrix-Valued Kernels

Dilin Wang*, Ziyang Tang*, Chandrajit Bajaj, Qiang Liu (*Equal contribution)
Department of Computer Science · The University of Texas at Austin

Overview

Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016)

- Given a target distribution p, find $q := \{x_i\}_{i=1}^n$ to approximate p.
- Iteratively move q towards the target p by minimizing KL:

$$\min_{\phi\in\mathcal{F}} \mathrm{KL}(q_{[\epsilon\phi]} \,\|\, p) - \mathrm{KL}(q \,\|\, p), \quad \text{with,}$$

$$q_{[\epsilon\phi]} \text{ is the empirical distribution of } \{x_i' = x_i + \underbrace{\epsilon\phi(x_i)}_{\text{incremental updates}}\}_{i=1}^n.$$

- Closed-form updates by taking $\mathcal{F}$ to be the unit ball of a RKHS.

Key limitations:

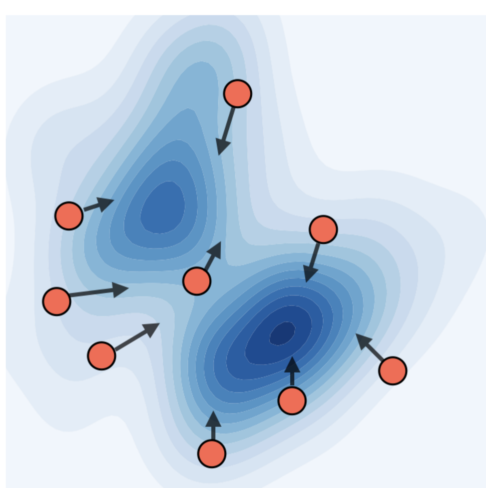
- Scalar-valued kernels; cannot encode potential correlations between different coordinates.

Our work:

- A generalization of SVGD with matrix-valued kernels.
- Improve SVGD by leveraging geometric information.

SVGD with scalar-valued kernels

- Finding the optimal transformation ϕ^* such that $\mathrm{KL}(q\|p)$ decreases as fast as possible:



$$\phi^* = \arg\max_{\phi\in\mathcal{F}} \left\{ -\frac{d}{d\epsilon}\mathrm{KL}(q_{[\epsilon\phi]}\|p)\Big|_{\epsilon=0} \right\},$$

$$= \arg\max_{\phi\in\mathcal{F}} \left\{ \mathbb{E}_{x\sim q}[\mathcal{P}^\top\phi(x)] \right\},$$

where $\underbrace{\mathcal{P}^\top\phi(x) := \nabla_x \log p(x)^\top\phi(x) + \nabla_x^\top\phi(x)}_{\text{Stein operator}}.$

- Taking $\mathcal{F}$ to be the unit ball of a RKHS $\mathcal{H}_k^d := \overbrace{\mathcal{H}_k \times \cdots \times \mathcal{H}_k}^{d}$ associated with kernel $k(x,x')$ (e.g. $k(x,x') = \exp(-\frac{(x-x')^2}{2h^2})$),

$$\phi_k^*(\cdot) \propto \mathbb{E}_{x\sim q}[k(\cdot,x)\nabla_x \log p(x) + \nabla_x k(\cdot,x)].$$

Why matrix-valued kernels

- Improve SVGD with preconditioning

$$x_i = x_i + \epsilon \overbrace{\underbrace{P}_{\text{preconditioning matrix}} \quad \phi_k^*(x_i)}^{\text{A unified approach?}}$$

- Matrix-valued kernels

$$\underbrace{K: \mathcal{X}\times\mathcal{X} \to \mathbb{R}^{d\times d}}_{\text{Matrix-valued}} \qquad \underbrace{k: \mathcal{X}\times\mathcal{X} \to \mathbb{R}^1}_{\text{scalar-valued}}.$$

Example: Separable kernels

· $K(x,x') = k(x,x')B$, where $B \in \mathbb{R}^{d\times d}$. It is easy to see that the corresponding optimal SVGD update is

$$\phi_K^*(\cdot) = B\phi_k^*(\cdot).$$

SVGD with Matrix-valued Kernels

- Let $\mathcal{F}$ be a unit ball of $\mathcal{H}_K$ with matrix-valued kernels, we have

$$\phi_K^*(\cdot) \propto \mathbb{E}_{x\sim q}[K(\cdot,x)\nabla_x \log p(x) + K(\cdot,x)\nabla_x],$$

$$\text{with,} \quad \underbrace{(K(\cdot,x)\nabla_x)_\ell}_{\ell-\text{th element}} = \sum_{m=1}^d \nabla_{x^m} K_{\ell,m}(\cdot,x).$$

Main Algorithm

- **Initialize** a set of particles $\{x_i^0\}_{i=1}^n$;
- **While** NOT converge; do

$$x_i \leftarrow x_i + \frac{\epsilon}{n}\sum_{j=1}^n \left[K(x_i,x_j)\nabla_{x_j}\log p(x_j) + K(x_i,x_j)\nabla_{x_j}\right].$$

Connections with other variants of SVGD

- Vanilla SVGD Liu and Wang, 16

$$K(x,x') = k(x,x')I.$$

- Graphical SVGD Wang et al., 18, Zhuo et al., 18

$$K(x,x') = \mathrm{diag}\left(\left\{k_\ell(x,x')\right\}_{\ell=1}^d\right).$$

- Gradient-free SVGD Han et al., 18:

$$K(x,x') = k(x,x')w(x)w(x')I, \quad w(x) = \frac{\rho(x)}{p(x)}.$$

Practical Choices of Matrix-Valued Kernels

Matrix-SVGD (average)

- Constant preconditioning matrix with RBF kernel:

$$K_Q(x,x') = Q^{-1}\exp\left(-\frac{1}{2h}\|x-x'\|_Q^2\right),$$

$$\text{with } \|x-x'\|_Q^2 = (x-x')^\top Q(x-x').$$

- $Q = \mathrm{Avg}[H(x_1),\cdots,H(x_n)]$, e.g. H: Hessian.

Matrix SVGD (mixture)

- Mixture preconditioning matrix with RBF kernel:

$$K(x,x') = \sum_{\ell=1}^m K_{Q_\ell}(x,x')w_\ell(x)w_\ell(x'),$$

where Q_ℓ as preconditioning of anchor point z_ℓ.

- We set the anchor points to be particles themselves.
- We take $w_\ell(x)$ as the Gaussian mixture probability:

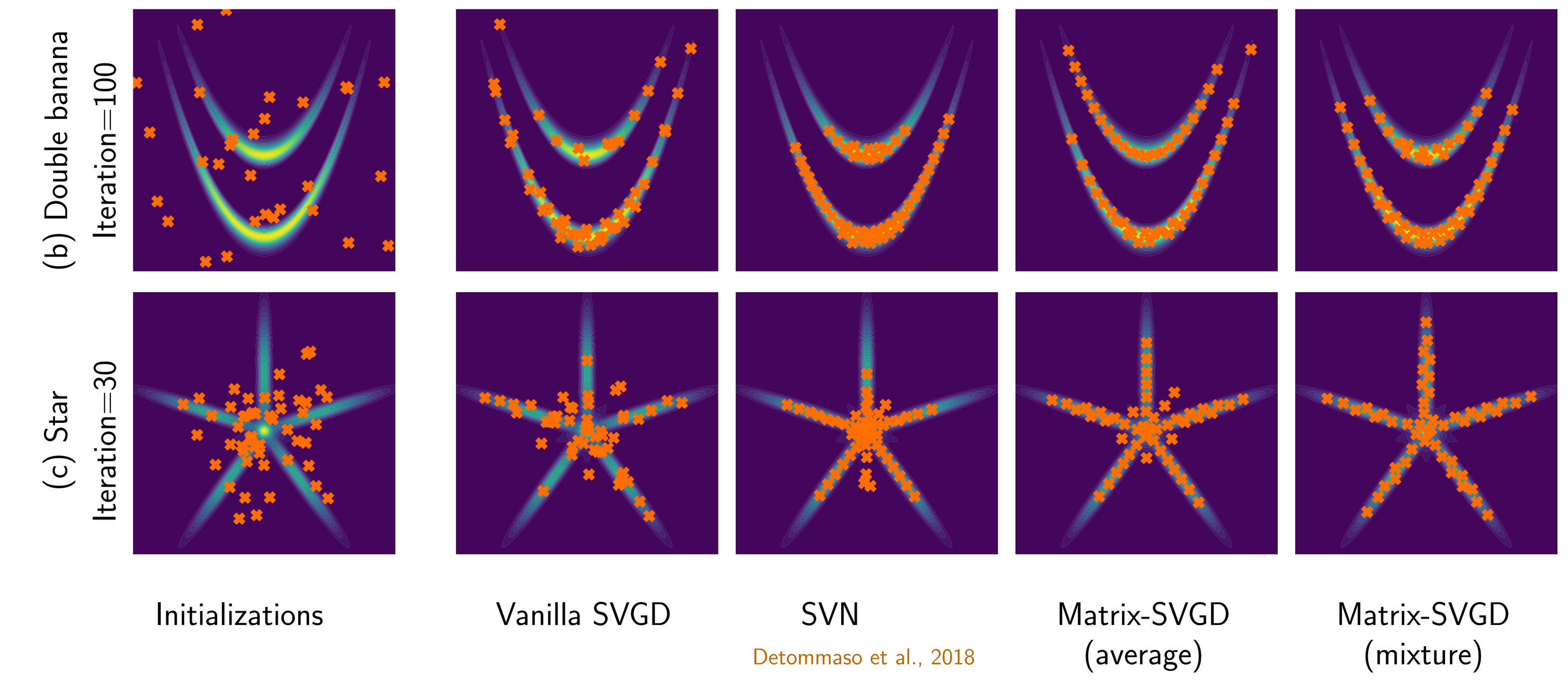
$$w_\ell(x) = \frac{\mathcal{N}(x; z_\ell, Q_\ell^{-1})}{\sum_{\ell'=1}^m \mathcal{N}(x; z_{\ell'}, Q_{\ell'}^{-1})},$$

$$\mathcal{N}(x; z_\ell, Q_\ell^{-1}) := \frac{1}{Z_\ell}\exp\left(-\frac{1}{2}\|x-z_\ell\|_Q^2\right).$$

Experiments

Two-Dimensional Toy Examples

- Search the best learning rate for all algorithms exhaustively.



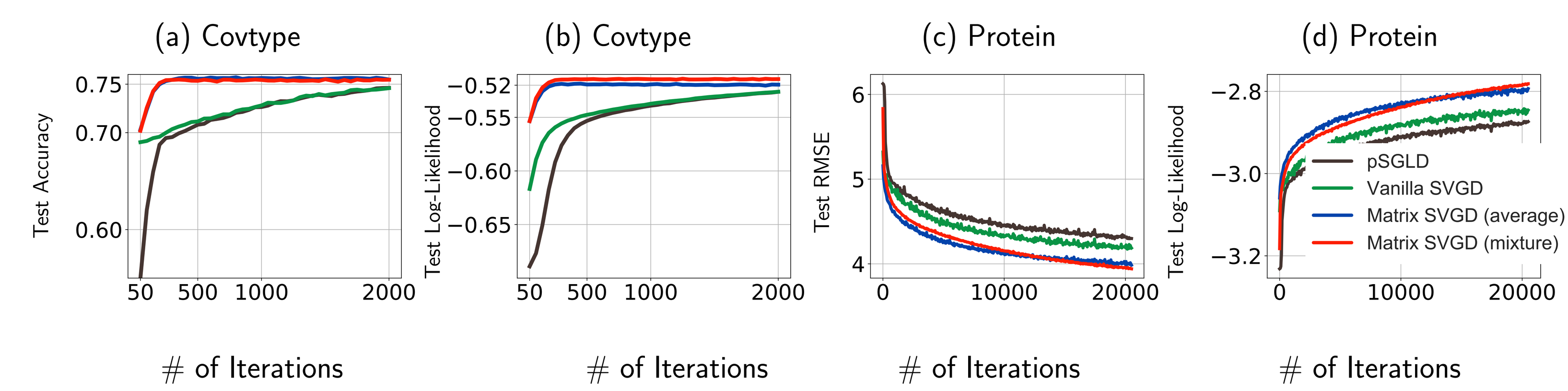
Bayesian Neural Network Regression

- Matrix-SVGD (mixture) generally performs the best among all algorithms.

Dataset	Test RMSE: Vanilla SVGD	Test RMSE: Matrix-SVGD (average)	Test RMSE: Matrix-SVGD (mixture)	Test Log-Likelihood: Vanilla SVGD	Test Log-Likelihood: Matrix-SVGD (average)	Test Log-Likelihood: Matrix-SVGD (mixture)
Boston	$2.785_{\pm 0.169}$	$2.898_{\pm 0.184}$	$\mathbf{2.717_{\pm 0.166}}$	$-2.706_{\pm 0.158}$	$\mathbf{-2.669_{\pm 0.141}}$	$-2.861_{\pm 0.207}$
Concrete	$5.027_{\pm 0.116}$	$4.869_{\pm 0.124}$	$\mathbf{4.721_{\pm 0.111}}$	$\mathbf{-3.064_{\pm 0.034}}$	$-3.150_{\pm 0.054}$	$-3.207_{\pm 0.071}$
Energy	$0.889_{\pm 0.024}$	$\mathbf{0.795_{\pm 0.025}}$	$0.868_{\pm 0.025}$	$-1.315_{\pm 0.020}$	$\mathbf{-1.135_{\pm 0.026}}$	$-1.249_{\pm 0.036}$
Kin8nm	$0.093_{\pm 0.001}$	$0.092_{\pm 0.001}$	$\mathbf{0.090_{\pm 0.001}}$	$0.964_{\pm 0.012}$	$0.956_{\pm 0.011}$	$\mathbf{0.975_{\pm 0.011}}$
Naval	$0.004_{\pm 0.000}$	$0.001_{\pm 0.000}$	$\mathbf{0.000_{\pm 0.000}}$	$4.312_{\pm 0.087}$	$5.383_{\pm 0.081}$	$\mathbf{5.639_{\pm 0.048}}$
Combined	$4.088_{\pm 0.033}$	$4.056_{\pm 0.033}$	$\mathbf{4.029_{\pm 0.033}}$	$-2.832_{\pm 0.009}$	$-2.824_{\pm 0.009}$	$\mathbf{-2.817_{\pm 0.009}}$
Wine	$0.645_{\pm 0.009}$	$\mathbf{0.637_{\pm 0.008}}$	$\mathbf{0.637_{\pm 0.009}}$	$-0.997_{\pm 0.019}$	$\mathbf{-0.980_{\pm 0.016}}$	$-0.988_{\pm 0.018}$
Protein	$4.186_{\pm 0.017}$	$3.997_{\pm 0.018}$	$\mathbf{3.852_{\pm 0.014}}$	$-2.846_{\pm 0.003}$	$-2.796_{\pm 0.004}$	$\mathbf{-2.755_{\pm 0.003}}$
Year	$8.686_{\pm 0.010}$	$8.637_{\pm 0.005}$	$\mathbf{8.594_{\pm 0.009}}$	$-3.577_{\pm 0.002}$	$-3.569_{\pm 0.001}$	$\mathbf{-3.561_{\pm 0.002}}$

SVGD Acceleration

- (a)-(b) Results of Bayesian Logistic regression on the Covtype dataset.
- (c)-(d) Average test RMSE and log-likelihood vs. training batches on the Protein dataset for Bayesian neural regression.



Classification With Recurrent Neural Networks

Method	MR (Pang & Lee, 05)	CR (Hu & Liu, 04)	SUBJ (Pang & Lee, 04)	MPQA (Wiebe et al., 05)
SGLD	20.52	18.65	7.66	11.24
pSGLD Li et al., 16	19.75	17.50	6.99	10.80
Vanilla SVGD	19.73	18.07	6.67	**10.58**
Matrix-SVGD (average)	19.22	17.29	6.76	10.79
Matrix-SVGD (mixture)	**19.09**	**17.13**	**6.59**	10.71

Table: Sentence classification erros measured with four benchmarks.

Acknowledgment This work is supported in part by NSF CRII 1830161 and NSF CAREER 1846421. We would like to acknowledge AWS and Google Cloud for their support.