Different Techniques for Stein Variational Gradient Descent

Min Hyung (Daniel) Kang

Department of Computer Science
Dartmouth College

May 22nd, 2017

1. Problem Setting
2. Background Information
3. Algorithms
4. Results
5. Future Direction

Problem Setting

Bayes Theorem

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(D)}$$

- $p(x|\mathcal{D})$: Posterior Distribution
- $p(\mathcal{D}|x)$: Likelihood
- $p(x)$: Prior
- $p(\mathcal{D})$: Normalization constant (Model Evidence)

Bayesian Inference

Bayesian Inference uses prior and likelihood to compute the posterior distribution according to Bayes' theorem.

Problem Setting

Bayesian Inference

Bayesian Inference uses prior and likelihood to compute the posterior distribution according to Bayes' theorem.

Monte carlo Markov Chain

- Generate samples by constructing a Markov chain whose equilibrium is the desired distribution.

Variational Inference

- Frames the problem into an optimization problem of approximating the distribution using a simpler distribution.

Problem Setting

Bayesian Inference

Bayesian Inference uses prior and likelihood to compute the posterior distribution according to Bayes' theorem.

Monte carlo Markov Chain

- Generate samples by constructing a Markov chain whose equilibrium is the desired distribution.
- Slow, problem with convergence, tough to scale up

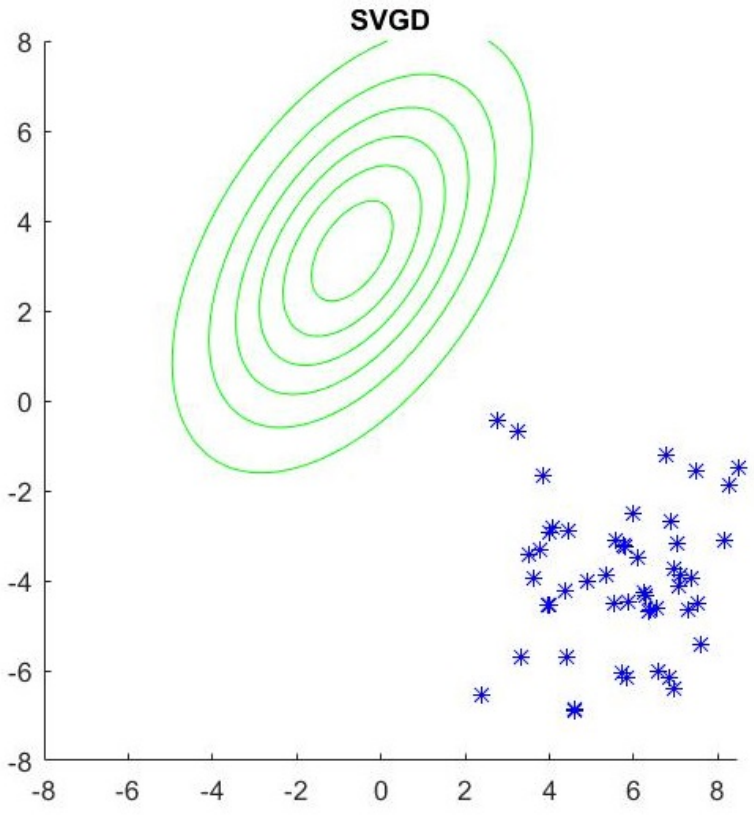
Variational Inference

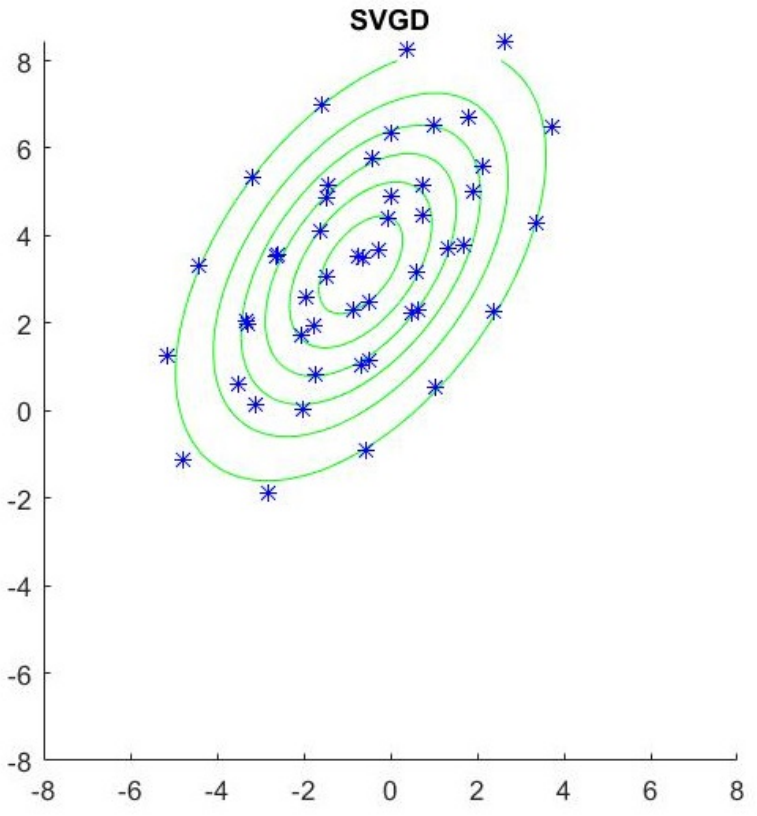
- Frames the problem into an optimization problem of approximating the distribution using a simpler distribution.
- Model-by-model derivation, careful choice of family of distributions

Problem Setting

Stein Variational Gradient Descent

We deterministically transport particles to reduce KL divergence to fit the true posterior distribution.





Problem Setting

Stein Variational Gradient Descent

We deterministically transport particles to reduce KL divergence to fit the true posterior distribution.

Tasks

- Can we make it faster?
- Can we make it better?

Reproducing Kernel Hilbert Space (RKHS)

Definition (Reproducing Kernel Hilbert Space)

Given positive definite kernel $k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, **Reproducing Kernel Hilbert Space (RKHS)** is closure of linear span

$$\{f : f(x) = \sum_{i=1}^{m} a_i k(x, x_i), a_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X}\}$$

with inner product

$$\langle f, g \rangle = \sum_{ij} a_i b_j k(x_i, x_j) \text{ for } g(x) = \sum_{i} b_i k(x, x_i)$$

Kernelized Stein Discrepancy (KSD)

Definition (Stein's Operator)

$$\mathcal{A}_p f(x) = s_p(x) f(x) + \nabla_x f(x) \tag{1}$$

Theorem (Stein's Identity)

For any smooth vector function f such that $\int_{x \in \mathcal{X}} \nabla_x \left(f(x) p(x) \right) dx = 0$,

$$\mathbb{E}_p \left[\mathcal{A}_p f(x) \right] = \mathbb{E}_p \left[s_p(x) f(x) + \nabla_x f(x) \right] = 0 \tag{2}$$

Definition (Kernelized Stein Discrepancy)

$$\mathbb{S}(p, q) = \max_{\phi \in \mathcal{H}^d} \left(\left[\mathbb{E}_{x \sim q} \left[trace(\mathcal{A}_p \phi(x)) \right] \right]^2, \text{ s.t. } ||\phi||_{\mathcal{H}^d} \leq 1 \right) \tag{3}$$

Stein Variational Gradient Descent (SVGD)

Idea

The general idea is that we minimize KL divergence between distribution after incremental transformation and target distribution.

KL divergence and Stein Operator

$$\nabla_\epsilon KL(q_{[T]}||p)|_{\epsilon=0} = -\mathbb{E}_{x\sim q}\left[tr(\mathcal{A}_p\phi(x))\right]; \quad T = x + \epsilon\phi(x) \tag{4}$$

Optimal perturbance and KSD

The optimal direction is then

$$\phi^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}\left[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)\right] \tag{5}$$

which gives the following negative gradient

$$\nabla_\epsilon KL(q_{[T]}||p)|_{\epsilon=0} = -\mathbb{S}(q,p) \tag{6}$$

Stein Variational Gradient Descent (SVGD)

Idea

The general idea is that we minimize KL divergence between distribution after incremental transformation and target distribution.

SVGD [5]

For each iteration t, we update the particles

$$x_i^{t+1} \leftarrow x_i^t + \epsilon^t \hat{\psi}^*(x_i^t)$$

$$\hat{\psi}^*(x) = \frac{1}{n} \sum_{j=1}^{n} \left[k(x_j^t, x) \nabla_{x_j^t} \log p(x_j^t) + \nabla_{x_j^t} k(x_j^t, x) \right]$$

ϵ^t is the step size at the t-th iteration.

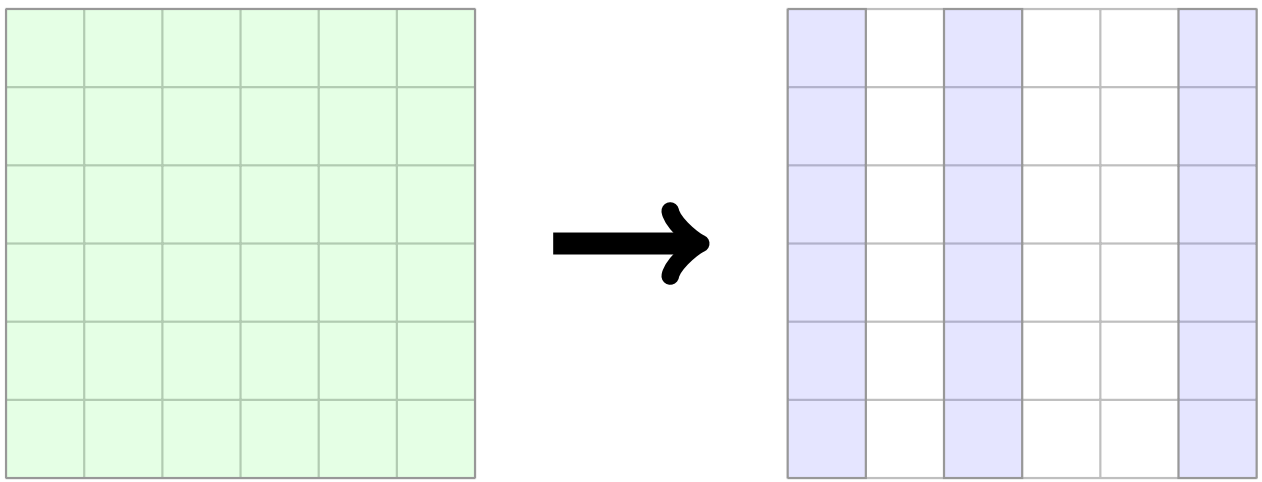
Demo

https://drive.google.com/open?id=0B58F7g3eKB1VX0I2enEtR2t1SHM

Random Subset

Intuition

Instead of computing the a value between every pair, we can take average of value with respect to a smaller subset.



Random Subset

Intuition

Instead of computing the a value between every pair, we can take average of value with respect to a smaller subset.

Algorithm 1 SVGD Random Subset (SVGD-RS)

Input: Target distribution with density function $p(x)$; Set of initial particles $\{x_i^0\}_{i=1}^n$; Positive definite kernel $k(x, x')$

Output: Set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution

for $t = 1 : maxIter$ **do**

Let $\{y_j^t\}_{j=1}^m$ be randomly selected m particles from $\{x_i^t\}_{i=1}^n$

$$\tilde{\psi}(x) = \frac{1}{m} \sum_{j=1}^{m} \left[k_h(y_j^t, x) \nabla_{y_j^t} \log p(y_j^t) + \nabla_{y_j^t} k_h(y_j^t, x) \right]$$

$x_i^{t+1} \leftarrow x_i^t + \epsilon^t \tilde{\psi}(x_i^t)$ where ϵ^t is the step size at the t-th iteration

end for

Random Subset + Control Functional

Intuition

We can assign weights to each of the subparticles.

Control Functional

We use the concept of control functional, a non-parametric extension of control variates.

Random Subset + Control Functional

Algorithm 2 SVGD Random Subset Control Functional (SVGD-RSCF)

Input: Target distribution with density function $p(x)$; Set of initial particles $\{x_i^0\}_{i=1}^n$; Positive definite kernel $k(x, x')$

Output: Set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution.

for $t = 1 : maxIter$ **do**

Let $\{y_j^t\}_{j=1}^m$ be randomly selected m particles from $\{x_i^t\}_{i=1}^n$.

Let $v^t = \frac{\mathbf{1}^T(\mathbf{K_0}+\lambda m\mathbf{I})^{-1}}{1+\mathbf{1}^T(\mathbf{K_0}+\lambda m\mathbf{I})^{-1}\mathbf{1}}$ where $(\mathbf{K_0})_{i,j} = k(y_i, y_j)$

Let $w_i^t = \frac{v_i^t}{\sum_j v_j^t}$

$$\tilde{\psi}(x) = \sum_{j=1}^{m} w_j^t \left[k_h(y_j^t, x)\nabla_{y_j^t} \log p(y_j^t) + \nabla_{y_j^t} k_h(y_j^t, x)\right]$$

$x_i^{t+1} \leftarrow x_i^t + \epsilon^t \tilde{\psi}(x_i^t)$ where ϵ^t is the step size at the t-th iteration.

end for

Induced Points - Idea

Induced Kernel

Define **induced kernel** as

$$k_y(x, x') = \frac{1}{m} \sum_{j=1}^{m} [k(x, y_j) k(x', y_j)]$$

where $\{y_i\}_{i=1}^m$ are **induced points**.

Induced Kernel Update

$$\phi(x') = \mathcal{A}_p k_y(x, x') = \frac{1}{m} \sum_{j=1}^{m} \left[\frac{1}{n} \sum_{i=1}^{n} [\mathcal{A}_p k(x_i, y_j)] \, k(x', y_j) \right]$$

Induced Points

Algorithm 3 Induced SVGD (I-SVGD)

Input: Target distribution with density function $p(x)$; Set of initial particles $\{x_i^0\}_{i=1}^n$; Positive definite kernel $k(x, x')$

Output: Set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution

for $t = 1 : maxIter$ **do**

Let $\{y_j^t\}_{j=1}^m$ be randomly selected m particles from $\{x_i^t\}_{i=1}^n$.

$$\tilde{\psi}(x) = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{n}\sum_{i=1}^{n}\left[k(x_i^t, y_j^t)\nabla_{x_i^t}\log p(x_i^t) + \nabla_{x_i^t}k(x_i^t, y_j^t)\right]k(x, y_j^t)\right]$$

$x_i^{t+1} \leftarrow x_i^t + \epsilon^t\tilde{\psi}(x_i^t)$ where ϵ^t is the step size at the t-th iteration

end for

Induced Points + Adversarial Updates

Adversarial Updates

From before :

$$\nabla_\epsilon KL(q_{[T]}||p)|_{\epsilon=0} = -\mathbb{S}(q,p)$$

Intuition : The amount of decrease in KL divergence is equal to the KSD. So what if we increase KSD inbetween regular particle updates?

Options we can change to increase KSD :

- Induced Points
- Parameters of the kernel (e.g. bandwidth of RBF kernel)

Induced Points + Adversarial Updates

Algorithm 4 Adversarial Induced SVGD (AI-SVGD)

Input: Target distribution with density function $p(x)$; Set of initial particles $\{x_i^0\}_{i=1}^n$; Positive definite kernel $k(x, x')$

Output: Set of particles $\{x_i^0\}_{i=1}^n$ that approximates the target distribution

Initialize $\{y_j^0\}_{j=1}^m$ as randomly selected m particles from $\{x_i^0\}_{i=1}^n$

for $t = 1 : maxIter$ **do**

$$\tilde{\psi}(x) = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{n}\sum_{i=1}^{n}\left[k(x_i^t, y_j^t)\nabla_{x_i^t}\log p(x_i^t) + \nabla_{x_i^t}k(x_i^t, y_j^t)\right]k(x, y_j^t)\right]$$

$$x_i^{t+1} \leftarrow x_i^t + \epsilon_x^t \tilde{\psi}(x_i^t)$$

for $\ell = 1 : \mathcal{L}$ **do**

$$y_j^{t,l+1} \leftarrow y_j^{t,l} + \epsilon_y^{t,l}\frac{1}{n^2}\sum_{x,x'}\nabla_{y_j^{t,1}}\mathcal{A}_p^x\mathcal{A}_p^{x'}k_y(x, x')$$

$$h^{t,l+1} \leftarrow h^{t,l} + \epsilon_h^{t,l}\frac{1}{n^2}\sum_{x,x'}\nabla_{h^{t,1}}\mathcal{A}_p^x\mathcal{A}_p^{x'}k_y(x, x')$$

end for

$$y^{t+1,1} \leftarrow y^{t,\mathcal{L}+1}, h^{t+1,1} \leftarrow h^{t,\mathcal{L}+1}$$

end for

where ϵ indicate step size for each variable at each iteration

Improvement Techniques

Regularization

$$\Delta y = \frac{1}{n^2} \sum_{x,x'} \nabla_y \mathcal{A}_p^x \mathcal{A}_p^{x'} k_y(x, x') - \alpha \frac{1}{m} \sum_{j=1}^{m} \nabla_y k(y, y_j) \tag{4}$$

U-Statistics

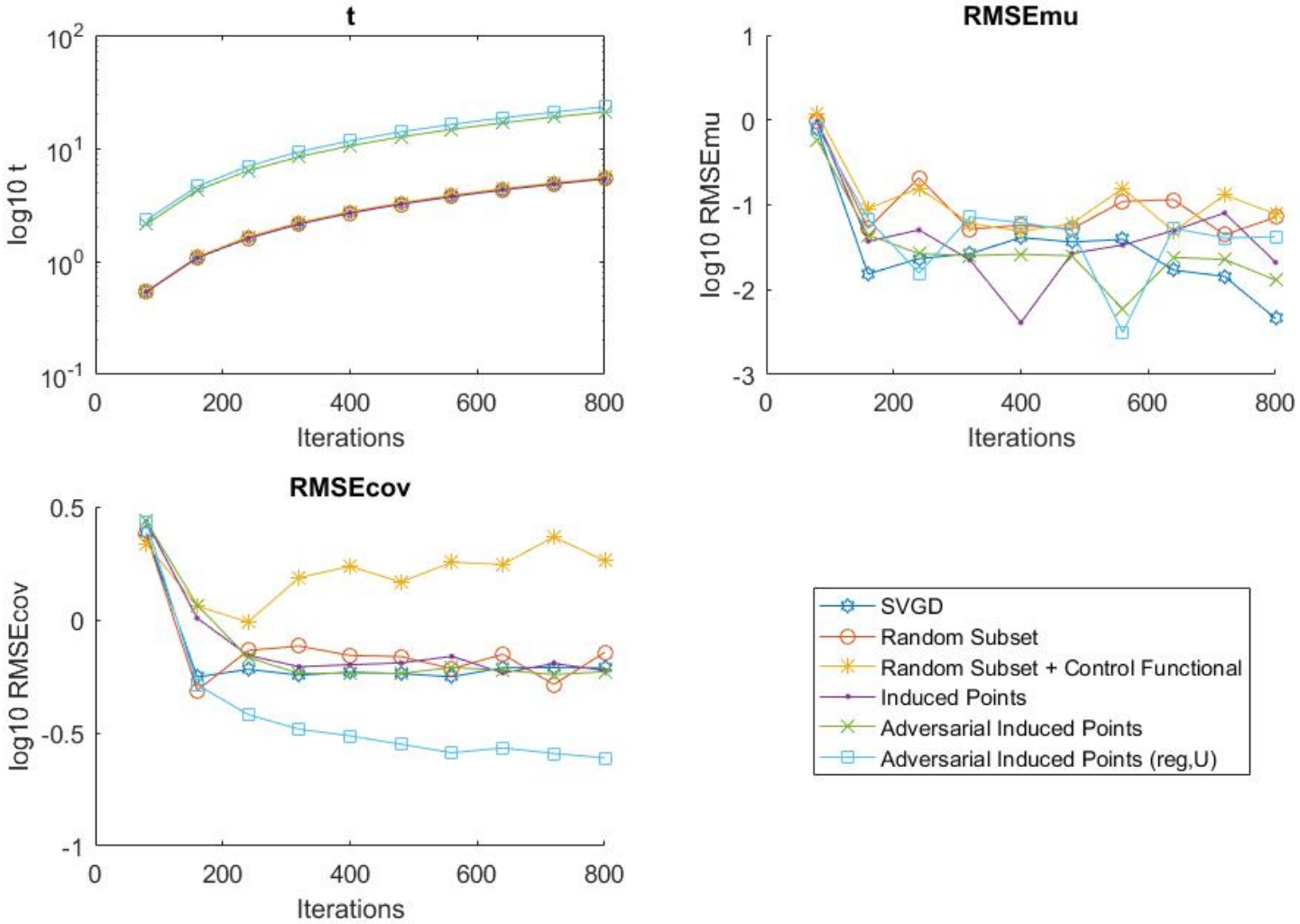
$$\Delta_y = \frac{1}{n(n-1)} \sum_{x \neq x'} \nabla_y \mathcal{A}_p^x \mathcal{A}_p^{x'} k_y(x, x') \tag{5}$$

Toy example : Gaussian

Demo

https://drive.google.com/open?id=0B58F7g3eKB1VeEI4VDhJOU5Tc2c

Toy example : Gaussian



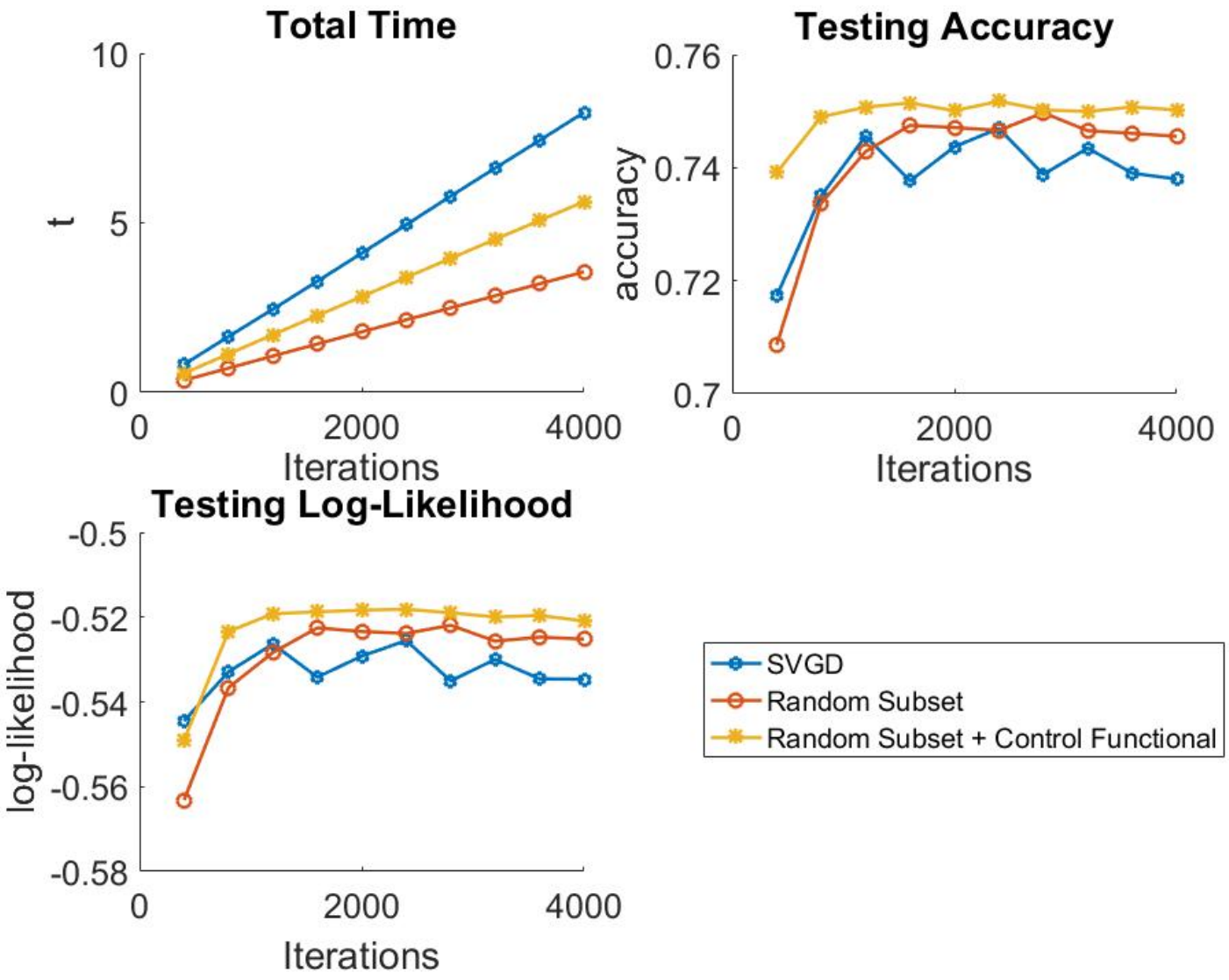
50 particles, 20 subparticles/induced points, 800 iterations.

Bayesian Logistic Regression

Model

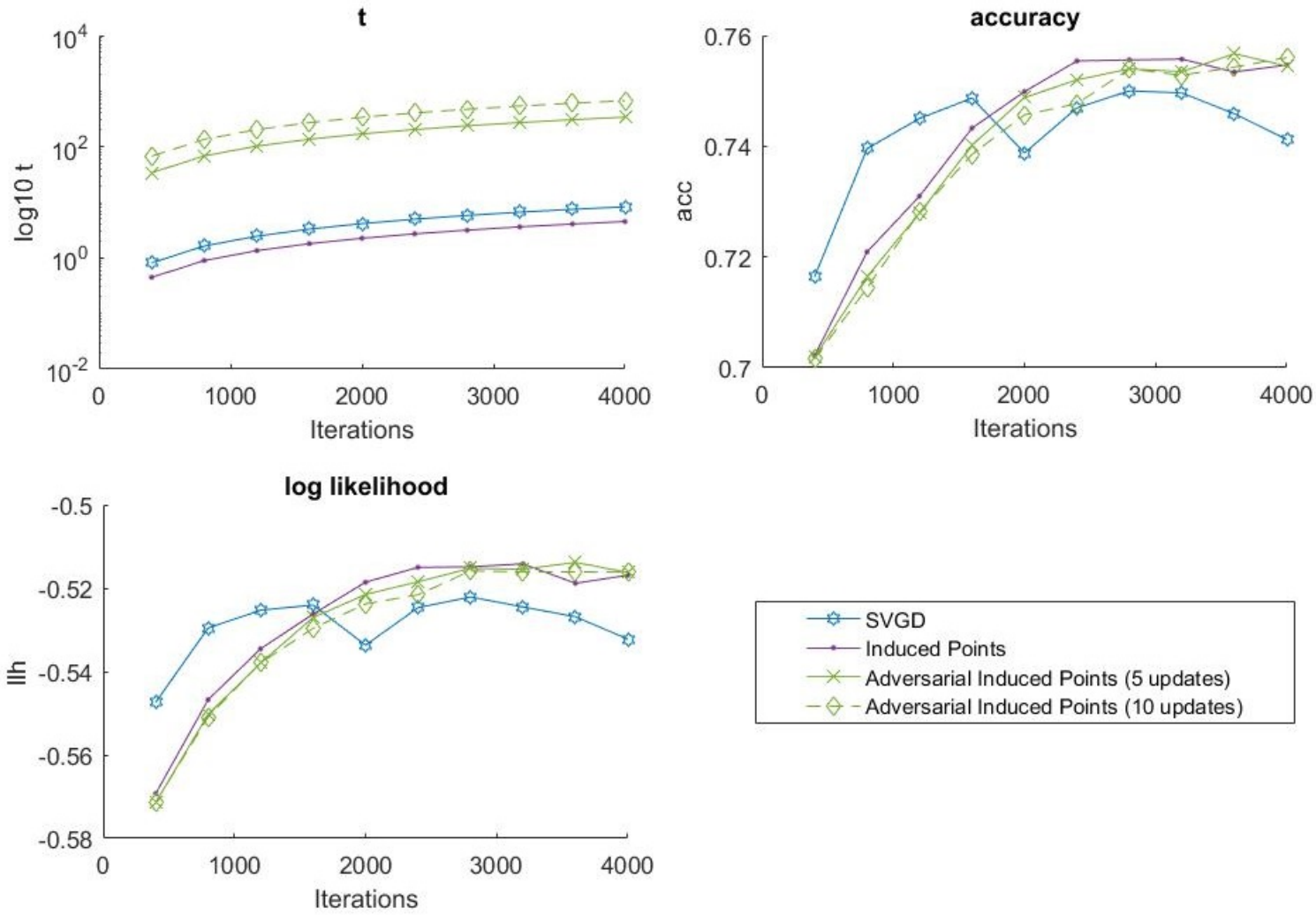
- $p(\alpha) = Gamma(\alpha; a_0, b_0)$
- $p(w_k|\alpha) = \mathcal{N}(w_k; 0, \alpha^{-1})$
- $p(c_t = 1|x_t, w) = \frac{1}{1+\exp(-w^T x_t)}$

Bayesian Logistic Regression



Varying number of particles/subparticles

Bayesian Logistic Regression



200 Particles, 40 Induced points, 10 trials

Bayesian Logistic Regression

Maximum Mean Discrepancy

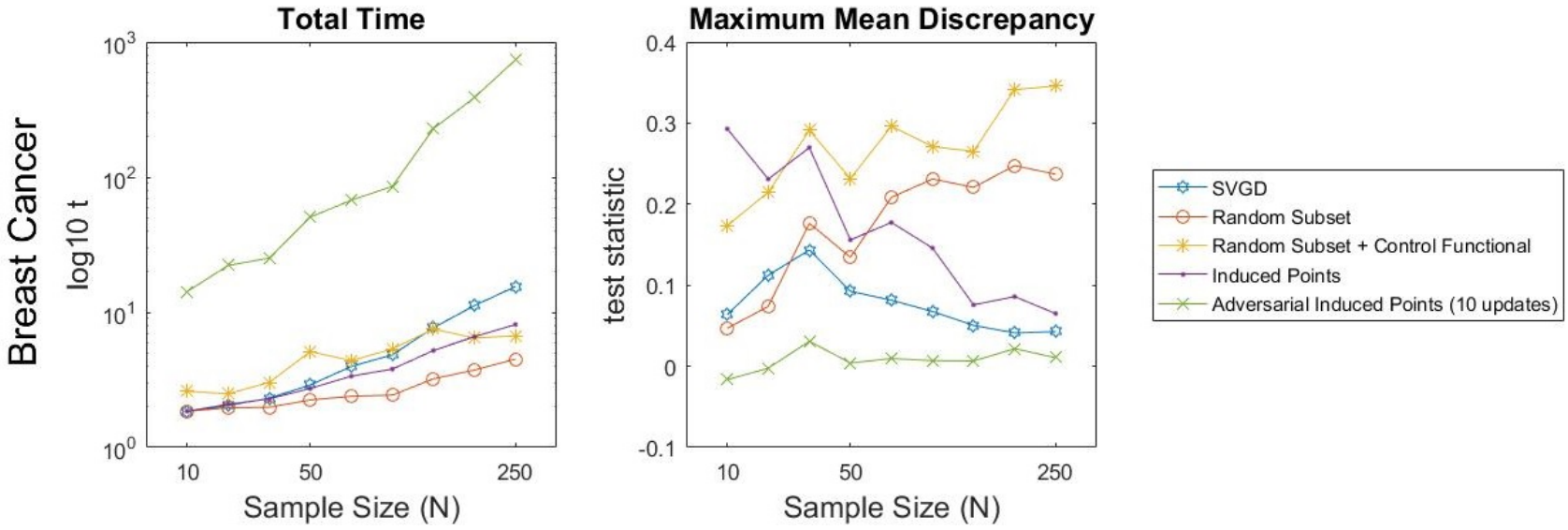
Maximum Mean Discrepancy (MMD) is a framework to compare distributions, testing whether two sets of samples are drawn from different distributions.

$$MMD\left[\mathcal{F}, p, q\right] = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim p}\left[f(x)\right] - \mathbb{E}_{y \sim q}\left[f(y)\right]\right) \tag{6}$$

or empirically,

$$MMD\left[\mathcal{F}, p, q\right] = \sup_{f \in \mathcal{F}} \left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i)\right) \tag{7}$$

Bayesian Logistic Regression



Varying number of particles/subparticles

Bayesian Neural Networks

Dataset	Avg. Test RMSE		
	SVGD	*RS*	*RS + CF*
Boston	2.2324 ± 0.0223	$\mathbf{2.2038 \pm 0.0204}$	2.2091 ± 0.0202
Concrete	6.0231 ± 0.0555	5.8865 ± 0.0481	$\mathbf{5.8637 \pm 0.0526}$
Energy	1.4454 ± 0.0429	1.4995 ± 0.0451	$\mathbf{1.4218 \pm 0.0498}$
Kin8nm	$\mathbf{0.1244 \pm 0.0010}$	0.1258 ± 0.0011	0.1255 ± 0.0011

Table 2: Comparison of test RMSE for different datasets

Bayesian Neural Networks

Dataset	Avg. Time (sec)		
	SVGD	*RS*	*RS + CF*
Boston	20.99 ± 0.69	$\mathbf{19.00 \pm 0.30}$	20.14 ± 0.61
Concrete	16.09 ± 0.79	$\mathbf{15.24 \pm 0.18}$	16.07 ± 0.13
Energy	17.21 ± 0.65	$\mathbf{15.58 \pm 0.34}$	16.53 ± 0.23
Kin8nm	17.52 ± 0.71	$\mathbf{15.76 \pm 0.28}$	16.72 ± 0.33

Table 4: Comparison of computation time for different datasets

Conclusion

- Random subset methods perform similar to original SVGD algorithm yet are much faster.
- Induced points methods provide novel approach to the Bayesian Inference problem that outperform original SVGD in its approximation of distributions.

Future Direction

- Optimal choice for size of subparticles and induced points.
- More theoretical understanding behind these algorithms.
- Instead of random subset, using a distribution to generate points.
- Ways to make the induced points algorithm perform faster.

References

[1] A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola. A kernel method for the two-sample problem. In *NIPS* , 2008
[2] J. M. Hernandez-Lobato and R. P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.
[3] W. Hoeffding. A class of statistics with asymptotically normal distribution. *Ann. Math. Statist.,* 19(3):293-325, 1948.
[4] M. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. volume 15, pages 1351-1381. 2014.
[5] Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *NIPS*, 2015.
[6] Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In *ICML*, 2016.
[7] C. Oates, M. Girolami, and N. Chopin. Control functionals for monte carlo integration. *Journal of the Royal Statistical Society: Series B*, 2017.