[Reviewer #1] Thank you for your valuable comments and suggestions.

On choosing a matrix kernel in practice: As the reviewer correctly noted, the problem of choosing an optimal kernel is a very hard problem. The difficulty lies on the fact that ideal optimal kernel should incorporate the intrinsic geometric information, but leads to no expensive computation.

In this work, we propose the mixture preconditining kernel as a practical heuristic to strike the balance between efficiency and computational cost. Compared with the constant preconditioning (which corresponds to using constant Hessian matrix), the mixture preconditioning allows us to incorporate different curvatures in different locations, which is of critical importance for complex models such as the "star" in Figure 1. Meanwhile, compared with the more theoretically motivated kernel derived with change of variables, the mixture preconditining kernel avoids taking high order derivatives and is much more computationally efficiently.

[Reviewer #2] Thank you for your valuable comments. We will improve our presentation based on your questions.

Comparisons with SVNM : The advantage of matrix SVGD over SVN is clear both theoretically and empirically. As we point out in L188-199, SVN can be in fact viewed as solving the same fixed point equation of SVGD with a scalar kernel and hence can not leverage the advantage of matrix kernels. For empirical comparison on the realistic examples, we did test the official Matlab code provide by the SVN authors, but the results we obtained were much worse than our methods and baselines (similar to the results in Fig 1), and hence did not investigate it further. For example, the RMSE score of SVN of BNN on the Boston dataset is $3.1830(\pm 0.1829)$ (averaged on 20 random trials), which is much worse than all the methods reported in Table 1 (< 2.9). We will investigate further and add discussion on this issue.

The unit ball is not unique: The RKHS $\mathcal{H}_k^d := \mathcal{H}_k \times \cdots \times \mathcal{H}_k$ must be equipped with inner product $\langle f, g\rangle_{\mathcal{H}_k^d} = \sum_{i=1}^d \langle f_i,\ g_i\rangle_{\mathcal{H}_k}$, which corresponds to using L_2 norm on $\mathbb{R}^d$. Using other norms on $\mathbb{R}^d$ can not define an RKHS.

Is R-SVGD a special case of our method: No, we will acknowledge out this.

[CBBGGMO2019] : Thanks for pointing it out. We were not aware this work by the time of submission. We will add a reference. But note that the preconditioned kernel in this paper is still a scalar kernel and hence substantially different from our matrix kernel.

Preconditioning matrices: We use Hessian matrix in Sec 4.1 and Fisher with K-FAC approximation in Sec 4.2-4.4.

How many particles: For the toy example, we use 50 particles for all the cases. For other applications, please see *line 235*, *line 245* and *line 264* for more details.

How many anchor points : Please see *line 212* (we use all the particles as the anchor points).

What kernel is used for computing MMD in Fig 1: We use the standard RBF kernel for evaluating MMD.

Different initialization?: We did use the same initialization for all methods with the same random seeds. Please see Fig 5, 6, 7; or code/bnn/trainer.py:L293, or code/rnn/acl_setence_classification.py: L384; in Figure 1(f), the plot is started from the 10-th iteration to avoid large values for better visualization.

Section 4.3 and 4.4: mini-batch size and #iteration : For Section 4.3, we use a mini-batch size of 100 for training; for large dataset such as year, we set the batch size to be 1000. In general, we grid search the least training epochs required for our implementation of SVGD to have a performance that matches the results reported in the original SVGD paper; we then train all other methods with the same training epochs. For Section 4.4, we use a mini batch-size of 50. we train all algorithms for 20 epochs;

the interchange of expectation and inner product should be justified: It is true once integrations involved are finite.

Reference: We will add a reference on the median trick and Lemma 2.

[Reviewer #3] Thank you for your valuable comments and suggestions.

L102: There looks to be a typo: Yes, $\mathcal{H}_k^d$ should be corrected to be $\mathcal{H}_{\mathbf{K}}$ here.

Experiments 1 & 2, is there any way to compute the algorithms on CPU time rather than iterations : As the other cases when Newton or natural gradient are used, the timing comparison heavily depends on implementation, approximation method of the matrix inversion, and many other factors such as matlab or python code and gpu-based or cpu-based implementation. Another important point, unique to our method, is that the SVGD with matrix kernels can yield *a set of better fixed points* than SVGD with scalar kernels, which may worth the effort even when the time-wise convergence is slower (in comparison, the standard Newton's method and natural gradient yield the same set of fixed point as standard gradient descent).