approxposterior: Approximate Posterior Distributions in Python

David P. Fleming[1] **and Jake VanderPlas**[1]

1 University of Washington

DOI: 10.21105/joss.00781

Software

- Review
- Repository
- Archive

Submitted: 24 April 2018
Published: 04 September 2018

Summary

This package is a Python implementation of "Bayesian Active Learning for Posterior Estimation" by (Kandasamy, Schneider, & Poczos, 2015) and "Adaptive Gaussian process approximation for Bayesian inference with expensive likelihood functions" (Wang & Li, 2017). These algorithms allows the user to compute approximate posterior probability distributions using computationally expensive forward models by training a Gaussian Process (GP) surrogate for the likelihood evaluation. The algorithms leverage the inherent uncertainty in the GP's predictions to identify high-likelihood regions in parameter space where the GP is uncertain. The algorithms then run the forward model at these points to compute their likelihood and re-trains the GP to maximize the GP's predictive ability while minimizing the number of forward model evaluations. Please read (Kandasamy et al., 2015) and (Wang & Li, 2017) for in-depth descriptions of the respective algorithms. *approxposterior* is under active development on GitHub and community participation is encouraged. The code is available on GitHub (Fleming, 2018).

The following figure is a simple demonstration of *approxposterior* produced using a Jupyter Notebook provided with the code on GitHub. In the left panel, we show the true posterior probability distribution computed by Markov Chain Monte Carlo (MCMC) compared against the result of *approxposterior*. The two distributions are in excellent agreement. In the right panel, we estimate how the performance of *approxposterior* compares against MCMC by tracking the number of forward model evaluations required for both methods to converge using the fiducial parameters given in (Wang & Li, 2017) and by tracking the computational time required for all parts of the *approxposterior* algorithm, such as training the GP. Since the (Wang & Li, 2017) forward model is analytic and hence requires little computational effort, we estimate computational times by adopting a range of forward model runtimes, from 10 microseconds to 10,000 seconds. For MCMC, the (Wang & Li, 2017) example requires 400,000 iterations to converge, so we can estimate how long MCMC would take to finish for a given forward model runtime by multiplying the number of iterations by the runtime since the forward model is evaluated each iteration. For *approxposterior*, we adopt a similar procedure, but also add the forward model runtimes for building the GP training set, the time required to train the GP, and the time required to derive the approximate posterior distribution.

In terms of *approxposterior* performance, we see two regimes: for very quick forward models, the GP training time dominates performance as the GP predictions take little time, about 30 microseconds. For slow forward models, the runtime is dominated by evaluating the forward model to generate the training set for the GP. In this regime, *approxposterior* is several orders of magnitude faster than MCMC.

Left: Joint posterior probability distribution of the two model parameters from the (Wang & Li, 2017) example. The black density map denotes the true distribution while the red contours denote the approximate distribution derived using *approxposterior*. The two distributions are in excellent agreement. Right: Total computational time required to

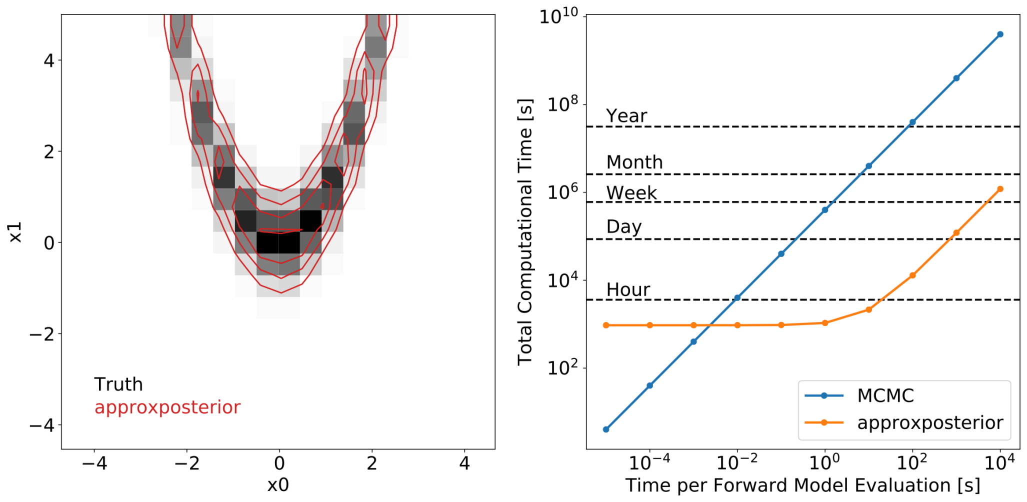

Figure 1: Left: Joint posterior probability distribution of the two model parameters from the Wang and Li (2017) example. The black density map denotes the true distribution while the red contours denote the approximate distribution derived using *approxposterior*. The two distributions are in excellent agreement. Right: Total computational time required to compute the posterior probability distribution of the model parameters from the Wang and Li (2017) example as a function of forward model evaluation time. The MCMC method (blue) runs the forward model for each MCMC iteration, while the orange curve was derived using the *approxposterior* BAPE implementation.

compute the posterior probability distribution of the model parameters from the (Wang & Li, 2017) example as a function of forward model evaluation time. The MCMC method (blue) runs the forward model for each MCMC iteration, while the orange curve was derived using the *approxposterior* BAPE implementation.

Acknowledgements

DPF was supported by NASA Headquarters under the NASA Earth and Space Science Fellowship Program - Grant 80NSSC17K0482. This work is supported in part by an NSF IGERT grant DGE-1258485.

References

Fleming, D. P. (2018). Approxposterior on github. Retrieved April 24, 2018, from https://github.com/dflemin3/approxposterior

Kandasamy, K., Schneider, J., & Poczos, B. (2015). Bayesian active learning for posterior estimation. In *International Joint Conference on Artificial Intelligence*.

Wang, H., & Li, J. (2017). Adaptive Gaussian process approximation for Bayesian inference with expensive likelihood functions. *ArXiv e-prints*. Retrieved from http://arxiv.org/abs/1703.09930