Sequential Neural Methods
for Likelihood-free Inference

Conor Durkan
University of Edinburgh
conor.durkan@ed.ac.uk

George Papamakarios
University of Edinburgh
g.papamakarios@ed.ac.uk

Iain Murray
University of Edinburgh
i.murray@ed.ac.uk

Abstract

Likelihood-free inference refers to inference when a likelihood function cannot be explicitly evaluated, which is often the case for models based on simulators. While much of the literature is concerned with sample-based 'Approximate Bayesian Computation' methods, recent work suggests that approaches relying on deep neural conditional density estimators can obtain state-of-the-art results with fewer simulations. The neural approaches vary in how they choose which simulations to run and what they learn: an approximate posterior or a surrogate likelihood. This work provides some direct controlled comparisons between these choices.

1   Introduction

In this paper, we consider Sequential Neural Posterior Estimation [15, 17, SNPE], Sequential Neural Likelihood [19, SNL], and Active Sequential Neural Likelihood [14, ASNL], three recent approaches to likelihood-free inference which leverage neural conditional density estimators to learn either a surrogate likelihood or approximate posterior directly. With each of the three models, we perform Bayesian inference over the parameters of the neural density estimators to prevent overfitting on small data sets, and in one case, investigate whether exploiting this uncertainty can lead to more efficient learning. Moreover, the modular nature of these methods allows us to perform direct comparison between them; though the implementation details of each have differed in the existing literature, here we use the same mixture density network architecture [1, MDN] for each method, and train each MDN using stochastic variational inference (SVI).

To our knowledge, this work provides the first head-to-head comparison of these sequential neural methods, and allows us to determine whether 'uncertainty as a guiding principle' leads to more efficient or accurate inference for this problem. We find that, for our experiments, learning a surrogate likelihood is more efficient than learning the approximate posterior directly, and that using an active learning component based on Bayesian model uncertainty performs as well as the heuristic used by SNL, but provides no meaningful improvement.

2   Problem & Methods

Formally, we are provided with a parametrized stochastic simulator $p(\mathbf{x} \mid \boldsymbol{\theta})$, which generates observations $\mathbf{x}$ given parameters $\boldsymbol{\theta}$. It is possible to sample from $p(\mathbf{x} \mid \boldsymbol{\theta})$, but not evaluate this likelihood explicitly. Given a single observed data point $\mathbf{x}_0$, we seek plausible settings of the simulator parameters $\boldsymbol{\theta}$ which generated that data. That is, we are interested in inferring the posterior distribution over parameters $p(\boldsymbol{\theta} \mid \mathbf{x}_0)$. We are free to choose a suitable prior $p(\boldsymbol{\theta})$.

Each of the models under consideration proceeds in a sequential manner, alternating between the following two steps:

1. **Generating data from the simulator**. A priori, and unlike standard machine learning tasks, we are provided with just a single observation $\mathbf{x}_0$. We must choose parameter settings $\boldsymbol{\theta}$, and then rely on the simulator to generate pairs $\left\{ \left(\boldsymbol{\theta}^{(n)}, \mathbf{x}^{(n)} \right) \right\}_{n=1}^{N}$ as training data.

2. **Improving the posterior approximation**. Once training data has been gathered, we can use it to improve our posterior approximation. The choice of method will determine whether we model the forward relationship $\boldsymbol{\theta} \rightarrow \mathbf{x}$ (learn a surrogate likelihood, or emulator), or the inverse $\mathbf{x} \rightarrow \boldsymbol{\theta}$ (learn a recognition network that gives the posterior directly).

Sequential Neural Posterior Estimation (SNPE). These methods fit a parametrized approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{x}_0)$ to the true posterior $p(\boldsymbol{\theta} \mid \mathbf{x}_0)$. In each round, the simulator is run using parameters sampled from the current approximate posterior [17, SNPE-A]. Here we use the particular method by Lueckmann et al. [15], which we denote SNPE-B.

Sequential Neural Likelihood (SNL). In contrast to SNPE, SNL learns a parametrized surrogate likelihood $q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta}) \approx p(\mathbf{x}_0 \mid \boldsymbol{\theta})$. SNL also samples the parameters to simulate next from the current approximate posterior, which requires an additional inference step, such as Markov chain Monte Carlo using the surrogate likelihood. SNL also allows for easier use of other neural density estimators, so Papamakarios et al. [19] use a state-of-the-art method MAF [18]. Here we test whether SNL would still outperform SNPE-B if using the same neural architecture (in this case an MDN).

Active Sequential Neural Likelihood (ASNL). Concurrently with SNL, Lueckmann et al. [14] proposed likelihood-free inference with emulator networks, a method which corresponds to SNL with an active learning component. We refer to this as active SNL, or ASNL. Rather than sampling from the approximate posterior each round to generate parameters $\boldsymbol{\theta}$, an acquisition function guides the choice of parameter at which to simulate next. The acquisition function exploits uncertainty in the approximate posterior, meaning we require a method to quantify uncertainty for the neural density estimator used as a surrogate likelihood. Lueckmann et al. [14] use the *MaxVar* rule of Järvenpää et al. [10], given by $\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})p(\boldsymbol{\theta})]$.

3 Experiments

We compare the three methods across three inference tasks (see appendix A for general details, appendix B for ASNL specific discussion, and appendix C for additional results).
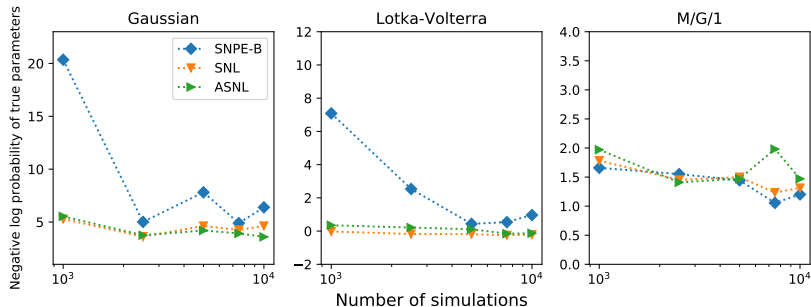

Figure 1: Comparison of true parameter log probability under the learned posteriors. Lower is better.

Table 1: Wall-clock time (in hours) per experiment with 10^4 simulations.

	Gaussian	Lotka–Volterra	M/G/1
SNL	**7.48**	**8.27**	**7.83**
ASNL	91.73	89.39	70.16

4 Discussion

Learning the likelihood rather than the posterior is a design choice. Both SNL and ASNL achieve a higher log probability for the true parameters than SNPE-B on both the toy problem and Lotka–Volterra, while all three methods exhibit similar performance on M/G/1 (see fig. 1). This suggests learning the likelihood seems preferable to learning the posterior, all other model choices equal. However, the Gaussian task was chosen such that the likelihood was tractable, but the posterior was complex and multimodal. As such, it could be argued that the neural density estimator for SNPE-B in this task should have been made more powerful to compensate.

Additionally, in both the Lotka–Volterra and M/G/1 models, we model summary statistics of the data, and not the observations themselves. Since summary statistics are generally much lower-dimensional than their associated observations, the burden on the neural density estimators modelling the likelihood is lessened. In contrast, direct posterior modelling with SNPE might converge faster than SNL when observations are relatively high-dimensional, and we need only learn a distribution over low-dimensional simulator parameters.

On the other hand, the advance of general-purpose neural density estimators [5, 18], in addition to task-specific models for images [22] and audio [21], may mean that modelling the high-dimensional observations directly is more achievable than it once was. We might expect the final layers of such deep models to represent something akin to the summary statistics manually chosen for the data here.

Quantifying uncertainty in neural networks is a difficult problem. The success of any active learning component in these sequential neural methods is likely heavily dependent on the choice of uncertainty quantification. In this scenario, exploiting model uncertainty with active learning requires the ability to accurately and robustly quantify uncertainty over neural density estimators. Generally, this is a difficult problem, not just for neural density estimators, but also for the wider field of deep learning. Tools for uncertainty quantification in neural networks range from approximate Bayesian inference approaches like SVI [3, 9, 12], expectation propagation [8], and Hamiltonian Monte Carlo [16, HMC], to ensemble methods [13]. Recent work [2, 4, 7] has highlighted the concepts of *aleatoric* (referring to stochasticity inherent to our data) and *epistemic* uncertainty (traditional model uncertainty), discussing methods to disentangle these two confounding factors in deep learning. Despite this progress, robust, general-purpose uncertainty estimation for neural networks is still an open challenge for machine learning researchers.

Practical considerations influence our machine learning methods. The active exploration of ASNL, though in theory more principled, does not seem to perform better than the sampling-based heuristic of SNL in practice. While being maximally efficient with simulations is preferable in principle, practical concerns may mean that wall-clock time is a more desirable metric. In this particular likelihood-free inference scenario, it could be argued that compute time taken by the ASNL acquisition function might be better allocated to additional simulation time for SNL, or a more complex neural density estimator (see table 1).

In our experiments, the neural density estimators are small enough that a gold standard like HMC might be used to perform inference. However, we found HMC difficult to integrate with the models under consideration, being sensitive to initialization, and requiring manual tuning per round of training. We are also unaware of any work in which HMC is used to perform approximate inference over a neural density estimator, and it may be the case that HMC requires specific tuning for this scenario. In contrast, SVI performed well as a plug-in method, though the approximate posterior distribution it can represent is limited. Once more, there is a trade-off between less expressive but practical methods, and more complex, flexible methods which may be difficult for practitioners to tailor for their particular problems.

Conclusions. Broadly, we find:

- Even when restricting SNL to use the same SVI-MDN architecture as SNPE-B, SNL learns faster on the given tasks, and unlike SNPE-B, SNL allows for easy integration of other advanced neural density architectures. Future comparisons might focus on higher-dimensional data, which may lead to different conclusions.

- The heuristic exploration of SNL seems just as effective as the more principled exploration of ASNL. While these neural networks are useful for Bayesian inference tasks, whether principled uncertainty estimates are useful for guiding exploration remains to be seen.

References

[1] Bishop, C. M. (1994). Mixture density networks.

[2] Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1192–1201.

[3] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

[4] Gal, Y. and Smith, L. (2018). Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with Bayesian neural networks. *arXiv preprint arXiv:1806.00667*.

[5] Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

[6] Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361.

[7] Hafner, D., Tran, D., Irpan, A., Lillicrap, T., and Davidson, J. (2018). Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv preprint arXiv:1807.09289*.

[8] Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

[9] Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM.

[10] Järvenpää, M., Gutmann, M. U., Pleska, A., Vehtari, A., and Marttinen, P. (2018). Efficient acquisition rules for model-based approximate Bayesian computation. *Bayesian Analysis*.

[11] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[12] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

[13] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

[14] Lueckmann, J.-M., Bassetto, G., Karaletsos, T., and Macke, J. H. (2018). Likelihood-free inference with emulator networks. *arXiv preprint arXiv:1805.09294*.

[15] Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, pages 1289–1299.

[16] Neal, R. M. (2012). *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media.

[17] Papamakarios, G. and Murray, I. (2016). Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. *Advances in Neural Information Processing Systems 29*.

[18] Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems 30*.

[19] Papamakarios, G., Sterratt, D. C., and Murray, I. (2018). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *arXiv preprint arXiv:1805.07226*.

[20] Shestopaloff, A. Y. and Neal, R. M. (2014). On Bayesian inference for the M/G/1 queue with efficient MCMC sampling. *arXiv preprint arXiv:1401.5548*.

[21] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. In *SSW*, page 125.

[22] Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756.

[23] Wilkinson, D. J. (2006). *Stochastic Modelling for Systems Biology*. Chapman and Hall/CRC.

[24] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

A   Experimental details

A.1   Bayesian MDN

The neural density estimator and its training procedure are fixed across methods for fair comparison. In all experiments, we use an MDN with a single hidden layer of 50 hyperbolic tangent units. The mixture distribution consists of 5 full-covariance Gaussian densities. We learn a diagonal Gaussian distribution over the weights and biases of the MDN using SVI and the local reparameterization trick [12], and use a standard normal prior. The density estimators are trained across 10 rounds using stochastic gradient descent with the Adam optimizer [11], with 1000 epochs per round.

A.2   Posterior inference

Both SNL and ASNL require an additional inference step to generate samples from the surrogate likelihood and prior. Following Papamakarios et al. [19], we used slice sampling, which performed robustly on each of our experiments.

In the interest of fair comparison, we used a kernel density estimate to approximate the log probability of the true parameters under the learned posterior, even though exact density evaluation of the posterior is available with SNPE-B. For each trained model, the kernel density estimate was based on 5000 samples from the posterior.

A.3   Inference tasks

Gaussian toy problem. The Gaussian problem outlined by Papamakarios et al. [19] describes a toy simulator with a tractable (indeed, Gaussian) likelihood. Despite this simplification, the posterior is complex and multimodal. Such a model can be used to compare the advantages of learning the likelihood versus learning the approximate posterior directly.

Lotka–Volterra. The Lotka–Volterra model [23] describes the evolution of a predator-prey population over time. Four parameters θ govern the birth and death rates of each population. The model is simulated using the Gillespie algorithm [6], and observations $\mathbf{x}$ are 9-dimensional summary statistics of the generated population time series. We follow the experimental setup of Papamakarios and Murray [17].

M/G/1 Queue. The M/G/1 queue model [20] describes how a server processes a waiting queue of customers. Its three parameters θ govern how long the server takes to respond to a customer and the

frequency of customer arrivals. Observations $\mathbf{x}$ are 5 equally-spaced quantiles of the distribution of inter-departure times. Again, we follow the experimental setup of Papamakarios and Murray [17].

B ASNL Details

B.1 Modifications

Lueckmann et al. [14] use an ensemble method [13] to quantify uncertainty over the parameters of their density estimator, whereas we use SVI to maintain fair comparison to other methods. Their method also describes gathering a single data point at a time using the acquisition function, which we found not to provide a sufficient training signal for the neural density estimator. Instead, we allowed ASNL to generate as much data per round as given to SNPE-B and SNL, again for fair comparison.

B.2 MaxVar implementation

The *MaxVar* acquisition function determines parameters $\boldsymbol{\theta}^{\star}$ at which to simulate by

$$\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})p(\boldsymbol{\theta})], \tag{1}$$

where the variance is taken with respect to the posterior distribution over neural density estimator parameters $\boldsymbol{\phi}$. This acquisition function focuses on areas of the parameter space where our estimate of the posterior varies the most. In each of our experiments, the prior on simulator parameters is box uniform (i.e. constant), so that the criterion can be equivalently written

$$\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}} \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})]. \tag{2}$$

We can approximate the function $f(\boldsymbol{\theta}) = \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})]$ by sampling an ensemble of neural density estimators from the variational posterior $q(\boldsymbol{\phi} \mid \mathcal{D})$, where $\mathcal{D} = \left\{ \left(\boldsymbol{\theta}^{(n)}, \mathbf{x}^{(n)}\right) \right\}_{n=1}^{N}$, and computing the sample variance of their density estimates for a particular value $\boldsymbol{\theta}$. However, neural density estimators generally yield log-densities, and in practice we found that the magnitude of these log-quantities was such that simply taking the exponent led to numerical issues. Fortunately, it is possible to sidestep this problem.

Computing probabilistic expressions in a numerically stable manner using log-quantities often involves the *log-sum-exp* trick. Given log-quantities $\{a_k\}_{k=1}^{K}$, the log-sum-exp trick rewrites the naive computation as

$$\log \sum_{k=1}^{K} \exp a_k = \max_{k} a_k + \log \sum_{k=1}^{K} \exp\left[a_k - \max_{k'} a_{k'}\right]. \tag{3}$$

On the right hand side of this equation, the largest value in the summand is now 1, when $a_k = \max_{k'} a_{k'}$. Any terms which now underflow are negligible when added to a value of at least 1, and so can safely be ignored.

We can also approximate a log-expectation of exponents with the log-sum-exp trick, since

$$\log \mathbb{E}[\exp a] \approx \log\left[\frac{1}{K} \sum_{k=1}^{K} \exp a_k\right] = \log \frac{1}{K} + \log \sum_{k=1}^{K} \exp a_k, \tag{4}$$

and this will prove useful in evaluating the MaxVar acquisition function.

Recall that we wish to find $\boldsymbol{\theta}$ which maximizes $f(\boldsymbol{\theta}) = \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})]$. This is the same $\boldsymbol{\theta}$ which maximizes $g(\boldsymbol{\theta}) = \log \mathbb{V}_{\boldsymbol{\phi}}[q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})]$, since the logarithm is monotonic. However, we only have sample values $\log q_{\boldsymbol{\phi}}(\mathbf{x}_0 \mid \boldsymbol{\theta})$, where $\boldsymbol{\phi} \sim q(\boldsymbol{\phi} \mid \mathcal{D})$. Phrasing the problem more generally for a non-negative random variable e^a, we wish to approximate $\log \mathbb{V}[e^a]$ in a numerically stable fashion, given log-samples a which may have large magnitude.

First rewrite

$$\log \mathbb{V}[e^a] = \log \mathbb{E}\left[(e^a - \mathbb{E}[e^a])^2\right] \tag{5}$$

$$= \log \mathbb{E}[\exp(2\log |e^a - \mathbb{E}[e^a]|)], \tag{6}$$

which is a log-expectation of exponents, and can be stably approximated by eq. (4). It remains to evaluate $\log|e^a - \mathbb{E}[e^a]|$ in a stable manner, given that we have access to log-samples a. To this end, write $\mathbb{E}[e^a] = e^b$, where $b = \log \mathbb{E}[e^a]$ can also be approximated by eq. (4), so that

$$\log|e^a - \mathbb{E}[e^a]| = \log|e^a - e^b| \tag{7}$$

$$= \log|e^a(1 - e^{b-a})| \tag{8}$$

$$= a + \log(1 - e^{b-a}) \quad \text{for } a > b. \tag{9}$$

Without loss of generality, we can assume $a > b$, and the above expression can also leverage the numerically stable log-one-plus function. Thus, the final expression is given by

$$\log \mathbb{V}[e^a] = \log \mathbb{E}\big[\exp\big(2\big(a + \log\big(1 - e^{b-a}\big)\big)\big)\big], \tag{10}$$

where $b = \log \mathbb{E}[e^a]$, $a = \log q_\phi(\mathbf{x}_0 \mid \boldsymbol{\theta})$, expectations are taken with respect to the distribution over neural density estimator parameters ϕ, and all log-expectation-exponent calculations can be approximated using eq. (4). We minimize the negative of this function using the L-BFGS-B optimizer [24] with constraints given by the box uniform priors on simulator parameters, and initializing each optimization run with a sample from the prior.

C   Additional Results

Though there seems to be no meaningful difference in the accuracy of the approximate posterior distributions learned by SNL and ASNL, it is interesting to compare the distributions of simulator parameters gathered by each of the methods towards the end of training. SNL, guided by the approximate posterior, should generate simulator parameters in a neighbourhood of the true parameters, if training has progressed suitably. From fig. 2, it seems as if this is exactly the case.

In fig. 3, ASNL exhibits different behaviour. Note first the difference in range on each of the axes; SNL generates parameters very close the true parameters, while SNL generates parameters across the entire range of the box uniform prior $[-5, 2]^4$. Though ASNL indeed focuses on neighbourhoods of the true parameters, it is also interested in other areas of parameter space. In particular, it generates many parameter settings near the boundary of the prior, and this is in line with results reported by Lueckmann et al. [14]. In some cases, it may be that this behaviour is desirable. While SNL is constrained to gather parameters where the likelihood is high, ASNL is free to explore the entire parameter space.
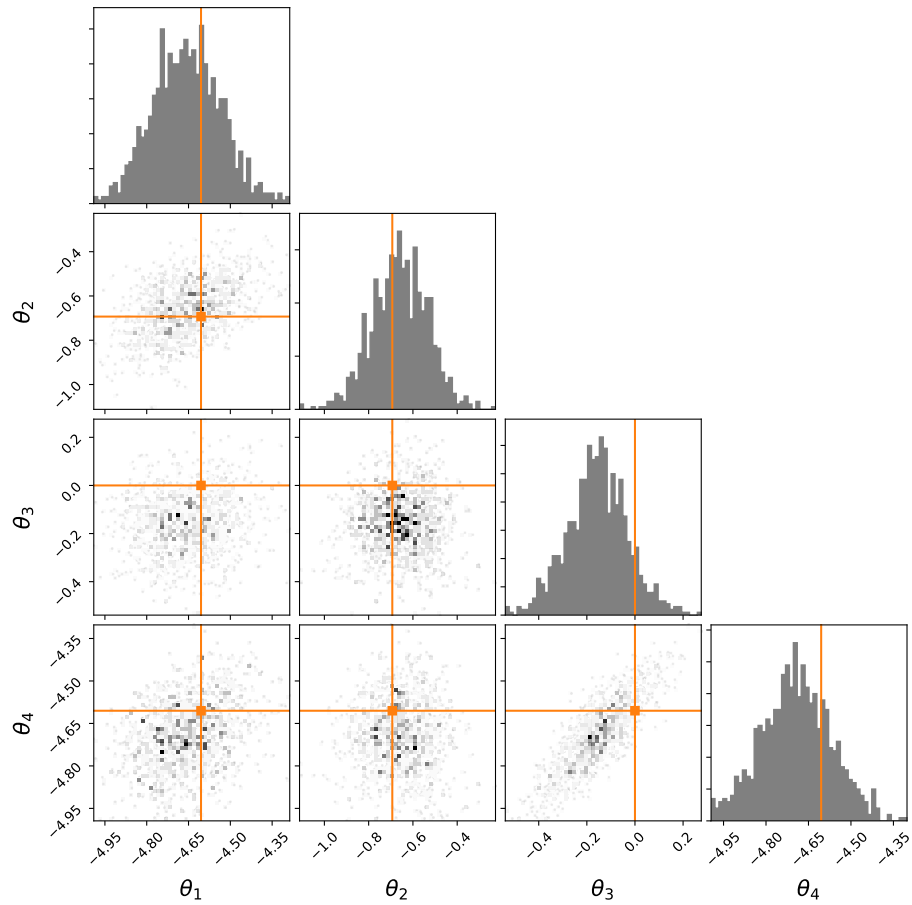
Figure 2: Simulator parameters generated using SNL before the final round of training for the Lotka–Volterra model. True parameters shown in orange. Since SNL generates parameters by performing MCMC on the current product of the approximate likelihood and prior, most of the gathered samples lie near the true parameter settings towards the end of training (compare to fig. 3).
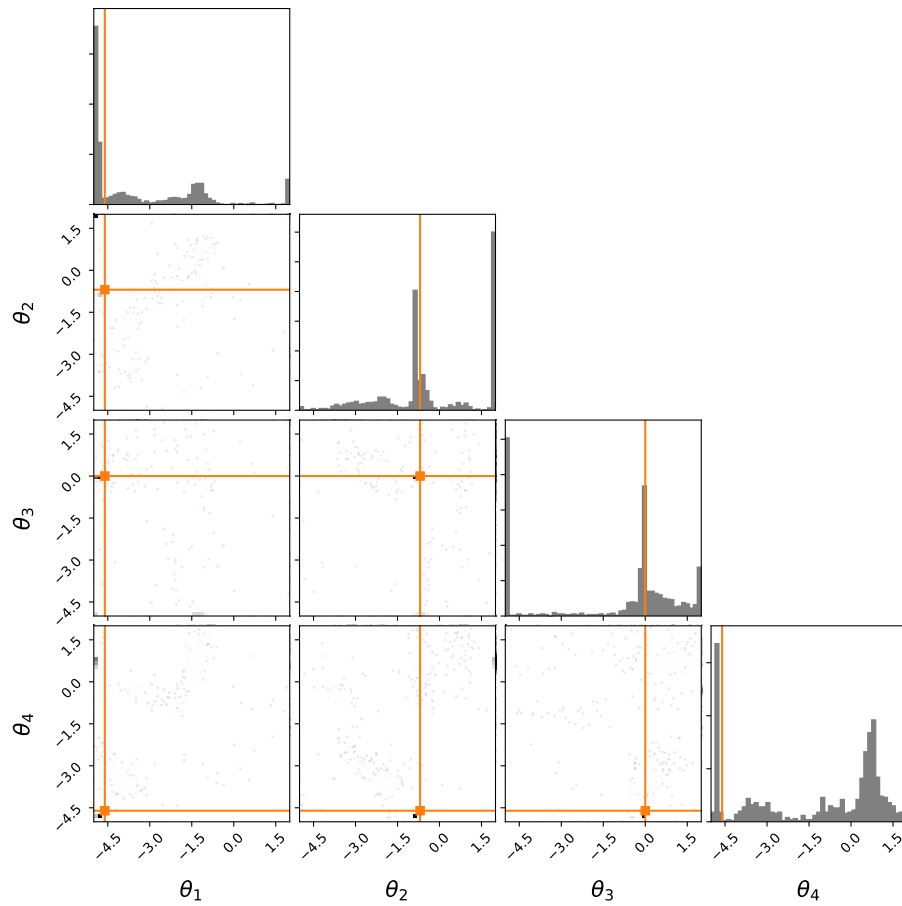
Figure 3: Simulator parameters generated using ASNL before the final round of training for the Lotka–Volterra model. True parameters shown in orange. ASNL generates simulator parameters using the MaxVar acquisition function, which focuses on regions of parameter space where our posterior approximation is most uncertain. Like SNL, many of the chosen parameters lie near the true values, but in contrast to SNL, ASNL seems also to be interested in the behaviour of the approximate posterior near the boundary defined by the box uniform prior (compare to fig. 2).