PNAS
www.pnas.org

Supplementary Information for

Learning complex models with invertible neural networks: a likelihood-free Bayesian approach

Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe

Stefan Radev.
E-mail: stefan.radev@psychologie.uni-heidelberg.de

This PDF file includes:

Other supplementary materials for this manuscript include the following:

Supporting Information Text

Complete code for all examples used in this paper is available at https://github.com/stefanradev93/cINN.

Performance metrics

In the following, the computation of the performance metrics used throughout the main text is detailed.

Normalized Root Mean Squared Error. The normalized root mean squared error (NRMSE) between a sample of true parameters $\{\theta^{(i)}\}_{i=1}^{n}$ and a sample of estimated parameters $\{\hat{\theta}^{(i)}\}_{i=1}^{n}$ is given by:

$$NRMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(\theta^{(i)} - \hat{\theta}^{(i)}\right)^2}{\theta_{max} - \theta_{min}}} \quad [1]$$

Due to the normalization factor $\theta_{max} - \theta_{min}$, the NRMSE is scale-independent, and thus suitable for comparing the recovery across parameters having different numerical ranges. The NRMSE is zero when the estimates are exactly equal to the true values.

Coefficient of Determination . The coefficient of determination R^2 gives the proportion of variance in a sample of true parameters $\{\theta^{(i)}\}_{i=1}^{n}$ that is "explained" by a sample of estimated parameters $\{\hat{\theta}^{(i)}\}_{i=1}^{n}$. It is computed as:

$$R^2 = 1 - \sum_{i=1}^{n} \frac{\left(\theta^{(i)} - \hat{\theta}^{(i)}\right)^2}{\left(\theta^{(i)} - \bar{\theta}^{(i)}\right)^2} \quad [2]$$

where $\bar{\theta}$ denotes the mean of the true parameter samples. When R^2 equals 1, it means that the estimates are perfect reconstructions of the true parameters.

Kullback-Leibler Divergence. The Kullback-Leibler divergence (D_{KL}) quantifies the increase in entropy incurred by approximating a target probability distribution P with a distribution Q. Its general form for absolutely continuous distributions is given by

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad [3]$$

where p and q denote the pdfs of P and Q. In the case where P and Q are both multivariate Gaussian distributions, the KL divergence can be computed in closed form (1):

$$D_{KL}(P||Q) = \frac{1}{2}\left[\log \frac{\det \boldsymbol{\Sigma}_q}{\det \boldsymbol{\Sigma}_p} + \mathrm{Tr}(\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\Sigma}_p) - d + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_q^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right] \quad [4]$$

where $\boldsymbol{\Sigma}_p$ and $\boldsymbol{\Sigma}_q$ denote the covariance matrices of p and q, $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$ the respective mean vectors, and d the number of dimensions of the Gaussian. In the case of diagonal Gaussian distributions, Eq.4 reduces to:

$$D_{KL}(P||Q) = \sum_{i=1}^{d}\left(\log \frac{\sigma_{q,i}}{\sigma_{p,i}} + \frac{\sigma_{p,i}^2 + (\mu_{q,i} - \mu_{p,i})^2}{2\sigma_{q,i}^2} - \frac{1}{2}\right) \quad [5]$$

Even though the KL divergence is not a proper distance metric, as it is not symmetric in its arguments, it can be used to quantify the error of approximation and serve as a metric for comparing different methods.

Simulation-Based Calibration. Simulation-based calibration is a recently proposed method for validating the accuracy of posterior samples generated by a Bayesian sampling method (2). It is based on the so called *self-consistency* of the Bayesian joint distribution. Given a sample from the prior distribution $\tilde{\theta} \sim p(\theta)$ and a sample from the data-generating process $\tilde{x} \sim p(x|\tilde{\theta})$, one can integrate $\tilde{\theta}$ and $\tilde{x}$ out of the Bayesian joint distribution to recover back the prior of θ:

$$p(\theta) = \int p(\theta, \tilde{\theta}, \tilde{x}) d\tilde{x} d\tilde{\theta} \quad [6]$$

$$= \int p(\theta, \tilde{x}|\tilde{\theta}) p(\tilde{\theta}) d\tilde{x} d\tilde{\theta} \quad [7]$$

$$= \int p(\theta|\tilde{x}) p(\tilde{x}|\tilde{\theta}) p(\tilde{\theta}) d\tilde{x} d\tilde{\theta} \quad [8]$$

If the Bayesian sampling method produces samples from the exact posterior, the equality implied by Eq.8 should hold regardless of the particular form of the posterior. Thus, any violation of this equality indicates some error incurred by the sampling method. The authors of (2) propose **Algorithm** 1 for visually detecting such violations:

Algorithm 1 is justified, since Eq.8 implies that the rank statistic defined in line 5 should be uniformly distributed. Hence, any deviations from uniformity indicate some error in the approximate posterior.

Algorithm 1 Simulation-based calibration (SBC) for a single parameter θ

for $i = 1, ..., n$ **do**
    Sample $\tilde{\theta}^{(i)} \sim p(\theta)$
    Simulate a dataset $\boldsymbol{x}^{(i)} = q(\tilde{\theta}^{(i)})$
    Draw posterior samples $\{\theta^{(l)}\}_{l=1}^{L} \sim p(\theta|\boldsymbol{x}^{(i)})$
    Compute rank statistic $r^{(i)} = \sum_{l=1}^{L} \mathbb{1}_{[\theta^{(l)} < \tilde{\theta}^{(i)}]}$
    Store $r^{(i)}$
end for
Create a histogram of $\{r^{(i)}\}_{i=1}^{n}$ and inspect it for uniformity

Invertible networks

Throughout all examples, we use a chain of 10 conditional affine coupling blocks (cACB). Each internal network of each ACB is implemented as a fully connected neural network with 3 to 4 hidden layers with 64 neurons each. See https://github.com/stefanradev93/cINN for complete implementation of all networks used throughout this paper.

Model details

Bayesian regression model.

Summary network. We use the recently introduced permutationally invariant neural network (3) for the *i.i.d.* regression data. This network architecture is a good choice for this example, as each observation is independent of all other, and thus all permutations of a given dataset should lead to the same parameter estimates. Furthermore, invariant networks are independent of the number of observations.

The Ricker model.

Summary network. We use a bidirectional long short-term memory (LSTM) recurrent neural network (4) for summarizing the Ricker time-series into fixed-size vectors. The LSTM network architecture is a good choice for this example, as it is able to capture long-term in datasets with temporal or spatial autocorrelations. LSTMs can also easily deal with variable length time-series.

Simulation. We place the following uniform priors over the Ricker model parameters:

$$\rho \sim \mathcal{U}(0, 15) \quad [9]$$
$$r \sim \mathcal{U}(1, 90) \quad [10]$$
$$\sigma \sim \mathcal{U}(0.05, 0.7) \quad [11]$$

These ranges are very broad, as datasets generated by extreme parameter values appear implausible in real-world scenarios. Nevertheless, we stick to broad priors for training, even though parameter recovery might degrade at the extremes. Figure XXX depicts some Ricker datasets generated by parameters drawn from the specified prior.

The Lévy-Flight Model.

Summary network. We use a permutationally invariant neural network (3) for the *i.i.d.* reaction times (RT) data. Similarly to the toy Regression example, each response in an RT dataset is assumed to be independent of all others, so permutations of the dataset must lead to the same parameter estimates.

Simulation. The following uniform priors over the LFM parameters are used for simulation:

$$v_0 \sim \mathcal{U}(0, 6) \quad [12]$$
$$v_1 \sim \mathcal{U}(-6, 0) \quad [13]$$
$$zr \sim \mathcal{U}(0.3, 0.7) \quad [14]$$
$$a \sim \mathcal{U}(0.6, 3) \quad [15]$$
$$t_0 \sim \mathcal{U}(0.3, 1) \quad [16]$$
$$\alpha \sim \mathcal{U}(1, 2) \quad [17]$$

These priors are broad enough to cover the range of realistic reaction times observed in choice RT experiments.

The stochastic SIR model.

Summary network. We use a $1D$ fully convolutional neural network (5) for summarizing the SIR time-series into fixed-size vectors. Here, we chose a convolutional network architecture over the previously mentioned LSTM, as convolutional networks are more computationally efficient. Further, we wanted to underline the utility of 1D convolutional networks for multidimensional time-series data. Finally, convolutional networks can also deal with variable input sizes.

Simulation. We place the following uniform priors over the two rate parameters of the stochastic SIR model:

$$\beta \sim \mathcal{U}(0.01, 1) \quad [18]$$
$$\gamma \sim \mathcal{U}(0.01, \beta) \quad [19]$$

Figure XXX depicts some SIR datasets generated by parameters drawn from the specified priors.

Single-Cell RNA Sequencing.

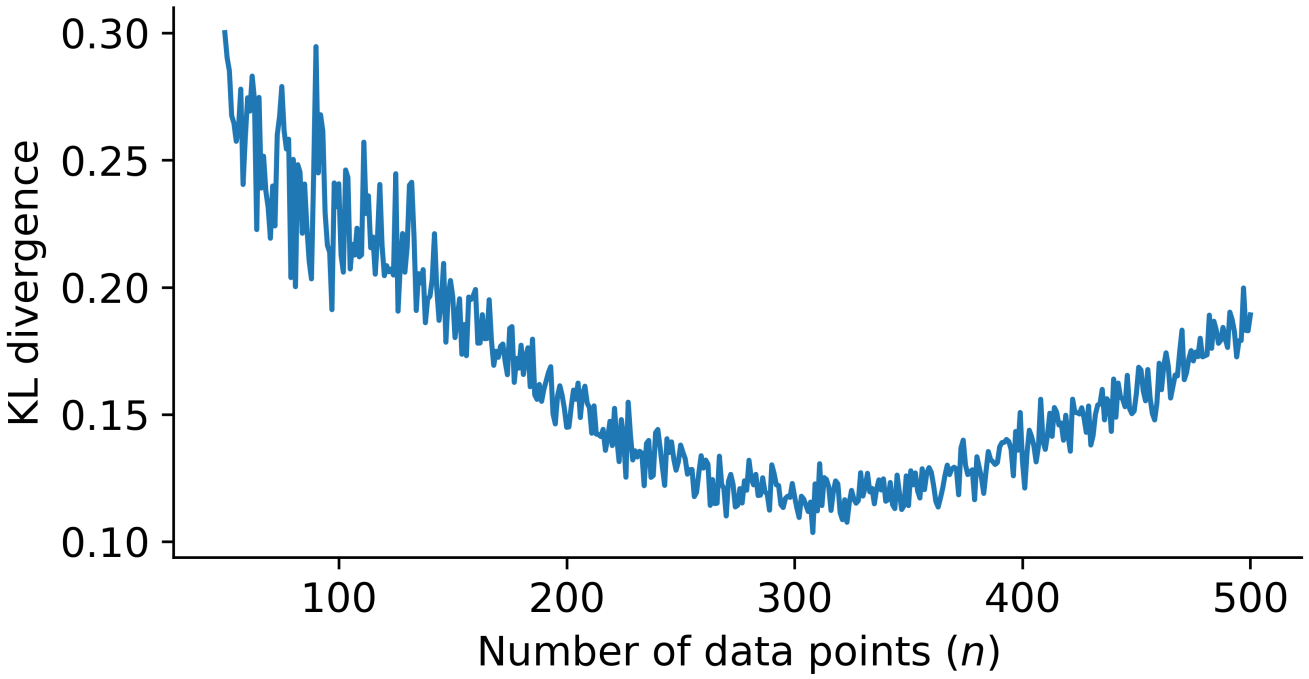

Fig. S1. KL divergence (D_{KL}) between the true and estimated posteriors of the regression weights in our toy Bayesian regression example. The D_{KL} is depicted at every n observations used during training.

Summary network. We use a network implementing the recently introduced *self-attention* mechanism (6). Self-attention architectures can learn to represent complicated pairwise relationships between observations. This property aligns nicely with the task at hand, since each entry in the observed count matrices is not independent of the other entries. Moreover, these dependencies do not exhibit the typical spatial patterns observed, for instance, in natural images.

Simulation. We place the following uniform priors over the parameters of the *Splat* simulation:

$$\alpha \sim \mathcal{U}(1, 2) \quad [20]$$
$$\beta \sim \mathcal{U}(0.1, 5) \quad [21]$$
$$\mu^L \sim \mathcal{U}(6, 10) \quad [22]$$
$$\sigma^L \sim \mathcal{U}(1, 2) \quad [23]$$
$$\pi^O \sim \mathcal{U}(0.01, 0.5) \quad [24]$$
$$\mu^O \sim \mathcal{U}(0.5, 5) \quad [25]$$
$$\sigma^O \sim \mathcal{U}(0.1, 2) \quad [26]$$
$$\phi \sim \mathcal{U}(0.1, 0.9) \quad [27]$$

Movie S1. Type caption for the movie here.

References

1. Hershey JR, Olsen PA (2007) Approximating the kullback leibler divergence between gaussian mixture models in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*. (IEEE), Vol. 4, pp. IV–317.
2. Talts S, Betancourt M, Simpson D, Vehtari A, Gelman A (2018) Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.
3. Bloem-Reddy B, Teh YW (2019) Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*.
4. Gers FA, Schmidhuber J, Cummins F (1999) Learning to forget: Continual prediction with lstm.
5. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 3431–3440.
6. Vaswani A, et al. (2017) Attention is all you need in *Advances in neural information processing systems*. pp. 5998–6008.