Simulation-Based Inference: Where Classical Statistics Meets Machine Learning

Mikael Kuusela

Department of Statistics and Data Science,
Carnegie Mellon University

AISSAI AI Uncertainty Workshop,
Paris, France

November 28, 2023

Joint work with: Julia Walchessen (CMU), Amanda Lenzi (Edinburgh), Luca Masserano (CMU), Tommaso Dorigo (INFN), Rafael Izbicki (São Carlos), and Ann B. Lee (CMU)

Motivation: Simulation-based inference

In most of statistical inference, we assume that our data $\boldsymbol{y}$ is generated from probability distribution $F_{\boldsymbol{\theta}}$ parameterized by a parameter of interest $\boldsymbol{\theta}$:

$$\boldsymbol{y} \sim F_{\boldsymbol{\theta}}$$

We might then be interested in:

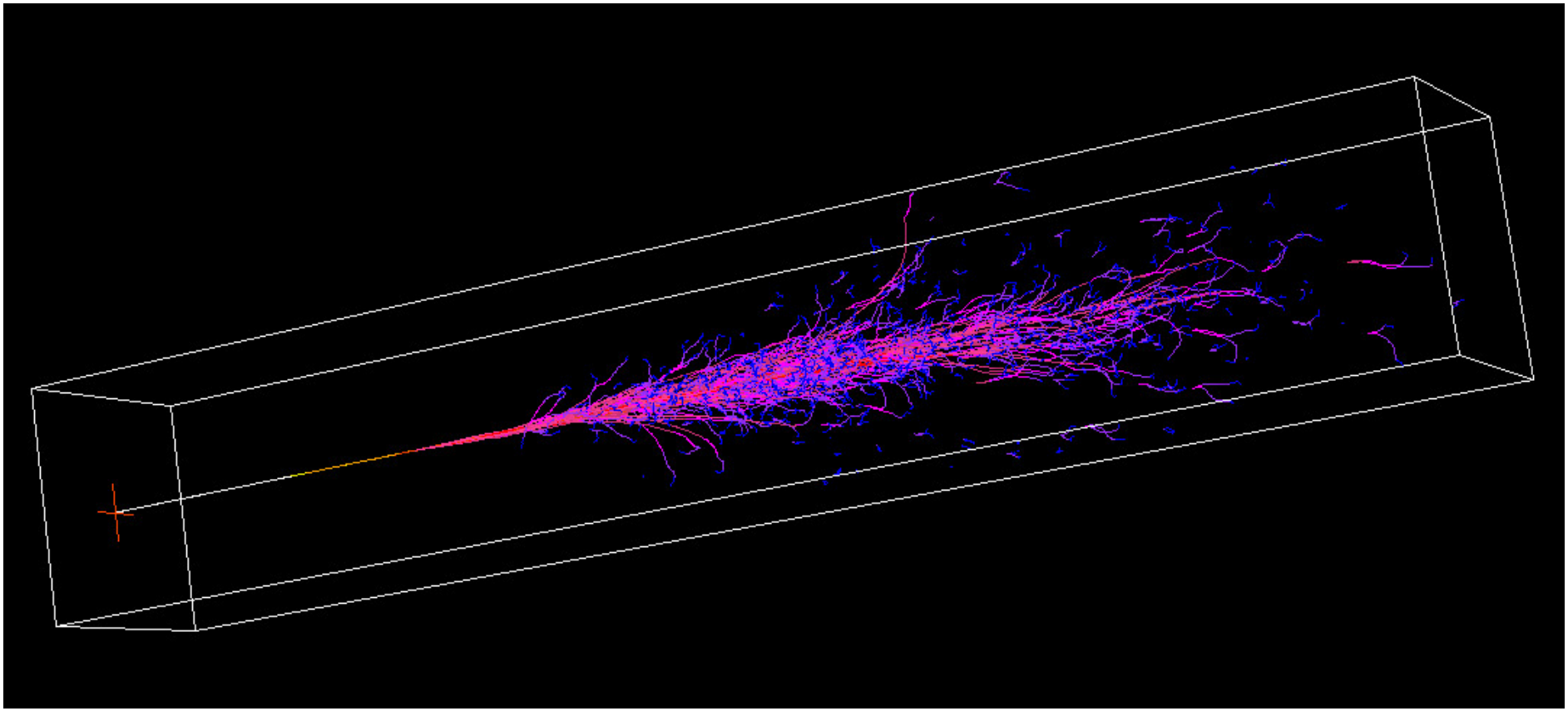
- Point estimates of $\boldsymbol{\theta}$
- Hypothesis tests about $\boldsymbol{\theta}$
- Confidence sets for $\boldsymbol{\theta}$
- ...

These are well-understood problems when we can write down and evaluate a closed-form expression for $F_{\boldsymbol{\theta}}$

However, there are many situations where $F_{\boldsymbol{\theta}}$ is implicitly defined through a generative process that allows *simulating* $\boldsymbol{y}$ for given $\boldsymbol{\theta}$ but does not easily admit a tractable expression for $F_{\boldsymbol{\theta}}$

$\Rightarrow$ Simulation-based inference (SBI)

Motivation: Simulated electromagnetic shower



Simulated electomagnetic shower for 24 GeV electron in iron
(Figure source: `https://www.mpp.mpg.de/~menke/elss/`)

Motivation: Intractable spatial processes

Max-stable processes are a popular class of models for spatial extremes

- Used to model extreme weather phenomena, such as extreme precipitation and heat waves

These processes are easy to simulate from but the number of terms in the likelihood function grows faster than exponentially in the number of spatial locations

- Exact likelihood computations become intractable even with HPC for $\gtrsim 10$ locations (Castruccio et al., 2016)

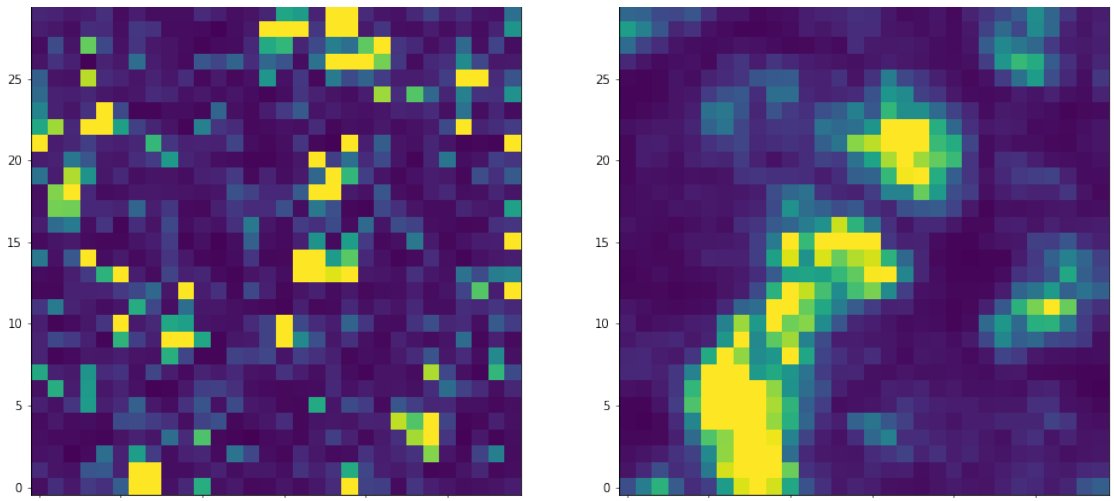


Figure: Two realizations of a Brown–Resnick spatial extremes process

Simulation-based inference

Ingredients:

- Sample of parameters: $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \sim p(\boldsymbol{\theta})$
- Corresponding simulations from the model: $\boldsymbol{y}_i \sim F_{\boldsymbol{\theta}_i},\ i = 1, \ldots, n$
- Observed data: $\boldsymbol{y}_{\text{obs}}$

Task: Infer $\boldsymbol{\theta}$ that generated $\boldsymbol{y}_{\text{obs}}$

Specifically, for uncertainty quantification, we want to obtain a frequentist confidence set $C(\boldsymbol{y})$ satisfying[1]

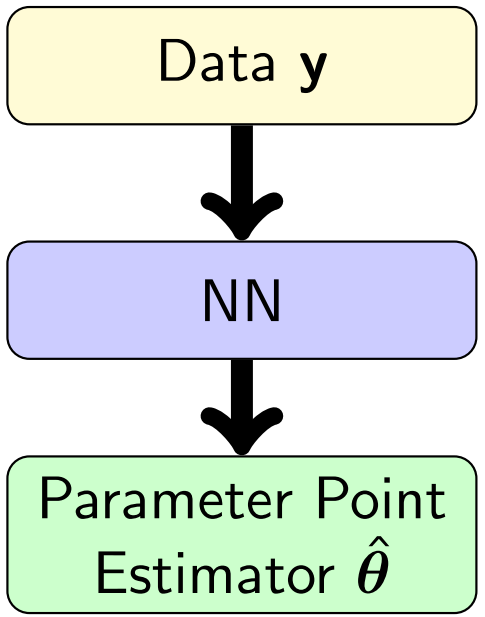
$$\mathbb{P}(\boldsymbol{\theta} \in C(\boldsymbol{y}) | \boldsymbol{\theta}) = 1 - \alpha, \quad \forall \boldsymbol{\theta}$$

Note: In most of these problems, $\boldsymbol{y}$ is high-dimensional

[1] This is sometimes called *conditional coverage*, as opposed to *marginal coverage* $\mathbb{P}(\boldsymbol{\theta} \in C(\boldsymbol{y})) = 1 - \alpha$.

Neural prediction for SBI

Given the simulated pairs $\{\boldsymbol{y}_i, \boldsymbol{\theta}_i\}_{i=1}^n$, an immediate idea is to train a neural network h that takes $\boldsymbol{y}$ as input and gives the corresponding parameter $\boldsymbol{\theta}$ as output



This *neural prediction* approach has been used for

- Particle reconstruction (e.g., Kieseler et al., 2022)
- Estimating galaxy masses (Ho et al., 2019)
- Estimating parameters of spatial models (Gerber and Nychka, 2021; Lenzi et al., 2023; Sainsbury-Dale et al., 2023)
- ...

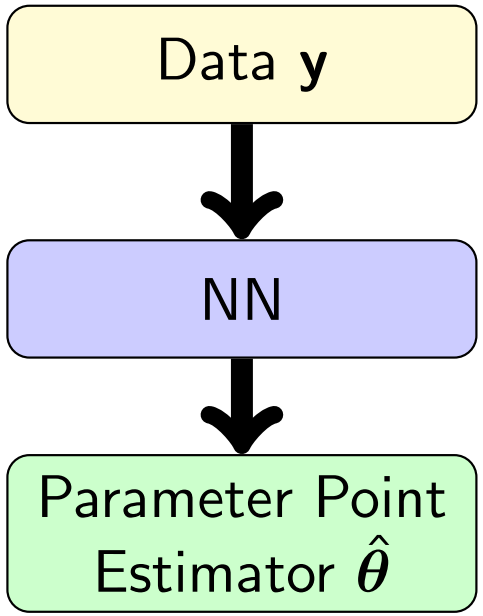
Neural prediction for SBI

The network is typically trained to minimize the mean squared error loss function:

$$L = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\theta}_i - h(\boldsymbol{y}_i))^2 \overset{n\to\infty}{\longrightarrow} \mathbb{E}((\boldsymbol{\theta} - h(\boldsymbol{y}))^2)$$

For large samples, the minimizer is the conditional expectation: $h(\boldsymbol{y}) = \mathbb{E}(\boldsymbol{\theta}|\boldsymbol{y})$

The neural predictor $\hat{\boldsymbol{\theta}}$ is therefore a regularized finite-sample estimator of $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{y})$



Neural prediction for SBI

Some limitations of neural prediction:

1. Prior dependence:
   - the distribution $p(\boldsymbol{\theta})$ used to sample the training parameters serves as a prior
   - $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{y})$ depends on this prior since the expectation is with respect to $p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$
   - the estimates $\hat{\boldsymbol{\theta}}$ are biased toward the prior mean
2. Uncertainty quantification:
   - Could train another network to estimate $\mathbb{V}(\boldsymbol{\theta}|\boldsymbol{y})$ and use this to quantify uncertainty
   - This should be understood as Bayesian UQ
   - May not have good frequentist properties
3. Multiple realizations:
   - Depending on the method used, inference for multiple i.i.d. realizations may require retraining the network

What else could be done using the sample $\{\boldsymbol{y}_i, \boldsymbol{\theta}_i\}_{i=1}^n$?

In the following, I'll describe our work Walchessen et al. (2023) where we learn the likelihood function of intractable spatial processes using this same sample

Neural likelihood

We will train a probabilistic neural network classifier $h(\boldsymbol{y}, \boldsymbol{\theta})$ whose odds transform is proportional to the likelihood:

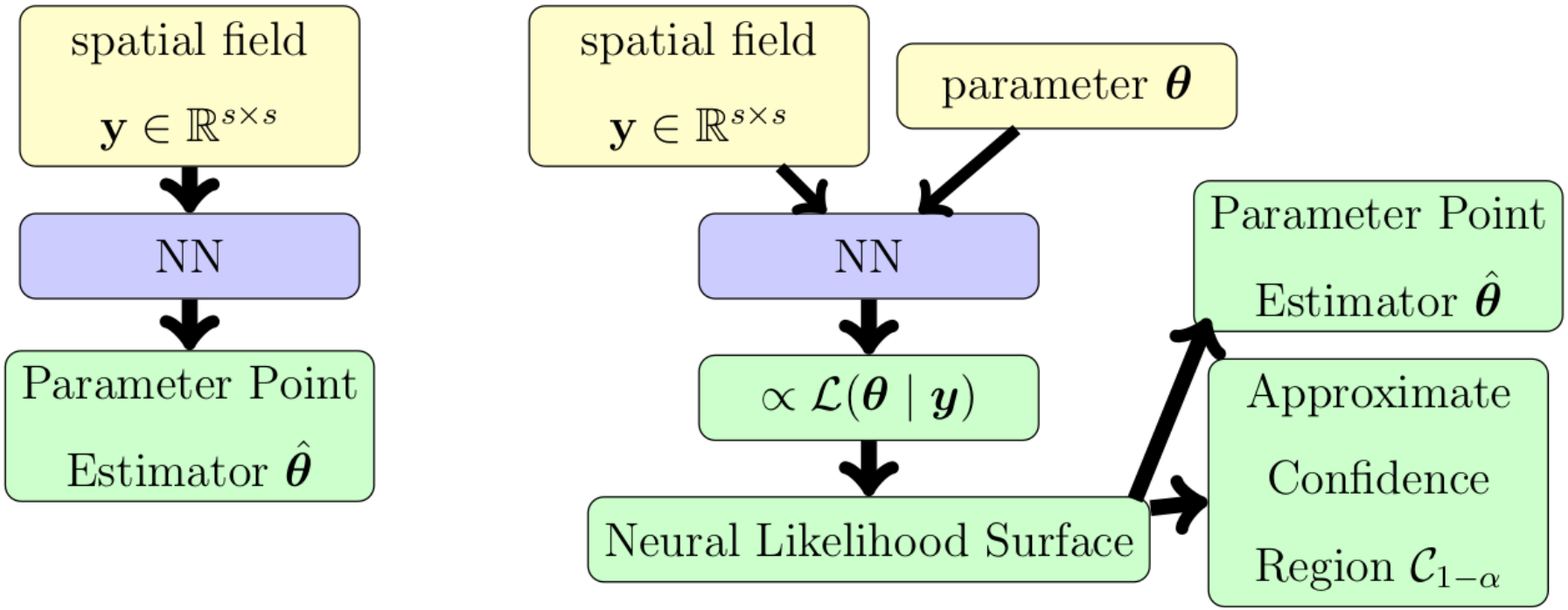
$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \text{odds}(h(\boldsymbol{y}, \boldsymbol{\theta})) = \frac{h(\boldsymbol{y}, \boldsymbol{\theta})}{1 - h(\boldsymbol{y}, \boldsymbol{\theta})}$$

So we can evaluate the classifier $h(\boldsymbol{y}, \boldsymbol{\theta})$ for given data $\boldsymbol{y}$ over a grid of $\boldsymbol{\theta}$ to trace out the likelihood surface corresponding to $\boldsymbol{y}$ (up to a multiplicative constant depending on $\boldsymbol{y}$ but not on $\boldsymbol{\theta}$)

Benefits of this approach:

1. Point estimation using maximum likelihood
2. Likelihood-based confidence sets for uncertainty quantification
3. Much faster to evaluate than exact likelihood or traditional likelihood approximations (once trained, the neural network is *amortized*)
4. Does not depend on a prior
5. Handling multiple replications is straightforward

Neural prediction vs. neural likelihood



Left: neural prediction; right: neural likelihood

Neural likelihood: training data

Generate parameters $\boldsymbol{\theta}_i$ from some distribution $p(\boldsymbol{\theta})$ over a bounded parameter space Θ

For each $\boldsymbol{\theta}_i$, simulate $f_i \sim \mathrm{SpatialProcess}(\boldsymbol{\theta}_i)$; then evaluate f_i on a regular grid $\mathcal{S}$ to obtain a simulated spatial observation $\boldsymbol{y}_i = f_i(\mathcal{S})$

This gives us paired training data $\mathcal{C}_1 = \{\boldsymbol{y}_i, \boldsymbol{\theta}_i\}_{i=1}^n$; this is our class 1

For class 2, we permute the pairing of $\boldsymbol{y}_i$ and $\boldsymbol{\theta}_i$: $\mathcal{C}_2 = \{\boldsymbol{y}_i, \boldsymbol{\theta}_{\pi(i)}\}_{i=1}^n$, where $i \mapsto \pi(i)$ is a random permutation of the index set $[n]$

Due to the permutation, $\boldsymbol{y}$ and $\boldsymbol{\theta}$ are independent in class 2 with the same marginal distributions as in class 1

Then we train a probabilistic classifier $h(\boldsymbol{y}, \boldsymbol{\theta})$ to separate $\mathcal{C}_1$ from $\mathcal{C}_2$

(In practice, we found that it is better to sample multiple $\boldsymbol{y}$ for each $\boldsymbol{\theta}$ but the core idea remains the same.)

Connection between classifier and likelihood

If we train the classifier to minimize the binary cross-entropy loss, then it will (asymptotically) learn the class probability: $h(\boldsymbol{y}, \boldsymbol{\theta}) = P(\mathcal{C}_1|(\boldsymbol{y}, \boldsymbol{\theta}))$

This class probability can be re-expressed as follows:

$$h(\boldsymbol{y}, \boldsymbol{\theta}) = P\big(\mathcal{C}_1 \mid (\boldsymbol{y}, \boldsymbol{\theta})\big) = \frac{p\big((\boldsymbol{y}, \boldsymbol{\theta}) \mid \mathcal{C}_1\big) P(\mathcal{C}_1)}{p\big((\boldsymbol{y}, \boldsymbol{\theta}) \mid \mathcal{C}_1\big) P(\mathcal{C}_1) + p\big((\boldsymbol{y}, \boldsymbol{\theta}) \mid \mathcal{C}_2\big) P(\mathcal{C}_2)}$$

$$= \frac{\frac{p((\boldsymbol{y},\boldsymbol{\theta})|\mathcal{C}_1)}{p((\boldsymbol{y},\boldsymbol{\theta})|\mathcal{C}_2)}}{\frac{p((\boldsymbol{y},\boldsymbol{\theta})|\mathcal{C}_1)}{p((\boldsymbol{y},\boldsymbol{\theta})|\mathcal{C}_2)} + 1} = \frac{\frac{p(\boldsymbol{y},\boldsymbol{\theta})}{p(\boldsymbol{y})p(\boldsymbol{\theta})}}{\frac{p(\boldsymbol{y},\boldsymbol{\theta})}{p(\boldsymbol{y})p(\boldsymbol{\theta})} + 1} = \frac{\frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y})p(\boldsymbol{\theta})}}{\frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y})p(\boldsymbol{\theta})} + 1} = \frac{\frac{p(\boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{y})}}{\frac{p(\boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{y})} + 1} = \frac{\frac{\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})}{p(\boldsymbol{y})}}{\frac{\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})}{p(\boldsymbol{y})} + 1}$$

Solving this for $\psi(\boldsymbol{y}, \boldsymbol{\theta}) = \frac{\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})}{p(\boldsymbol{y})}$ gives

$$\psi(\boldsymbol{y}, \boldsymbol{\theta}) = \frac{h(\boldsymbol{y}, \boldsymbol{\theta})}{1 - h(\boldsymbol{y}, \boldsymbol{\theta})}$$

and hence

$$\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}) = p(\boldsymbol{y})\psi(\boldsymbol{y}, \boldsymbol{\theta}) \propto \frac{h(\boldsymbol{y}, \boldsymbol{\theta})}{1 - h(\boldsymbol{y}, \boldsymbol{\theta})}$$

Classifiers and Density Ratios

The previous result is a special case of a more general result. Let:

$$\mathcal{C}_1: \quad \boldsymbol{z}_1^1, \boldsymbol{z}_2^1, \ldots, \boldsymbol{z}_n^1 \overset{\text{iid}}{\sim} p_1$$

$$\mathcal{C}_2: \quad \boldsymbol{z}_1^2, \boldsymbol{z}_2^2, \ldots, \boldsymbol{z}_n^2 \overset{\text{iid}}{\sim} p_2$$

Train a classifier $h(\boldsymbol{z})$ to separate $\mathcal{C}_1$ from $\mathcal{C}_2$ by minimizing the binary cross-entropy loss; then the classifier will asymptotically give:

$$h(\boldsymbol{z}) = P(\mathcal{C}_1 \mid \boldsymbol{z}) = \frac{p(\boldsymbol{z} \mid \mathcal{C}_1)P(\mathcal{C}_1)}{p(\boldsymbol{z} \mid \mathcal{C}_1)P(\mathcal{C}_1) + p(\boldsymbol{z} \mid \mathcal{C}_2)P(\mathcal{C}_2)} = \frac{\frac{p(\boldsymbol{z}|\mathcal{C}_1)}{p(\boldsymbol{z}|\mathcal{C}_2)}}{\frac{p(\boldsymbol{z}|\mathcal{C}_1)}{p(\boldsymbol{z}|\mathcal{C}_2)} + 1} = \frac{\frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})}}{\frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})} + 1}$$

Solving this for $\psi(\boldsymbol{z}) = \frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})}$ gives $\psi(\boldsymbol{z}) = \frac{p_1(\boldsymbol{z})}{p_2(\boldsymbol{z})} = \frac{h(\boldsymbol{z})}{1-h(\boldsymbol{z})} = \text{odds}(h(\boldsymbol{z}))$

The classifier $h(\boldsymbol{z})$ is learning the density ratio $p_1(\boldsymbol{z})/p_2(\boldsymbol{z})$!

- When $h(\boldsymbol{z})$ is a (deep) neural network, this works amazingly well for high-dimensional $\boldsymbol{z}$
- This can be used to learn likelihood ratios (Cranmer et al., 2015), to perform two-sample testing (Chakravarti et al., 2023), for transfer learning (Manole et al., 2022) and many other tasks in high-dimensional spaces

Calibration with Platt scaling

Learning the likelihood accurately depends on the accuracy of learning the class probability $P(\mathcal{C}_1|(\boldsymbol{y}, \boldsymbol{\theta}))$

But large-scale neural networks are known to produce biased estimates of $P(\mathcal{C}_1|(\boldsymbol{y}, \boldsymbol{\theta}))$ for finite samples (Guo et al., 2017)

A standard solution to this is to apply *post-hoc calibration* on the network

We do this using Platt scaling (Platt, 1999) which simply means fitting a logistic regression to the training class labels using the neural network output as a covariate:

$$\log\left(\frac{\pi(p)}{1-\pi(p)}\right) = \beta_0 + \beta_1 \log\left(\frac{p}{1-p}\right),$$

where p is the output of the uncalibrated neural network and $\pi(p)$ is the calibrated class probability

Calibration with Platt scaling

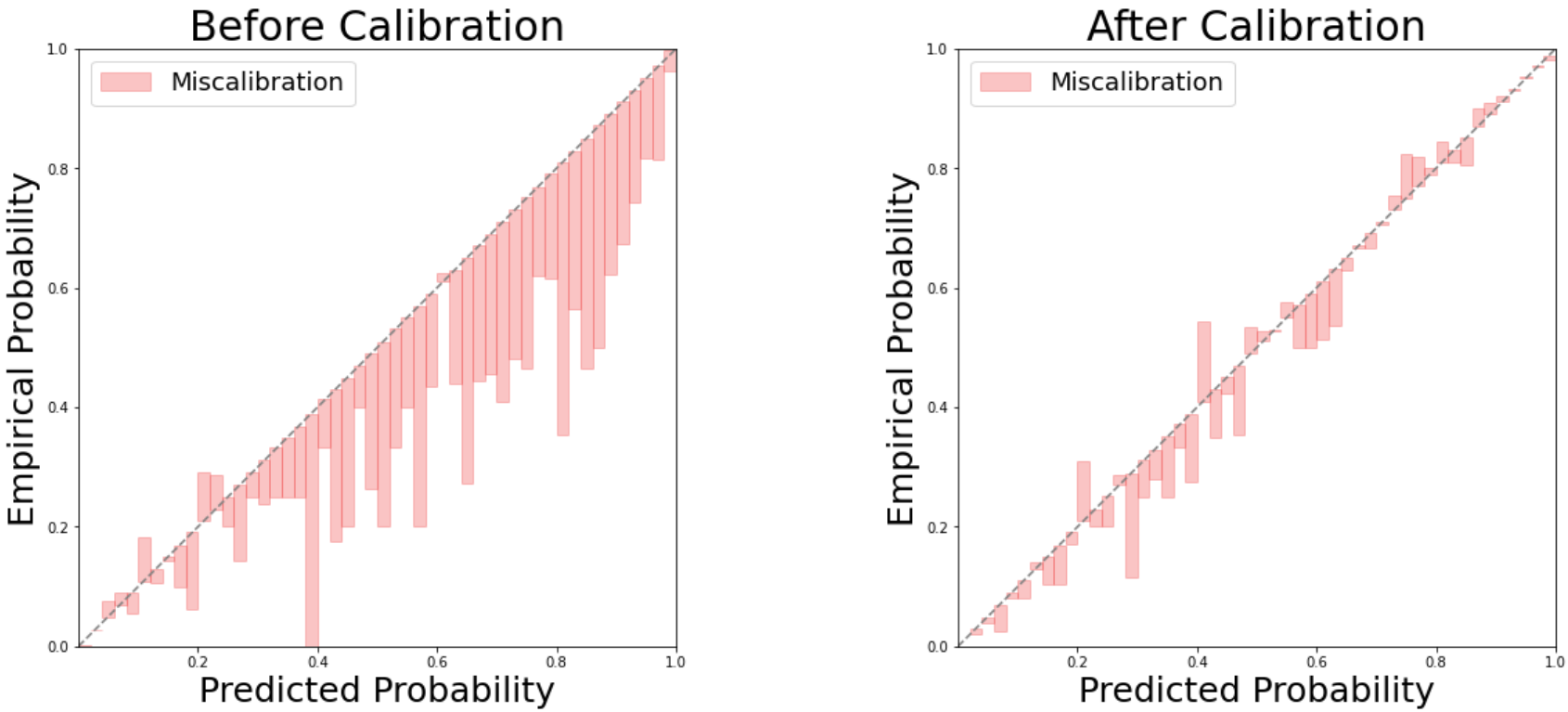


Figure: Reliability diagrams (empirical probability vs. predicted probability) before calibration (left) and after calibration (right) for a Gaussian process.

Using the neural likelihood

We can extract the following information from the trained and calibrated classifier $h(\boldsymbol{y}, \boldsymbol{\theta})$:

1. Log-likelihood surface: For fixed $\boldsymbol{y}$, perform a vectorized evaluation of $\log \psi(\boldsymbol{y}, \boldsymbol{\theta}) = \log\left(\frac{h(\boldsymbol{y},\boldsymbol{\theta})}{1-h(\boldsymbol{y},\boldsymbol{\theta})}\right)$ for $\boldsymbol{\theta} \in \Theta^L$, where Θ^L is a fine grid on the parameter space Θ
2. Point estimator: Find

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta^L}{\arg\max}\, \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \underset{\boldsymbol{\theta}\in\Theta^L}{\arg\max}\, \log(p(\boldsymbol{y})\psi(\boldsymbol{y}, \boldsymbol{\theta})) = \underset{\boldsymbol{\theta}\in\Theta^L}{\arg\max}\, \log \psi(\boldsymbol{y}, \boldsymbol{\theta})$$

   using the above grid evaluations
3. Approximate confidence region: Find

$$\begin{aligned}&\{\boldsymbol{\theta} \in \Theta^L \mid 2\big(\log \mathcal{L}(\hat{\boldsymbol{\theta}} \mid \boldsymbol{y}) - \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y})\big) \leq \chi^2_{k,1-\alpha}\} \\ &= \{\boldsymbol{\theta} \in \Theta^L \mid 2\big(\log \psi(\boldsymbol{y}, \hat{\boldsymbol{\theta}}) - \log \psi(\boldsymbol{y}, \boldsymbol{\theta})\big) \leq \chi^2_{k,1-\alpha}\},\end{aligned}$$

   using the above grid evaluations

Notice that in 2 and 3, the unknown constant $p(\boldsymbol{y})$ cancels out

Case study 1: Gaussian process

We first test our method on learning the likelihood function of a Gaussian process

- Here the exact likelihood is known so we can compare the neural likelihood to the exact likelihood

We consider a zero-mean spatial Gaussian process with exponential covariance $k(\boldsymbol{s}_1, \boldsymbol{s}_2) = v \cdot \exp\left(-\frac{\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|}{\ell}\right)$, so the unknown parameter is $\boldsymbol{\theta} = (\ell, v)$

The realizations $\boldsymbol{y}$ are generated on a 25×25 grid on the domain $\mathcal{D} = [-10, 10] \times [-10, 10]$

The parameter space Θ is set to be the bounded domain $(0, 2) \times (0, 2)$ (expanded to $(0, 2.5) \times (0, 2.5)$ when training the neural network)

Training data: 3000 parameters $\boldsymbol{\theta}$ sampled using Latin hypercube sampling; 500 field realizations $\boldsymbol{y}$ for each $\boldsymbol{\theta}$

Gaussian Process Neural Likelihood Surfaces

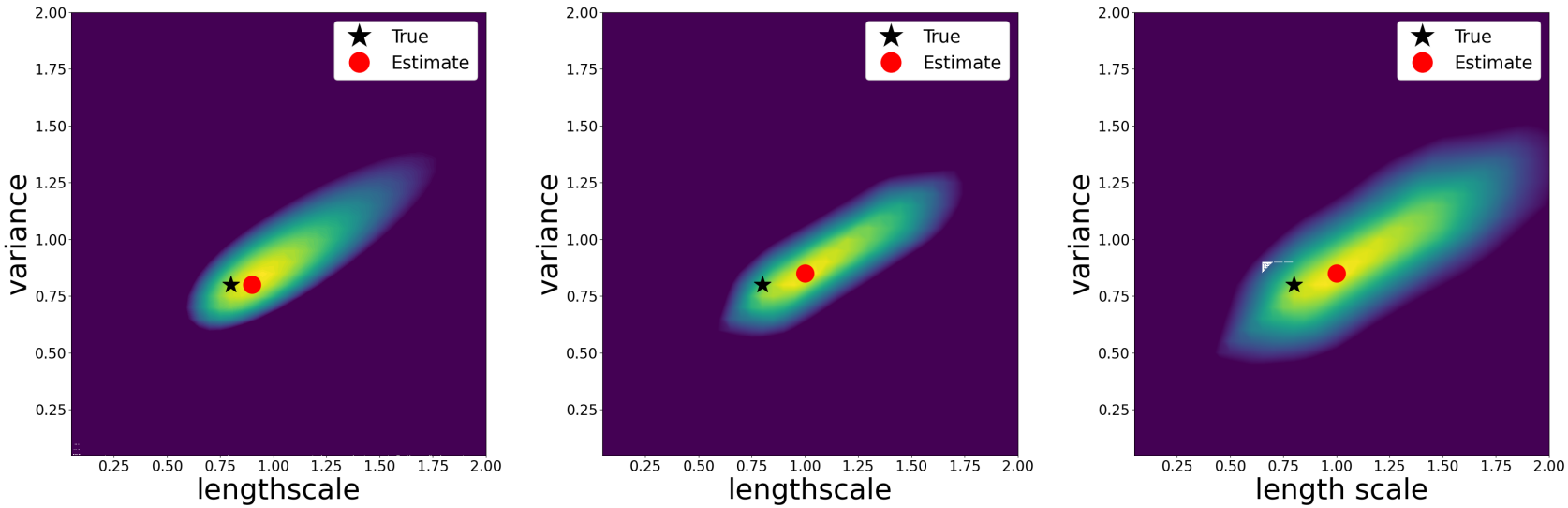


Figure: Exact log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for $\ell = 0.8$ and $v = 0.8$

Gaussian Process Neural Likelihood Surfaces

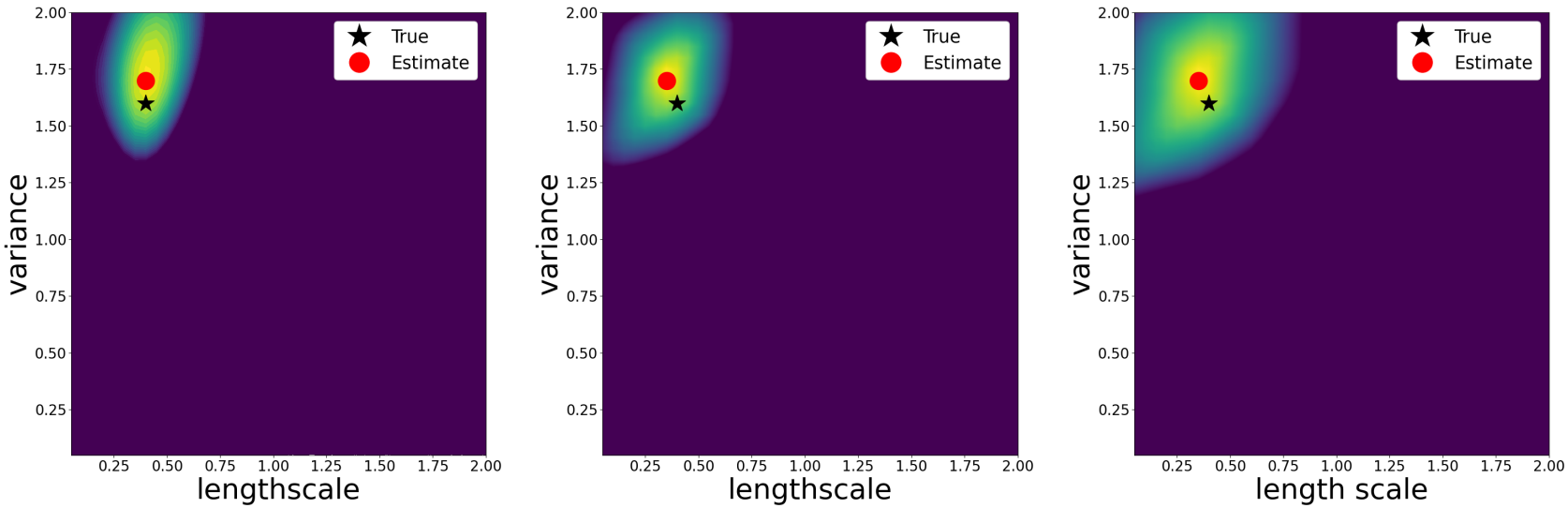


Figure: Exact log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for $\ell = 0.4$ and $v = 1.6$

Gaussian Process Neural Likelihood Surfaces

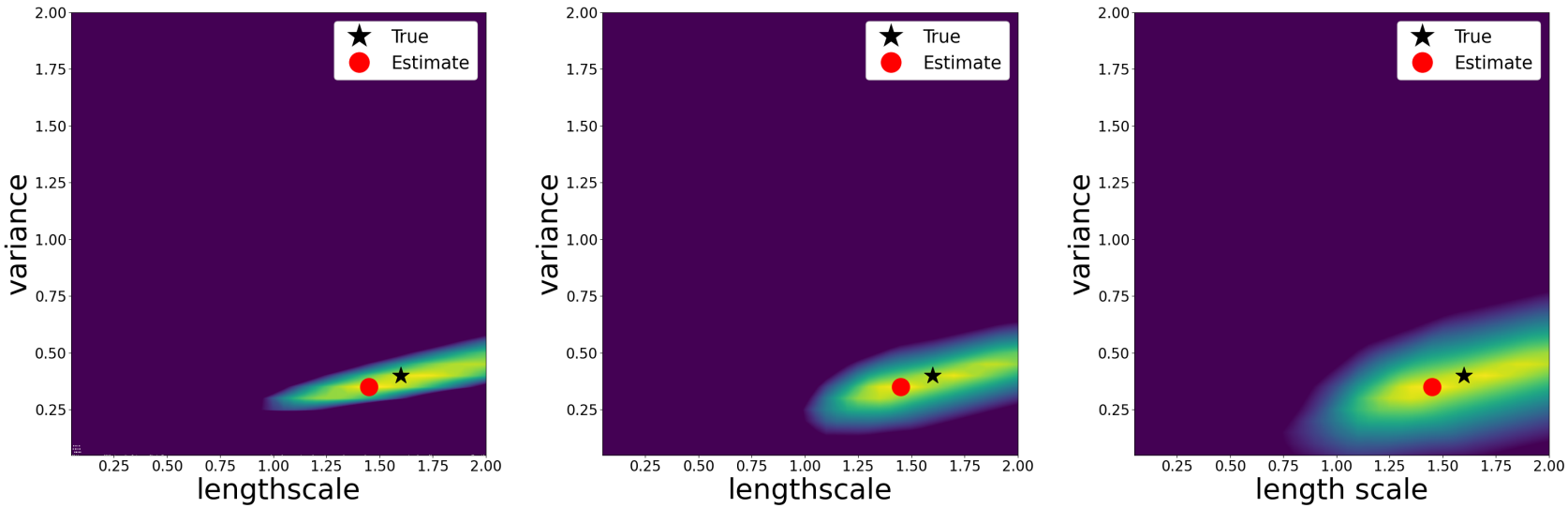


Figure: Exact log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for $\ell = 1.6$ and $v = 0.4$

Gaussian Process Approximate Confidence Regions

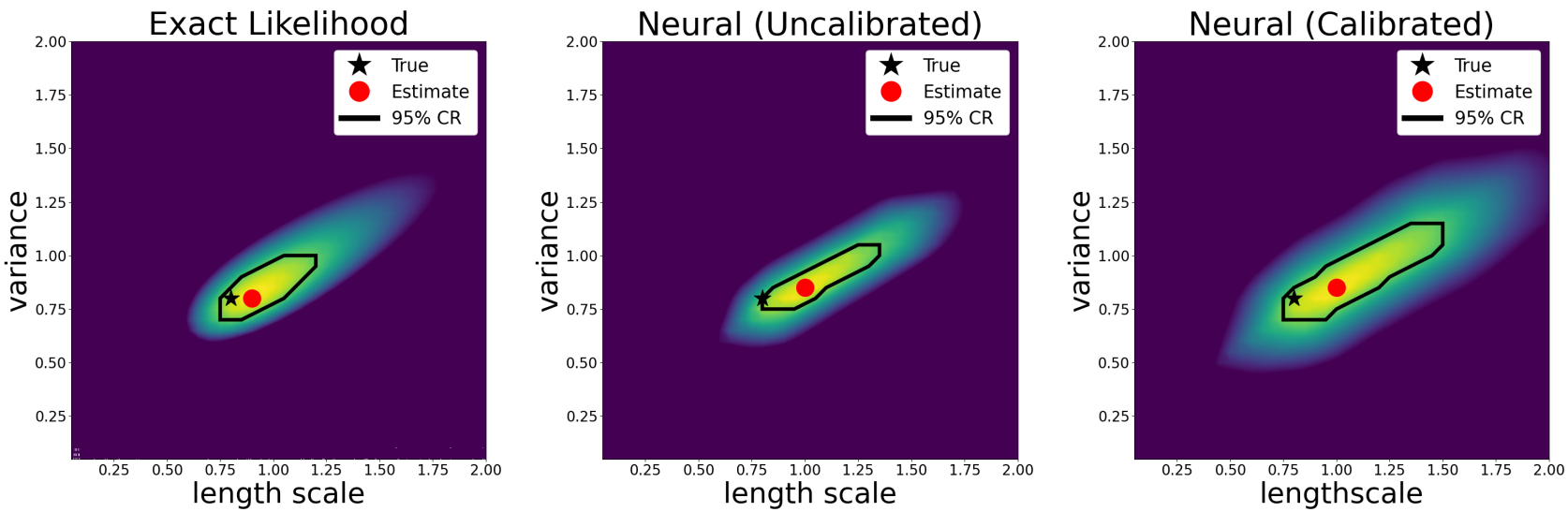


Figure: Exact log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for $\ell = 0.8$ and $v = 0.8$ along with 95% approximate confidence regions

Empirical Coverage for Gaussian Process

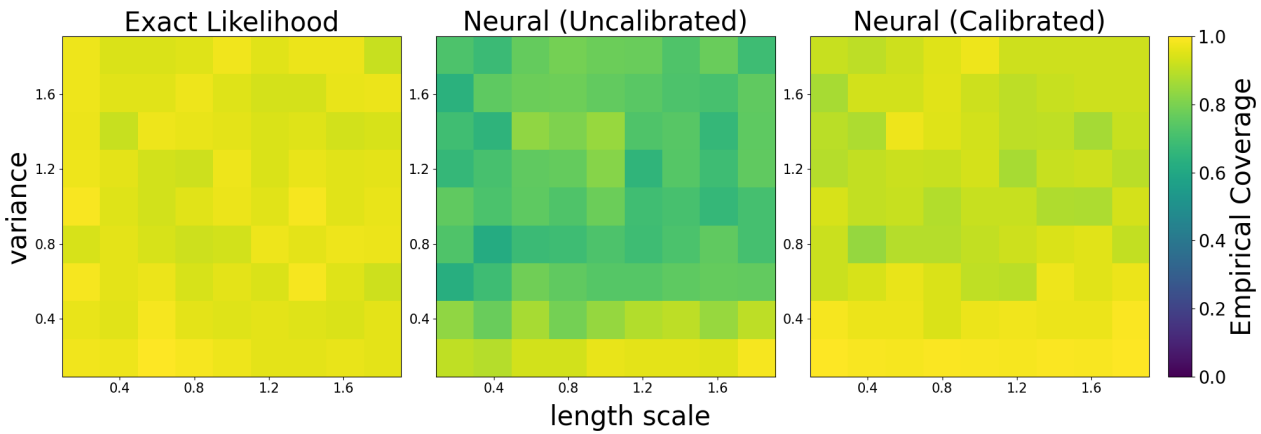


Figure: Empirical coverage

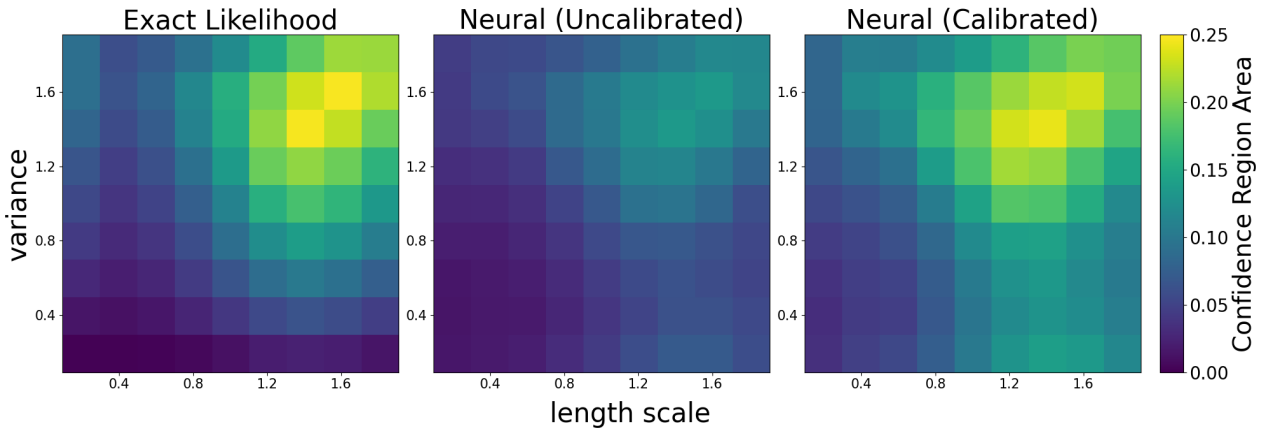


Figure: Confidence region area

Timing Study for Gaussian Process

Table: Time to produce neural and exact likelihood surfaces on a 40×40 grid over Θ for 50 realizations of a Gaussian process observed on a 25×25 grid.

Type of surface and method	average (sec)	std. dev. (sec)
exact likelihood	71.67	1.02
unvectorized neural likelihood	13.46	0.26
vectorized neural likelihood	**2.26**	0.34

Case study 2: Brown–Resnick Process

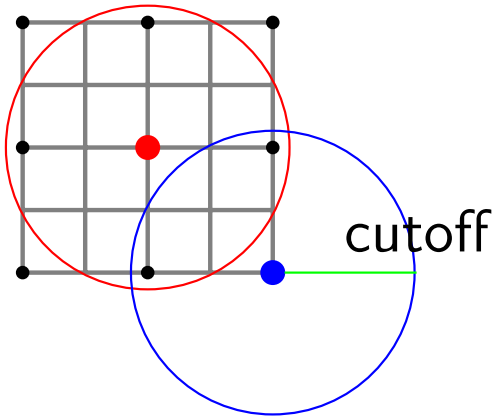
We now apply our method to estimating the likelihood function of a Brown–Resnick process

In this case, the exact likelihood is intractable (and hence unknown) so we compare our results to those obtained using pairwise likelihood

$$\log \mathcal{L}_{\mathrm{pw}}(\boldsymbol{\theta} \mid \boldsymbol{y}) = \sum_{j_1=1}^{n} \sum_{j_2 > j_1} w_{j_1,j_2} \left(\log \left(V_1(y_{j_1}, y_{j_2}) V_2(y_{j_1}, y_{j_2}) - V_{12}(y_{j_1}, y_{j_2}) \right) - V(y_{j_1}, y_{j_2}) \right)$$

The weights are taken to be:

$$w_{j_1,j_2} = \begin{cases} 1, & \text{if } \|\boldsymbol{s}_{j_1} - \boldsymbol{s}_{j_2}\| \leq \delta, \\ 0, & \text{otherwise.} \end{cases}$$



Results for pairwise likelihood depend on the distance cutoff δ

Brown–Resnick details

For Brown–Resnick, the parameter is $\boldsymbol{\theta} = (\lambda, \nu)$, where λ is a range parameter and ν is a smoothness parameter

We set the parameter space Θ to be the bounded set $(0, 2) \times (0, 2)$

Similar to the previous case study, the realizations $\boldsymbol{y}$ are generated on a 25×25 grid on the domain $\mathcal{D} = [-10, 10] \times [-10, 10]$

Training data sample sizes are the same as for Gaussian process

Brown–Resnick Neural Likelihood Surfaces

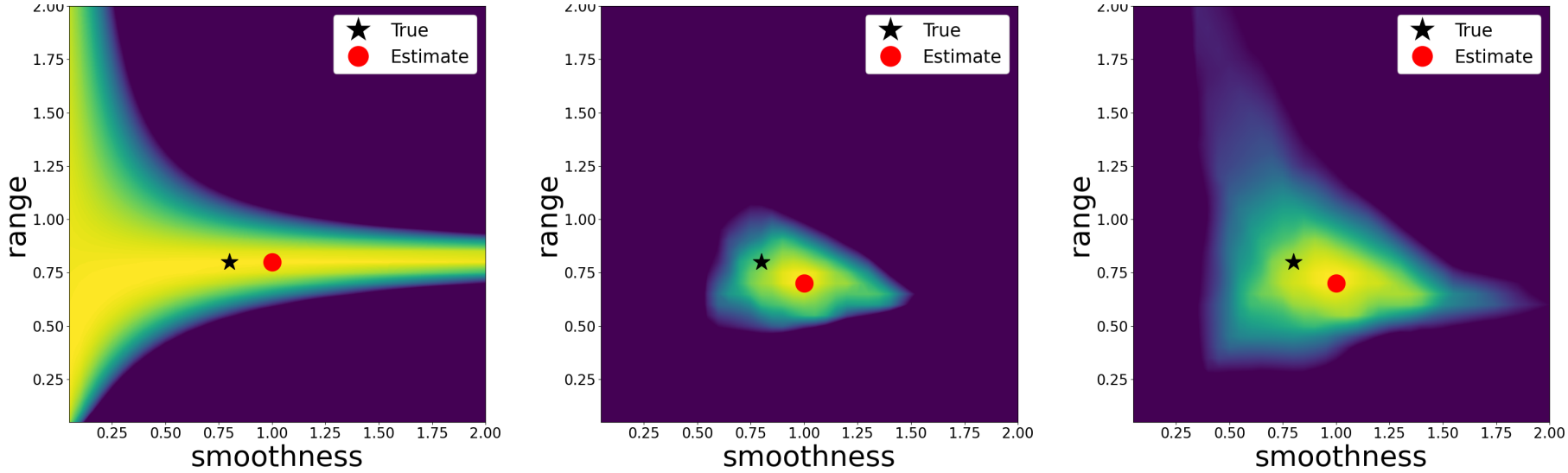


Figure: Pairwise log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for range $\lambda = 0.8$ and smoothness $\nu = 0.8$ with distance cut-off $\delta = 1$

Brown–Resnick Neural Likelihood Surfaces

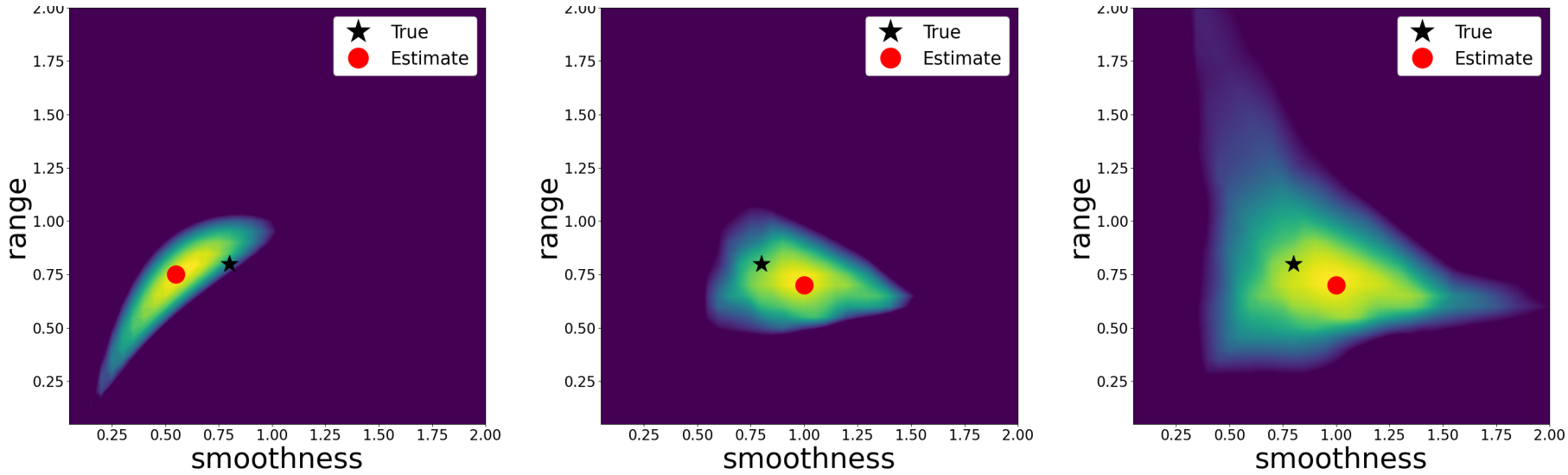


Figure: Pairwise log-likelihood (left) and log-neural likelihood before (middle) and after calibration (right) for range $\lambda = 0.8$ and smoothness $\nu = 0.8$ with distance cut-off $\delta = 2$

Brown–Resnick Approximate Confidence Regions

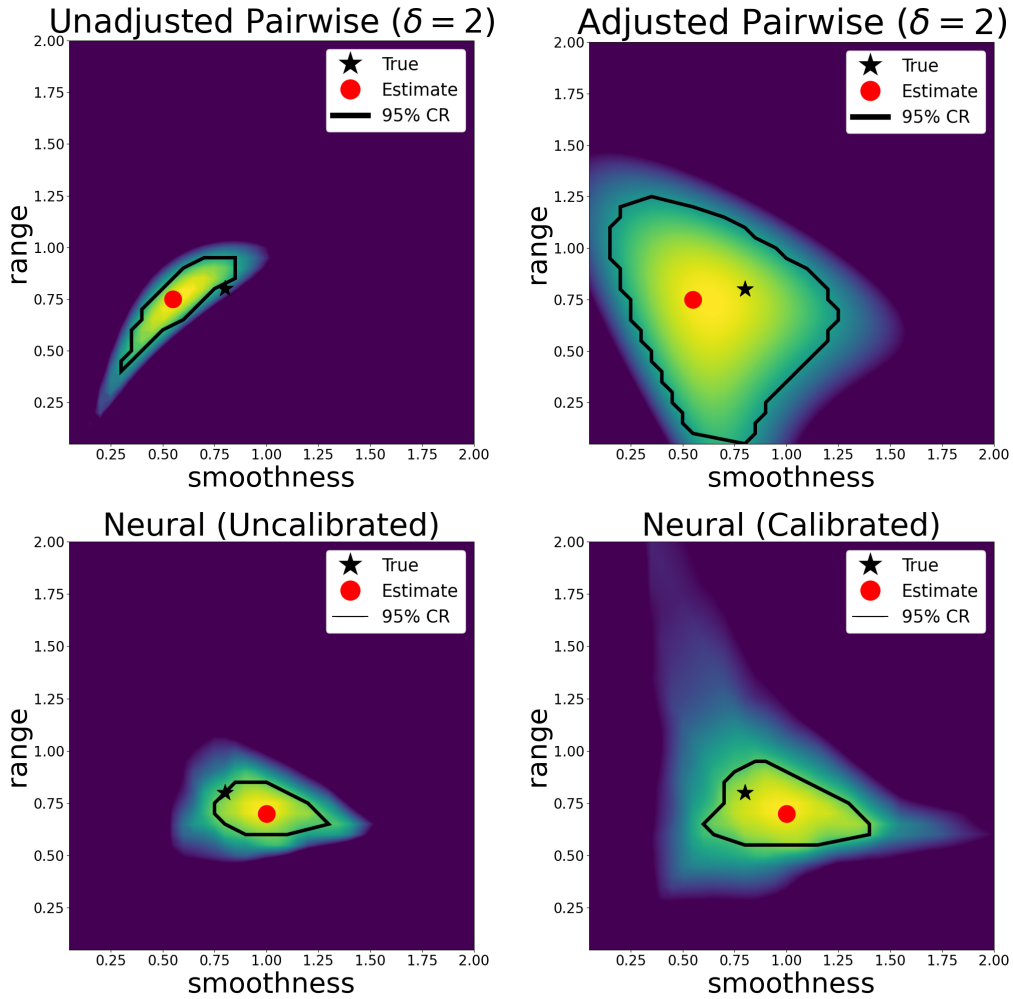


Figure: Unadjusted pairwise log-likelihood (top left), adjusted pairwise log-likelihood (top right), log-neural likelihood before (bottom left) and after calibration (bottom right) for range $\lambda = 0.8$ and smoothness $\nu = 0.8$ with distance cut-off $\delta = 2$ along with the corresponding 95% approximate confidence regions

Empirical Coverage for Brown–Resnick

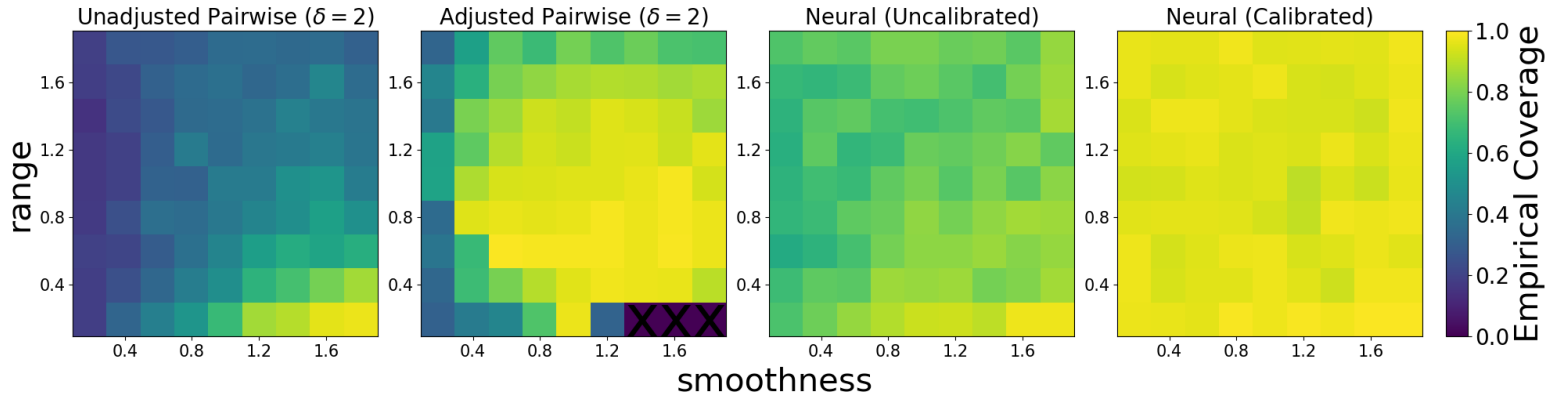


Figure: Empirical coverage

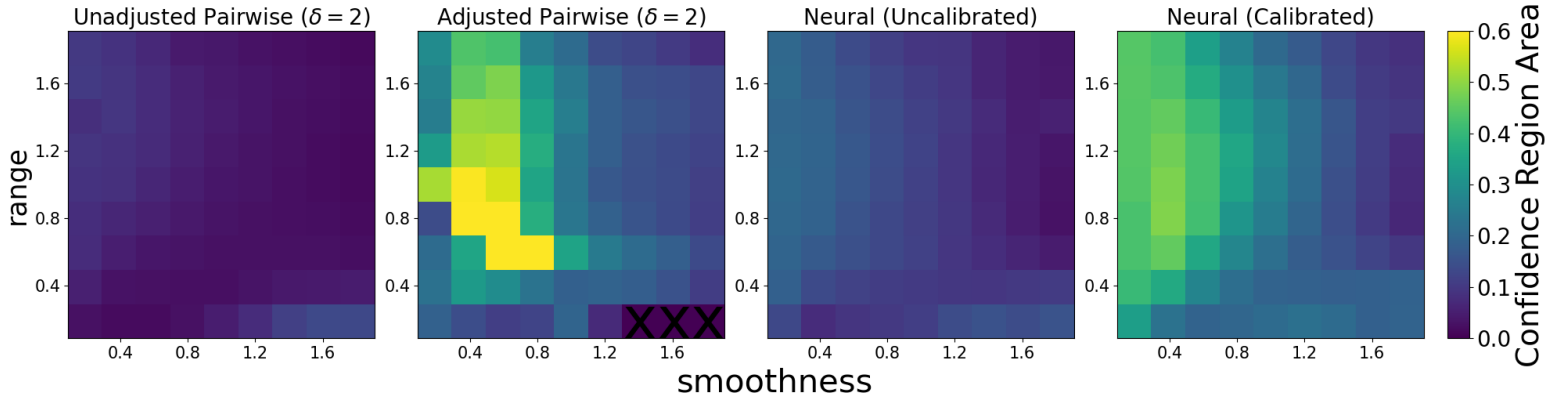


Figure: Confidence region area

Timing study for Brown–Resnick

Table: Time to produce neural and pairwise likelihood surfaces on a 40×40 grid over Θ for 50 realizations of a Brown–Resnick process on a 25×25 grid on spatial domain $[-10, 10] \times [-10, 10]$.

Type of surface and method	average (sec)	std. dev. (sec)
pairwise likelihood ($\delta = 1$)	5.05	0.34
pairwise likelihood ($\delta = 2$)	5.38	0.27
pairwise likelihood ($\delta = 5$)	5.86	0.28
pairwise likelihood ($\delta = 10$)	7.33	0.16
vectorized neural likelihood	**2.24**	0.13
unvectorized neural likelihood	14.39	0.03

From neural prediction to frequentist confidence sets

Let's assume we have trained an algorithm that gives us access to the conditional mean $\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]$ and the conditional (co)variance $\mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]$

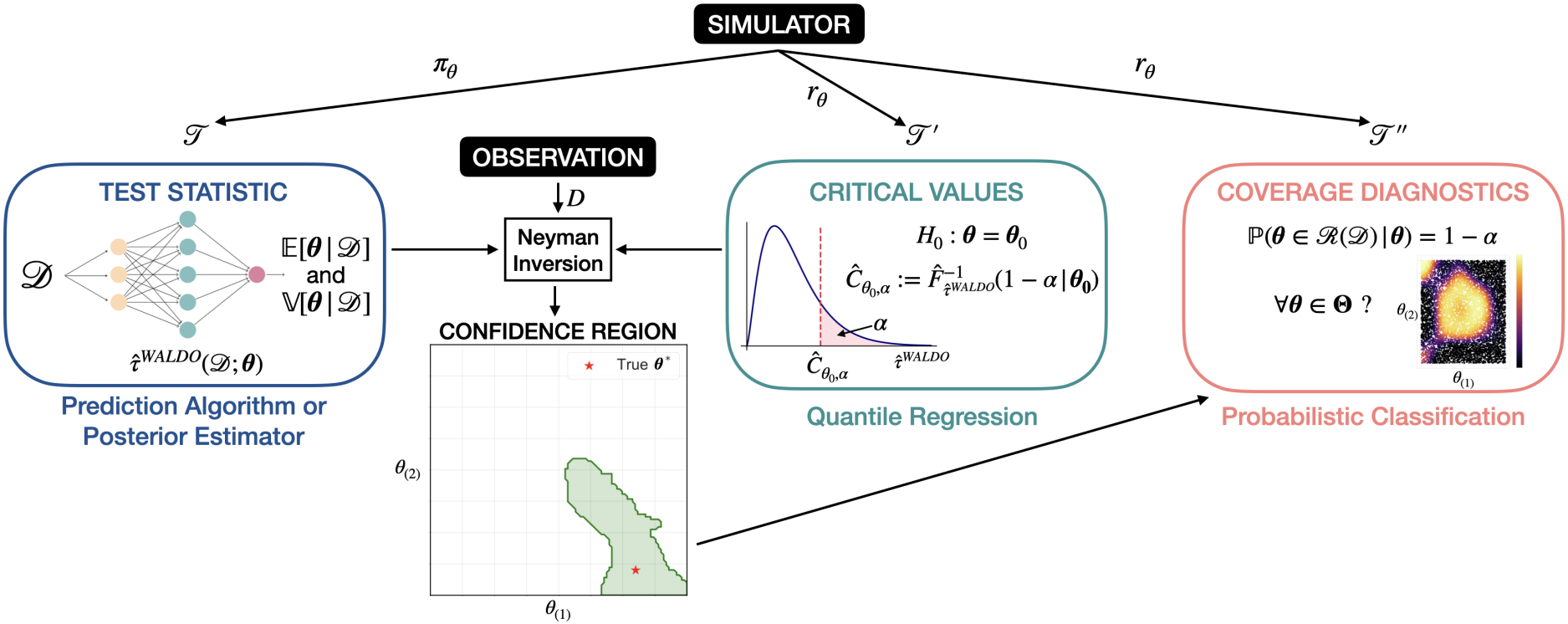
For example:

- Deep neural network trained to predict θ and $(\theta - \mathbb{E}[\theta|\mathcal{D}])^2$ for squared error loss
- Neural posterior of $\boldsymbol{\theta}$
- ...

A wide range of existing works in SBI either produce these quantities or can be easily adapted to produce these quantitites

Can we leverage these outputs to perform frequentist uncertainty quantification for $\boldsymbol{\theta}$?

Waldo



Schematic of WALDO (Masserano et al., 2023): We use the outputs of neural prediction to form a test statistic which is inverted to obtain a frequentist confidence set

This framework is a special case of LF2I (Likelihood-Free Frequentist Inference; Dalmasso et al. (2023)) developed by Ann Lee and her group at CMU

Waldo: test statistic

We want to perform the test

$$H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0 \quad \text{vs.} \quad H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$$

using $\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]$ and $\mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]$

Proposed test statistic:

$$\tau^{\text{WALDO}}(\mathcal{D}; \boldsymbol{\theta}_0) = (\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}] - \boldsymbol{\theta}_0)^T \mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]^{-1} (\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}] - \boldsymbol{\theta}_0)$$

This is analogous to the classical Wald test statistic

$$\tau^{\text{WALD}}(\mathcal{D}; \theta_0) = \frac{(\hat{\theta}^{\text{MLE}} - \theta_0)^2}{\mathbb{V}(\hat{\theta}^{\text{MLE}})},$$

hence the name WALDO

Waldo: critical values

To be able to invert the previous test, we need to know the null distribution of τ^{WALDO} for all $\boldsymbol{\theta}_0$

More specifically, to obtain a $1-\alpha$ confidence set for $\boldsymbol{\theta}$, we need to know the $1-\alpha$ quantiles of the null distributions

This is a quantile regression problem and can be solved by simulating data $\mathcal{D}$ under $\boldsymbol{\theta}_0$ for a sample of $\boldsymbol{\theta}_0$'s, computing the corresponding τ^{WALDO} and then regressing the quantiles of τ^{WALDO} on $\boldsymbol{\theta}_0$

$\Rightarrow$ Estimated critical values $\hat{C}_{\boldsymbol{\theta}_0,\alpha}$ for all $\boldsymbol{\theta}_0$

Waldo: Neyman inversion

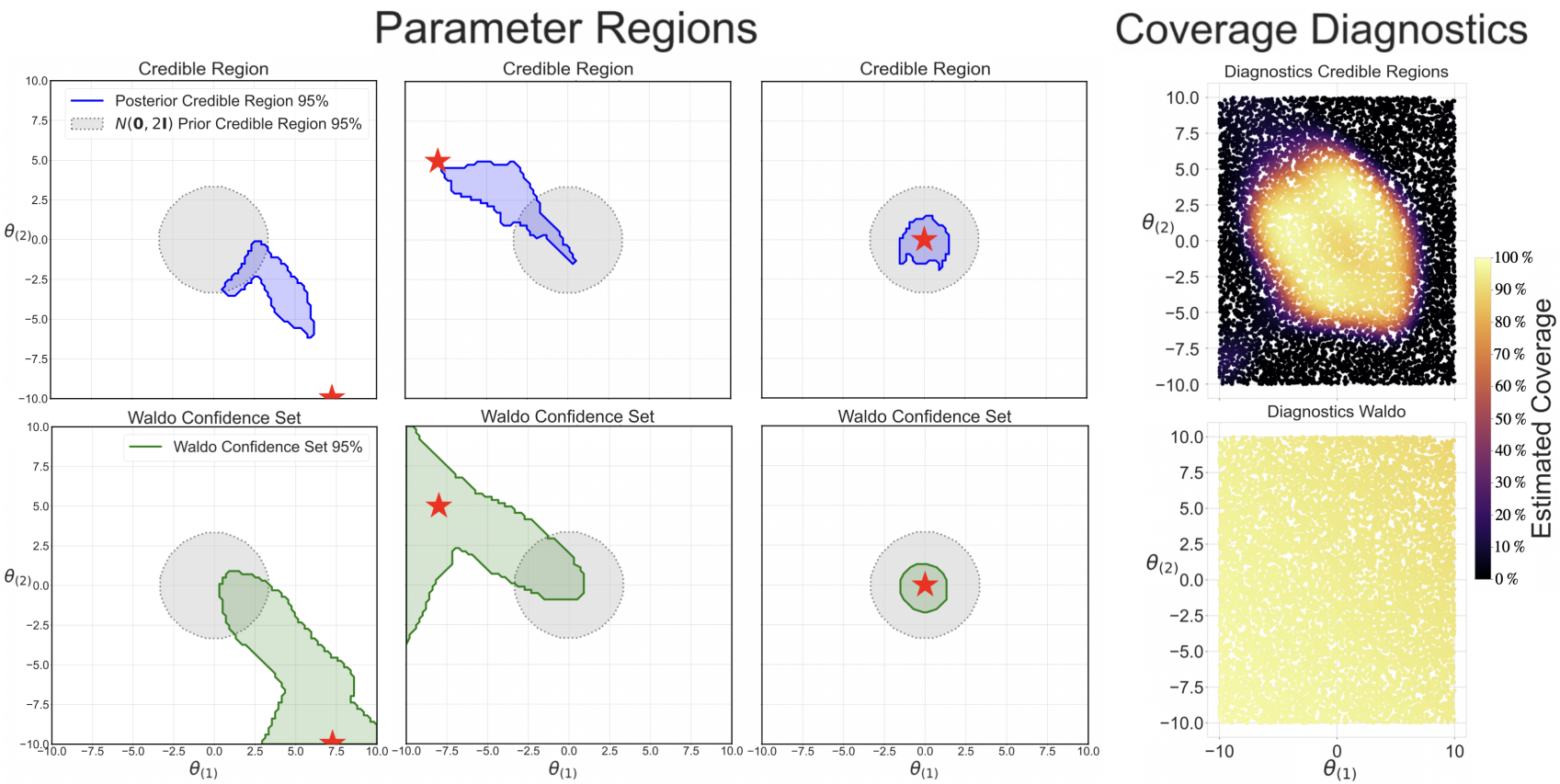
The $1 - \alpha$ confidence set $\mathcal{R}(D)$ contains all those parameter values that are not rejected at level α:

$$\mathcal{R}(D) = \{\boldsymbol{\theta}_0 \in \boldsymbol{\Theta} : \tau^{\textsc{Waldo}}(D; \boldsymbol{\theta}_0) \leq \hat{C}_{\boldsymbol{\theta}_0, \alpha}\}$$

This gives guaranteed finite-sample coverage as long as the quantiles $\hat{C}_{\boldsymbol{\theta}_0, \alpha}$ are estimated accurately enough

- This holds true no matter how well the original neural estimators $\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]$ and $\mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]$ were trained
- This also holds true no matter what prior was used for $\boldsymbol{\theta}$ (but good priors can lead to smaller confidence sets)

Waldo vs. neural posterior



Waldo applied to the neural posterior of $\boldsymbol{\theta}$ in the model $\mathcal{D}|\boldsymbol{\theta} \sim \frac{1}{2}\mathcal{N}(\boldsymbol{\theta}, \mathrm{I}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\theta}, 0.01 \cdot \mathrm{I})$ with prior $\boldsymbol{\theta} \sim \mathcal{N}(0, 2 \cdot \mathrm{I})$

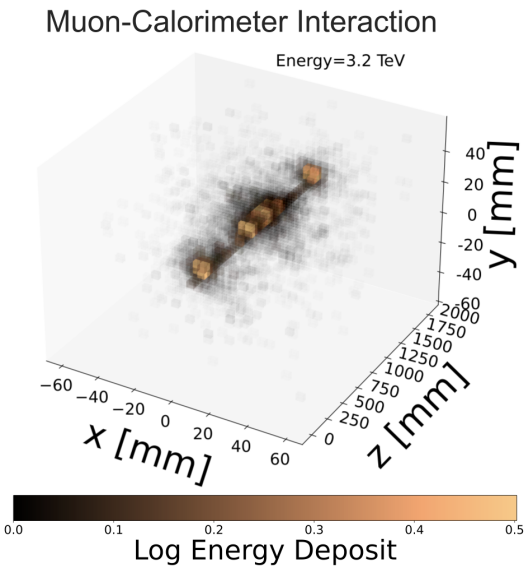
Waldo: Calorimetric muon energy reconstruction

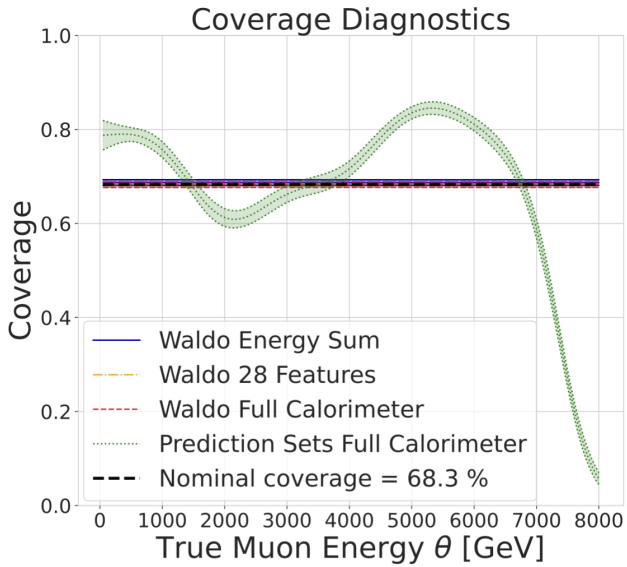
We apply WALDO to reconstructing muon energy from calorimetric energy deposits

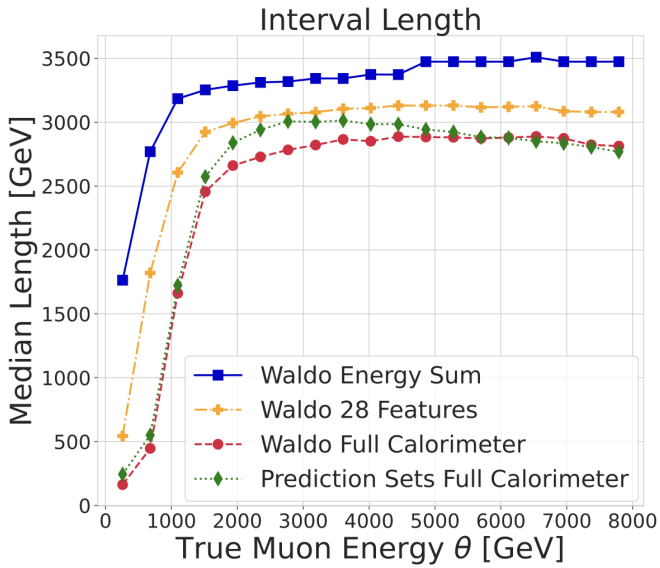
- The problem setting, data and CNN come from Kieseler et al. (2022); Dorigo et al. (2022) who used predictive inference for this problem

Here:

- θ = energy of the incoming muon
- $\mathcal{D}$ = calorimetric measurements (energy sum; 28 hand-chosen features; or raw calorimeter readout in 51200 dimensions)







Future directions: Model discrepancy

Current SBI methods assume access to a reliable simulator to generate $\boldsymbol{y} \sim F_{\boldsymbol{\theta}}$

But more often than not, the simulator is only approximately correct

This raises many questions:

- Are neural estimators / confidence sets robust against misspecification of $F_{\boldsymbol{\theta}}$?
  - How do the networks behave if evaluted for $\boldsymbol{y}$ outside the support of the training data?
  - Classical estimators (e.g., MLE) have known robustness properties against model misspecification; do any of these carry over to neural inference?
- How to perform SBI with misspecified simulators?
  - Parameterize the model discrepancy using nuisance parameters?
  - Model the discrepancy statistically?
  - Learn corrections from sidebands / control channels?

Future directions: High-dimensional parameter spaces

In principle, the parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^p$ could be arbitrarily high-dimensional in the previous methods

But, in practice, as the dimension p of $\boldsymbol{\theta}$ grows, at some point it becomes impossible to sample the parameter space and learn the $\boldsymbol{\theta}$ dependence accurately enough

It is not well-known up to which p does each SBI method provide realiable inferences

Future directions: High-dimensional parameter spaces

Even if a given method worked for large p, this would give us a very high-dimensional confidence set

What would we do with such a set?

- It's very difficult for humans to understand and process high-dimensional sets
- And any lower-dimensional projection of the high-dimensional confidence set would be an extremely conservative confidence set for the lower-dimensional parameter

Potential answer (Batlle et al., 2023): One could aim to invert for μ in tests of the form

$$H_0 : \boldsymbol{\theta} \in \Phi_\mu \cap \boldsymbol{\Theta} \quad \text{versus} \quad H_1 : \boldsymbol{\theta} \in \boldsymbol{\Theta} \setminus \Phi_\mu,$$

where

$$\Phi_\mu := \{\boldsymbol{\theta} : \varphi(\boldsymbol{\theta}) = \mu\} \subseteq \mathbb{R}^p$$

and φ is a quantity of interest (e.g., a given element of $\boldsymbol{\theta}$)

Conclusions and outlook

- SBI enables classical statistical inference in situations where the likelihood is either unavailable, intractable or computationally intensive
- A key tool is to use machine learning to handle the curse of dimensionality
- Many methods have been developed for learning likelihood functions, posteriors, likelihood ratios, confidence sets, etc., in the SBI setting
- A key feature of many of these is that they are *amortized*: there is an overhead cost from the training, but once the model is trained, it can be evaluated for an arbitrary number of new observations without retraining
- Many open problems remain, especially with regards to misspecified simulators and high-dimensional parameter spaces
- Papers covered in this talk:
  - J. Walchessen, A. Lenzi, and M. Kuusela, Neural likelihood surfaces for spatial processes with computationally intensive or intractable likelihoods, arXiv:2305.04634 [stat.ME], 2023
  - L. Masserano, T. Dorigo, R. Izbicki, M. Kuusela, and A. Lee. Simulator-based inference with Waldo: Confidence regions by leveraging prediction algorithms and posterior estimators for inverse problems. *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, PMLR, 2023
- Code for Waldo (and LF2I more generally) available on ▸ GitHub

References I

P. Batlle, P. Patil, M. Stanley, H. Owhadi, and M. Kuusela, Optimization-based frequentist confidence intervals for functionals in constrained inverse problems: Resolving the Burrus conjecture, arXiv:2310.02461 [math.ST], 2023.

S. Castruccio, R. Huser, and M. G. Genton, High-Order Composite Likelihood Inference for Max-Stable Distributions and Processes, *Journal of Computational and Graphical Statistics*, 25(4):1212–1229, 2016.

P. Chakravarti, M. Kuusela, J. Lei, and L. Wasserman, Model-independent detection of new physics signals using interpretable semi-supervised classifier tests, *The Annals of Applied Statistics*, 2023. To appear, preprint arXiv:2102.07679 [stat.AP].

K. Cranmer, J. Pavez, and G. Louppe, Approximating likelihood ratios with calibrated discriminative classifiers, 2015, preprint arXiv:1506.02169 [stat.AP].

N. Dalmasso, L. Masserano, D. Zhao, R. Izbicki, and A. B. Lee, Likelihood-free frequentist inference: Bridging classical statistics and machine learning for reliable simulator-based inference, arXiv:2107.03920 [stat.ML], 2023.

T. Dorigo, S. Guglielmini, J. Kieseler, L. Layer, and G. C. Strong, Deep regression of muon energy with a k-nearest neighbor algorithm, *arXiv preprint arXiv:2203.02841*, 2022.

References II

F. Gerber and D. Nychka, Fast Covariance Parameter Estimation of Spatial Gaussian Process Models using Neural Networks, *Stat*, 10(1):e382, 2021. doi: https://doi.org/10.1002/sta4.382. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.382`.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

M. Ho, M. M. Rau, M. Ntampaka, A. Farahi, H. Trac, and B. Póczos, A robust and efficient deep learning method for dynamical mass measurements of galaxy clusters, *The Astrophysical Journal*, 887(1):25, 2019.

J. Kieseler, G. C. Strong, F. Chiandotto, T. Dorigo, and L. Layer, Calorimetric measurement of multi-TeV muons via deep regression, *The European Physical Journal C*, 82(1):1–26, 2022.

M. Kuusela and M. L. Stein, Locally stationary spatio-temporal interpolation of Argo profiling float data, *Proceedings of the Royal Society A*, 474:20180400, 2018.

A. Lenzi, J. Bessac, J. Rudi, and M. L. Stein, Neural Networks for Parameter Estimation in Intractable Models, *Computational Statistics & Data Analysis*, 185:107762, 2023. doi: https://doi.org/10.1016/j.csda.2023.107762.

References III

T. Manole, P. Bryant, J. Alison, M. Kuusela, and L. Wasserman, Background modeling for double Higgs boson production: Density ratios and optimal transport, 2022, preprint arXiv:2208.02807 [stat.AP].

L. Masserano, T. Dorigo, R. Izbicki, M. Kuusela, and A. Lee. Simulator-based inference with WALDO: Confidence regions by leveraging prediction algorithms and posterior estimators for inverse problems. In F. Ruiz, J. Dy, and J.-W. van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 2960–2974. PMLR, 25–27 Apr 2023.

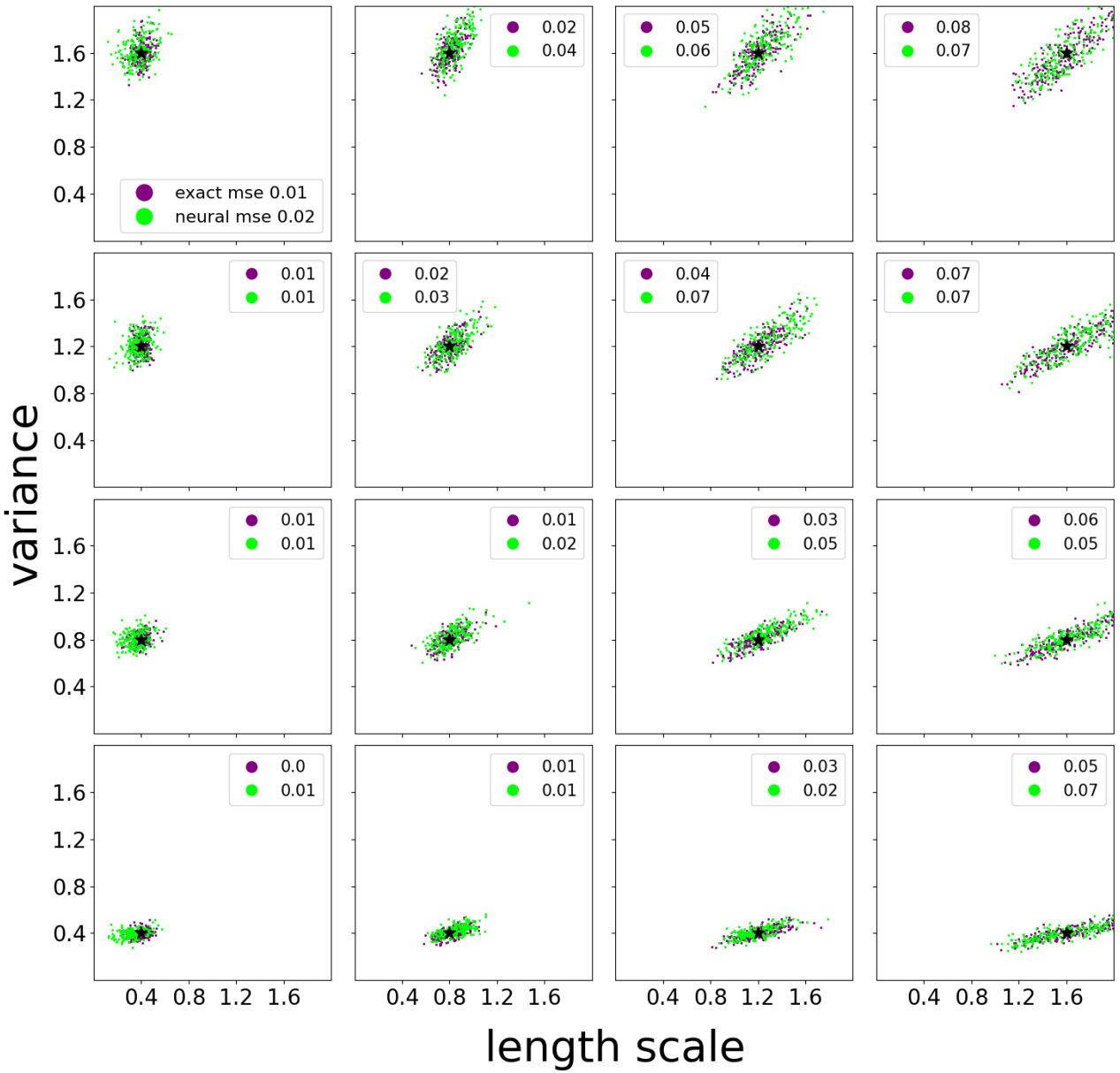
J. Platt, Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

M. Sainsbury-Dale, A. Zammit-Mangion, and R. Huser, Neural Point Estimation for Fast Optimal Likelihood-Free Inference, arXiv:2208.12942, 2023.

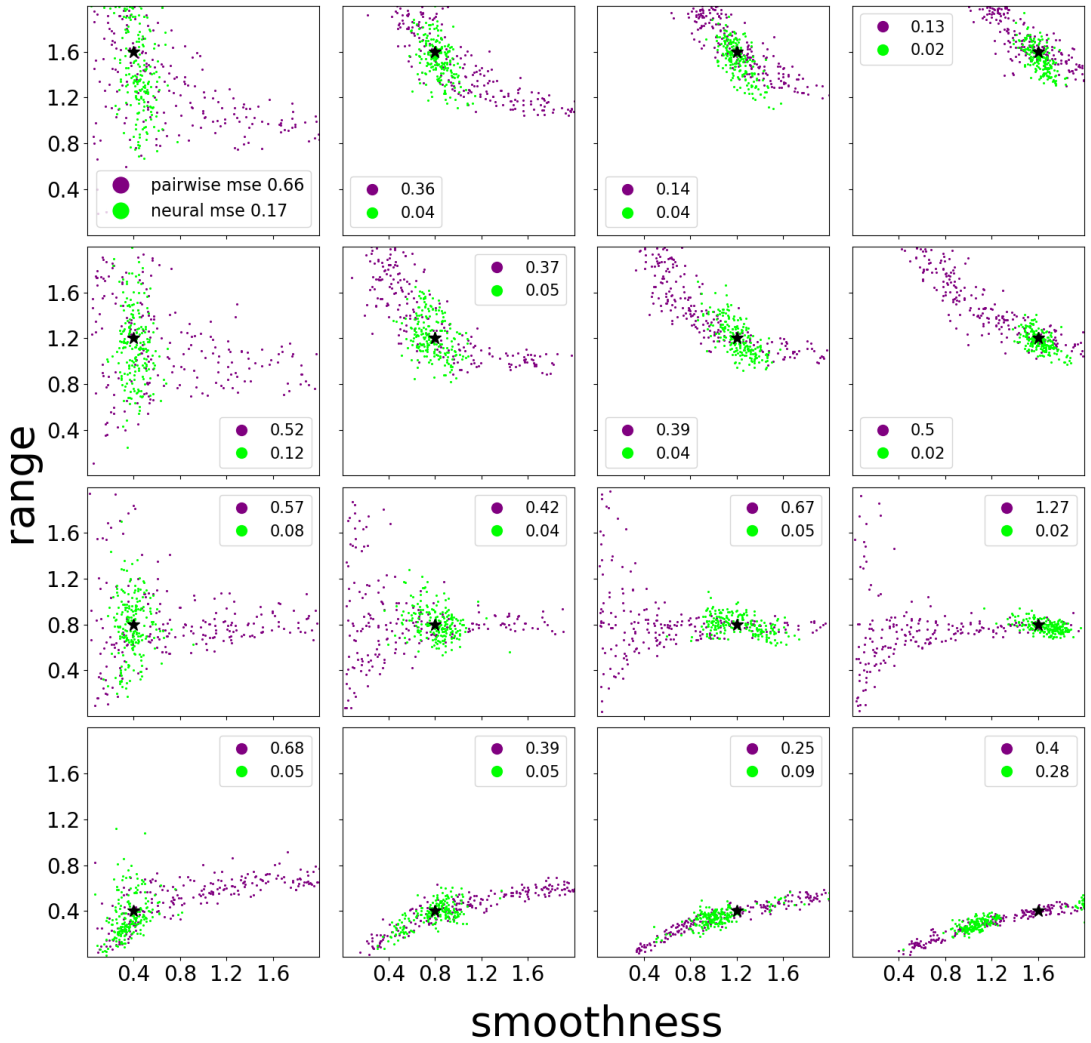
J. Walchessen, A. Lenzi, and M. Kuusela, Neural likelihood surfaces for spatial processes with computationally intensive or intractable likelihoods, arXiv:2305.04634 [stat.ME], 2023.

Backup

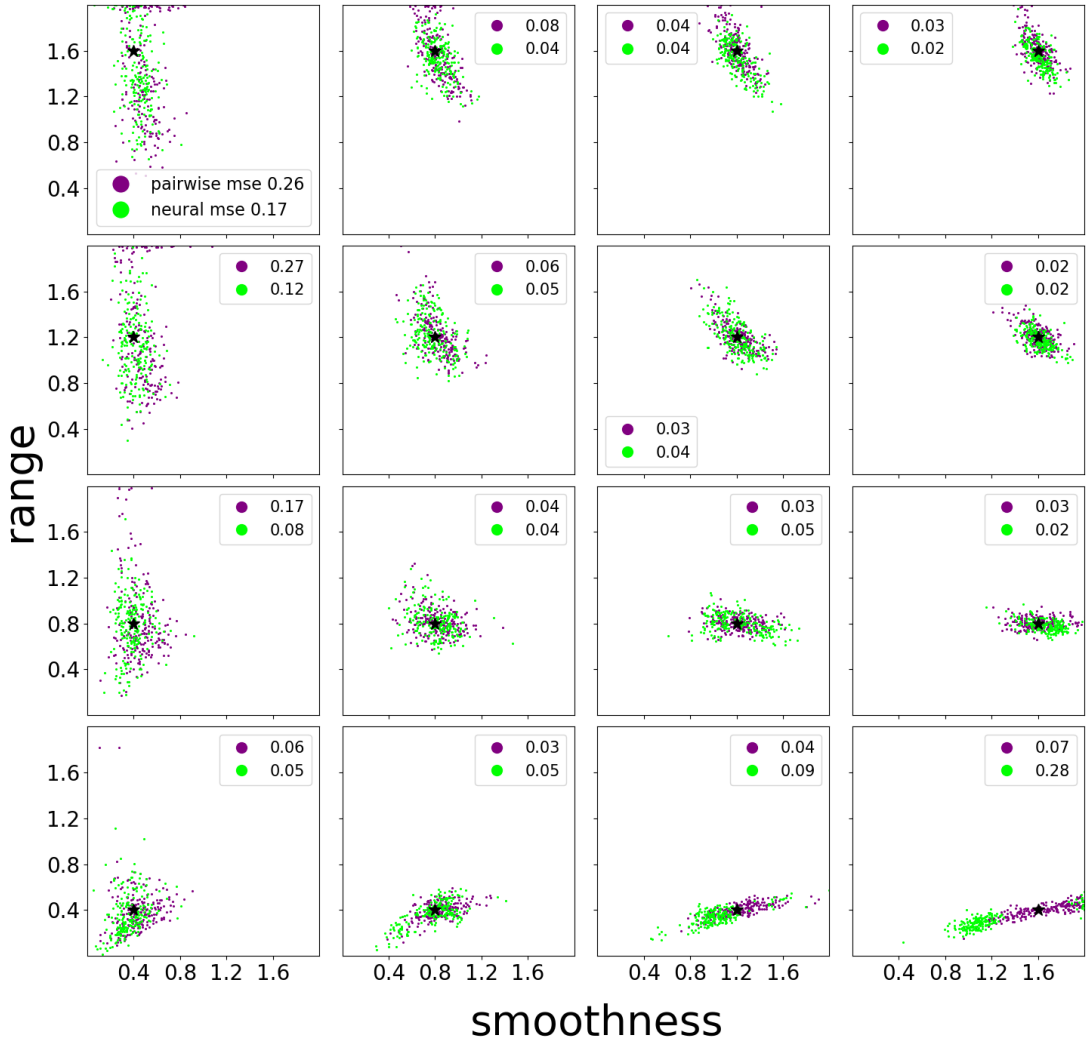
Neural vs Exact Likelihood Estimates



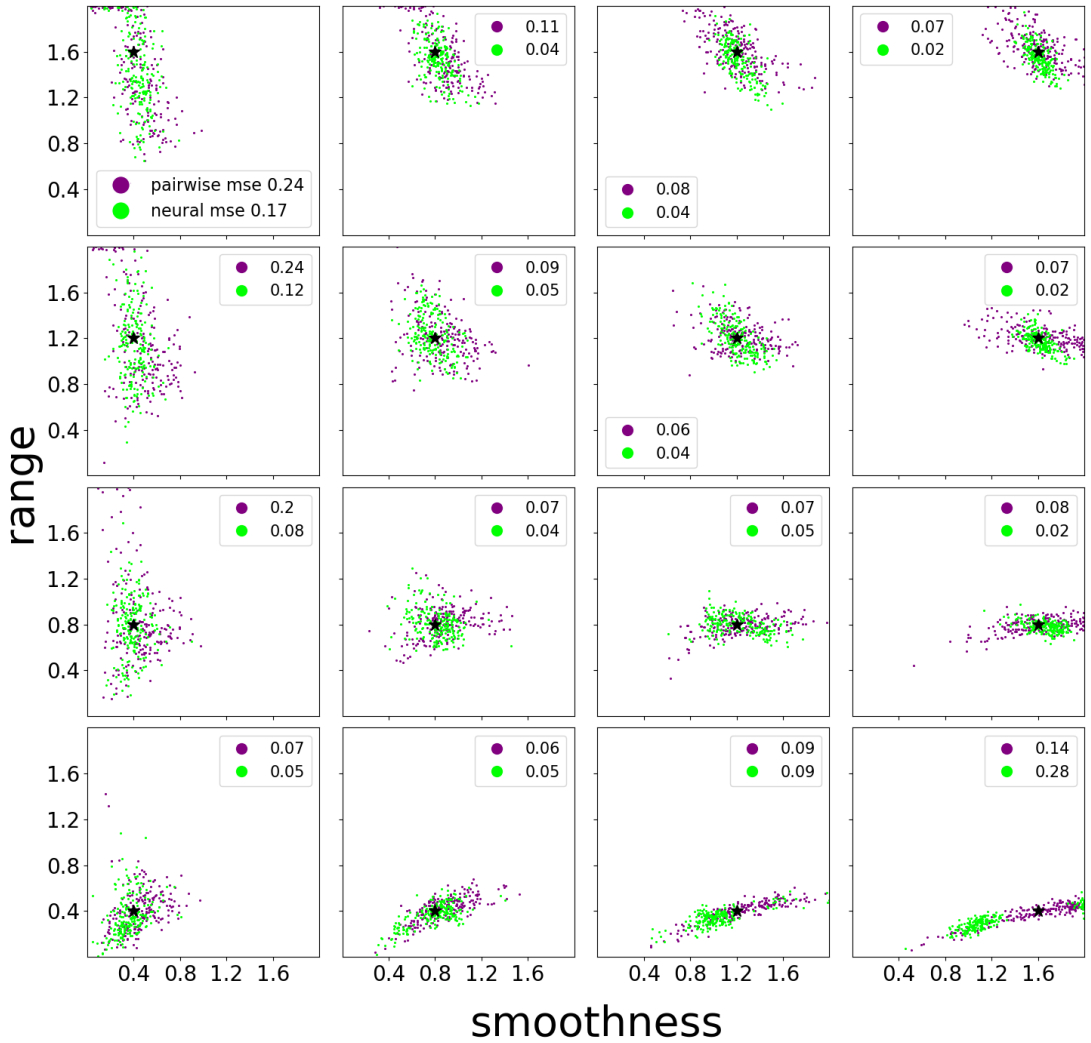
Neural vs Pairwise Likelihood Estimates ($\delta = 1$)



Neural vs Pairwise Likelihood Estimates ($\delta = 2$)



Neural vs Pairwise Likelihood Estimates ($\delta = 5$)



Motivation: Large and complex spatial data

Likelihood-based parameter inference for large and complex spatial data tends to be computationally intensive or wholly intractable

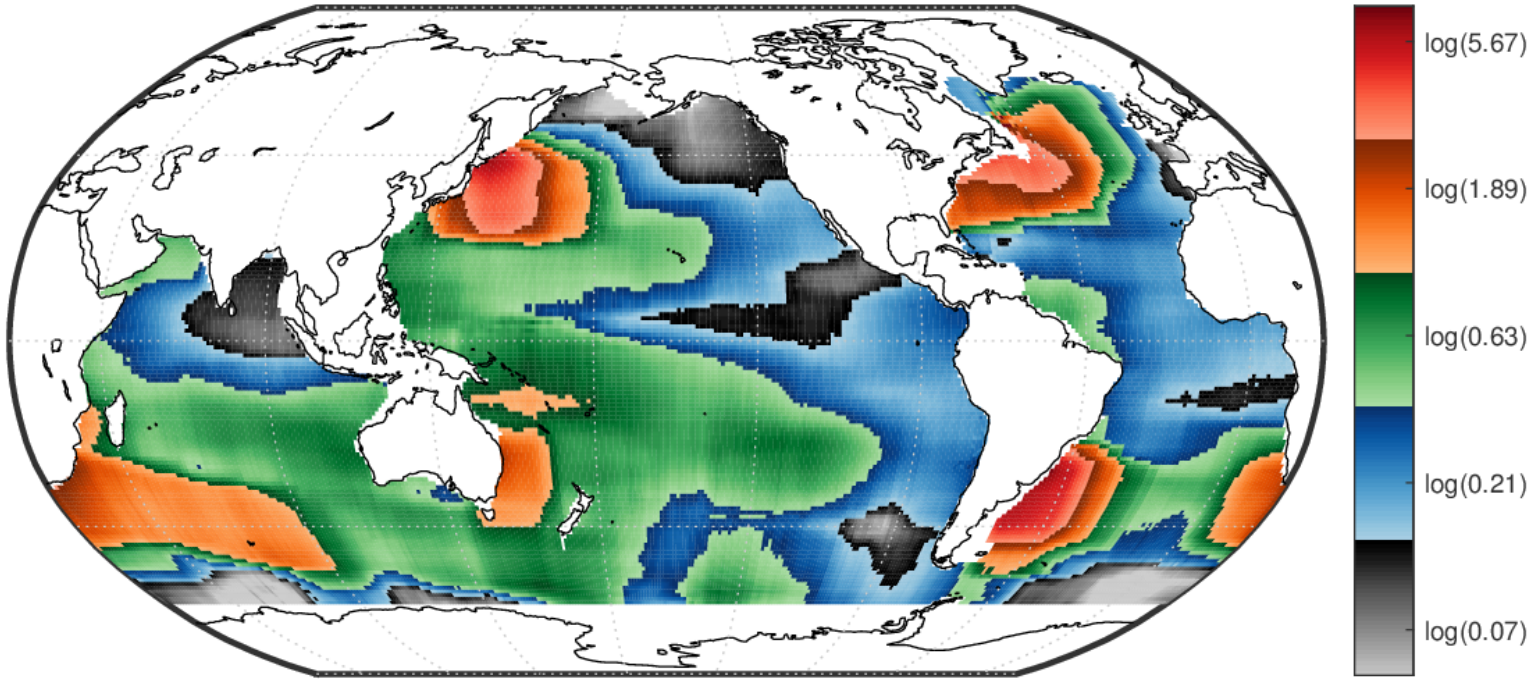
Likelihood computations for Gaussian processes scale as $\mathcal{O}(n^3)$
$\Rightarrow$ slow for large n

Many non-Gaussian spatial processes have intractable likelihood functions:

- GP + Student-t nugget
- GP + spatial ARCH
- Max-stable processes
- ...

Lots of past research on approximate likelihoods for situations like these (low-rank approximations, covariance tapering, Vecchia approximation, Laplace approximation, composite likelihood,...)

Motivation: Large and complex spatial data



Estimated variance at 300 m depth based on fitting locally stationary Gaussian processes (Kuusela and Stein, 2018)

This figure contains $\sim$30 000 numerically computed MLEs

Takes several hours to compute even after parallelization on HPC

Classifier architecture

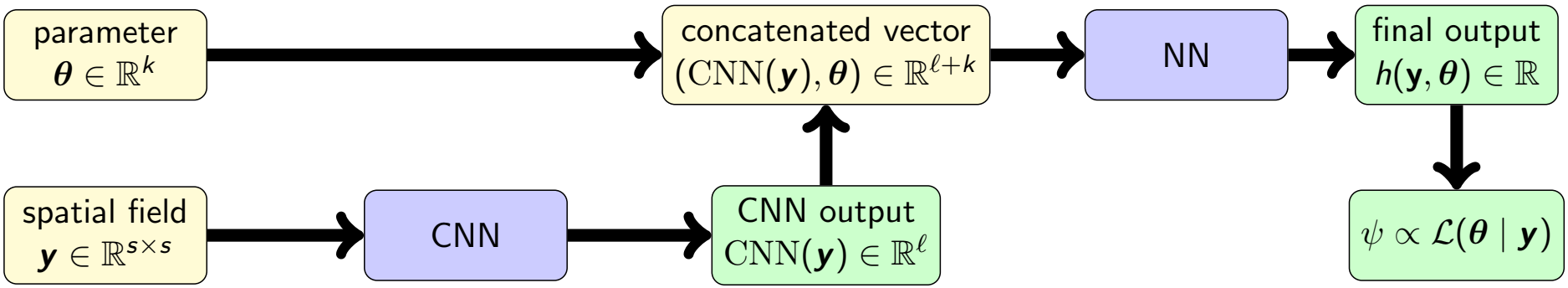


Figure: The basic structure of our neural network. The convolutional neural network (CNN) part is similar to the CNN used in Lenzi et al. (2023).

Adjusted pairwise likelihood

Using

$$\{\boldsymbol{\theta} \in \Theta^L \mid 2\big(\log \mathcal{L}(\hat{\boldsymbol{\theta}} \mid \boldsymbol{y}) - \log \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{y})\big) \leq \chi^2_{k,1-\alpha}\}$$

as an approximate confidence region is justified as long as

$$2\big(\log \mathcal{L}(\hat{\boldsymbol{\theta}} \mid \boldsymbol{y}) - \log \mathcal{L}(\boldsymbol{\theta}^* \mid \boldsymbol{y})\big) \overset{a}{\sim} \chi^2_k,$$

where $\boldsymbol{\theta}^*$ the true value of $\boldsymbol{\theta}$

This is in most cases true for the exact likelihood and the exact MLE, but not for the pairwise likelihood

To recover the approximate χ^2_k distribution, it is necessary to adjust the pairwise log-likelihood function $\ell_{\mathrm{pw}}(\boldsymbol{\theta}) = \log \mathcal{L}_{\mathrm{pw}}\big(\boldsymbol{\theta} \mid \boldsymbol{y}\big)$ as follows:

$$\ell^{\mathrm{a}}_{\mathrm{pw}}(\boldsymbol{\theta}) = \ell_{\mathrm{pw}}(\hat{\boldsymbol{\theta}}_{\mathrm{pw}} + \boldsymbol{C}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\mathrm{pw}})),$$

where $\boldsymbol{C}$ is a matrix involving the Hessian of ℓ_{pw} and the asymptotic covariance matrix of $\hat{\boldsymbol{\theta}}_{\mathrm{pw}}$

Training data via permutation

First Class

$$\begin{bmatrix} (\boldsymbol{y}_{1,1}, \boldsymbol{\theta}_1) & \cdots & (\boldsymbol{y}_{1,n}, \boldsymbol{\theta}_1) \\ \vdots & \ddots & \vdots \\ (\boldsymbol{y}_{m,1}, \boldsymbol{\theta}_m) & \cdots & (\boldsymbol{y}_{m,n}, \boldsymbol{\theta}_m) \end{bmatrix}$$

Second Class

$$\begin{bmatrix} (\boldsymbol{y}_{1,1}, \boldsymbol{\theta}_{\pi_1(1)}) & \cdots & (\boldsymbol{y}_{1,n}, \boldsymbol{\theta}_{\pi_n(1)}) \\ \vdots & \ddots & \vdots \\ (\boldsymbol{y}_{m,1}, \boldsymbol{\theta}_{\pi_1(m)}) & \cdots & (\boldsymbol{y}_{m,n}, \boldsymbol{\theta}_{\pi_n(m)}) \end{bmatrix}$$

Calibration with Platt scaling

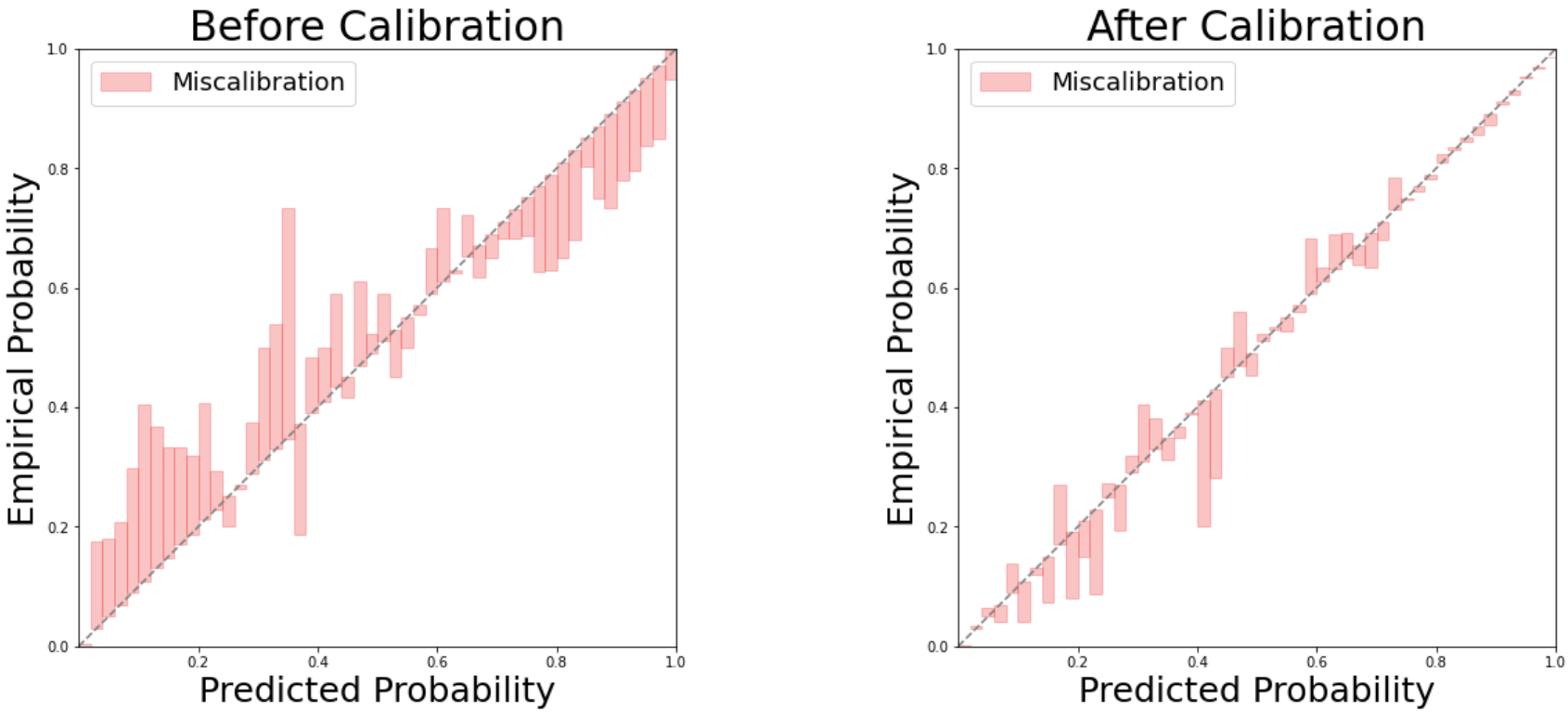


Figure: Reliability diagrams (empirical probability vs. predicted probability) before calibration (left) and after calibration (right) for a Brown–Resnick process.