Simulation-based inference (sbi) for pulsar population synthesis

Dr. Vanessa Graber (graber@ice.csic.es)

in collaboration with Michele Ronchi, Celsa Pardo Araujo, and Nanda Rea

Cassiopeia A supernova remnant
(credit: NASA/CXC/SAO)
Outline

- Neutron stars
- Pulsar population synthesis
- Machine learning and sbi
- Inference results
- Outlook

Cassiopeia A supernova remnant
(credit: NASA/CXC/SAO)
Neutron-star formation

- Neutron stars are one of three types of compact remnants, created during the final stages of stellar evolution.

- When a massive star of 8 - 25 solar masses runs out of fuel, it collapses under its own gravitational attraction and explodes in a supernova.

- During the collapse, electron capture processes \((p + e^- \rightarrow n + \nu_e)\) produce (a lot of) neutrons.

Snapshot of a 3D core-collapse supernova simulation (Mösta et al., 2014)

**Snapshot of a 3D core-collapse supernova simulation (Mösta et al., 2014)**

- **mass:** 1.2 - 2.1 M\(\odot\)
- **radius:** 9 - 15 km
- **density:** \(10^{15}\) g/cm\(^3\)

graber@ice.csic.es
Lighthouse radiation

- Neutron stars have **extreme magnetic fields** between $10^8$ - $10^{15}$ G. For comparison, the Earth’s magnetic field is 0.5 G.

- Because rotation and magnetic axes are misaligned, neutron stars emit radio beams **like a lighthouse**.

- These pulses can be observed with radio telescopes. This is how neutron stars were first detected and why we call them **pulsars**.

Dame Jocelyn Bell Burnell in front of her radio telescope in Cambridge, UK.

graber@ice.csic.es
The neutron-star zoo

- Pulsars are **very precise clocks** and we time their pulses to **measure rotation periods** $P$ and derivatives $\dot{P}$.

- We now observe neutron stars as pulsars **across** the electromagnetic spectrum.

  - **~ 3,000 pulsars** are known to date

- Grouping neutron stars in the **$P\dot{P}$-plane** according to their observed properties serves as a diagnostic tool to **identify different neutron-star classes**.

---

Period period-derivative plane for the pulsar population. Data taken from the ATNF Pulsar Catalogue (Manchester et al., 2005)
The neutron-star zoo

● Pulsars are **very precise clocks** and we time their pulses to **measure rotation periods** $P$ and derivatives $\dot{P}$.

● We now observe neutron stars as pulsars **across** the **electromagnetic spectrum**.

Focus on the isolated radio-pulsar population

~ **3,000 pulsars** are known to date

● Grouping neutron stars in the **$PP$-plane** according to their observed properties serves as a diagnostic tool to **identify different neutron-star classes**.

Period period-derivative plane for the pulsar population. Data taken from the ATNF Pulsar Catalogue (Manchester et al., 2005)
Outline

- Neutron stars
- Pulsar population synthesis
- Machine learning and sib
- Inference results
- Outlook

Cassiopeia A supernova remnant
(credit: NASA/CXC/SAO)
General idea

- We can estimate the total number of neutron stars in our Galaxy

\[
\text{CC supernova rate: } \sim 2 \text{ per century} \times \text{Galaxy age: } \sim 13.6 \text{ billion years} = \text{NS number: } \sim 2.8 \times 10^8
\]

- We only detect a very small fraction of all neutron stars. Population synthesis bridges this gap focusing on the full population of neutron stars (e.g. Faucher-Giguère & Kaspi 2006, Lorimer et al. 2006, Gullón et al. 2014, Cieślar et al. 2020):

  - model birth properties with Monte-Carlo approach
  - evolve properties forward in time
  - apply filters to mimic observational biases/limits
  - compare mock simulations to observations to constrain input
Goals

- Population synthesis allows us to constrain the natal properties of neutron stars and their birth rates.

- This is for example relevant for:
  - Massive star evolution
  - Gamma-ray bursts
  - Fast-radio bursts
  - Peculiar supernovae

- We can also learn about evolutionary links between different neutron-star classes (e.g., Viganó et al., 2013). This is important because estimates for the Galactic core-collapse supernova rate are insufficient for to explain the independent formation of different classes of pulsars (Keane & Kramer, 2008).

<table>
<thead>
<tr>
<th>PSRs</th>
<th>RRATs</th>
<th>XDINSs</th>
<th>Magnetars</th>
<th>Total</th>
<th>CCSN rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8 ± 0.5</td>
<td>5.6^{+4.3}_{-3.3}</td>
<td>2.1 ± 1.0</td>
<td>0.3^{+1.2}_{-0.2}</td>
<td>10.8^{+7.0}_{-5.0}</td>
<td>1.9 ± 1.1</td>
</tr>
<tr>
<td>1.4 ± 0.2</td>
<td>2.8^{+1.6}_{-1.6}</td>
<td>2.1 ± 1.0</td>
<td>0.3^{+1.2}_{-0.2}</td>
<td>6.6^{+4.0}_{-3.0}</td>
<td>1.9 ± 1.1</td>
</tr>
<tr>
<td>1.1 ± 0.2</td>
<td>2.2^{+1.7}_{-1.3}</td>
<td>2.1 ± 1.0</td>
<td>0.3^{+1.2}_{-0.2}</td>
<td>5.7^{+2.7}_{-2.7}</td>
<td>1.9 ± 1.1</td>
</tr>
<tr>
<td>1.6 ± 0.3</td>
<td>3.2^{+2.5}_{-1.9}</td>
<td>2.1 ± 1.0</td>
<td>0.3^{+1.2}_{-0.2}</td>
<td>7.2^{+5.0}_{-3.4}</td>
<td>1.9 ± 1.1</td>
</tr>
<tr>
<td>1.1 ± 0.2</td>
<td>2.2^{+1.7}_{-1.3}</td>
<td>2.1 ± 1.0</td>
<td>0.3^{+1.2}_{-0.2}</td>
<td>5.7^{+2.7}_{-2.7}</td>
<td>1.9 ± 1.1</td>
</tr>
</tbody>
</table>
Dynamical evolution I

- Neutron stars are born in star-forming regions, i.e., in the Galactic disk along the Milky Way’s spiral arms, and receive kicks during the supernova explosions.

- We make the following assumptions:
  - Spiral-arm model (Yao et al., 2017) plus rigid rotation with $T = 250$ Myr
  - Exponential disk model with scale height $h_c$ (Wainscoat et al., 1992)
  - Single-component Maxwell kick-velocity distribution with dispersion $\sigma_k$ (Hobbs et al., 2005)
  - Galactic potential (Marchetti et al., 2019)

For Monte-Carlo approach, we vary two uncertain parameters $h_c$ and $\sigma_k$. 

$\mathcal{P}(z) = \frac{1}{h_c} e^{-\frac{|z|}{h_c}}$

$\mathcal{P}(v_k) = \sqrt{\frac{2 v_k^2}{\pi \sigma_k^3}} e^{-\frac{v_k^2}{2\sigma_k^2}}$
Dynamical evolution II

- For our Galactic model $\Phi_{MW}$, we evolve the stars’ position & velocity by solving Newtonian equations of motion in cylindrical galactocentric coordinates:

$$\ddot{r} = -\nabla \Phi_{MW}$$

Galactic evolution tracks for $h_c = 0.18$ kpc, $\sigma = 265$ km/s.
Magneto-rotational evolution I

- The neutron-star magnetosphere exerts a *torque onto the star*. This causes *spin-down* and alignment of the magnetic and rotation axes.

- Neutron star *magnetic fields decay* due to the Hall effect and Ohmic dissipation in the outer stellar layer (crust) (e.g., Viganó et al., 2013 & 2021).

- We make the following assumptions:
  - Initial periods follow a log-normal with $\mu_P$ and $\sigma_P$ (Igoshev et al., 2022)
  - Initial fields follow a log-normal with $\mu_B$ and $\sigma_B$ (Gullón et al., 2014)
  - Above $\tau \sim 10^6$ yr, field decay follows a power-law with $B(t) \sim B_0 (1 + t/\tau)^\alpha$.

$$P(\log P_0) = \frac{1}{\sqrt{2\pi}\sigma_P} \exp\left(-\frac{(\log P_0 - \mu_P)^2}{2\sigma_P^2}\right)$$

Here, we vary the five uncertain parameters $\mu_P$, $\mu_B$, $\sigma_P$, $\sigma_B$ and $\alpha$. 

graber@ice.csic.es
To model the magneto-rotational evolution, we numerically solve two coupled ordinary differential equations for the period and the misalignment angle (Aguilera et al., 2008; Philippov et al. 2014).

We use results from 2D magneto-thermal simulations to determine the evolution of the magnetic field.

This allows us to follow the stars’ P and P evolution in the PP-plane.

PP evolution tracks for $\mu_p = -0.6$, $\sigma_p = 0.3$, $\mu_B = 13.25$ and $\sigma_B = 0.75$. 

graber@ice.csic.es
Radio emission and detection

- The stars’ **rotational energy** \( E_{\text{rot}} \) is converted into coherent radio emission. We assume that the corresponding **radio luminosity** \( L_{\text{radio}} \) is proportional to the loss of \( E_{\text{rot}} \) (Faucher-Giguère & Kaspi, 2006; Gullón et al., 2014). \( L_0 \) is taken from observations.

- As **emission is beamed**, ~ 90% of pulsars do not point towards us. For those intercepting our line of sight, compute **radio flux** \( S_{\text{radio}} \) & **pulse width** \( W \).

\[
L_{\text{radio}} = L_0 \left( \frac{\dot{P}}{P^3} \right)^{1/2} \propto \dot{E}_{\text{dot}}^{1/2}
\]

\[
S_{\text{radio}} = \frac{L_{\text{radio}}}{\Omega_{\text{beam}} d^2}
\]

A pulsar counts as detected, if it **exceeds the sensitivity threshold** for a survey recorded with a specific radio telescope.
Three pulsar surveys

- We compare our simulated populations with three surveys from Murriyang (the Parkes Radio Telescope):
  - Parkes Multibeam Pulsar Survey (PMPS): 1,009 isolated pulsars
  - Swinburne Parkes Multibeam Pulsar Survey (SMPS): 218 isolated pulsars
  - High Time Resolution Universe Survey (HTRU): 1,023 isolated pulsars

Can we constrain birth properties by looking at a current snapshot of the pulsar population?
Outline

- Neutron stars
- Pulsar population synthesis
- Machine learning and sbi
- Inference results
- Outlook

Cassiopeia A supernova remnant (credit: NASA/CXC/SAO)
graber@ice.csic.es
Comparing models and data

- Comparing observations to models and **constraining regions of the parameter space** that are **most probable given the data** is fundamental to many fields of science.

- Pulsar population synthesis is complex and has **many free parameters**. To compare synthetic simulations with observations, people have
  - Randomly sampled and then optimised ‘by eye’ (e.g., Gonthier et al., 2007)
  - Compared distributions of individual parameters using $\chi^2$- and KS-tests (e.g., Narayan & Ostriker, 1990; Faucher-Giguère & Kaspi, 2006)
  - Used annealing methods for optimisation (Gullón et al., 2014)
  - Performed Bayesian inference for simplified models (Cieślar et al., 2020)

These methods do not scale well and are **difficult to use** with the **multi-dimensional data** produced in population synthesis.
Deep learning

- Deep Learning is a subfield of Artificial Intelligence and Machine Learning. It focuses on using multi-layered neural networks to learn from large datasets. Different to classical ML approaches, deep learning does **not require external feature engineering**.

- Recognising features in a hierarchical way allows deep neural networks to model **complex non-linear relationships** for large input data. This makes deep learning powerful when **working with unstructured data such as images**, where the number of features / pixel can easily exceed millions.

graber@ice.csic.es

Credit: www.bigdata-insider.de (top), Acheron Analytics (bottom)
A neural network is composed of layers, which represent stacks of neurons (objects holding a single numerical value). Each layer encodes a simplified representation of the input data.

A deep-learning algorithm learns more and more about the input as the data is passed through successive network layers.

The Multilayer Perceptron is the simplest set-up where input and output are fully connected. In a CNN, not all nodes are connected, which reduces the number of trainable parameters and allows more flexibility for training.
Convolutional neural networks (CNNs)

- A neural network is composed of layers, which represent stacks of neurons (objects holding a single numerical value). Each layer encodes a simplified representation of the input data.

- A deep-learning algorithm learns more and more about the input as the data is passed through successive network layers.

- The Multilayer Perceptron is the simplest set-up where input and output are fully connected. In a CNN, not all nodes are connected, which reduces the number of trainable parameters and allows more flexibility for training.
Convolutional and max pooling layers

- Besides fully connected layers, CNNs are composed of two types of filters:

Convolutional filters

These filters recognise features, such as detecting edges of an object in an image.

Max-pooling layers

These filters extract the most relevant features, helping to speed up the training process.

Proof of concept study I

- In Ronchi et al. (2021), we focused on the dynamical evolution and simulated a database of 128 x 128 (=16,384) synthetic neutron-star populations.
  
  Vary scale height $h_c$ in range [0.02-2] kpc
  
  Vary dispersion $\sigma_k$ of kick distribution between [1-700] km/s

- We perform supervised ML and train a CNN to extract labels $h_c$ and $\sigma$ from position / velocity maps:
Proof of concept study II

- **Training info:** We use the root mean square error as the loss function and validation metric, Kaiming initialisation, Adam for gradient-descent optimisation, and apply a 80 / 20% split of the full dataset for training and validation.

- The **CNN recovers the input values** very well. To visualise this, we can look at the **relative error between target and predicted labels**.
Proof of concept study II

- **Training info:** We use the root mean square error as the loss function and validation metric, Kaiming initialisation, Adam for gradient-descent optimisation, and apply a 80 / 20% split of the full dataset for training and validation.

- The **CNN recovers the input values** very well. To visualise this, we can look at the relative error between target and predicted labels.

  ![Graphs showing relative error](image)

  MRE = 0.061 gives ME = 0.01 kpc at $h_c = 0.18$ kpc

  MRE = 0.039 gives ME = 10 km/s at $\sigma_k = 265$ km/s

We did **not include observational biases** and assumed all simulated stars are detectable! **216 pulsars have measured proper motions**, insufficient for this precision.
Our initial study focused on **deducing point estimates**. However, we often do not require exact estimates but **knowledge of probable regions** is sufficient.

This is where **Bayesian inference** comes in: based on some prior knowledge $\pi(\theta)$, a stochastic model and some observation $x$, we want to infer the most likely distribution $P(\theta|x)$ for our model parameters $\theta$ given the data $x$. This is encoded in Bayes’ Theorem:

$$P(\theta|x) \propto \pi(\theta) \, \mathcal{L}(x|\theta)$$

$$P(\theta|x) = \frac{P(\theta) \, \mathcal{L}(x|\theta)}{\mathcal{Z}}$$

where $\mathcal{Z} = \int P(\theta) \, \mathcal{L}(x|\theta) d\theta$.

For complex simulators, the **likelihood is defined implicitly and often intractable**. This is overcome with **simulation-based (likelihood-free) inference** (see e.g. Cranmer et al., 2020).
Simulation-based inference I

- To perform **Bayesian inference for any kind of (stochastic) forward model** (e.g. those specified by simulators), we use the following approach:

  1. sample $\theta_i$ from prior $\pi(\theta)$ for $i = 1, \ldots, N$

  2. run simulator for $\theta_i$ to produce observations $x_i \sim p(x|\theta_i)$

  3. train a conditional density estimator $q_\phi(\theta|x)$ on simulated data to predict parameters

  4. use density estimator to find approximate posterior for empirical data, i.e., $\hat{p}(\theta|x_0) \sim q_\phi(\theta|x_0)$

graber@ice.csic.es
Different approaches (all relying on deep learning) exist to learn a probabilistic association between the simulated data and the underlying parameters. These algorithms essentially focus on different pieces of Bayes’ theorem:

- Neural Posterior Estimation (NPE) (e.g., Papamakarios & Murray, 2016)
- Neural Likelihood Estimation (NLE) (e.g., Papamakarios et al., 2019)
- Neural Ratio Estimation (NRE) (e.g., Hermans et al., 2020; Delaunoy et al., 2022)

All methods exist in sequential form (SNPE, SNLE, SNRE), which adds a fifth step to workflow. Instead of sampling from the prior, we adaptively generate simulations from the posterior. This typically requires fewer simulations.

We focus on NPE. This allows us to directly learn the posterior distribution. In contrast, NLE and NRE need an extra (potentially time consuming) MCMC sampling step to construct a posterior.
Simulation-based inference

To perform **Bayesian inference for any kind of (stochastic) forward model** (e.g. those specified by simulators), we use the following approach:

1. sample $\theta_i$ from prior $\pi(\theta)$ for $i = 1, \ldots, N$

- [Diagram: 1. Sample $\theta_i$ from prior $\pi(\theta)$]

2. run simulator for $\theta_i$ to produce observations $x_i \sim p(x|\theta_i)$

- [Diagram: 2. Run simulator for $\theta_i$ to produce observations $x_i \sim p(x|\theta_i)$]

3. train a conditional density estimator $q_\phi(\theta|x)$ on simulated data to predict parameters

- [Diagram: 3. Train a conditional density estimator $q_\phi(\theta|x)$ on simulated data to predict parameters]

4. use density estimator to find approximate posterior for empirical data, i.e.,
   $$\hat{p}(\theta|x_0) \sim q_\phi(\theta|x_0)$$

- [Diagram: 4. Use density estimator to find approximate posterior for empirical data]

5. generate simulations adaptively

- [Diagram: 5. Generate simulations adaptively]

graber@ice.csic.es
Outline

- Neutron stars
- Pulsar population synthesis
- Machine learning and sbi
- Inference results
- Outlook
Workflow

● With our complex population synthesis simulator, we fix the dynamics to a fiducial model and focus on the magneto-rotational evolution.

● From our simulated populations, we generate summary statistics: density maps for three surveys in the PP-plane.

● To perform the inference, we use the PyTorch package sbi (Tejero-Cantero et al., 2020; https://www.mackelab.org/sbi/). Our trainable neural network has two parts:
  ○ CNN (see Ronchi et al., 2021): compresses the data into a latent vector.
  ○ Mixture density network: our posterior is approximated by a mixture of 10 Gaussians components; we learn the means, stds and coefficients.

● We initialise the CNN with Kaiming, use 89% of data for training, 10% for validation and 1% for testing, set the batch size to 8, and learning rate to $5 \times 10^{-4}$. 

graber@ice.csic.es
As our conditional density estimator is represented by a neural network, we can directly evaluate the posterior distributions for a given (test) observation.

We recover narrow, well-defined posteriors for all five parameters that typically contain the ground truth (parameters used for the forward simulation) at the 95% credibility level.
Posterior distributions for observed population

With our optimised neural network, we can also infer the posteriors for the pulsar population recorded in our three surveys and recover the 95% credibility intervals:

\[\mu_B = 13.07^{+0.07}_{-0.08}\]
\[\sigma_B = 0.43^{+0.03}_{-0.03}\]
\[\mu_P = -0.98^{+0.25}_{-0.29}\]
\[\sigma_P = 0.54^{+0.33}_{-0.25}\]
\[\alpha = -1.77^{+0.35}_{-0.38}\]
Outline

- Neutron stars
- Pulsar population synthesis
- Machine learning and sbi
- Inference results
- Summary and outlook
Take-home points

- Neutron stars are **compact remnants** that **emit pulsed radiation** across the electromagnetic spectrum.
- **Standard radio pulsars** constitutes the largest class of observed neutron star.
- Pulsar **population synthesis** bridges gap between known pulsars and the invisible population.
- It allows us to **constrain birth rates** of different neutron star classes **and birth properties**.

- **Deep learning** with neural networks is ideal to **analyse high-dimensional astrophysical data**.
- **Simulation-based inference** has opened up the possibility for statistical inference **for complex simulators**.
Outlook

- There are **several directions** that we have started to look into:

  ![Road](image.png)
Outlook

- There are **several directions** that we have started to look into:

**IMPROVING THE SIMULATOR**
- Explore **different assumptions on initial period and magnetic-field distributions**
- Extend framework to model also gamma-ray and X-ray emission and **predict the multi-wavelength emission**

**IMPROVING SBI**
- **Test other approaches**
- Expand the approach to **active learning** and derive posteriors sequentially **using SNPE**, etc.

graber@ice.csic.es
THANK YOU

Cassiopeia A supernova remnant
(credit: NASA/CXC/SAO)