Massive Black Hole Binary parameter estimation using Masked Autoregressive Flows

Ivan Martin Vilchez

LISA Data Analysis: from Classical Methods to Machine Learning

L2IT Toulouse, November 21–25 2022

Institute of Space Sciences

EXCELENCIA MARÍA DE MAEZTU

CSIC CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

IEEC Institut d'Estudis Espacials de Catalunya

UAB Universitat Autònoma de Barcelona

Towards a LISA DA pipeline

LISA L1 data $\rightarrow$???? $\rightarrow$ Posterior $p(\boldsymbol{\theta}|\mathbf{d})$

Bayesian inference

Likelihood $\log p(\,\mathbf{d} \mid \boldsymbol{\theta}\,) = \sum_{n=\{A,E,T\}} 2\,(\,\mathbf{h}(\boldsymbol{\theta}) \mid \mathbf{d}\,)_n - (\,\mathbf{h}(\boldsymbol{\theta}) \mid \mathbf{h}(\boldsymbol{\theta})\,)_n$

integrals!

forward simulations!

$$p(\,\boldsymbol{\theta} \mid \mathbf{d}\,) = \frac{p(\,\mathbf{d} \mid \boldsymbol{\theta}\,)\, p(\boldsymbol{\theta})}{p(\mathbf{d})}$$

Prior

Posterior

Evidence

even harder integrals!

$$p(\mathbf{d}) = \int p(\,\mathbf{d} \mid \boldsymbol{\theta}\,)\, d\boldsymbol{\theta}$$

Density estimation

- In practice, "classical" methods usually sample $p(\boldsymbol{\theta} \mid \mathbf{x}_{\text{obs}})$ with e.g. MCMC.
  - Use $p(\boldsymbol{\theta} \mid \mathbf{x}_{\text{obs}}) \propto p(\mathbf{x}_{\text{obs}} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$ to get posterior samples.
  - Sampling the likelihood can be tricky and take a long time…

- We'll try avoid computing $p(\mathbf{x} \mid \boldsymbol{\theta})$ at all using Machine Learning. Options:
  1. Estimate $p(\mathbf{x}, \boldsymbol{\theta})$, then substitute $\mathbf{x}_{\text{obs}}$ to find $p(\boldsymbol{\theta} \mid \mathbf{x}_{\text{obs}})$
  2. Estimate $p(\boldsymbol{\theta} \mid \mathbf{x})$ directly, then substitute $\mathbf{x}_{\text{obs}}$
  3. Estimate $p(\mathbf{x} \mid \boldsymbol{\theta})$ and use Bayes to find $p(\boldsymbol{\theta} \mid \mathbf{x}_{\text{obs}})$

Normalising Flows

Consider a complicated "target distribution" $q(x)$.
Can we find an invertible transformation $x = f(z)$ such that, starting from a simple "base distribution" $p(z)$, we recover $q(x)$ via change of variable?

$$\int q(x)\,dx = \int p(z)\,dz = 1$$

$$q(x) = p(z)\left|\frac{dz}{dx}\right| = p\big(f^{-1}(x)\big)\left|\frac{df^{-1}(x)}{dx}\right|$$

For multivariate distributions,

$$q(\mathbf{x}) = p(\mathbf{z})\,|\det \mathbf{J}[f^{-1}(x)]|$$

In practice, this determinant is generally **expensive** to compute!

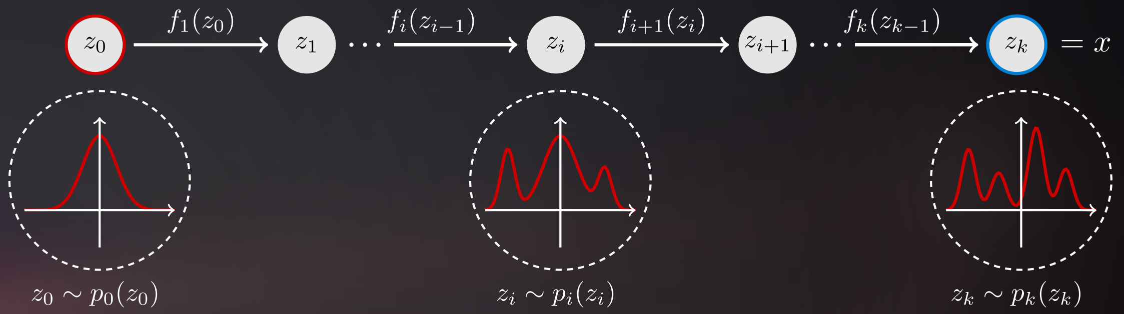


Image: Janosh Riebesell @ github

To create complex transformations, use compositions of simpler ones → a *flow* of transformations:

$$f(\mathbf{z}) = [f_k \circ f_{k-1} \circ \cdots \circ f_2 \circ f_1](\mathbf{z})$$

$$q(\mathbf{x}) = p(\mathbf{z}) \prod_{i=1}^{k} |\det \mathbf{J}[f_i(\mathbf{z}_{i-1})]|^{-1}$$

Masked Autoregressive Flows (MAF)

In summary: we will be chaining some transformation $f_i(\mathbf{z})$, which needs to be:

- Invertible
- Differentiable
- Easy-to-compute $\det \mathbf{J}$ (e.g., triangular $\mathbf{J}$)
- Expressive!

Several exist in the literature.

We choose: $x_j(\mathbf{z}) = \mu_j + z_j\,\sigma_j \quad \begin{cases} \mu_j = f_{\mu_j}(\mathbf{x}_{1,\dots,j-1}), \\ \sigma_j = f_{\sigma_j}(\mathbf{x}_{1,\dots,j-1}) \end{cases}$

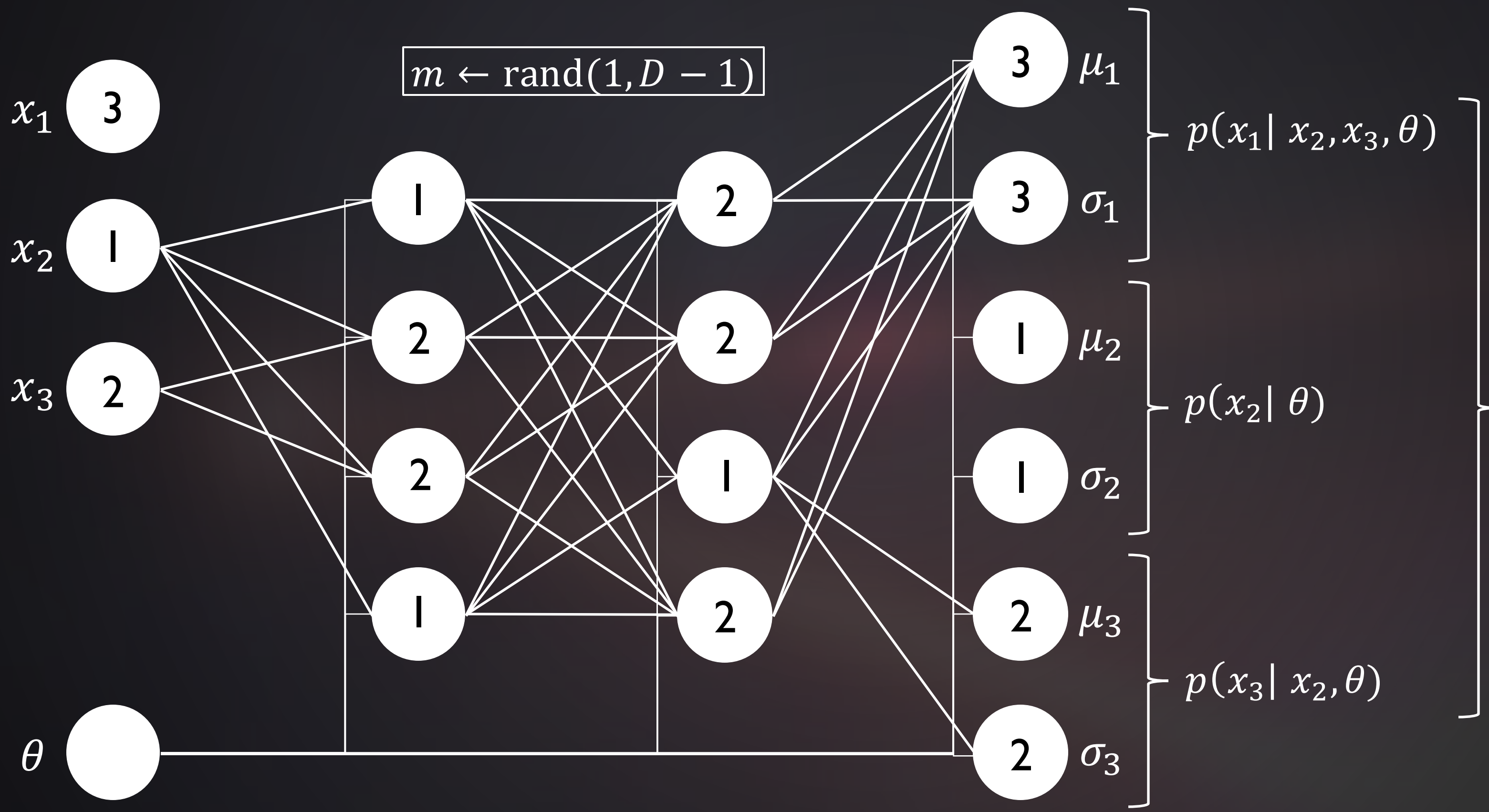
This is equivalent to modeling

$$q(\mathbf{x}) = \prod_{j=1}^{D} p(\, x_j \mid \mathbf{x}_{1,\dots,j-1}\,) = p(x_1)\, p(\, x_2 \mid x_1\,)\, p(\, x_3 \mid x_2, x_1\,) \dots \longrightarrow$$ This is our loss function!

$$\mathcal{N}(\, x_j \mid \mu_j(x_{1,\dots,j-1}), \sigma_j^2(x_{1,\dots,j-1})\,)$$

(Papamakarios et. al., arXiv: 1705.07057)

Masked Autoencoder for Distribution Estimation (MADE)



$$p(\mathbf{x} \mid \theta) = p(x_2 \mid \theta) \times p(x_3 \mid x_2; \theta) \times p(x_1 \mid x_2, x_3; \theta)$$

(Germain et. al., arXiv: 1502.03509)

MADE in practice: Masks

It is computationally easier to introduce masks $\mathbf{M}$ so that:

$$\mathbf{h}^l(\mathbf{h}^{l-1}) = g(\,(\mathbf{W}^l \odot \mathbf{M}^{\mathbf{W}^l})\mathbf{h}^{l-1} + \mathbf{b}^l + \mathbf{W}^l_\theta \boldsymbol{\theta})$$

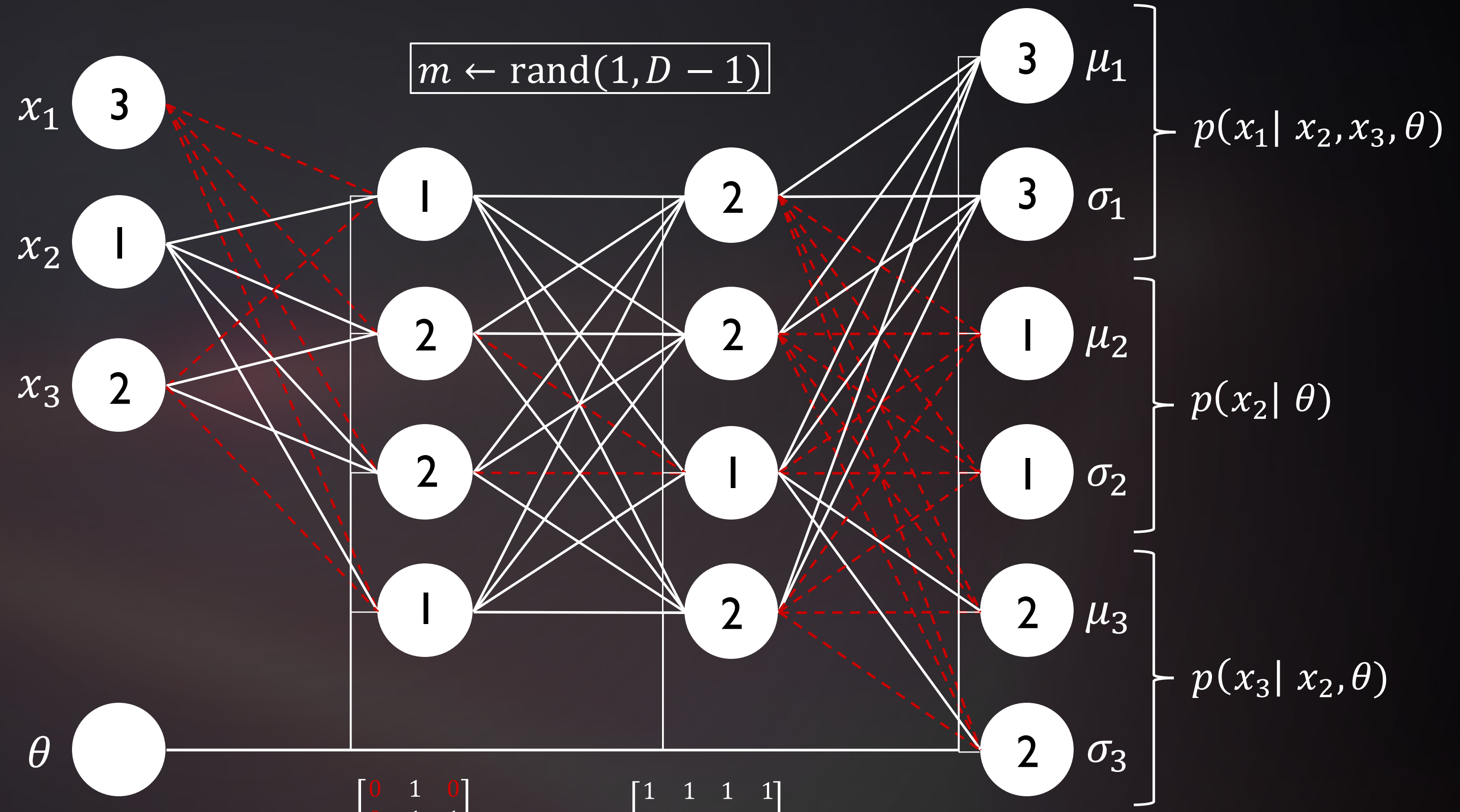
For hidden layers:

$$\mathbf{M}^{\mathbf{W}^l}_{k',k} = \begin{cases} 1 & \text{if } m^l(k') \geq m^{l-1}(k) \\ 0 & \text{otherwise.} \end{cases}$$

For the output layer:

$$\mathbf{M}^{\mathbf{W}^L}_{k',k} = \begin{cases} 1 & \text{if } m^L(k') > m^{L-1}(k) \\ 0 & \text{otherwise.} \end{cases}$$

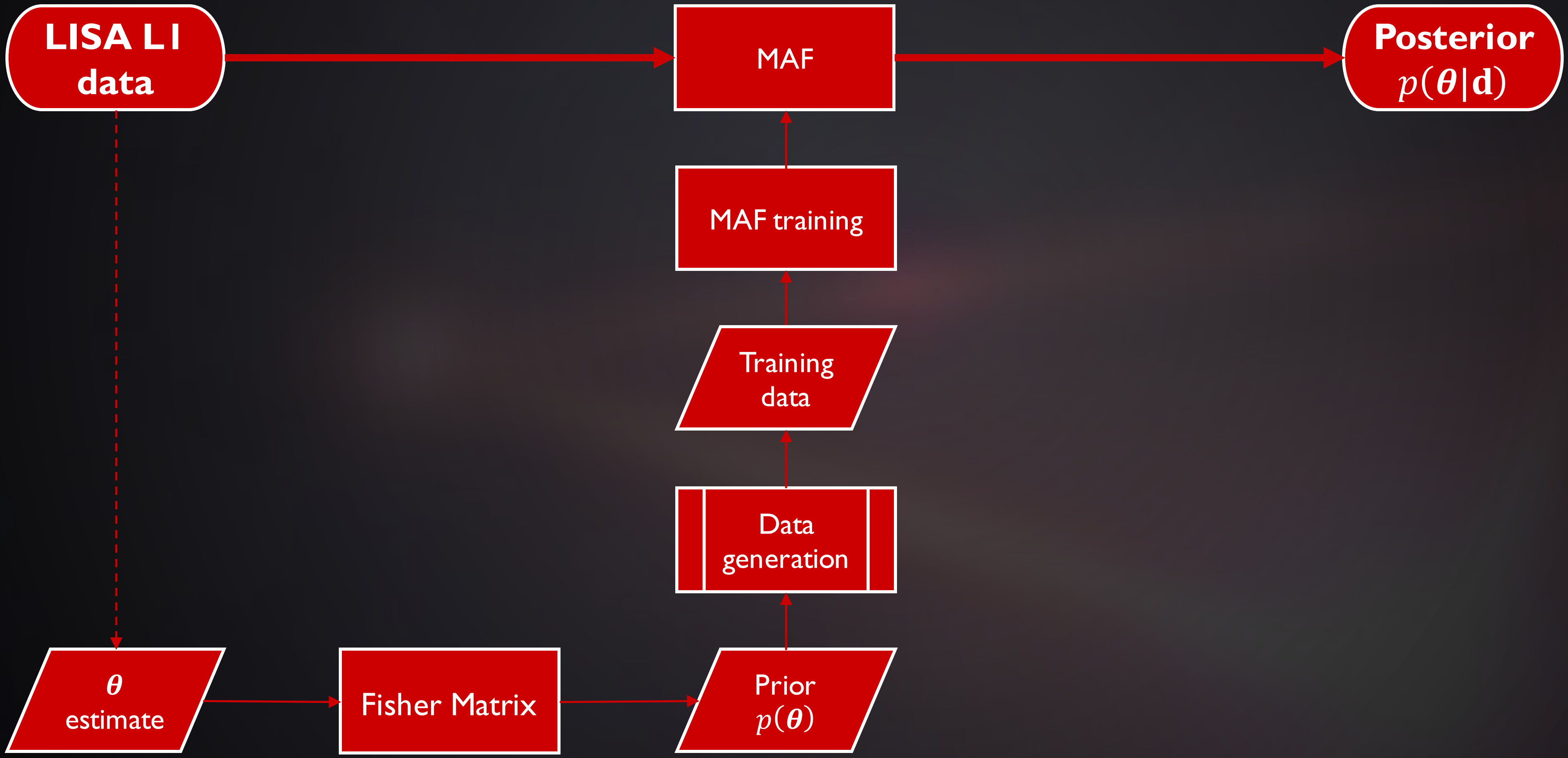
Stack several of these, with randomized ordering! ➡ **MAF**



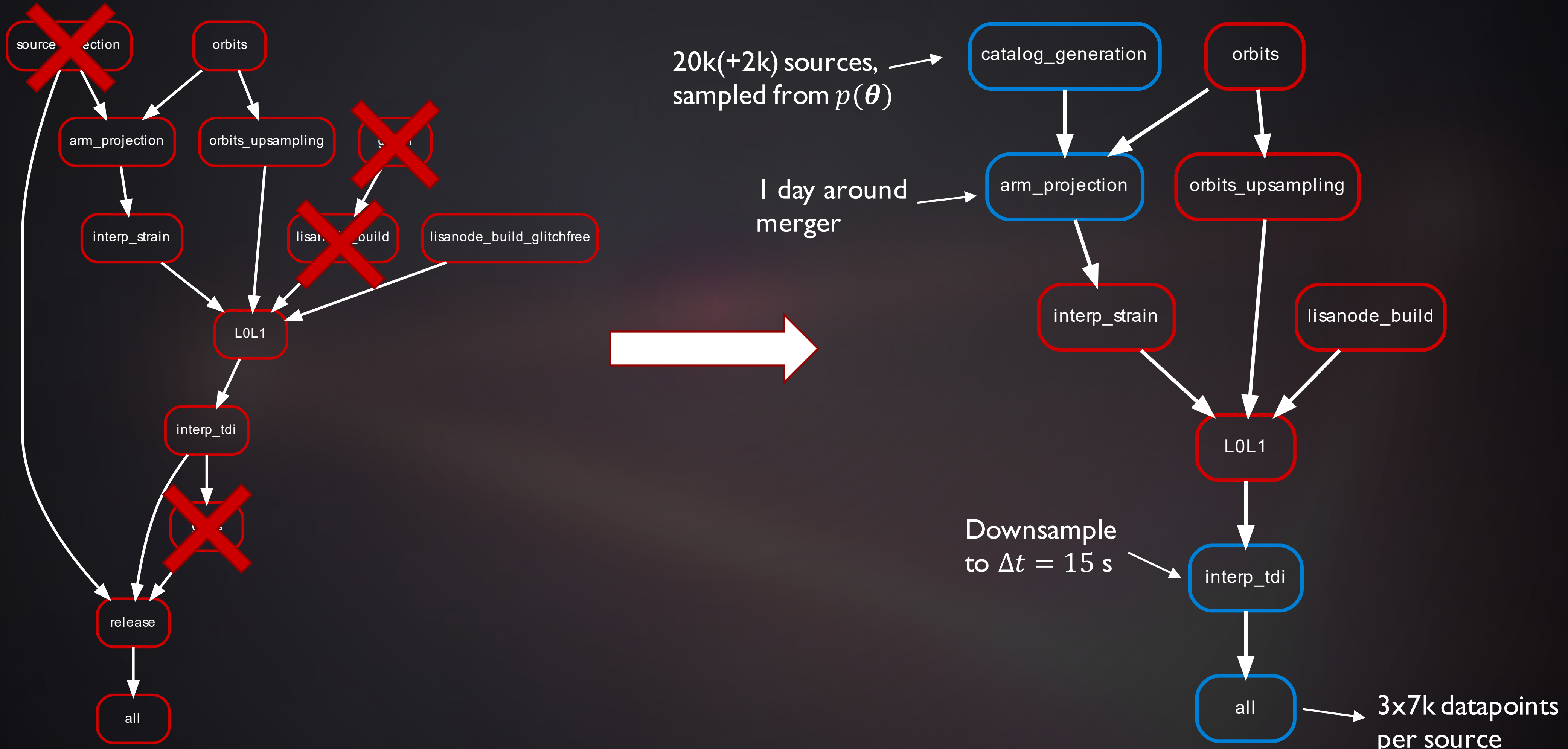
$$\mathbf{M}^{\mathbf{W}^1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{M}^{\mathbf{W}^2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \mathbf{M}^{\mathbf{W}^3} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(Germain et. al., arXiv: 1502.03509)

Towards a LISA DA pipeline



Data generation



Principal Component Analysis (PCA)

- Define

$$\mathrm{X} = \underbrace{\begin{bmatrix} | & | & \dots & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_n \\ | & | & \dots & | \end{bmatrix}}_{(n \text{ columns})} \Bigg\} \text{(}m\text{ rows)} \qquad (m \geq n)$$

- One can decompose $\underset{(m,n)}{\mathrm{X}} = \underset{\substack{(m,m)\\ \text{unitary}}}{\mathrm{U}} \, \underset{\substack{(m,n)\\ \text{diag.}}}{\Sigma} \, \underset{\substack{(n,n)\\ \text{unitary}}}{\mathrm{V}^T} \longrightarrow \mathbf{d}_j = \mathrm{U}\,\Sigma\,\mathrm{V}_j^T$

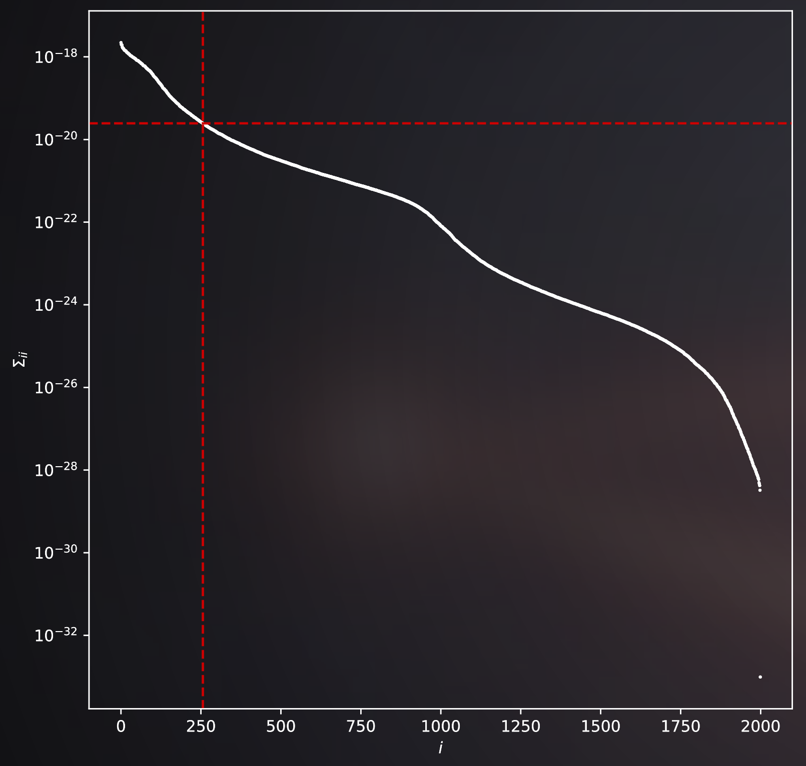
Singular Value Decomposition (SVD)

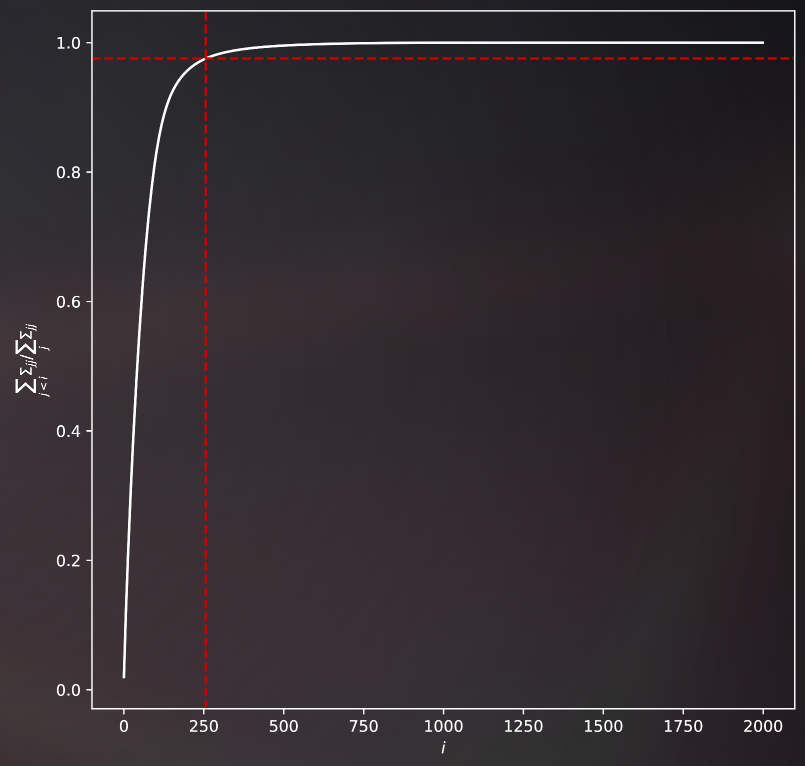
- Approximately, one can decompose

$$\mathrm{X} \approx \underset{(m,n)}{\widetilde{\mathrm{X}}_r} = \underset{(m,r)}{\widetilde{\mathrm{U}}} \, \underset{(r,r)}{\widetilde{\Sigma}} \, \underset{(r,n)}{\widetilde{\mathrm{V}}^T} \qquad (r < n)$$

- Then, for new $\underset{(m)}{\mathbf{d}} \rightarrow \underset{(r)}{\mathbf{x}} = \underset{(r,m)}{\widetilde{\mathrm{U}}^T} \underset{(m)}{\mathbf{d}}$ is a data summary!

Principal Component Analysis (PCA)





Some results

Validation

PRELIMINARY!!!

PRELIMINARY!!!

Some results

Spritz

PRELIMINARY!!!

PRELIMINARY!!!

Some results

"Spritz-like"

PRELIMINARY!!!

PRELIMINARY!!!

Conclusions

- We were able to successfully model the log-likelihood as a MAF

- Applications:
  - Very fast evaluation of the likelihood
  - GPU-accelerated MCMC, with gradients!
  - Fast waveform emulators, possibly?

- Caveats:
  - Tuned to narrow prior (but should be generalizable with enough compute power)
  - Issues with actual LDC data
  - Plenty of fine-tuning avenues