2024-25 Part III/MASt Astrophysics Research Project: Interim Report Coversheet

For completion by the student:					
Project Title:	**Developing machine learning tools for the analysis of gravitational wave signals**				
Student Name:	**Krish Nanavati**	**Signature:**	[illegible]	**Date:**	**06/12/24**
Student Declaration:	I declare that this interim report represents work undertaken as part of the NST Part III Astrophysics/MASt Astrophysics Research Project. It is the result of my own work and, includes nothing which was performed in collaboration. No part of the report has been submitted for any degree, diploma, or any other qualification at any other university. I also declare that an electronic PDF file containing this cover sheet and work has been uploaded to the Part III/MASt Astrophysics Moodle site on this date.				

For completion by the Supervisor(s):					
Supervisor 1 Name:	**James Alvey**	**Signature:**	[illegible]	**Date:**	**06/12/24**
Supervisor 1 Declaration:	I declare that I have reviewed this interim report prior to submission by the student.				

Interim Report

Part III Astrophysics Research Project

Developing machine learning tools for the analysis of gravitational wave signals

Krish Nanavati

November 2024

1 Introduction

Gravitational wave data analysis involves high-dimensional, complex data, for which fast and efficient analysis strategies are desired. The aim of this project is to implement the Neural Posterior Estimation (NPE) algorithm into `peregrine`, a simulation-based inference code for analysing gravitational wave data (Bhardwaj et al. 2024). Currently it uses the Truncated Marginal Neural Ratio Estimation (TMNRE) algorithm to estimate the posterior, in which the likelihood-to-prior ratio is estimated for one specific observation. We wish to look at sequential NPE and compare the two approaches on LIGO detections, such as GW150914 (Abbott et al. 2016).

2 Gravitational Waves Theory

First, we present the theory behind gravitational waves, their detection, and the Bayesian analysis of the signals. The linearised vacuum Einstein equations admit transverse, plane-wave solutions for perturbations to the spacetime metric - these are gravitational waves (Le Tiec and Novak 2017).

This project concerns gravitational waves produced by binary black hole mergers. The waveforms we use are from the 15-parameter `IMRPhenomXPHM` model (Pratten et al. 2021). Gravitational wave data analysis broadly consists of two tasks: detection and parameter inference - the latter is the focus of this project. We will be using simulation-based techniques, discussed in detail later, and data provided by the LIGO collaboration.

Understanding the form of the data in the detections is essential. We are given a power spectral density (PSD), $S_n(f)$, given by

$$\langle n^2(t)\rangle = \langle |\tilde{n}(f)|^2\rangle = \int_0^\infty S_n(f)df$$

where $n(t)$ is the noise in the signal. This is a key quantity in Bayesian inference appearing throughout our analysis. Other relevant quantities in the data we are given include the signal to noise ratio, S/N and the signal to noise ratio in energy (Maggiore 2007).

3 Machine Learning Theory

Next we study the machine learning tools we will be using. For this project, we will be using a variety of supervised (labelled data) and unsupervised (unlabelled data) machine learning techniques for density estimation.

Neural networks will be used because they can enhance the performance and capability of high dimensional parameter inference. We wish to use neural-network-enhanced generative models for posterior estimation. The most relevant models for our task are normalizing flows. These take a simple base density and generate a complicated one by applying a series of bijective transformations - the probability distribution changes using the following formula:

$$\log p_X(x) = \log p_Z(z_K) - \sum_{k=1}^{K} \log|\det \frac{\partial f_k(z_{k-1})}{\partial z_{k-1}}|$$

where the f_k are our transformations (Kobyzev, Prince, and Brubaker 2021).

With normalizing flows, we have a direct and explicit evaluation of the final probability distribution, as opposed to other methods where we get an estimate of the lower bound of the posterior. Another benefit is their expressiveness - in the infinite data limit, they can model any target distribution exactly from a given base. They are trained by backpropagation, where we start with the final transformation and optimise the transformations used in reverse order. One commonly chosen loss function is the Kullback-Leibler divergence between the target posterior and the approximated posterior, given by:

$$\mathcal{L} = \mathbb{E}_{z\sim p_Z(z)}\left[-\log p(\mathcal{D}|f_\phi(z)) - \log p(f_\phi(z)) + \log p_Z(z) + \log\left|\det J_{f_\phi}(z)\right|\right]$$

where $p_Z(z)$ is our base distribution, $\det J_{f_\phi}(z)$ is the Jacobian, $p(\mathcal{D}|f_\phi(z))$ is the likelihood and $p(f_\phi(z))$ is the prior over some parameters (Papamakarios et al. 2021).

We use simulation-based inference (SBI) throughout the project to approximate the likelihood. In situations like black hole mergers, where the latent spaces are too large to integrate over, simulators can produce data given parameters drawn from a prior. Posterior estimation methods with SBI include approximate Bayesian computation and using NPE with simulated data. The simulators can also be trained by active learning throughout, improving sample efficiency for inference (Cranmer, Brehmer, and Louppe 2020).

4 Development of Tools and Implementation of Theory

The majority of the project will be using the machine learning tools in the `pytorch` library, including `pytorch` Lightning. This simplifies handling complex datasets and provides a robust way to build scalable models, with flexibility for hardware acceleration.

In `pytorch` I have built increasingly complex models that can be taken forward into the project. First, using trainable parameters for linear regression in 1D, then fitting a gamma distribution, and using normalizing flows to fit a 2D Gaussian. I enhanced training by including validation training sets and validation loss tracking, learning rate schedulers and early stopping. I use `wandb` for visualisation and analysis of the training. I am currently working on implementing a Masked Autoregressive Flow (MAF) for more expressive and efficient density estimation. Examples of output are shown at the end of the report.

I have worked through the gravitational wave specific `jimgw` notebook for the analysis of a specific gravitational wave detection, GW150914 (Wong 2024). It takes the original data, filters it, analyses the noise samples, forms the PSD and plots the waveform, where we can observe the signal during the inspiral, coalescence and ringdown phases.

I am now working with the `sbi` repository, running and testing the tutorial examples (Tejero-Cantero et al. 2020). I specifically looked at NPE and implementing fully amortized posterior estimation, multi round inference and the impacts of using different density estimators and summary statistics, all of which will play a role in the final project.

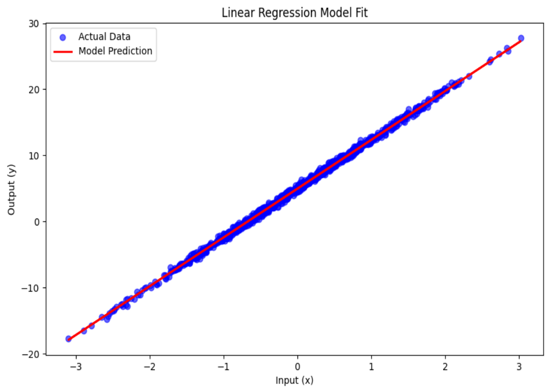
Finally, I studied the `peregrine` paper and its methodology for gravitational wave data analysis. `peregrine` uses a sequential approach which means that it requires only 2% of the waveform evaluations nested sampling does. We will be looking at implementing a similar sequential NPE algorithm in this project.

5 Next Steps

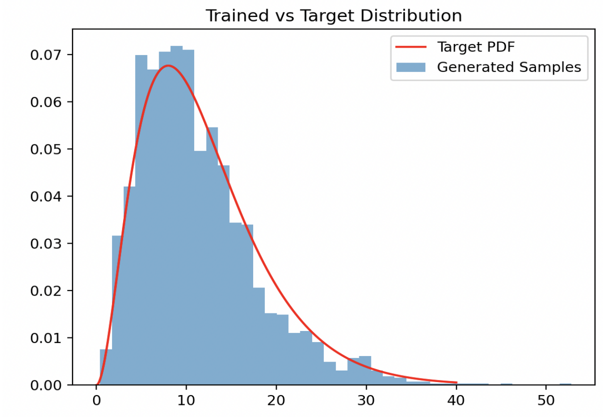
The next steps are as follows:

1. Implement a neural posterior estimation algorithm for some simple toy models or simpler data, which is analytically solvable so we can compare.
2. Study the theory behind sequential NPE (SNPE) and then apply this to the toy model as well.
3. Take this work done and apply it to the LIGO data we have, experimenting with and comparing different density estimators such as MAFs and neural spine flows.
4. Study some appropriate tests that can be done to compare the TMNRE and the SNPE algorithms and then carry them out

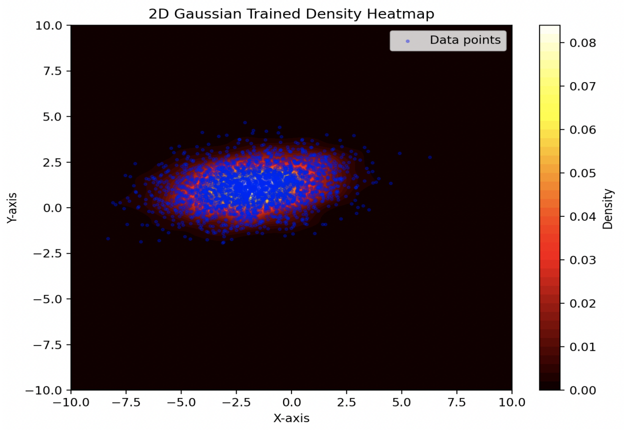
After this, depending on the outcome of the project and the time remaining, we can consider extensions to the project. One idea is comparing different summary statistics, seeing how this choice affects the accuracy and speed of the program.



(a) Linear Regression Fitting



(b) Sampling from trained gamma distribution



(c) Density Estimation for 2D Gaussian

Figure 1: Density Estimation with `pytorch` Examples