Generative Adversarial Networks for LIGO Parameter Estimation

Hunter Gabbard* and Chris Messenger
SUPA, School of Physics and Astronomy,
University of Glasgow,
Glasgow G12 8QQ, United Kingdom

(Dated: August 9, 2018)

This may be an abstract one day.

Introduction.— With the recent detections of six binary black holes [?] and a binary neutron merger GW170817 [?] by the LIGO-Virgo Scientific Collaboration (LVC), the era of gravitational-wave astronomy has begun. The LVC is composed of three detectors in Hanford, Washington State, Livingston, Louisiana, and Pisa, Italy. Once the detectors will have reached design senstivity, they will be able probe a search distance on the order of 200 Mpc. As the detectors become more sensitive, we will be limited by the sheer number of signals to process. As such, it is of paramount importance that we are able to get full parameter estimate posteriors on signals in a timely and efficient manner.

In order to determine the parameters of a detection, we apply Bayes' theorem [? ? ? ? ? ? ? ?]. We can gain most information concerning the parameters of a source through the probability density function (PDF) of all unknown parameters, given data from both detectors.

The PDF posterior is computed from a liklihood of the data given given some parameters multiplied by a prior PDF on the parameters made independent of the recorded data. We can compute marginalized PDFs using a suite of algorithms from C/C++ package known as `LALInference` [?]. Two independent stochastic sampling techniques are utilized in `LALInference` known as Markov-chain Monte Carlo [?] and nested sampling [?].

A variant of machine learning, generative adversarial networks (GANs), has gained some traction in recent years [?]. Some successful applications of GANs can be seen in image retrieval of historical archives [?], text to image synthesis [?], simulation of high energy particle showers [?], along with many others.

GANs are composed of two neural networks, a discriminator and a generator. The generator network attempts to map random latent variables to some distribution which the user wants the GAN to approximate, whereas the discriminator network will attempt to distinguish between samples from the real distribution and fake samples made by the generator network. You need only to feed into the generator as input a vector of latent random variables (can be uniform, Gaussian, etc.). The variables are latent in that they are not directly related to the distribution that the generator is trying to emulate, however in our case they do have an underlying Gaussian distribution. An additional motivation for using GANs is that they are also semi-supervised, so few training samples are needed, and you are left with by default two tools (one for classification and one for sample generation).

We use a variant of a GAN called deep convolutional generative adversarial networks (DCGAN) [?]. DCGANs are composed of two networks which act as the generator and discriminator, however the difference between DCGANs and their more standard GAN brethren is that both networks are entirely made of convolutional layers. We follow the architecture guidelines laid out in Radford et al. [?] which are as follows: replace pooling (downsampling) layers with strided convolutions (alternative to pooling), use batch normalization in both the generator and the discriminator, remove all fully connected hidden layers, use ReLU activation functions [?] in the generator except for the output, and use LeakyReLU activation functions [?] in the discriminator for all layers.

To-do:

- GWs.

Methods— Given any signal burried in noise, the GAN will attempt to produce reconstructed waveform estimates on the true underlying GW template waveform. After having produced reconstructed waveforms, these estimates are then fed to a convolutional neural network which has been trained seperately on noise-free waveforms to produce point estimates on their parameters.

All weights in the GAN are initialized randomnly over a uniform distribution prior to training. The generator portion of the network takes as input a vector of latent variables 100 samples long. After one training iteration of the generator, there is a second training procedure whereby we append an additional layer to the generator model. This final layer is composed of two neurons where one computes the mean of the difference between the generator output and h_t and the other computes the standard deviation of the difference between the generator output and h_t. This final layer enforces the generator to make an output which, when subtracted from h_t, will be a time series with unit variance and zero mean. The discriminator network then has to decide whether what the generator has made is either "real" or "fake". "Real" in this case, refers to gravitational wave template waveforms spanning a large parameter space and "fake" being the generator produced estimates on the h_t noise-free waveform.

Our GAN is trained using a BBH waveform of the same parameters as those from GW150914 which has been injected in Gaussian noise and is produced using `LALInference`. The injection is whitened using a simulated detector noise power spectral density (PSD) which is equivalent to what is expected from the Advanced LIGO design senstivity curves. Whitening is done in order to rescale the noise contribution at each frequency to have equal power.

The discriminator needs a set of "real" gravitational wave signals to distinguish from what the generator has made, so we simulated those using a library of gravitational wave data analysis routines called `LALSuite`. The parameters for these signals are distributed according to an astrophysical distribution. Systems are simulated such that $m_1 > m_2$, component masses are in the range from 5 - $95 M_\odot$, and there is zero spin. Signals are given a random right ascension and declination where we assume an isotropic prior on the sky. The phase is drawn using a uniform prior on the range $[0, 2\pi]$, whereas the cosine of the inclination angle is uniformly drawn between $[-1, 1]$. We have found that simulating $\sim 8,000$ templates is sufficient to cover the entire parameter space.

Training continues until both loss functions have reached some sort of equilibrium (usually known as a local Nash equilibrium). However, such local Nash equilibriums can ocassionally be off from the true Nash equilibrium, so it can be difficult determining a training stopping threshold. Training accuracy is usually quantified through use of the loss functions of the GAN, visual comparison of generator output to the true noise-free waveform and visual comparisons of the estimated posterior distributions of both methods. To-do:

- How do you incorporate priors.
- Convergence.
- Volume of training data.

Results—
To-do:
GW150914. Show PE estimates on mass, spins, etc.
Small section on waveform reconstruction. Plot of this as well.

- Which priors were used. Same as GW150914 analysis.

Conclusions—
To-do:
Percision that we get, speed (sell this hard),

- Sell speed, with the right caveats.
- Waveform reconstruction.
- We are not model indepenent.
- Non-Gaussian noise.

Acknowledgements.— We would like to acknowledge valuable input from the LIGO-Virgo Collaboration specifically from T. Dent, R. Reinhard, I. Siong Heng, M. Cavalgia, and the compact binary coalescence and machine-learning working groups. The authors also gratefully acknowledge the Science and Technology Facilities Council of the United Kingdom. CM is supported by the Science and Technology Research Council (grant No. ST/ L000946/1).