LSTM Model for prediction masses

Marcin Wierzbiński

AstroCeNT

November 10, 2020

Motivation

- Advanced LIGO has detected gravitational waves from many black-hole and neuron-stars mergers.
- However, signals from many of these object still are parametrized by masses.
- Goal: predict chirp mass from signal

Recurrent neural networks

- A recurrent neural network (RNN) is a subclass of neural networks which specializes in processing sequential inputs, like gravitational-wave data.
- For our analysis, we use a special type of RNNs called a Long Short-Term Memory (LSTM) network.

Long Short-Term Memory Networks

The advantages of LSTM networks over RNNs:

- The cell state carry information from past inputs. The network can learn long-term dependencies and elimete the vanishing (exploding) gradient problem.
- Gates let information to be added to (inputs gates) and removed from (forget gates) the cell state

Model of network

- To generate mock data we used core package to analyze gravitational-wave data, find signals, and study their parameters: PyCBC
- We use a recurrent neural network (LSTM) as subclass of neural networks which specializes in processing sequential inputs.
- We have model mass chrip:
  - $$m_{chirp} = (m_1 \times m_2)^{3/5}/(m_1 + m_2)^{1/5}$$

    $Y_{acc} = [m_{chirp}]$

Visualizing the mock data

approximant	mass1	mass2	spin1z	spin2z	inclination	coa_phase	delta_t	f_lower
'SEOBNRv4'	100	10	0.9	0.4	1.23	2.45	1.0/4096	40

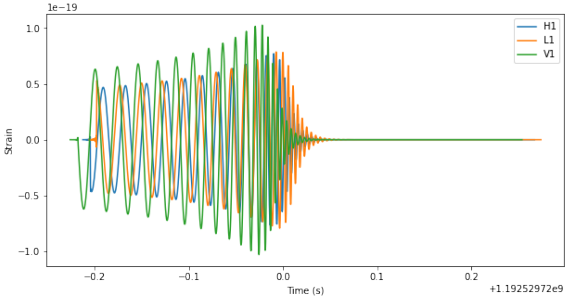


Figure: Example of generated waveform without noise from (L1) LIGO Livingston and LIGO Hanford(H1) and LIGO Virgo (V1) using above parameters

Hyper-parameter tuning

- Hyper-parameters determine the network structure(e.g number of LSTMs layers) and govern the learning process (e.g learning rate), and other useful parameters.
- A major strength of LSTMs is the ability to store and use information from past time.
- To search [1] for the optimal parameters, we apply algorithm: *Tree-structured Parzen estimator*.
- Validation loss function as minimization for *model.evaluate(X_Test, y_Test)*.

hyper-parameter	a priori	optimal
activation	$choice('relu', 'tanh')$	*tanh*
lr	$loguniform(np.log(10^{-6}), np.log(10^{-2}))$	9.35×10^{-5}
dropout	$uniform(0.0, 1.0)$	0.37
reg	$uniform(10^{-6}, 10^{-3})$	3.57×10^{-5}
numberofneurons	$uniformint(64, 1024)$	546

[1] We use library hyperopt

Model training- learning curves

- epochs: 100
- batch size: 32

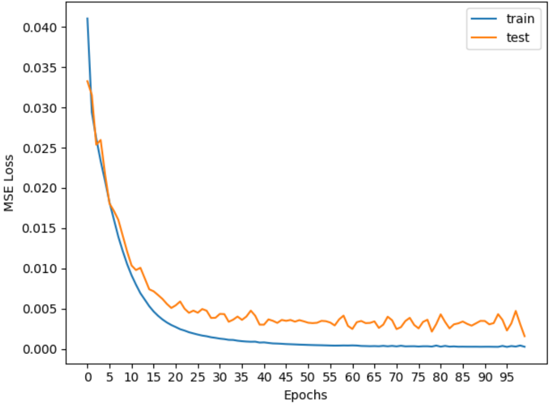
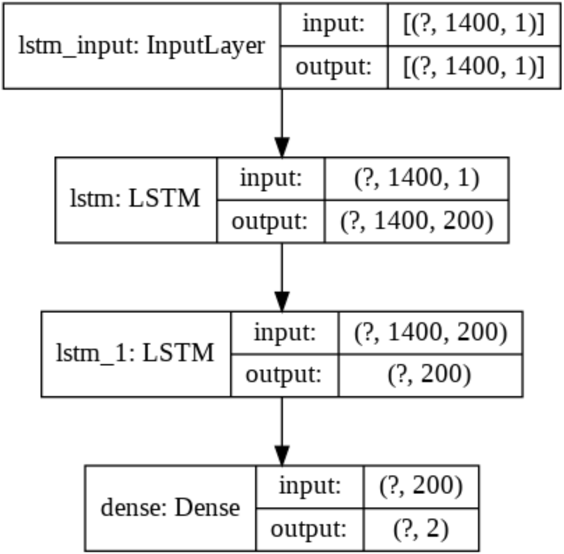


Figure: Learning curves for model with chirp mass. $RMSE \sim 0.05$

Model Structure



Error histogram - for arbitary unit

Masses were normalization. Our network requires masses normalize in interval $[0, 1]$

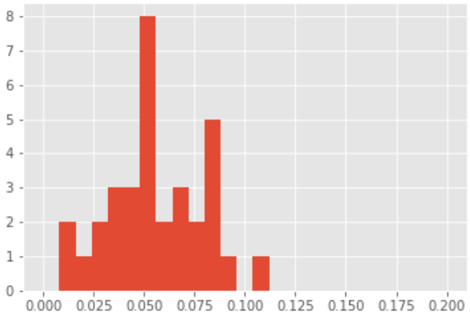


Figure: After normalization we calculate the difference between Y_{pred} and $Y_{observation}$ (X-axis.)