Understanding interferometer lock losses with machine learning

Laurel White

May 2019

1 Introduction

The Laser Interferometer Gravitational-Wave Observatory (LIGO) operates two detectors in search of gravitational waves. In each, a laser beam is split into two separate beams which are sent down two perpendicular arms, reflected at the end of each, and recombined such that there is near perfect destructive interference. Passing gravitational waves distort spacetime, creating a phase difference between the two beams and affecting the interference pattern. In order for one of these detectors to be sensitive enough to detect the minute phase shifts caused by gravitational waves, it must be in a stable state known as "lock". In lock, the mirrors are controlled with extreme precision such that the laser beams resonate in each cavity, building up power and signal strength, and there is destructive interference [1]. The commissioners who operate the detectors must work through several stages of the "lock acquisition" process before an interferometer is stable and data collection can begin.

The loss of lock leads to a significant amount of interferometer downtime as the commissioners work to realign the different components. While a "lock loss" is sometimes caused by a known source, such as an earthquake, there are other times when the reason is less apparent. There are thousands of auxiliary channels that collect data about the state of the interferometer, and we can potentially use machine learning techniques to identify features in these channels that signify the approach of a lock loss.

This project is based on previous work carried out by Jameson Rollins [2]. Rollins successfully created a dataset of "lock loss" and "no lock loss" times, extracted amplitude spectral densities (ASDs) from auxiliary channels during those times, reduced the ASD features to a smaller, more relevant set, and worked with a clustering algorithm that can be used to group related features. We will now build on this work to reach its end goal of better understanding lock losses so that we can reduce lost time during future data collection.

2 Objectives

We hypothesize that there are features in the auxiliary data channels which signal the approach of lock loss events and which can be linked back to underlying causes of lock loss. We intend to identify these causes and share the information with the detector commissioners so that they can prevent such events in the future.

In order to do this, we will first have to identify the features which are most likely to predict lock loss. This is crucial, because selection of features that are not correlated to loss of lock will not yield meaningful results. We will then have to design and implement an optimized regression algorithm, which needs to be able to reduce a large number of features to an appropriate number on which to run. Lastly, we will have to implement a clustering algorithm on the features identified by the regression in order to group similar features. A successful end product will be a set of data features that are proven to precede lock loss events and whose underlying causes can be determined so that the lock loss problem can be approached at the root.

3 Approach

The three main steps in this project are feature extraction, regression with regularization, and clustering. Our dataset will consist of the auxiliary channel data recorded during times preceding each lock loss event as well as "clean" data from those same channels recorded during stable times well removed from lock losses.

There is an automated tool available to us that can be used to identify times of lock loss, but we will manually identify clean times.

We will first extract features from the data samples that we believe to be indicative of impending lock loss. The previous work used ASDs, which are calculated by using Fourier analysis to determine the power in each frequency bin that is present in an auxiliary channel at a given time. An unstable oscillation, which is a proposed cause of loss of lock, would appear as an increasing power over time in a specific bin. However, it is possible to perform the feature extraction differently. We can use more frequency bins, or we can use the amplitude values as extracted from the time series data before any Fourier transform, among other options. We will manually look at auxiliary channel data from times before lock loss in order to decide which features are best to use. It may be valuable to use more than one type, as there may be multiple causes of lock loss that appear differently in the data. We will then use HTCondor to complete the feature extraction for our dataset. This will be computationally intensive, but each feature extraction need only be completed once.

We will use scikit-learn [3], a Python package, to analyze our dataset with machine learning. We will apply regression to develop a linear model that correlates the features in our data to the "lock loss" or "clean" labels and identify those features that contribute most to the model. We will use a regression algorithm that implements regularization to avoid overfitting the data by using too many channels and features. For this, we choose the Lasso model. It builds upon the ordinary least-squares linear regression, but it reduces the number of features used to create the model by driving coefficients of features that do not contribute significantly to zero. The degree to which it does this is controlled by the α parameter, and we will test different values of α in order to determine the optimal number of coefficients to use in our final model. One of the advantages of using the scikit-learn package for this task is that we can easily repeat the analysis several times using different algorithms or parameter values with a minimal amount of manual labor.

Once the regression is complete, we will be left with a set of features that are highly correlated with lock loss times. We will use one of the clustering algorithms available within scikit-learn to group these data. Previously, the mean shift clustering algorithm has been used, but we can test several, as with the regression. We will manually examine the output groups of features to determine whether they are actually similar or not and to try to identify an underlying cause of each group.

4 Algorithms

This project will make use of multiple algorithms. The following descriptions are based on the previous work [2] and the scikit-learn documentation [3].

The most important algorithm that we will use is Lasso regression. We will use our extracted features as our inputs, and they will be compiled into an m-by-n matrix $\mathbf{X}$. Each of the n columns will contain features that were extracted from one channel, and the total set of columns will include data from all of the channels. Each of the m rows will contain features that were extracted from the same time in the data. We will then create an output vector $\mathbf{y}$ with m components containing the "lock loss" and "clean" labels as either "positive" or "negative" values. For a given integer c where $1 \leq c \leq m$, the c-th component of $\mathbf{y}$ will be the label corresponding to the time from which the c-th row in $\mathbf{X}$ was extracted. The regression then solves

$$\mathbf{X}\mathbf{w} = \mathbf{y} \tag{1}$$

for a vector of coefficients $\mathbf{w}$ that linearly maps $\mathbf{X}$ onto $\mathbf{y}$. It is important to note that we will be working with far more features and auxiliary channels than there will be sample times in our dataset. Therefore, n will be much greater than m, and equation [1] will be underdetermined. This is why we will need to implement regularization, which the Lasso algorithm does by solving equation [1] such that it minimizes

$$\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \alpha||\mathbf{w}||_1 \tag{2}$$

where $||\mathbf{w}||_1$ is the 1-norm of $\mathbf{w}$

$$||\mathbf{w}||_1 = \sum_i |\mathbf{w}_i| \tag{3}$$

and α is the regularization coefficient responsible for penalizing the coefficients of those features that do not contribute significantly to the model. A higher α value drives more coefficients in $\mathbf{w}$ to be zero so that the final linear equation depends on less features.

5 Project Schedule

We will spend the first three weeks completing the feature extraction. This will involve compiling the dataset, manually examining auxiliary channels, writing code to extract the features, and running the code.

We will then spend three weeks on the regression. We will need to design code that will make our data compatible with the scikit-learn Lasso algorithm, run the regression, and examine the results to determine if an appropriate number of relevant features has been selected. We will repeat the regression with an adjusted α value as necessary.

We will spend three weeks on the clustering and final analysis. This will involve selecting a clustering algorithm, writing code to implement it on our selected features, and manually investigating the resulting groups to look for similarities between features that are indicative of underlying causes. If the clustering does not prove useful, we will repeat this process with a new algorithm

Lastly, we will spend one week summarizing the project results. This includes writing a final report, preparing a presentation, and communicating our results with the commissioners.

6 References

[1] Evans, M., et al. (2002). Lock acquisition of a gravitational-wave interferometer. *Optics Letters, 27(8).*
[2] Rollins, J. (2017, July 20). Machine learning for lock loss analysis.
https://dcc.ligo.org/public/0144/G1701409/001/main.pdf.
[3] Documentation of scikit-learn 0.20.3. (2018). Retrieved from
https://scikit-learn.org/stable/documentation.html