Understanding interferometer lock losses with machine learning

Laurel White[1] **and Jameson Rollins**[2]
[1]Syracuse University
[2]California Institute of Technology

Abstract. The Laser Interferometer Gravitational-Wave Observatory (LIGO) detectors are complex systems that must be extremely stable to detect gravitational-wave signals. There are numerous control loops in place to maintain detector stability, known as "lock," but, on occasion, the detector loses lock and a time-consuming lock acquisition process must be undertaken to regain it. These lock losses reduce the amount of data that LIGO is able to collect. In this project, we attempt to use machine learning to analyze past lock loss events and determine the underlying causes so that we can prevent lock losses in the future.

1. Introduction

The Laser Interferometer Gravitational-Wave Observatory (LIGO) operates two detectors in search of gravitational waves. In each detector, 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 signal the approach of a lock loss.

This project is based on previous work carried out by Jameson Rollins [2]. Rollins 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 follow the model used in 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 work 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 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 does not yield meaningful results. We then have to implement an optimized regression algorithm, which needs to be able to reduce a large number of channels to only those relevant to our analysis. Lastly, we have to implement a clustering algorithm on the channels and features identified by the regression as contributors to lock loss in order to group similar features. A successful end product is a set of 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. Work Completed

The three main steps in this project are feature extraction, regression with regularization, and clustering, but we first must compile the dataset. While there are thousands of auxiliary channels monitoring each of the detectors, we choose to focus on a reduced set to decrease the amount of data we are working with. We limit our analysis to the Hanford detector, and we eliminate any channels that are used for calibration or that are not intended for offline analysis. We identify 276 times at which a lock loss occurred while the interferometer was in its stable observing mode during the second LIGO observing run, O2. We generate our initial "positive" data samples, which have the lock loss label, by selecting 4 seconds of data before each lock loss from each channel of interest. We end our data samples 0.25 seconds before the identified lock loss times to allow for the delay between when the interferometer actually loses lock and when the automated lock acquisition system, Guardian, identifies the change in state. We then select 1086 4-second "negative" data samples from O2 in which the detector was observing. These segments are 1000 seconds removed from lock losses and from each other. The length of the samples is a parameter which can be tuned, so the analysis may need to be repeated with different lengths and the results compared to find the optimal value.

To begin our analysis, we look for features in the data samples that we believe to be indicative of impending lock loss. We cannot perform the regression using the raw timeseries data in which the amplitude of each channel is stored versus time because we are using 5385 channels that record up to 16384 data points per second and 1362 total 4-second samples from each channel. This amount of data would cause our regression to be time-consuming as well as severely under-determined, and the results would not be meaningful. The solution to this problem is to perform calculations on the samples that reduce the amount of data points while preserving the characteristics that allow us to distinguish between positive and negative samples.

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 in a control loop, which is a proposed cause of lock losses, would appear as increasing power over time in a specific bin of the ASDs for the relevant channels. However, with channel sample rates in the kilohertz range, in order to significantly reduce the amount of data, the frequency bins must be large, which may hide interesting features in the data.

We reference the documentation for a feature extraction package called tsfresh [3] and decide to extract the following features from our timeseries data:

- Difference between maximum and minimum values
- Complexity
- Absolute energy
- Mean value
- Standard deviation
- Variance
- Kurtosis
- Skewness
- Sample entropy
- Approximate entropy
- Absolute sum of changes
- Mean absolute change

The description of each feature except the first can be found in the tsfresh documentation.

We use HTCondor to parallelize the feature extraction for our dataset. We are currently working on running the feature extraction. We have removed channels from our list which were created during the middle of O2 because they cannot be used for all sample times. We have also removed sample entropy and approximate entropy from the list of features to extract because they are extremely time-consuming.

4. Work To Be Done

Once the extraction is complete, we will compile the extracted features into a large matrix. We will use scikit-learn [4], 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 (see Appendix A). 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 have to carefully select the value of α such that we do not end up with too many or too few features to analyze. We will use the LassoCV model, which will allow us to 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 with small tweaks while inputting a minimal amount of manual labor.

Once the regression is complete, we will need to evaluate its success. We are currently researching possible methods of evaluation, such as plotting the receiver operating characteristic curves. A successful regression will ideally result in a set of features from specific channels that are highly correlated with the difference between lock losses and stable 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. A good test of how well our methods have worked is whether or not we recover features in the seismic channels that are indicative of the approach of an earthquake-induced lock loss. We may need to closely study the channels and control loops in the interferometer to understand how they are related and how specific problems arise. We may also need to repeat some or all of the steps in the analysis to tune parameters such as the sample length and the α value.

5. Conclusion

Currently, we are nearly complete with our first round of feature extraction on the selected dataset. We will soon be able to perform the regression for the first time. The results of the extraction and regression should allow us to evaluate how well our approach is working towards the goal of solving the mystery of lock losses.

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 tsfresh. (2019). Retrieved from https://tsfresh.readthedocs.io/en/latest/

[4] Documentation of scikit-learn 0.20.3. (2018). Retrieved from https://scikit-learn.org/stable/documentation.html

Appendix A. Lasso

The main algorithm that we use is Lasso regression. The following description is based on the previous work [2] and the scikit-learn documentation [4]. We use our extracted features as our inputs, and they are compiled into an m-by-n matrix $\mathbf{X}$. Each of the n columns contains features of the same type that were extracted from one channel, and the total set of columns includes each feature extracted from each channel. Each of the m rows contains features that were extracted from the same sample time in the data. We 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 $\mathbf{y}$ is the label corresponding to the time from which the c-th row in $\mathbf{X}$ was extracted. The regression then solves

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

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

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

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

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

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.