Data Mining in Gravitational Wave Analysis

J. Andrew Casey-Clyde

May 24, 2017

Abstract

Gravitational wave detectors such as LIGO and Virgo offer new insights into our universe, yet also present challenges for analysis, primarily due to the sensitivity of the detectors to terrestrial sources of movement, as well as to the volume of data generated by these detectors. However, given the natures of these challenges, they are well suited to machine learning solutions. In this paper, I present techinques for the identification of transient events in LIGO timeseries data, as well as an example of how these transients may then be further analyzed using neural networks. Testing transient identification against hardware injections, I find 10 transients in data containing a total of 13.

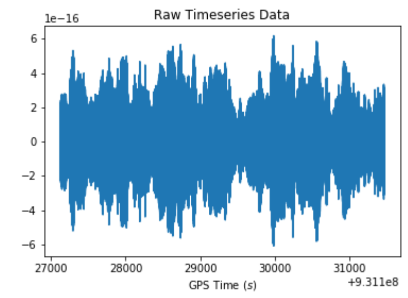
1 Introduction

The search for gravitaional waves was a long one. Sparked in 1916 by Albert Einstein's theory of General Relativity, it wasn't until 100 years later, in 2016, that the first two gravitaitonal wave events were confirmed using data from Advanced LIGO's Hanford (H1) and Livingston (L1) detectors[1] [2]. LIGO's instruments yield large quantities of data, which can be a double edged sword from an analysis perspective. The data is also subject to large amounts of terrestrial noise, which may be confused for transient gravitational wave events. This makes the data difficult for manual examination, so instead we turn to clustering and machine learning techniques, which offer powerful tools for extracting meaning from all this data.

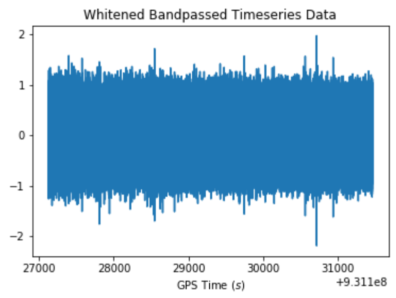
In this paper, I apply the Kliene Welle algorithm[3][4] to the first two weeks of LIGO's S6 data release, which uses clustering to find energy overdensities in gravitational wave timeseries and identify transient events. I then use a Multi-Layer Perceptron (MLP) neural network to classify these events, with LIGO hardware injections serving as simulations of various classes of gravitaitonal waves.

2 Methods

In order to classify transients in the LIGO data, they must first be found. To this end, I apply the Kleine Welle algorithm, which decomposes wavelets into



(a) Strain data before whitening



(b) Strain data after whitening and applying a butterworth bandpass filter. While data quality has been improved, it is still difficult to pick out individual transients by eye, making this a task much better suited to computational tools.

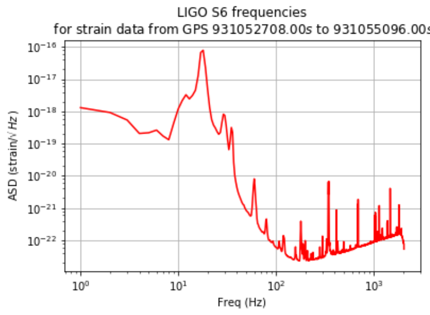
Figure 1: Strain data for a section of LIGO S6 data. Each data file contains 4096 seconds of strain data, and may possibly include hardware injections. Injection periods are ~ 100s in duration, though actual events are expected to take place over $\sim$ ms timescales. The cleaned data has been whitened by dividing its fourier transform by its ASD, and then had a butterworth bandpass applied from $80 - 300$Hz

the time-scale domain and searches for regions of energy overdensities in the signal[5][4]. These techniques may be applied to all detector channels to extract transients, and in fact will be necessary for differentiating between terrestrial and astrophysical transient sources.

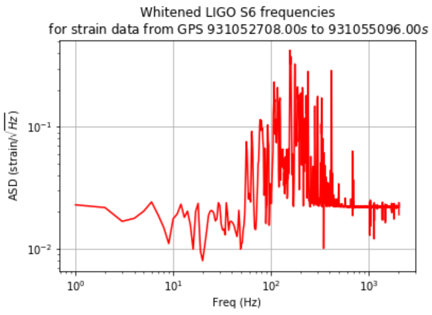
2.1 Data Cleaning

The LIGO gravitational wave data is characterized by Gaussian noise, with non-Gaussian transient events. As such, before applying the Kleine Welle algorithm to search for overdensities, it is useful to first whiten the data by taking its Fourier transform and dividing by the amplitude spectral density (ASD), which is the square root of the power spectral density, and then transforming back[6]. This helps to suppress signal due to noise, enhancing transient signals. An example of strain data pre and post whitening and bandpassing is shown in Figure 1, and its ASD at all stages of the cleaning process in Figure 2.

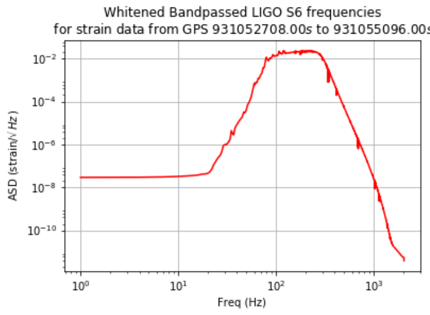
Additionally, a butterworth bandpass filter may be applied to the whitened data[4][6] in order to remove regions of large or fluctuating background noise, such as those regions outside $80 - 300$Hz in Figure 2a.



(a) ASD before whitening



(b) ASD after whitening



(c) Whitened ASD with a butterworth filter applied at critical frequencies 80Hz and 300Hz

Figure 2: ASD vs. frequency for a segment of LIGO S6 data, before and after whitening. Notice that in about the $80-300$Hz range, the noise is at a minumum, and relatively constant. This is the range we wish to search for transient events.

2.2 Event Identification

Once the data has been whitened and bandpassed, it can be fed through the Kleine Welle algorithm. The algorithm works by decomposing the time domain strain data into a wavelet decomposition, whose coefficients can be used to search for signal energy overdensities[5][4]. In general, the wavelet transform is defined by the equation[4][7]

$$W_f(u,s) = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt, \tag{1}$$

where ψ is some mother wavelet (in our case, a Haar wavelet) whose height and width have been scaled using the factor s. By applying this transformation across multiple wavelet scales, and on blocks of the timeseries whose widths match the width of the mother wavelet at that scale, the time series signal can be broken down into component wavelets, whose coefficients are given by the above transformation for each wavelet block. The calculation of this transformation can be made computationally inexpensive through the discretization of s to a dyadic sequence, such that $s \in \{2^j\}_{j \in \mathbb{Z}}$[4][7].

Applying this to the LIGO timeseries data at multiple scales (and switching to j, for convenience with the dyadic series) yields a series of detail coefficients D_j for each scale, with elements d_{ij}, whose values scale with signal energy and approach a zero mean Gaussian[4]. We can threshold on these elements to look for outliers by using the standard deviation of each decomposition level, σ_j, such that $d_{ij}/\sigma_j > 5$. Elements which pass this threshold are then normalized to create a set of square normalized coefficients such that

$$E_j = D_j^2/\sigma_j^2 = \{\epsilon_{ij}\}, \tag{2}$$

where the elements ϵ_{ij} correspond to the same elements d_{ij}. Taking only those ϵ_{ij} whose corresponding detail coefficients passed the earlier thresholding, we can cluster these over time, decomposition scale, and normalized energy. The mean shift clustering method is particularly well suited to this problem, as it does not require prior knowledge or the number of clusters present[8], which in principle we cannot know for a given segment. Several examples of this clustering are shown in Figure 3.

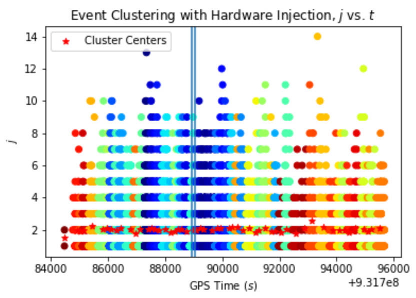
The elements of a given cluster C can then be added to form the total normalized cluster energy[4],

$$E_c = \sum_{(i,j)\in C} \epsilon_{ij}. \tag{3}$$

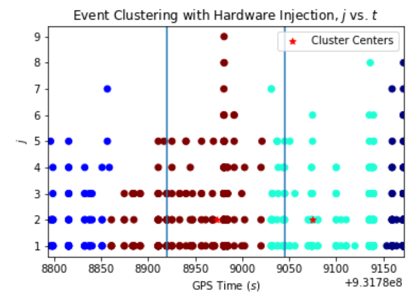
The significance of a cluster is then defined as[4]

$$S = -\ln \int_{E_c}^{+\infty} \chi_N^2(E) dE, \tag{4}$$

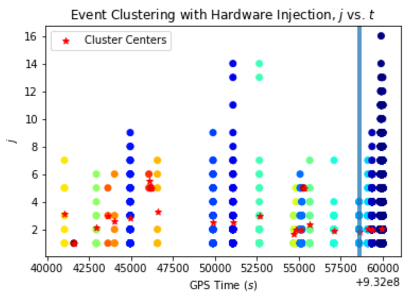
where the number of degrees of freedom N is the number of elements in the cluster. With this, it simply becomes a matter or thresholding the significance of each cluster.



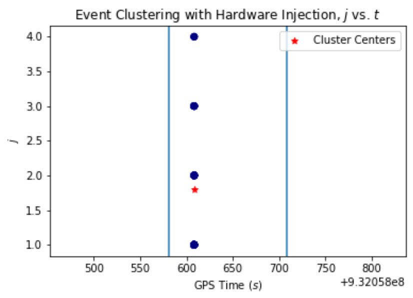
(a)



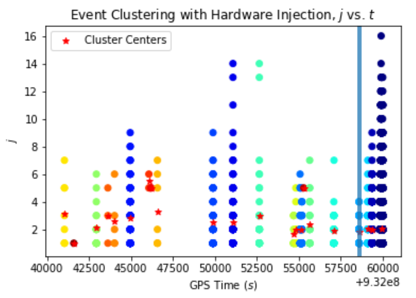
(b)



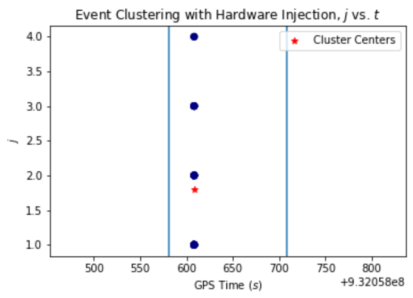
(c)



(d)



(e)



(f)

Figure 3: Examples of clustered segments, including hardware injections, with closeups of each hardware injection on the right. The first section (Figs 3a and 3b) has noticably more transients than the other sections, potentially due to persistent environmental conditions (such as a storm). Due to the nature of the detectors, they are also highly sensitive to environmental noise[9].

2.3 Event Classification

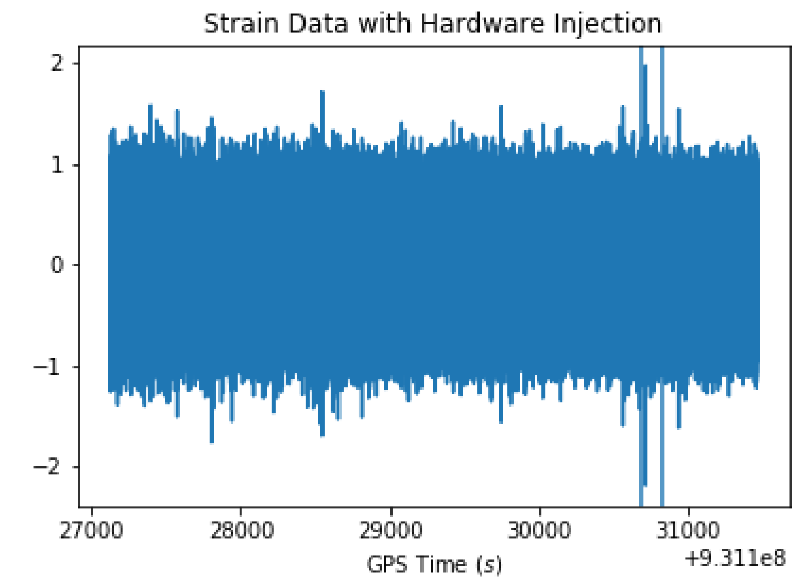
Once transients have been identified in the strain data, they can be classified using a neural network; in this case, I choose to use a Multi-Layered Perceptron, one of the more popular neural networks. For training data, we can use hardware injected events for astrophysical event classification. Additionally, we also need non-astrophysical events to train on. These can be generated by looking for events in each detector that do not overlap with events from the other detector (i.e., events in H1 that do not overlap with events from L1). By applying the hardware injections and independent events as labels, we can then form a set of data for training and testing.

3 Analysis

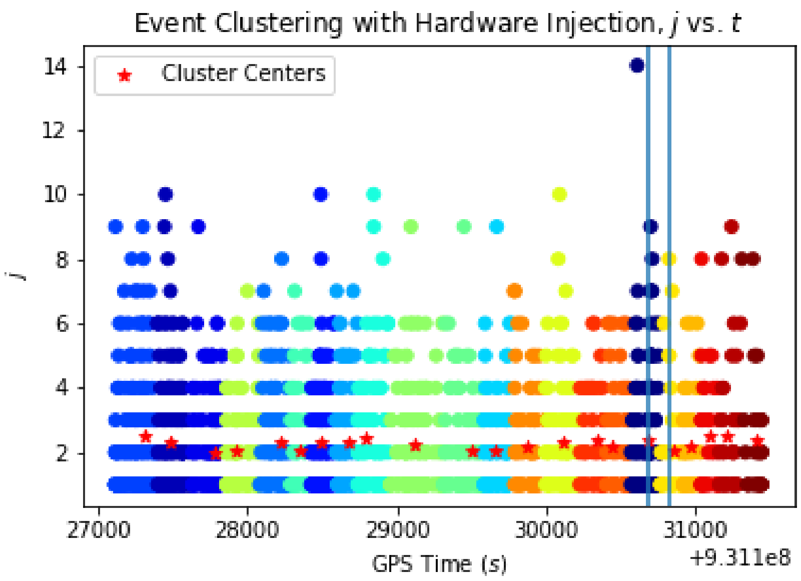
Keeping in mind that each LIGO science run contains data collected using progressively more sensitive instruments, I choose to apply these techniques to the first two weeks of the S6 data release[10], from the most recent data collection run before Advanced LIGO instruments came online, which offers the best sensitivity for publically available data. This data has been downsampled (by LIGO) from 16384Hz to 4096Hz[10]. The data files must be cached locally, so local storage availability is a factor when determining how much data may be examined. The data files themselves contain strain data in the form of a timeseries, as well as meta data, including channels indicating when hardware injections of simulated gravitational wave data were active, as shown in Figure 4. These data files are loaded as a series of time segments using LIGO's readligo python module[11], with each data segment being run through a Kleine Welle algorithm as described above.

Wavelet decomposition (eq. 1) is carried out using the open source PyWavelets package[12]. The detail coefficients, D_j, are then examined at each scale j, and statistically significant ($d_{ij} > 5\sigma_j$) points are found. These points are clustered in time, scale, and square normalized energy, ϵ_{ij}, into event clusters using a Mean Shift algortihm[13], which does not require prior knowledge of the number of clusters, which in general would not be known. For the purposes of this paper, and because I check this algorithm against injection periods, the Mean Shift bandwidth is set to 100s, which is the approximate timescale given for injection periods. However, in practice this bandwidth can (and should be) tuned to smaller values that are more representative of event timescales for transient searches. The significance of each cluster (as defined by equations 3 and 4) is then calculated, and events with a significance under 20 are discarded. When a significant cluster ($S > 20$) is found, it is recorded in a data file, with start and end times, event length, the significance of the event, and the cluster center (in relative time, scale, and energy). Using this technique, I find 1312 events from H1, and 1188 events from L1.

In order to make use of a classification neural network, we first need labeled data to train it with. The S6 LIGO data includes injection time periods, i.e.



(a) Strain data with hardware injection



(b) Clustering in time and scale, including cluster centers. Each cluster represents a transient event.

Figure 4: Clustering in time and scale for a section of strain data that includes a hardware injection. Blue bars in both images mark the start and end of the hardware injection period.

when and for how long each type (Compact Binary Coalescence (CBC), Burst, and Stochastic) of hardware injection was active, which can be used to generate this labeled data. By comparing these activation periods to the start and end times of each detected event, we can identify which of our events correspond to hardware injections. Additionally, we can use the hardware injection data to count the number of injections, to gauge how well we can detect injections. In this portion of the season S6 data, there exist 7 CBC injections and 1 Burst injection in H1, and 3 CBC and 2 Burst injections in L1 (with no Stochastic injections in either detector). Comparatively, the Kleine Welle algorithm is able to pick out 6 CBC and 1 Burst cluster in H1, and 1 CBC and 2 Burst clusters in L1. It's noted, however, that the number of CBC injections found is very sensitive to both, the clustering banwidth and the σ_j filtering level, with a change in the latter leading to more CBC injections found than exist. This sensitivity may possibly be due to the large range in wavelet widths that a CBC event may span, though more investigation should be done to confirm this. Additionally, we can confirm some events as being terrestrial in origin by comparing the data from each detector, H1 and L1, to each other and looking for events which do not overlap in time, $\pm$100ms, following the same time window used by Biswas, et al[5], and find 1139 terrestrial events in H1 and 1016 terrestrial events in L1.

All identified events can then be labeled and used for training and testing the neural network. For neural network model, I choose to use a Multi-Layer Perceptron (MLP), though in principle, any neural network which can be trained for classificaiton will work. For my data, I train the MLP with event data that includes each events duration, cluster energy, significance, and cluster center. To characterize the performance of the MLP, I apply cross validation to the labeled data containing all of the injected events with $k = 10$, getting a score of 0.99 ± 0.03. However, due to the low number of injection events at hand, this score should not be regarded as conclusive, as the training samples are simply too small.

By applying the MLP to the H1 and L1 data seperately, these events may be validated against each other to find gravitational wave candidates. However, in applying this I find no non-terrestrial events. Considering the section of data used, this is more likely due to necessitating further refinements to the clustering process, and could also likely be helped with a larger section of data containging more injections.

4 Conclusion

Machine learning offers powerful tools for dealing with the vast quanitities of data that need to be searched for gravitational waves. As more detectors come online, and the sensitivity and time resolution of these detectors increase, techniques such as these will become invaluable in the search for gravitational waves. In particular, the Kleine Welle algorithm seems to be well suited to finding transient events. Further improvements to this method should first be directed at the clustering methods used, and in particular in resolving the issue of fully clustering CBC wavelets into a single event, due to the large range in component wavelet widths that may comprise these events.

In all, this algorithm offers a powerful tool for finding transients for further analysis with other tools, such as neural networks. A more in depth analysis of these applications fell beyond the scope of this paper (due largely to hardware constraints), however the brief application of an MLP neural network may at least serve as a proof of concept that this type of data is not incompatible with neural network analysis.

References

[1] The LIGO Scientific Collaboration and Others. GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. feb 2016.

[2] B.P. Abbott and Others. GW151226: Observation of Gravitational Waves from a 22-Solar-Mass Binary Black Hole Coalescence. *Phys. Rev. Lett.*, 116(24):241103, jun 2016.

[3] S Chatterji, L Blackburn, G Martin, and E Katsavounidis. Multiresolution techniques for the detection of gravitational-wave bursts. *Class. Quantum Gravity*, 21(20):S1809–S1818, oct 2004.

[4] Lindy Blackburn. KleineWelle Technical Document, 2007.

[5] Rahul Biswas, Lindy Blackburn, Junwei Cao, Reed Essick, Kari Alison Hodge, Erotokritos Katsavounidis, Kyungmin Kim, Young-Min Kim, Eric-Olivier Le Bigot, Chang-Hwan Lee, John J. Oh, Sang Hoon Oh, Edwin J. Son, Ruslan Vaulin, Xiaoge Wang, and Tao Ye. Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data. *Phys. Rev. D, vol. 88, Issue 6, id. 062003*, 88(6), mar 2013.

[6] LIGO Scientific Collaboration and Jonah Kanner. SIGNAL PROCESSING WITH GW150914 OPEN DATA.

[7] Stéphane Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 2nd edition, 1999.

[8] Željko Ivezić, Andrew J. Connolly, Jacob T. VanderPlas, and Alexander Gray. *Statistic, Data Mining, and Machine Learning in Astronomy.* Princeton University Press, Princeton, New Jersey, 2014.

[9] Jessica McIver, for the LIGO Scientific Collaboration, and for the Virgo Collaboration. Data Quality Studies of Enhanced Interferometric Gravitational Wave Detectors. apr 2012.

[10] LIGO Scientific Collaboration. LIGO Open Science Center release of S6, 2015.

[11] Jonah Kanner, Roy Williams, and Alan Weinstein. readligo python module, 2017.

[12] Filip Wasilewski. PyWavelets - Wavelet Transforms in Python PyWavelets Documentation.

[13] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in {P}ython. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.