

Sensitivity study using machine learning algorithms on simulated *r*-mode gravitational wave signals from newborn neutron stars

Antonios Mytidis,¹ Athanasios Aris Panagopoulos,² Orestis P. Panagopoulos,³
Andrew Miller,^{1,4,5} and Bernard Whiting¹

¹*Department of Physics, University of Florida, 2001 Museum Road, Gainesville,
Florida 32611-8440, USA*

²*Department of Computer Science, California State University Fresno,
5241 North Maple Avenue, Fresno, California 93740, USA*

³*Department of Computer Information Systems, California State University Stanislaus,
One University Circle, Turlock, California 95382, USA*

⁴*INFN, Sezione di Roma, I-00185 Roma, Italy*

⁵*Università di Roma La Sapienza, I-00185 Roma, Italy*



(Received 10 August 2015; revised manuscript received 8 November 2018; published 15 January 2019)

This is a follow-up sensitivity study on *r*-mode gravitational wave signals from newborn neutron stars illustrating the applicability of machine learning algorithms for the detection of long-lived gravitational wave transients. In this sensitivity study, we examine three machine learning algorithms (MLAs): artificial neural networks, support vector machines, and constrained subspace classifiers. The objective of this study is to compare the detection efficiencies that MLAs can achieve to the efficiency of the conventional (seedless clustering) detection algorithm discussed in an earlier paper. Comparisons are made using two distinct *r*-mode waveforms. For the training of the MLAs, we assumed that some information about the distance to the source is given so that the training was performed over distance ranges not wider than half an order of magnitude. The results of this study suggest that we can use the machine learning algorithms as part of an investigative stage in the pipeline that would be able to provide very fast and solid triggers for further, and more intense, investigation.

DOI: [10.1103/PhysRevD.99.024024](https://doi.org/10.1103/PhysRevD.99.024024)

I. INTRODUCTION

In the late 1990s, the *r*-mode quasitoroidal pulsations of a neutron star became very promising for generating strong gravitational wave signals due to the Chandrasekhar-Friedman-Schutz instability they exhibit [1–3]. *R*-modes of any harmonic, frequency, and amplitude are subject to this instability at any angular velocity of the star [4,5]. Therefore, even the smallest toroidal perturbations in the velocity of the neutron star mass currents will keep increasing in amplitude. In the absence of a saturation mechanism, these small perturbations could eventually reach energy values of the order of the rotational energy of the neutron star.

In considering the saturation amplitude, α , its normalization is such that values of order 1 carry energy of the same order of magnitude as the total rotational energy of the neutron star. Some authors have introduced damping mechanisms that can cause saturation at *r*-mode oscillation amplitudes of order $10^{-4} - 10^{-2}$ dimensionless units [6], while others have introduced mechanisms that cause saturation at amplitudes equal to or larger than 10^{-1} [7]. Some of the factors that can affect the order of α are the equation of state (EOS) of the matter in the center of the

neutron star [8], the magnitude of the magnetic fields on the neutron star [9,10], the coupling of the *r*-modes with other inertial modes [11], and magnetohydrodynamic coupling to the stellar magnetic field [12]. Therefore, an *r*-mode detection and a subsequent estimation of the saturation amplitude will impact all of the above theories depending on the order of magnitude of α they predict.

The physical significance of an *r*-mode gravitational wave detection has been extensively studied over the past 15 years [13–15]. Theoretical studies suggest that (assuming the *r*-mode oscillation amplitude grows sufficiently large) *r*-mode gravitational radiation (primarily in the $m = 2$ harmonic) could carry away most of the angular momentum of a rapidly rotating newborn neutron star. Therefore, an *r*-mode detection would also (i) provide an explanation of the low rotational frequencies of the observed neutron stars when compared to their possible rotational frequencies at birth, (ii) set constraints on the equation of state of the matter in the core of the neutron star, and (iii) set upper bounds on α and settle the debate about the magnitude of the saturation amplitude of the *r*-mode oscillations on neutron stars [16].

In a previous study, we argued that the most promising *r*-mode gravitational wave sources are newborn neutron

stars [8]. In Sec. I A [Eq. (3)], we show that, due to their high angular velocities, newborn neutron stars will emit the most powerful r -mode gravitational radiation among all other possible sources. Therefore, the design of an r -mode search from newborn neutron stars depends on an electromagnetic trigger from a supernova (type-I or type-II) event. The distance to the r -mode source is needed to extract any information about the magnitude of α because an r -mode detection can only give an estimate for the ratio α/d , as shown in Sec. II, Eq. (5). Distances to type-I supernovae can be calculated using the standard candle method with an error between 5% and 10%. [17]. Distances to type-II supernovae can be calculated using the expanding photosphere method giving an error of 10%–15% [18,19].

The results of our previous sensitivity study showed that advanced LIGO (aLIGO) can be sensitive to r -mode signals from newborn neutron stars only within our local group of galaxies. Since distances to galaxies in our local group are already known, a supernova event within our local group would automatically give information about the distance to the hypothetical r -mode gravitational radiation source. The latest supernova event in our local group (SN2014J) occurred in January of 2014 in the galaxy Messier 82 (M82) in the nearby group of galaxies M81, and it was a type-I supernova [20]. This galaxy is at a distance of 3.5 Mpc from the Earth. This is a factor of 3 farther than the aLIGO sensitivity that our previous sensitivity study showed. At that distance, the supernova event rate is only three to six per century [21,22], and the best we can do in order to be ready for the next event is to increase the sensitivity of our algorithms or apply a new class of more efficient algorithms.

In this study, we investigated the applicability of machine learning algorithms (MLAs) as decision makers (signal or not) for the detection of r -mode gravitational waves. This study was performed by integrating the MLAs in the stochastic pipeline [23–25]. The objective of this pipeline is to explore the possibility of sources of *long-lived* gravitational wave transients lasting from many seconds to weeks. Searches for long-lived gravitational wave transients have a strong scientific motivation. This is a cross-correlation-based analysis pipeline, which was formed to bridge the gap between short $\mathcal{O}(s)$ burst analyses and stochastic analyses (in which the signal is assumed to persist through the duration of the data-taking run). The pipeline is framed as a pattern recognition problem.

The investigation we present in this paper is a preliminary one with the target to initiate further research in this field. The purpose of this paper is to provide some insight into how we can harness the power of MLAs and use them for the r -mode gravitational wave searches. Additionally, the methodology we followed here may also be used for a broader investigation of the applicability of MLAs for the detection of other long-lived gravitational wave transients. The aim of this paper is not to demonstrate how we can use

MLAs in order to make a detection announcement. Instead, the aim is to perform a preliminary investigation on how we can use raw data taken by the LIGO detectors, preprocess them, and feed them into three separate MLAs. The ultimate target is to use MLAs in a way that we can facilitate the searches for long-lived gravitational wave transients in the stochastic pipeline.

A. R -mode model

The r -mode gravitational wave model we used in our present study as well as in our previous work [8] is based on the 1998 model of Owen *et al.* Though very simplistic, this model is still a very good approximation for the early stages of the neutron star spin-down [15]. More complicated numerical methods have shown that an r -mode saturation amplitude $\alpha = 10^{-2}$ can result in a spin-down of which the energy loss can be detected as gravitational radiation by aLIGO [6]. When this saturation amplitude is used in the 1998 model, we see that there is a good agreement in the angular velocity evolution of the neutron star up to several months after the start of the neutron star spin-down [8]. The evolution of the gravitational wave frequency emitted by the neutron star in the 1998 model by Owen *et al.* is described by

$$f(t) = \frac{1}{(f_o^{-6} + \mu t)^{\frac{1}{6}}}, \quad (1)$$

where μ is an EOS dependent parameter [8]. For a polytropic EOS, this parameter is expressed as a function of α as follows:

$$\mu = 1.1 \times 10^{-20} |\alpha|^2 \frac{\text{s}^{-1}}{\text{Hz}^6}. \quad (2)$$

For the same model and the same EOS, the gravitational radiation power is given by

$$\dot{E} \approx 3.5 \times 10^{19} f^8 |\alpha|^2 \text{ W}. \quad (3)$$

This model depends on two parameters: the (dimensionless) saturation amplitude, α , of the r -mode oscillations and the initial gravitational wave spin-down frequency f_o . The theoretical predictions for the values of these parameters were discussed extensively in our previous paper. The values we considered for α lie in the range of 10^{-3} – 10^{-1} , while the values we considered for f_o lie in the interval of 600–1600 Hz. Due to the wide range within which the values of these parameters lie, we cannot effectively use a matched filtering algorithm. Instead, we have to develop techniques that could detect all possible distinct waveforms.

B. Previous work

In our previous paper, a seedless clustering (SC) algorithm was used [26]. This seedless clustering algorithm integrates the signal-to-noise ratio (SNR) of pixels along predetermined monotonic curves (clusters) with arbitrary start and stop times and the constraint that there is a minimum total duration. Then, the algorithm performs the weighted sum of the pixel-SNR values along each curve to calculate the cluster SNR. After repeating this T times (where T is a free parameter of the algorithm), the algorithm records the largest cluster-SNR value. The algorithm records the largest cluster-SNR value for each one of the frequency-time (ft) maps that goes through the pipeline. This method is not dependent on any knowledge of the signal, and it can be applied generically to any long-lived gravitational wave transients. In particular, it is unable to discriminate between r -modes and other possible gravitational wave sources. Knowledge of the r -mode signal can be used to make minor modifications in the clustering algorithm; however, there was not much hope for a dramatic improvement in the efficiency. Nevertheless, we were able to recover signals of magnitude five times weaker than the noise.

In the sensitivity study performed for the clustering algorithm, we used nine distinct waveforms. These were chosen by taking (α, f_o) pairs using three values (10^{-1} , 10^{-2} , 10^{-3}) for α and three values (700, 1100, and 1500 Hz) for f_o . In this sensitivity study for the MLAs, for comparison purposes, we used two of these waveforms: $(f_o = 1500 \text{ Hz}, \alpha = 0.1)$ and $(f_o = 1100 \text{ Hz}, \alpha = 0.01)$. These waveforms as well as their corresponding power decays are shown in Figs. 1 and 2, respectively. MLAs are well suited especially for cases where the signal is not precisely (but only crudely) known. This paper is based on three specific MLAs: artificial neural networks (ANNs) [27], support vector machines (SVMs) [28], and constrained subspace classifiers (CSCs) [29]. All three methods are considered novel applications in the area of long transient gravitational wave searches.

This paper is organized as follows. In Sec. II, we present the details of the sensitivity study design. A more detailed description about how the data are prepared is given in Appendix D. That discussion includes the resolution reduction performed on the data maps before we perform the training of the MLAs. After the training is performed, we present the plot in Fig. 10 of the training efficiencies as functions of the resolution reduction factor. This plot provides the motivation for reducing the resolution of the data maps by a factor of 10^{-2} per axis. In Sec. III, we present a summary of the three MLAs we used for our sensitivity study: Sec. III A describes the training of the ANN algorithm, Sec. III B describes the training of the SVM, and Sec. III C describes the training of the CSC algorithm. The details of the mathematical formulations are presented in the Appendices: Appendix A for the ANN,

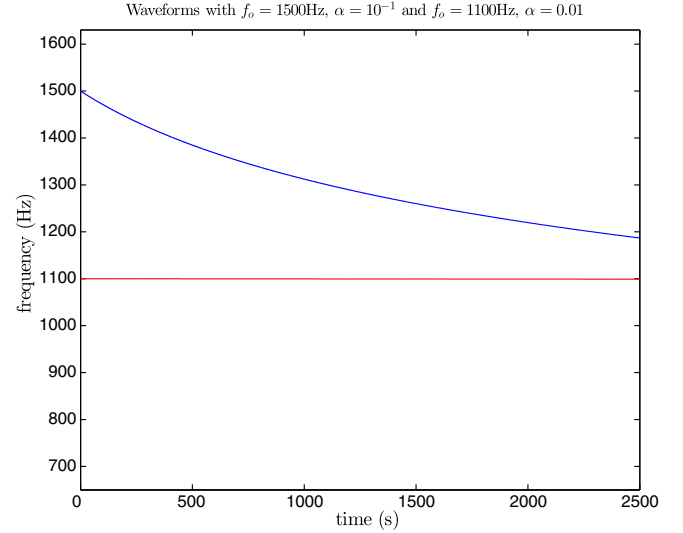


FIG. 1. The $(f_o = 1500 \text{ Hz}, \alpha = 0.1)$ waveform is the most powerful waveform considered in our sensitivity studies for both the clustering and the MLAs. The second waveform we chose has an amplitude 25 times smaller than the first one. This waveform has parameters $(f_o = 1100 \text{ Hz}, \alpha = 0.01)$ and is approximately monochromatic for the durations for which our sensitivity studies were designed. The clustering algorithm could detect the weaker signal at distances not further than a few kiloparsecs.

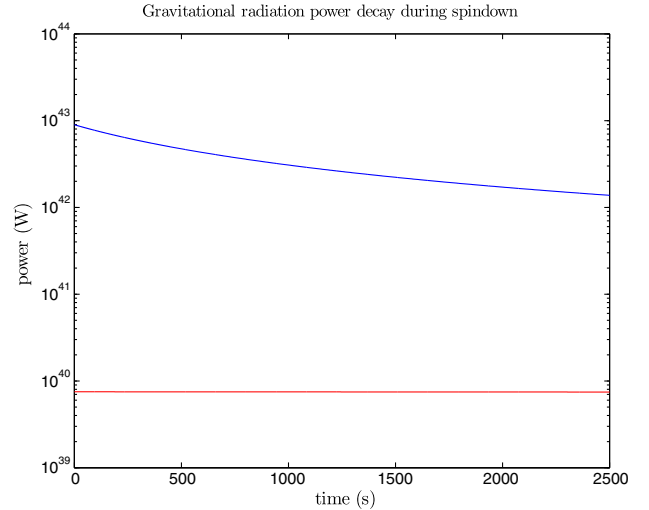


FIG. 2. These power evolutions correspond to the signals in Fig. 1. We see that the blue plot corresponds to a rapidly decaying signal: within 2500 s, the radiation power drops to 17% of its original value. The red plot corresponds to a power decay that dropped to only 99.9% of the initial. The power of this (red) signal is about 3 orders of magnitude lower than that of the signal plotted in blue. We chose this weak signal so that we can examine how the MLAs compare to the clustering algorithm for both powerful and weak signals.

Appendix B for the SVM, and Appendix C for the CSC. In Sec. IV, we present the results of our sensitivity study and compare the MLA efficiencies to the SC algorithm

efficiencies. Finally, in Sec. V, we summarize our conclusions and topics for future work.

II. SENSITIVITY STUDY DESIGN

The sensitivity study design for the MLAs has several differences from our previous sensitivity study for the clustering algorithms. For the latter, we only had to produce nine waveforms with $\alpha = 10^{-3}$, 10^{-2} , 10^{-1} , and $f_o = 1500$, 1100 , and 700 Hz. For each waveform, we created injection sets (100 injections per set), and each injection set corresponded to a specific injection distance. Using appropriate distance ranges, we proceeded with this method until we got a 50% detection rate. The distance corresponding to that success rate was marked as our detection distance. The detection threshold was taken to be the loudest “cluster SNR” [23] (as seen by the SC algorithm [26]) among 1000 frequency-time maps with pure detector noise. This threshold cluster SNR determined a false alarm probability (FAP) of 0.1%.

The approach for the MLA sensitivity study was different. The MLAs were trained not only for those nine waveforms but also for as many as possible distinct waveforms. In the paragraphs that follow, we discuss why we chose 11 350 distinct waveforms, each one injected at a distinct distance. Those 11 350 waveforms had α and f_o parameter values uniformly distributed in their corresponding parameter value ranges. By training the MLAs with these 11 350 distinct injections, we succeeded in getting the MLAs to recognize not only these 11 350 waveforms but also all possible waveforms in the whole range of α and f_o values at all possible distances (assuming the signal strength was high enough). This result of getting the MLAs recognizing signals outside the set of the signals used for training is called “generalization”.

A. Choice of the f_o and α parameter values

From Eqs. (1) and (2), we have the two model parameters α and f_o that determine the shape of the waveform. Apart from the shape, the injections that were produced for the training of the MLAs were also dependent on the pixel brightness or pixel SNR. For a single pixel in the ft maps, the SNR satisfies [23]

$$\text{SNR}(t, f, \hat{\Omega}) \propto \text{Re}[\hat{Q}_{ij}(t, f, \hat{\Omega})C_{ij}(t, f)], \quad (4)$$

where $i = 1$ and $j = 2$ are indices corresponding to the two aLIGO detectors [30,31], $\hat{Q}_{ij}(t, f, \hat{\Omega})$ is a filter function that depends on the source direction $\hat{\Omega}$ [32], and $C_{ij} \equiv 2\tilde{h}_i^*(t, f)\tilde{h}_j(t, f)$ is the cross spectral density, $\tilde{h}$ being the Fourier transform of the gravitational wave strain amplitude h . The latter is expressed in Ref. [9] as a function of the distance d to the source, the gravitational wave frequency f , and the r -mode oscillation amplitude α , by

$$h \approx 1.5 \times 10^{-23} \left(\frac{1 \text{ Mpc}}{d} \right) \left(\frac{f}{1 \text{ kHz}} \right)^3 |\alpha|. \quad (5)$$

For the construction of the injection maps, we chose the three parameter values f_o , α , and h^2 to be uniformly distributed within predetermined value ranges as explained below.

Each injection set that was produced and used for the MLA training was limited to 11 350 injection maps and 11 350 noise maps. This was due to the computational resources available as well as the time needed to produce the 22 700 maps. For each injection, the waveform was randomly chosen in such a way that the α value was randomly chosen from a uniform distribution of 11, 350 α values in the range of 10^{-3} – 10^{-1} , the f_o value was randomly chosen from a uniform distribution of 11,350 f_o values in the range of 600–1600 Hz, and for the h^2 values we picked three ranges (for three separate MLA trainings), the choice of which is discussed in the next subsection, Sec. II B.

B. Choice of the h^2 parameter values

The results of the sensitivity study for the clustering algorithm showed that for a signal of $f = 1500$ Hz and $\alpha = 0.1$ the detection distance was up to 1.2 Mpc. Using Eq. (5), we see that the SC algorithm can detect gravitational wave strains of value $h \approx 4 \times 10^{-24}$. Values of the same order are obtained if we substitute the results for the other eight waveforms. For example, from Table I in Ref. [8], we see that for $f = 700$ Hz and $\alpha = 0.01$ we get a detection distance of 0.043 Mpc. Substituting in Eq. (5), we get $h = 1.2 \times 10^{-24}$. Therefore, the value of $h \approx 10^{-24}$ will become a reference point because this is the value of the gravitational wave strain the MLAs will have to detect if they are shown to be at least as efficient as the SC algorithm [26].

If we consider supernova events at distances in the range from 1 kpc to 1 Mpc, then the corresponding range for the gravitational wave strain values is $h \approx 10^{-24}$ to 10^{-21} . Therefore, there are several approaches in determining the range of h^2 values for the injection maps produced for the training of the MLAs. The first approach was to produce one set of data with injections at distances distributed in such a way that the h^2 values are uniformly distributed in the range:

(a) from 10^{-48} to 10^{-42} ($10^{-24} \leq h \leq 10^{-21}$).

In case the 11 350 noise maps plus the 11 350 injection maps will not be sufficient to achieve generalization during the training of the MLAs in the above range of values of h , the alternative steps would be to create injections with values of h in smaller ranges. Therefore, we chose to produce three sets of data such that the h^2 values are uniformly distributed in the following ranges:

(b) from $10^{-46.4}$ to $10^{-45.4}$ ($10^{-23.2} \leq h \leq 10^{-22.7}$).

(c) from $10^{-47.4}$ to $10^{-46.4}$ ($10^{-23.7} \leq h \leq 10^{-23.2}$).

(d) from $10^{-48.0}$ to $10^{-47.4}$ ($10^{-24.0} \leq h \leq 10^{-23.7}$).

The last choice of 10^{-24} is such that the waveform with ($f_o = 1500$ Hz, $\alpha = 0.1$) may be detectable up to a distance of 5 Mpc, depending on the MLA detection efficiencies. Note that at those distances (in the neighborhood of the Milky Way) the supernova event rate is one every one to two years [33,34].

After producing the simulated data using the stochastic pipeline, the noise and injection maps represent data in the frequency-time domain. Hence, each data map is called a ft map. These ft maps data (already normalized) are preprocessed further so that we can create the “data matrix” that will include all the data that will be imported and used for the MLA training. Each row in the data matrix corresponds to a ft map, and each column corresponds to each pixel of the ft map. The details of this preprocessing are explained in Appendix D. For the discussion on the MLAs that follows, we will refer to our data used for the MLA training as the data matrix.

III. MACHINE LEARNING ALGORITHMS

There have only been a couple of applications of MLAs in the area of the detection of gravitational waves. One application is for gravitational wave searches of black hole binary coalescence with an application of random forests algorithms (RFAs) [35], and the other one is about the identification of noise artifacts (or glitches) in gravitational wave data with an application of ANNs, SVMs, and RFAs [36]. Our study is investigating the application of three MLAs (ANNs, SVMs, and CSCs) for the detection of long transient gravitational wave signals. These are signals of duration from O (sec) up to O (months), and they are in the middle of the spectrum (in terms of duration) between gravitational wave bursts and continuous waves. ANNs have been extensively studied and established [37]. Similarly, SVMs [38] and CSCs [29] have been broadly applied.

For the investigation of all three MLAs, the full set of data we had available was split into 90% for the training set and a 10% for the test set. The training efficiencies mentioned in Figs. 3–5 as well as in Sec. IV are all referring to how well the MLAs can detect signals in the “unknown” 10% of data that were left out of the training process. Before the selection of the training and test sets, all data were shuffled, and then the training and test sets were randomly selected.

A. Artificial neural network

The applicability of an artificial neural network was investigated as a pattern recognition algorithm [39] for the detection of r -mode gravitational waves. The aim was to train the ANN (Appendix A) in order to make it capable of recognizing the ft maps that contain r -mode signals and the

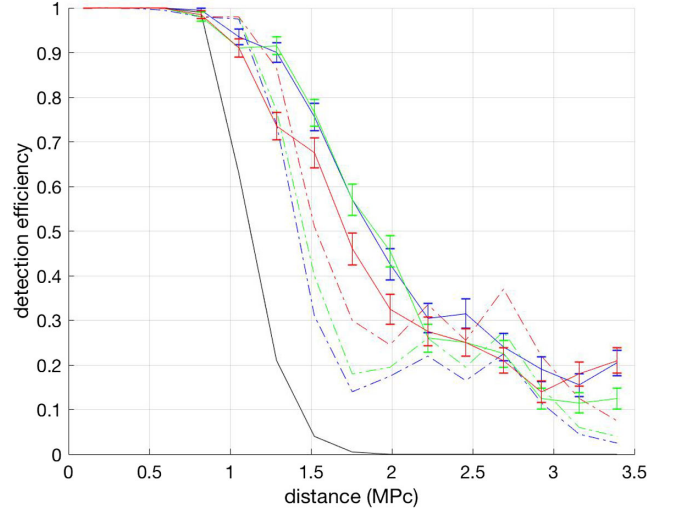


FIG. 3. These are the detection efficiencies for the ($f_o = 1500$ Hz, $\alpha = 0.1$) waveform binned every 0.2 kpc. This waveform was the most powerful signal we used in our previous sensitivity study [8]. The dashed-dotted black line corresponds to the SC algorithm sensitivity study at a FAP equal to 0.1%. The (blue, green, and red) dashed-dotted lines correspond to the detection efficiencies of the SC algorithm at FAPs of 4%, 5%, and 10%. The (blue, green, and red) solid lines correspond to the detection efficiencies of the ANN (at 4%), SVM (at 5%), and CSC (at 10%), respectively. This plot demonstrates that (when compared for the same FAP) the MLAs performance is at least as good as that of the SC algorithm. At the 50% FDR, the ANN shows an increase of $\sim 17\%$ in the detection distance—from ~ 1.45 (of the SC algorithm dashed-dotted blue line) to ~ 1.80 Mpc (of the solid blue line). The SVM shows an increase of $\sim 23\%$ —from ~ 1.50 (of the SC algorithm dashed-dotted green line) to ~ 1.85 Mpc of the solid-green line. The CSC shows an increase of $\sim 13\%$ —from ~ 1.50 (of the SC algorithm dashed-dotted red line) to ~ 1.70 Mpc of the solid red line.

ft maps that contain pure detector noise. If successful for the r -mode gravitational wave searches, the applicability of the ANNs may also be investigated in the stochastic pipeline for the detection of other long-lived gravitational wave transients.

The data matrix we used is a $2N \times d$ matrix where $N = 11$, 350 is the number of data (ft maps) with simulated detector noise. This is equal to the number ($N = 11$, 350) of simulated data (ft maps) with noise plus injected signals. According to Appendix D, the number of columns of the data matrix is chosen to be $d = 550$, and this is equal to the dimensionality of the input layer $d = 550$. The dimensionality of the hidden layer is $K = 50$, and the dimensionality of the output layer is $L = 2$. The “hidden” layer used “neurons” with the logistic sigmoid function (A1), while the output layer used neurons with the softmax activation function (A5), which is typically used in classification problems to achieve a 1-to- n output encoding [40,41].

Using the above data matrix, we performed a batch training of the neural network. After experimenting with

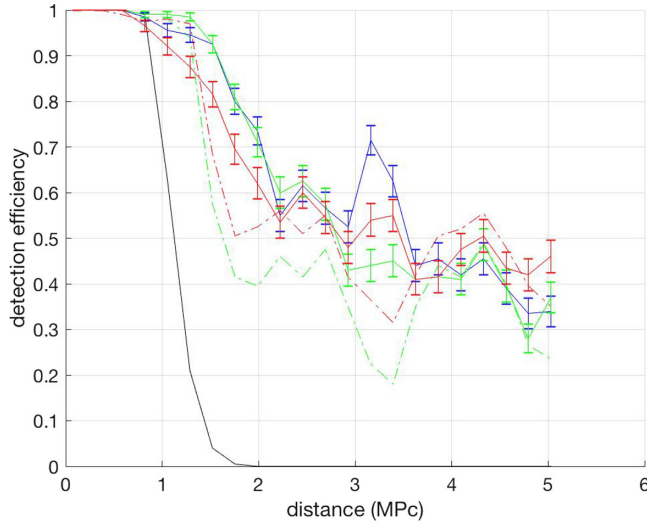


FIG. 4. These are the detection efficiencies for the ($f_o = 1500$ Hz, $\alpha = 0.1$) waveform binned every 0.2 kpc. This waveform was the most powerful signal we used in our previous sensitivity study [8]. The dashed-dotted black line corresponds to the SC algorithm sensitivity study at a FAP equal to 0.1%. The (blue, green, and red) dashed-dotted lines correspond to the detection efficiencies of the SC algorithm at FAPs of 18%, 22%, and 36%, respectively, though the detection efficiencies at 18% and 22% do not differ, so the lines overlap. The (blue, green, and red) solid lines correspond to the detection efficiencies of the ANN (at 18%), SVM (at 22%), and CSC (at 36%), respectively. At the 50% FDR, the ANN algorithm shows an increase of $\sim 137\%$ in the detection distance—from 1.50 (of the SC algorithm dashed-dotted blue line) to 3.55 Mpc (of the solid blue line). The SVM algorithm shows an increase of $\sim 83\%$ in the detection distance—from 1.50 (of the SC algorithm dashed-dotted green line) to 2.75 Mpc (of the solid green). The CSC shows an increase of $\sim 46\%$ in the detection distance—from ~ 2.60 (of the SC algorithm dashed-dotted red line) to ~ 3.40 Mpc (of the solid red line). The distance range covered in this set has a practical significance because it covers: (a) the distance of 3.5 Mpc at which the January 2014 supernova occurred in M82 and (b) the distance of 5 Mpc at which the supernova event rate in the Milky Way neighborhood is about one every one to two years.

various parameter populations, we used a learning rate of 0.02 and a momentum of 0.9. For the training, we used a batch version of the gradient descent as the optimization algorithm. To avoid overfitting and maintain the ability of the network to generalize, we used the “early stopping” technique. The results as shown in Figs. 3–5 demonstrate that the ANN algorithms performance is at least as good as that of the SC algorithm.

B. Support vector machine

The second MLA we trained is a support vector machine (SVM). This method is based on a well-formulated and mathematically sound theory [40]. The mathematical formulation of this algorithm is described in detail in Appendix B. In the SVM formulation, we treat the noise

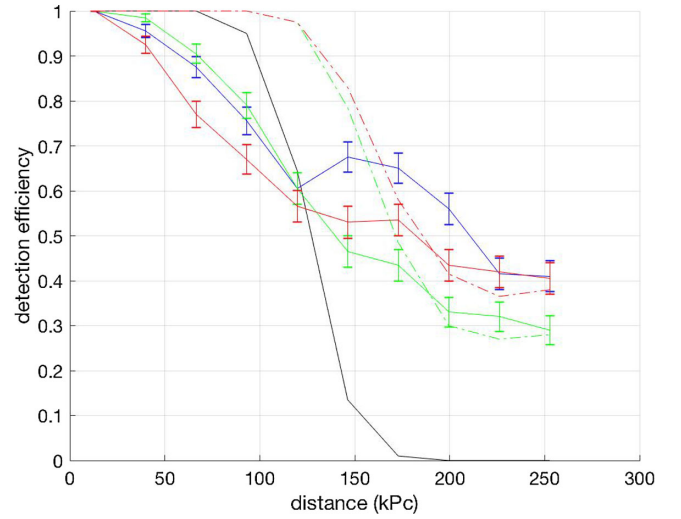


FIG. 5. These are the detection efficiencies for the ($f_o = 1100$ Hz, $\alpha = 0.01$) waveform binned every 0.2 kpc. In our previous sensitivity study this waveform was shown to be detectable only at distances that cover the Milky Way galaxy. This waveform is approximately monochromatic for the first 2500 s right after the signal starts getting emitted by the source. This is the time interval over which our sensitivity study was performed. The (blue, green, and red) dashed-dotted lines correspond to the detection efficiencies of the SC algorithm at FAPs of 18%, 22%, and 36%, respectively, though the detection efficiencies at 18% and 22% do not differ, so the lines overlap. The (blue, green, and red) solid lines correspond to the detection efficiencies of the ANN (at 18%), SVM (at 22%), and CSC (at 36%), respectively. At the 50% FDR, the ANN shows an increase of $\sim 18\%$ in the detection distance—from ~ 170 (of the SC algorithm dashed-dotted blue line) to ~ 210 kpc (of the solid blue line). The SVM shows a decrease of $\sim 24\%$ —from ~ 170 (of the SC algorithm dashed-dotted green line) to ~ 140 kpc of the solid green line. The CSC shows a decrease of $\sim 3\%$ —from ~ 185 kpc (of the SC algorithm dashed-dotted red line) to ~ 10 kpc of the solid red line.

fit maps as rows of a $N \times d$ matrix X'_1 and fit maps with r -mode injections as rows of a $N \times d$ matrix X'_2 as points in a d -dimensional space. The solution to the SVM optimization problem is to find the optimal hypersurface that would separate (and hence classify) the noise points from the injection points.

The above problem is a convex optimization problem and is formulated in Appendix B. It is solved using a state-of-the-art sequential minimal optimization solver, LIBSVM. In our case, we assumed that the classification problem is a nonlinear one; hence, we introduced the radial basis function (kernel) function (RBF). The constant γ was taken to be the default (by LIBSVM) value and equal to $1/d$. For the other parameter, C was estimated to have an optimal value in the range of 10^3 to 10^5 .

C. Constrained subspace classifier

The third algorithm we used is a CSC as explained in Appendix C. The separation of the two classes is based

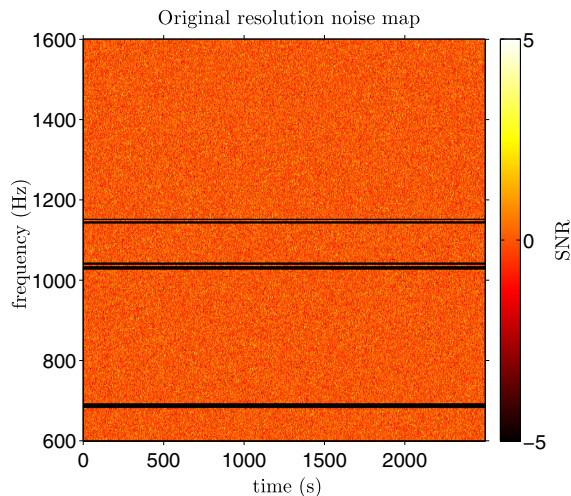


FIG. 6. This is one of the noise fit maps with the original resolution of 1000×5000 pixels. The pixels along the vertical axis correspond to 1 Hz each. The pixels along the horizontal axis correspond to 0.5 s each; hence, the total duration of the map is 2500 s. The frequency cuts are well-known seismic frequency bands and suspension vibration modes.

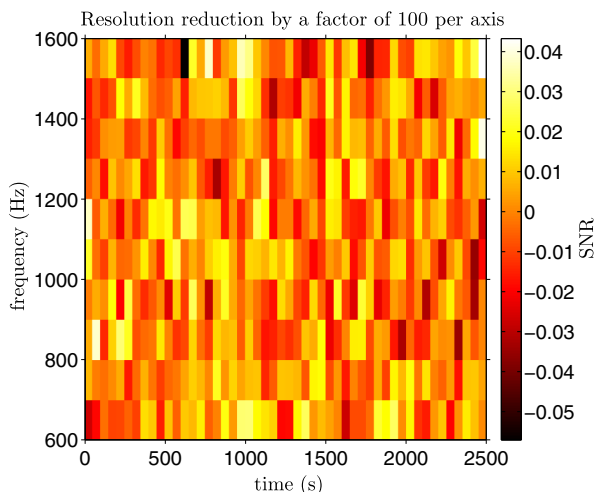


FIG. 7. The highest training efficiency for the MLAs was achieved with resolution reduction by a factor of 100 per axis, as seen in Fig. 10. This reduced 10×50 resolution ft map corresponds to the full resolution noise map in Fig. 6. For the resolution reduction, we used bicubic interpolation as provided by the MATLAB `IMRESIZE.m` function. The frequency cuts were substituted with zeros before reducing the resolution.

on projecting the noise data points (represented by the $N \times d$ matrix X'_1) onto a d_1 -dimensional subspace and similarly projecting the injection data points (represented by the $N \times d$ matrix X'_2) onto a d_2 -dimensional subspace, where $d_1 = d_2 < d$. Choosing the right trade-off between optimality and speed, we picked dimensionalities $d_1 = d_2 = 100$ for some cases (most powerful injections) and $d_1 = d_2 = 200$ for some others

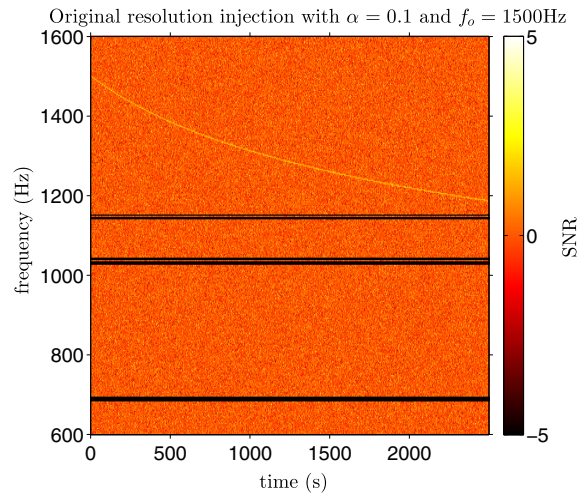


FIG. 8. This is an injection added to the noise fit map shown in Fig. 6. The waveform has parameters $\alpha = 0.1$ and $f_0 = 1500$ Hz. The duration of the injection is 2500 s and corresponds to a distance to the source of 117 kpc. Injections at longer distances are harder to see by eye in the original resolution maps. The contrast between signal pixels and noise pixels is higher in the reduced resolution maps as shown in Fig. 7. This makes it easier to see the injections in the reduced resolution maps rather than the full resolution ft maps.

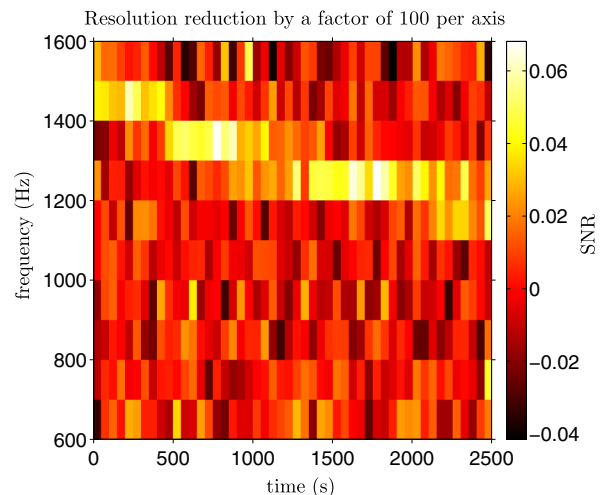


FIG. 9. This reduced 10×50 resolution ft map corresponds to the full resolution map in Fig. 8. Despite the 10 000 times reduced resolution as compared to the ft map of Fig. 8, the r -mode injection is still visible. It turns out that the reduced resolution ft maps increase the training efficiency for the MLAs, according to Fig. 10. However, for the parameter estimation algorithms, we use the full resolution ft maps.

(weakest injections). The constraint of the problem is the relative orientation of the two subspaces that is determined by the parameter C . This parameter was chosen after doing several runs, and it was found to take values between 10^4 and 10^5 .

IV. RESULTS AND DISCUSSION

When using the SC algorithm in Ref. [8], the FAP is easily controlled by adjusting the SNR threshold above which a ft map is considered to include an r -mode signal. This is not the case for the MLAs we used where the FAP is given after the training is performed as part of the training output. For this reason, to draw fair comparisons, we adjusted the FAP of the SC algorithm to match the output FAP of the MLAs. In Figs. 3–5, the results of the SC algorithm are compared with the results of the ANN, SVM, and CSC for the same FAP.

The first attempt to train all three MLAs was done with data produced with h taking values over the range of $10^{-24} \leq h \leq 10^{-21}$. Using this range of values for h , the MLAs did not outperform the SC algorithm. This was probably due to the fact that the number 11 350 of distinct signals used for the training was too small for the MLAs to achieve generalization, and hence the training efficiencies are too low. To avoid this, the next steps involved training the same number of data over smaller ranges of values of h .

In Fig. 3, we present the detection efficiency results for the SC algorithm and the three MLAs on the ($\alpha = 0.1, f_o = 1500$ Hz) waveform. The MLAs were trained with data produced with h taking values over the range of $10^{-23.7} \leq h \leq 10^{-23.2}$. This range of h values is such that it includes the distance d' at which the SC algorithm has a 50% detection efficiency (for this particular waveform). This implies that the MLAs were trained with signals injected at distances a little shorter than d' up to distances 1.5–2 times longer than d' . This particular choice resulted in an MLA performance that is at least as good as that of the SC algorithm. The training of the MLAs on this training set resulted in false alarm probabilities of 4%, 5%, and 10% for the ANN, SVM, and CSC, respectively.

At the 50% false dismissal rate (FDR), the ANN shows an increase of $\sim 17\%$ in the detection distance, from ~ 1.45 (of the SC algorithm dashed-dotted blue line) to ~ 1.80 Mpc (of the solid blue line). The SVM shows an increase of $\sim 23\%$, from ~ 1.50 (of the SC algorithm dashed-dotted green line) to ~ 1.85 Mpc of the solid green line. The CSC shows an increase of $\sim 13\%$, from ~ 1.50 (of the SC algorithm dashed-dotted red line) to ~ 1.70 Mpc of the solid red line.

In Fig. 4, we present the detection efficiency results for the SC algorithm and the three MLAs on the ($\alpha = 0.1, f_o = 1500$ Hz) waveform. The latter were trained with data produced with h taking values over the range of $10^{-24.0} \leq h \leq 10^{-23.7}$. This range of h values is such that all of the injection distances of the training set were higher than the distance d' , at which the SC algorithm has a 50% detection efficiency (for this particular waveform). The training of the MLAs with injections at distances longer than d' was done in order to push the limits of the MLAs and see how much (if at all) they can outperform the SC algorithm.

The training of the MLAs on this training set resulted in high false alarm probabilities of 18%, 22%, and 36% for the ANN, SVM, and CSC, respectively. At the 50% FDR, the ANN algorithm shows an increase of $\sim 137\%$ in the detection distance, from 1.50 (of the SC algorithm dashed-dotted blue line) to 3.55 Mpc (of the solid blue line). The SVM algorithm shows an increase of $\sim 83\%$ in the detection distance, from 1.50 (of the SC algorithm dashed-dotted green line) to 2.75 Mpc (of the solid green line). The CSC shows an increase of $\sim 46\%$ in the detection distance, from ~ 2.60 (of the SC algorithm dashed-dotted red line) to ~ 3.40 Mpc (of the solid red line). The distance range covered in this set has a practical significance because it covers: (a) the distance of 3.5 Mpc at which the January 2014 supernova occurred in M82 and (b) the distance of 5 Mpc at which the supernova event rate in the Milky Way neighborhood is about one every one to two years.

In Fig. 5, we present the detection efficiency results for the SC algorithm and the three MLAs on the ($\alpha = 0.01, f_o = 1100$ Hz) waveform. The MLAs were trained with data produced with h taking values over the range of $10^{-24} \leq h \leq 10^{-23.7}$. This range of h values is such that it includes the distance d' at which the SC algorithm has a 50% detection efficiency (for this particular waveform). This implies that the MLAs were trained with signals injected at distances a little shorter than d' up to distances 1.5–2 times longer than d' . This particular choice resulted in an MLA performance that is not as good as our results for the ($\alpha = 0.1, f_o = 1500$ Hz) waveform. The training of the MLAs on this training set resulted in false alarm probabilities of 18%, 22%, and 36% for the ANN, SVM, and CSC, respectively. At the 50% FDR, the ANN shows an increase of $\sim 18\%$ in the detection distance, from ~ 170 (of the SC algorithm dashed-dotted blue line) to ~ 210 kpc (of the solid blue line). The SVM shows a decrease of $\sim 24\%$, from ~ 170 (of the SC algorithm dashed-dotted green line) to ~ 140 kpc of the solid green line. The CSC shows a decrease of $\sim 3\%$, from ~ 185 (of the SC algorithm dashed-dotted red line) to ~ 10 kpc of the solid red line.

V. CONCLUSIONS

For computational efficiency, the most computationally expensive part of this study was the production of the one set of 11 350 noise ft maps and the three sets of 11,350 injection ft maps (each set requires up to 10 GB of memory and up to one week on a 50 node cluster). The three sets of injections examined the three different ranges of values for h (those corresponded to three different ranges of values for the distance). In practice, we will know the distance to the source, so we will have to produce only one set of injections that will be determined according to that distance.

For training/testing speeds, once we have the method (that is presented in this paper), the training of the CSC method requires 10 min, the training of the SVM method

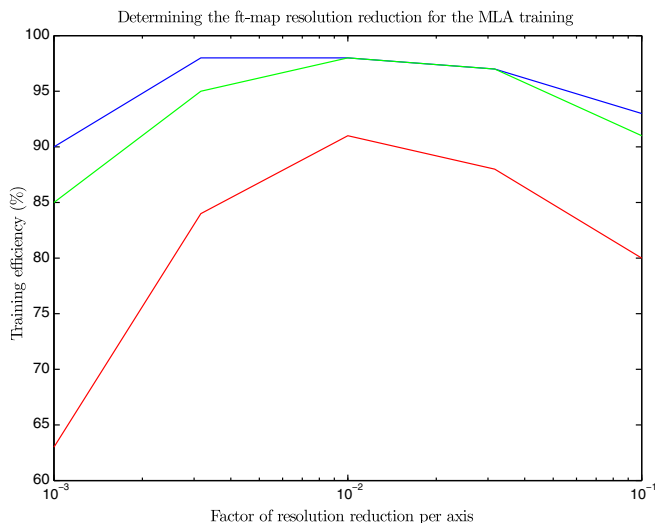


FIG. 10. Training efficiencies of ANN (blue), SVM (green), and CSC (red) vs the resolution reduction (per axis). For SVM and CSC, there is a clear peak at a resolution reduction factor of 10^{-2} . The ANN peak seems to be a little off, but for uniformity, we used 10^{-2} for all three MLAs. The training of all three MLAs was performed using the ($\alpha = 0.1$, $f_o = 1500$) waveform. No tests have been performed to verify the validity of these plots for other waveforms or other h value ranges.

requires about 30 min, while the training of the ANN method requires about 8 hr. After the training is done, the decision making about the presence of a signal or not takes about 2 s for 100 ft maps. The MLAs are much faster when it comes to the decision making process than the SC algorithm is (that takes up to 5 min for one ft map).

For the detection performance, Fig. 3 shows the detection efficiencies of MLAs that were trained with signals injected at distances a little shorter than the distance d' , at which the SC algorithm has a 50% success rate, up to distances 1.5–2 times longer than d' . When compared to Fig. 4 that shows the detection efficiencies of MLAs trained with signals injected at distances longer than d' (from 2.2 up to 4.3 times longer), we observe that the MLAs of Fig. 3 do not perform as well. In both figures, the MLAs outperform the SC algorithm by a factor of 1.2 (Fig. 3) up to a factor of 1.8 (Fig. 4). Training the MLAs with injections at distances shorter than d' was to ensure that the MLAs can detect signals injected at distances 0.7–0.8 that of d' , and training the MLAs with injections at distances longer than d' was done in order to push the limits of the MLAs and see how much (if at all) they can outperform the SC algorithm.

Low detection efficiency for the (0.01, 1100 Hz) waveform: We suspected that the low detection efficiencies for the second waveform (weakest signal) as seen in Fig. 5 are due to the resolution reduction factor of 10^{-2} we used. This resolution reduction factor was shown (in Fig. 10) to maximize the training efficiency for the strongest signals (Figs. 3 and 4) (0.1, 1500 Hz). We did not derive the

optimal value of this reduction factor for the weaker signals. In other words, we have not tested whether the weaker signals have maximum training efficiencies at a different resolution reduction than the one we used for the strongest signal. This needs further investigation.

For false alarm probabilities, in our study, FAPs of 4%–10% (for Fig. 3) and 18%–36% (for Figs. 4 and 5) are considered very high; however, a more carefully chosen training set may result in lower FAPs. The first suggestion would be to train the MLAs with a higher number of noise and injection ft maps. If that is not possible (due to data availability), we may train the MLAs with injections at distances over a range of (h^2) values that is smaller than those in the current training sets. Similarly, we can use smaller ranges of parameter values for α and f_o . We can also try to increase the ratio of noise maps over injection maps in the training set so that the MLAs may recognize the noise maps more efficiently. Specifically for the ANN, one way we may try to reduce the FAP is by exploring different topologies in the neural network architecture. For SVM and CSC, we may introduce a cost function to suppress the FAP to acceptable values.

For the most powerful signals, the false alarm probability is about 3%. This FAP is further decreased down to 0.3% when we used a number of noise maps two times higher than the number of signal-injection maps. However, this was done at the expense of the true positive probability (that decreased from 99% to 96%). Therefore, for signals from nearby sources the MLAs were shown to have a FAP comparable to that of the conventional SC algorithm.

For search optimization, there are many ways that we can further optimize the MLAs specifically designed for the search of r -mode gravitational radiation. One way is by customizing the ft-map resolution reduction. Instead of using bicubic interpolation, we may use a resolution reduction algorithm specifically designed for the r -mode signals so that the averaging is done along the r -mode signal curves. Since the r -mode search is a targeted search (using a supernova electromagnetic or neutrino trigger), the distance to the source can be estimated with an accuracy of 10% – 15% [17,19]. This distance range can then be used to produce injection ft maps with which the MLAs will be trained. In this way, the training can be optimized for the distance of the detectors to the external trigger.

For the search constraints, our current method is specifically designed for r -mode gravitational wave searches. A different signal (e.g., gravitational waves sourcing from other neutron star oscillation modes) would require their own training set produced over the specific model parameter values. This is a quite different approach than that of the SC algorithm that is generically designed for the detection of any type of signal. Our current method involves the production of at least 10 000 ft maps (that may be overlapping), and any amount of data that will not be enough for the production of this many ft maps will limit

the sensitivity of the search. At the same time, the higher the number of the ft maps used for training, the more we may increase the training efficiencies of the MLAs.

For resolution reduction, we did not examine robustness of the resolution reduction results on other signals (with different α and f_o); therefore, this method may have to be repeated for different r -mode waveforms as well as different long duration gravitational wave transients. However, if we use Google's TENSORFLOW [54], based on graphic processing units, we may be able to train computationally expensive algorithms, such as region convolutional neural networks, without needing to reduce the resolution of the ft-map.

Despite the high FAP, the MLAs have an extremely important advantage over the SC algorithm. After performing the training stage of the MLAs, the testing stage is lightning fast (testing with the SC algorithm may take several tens of minutes vs fractions of a second that are needed by MLAs). This implies that we can use the MLAs as part of an investigative stage in the pipeline that would be able to provide very fast and solid triggers for further, and more intense, investigation.

For pipeline suitability, the ANN, SVM, and CSC (and very likely other machine learning algorithms not tested yet) are a suitable class of decision making algorithms in the search for not only r -mode gravitational waves but in the search for long transient gravitational waves in general. The results in this paper demonstrate that the stochastic pipeline would benefit from utilizing machine learning algorithms for determining the presence of a signal or not.

The aim of this paper was not to demonstrate how we can use MLAs in order to make a detection announcement. Instead, our aim was to perform a preliminary investigation on how we can use raw data taken by the LIGO detectors, preprocess it, and feed it in three separate MLAs. The purpose of this paper was fulfilled since we were able to obtain preliminary results that can encourage us (as well as other groups) for further investigation, including addressing the issue of high false alarm probability.

Suggestions for Future Work: Future developments include optimization of the current methods as well as the use of other supervised machine learning algorithms such as random forests [42]. Random forests can deal with the high dimensionality of our data by revealing features that contribute very low information to our analysis, which can be discarded prior to classification. With respect to the ANN, we plan to train a deep convolutional neural network [43], which appears to be very promising for image classification.

For training with more data, out of the 5 million columns (of our 22,700 x 5,000,000 matrix), only 22,700 are linearly independent (row rank = column rank). This means that the information we can extract from the columns is limited by the number of data rows we produce. This suggests that upon production of a higher number of data rows (ft maps) the MLAs will be able to extract more

information from the data matrix, and quite possibly the training efficiencies will improve.

ACKNOWLEDGMENTS

This work has been supported by Grants No. PHY 0855313 and No. PHY 1205512 from the NSF.

APPENDIX A: ARTIFICIAL NEURAL NETWORKS

After resolution reduction, the original $2N \times D$ data matrix gets reduced to a $2N \times d$ data matrix, X' . The latter will be presented as input into a feed-forward neural network with an input layer of dimensionality d . For the training of the ANN, we randomly picked 90% of the first N (injection data) rows and also 90% of the second N (noise data) rows. The other 10% of the (injection and noise) rows was used to determine the training efficiency of the trained algorithm. The ANN had one hidden layer with a number of nodes (neurons) equal to K and an output layer with two neurons that would “fire” for “signal” or “no signal.” The hidden layer used neurons with the logistic sigmoid function [41]

$$\sigma(a_j) = \frac{1}{1 + \exp(-a_j)}, \quad (\text{A1})$$

where a_j ($j = 1, 2, \dots, d$) are the values presented at one neuron in the hidden layer. The purpose of the hidden layer is to allow for nonlinear combinations of the input values to be forwarded to the output layer. These combinations in the hidden layer carry forward “features” from the input to the output layer that would not be possible to be extracted from each individual neuron in the input layer, enabling nonlinear classification. The number of hidden layers and hidden neurons was chosen, as is typically done, after experimentation with various ANN architectures, aiming to enhance the accuracy, the robustness, and the generalization ability of the ANN, along with the training efficiency and feasibility.

Starting from the first ft map in the data matrix X , i.e., starting from the row vector x_1 where

$$x_1 = \{x_{ij} | i = 1 \text{ and } j = 1, 2, \dots, d\}, \quad (\text{A2})$$

we have d values that are fed into the input layer of the neural network. These values are then nonlinearly combined in each hidden neuron to get K many output values forwarded to the output layer, given by

$$x'_{1k} = \sigma \left(\sum_{j=1}^d w_{kj}^{(1)} x_{1j} + w_{k0}^{(1)} \right), \quad (\text{A3})$$

where $k = 1, 2, \dots, K$ is the index corresponding to each neuron in the hidden layer and the superscript (1) represents the hidden layer. The parameters w_{kj} are called the weights,

while the parameters w_{k0} are called the biases of the neural network.

The “output” layer used neurons with the soft-max activation function, which is typically used in classification problems to achieve a 1-to- n output encoding [40]. In particular, the softmax function rescales the outputs in order for all of them to lie within the range $[0, 1]$ and to sum up to 1. This is done by normalizing the exponential of the input b_k to each output neuron over the exponential of the inputs of all neurons in the output layer:

$$\text{soft-max}(b_k) = \frac{\exp(b_k)}{\sum_k (\exp(b_k))}. \quad (\text{A4})$$

When the values from Eq. (A3) are presented in the output layer, we get the result

$$x''_l = \text{soft-max} \left(\sum_{k=1}^K w_{lk}^{(2)} x'_{1k} + w_{l0}^{(2)} \right) \quad (\text{A5})$$

(where $l = 1, 2$) as the output value in the single neuron of the output layer. Equation (A5) shows how information in the output layer is a function of the information in the hidden layer. This is similar to Eq. (A3) showing how information in the hidden layer is a function of the information in the input layer. These two equations represent the *forward propagation* of information in the neural network.

Equation (A5) also shows that a neural network is a nonlinear function, $\mathcal{F}$, from a set of input variables $\{x_i\}$ such that $i \in \{1, 2, \dots, 2N\}$ as defined by Eq. (A2) i.e., x_i are row vectors of the matrix X' to a set of output variables $\{x''_l\}$ such that $l \in \{1, 2\}$ i.e., the output layer has dimensionality equal to $L = 2$ (2 neurons: one fires for noise and the other fires for injection). To merge the weights $w_{kj}^{(1)}$ and biases $w_{k0}^{(2)}$ into a single matrix (and similarly merge with the weights $w_{lk}^{(2)}$ and biases $w_{l0}^{(2)}$), we need to redefine x_1 as given by Eq. (A2) to

$$x_1 = \{x_{ij} | i = 1 \text{ and } j = 0, 1, 2, \dots, d \text{ and } x_{10} = 1\} \quad (\text{A6})$$

and similarly redefine all row vectors of X' as well as all the output row vectors from the hidden layer. Then, the nonlinear function $\mathcal{F}$ is controlled by a $(K+1) \times (d+1)$ matrix $\mathbf{w}^{(1)}$ and a $2 \times (K+1)$ matrix $\mathbf{w}^{(2)}$ of adjustable parameters. Training a neural network corresponds to calculating these parameters.

Numerous algorithms for training ANN exist [41] and in general can be classified as being either sequential or batch training methods: (i) Sequential (or “online”) training: A “training item” consists of a single row (one ft map) of the data matrix. In each iteration, a single row is passed through the network. The weight and bias values are adjusted for every training item based on the difference between computed outputs and the training data target outputs.

(ii) Batch training: A training item consists of the matrix X' (all $2N$ rows of the data matrix). In each iteration, all rows of X' are successively passed through the network. The weight and bias values are adjusted only after all rows of X' have passed through the network. In general, batch methods perform a more accurate estimate of the error (i.e., the difference between the outputs and the training data target outputs), and hence (with sufficiently small learning rate [44]) they lead to a faster convergence. As such, we used a batch version of gradient descent as the optimization algorithm. This form of the algorithm is also known as “back propagation” because the calculation of the first (or hidden) layer errors is done by passing the layer 2 (or output) layer errors back through the $w^{(2)}$ matrix. The back-propagation gradient descent for ANNs in batch training is summarized as follows:

Algorithm 1. Gradient descent for ANN.

```

1. Initialize  $w$  (and biases) randomly.
while error on the validation set satisfies certain criteria do
for  $i = 1 : 2N$  do
2. Feed-forward computation of the input vector  $x_i$ .
3. Calculate the error at the output layer.
4. Calculate the error at hidden layer.
5. Calculate the mean error.
6. Update  $w$  of the output layer.
7. Update  $w$  of the hidden layer.
end for
end while

```

Out of the 90% of the data that was (randomly) chosen for the training, 10% of that was used as a validation set. The latter is used in the early stopping technique that is used to avoid overfitting and maintain the ability of the network to generalize. Generalization is the ability of a trained ANN to identify not only the points that were used for the training but also points in between the points of the training set. For each iteration, the detection efficiency of the ANN is tested on the validation set. When the error on the validation set drops by less than 10^{-3} for two consecutive iterations, then we do the early stopping, and the training is stopped.

The learning rate of the gradient-descent algorithm determines the rate at which the training of the network is moving toward the optimal parameters. It should be small enough not to skip the optimal solution but large enough so that the convergence is not too slow. A crucial challenge for the algorithm is not to converge to local minima. This can be avoided by adding a fraction of a weight update to the next one. This method is called the “momentum” of the training of the network. Adding momentum to the training implies that for a gradient of constant direction the size of the optimization steps will increase. As such, the momentum should be used with a relatively small learning rate in order not to skip the optimal solution.

APPENDIX B: SUPPORT VECTOR MACHINE

The second MLA we trained is a SVM. This method gained popularity over the ANNs because it is based on well-formulated and mathematically sound theory [40]. In the following paragraphs, we give a brief introduction to the SVM mathematical formulation.

In the SVM formulation all data are represented as points in a vector space. Each noise ft-map as well as each r-mode injection ft-map are treated as d -dimensional points in a d -dimensional vector space (where d is the number of pixels in the corresponding ft-map). The idea behind the formulation of the SVM optimization problem is to find the optimal hypersurface that would separate (and hence classify) the noise points from the injection points. For this discussion, we will need the following definitions:

Definition 1: The distance of a point x_i to a flat hypersurface $\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$ is given by

$$d_{x_i}(w, b) = z_i \times (\langle w, x_i \rangle + b), \quad (\text{B1})$$

where w is a unit vector perpendicular to the flat hypersurface, b is a constant, and $z_i = +1$ for $\langle w, x_i \rangle + b > 0$ and $z_i = -1$ for $\langle w, x_i \rangle + b < 0$. The index i (in x_i) takes values from the set $\{1, 2, 3, \dots, 2N\}$. In the following discussion, each point x_i that lies above the hypersurface pairs with a value $z_i = 1$, and each point x_i that lies below the hypersurface pairs with a value of $z_i = -1$.

Definition 2: The “margin,” $\gamma_S(w, b)$, of any set, $\mathcal{S}$, of vectors is defined as the minimum of the set of all distances $\mathcal{D} = \{d_{x_i}(w, b) | x_i \in \mathcal{S}\}$ from the hypersurface $\mathcal{H}$. For the purpose of our discussion, the set $\mathcal{S}$ is the union of the set of all noise points and the set of all injection points.

For definition 3, we assume that a training set consists of points x_i with each one belonging to one of two distinct data classes denoted by $y_i = 1$ (for one class) and $y_i = -1$ (for the other class). We may further assume that the set of all noise points belongs to the class represented by $y_i = -1$ while the set of all injection points belongs to the class represented by $y_i = +1$.

Definition 3: A training set $\{(x_1, y_1), \dots, (x_n, y_n) | x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}\}$ is called “separable” by a hypersurface $\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$ if both a unit vector w ($\|w\| = 1$) and a constant b exist so that the following inequalities hold,

$$\langle w, x_i \rangle + b \geq \gamma_S \quad \text{if } y_i = +1 \quad (\text{B2})$$

$$\langle w, x_i \rangle + b \leq -\gamma_S \quad \text{if } y_i = -1, \quad (\text{B3})$$

where $\mathcal{S} = \{x_i | i = 1, 2, \dots, n\}$ and γ_S is given by Definition 2.

For the purpose of our discussion, d is the dimensionality of the points x_i (this dimensionality corresponds to the number of pixels in each ft map), and $n = 2N$ is the number

of our (ft-maps) data points. Using the fact that the hypersurface is defined up to a scaling factor c , i.e., $\mathcal{H} = \{x | \langle cw, x \rangle + cb = 0\}$, we can take c such that $c\gamma_S = 1$, and hence we can rewrite Eqs. (B2) and (B3) as

$$y_i \times (\langle cw, x_i \rangle + cb) \geq 1 \quad \text{for all } i = 1, 2, \dots, n. \quad (\text{B4})$$

Defining $w' = cw$, i.e., $\|w'\| = c$, and dividing Eq. (B4) by c , we get

$$y_i \times \left(\left\langle \frac{w'}{\|w'\|}, x_i \right\rangle + b \right) \geq \frac{1}{\|w'\|} \quad \text{for all } i = 1, 2, \dots, n. \quad (\text{B5})$$

1. Formulation of the SVM optimization problem

Given a training set, that is, a data matrix $X' = \begin{pmatrix} X'_1 \\ X'_2 \end{pmatrix}$, X'_1 being a $N \times d$ matrix representing the noise points and X'_2 being a $N \times d$ matrix representing the injection points, we want to find the “optimal separating hypersurface” (OSH), that separates the row vectors of X'_1 from the row vectors of X'_2 . According to Definition 3, this translates to maximizing the margin γ_S . In other words, we want to find a unit vector w and a constant b that maximize $\frac{1}{\|w'\|}$. Therefore, the SVM optimization problem can be expressed as follows,

$$\min_{w, b} \frac{1}{2} \|w'\|^2 \quad \text{subject to} \quad (\text{B6})$$

$$1 - y_i \times (\langle w', x_i \rangle + b') \leq 0 \quad \text{for all } i = 1, 2, \dots, n, \quad (\text{B7})$$

where $b' = cb$. This is a quadratic (convex) optimization problem with linear constraints and can be solved by seeking a solution to the Lagrangian problem dual to Eqs. (B6) and (B7).

Before formulating the Lagrangian dual, we introduce the “slack variables,” ξ_i ($i = 1, 2, \dots, n$), that are used to relax the conditions in Eq. (B4) and account for outliers or “errors”. Instead of solving Eq. (B6), we seek a solution to

$$\min_{w, b} \frac{1}{2} \|w'\|^2 + C \sum_{i=1}^n \xi_i \quad \text{subject to} \quad \xi_i \geq 0$$

$$\text{and } 1 - y_i \times (\langle w', x_i \rangle + b') - \xi_i \leq 0$$

$$\text{for all } i = 1, 2, \dots, n. \quad (\text{B8})$$

The slack variables ξ_i measure the distance of a point that lies on the wrong side of its “margin hypersurface.” Using the Lagrange multipliers

$$\alpha_i \geq 0 \quad \text{and} \quad \beta_i \geq 0, \quad (\text{B9})$$

the Lagrangian dual formulation of Eq. (B8) is to maximize the following Lagrangian:

$$\begin{aligned} \mathcal{L}(w', b, \xi_i, \alpha, \beta) = & \frac{1}{2} \|w'\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \beta_i \xi_i \\ & + \sum_{i=1}^n \alpha_i (1 - y_i \times (\langle w', x_i \rangle + b) - \xi_i). \end{aligned} \quad (\text{B10})$$

Using the stationary first order conditions for w' , b , and ξ_i ,

$$\frac{\partial \mathcal{L}}{\partial w'_j} = w'_j - \sum_{i=1}^n \alpha_i y_i x_{ij} = 0, \quad \forall j = 1, 2, \dots, d, \quad (\text{B11a})$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^n \alpha_i y_i = 0, \quad (\text{B11b})$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0, \quad \forall i = 1, 2, \dots, n \quad (\text{B11c})$$

(where x_{ij} is the j th entry of the x_i data point), the Lagrangian dual as given in expression (B10) can be reexpressed only in terms of the α_i Lagrange multipliers, as follows,

$$\mathcal{L}(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_j, x_i \rangle, \quad (\text{B12})$$

and hence we can evaluate the α_i Lagrange multipliers by solving the following optimization problem:

$$\max_{\alpha_i} \mathcal{L}(\alpha_i) \quad \text{subject to} \quad \sum_{i=1}^n \alpha_i y_i = 0, \quad (\text{B13a})$$

$$0 \leq \alpha_i \leq C, \quad \forall i = 1, 2, \dots, n. \quad (\text{B13b})$$

Defining $G_{ij} = y_i y_j x_j^\top x_i$, problem (B13a)–(B13b) is equivalently expressed as

$$\min_{\alpha_i \in \mathcal{R}^n} \frac{1}{2} \alpha^\top G \alpha - e^\top \alpha \quad (\text{B14a})$$

$$\text{subject to } y^\top \alpha = 0 \quad (\text{B14b})$$

$$\text{and } 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n, \quad (\text{B14c})$$

where $e^\top$ is an n -dimensional row vector equal to $e^\top = (1, 1, \dots, 1)$ and (B14c) is derived from (B11c) together with (B9).

Since the objective function in Eq. (B14) is quadratic and all the constraints are affine, the problem defined by these equations is a quadratic optimization problem. Using the fact that (by construction) the sum of all the entries of G can

be written as a sum of squares and also using that $\alpha_i \geq 0$, we can see that G is positive semidefinite, which implies that the problem is convex. Convex problems offer the advantage of global optimality; that is any local minimum is also the global one. Several methods have been proposed for solving such problems including primal, dual, and parametric algorithms [45].

After solving the optimization problem defined by expressions (B14a)–(B14c), i.e., after evaluating all the α_i ($i = 1, 2, \dots, n$), we can find the vector w using (B11). The constant b can be found by using the Karush-Kuhn-Tucker complementarity conditions [46],

$$\alpha_i \{-1 + y_i \times (\langle w', x_i \rangle + b') + \xi_i\} = 0 \quad (\text{B15a})$$

$$\beta_i \xi_i = 0, \quad (\text{B15b})$$

along with Eq. (B11c). For any α_i satisfying $0 < \alpha_i < C$, Eq. (B11c) implies that $\beta_i > 0$, and hence (B15b) implies that $\xi_i = 0$. Consequently, we can use the x_i corresponding to the aforementioned α_i to solve Eq. (B15a) for b' .

The vector w' and the constant b' define a hypersurface defined by $\langle w', x_i \rangle + b' = 0$. During the testing phase, a new data point, x_i , is classified according to

$$\text{class}(x_i) = \text{sgn}(\langle w', x_i \rangle + b'). \quad (\text{B16})$$

For $\text{class}(x_i) = -1$, we classify the x_i point as noise, and for $\text{class}(x_i) = +1$, we classify the x_i point as injection.

We choose to solve the convex quadratic problem as defined in Eq. (B14) with sequential minimal optimization (SMO) [48]. SMO modifies only a subset of dual variables α_i at each iteration, and thus only some columns of G are used at any one time. A smaller optimization subproblem is then solved, using the chosen subset of α_i . In particular, at each iteration, only two Lagrange multipliers that can be optimized are computed. If a set of such multipliers cannot be found, then the quadratic problem of size 2 is solved analytically. This process is repeated until convergence. The integrated software for support vector classification (LIBSVM) [47] is a state-of-the-art SMO-type solver for the quadratic problem found in the SVM formulation. SMO outperforms most of the existing methods for solving quadratic problems [49]. Hence, we choose to use it for training the SVM, using the LIBSVM routine SVMTRAIN.

2. Nonlinear SVM

The soft margins ξ_i can only help when data are “reasonably” linearly separable. However, in most real world problems, data are not linearly separable. To deal with this issue, we transform the data into a feature (Hilbert) space, $\mathcal{H}$ (a vector space equipped with a norm and an inner product), where a linear separation might be possible due to the choice of the dimensionality

of $\mathcal{H}$, $\dim(\mathcal{H}) \geq \dim(\mathbb{R}^d)$. The transformation is represented by

$$\Phi: \mathbb{R}^d \rightarrow \mathcal{H} \quad \text{such that } \Phi(x_i) \in \mathcal{H}. \quad (\text{B17})$$

From Eqs. (B12) and (B17), we see that the nonlinear SVM formulation depends on the data only through the dot products $\Phi(x_i) \cdot \Phi(x_j)$ in $\mathcal{H}$. These dot products are generated by a real-valued “comparison function” (called the “kernel” function) $k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ that generates all the pairwise comparisons $K_{ij} = k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. We represent the set of these pairwise similarities as entries in an $n \times n$ matrix, K . The use of a kernel function implies that neither the feature transformation Φ nor the dimensionality of $\mathcal{H}$ are required to be explicitly known.

Definition 4: A function $k: \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}$ is called a positive semi-definite kernel if and only if it is: symmetric and (ii) positive semi-definite.

A function k is symmetric if $k(x_i, x_j) = k(x_j, x_i)$ for any $x_i, x_j \in \mathcal{L}$. A function is positive semi-definite if

$$c^\top K c = \sum_{i=1}^n \sum_{j=1}^n c_i c_j k(x_i, x_j) \geq 0 \quad (\text{B18})$$

for any $x_i, x_j \in \mathcal{L}$, any $c \in \mathbb{R}^n$ ($c_i, c_j \in \mathbb{R}, i = 1, 2, \dots, n$) and the $n \times n$ matrix K has elements $K_{ij} = k(x_i, x_j)$ where $i, j \in \{1, 2, \dots, n\}$.

The nature of the data we are using strongly suggests that our data points are not linearly separable in the original feature space. Therefore, we choose to solve the dual formulation as given by Eq. (B14) where G is now defined by $G_{ij} = y_i y_j k(x_i, x_j)$ so that we can use the “Kernel trick.” Solving the dual problem has the additional advantage of obtaining a sparse solution; most of the α_i will be zero (those that satisfy $0 < \alpha_i \leq C$ are the support vectors that define the hypersurface). For the purpose of our study, we used the RBF kernel defined by

$$k(x_i, x_j) = \exp\left(-\gamma \frac{\|x_i - x_j\|^2}{\sigma^2}\right), \quad (\text{B19})$$

where γ is a free parameter and σ is the standard deviation of the x_i that is equal to 1 due to normalization. Typically, the free parameters (γ and C) are calculated by using the cross-validation (grid search) method on the data set, meaning that we split the data set into several subsets and the optimization problem is solved on each subset with different parameter values for γ and C . We then choose the parameter values that give the lowest minimum value of the objective function. In our study, we chose the default (by LIBSVM) value of γ that was set equal to $\gamma = 1/d$. To determine the value of the parameter C , we plotted training efficiencies against several values of C . We determined that C should be in the range of 10^4 – 10^5 . All experiments with

SVM are conducted with 90/10 split on data, where 90% of the data is randomly selected for training and the remaining 10% is used for testing.

Using the Kernel trick, we substitute x_i with $\Phi(x_i)$ in Eqs. (B8)–(B16). Then, Eq. (B12) is reexpressed as

$$\mathcal{L}(\alpha_i) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \Phi(x_j), \Phi(x_i) \rangle. \quad (\text{B20})$$

After solving (B14), the α_i ($i = 1, 2, \dots, n$) are substituted in (B11), which we solve for w'_j to get

$$w'_j = \sum_{i=1}^n \alpha_i y_i \Phi_j(x_i) \quad \forall j = 1, 2, \dots, d, \quad (\text{B21})$$

where $\Phi_j(x_i)$ is the j th entry of the $\Phi(x_i)$ transformed data point. Since the transformation Φ is not obtained directly, we never calculate the w' vector explicitly. Nevertheless, we can substitute expression (B21) in (B15a) and solve the latter for b' (when $\xi_k = 0$ and $\alpha_k \neq 0$) as follows,

$$b' = 1 - y_k \times \sum_{i=1}^n \alpha_i y_i \langle \Phi(x_i), \Phi(x_k) \rangle, \quad (\text{B22})$$

where this result should be independent of which k we use. Having the expression (B21) for the vector w' and the expression (B22) for the constant b' , we can classify a new data point during the testing phase according to

$$\text{class}(x_i) = \text{sgn}(\langle w', \Phi(x_i) \rangle + b'). \quad (\text{B23})$$

From (B23), we see that we are able to calculate the new (flat) hypersurface in the new feature (Hilbert) space simply through inner products of $\langle \Phi(x_i), \Phi(x_j) \rangle$.

APPENDIX C: CONSTRAINED SUBSPACE CLASSIFIER

The idea in the CSC method is similar to the idea used in the SVM. In the latter, the target was to separate the noise points (or noise vectors) from the injection points (or injection vectors) using a hypersurface. In the CSC method, the idea is to project the noise vectors, rows of X'_1 ($N \times d$ matrix), onto a d_1 -dimensional subspace S_1 (of dimensionality $d_1 < d$) of the d -dimensional space and also project the injection vectors, rows of X'_2 (also a $N \times d$ matrix), onto a subspace S_2 (of dimensionality $d_2 < d$). That is, we seek to find two (optimal) subspaces such that we can classify data (ft-map) points according to their distance from each subspace: points closer to the subspace S_1 are classified as “noise points,” and points closer to the subspace S_2 are classified as injection points.

The optimality of the choice of each subspace depends on the chosen basis vectors, the chosen dimensionalities, d_1 and d_2 , of each subspace, as well as the relative orientation

between the two subspaces. Each choice corresponds to a given variance of the projected data: the closer the variance of the projected points is to the variance of the original data set, the more optimal the subspaces are considered.

1. Projection operator

Let S be a data space of dimension equal to the number of features, d , of the selected data set (for our study, d is the dimensionality of the ft maps after the resolution reduction). We can always find an orthonormal basis for S (using the Gram-Schmidt process) given by

$$U_d = \{u_1, u_2, \dots, u_d\} \quad \text{with} \quad u_i \in \mathbb{R}^d \quad \forall i = 1, 2, \dots, d, \quad (\text{C1})$$

i.e., $U_d \in \mathbb{R}^{d \times d}$. We seek to find a subspace of S of dimension $d_1 < d$. Since reducing the dimensionality brings the data points closer to each other, thus reducing the variance, we try to reduce the number of features from d to d_1 while trying to maintain the variance of the data distribution as high as possible.

To achieve the dimensionality reduction, we seek to find a projection operator that projects the data points from $\mathbb{R}^d$ to a (dimensionally reduced) subspace $\mathbb{R}^{d_1}$ of orthonormal basis given by

$$U_{d_1} = \{u_1, u_2, \dots, u_{d_1}\} \quad \text{with} \quad u_i \in \mathbb{R}^d \quad \forall i = 1, 2, \dots, d_1, \quad (\text{C2})$$

i.e., $U_{d_1} \in \mathbb{R}^{d \times d_1}$. By definition, the projection operator is given by

$$P = Q(Q^\top Q)^{-1}Q^\top \quad (\text{C3})$$

and projects a vector onto the space spanned by the columns of Q . Therefore, we may take the columns of Q to be the orthonormal vectors given in (C2), that is $Q = U_{d_1}$. In that case, Eq. (C3) becomes

$$P = U_{d_1}(U_{d_1}^\top U_{d_1})^{-1}U_{d_1}^\top, \quad (\text{C4})$$

which is the projection operator onto the space spanned by the column vectors of U_{d_1} .

Since Eq. (C1) is an orthonormal basis for $\mathbb{R}^d$, then $U_{d_1}^\top U_{d_1} = I_{d_1}$. Therefore, the expression of the projection operator that can project the (data) vectors in $\mathbb{R}^d$ onto its subspace $\mathbb{R}^{d_1}$ is given by

$$P = U_{d_1} U_{d_1}^\top. \quad (\text{C5})$$

In case $d_1 = d$, then $P = U_d U_d^\top$. Since U_d is a square matrix of which the columns are orthonormal, this implies that its rows are also orthonormal. Orthonormality of the

columns of U_d implies $U_d^\top U_d = I_d$ (i.e., $U_d^\top$ is the left inverse of U_d), and orthonormality of the rows of U_d implies $U_d U_d^\top = I_d$ (i.e., $U_d^\top$ is the right inverse of U_d). Therefore, for the special case that $d_1 = d$, we have that $U_d^\top$ is the inverse of U_d or

$$U_d^\top = U_d^{-1}. \quad (\text{C6})$$

2. Principal component analysis

To introduce principal component analysis (PCA), we will use the definition of the data matrix X'_1 as a $N \times d$ noise matrix as well as the definition of X'_2 as a $N \times d$ injection data matrix. Using the projection operator as given by expression (C5), we want to project the ft maps of X'_1 in a subspace $\mathbb{R}^{d_1}$ of $\mathbb{R}^d$ ($d_1 < d$). Let x_i be the original $1 \times d$ row vector in $\mathbb{R}^d$. We project the column vector $x_i^\top$ onto $\mathbb{R}^{d_1}$, thus defining $\tilde{x}_i^\top = U_{d_1} U_{d_1}^\top x_i^\top$. Then, the norm of the difference between the original and the projected (column) vectors can be expressed as

$$\|x_i^\top - \tilde{x}_i^\top\| = \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|, \quad (\text{C7})$$

where $U_{d_1} \in \mathbb{R}^{d \times d_1}$. In PCA, we want to find the subspace $\mathbb{R}^{d_1}$ such that

$$\sum_{i=1}^n \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|^2 \text{ is minimized} \quad \text{subject to } U_{d_1}^\top U_{d_1} = \mathcal{I}_{d_1}. \quad (\text{C8})$$

This subspace $\mathbb{R}^{d_1}$ is defined as the d_1 -dimensional hyper-surface that is spanned by the (reduced) orthonormal basis $\{u_1, u_2, u_3, \dots, u_{d_1}\}$; i.e., finding such a basis is equivalent to defining the subspace $\mathbb{R}^{d_1}$.

Using the definition of the Frobenius norm for a $m \times n$ matrix A ,

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2} = \sqrt{\text{trace}(A^* A)}, \quad (\text{C9})$$

where A^* is the conjugate transpose of A , we get

$$\sum_{i=1}^n \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|_F^2 = \text{tr}\{X_1'^\top X_1'(\mathcal{I} - U_{d_1} U_{d_1}^\top)\}, \quad (\text{C10})$$

where $X'_1 \in \mathbb{R}^{n \times d}$ (where $n = 2N$). Thus, the optimization problem in Eq. (C8) reduces to [50]

$$\min_{U_{d_1}} \text{tr}\{X_1'^\top X_1'(\mathcal{I} - U_{d_1} U_{d_1}^\top)\} \quad \text{subject to } U_{d_1}^\top U_{d_1} = \mathcal{I}_{d_1}. \quad (\text{C11})$$

Since $\text{tr}\{X_1'^\top X_1'\}$ is a constant, the optimization problem can be rewritten as

$$\max_{U_{d_1}} \text{tr}\{U_{d_1}^\top X_1'^\top X_1' U_{d_1}\} \quad \text{subject to } U_{d_1}^\top U_{d_1} = \mathcal{I}_{d_1}. \quad (\text{C12})$$

To solve Eq. (C12), we define the Lagrangian dual problem by

$$\begin{aligned} \mathcal{L}(U_{d_1}, \lambda_{ij}) = & \text{tr}(U_{d_1}^\top X_1'^\top X_1' U_{d_1}) \\ & - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left(\sum_{k=1}^d U_{jk}^\top U_{ki} - \delta_{ji} \right) \end{aligned}$$

where $\delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j. \end{cases} \quad (\text{C13})$

Since $U_{d_1}^\top U_{d_1}$ is a symmetric $d_1 \times d_1$ matrix, then the orthonormality condition in Eq. (C12) represents a total of $d_1 \times (d_1 + 1)/2$ conditions. Therefore, for the Lagrangian dual problem [as shown in Eq. (C13)], we need to introduce $d_1 \times (d_1 + 1)/2$ Lagrange multipliers λ_{ij} . Hence, we require that λ_{ij} is a symmetric matrix. Also, since each term in (C13) involves symmetric matrices, then the following first order optimality conditions,

$$\frac{\partial \mathcal{L}}{\partial \lambda_{pq}} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial U_{lm}} = 0, \quad (\text{C14})$$

can be solved for λ_{ij} only if the latter is symmetric. Using Eqs. (C13) and (C14), we get

$$\begin{aligned} \frac{\partial}{\partial \lambda_{pq}} \left[\sum_{i=1}^{d_1} \sum_{j=1}^d \sum_{k=1}^d U_{ij}^\top (X_1^\top X_1)_{jk} U_{ki} \right. \\ \left. - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left(\sum_{k=1}^d U_{jk}^\top U_{ki} - \delta_{ji} \right) \right] = 0 \end{aligned} \quad (\text{C15})$$

and

$$\begin{aligned} \frac{\partial}{\partial U_{lm}} \left[\sum_{i=1}^{d_1} \sum_{j=1}^d \sum_{k=1}^d U_{ij}^\top (X_1^\top X_1)_{jk} U_{ki} \right. \\ \left. - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left(\sum_{k=1}^d U_{jk}^\top U_{ki} - \delta_{ji} \right) \right] = 0. \end{aligned} \quad (\text{C16})$$

Equation (C15) implies the $d_1 \times (d_1 + 1)/2$ equations

$$\sum_{k=1}^d U_{qk}^\top U_{kp} = \delta_{qp}, \quad (\text{C17})$$

while Eq. (C16) implies the $d \times d_1$ equations

$$\begin{aligned} \sum_{j=1}^d U_{mj}^\top (X_1'^\top X_1')_{jl} + \sum_{k=1}^{d_1} (X_1'^\top X_1')_{lk} U_{km} \\ - \sum_{j=1}^{d_1} \lambda_{mj} U_{jl} - \sum_{i=1}^{d_1} \lambda_{im} U_{li} = 0. \end{aligned} \quad (\text{C18})$$

Using the fact that $X_1'^\top X_1'$ is symmetric, the first two terms of Eq. (C18) can be combined to a single term, and similarly (using the symmetry of λ_{ij}), the last two terms of Eq. (C18) can be combined to a single term to get

$$\sum_{j=1}^d U_{mj}^\top (X_1'^\top X_1')_{jl} - \sum_{i=1}^{d_1} \lambda_{mi} U_{il}^\top = 0. \quad (\text{C19})$$

Equations (C19) and (C17) are sufficient to solve for λ_{ij} and U_{kl} . Right-multiplying Eq. (C19) by U_{ln} and summing over $1 \leq l \leq d$, we get

$$\sum_{l=1}^d \sum_{j=1}^d U_{mj}^\top (X_1'^\top X_1')_{jl} U_{ln} - \sum_{i=1}^{d_1} \lambda_{mi} \sum_{l=1}^d U_{il}^\top U_{ln} = 0. \quad (\text{C20})$$

Using Eq. (C17), then Eq. (C20) becomes

$$\sum_{l=1}^d \sum_{j=1}^d U_{mj}^\top (X_1'^\top X_1')_{jl} U_{ln} = \lambda_{mn}. \quad (\text{C21})$$

Equations (C21) and (C19) represent a set of $d_1 \times (d_1 + 1)/2$ and $d_1 \times d$ equations, respectively. These can be solved to obtain the $d_1 \times (d_1 + 1)/2$ degrees of freedom of λ_{ij} and the $d_1 \times d$ degrees of freedom of U_{d_1} .

The lhs of Eq. (C21) represents the a_{mn} elements of a $d_1 \times d_1$ matrix, and similarly the rhs of (C21) represents the λ_{mn} elements of another $d_1 \times d_1$ matrix. Equation (C21) implies an entry-by-entry equation ($a_{mn} = \lambda_{mn}$) between the two matrices. Choosing $m = n$ and summing Eq. (C21) over $1 \leq m \leq d_1$ implies that the sum along the diagonal of the matrix on the lhs is equal to the sum along the diagonal of the matrix on the rhs or equivalently

$$\sum_{m=1}^{d_1} \sum_{l=1}^d \sum_{j=1}^d U_{mj}^\top (X_1'^\top X_1')_{jl} U_{lm} = \sum_{m=1}^{d_1} \lambda_{mm}. \quad (\text{C22})$$

Noting that the lhs of (C22) is the trace of the lhs of (C21), we can rewrite (C22) as

$$\text{tr}(U_{d_1}^\top X_1'^\top X_1' U_{d_1}) = \sum_{m=1}^{d_1} \lambda_{mm}. \quad (\text{C23})$$

To interpret the λ_{mm} , we use a theorem according to which the trace of a matrix is equal to the sum of its eigenvalues. Therefore, we can identify the λ_{mm} for $1 \leq m \leq d_1$ as the eigenvalues of the symmetric matrix $(X_1 U_{d_1})^\top (X_1 U_{d_1})$. However, these d_1 eigenvalues are d_1 out of the total d eigenvalues of $X_1^\top X_1$. This can be shown by using the invariance of a trace under similarity transformations (in this case, under conjugacy). Using Eq. (C6), we can rewrite Eq. (C23) for $d_1 = d$ as

$$\text{tr}(U_d^{-1} X_1'^T X_1' U_d) = \text{tr}(X_1'^T X_1') = \sum_{m=1}^d \lambda_{mm}. \quad (\text{C24})$$

Therefore, the maximum of the objective function $F = \text{tr}(U_{d_1}^T X_1'^T X_1' U_{d_1})$ in expression (C12) is equal to the summation of the d_1 largest eigenvalues of $X_1'^T X_1'$. Therefore, the orthonormal basis for the lower-dimensional subspace is given by the set of the eigenvectors corresponding to the d_1 largest eigenvalues of the symmetric matrix $X_1'^T X_1'$.

3. Formulation of CSC

Consider the binary classification problem with $X_1' \in \mathbb{R}^{n \times d}$ and $X_2' \in \mathbb{R}^{n \times d}$ as the data matrices corresponding to two data classes, $\mathcal{C}_1$ (noise points) and $\mathcal{C}_2$ (injection points), respectively. The number of data samples in $\mathcal{C}_1$ is the same as the number of data samples in $\mathcal{C}_2$ and is equal to $n/2$. The corresponding number of features is given by d for both classes $\mathcal{C}_1$ and $\mathcal{C}_2$.

We attempt to find two linear subspaces $\mathcal{S}_1 \subseteq \mathcal{C}_1$ and $\mathcal{S}_2 \subseteq \mathcal{C}_2$ that best approximate the data classes. Without loss of generality, we assume the dimensionality of these subspaces to be the same and equal to d_1 . Let

$$U = [u_1, u_2, \dots, u_{d_1}] \in \mathbb{R}^{d \times d_1} \quad (\text{C25})$$

and

$$V = [v_1, v_2, \dots, v_{d_1}] \in \mathbb{R}^{d \times d_1} \quad (\text{C26})$$

represent matrices of which the columns are orthonormal bases of the subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. If we attempted to find $\mathcal{S}_1$ independently from $\mathcal{S}_2$, then we would have to capture the maximal variance of the data projected onto $\mathcal{S}_1$ separately from the maximal variance of the data projected onto $\mathcal{S}_2$. That would be equivalent to solving the following two optimization problems [51]:

$$\max_{U \in \mathbb{R}^{d \times d_1}} \text{tr}(U^T X_1'^T X_1' U) \quad \text{subject to } U^T U = I_{d_1} \quad (\text{C27})$$

and

$$\max_{V \in \mathbb{R}^{d \times d_1}} \text{tr}(V^T X_2'^T X_2' V) \quad \text{subject to } V^T V = I_{d_1}. \quad (\text{C28})$$

The solution to the optimization problem as shown in expression (C27) is given by the eigenvectors (the columns of the orthonormal basis U of $\mathcal{S}_1$) corresponding to the d_1 largest eigenvalues of the matrix $X_1'^T X_1'$. Similarly, the solution to the optimization problem as shown in expression (C28) is given by the eigenvectors (the columns of the orthonormal basis V of $\mathcal{S}_2$) corresponding to the d_1 largest

eigenvalues of the matrix $X_2'^T X_2'$. Though the subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$ are good approximations to the two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, these projections may not be the ideal ones for classification purposes as each one of them is obtained without the knowledge of the other.

In the CSC, the two subspaces are found simultaneously by considering their relative orientation. This way, CSC allows for a trade-off between maximizing the variance of the projected data onto the two subspaces and the relative orientation between the two subspaces. The relative orientation between the two subspaces is generally defined in terms of the principal angles. The optimization problem in CSC is formulated as follows:

$$\begin{aligned} \max_{U, V \in \mathbb{R}^{d \times d_1}} & \text{tr}(U^T X_1'^T X_1' U) + \text{tr}(V^T X_2'^T X_2' V) \\ & + C \text{tr}(U^T V V^T U) \\ \text{subject to } & U^T U = I_{d_1}, \quad V^T V = I_{d_1}. \end{aligned} \quad (\text{C29})$$

The last term of the objective function $G = \text{tr}(U^T X_1'^T X_1' U) + \text{tr}(V^T X_2'^T X_2' V) + C \text{tr}(U^T V V^T U)$ is a measure of the relative orientation between the two subspaces as defined in Ref. [29]. The parameter C controls the trade-off between the relative orientation of the subspaces and the cumulative variance of the data as projected onto the two subspaces. For large positive values of C , the relative orientation between the subspaces reduces (the two subspaces become more “parallel”), while for large negative values of C , the relative orientation increases (the two subspaces become more “perpendicular” to each other).

This problem is solved using an alternating optimization algorithm described in Ref. [29]. For a fixed V , expression (C29) reduces to

$$\max_{U \in \mathbb{R}^{d \times d_1}} \text{tr}(U^T (X_1'^T X_1' + C V V^T) U) \quad \text{subject to } U^T U = I_{d_1}. \quad (\text{C30})$$

The solution to the optimization problem (C30) is obtained by choosing the eigenvectors corresponding to the d_1 largest eigenvalues of the symmetric matrix $X_1'^T X_1' + C V V^T$. Similarly, for a fixed U , expression (C29) reduces to

$$\max_{V \in \mathbb{R}^{d \times d_1}} \text{tr}(V^T (X_2'^T X_2' + C U U^T) V) \quad \text{subject to } V^T V = I_{d_1}, \quad (\text{C31})$$

where the solution to the optimization problem (C31) is again obtained by choosing the eigenvectors corresponding to the d_1 largest eigenvalues of the symmetric matrix $X_2'^T X_2' + C U U^T$.

The algorithm for CSC can be summarized as follows:

Algorithm 2. CSC (X'_1, X'_2, d_1, C)

-
1. Initialize U and V such that $U^T U = I_{d_1}$, $V^T V = I_{d_1}$.
 2. Find eigenvectors corresponding to the d_1 largest eigenvalues of the symmetric matrix $X'_1{}^T X'_1 + C V V^T$.
 3. Find eigenvectors corresponding to the d_1 largest eigenvalues of the symmetric matrix $X'_2{}^T X'_2 + C U U^T$.
 4. Alternate between 2 and 3 until one of the termination rules below is satisfied.
-

We define the following three termination rules:

- (i) Maximum limit Z on the number of iterations.
- (ii) Relative change in U and V at iteration m and $m + 1$,

$$\begin{aligned} \text{tol}_U^m &= \frac{\|U^{(m+1)} - U^{(m)}\|_F}{\sqrt{N}}, \\ \text{tol}_V^m &= \frac{\|V^{(m+1)} - V^{(m)}\|_F}{\sqrt{N}} \end{aligned} \quad (\text{C32})$$

where $N = d \times d_1$ and the subscript F denotes the Frobenius norm.

- (iii) Relative change in the value of the objective function G as shown in expression (C29) at iteration m and $m + 1$,

$$\text{tol}_f^m = \frac{G^{(m+1)} - G^{(m)}}{|G^{(m)}| + 1}. \quad (\text{C33})$$

The value of Z was set to 2000, while tol_f^m , tol_U^m , and tol_V^m are all set at the same value of 10^{-6} . From Eq. (C9), we see that the factor of $1/\sqrt{N}$ in (C32) results in the averaging of the squares of all the entries of the matrices $(U^{(m+1)} - U^{(m)})$ or $(V^{(m+1)} - V^{(m)})$. This regularization factor keeps the tolerance values independent of the data set.

After solving the optimization problem (C29) (by utilizing algorithm 2), a new point x is classified by computing the distances from the two subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$ defined by

$$\text{dist}(x, \mathcal{S}_1) = \text{tr}(U^T x^T x U) \quad (\text{C34})$$

and

$$\text{dist}(x, \mathcal{S}_2) = \text{tr}(V^T x^T x V). \quad (\text{C35})$$

The class of x is defined by

$$\text{class}(x) = \arg \left\{ \min_{i \in \{1,2\}} \{\text{dist}(x, \mathcal{S}_i)\} \right\}. \quad (\text{C36})$$

In our case, if x is closer to $\mathcal{S}_1$, then x is classified as noise (or no signal), and if x is closer to $\mathcal{S}_2$ then x is classified as an r -mode injection (or “presence of signal”).

APPENDIX D: DATA PREPARATION

1. Production of the data matrix for the MLA training

We start with the data maps in the ft maps produced by the stochastic transient analysis multidetector pipeline [23]. Let N be the number of noise maps. The number of (r -mode) injection maps is also equal to N . These ft maps are produced using simulated data recolored with the aLIGO sensitivity noise curve. Each map has a size of $F \times T$ pixels with each pixel along the vertical axis corresponding to δf Hz and each pixel along the horizontal axis corresponding to δt s; hence, the length of the map along the vertical axis is $(F\delta f)$ Hz, and the length of the map along the horizontal axis is $(T\delta t)$ s. This ft map is reshaped to a $1 \times D$ (where $D = FT$) row vector. We reshaped all $2N$ ft maps (each one of size $F \times T$) and produced $2N$ row vectors x_i with $i \in \{1, 2, \dots, 2N\}$. The rows with $i \in \{1, 2, \dots, N\}$ correspond to the noise ft maps, while the rows with $i \in \{N + 1, N + 2, \dots, 2N\}$ correspond to the injection ft maps.

We then used the rows x_i with $i \in \{1, 2, \dots, N\}$ to produce a $N \times D$ noise data matrix, X_1 , and we also used the rows with x_i with $i \in \{N + 1, N + 2, \dots, 2N\}$ to produce a $N \times D$ injection data matrix, X_2 . The MLAs would take as input the $2N \times D$ data matrix given by

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}. \quad (\text{D1})$$

Each row x_i with $i \in \{1, 2, \dots, 2N\}$ of the data matrix X corresponds to a single ft map. The total number of rows is equal to the number of data points, $n = 2N$, while the total number of columns (i.e., the total number of features) is equal to $D = FT$, where D is the dimensionality of the feature space in which each single ft map lives.

For any matrix, we know that the row rank is equal to column rank; therefore, the number of linearly independent columns of X is equal to $2N$. This number is determined by the limited number ($n = 2N$) of ft maps we could produce. This means that, even though each single ft map lives in a D -dimensional space ($D \gg n$), we can only approximate these ft maps as vectors living in an n -dimensional space (subspace of the D -dimensional space). The best approximation of this subspace would be the one in which the most “dominant” n features (out of the total number of D) constitute a basis of the subspace. A well-known method of choosing the n most dominant features is described by the PCA [52], or see Appendix C 2. However, the (RAM) memory required to perform PCA on X is beyond 1 Tbyte, thus making it practically impossible to perform PCA on X with realistically available computing resources.

A reliable approach to solving the problem of the high dimensionality of the features ($D \gg n$) is to seek MLAs that will naturally select d features (with $d \ll D$) such that $d \leq n$ [53]. Three classes of MLAs that can achieve this are

the ANN, SVM, and CSC methods. However, the data matrix is too large to attempt to perform any MLAs on it. Therefore, the only way out of these restrictions the data matrix size imposes is to perform resolution reduction for each $F \times T$ ft map (before reshaping each one of them to a row vector). After the resolution reduction, performing further feature selection would still benefit the training of the algorithms in terms of speed. The right choice of features can significantly decrease the training time without noticeably affecting the training efficiencies.

A resolution reduction on the ft maps would result in a number of $2N$ row vectors (of dimensionality $1 \times d$) such that $d \ll D$. The desired effect of the resolution reduction would be to get $d \leq n$. The first guess for such a reduction would be to choose a factor of D/n . That would be equivalent to a reduction by a factor of $\sim \sqrt{\frac{D}{n}}$ along each axis (frequency and time) of the ft map. However, it turned out that this is not the optimal resolution (per axis) reduction factor. The following two subsections describe the experimentation on the reduction factor.

2. Resolution reduction: Bicubic interpolation

To perform the resolution reduction, we used the `IMRESIZE` MATLAB function. The original ft map of $F \times T$ pixels consists of a $(F + 1) \times (T + 1)$ point grid. `IMRESIZE` will first decrease the number of points in the point grid according to the chosen resolution reduction factor, r . Interpolation is then used to calculate the surface within each pixel in the new point grid. The result is a new ft map of dimensionality $\frac{F}{r} \times \frac{T}{r}$ with a number of pixels equal to $d = \frac{FT}{r^2} = \frac{D}{r^2}$.

We used the bicubic interpolation option of the `IMRESIZE` function. According to this, the surface within each pixel can be expressed by

$$S(t, f) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} t^i f^j. \quad (\text{D2})$$

The bicubic interpolation problem is to calculate the 16 a_{ij} coefficients. The 16 equations used for these calculations consist of the following conditions at the four corners of each pixel:

- (a) the values of $S(t, f)$
- (b) the derivatives of $S(t, f)$ with respect to t
- (c) the derivatives of $S(t, f)$ with respect to f
- (d) the cross-derivatives of $S(t, f)$ with respect to t and f

Determining the resolution reduction factor that would yield the best training efficiencies for the MLAs was not a

very straightforward task. To do so, we performed a series of tests using the set of N noise ft maps and the set of N injection ft maps. The injected signal SNR values lay in a range such that $10^{-23.7} \leq h \leq 10^{-23.2}$.

3. Resolution reduction vs training efficiency

We tested five different resolution reduction factors ($r = 10^{-1}$, $r = 10^{-1.5}$, $r = 10^{-2}$, $r = 10^{-2.5}$, and $r = 10^{-3}$) where the value of r corresponds to the factor by which each axis resolution is reduced. With $N = 11$, 350 and $F = 1001$, $T = 4999$ (such that $D = 5,003,999$), the resulting $(2N \times d)$ data matrices had dimensions 22700×50500 , 22700×5155 , 22700×550 , 22700×64 , and 22700×10 , respectively. Subsequently, each of the three MLAs was trained, and the training efficiencies were plotted against the resolution reduction factors. The results are shown in Fig. 10. From the plots, we see that the training efficiencies first improve as we lower the resolution. For too low or too high resolution reductions, the training efficiencies decrease. This behavior was consistent on all three MLAs. At a reduction factor of 100 per axis, we have the maximum training efficiency. Resolution reduction offers two advantages: (a) it increases the MLA training efficiency, and (b) it reduces the training time. Using the results from Fig. 10, we determined that the best resolution reduction would be the factor of $r = 10^{-2}$. This results in a data matrix with dimensions of 22700×550 (disc space of 84 Mbytes). Examples of full-resolution maps are shown in Fig. 6 (for noise) and Fig. 8 (for noise plus signal). The corresponding reduced-resolution maps are shown in Fig. 7 and Fig. 9.

After dimensionality reduction, the matrices X_1 and X_2 become X'_1 with row vectors $x_i \in \mathbb{R}^{550}$ where $i \in \{1, 2, \dots, 11, 350\}$ and X'_2 with row vectors $x_i \in \mathbb{R}^{550}$ where $i \in \{11, 351, \dots, 22700\}$. Both of the X'_1 and X'_2 have a reduced dimensionality $11, 350 \times 550$. Similarly, we define the dimensionally reduced $22,700 \times 550$ data matrix

$$X' = \begin{pmatrix} X'_1 \\ X'_2 \end{pmatrix}. \quad (\text{D3})$$

The number of rows, $n = 22,700$, is the number of data points (ft maps), and the number of columns, $d = 550$, is the number of features of each point or the dimensionality of the space in which each ft map lives (after the resolution reduction).

- [1] S. Chandrasekhar, *Phys. Rev. Lett.* **24**, 611 (1970).
- [2] J. L. Friedman and B. F. Schutz, *Astrophys. J.* **221**, 937 (1978).
- [3] J. L. Friedman and B. F. Schutz, *Astrophys. J.* **222**, 281 (1978).
- [4] N. Andersson, *Astrophys. J.* **502**, 708 (1998).
- [5] J. L. Friedman and S. M. Morsink, *Astrophys. J.* **502**, 714 (1998).
- [6] R. Bondarescu, S. A. Teukolsky, and I. Wasserman, *Phys. Rev. D* **79**, 104003 (2009).
- [7] M. G. Alford, S. Mahmoodifar, and K. Schwenzer, *Phys. Rev. D* **85**, 044051 (2012).
- [8] A. Mytidis, M. Coughlin, and B. Whiting, *Astrophys. J.* **810**, 27 (2015).
- [9] W. C. Ho and D. Lai, *Astrophys. J.* **543**, 386 (2000).
- [10] J. E. Staff, P. Jaikumar, V. Chan, and R. Ouyed, *Astrophys. J.* **751**, 24 (2012).
- [11] A. K. Schenk, P. Arras, E. E. Flanagan, S. A. Teukolsky, and I. Wasserman, *Phys. Rev. D* **65**, 024001 (2001).
- [12] L. Rezzolla, F. K. Lamb, and S. L. Shapiro, *Astrophys. J. Lett.* **531**, L139 (2000).
- [13] N. Andersson and K. D. Kokkotas, *Int. J. Mod. Phys. D* **10**, 381 (2001).
- [14] L. Lindblom, B. J. Owen, and S. M. Morsink, *Phys. Rev. Lett.* **80**, 4843 (1998).
- [15] B. J. Owen, L. Lindblom, C. Cutler, B. F. Schutz, A. Vecchio, and N. Andersson, *Phys. Rev. D* **58**, 084020 (1998).
- [16] C. D. Ott, A. Burrows, T. A. Thompson, E. Livne, and R. Walder, *Astrophys. J. Suppl. Ser.* **164**, 130 (2006).
- [17] D. Branch and G. A. Tammann, *Annu. Rev. Astron. Astrophys.* **30**, 359 (1992).
- [18] R. P. Kirshner and J. Kwan, *Astrophys. J.* **193**, 27 (1974).
- [19] B. P. Schmidt, R. P. Kirshner, R. G. Eastman, M. M. Phillips, N. B. Suntzeff, M. Hamuy, J. Maza, and R. Aviles, *Astrophys. J.* **432**, 42 (1994).
- [20] A. Goobar, J. Johansson, R. Amanullah, Y. Cao, D. A. Perley, M. M. Kasliwal, R. Ferretti, P. E. Nugent, C. Harris, A. Gal-Yam, E. O. Ofek *et al.*, *Astrophys. J. Lett.* **784**, L12 (2014).
- [21] Y. S. Li and S. D. White, *Mon. Not. R. Astron. Soc.* **384**, 1459 (2008).
- [22] F. Mannucci, D. Maoz, K. Sharon, M. T. Botticella, M. D. Valle, A. Gal-Yam, and N. Panagia, *Mon. Not. R. Astron. Soc.* **383**, 1121 (2008).
- [23] E. Thrane, S. Kandhasamy, C. D. Ott *et al.*, *Phys. Rev. D* **83**, 083004 (2011).
- [24] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari *et al.*, *Classical Quantum Gravity* **35**, 065009 (2018).
- [25] B. P. Abbott, R. Abbott, R. Adhikari, P. Ajith, B. Allen, G. Allen, R. S. Amin, S. B. Anderson, W. G. Anderson, M. A. Arain *et al.*, *Phys. Rev. D* **93**, 042005 (2016).
- [26] E. Thrane and M. Coughlin, *Phys. Rev. D* **88**, 083010 (2013).
- [27] S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice Hall, Englewood Cliffs, 1999), 2nd ed., pp. 23–71.
- [28] V. N. Vapnik and V. Vapnik, *Statistical Learning Theory* (Wiley, New York, 1998).
- [29] O. P. Panagopoulos, V. Pappu, P. Xanthopoulos, and P. M. Pardalos, *Omega* **59**, 40 (2006). doi: 10.1016/j.omega.2015.05.009
- [30] B. P. Abbott, R. Abbott, R. Adhikari, P. Ajith, B. Allen, G. Allen, R. S. Amin, S. B. Anderson, W. G. Anderson, M. A. Arain *et al.*, *Rep. Prog. Phys.* **72**, 076901 (2009).
- [31] B. P. Abbott, R. Abbott, R. Adhikari, P. Ajith, B. Allen, G. Allen, R. S. Amin, S. B. Anderson, W. G. Anderson, M. A. Arain *et al.*, *Classical Quantum Gravity* **32**, 074001 (2015).
- [32] B. P. Abbott, R. Abbott, R. Adhikari, P. Ajith, B. Allen, G. Allen, R. S. Amin, S. B. Anderson, W. G. Anderson, M. A. Arain *et al.*, *Phys. Rev. D* **72**, 102004 (2005).
- [33] P. M. Dragicevich, D. G. Blair, and R. R. Burman, *Mon. Not. R. Astron. Soc.* **302**, 693 (1999).
- [34] S. Ando, J. F. Beacom, and H. Yuksel, *Phys. Rev. Lett.* **95**, 171101 (2005).
- [35] P. T. Baker, S. Caudill, K. A. Hodge, D. Talukder, C. Capano, and N. J. Cornish, *Phys. Rev. D* **91**, 062004 (2015).
- [36] R. Biswas, L. Blackburn, J. Cao, R. Essick, K. A. Hodge, E. Katsavounidis, K. Kim, Y. M. Kim, E. O. Le Bigot, C. H. Lee, J. J. Oh, S. H. Oh, E. J. Son, Y. Tao, R. Vaulin, and X. Wang, *Phys. Rev. D* **88**, 062003 (2013).
- [37] K. J. Basu, D. Bhattacharyya, and K. Tai-hoon, *Int. J. Software Eng. Appl.* **4**, 2 (2010).
- [38] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, New York, 2000).
- [39] C. M. Bishop, *Neural Networks for Pattern Recognition* (Aston University, Clarendon Press, Birmingham, Oxford, 1995).
- [40] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer-Verlag, Berlin, Heidelberg, 2008).
- [41] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (MIT, Cambridge, MA, 2012).
- [42] L. Breiman, *Mach. Learn.* **45**, 5 (2001).
- [43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Commun. ACM* **60**, 84 (2017).
- [44] D. R. Wilson and T. R. Martinez, *Neural Netw.* **16**, 1429 (2003).
- [45] D. Goldfarb and A. Idnani, *Mathematical Program.* **27**, 1 (1983).
- [46] R. Fletcher, *Practical Methods of Optimization* (John Wiley & Sons Ltd, New York, 1987).
- [47] C. Chang and C. Lin, *ACM Trans. Intell. Syst. Technol.* **2**, 27 (2011).
- [48] J. Platt *et al.*, Technical Report No. msr-tr-98-14, Microsoft Research, 1998.
- [49] J. Platt *et al.*, *Advances in Kernel Methods* (MIT Press, Cambridge, MA, 1999), Vol. 4, pp. 185–208.
- [50] R. Vidal, Y. Ma, and S. Sastry, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE, New York, 2005), Vol. 27, pp. 1945–1959.
- [51] J. Laaksonen, *Artificial Neural Networks ICANN'97*, (Springer, Berlin, 1997), pp. 637–642.
- [52] I. Jolliffe, *Principal Component Analysis*, Springer Series in Statistics (Springer-Verlag, New York, 2002), ISBN 9781475719048.
- [53] I. M. Johnstone and D. M. Titterton, *Proc. R. Soc. A* **367**, 4237 (2009).
- [54] M. Abadi *et al.*, in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA, USA, 2016 (USENIX Association, Berkeley, 2016), Vol. 16, pp. 265–283.