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

Antonis Mytidis,1 Athanasios A. Panagopoulos,2 Orestis P. Panagopoulos,3 and Bernard Whiting1

1Department of Physics, University of Florida, 2001 Museum Road, Gainesville, FL 32611-8440, USA
2Department of Electronics and Computer Science, University of Southampton, Southampton, UK
3Department of Industrial Engineering and Management Systems, University of Central Florida, 12800 Pegasus Dr., Orlando, FL 32816, USA

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 (ANNs), support vector machines (SVMs) and constrained subspace classifiers (CSCs). The objective of this study is to compare the detection efficiency that MLAs can achieve with the efficiency of conventional detection algorithms discussed in an earlier paper. Comparisons are made using 2 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 machine learning algorithms are suitable for the detection of long-lived gravitational-wave transients and that when assuming knowledge of the distance to the source, MLAs are at least as efficient as conventional methods.

I. INTRODUCTION

It was demonstrated (1998) that neutron star r-mode oscillations of any harmonic, frequency and amplitude belong to the group of non-axisymmetric modes that can be driven unstable due to gravitational radiation [8,7,10]. This theory suggested 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 in the form of r-mode gravitational radiation.

In a previous paper [9] we presented a sensitivity study of a seedless clustering detection algorithm based on r-mode waveforms predicted by the Owen et. al. 1998 model

\[ f(t) = \frac{1}{(f_0 - \mu t)^\frac{1}{2}} \]  

(1)

\[ \mu = 1.1 \times 10^{-20} |\alpha|^2 \text{ s}^{-1} \text{ Hz}^6 \]  

(2)

and the gravitational radiation power given by

\[ \dot{E} \approx 3.5 \times 10^{19} f_0^8 |\alpha|^2 \text{ W.} \]  

(3)

This model depends on two parameters: the (dimensionless) saturation amplitude, \(|\alpha|\), of the r-mode oscillations and the initial gravitational wave spindown frequency \(f_0\). The theoretical predictions for the values of these parameters were discussed extensively in our previous paper. The values we considered for \(\alpha\) lie in the range of \(10^{-3} - 10^{-1}\) while the values we considered for \(f_0\) 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.

In our previous paper, a seedless clustering algorithm was used. This algorithm is based on the statistical significance of signal to noise ratios (snr) of clusters of above the threshold snr pixels. 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 5 times weaker than the noise.

In the sensitivity study performed for the clustering algorithm, we used 9 distinct waveforms. These were chosen by taking \((\alpha, f_0)\) pairs using 3 values \((10^{-1}, 10^{-2}, 10^{-3})\) for \(\alpha\) and 3 values \((700\text{ Hz}, 1100\text{ Hz}, 1500\text{ Hz})\) for \(f_0\). In this sensitivity study for the MLAs, for comparison purposes, we used 2 of these waveforms: \((f_0 = 1500\text{ Hz}, \alpha = 0.1)\) and \((f_0 = 1100\text{ Hz}, \alpha = 0.01)\). These waveforms as well as their corresponding power decays are shown in Fig.1 and Fig.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: ANN [5], SVM [4] and CSC [8]. All three methods are considered novel applications in the area of long transient gravitational wave searches.

This paper is designed as follows: The next section describes the preparation of the data used for the training of the MLAs. The discussion extends to the resolution reduction of the data maps and also to the training efficiencies as a function of resolution reduction for all three MLAs. In section 3 we present a brief introduction into ANN methods, in section 4 we do the same for SVM methods and in section 5 we present a similar introduction for CSCs. In section 6 we present and discuss our results and compare the MLA efficiencies to the conventional (clustering) algorithms efficiencies. Finally, in section 7, we summarize our conclusions including topics for future work.
II. DATA PREPARATION

A. Choice of the \( f_o \) and \( \alpha \) parameter values

From equations (1) and (2) we have the two model parameters \( \alpha \) 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 signal-to-noise ratio (SNR). For a single pixel in the frequency-time maps (ft-maps) the SNR satisfies [11]

\[
\text{SNR}(t, f, \hat{\Omega}) \propto \text{Re} \left[ \hat{Q}_{ij}(t, f, \hat{\Omega}) \hat{C}_{ij}(t, f) \right] \tag{4}
\]

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

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

For the construction of the injection maps we chose the 3 parameter values \( f_o, \alpha \) and \( h^2 \) to be uniformly distributed within the corresponding predetermined ranges. The values of the distance \( d \) are constrained accordingly such that the above conditions are satisfied.

Each injection set that was produced and used for the MLA training was limited to 11350 injection maps and 11350 noise maps. This was due to the finite amount of data that was produced during the S5 LIGO run, the computational resources available as well as the time needed to produce the 22700 maps. For each injection the waveform was randomly chosen in such a way that the \( \alpha \) value was randomly chosen from a uniform distribution of 11350 \( \alpha \) values in the range of \( 10^{-3} - 10^{-1} \), the \( f_o \) value was randomly chosen from a uniform distribution of 11350 \( f_o \) values in the range of 600 – 1600 Hz, and for the \( h^2 \) values we picked 3 ranges (for 3 separate MLA trainings), whose choice is discussed in the next paragraph.

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 equation (5) we see that the conventional algorithms 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 8 waveforms. For example from table 1 in [9] we see that for \( f = 700 \) Hz and \( \alpha = 0.01 \) we get a detection distance of 0.043 Mpc. Substituting in equation (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 gravitational wave strain the MLAs will have to detect if they are shown to be at least as efficient as the conventional clustering algorithms [12].

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:

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

(ii) produce three sets of data such that the \( h^2 \) values are uniformly distributed in the following ranges:
- (a) from \( 10^{-46.4} \) to \( 10^{-45.4} \) (i.e. \( 10^{-23.2} \leq h \leq 10^{-22.7} \)).
- (b) from \( 10^{-47.4} \) to \( 10^{-46.4} \) (i.e. \( 10^{-23.7} \leq h \leq 10^{-23.2} \)).
- (c) from \( 10^{-48.0} \) to \( 10^{-47.4} \) (i.e. \( 10^{-24.0} \leq h \leq 10^{-23.7} \)).

The last choice of \( 10^{-24} \) is such that the waveform with \((f_0 = 1500 \text{ Hz}, \alpha = 0.1)\) 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 Milky Way) the supernova event rate is 1 every 1-2 years.

C. Production of the data matrix for the MLA training

We start with the data in the frequency-time domain (ft-maps) produced by the stochastic transient analysis multi-detector pipeline (STAMP) [11]: 11350 noise maps and 11350 (r-mode) injection maps. These ft-maps are produced using time-shifted S5 data recolored with the aLIGO sensitivity noise curve. Each map has a size of 1001 x 4999 pixels with each pixel along the vertical axis corresponding to 1 Hz and each pixel along the horizontal axis corresponding to 0.5 s, hence the length of the map is 2499.5 s. This ft-map is reshaped to a \( 1 \times 5003999 \) row vector that occupies a disc space of 37.4 MB. We reshaped all 22700 ft-maps (each one of size 1001 x 4999) and produced 22700 row vectors \( x_i \) with \( i \in \{1, 2, \ldots, 22700\} \). The rows with \( i \in \{1, 2, \ldots, 11350\} \) correspond to the noise ft-maps while the rows with \( i \in \{11351, 11352, \ldots, 22700\} \) correspond to the injection ft-maps.

We then used the rows \( x_i \) with \( i \in \{1, 2, \ldots, 11350\} \) to produce a \( 11350 \times 5003999 \) noise data matrix, \( X_1 \), given by

\[
X_1 = \begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_{11350}
\end{pmatrix}
\]

(6)

and we also used the rows with \( x_i \) with \( i \in \{11351, 11352, \ldots, 22700\} \) to produce a \( 11350 \times 5003999 \) injection data matrix, \( X_2 \), given by

\[
X_2 = \begin{pmatrix}
  x_{11351} \\
  x_{11352} \\
  \vdots \\
  x_{22750}
\end{pmatrix}
\]

(7)

The MLAs would take as an input the \( 22700 \times 5003999 \) data matrix given by

\[
X = \begin{pmatrix}
  X_1 \\
  X_2
\end{pmatrix}
\]

(8)

Each row \( x_i \) with \( i \in \{1, 2, \ldots, 22700\} \) of the data matrix \( X \) corresponds to a single ft-map. The total number of rows is equal to the number of data points, \( n = 22700 \), while the total number of columns (i.e. the total number of features) is equal to \( D = 5003999 \), where \( D \) is the dimensionality of the feature space in which each single ft-maps lives.

For any matrix we know that row rank = column rank, therefore, the number of linearly independent columns of \( X \) is equal to 22700. This number is determined by the limited number \( (n = 22700) \) of ft-maps we could produce. This means that even though each single ft-map lives in a \( 5003999 \)-dimensional space, we can only approximate these ft-maps as vectors living in a \( 22700 \)-dimensional space (subspace of the \( 5003999 \)-dimensional space). The best approximation of this subspace would be the one in which the most ‘dominant’ \( 22700 \) features (out of the total number of \( 5003999 \)) constitute a basis of the subspace. A well known method of choosing the \( 22700 \) most dominant features is described by the principal component analysis (PCA) [13] or see section [VB]. However, the data matrix \( X \) is of size 848.8 GB and this makes the (RAM) memory required to perform PCA on \( X \) beyond 1TB, making it practically impossible to perform PCA on \( X \) with realistically available computing resources.

A reliable approach to solve the problem of the high dimensionality of the features \((D \gg n)\) is to seek MLAs that will naturally select \( d \)-many features (with \( d \ll D \)) such that \( d \leq n \) [14]. 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 \( 1001 \times 4999 \) 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 \( 22700 \times 1 \times d \) row vectors 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 \( D/n \sim 220 \). That would be equivalent to a reduction by a factor of \( \sim 15 \) 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 sub-sections describe the experimentation on the reduction factor.

D. Resolution reduction: bicubic interpolation

To perform the resolution reduction, we used the imresize matlab function. The original ft-map of 1001×4999 pixels consists of a 1002×5000 point grid. Imresize will first decrease the number of points in the point grid according to the chosen resolution reduction factor, r. When r = 10⁻², the imresize function gives a new ft-map of dimensionality 11×50, the new point grid will be 12×51. Interpolation is then used to calculate the surface within each pixel in the new point grid.

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

\[ S(t, f) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} t^i f^j \]  

(9)

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 4 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 11350 noise ft-maps and the set of 11350 injection ft-maps. The injected signal SNR values lay in a range such that \( 10^{-23.7} \leq h \leq 10^{-23.2} \).

E. Resolution reduction versus training efficiency

We tested 5 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. The resulting data matrices had dimensions 22700 × 50500, 22700 × 5155, 22700 × 550, 22700 × 64 and 22700 × 10 respectively. Subsequently each of the three MLAs were trained and the training efficiencies were plotted against the resolution reduction factors. The results are shown in Fig.3. 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.3 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 84MB).

After dimensionality reduction, the matrices \( X_1 \) and \( X_2 \) as given by equations (6) and (7) respectively become \( X'_1 \) and \( X'_2 \) each one of a reduced dimensionality 11350 × 550. We define these matrices as follows

\[
X'_1 = \begin{pmatrix}
    x_1 \\
    x_2 \\
    \ddots \\
    x_{11350}
\end{pmatrix} \quad (10)
\]

with row vectors \( x_i \in \mathbb{R}^{550} \) where \( i \in \{1, 2, ..., 11350\} \) and

\[
X'_2 = \begin{pmatrix}
    x_{11351} \\
    x_{11352} \\
    \ddots \\
    x_{22750}
\end{pmatrix} \quad (11)
\]

with row vectors \( x_i \in \mathbb{R}^{550} \) where \( i \in \{11351, ..., 2270\} \). Similarly we define the dimensionally reduced 22700 × 550 data matrix

\[
X' = \begin{pmatrix}
    X'_1 \\
    X'_2
\end{pmatrix} \quad (12)
\]

The number of rows, \( n = 22700 \), 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).
III. ARTIFICIAL NEURAL NETWORKS

The 22700 × 550 data matrix, \(X\)', will be presented as input into a feed-forward neural network with an input layer of dimensionality equal to the number of columns of the data matrix (i.e. \(d = 550\)). For the training of the ANN we randomly picked 90% of the first 11350 (injection data) rows and also 90% of the second 11350 (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 50 nodes (‘neurons’) and an output layer with two ‘neurons’ that would ‘fire’ for ‘signal’ or ‘no signal’, respectively. The ‘hidden’ layer used ‘neurons’ with the logistic 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:

\[
\text{soft-max}(b_k) = \frac{\exp(b_k)}{\sum_k \exp(b_k)}. \tag{16}
\]

When the values from equation (15) are presented in the output layer we get the result

\[
x''_{1l} = \text{soft-max}\left(\sum_{k=1}^{50} w_{lk}^{(2)} x'_{1k} + w_{l0}^{(2)}\right) \tag{17}
\]

(where \(l = 1, 2\) as the output value in the single neuron of the output layer. Equation (17) represents the ‘forward propagation’ of information in the neural network since the inputs are ‘propagated forward’ to produce the outputs of the ANN, according to the particular ‘weights’ and ‘biases’.

Equation (17) also shows that a neural network is a non-linear function, \(F\), from a set of input variables \(\{x_i\}\) such that \(i \in \{1, 2, \ldots, 22700\}\) as defined by equation (14) i.e. \(x_i\) are row vectors of the matrix \(X\)’ as a set of output variables \(\{x''_l\}\) such that \(l \in \{1, 2\}\) i.e. the output layer has 2 neurons (one fires for noise and the other fires for injection). To merge the weights \(w_{kj}^{(1)}\) and biases \(w_{k0}^{(1)}\) into a single matrix (and similarly do with the weights \(w_{kj}^{(2)}\) and biases \(w_{k0}^{(2)}\)) we need to redefine \(x_1\) as given by equation (14) to

\[
x_1 = \{ x_{ij} | i = 1 \text{ and } j = 0, 1, 2, \ldots, 550 \text{ and } x_{10} = 0 \}\tag{18}
\]

and similarly redefine all row vectors of \(X\)’ as well as all the output row vectors from the hidden layer. Then the non-linear function \(F\) is controlled by a \((50 + 1) \times (550 + 1)\) matrix \(w^{(1)}\) and a \(2 \times (50 + 1)\) matrix \(w^{(2)}\) of adjustable parameters. Training a neural network corresponds to calculating these parameters.

Numerous algorithms for training ANN exist 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 22700 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) they lead to a faster convergence. As such, we used a batch version of gradient descent as the optimization algorithm. This form of 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:
The distance of a point $x_i$ to a flat hypersurface $H = \{ x | \langle w, x \rangle + b = 0 \}$ is given by
\[ d_{x_i}(w, b) = z_i \times (\langle w, x_i \rangle + b) \] (19)
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 $\{1, 2, 3, ..., 22700\}$. 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, $S$, of vectors is defined as the minimum of the set of all distances $D = \{ d_{x_i}(w, b) | x_i \in S \}$ from the hypersurface $H$. For the purpose of our discussion the set $S$ 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 $\{(x_1, y_1), ..., (x_n, y_n) | x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}\}$ is called ‘separable’ by a hypersurface $H = \{ x | \langle w, x \rangle + b = 0 \}$ if both a unit vector $w (\|w\| = 1)$ and a constant $b$ exist so that the following inequalities hold:
\[ \langle w, x_i \rangle + b \geq \gamma_S \quad \text{if} \quad y_i = +1 \] (20)
\[ \langle w, x_i \rangle + b \leq -\gamma_S \quad \text{if} \quad y_i = -1 \] (21)
where $S = \{x_i | i = 1, 2, ..., n\}$ and $\gamma_S$ is given by definition 2.

For the purpose of our discussion $d = 550$ is the dimensionality of the points $x_i$ (this dimensionality corresponds to the number of pixels in each ft-map) and $n = 22700$ 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. $H = \{ x | \langle cw, x \rangle + cb = 0 \}$, we can take $c$ such that $c\gamma_S = 1$ and hence we can rewrite equations (20) and (21) as
\[ y_i \times (\langle cw, x_i \rangle + cb) \geq 1 \quad \text{for all} \ i=1,2,\ldots,n. \] (22)
Defining $w' = cw$ i.e. $\|w'\| = c$ and dividing equation (22) by $c$ we get
\[ y_i \times (\langle w', x_i \rangle + b) \geq \frac{1}{\|w'\|} \quad \text{for all} \ i=1,2,\ldots,n. \] (23)

**Formulation of the SVM optimization problem:** Given a training set, that is, a data matrix $X' = \left( X_1', X_2' \right)$, $X_1'$ being the noise points and $X_2'$ being 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’
Lagrange multipliers

The slack variables \( \xi \)

(25).

Before formulating the Lagrangian dual we introduce the 'slack variables', \( \xi \) \((i = 1, 2, ..., n)\), that are used to relax the conditions in equation (22) and account for outliers or 'errors'. Instead of solving equation (24) we seek a solution to

\[
\min_{w,b} \frac{1}{2} \|w'\|^2 \quad \text{subject to} \quad 1 - y_i \times (\langle w', x_i \rangle + b') \leq 0 \quad \text{for all } i=1,2,...,n
\]

(25).

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 equations (24) and (25).

Defining \( G_{ij} = y_i y_j x_i^\top x_j \) problem (31a)-(31b) is equivalently expressed as

\[
\min_{\alpha} \frac{1}{2} \alpha^\top G \alpha - e^\top \alpha \quad \text{subject to} \quad y^\top \alpha = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, ..., n
\]

(32a)

and

\[
\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad C \geq 1
\]

(32b)

and

\[
\sum_{i=1}^{n} \alpha_i y_i = 0
\]

(32c)

where \( e^\top \) is a \( n \)-dimensional row vector equal to \( e^\top = (1, 1, ..., 1) \) and (32c) is derived from (29c) together with (27).

Since the objective function in equation (32) 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 [18].

After solving the optimization problem defined by expressions (32a)-(32c), i.e. after evaluating all the \( \alpha_i \) \((i = 1, 2, ..., n)\), we can find the vector \( w \) using (29a). The constant \( b \) can be found by using the Karush-Kuhn-Tucker (KKT) complementarity conditions [19].

\[
\alpha_i \{ -1 + y_i \times (\langle w', x_i \rangle + b') + \xi_i \} = 0
\]

(33a)

\[
\beta_i \xi_i = 0
\]

(33b)

along with equation (29c). For any \( \alpha_i \) satisfying \( 0 < \alpha_i < C \), equation (29c) implies that \( \beta_i > 0 \) and hence (33b) implies that \( \xi_i = 0 \). Consequently, we can use the \( x_i \) corresponding to the aforementioned \( \alpha_i \) to solve equation (33a) for \( b' \).

Having calculated the vector \( w' \) and the constant \( b' \) is equivalent to knowing the hypersurface defined by \( \langle w', x_i \rangle + b' = 0 \). During the testing phase a new data point, \( x_i \), is classified according to

\[
\text{class}(x_i) = \text{sgn}(\langle w', x_i \rangle + b').
\]

(34)

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 equation (32) with sequential minimal optimization (SMO) [20]. SMO modifies only a subset of dual variables \( \alpha_i \) 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 \( \alpha_i \). 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 two is solved analytically. This process is repeated until convergence. The integrated software for support vector classification (LIBSVM) [21] 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 [22]. Hence we choose to use it for training the SVM, using the LIBSVM routine 'svmtrain'.
Non-linear SVM: The soft margins $\xi_i$ can only help when data are 'reasonably' linearly separable. However, in most real world problems, data is 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

$$\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$$

such that $\Phi(x_i) \in \mathcal{H}$. (35)

From equations (30) and (35) we see that the non-linear 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 $\Phi$ 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: (i) symmetric, that is $k(x_i, x_j) = k(x_j, x_i)$ for any $x_i, x_j \in \mathcal{L}$ and (ii) positive semi-definite, that is

$$c^\top K c = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \quad (36)$$

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

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 equation (32) 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 $\alpha_i$ 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 Radial Basis Function (RBF) kernel defined by

$$k(x_i, x_j) = \exp \left( -\gamma \frac{\|x_i - x_j\|^2}{\sigma^2} \right) \quad (37)$$

where $\gamma$ is a free parameter and $\sigma$ is the standard deviation of the $x_i$. Typically free parameters are calculated by using the cross validation method on the data set, meaning that we split the data set into several subsets and the optimization problem is solved on each subset by using a kernel with a different parameter value $\gamma$. We then choose the parameter value that gives the lowest minimum value of the objective function. It has been seen in many previous applications that the value of $\gamma$ giving optimal results was equal to $\gamma = 1/d = 1/550$. 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 equations (26)-(34). Then equation (30) is re-expressed as

$$\mathcal{L}(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \Phi(x_j), \Phi(x_i) \rangle. \quad (38)$$

After solving (32), the $\alpha_i$ $(i = 1, 2, ..., n)$ are substituted in (39) that we solve for $w'_j$ to get

$$w'_j = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \quad \forall j = 1, 2, ..., d \quad (39)$$

where $\Phi_j(x_i)$ is the $j^{th}$ entry of the $\Phi(x_i)$ transformed data point. Since the transformation $\Phi$ is not obtained directly we never calculate the $w'$ vector explicitly. Nevertheless, we can substitute expression (39) in (33) and solve the latter for $b'$ (when $\xi_k = 0$ and $\alpha_k \neq 0$) as follows

$$b' = 1 - y_k \times \sum_{i=1}^{n} \alpha_i y_i \langle \Phi(x_i), \Phi(x_k) \rangle \quad (40)$$

where this result should be independent of which $k$ we use. Having the expression (39) for the vector $w'$ and the expression (40) for the constant $b'$ we can classify a new data point during the testing phase according to

$$\text{class}(x_i) = \text{sgn}(\langle w', \Phi(x_i) \rangle + b'). \quad (41)$$

From (41) we see that we are able to calculate the new (flat) hypersurface in the new feature (Hilbert) space simply through inner products of $(\Phi(x_i), \Phi(x_j))$.

V. CONstrained Subspace CLASSifier

The idea in the constrained subspace classifier (CSC) method is similar to the idea used in 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 $X'$ (equation (10)), onto a $d_1$-dimensional subspace $S_1$, (of dimensionality $d_1 < d$) of the $d$-dimensional space ($d = 550$) and also project the injection vectors, rows of $X'_2$ (equation (11)), 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
implies that the columns of the matrix whose columns are orthonormal, this implies that its rows are also orthonormal. Orthonormality of the rows of $U_d$ implies $U_dU_d^\top = I_d$ (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

$$U_d^\top = U_d^{-1}.$$  

### A. The projection operator

Let $S$ be a data space of dimension equal to the number of features, $d$, of the selected dataset (for our study $d$ is the dimensionality of the ft-maps after the resolution reduction i.e. $d = 550$). We can always find an orthonormal basis for $S$ (using the Gram-Schmidt process) given by

$$U_d = \{ u_1, u_2, \ldots, u_d \} \text{ with } u_i \in \mathbb{R}^d \quad \forall i = 1, 2, \ldots, d \quad (42)$$

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

$$U_{d_1} = \{ u_1, u_2, \ldots, u_{d_1} \} \text{ with } u_i \in \mathbb{R}^d \quad \forall i = 1, 2, \ldots, d_1 \quad (43)$$

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

$$P = Q(Q^TQ)^{-1}Q^T$$

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 (43), that is $Q = U_{d_1}$. In that case, equation (44) becomes

$$P = U_{d_1}(U_{d_1}^\top U_{d_1})^{-1}U_{d_1}^\top$$

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

Since equation (42) is an orthonormal basis for $\mathbb{R}^d$ then $U_d^\top U_d = I_d$. 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

$$P = U_d U_d^\top.$$  

In case $d_1 = d$ then $P = U_d U_d^\top$. Since $U_d$ is a square matrix whose 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

### B. Principal component analysis (PCA)

To introduce PCA we will use the definition of the data matrix $X'_d$ as given by equation (10). Using the projection operator as given by expression (45) we want to project the ft-maps of $X'_d$ 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 $x_i$ onto $\mathbb{R}^{d_1}$ thus defining $x_i^\top = U_{d_1} U_{d_1}^\top x_i^\top$. Then the norm of the difference between the original and the projected (column) vectors can be expressed as

$$\|x_i^\top - x_i^\top\| = \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|$$

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

$$\sum_{i=1}^n \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|^2$$

is minimized

subject to $U_{d_1}^\top U_{d_1} = I_{d_1}$.

This subspace $\mathbb{R}^{d_1}$ is defined as the $d_1$-dimensional hypersurface that is spanned by the (reduced) orthonormal basis $\{u_1, u_2, u_3, \ldots, u_{d_1}\}$, 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 $A$,

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

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

$$\sum_{i=1}^n \|x_i^\top - U_{d_1} U_{d_1}^\top x_i^\top\|^2 = \text{tr} \left\{ X'_d^\top X'_d (I - U_{d_1} U_{d_1}^\top) \right\}$$

where $X'_d \in \mathbb{R}^{n \times d}$ (where $n = 22700$ and $d = 550$ as shown in equation (10)). Thus the optimization problem in equation (51) reduces to

$$\min_{U_{d_1}} \text{tr} \left\{ X'_d^\top X'_d (I - U_{d_1} U_{d_1}^\top) \right\}$$

subject to $U_{d_1}^\top U_{d_1} = I_{d_1}$.

Since $\text{tr} \left\{ X'_d^\top X'_d \right\}$ is a constant, the optimization problem can be re-written as

$$\max_{U_{d_1}} \text{tr} \left\{ U_{d_1} X'_d^\top X'_d U_{d_1} \right\}$$

subject to $U_{d_1}^\top U_{d_1} = I_{d_1}$. 

(53)
To solve equation (53) we define the Lagrangian dual problem by

\[ \mathcal{L}(U_{d_1}, \lambda_{ij}) = \text{tr}(U_{d_1}^T X_1^{T} X_1^T U_{d_1} - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left( \sum_{k=1}^{d} U_{jk} U_{ki} - \delta_{ji} \right) ) \]

\[ - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left( \sum_{k=1}^{d} U_{jk} U_{ki} - \delta_{ji} \right) \] (54)

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

Since \( U_{d_1}^T U_{d_1} \) is a symmetric \( d_1 \times d_1 \) matrix then the orthonormality condition in equation (53) represents a total of \( d_1 \times (d_1 + 1)/2 \) conditions. Therefore, for the Lagrangian dual problem (as shown in equation (54)) we need to introduce \( d_1 \times (d_1 + 1)/2 \) Lagrange multipliers \( \lambda_{ij} \). Hence we require that \( \lambda_{ij} \) is a symmetric matrix. Also since each term in (54) involves symmetric matrices then the following first order optimality conditions

\[ \frac{\partial \mathcal{L}}{\partial \lambda_{pq}} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial U_{lm}} = 0. \] (55)

can be solved for \( \lambda_{ij} \) only if the latter is symmetric. Using equations (54) and (55) we get

\[ \frac{\partial}{\partial \lambda_{pq}} \left[ \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} U_{ij}^T (X^T X)_{jk} U_{ki} - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left( \sum_{k=1}^{d} U_{jk} U_{ki} - \delta_{ji} \right) \right] = 0 \]

and

\[ \frac{\partial}{\partial U_{lm}} \left[ \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} U_{ij}^T (X^T X)_{jk} U_{ki} - \sum_{i=1}^{d_1} \sum_{j=1}^{d_1} \lambda_{ij} \left( \sum_{k=1}^{d} U_{jk} U_{ki} - \delta_{ji} \right) \right] = 0. \] (56)

Equation (56) implies the \( d_1 \times (d_1 + 1)/2 \) equations

\[ \sum_{k=1}^{d} U_{jk} U_{kp} = \delta_{qp} \] (58)

while equation (57) implies the \( d \times d_1 \) equations

\[ \sum_{j=1}^{d} U_{mj}^T (X_1^T X_1^T)_{jl} U_{lk} = \sum_{k=1}^{d} (X_1^T X_1^T)_{jk} U_{km} - \sum_{j=1}^{d_1} \lambda_{mj} U_{jl} - \sum_{i=1}^{d} \lambda_{im} U_{ii} = 0. \] (59)

Using the fact that \( X_1^T X_1^T \) is symmetric, the first two terms of equation (59) can be combined to a single term and similarly (using the symmetry of \( \lambda_{ij} \)) the last two terms of equation (59) can be combined to a single term to get

\[ \sum_{j=1}^{d} U_{mj}^T (X_1^T X_1^T)_{jl} - \sum_{i=1}^{d_1} \lambda_{mi} U_{il}^T U_{il} = 0. \] (60)

Equations (60) and (58) are sufficient to solve for \( \lambda_{ij} \) and \( U_{ik} \). Right-multiplying equation (60) by \( U_{ln} \) and summing over \( 1 \leq l \leq d \) we get

\[ \sum_{l=1}^{d} \sum_{j=1}^{d} U_{mj}^T (X_1^T X_1^T)_{jl} U_{ln} - \sum_{i=1}^{d_1} \lambda_{mi} \sum_{l=1}^{d} U_{il}^T U_{ln} = 0. \] (61)

Using equation (58) then equation (61) becomes

\[ \sum_{l=1}^{d} \sum_{j=1}^{d} U_{mj}^T (X_1^T X_1^T)_{jl} U_{ln} = \lambda_{mn}. \] (62)

Equations (58) and (60) 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 \( \lambda_{ij} \) and the \( d_1 \times d \) degrees of freedom of \( U_{d_1} \).

The left hand side (LHS) of equation (62) represents the \( a_{mn} \) elements of a \( d_1 \times d_1 \) matrix and similarly the right hand side (RHS) of (62) represents the \( \lambda_{mn} \) elements of another \( d_1 \times d_1 \) matrix. Equation (62) implies an entry-by-entry equation \((a_{mn} = \lambda_{mn})\) between the two matrices. Choosing \( m = n \) and summing equation (62) 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

\[ \sum_{m=1}^{d_1} \sum_{l=1}^{d} \sum_{j=1}^{d} U_{mj}^T (X_1^T X_1^T)_{jl} U_{ln} = \sum_{m=1}^{d_1} \lambda_{mm}. \] (63)

Noting that the LHS of (63) is the trace of the LHS of (62) we can re-write (63) as

\[ \text{tr}(U_{d_1} U_{d_1}^T X_1^T X_1^T U_{d_1}) = \sum_{m=1}^{d_1} \lambda_{mm}. \] (64)

To interpret the \( \lambda_{mm} \) 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 \( \lambda_{mm} \) for \( 1 \leq m \leq d_1 \) as the eigenvalues of the symmetric matrix \((X_1 U_{d_1})^T (X_1 U_{d_1})\). However, these \( d_1 \) eigenvalues are \( d_1 \) out of the total \( d \) eigenvalues of \( X_1^T X_1 \). This can be shown by using the invariance of trace under similarity transformations (in this case under conjugacy). Using equation (64) we can re-write equation (64) for \( d_1 = d \) as

\[ \text{tr}(U_{d}^{-1} X_1^T X_1^T U_{d}) = \text{tr}(X_1^T X_1^T) = \sum_{m=1}^{d} \lambda_{mm}. \] (65)

Therefore, the maximum of the objective function \( F = \text{tr}(U_{d}^{-1} X_1^T X_1^T U_{d}) \) in expression (53) 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 \( X_1^T X_1 \).
C. Formulation of CSC

Consider the binary classification problem with \(X_1' \in \mathbb{R}^{n \times d}\) and \(X_2' \in \mathbb{R}^{n \times d}\) be the data matrices corresponding to two data classes, \(C_1\) (noise points) and \(C_2\) (injection points) respectively. The number of data samples in \(C_1\) is the same as the number of data samples in \(C_2\) and is equal to \(n/2\). The corresponding number of features is given by \(d\) for both classes \(C_1\) and \(C_2\) (in our case \(n/2 = 11550\) and \(d = 550\)).

We attempt to find two linear subspaces \(S_1 \subseteq C_1\) and \(S_2 \subseteq C_2\) 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

\[
U = [u_1, u_2, \ldots, u_{d_1}] \in \mathbb{R}^{d \times d_1}
\]

and

\[
V = [v_1, v_2, \ldots, v_{d_1}] \in \mathbb{R}^{d \times d_1}
\]

represent matrices whose columns are orthonormal bases of the subspaces \(S_1\) and \(S_2\) respectively. If we attempted to find \(S_1\) independently from \(S_2\) then we would have to capture the maximal variance of the data projected onto \(S_1\) separately from the maximal variance of the data projected onto \(S_2\). That would be equivalent to solving the following two optimization problems [24]

\[
\max_{U \in \mathbb{R}^{d \times d_1}} \text{tr}(U^TX_1'^TX_1'(U))
\]

subject to \(U^TU = I_{d_1}\)

and

\[
\max_{V \in \mathbb{R}^{d \times d_1}} \text{tr}(V^TX_2'^TX_2'(V))
\]

subject to \(V^TV = I_{d_1}\).

The solution to the optimization problem as shown in expression (68) is given by the eigenvectors (the columns of the orthonormal basis \(U\) of \(S_1\)) corresponding to the \(d_1\) largest eigenvalues of the matrix \(X_1'^TX_1'\). Similarly, the solution to the optimization problem as shown in expression (69) is given by the eigenvectors (the columns of the orthonormal basis \(V\) of \(S_2\)) corresponding to the \(d_1\) largest eigenvalues of the matrix \(X_2'^TX_2'\). Though the subspaces \(S_1\) and \(S_2\) are good approximations to the two classes \(C_1\) and \(C_2\) 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 constrained subspace classifier (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

\[
\max_{U, V \in \mathbb{R}^{d \times d_1}} \text{tr}(U^TX_1'^TX_1'(U)) + \text{tr}(V^TX_2'^TX_2'(V)) + C \text{tr}(U^TvV^TU)
\]

subject to \(U^TU = I_{d_1}, \ V^TV = I_{d_1}\).

The last term of the objective function \(G = \text{tr}(U^TX_1'^TX_1'(U)) + \text{tr}(V^TX_2'^TX_2'(V)) + C \text{tr}(U^TVV^TU)\) is a measure of the relative orientation between the two subspaces as defined in [8]. 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 [8]. For a fixed \(V\), expression (70) reduces to

\[
\max_{U \in \mathbb{R}^{d \times d_1}} \text{tr}(U^TX_1'^TX_1'(U)) + C \text{tr}(U^TVV^TU)
\]

subject to \(U^TU = I_{d_1}\).

The solution to the optimization problem (71) is obtained by choosing the eigenvectors corresponding to the \(d_1\) largest eigenvalues of the symmetric matrix \(X_1'^TX_1' + CVV^T\). Similarly, for a fixed \(U\), expression (70) reduces to

\[
\max_{V \in \mathbb{R}^{d \times d_1}} \text{tr}(V^TX_2'^TX_2'(V)) + CUU^TU
\]

subject to \(V^TV = I_{d_1}\).

where the solution to the optimization problem (72) is again obtained by choosing the eigenvectors corresponding to the \(d_1\) largest eigenvalues of the symmetric matrix \(X_2'^TX_2' + CUU^T\).

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 \(U^TU = I_{d_1}, V^TV = I_{d_1}\).
2. Find eigenvectors corresponding to the \(d_1\) largest eigenvalues of the symmetric matrix \(X_1'^TX_1' + CVV^T\).
3. Find eigenvectors corresponding to the \(d_1\) largest eigenvalues of the symmetric matrix \(X_2'^TX_2' + CUU^T\).
4. Alternate between 2 and 3 until one of the termination rules below is satisfied.

We define the following three termination rules:

- **Maximum limit \(Z\) on the number of iterations,**
- **Relative change in \(U\) and \(V\) at iteration \(m\) and \(m+1\),**

\[
tol_U^m = \frac{\|U^{(m+1)} - U^{(m)}\|_F}{\sqrt{N}},
\]

\[
tol_V^m = \frac{\|V^{(m+1)} - V^{(m)}\|_F}{\sqrt{N}},
\]

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

- **Relative change in the value of the objective function \(G\) as shown in expression (70) at iteration \(m\) and \(m+1\),**

\[
tol^m_j = \frac{G^{(m+1)} - G^{(m)}}{|G^{(m)}| + 1}.
\]
The value of $Z$ was set to 2000, while $tol^m_r$, $tol^m_{f_r}$ and $tol^m_v$ are all set at the same value of $10^{-6}$. From equation (50) we see that the factor of $1/\sqrt{N}$ in (73) 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 tolerances values independent of the data set.

After solving the optimization problem (70) (by utilizing algorithm 2) a new point $x$ is classified by computing the distances from the two subspaces $S_1$ and $S_2$ defined by

$$\text{dist}(x, S_1) = \text{tr}(U^T x^T x U)$$

and

$$\text{dist}(x, S_2) = \text{tr}(V^T x^T x V).$$

The class of $x$ is defined by

$$\text{class}(x) = \arg\{ \min_{i \in \{1, 2\}} \{\text{dist}(x, S_i)\} \}. \quad (77)$$

In our case, if $x$ is closer to $S_1$ then $x$ is classified as noise (or ‘no signal’) and if $x$ is closer to $S_2$ then $x$ is classified as a $r$-mode injection (or ‘presence of signal’).

### VI. RESULTS AND DISCUSSION

When using the conventional clustering algorithm in [9] the false alarm rate (FAR), or false positives, is easily controlled by adjusting the SNR threshold above which a ft-map is considered to include a $r$-mode signal. This is not the case for the MLAs we used where the FAR is given after the training is performed as part of the training output. For this reason, to draw fair comparisons, we adjusted the FAR of the conventional algorithm to match the output FAR of the MLAs. In each of the tables II-VI, the results of the conventional algorithm are presented in 4-tuples: the first entry corresponds to a sensitivity result with a fixed FAR equal to 0.1% and the second, third and forth entries are results corresponding to the same FAR as that of the ANN, SVM and CSC respectively. The results presented on tables III, IV and VI are also plotted in Fig.4, Fig.5 and Fig.6 respectively.

Table I presents the results of the conventional algorithm and the three MLAs on the $(\alpha = 0.1, f_o = 1500\text{ Hz})$ waveform. The latter were trained with data produced with $h$ taking values over the range of $10^{-24.0} \leq h \leq 10^{-21}$. The MLAs did not outperform the conventional algorithm. The number of data that was produced was limited due to the finite amount of data that was produced during the S5 LIGO run. This amount of data was too small for the MLAs to achieve generalization, hence the training efficiencies are too low. To avoid this the next steps involved training of the same number of data over smaller ranges of values of $h$.

In Table II we present the detection efficiency results for the conventional algorithm and the three MLAs on the $(\alpha = 0.1, f_o = 1500\text{ Hz})$ waveform. The MLAs were trained with data produced with $h$ taking values over the range of $10^{-23.2} \leq h \leq 10^{-22.7}$. There was not a lot of expectation that the MLAs would outperform the conventional algorithms because the MLAs were trained on a data set whose injection distances were lower than the distance at which the conventional algorithm had a 50% detection efficiency.

In table III we present the detection efficiency results for the conventional algorithm and the three MLAs on the $(\alpha = 0.1, f_o = 1500\text{ Hz})$ waveform. The latter were trained with data produced with $h$ taking values over the range of $10^{-23.7} \leq h \leq 10^{-23.2}$. The training of the MLAs on this training set resulted in false alarm rates 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 $\sim 20\%$ in the detection distance - from 1.5Mpc (of the conventional algorithm dash-dot blue line) to 1.8Mpc. The SVM shows an increase of $\sim 16\%$ - from 1.55Mpc (of the conventional algorithm dash-dot green line) to 1.8Mpc. The CSC shows an increase of $\sim 10\%$ - from 1.6Mpc (of the conventional algorithm dash-dot red line) to 1.75Mpc.

In table IV we present the detection efficiency results for the conventional algorithm and the three MLAs on the $(\alpha = 0.1, f_o = 1500\text{ Hz})$ waveform. The latter were trained with data produced with $h$ taking values over the range of $10^{-24.0} \leq h \leq 10^{-23.7}$. The training of the MLAs on this training set resulted in high false alarm rates of 18%, 22% and 36% for the ANN, SVM and CSC respectively. At the 50% FDR, both the ANN and the SVM algorithms show an increase of $\sim 75\%$ in the detection distance - from 1.6Mpc (of the conventional algorithm dash-dot blue and green lines) to 2.8Mpc. The CSC shows no increase - both dash-dot and solid red lines stay at 50% up to distances of $\sim 2.9\text{Mpc}$. The distance range covered in this set has a practical significance because it covers: (a) the distance of 3.5Mpc at which the January 2014 supernova occured in M82 and (b) the distance of 5Mpc at which the supernova event rate in the Milky Way neighborhood is about 1 every 1-2 years.

In Table V we present the detection efficiency results for the conventional algorithm and the three MLAs on the $(\alpha = 0.01, f_o = 1100\text{ Hz})$ waveform. This is a much weaker signal than the $(\alpha = 0.1, f_o = 1500\text{ Hz})$ as can be seen from [9]. The MLAs were trained with data produced with $h$ taking values over the range of $10^{-23.7} \leq h \leq 10^{-23.2}$. The MLAs did not outperform the conventional algorithms. This was expected because the MLAs were trained on a data set whose injection distances were lower than the distance at which the conventional algorithm had a 50% detection efficiency.

In Table VI we present the detection efficiency results for the conventional algorithm and the three MLAs on the $(\alpha = 0.01, f_o = 1100\text{ Hz})$ waveform. The MLAs were trained with data produced with $h$ taking values over the range of $10^{-23.7} \leq h \leq 10^{-23.2}$. The training of the MLAs on this training set resulted in false alarm rates of 18%, 22% and 36% for the ANN, SVM and CSC respectively. At the 50% false dismissal rate (FDR), the ANN shows an increase of $\sim 20\%$ in the detection distance - from 175Kpc (of the conventional algorithm dash-dot blue line) to 210Kpc. The SVM shows no increase while the CSC shows a small increase.
while the distance at which the conventional algorithm detects 50% in the range of $1.1350 \times 10^{-24}$ to $1.1350 \times 10^{-21}$. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_0 \leq 1600$ and $h$ values distributed in the range of $10^{-24} \leq h \leq 10^{-21}$.

| Distance ($\times 1170$ Kpc) | Signal amplitude ($h$) | Conventional (%) | ANN (%) | SVM (%) | CSC (%) |
|-------------------------------|------------------------|------------------|---------|---------|---------|
| 0.1                           | $4.33 \times 10^{-23}$  | 100              | 100     | 100     | 0       |
| 0.2                           | $2.16 \times 10^{-23}$  | 100              | 100     | 100     | 0       |
| 0.3                           | $1.44 \times 10^{-23}$  | 100              | 100     | 96      | 0       |
| 0.4                           | $1.08 \times 10^{-23}$  | 100              | 83      | 47      | 0       |
| 0.5                           | $0.85 \times 10^{-24}$  | 100              | 42      | 13      | 0       |
| 0.6                           | $0.72 \times 10^{-24}$  | 100              | 17      | 0       | 0       |
| 0.7                           | $0.61 \times 10^{-24}$  | 100              | 3       | 0       | 0       |
| 0.8                           | $0.54 \times 10^{-24}$  | 98               | 1       | 0       | 0       |
| 0.9                           | $0.48 \times 10^{-24}$  | 76               | 0       | 0       | 0       |
| 1.0                           | $0.43 \times 10^{-24}$  | 50               | 0       | 0       | 0       |
| 1.1                           | $0.39 \times 10^{-24}$  | 29               | 0       | 0       | 0       |
| 1.2                           | $0.36 \times 10^{-24}$  | 13               | 0       | 0       | 0       |
| 1.3                           | $0.33 \times 10^{-24}$  | 4                | 0       | 0       | 0       |
| 1.4                           | $0.30 \times 10^{-24}$  | 4                | 0       | 0       | 0       |
| 1.5                           | $0.28 \times 10^{-24}$  | 1                | 0       | 0       | 0       |

$^a$ 1170 Kpc is the distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.
$^b$ Calculated using (5) and substituting the parameter values $\alpha = 0.1$ and $f_0 = 1500$ Hz and a distance given by the first column.
$^c$ Results are based on detection efficiencies on full resolution maps.
$^d$ Highest training efficiency (99%) with parameter values: momentum=0.9, learning rate=0.02. True positive 98%. False positive: < 0.01%
$^e$ Highest training efficiency (98%) with parameter values: $C = 10^3$, True positive: 98%. False positive: < 0.01%
$^f$ Highest training efficiency (90%) with parameter values: $d_1 = 100, C = 1$. True positive: 80%. False positive: < 0.01%

### TABLE II. Detection efficiencies for injections with waveform parameters $\alpha = 0.1$ and $f_0 = 1500$ Hz. The training was performed with 11350 noise maps and 11350 injection maps. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_0 \leq 1600$ and $h$ values distributed in the range of $6.31 \times 10^{-24} = 10^{-23.2} \leq h \leq 10^{-22.7} = 2.00 \times 10^{-23}$. The signal amplitudes that lie within this range are in blue text while the distance at which the conventional algorithm detects 50% with FAR = 0.1% is in red text.

| Distance ($\times 1170$ Kpc) | Signal amplitude ($h$) | Conventional (%) | ANN (%) | SVM (%) | CSC (%) |
|-------------------------------|------------------------|------------------|---------|---------|---------|
| 0.1                           | $4.33 \times 10^{-23}$  | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.2                           | $2.16 \times 10^{-23}$  | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.3                           | $1.44 \times 10^{-23}$  | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.4                           | $1.08 \times 10^{-23}$  | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.5                           | $0.85 \times 10^{-24}$  | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.6                           | $0.72 \times 10^{-24}$  | (100, 100, 100, 100) | 100     | 100     | 96      |
| 0.7                           | $0.61 \times 10^{-24}$  | (100, 100, 100, 100) | 100     | 100     | 97      |
| 0.8                           | $0.54 \times 10^{-24}$  | (98, 97, 97, 97)    | 89      | 81      | 64      |
| 0.9                           | $0.48 \times 10^{-24}$  | (76, 96, 87, 96)    | 54      | 12      | 20      |
| 1.0                           | $0.43 \times 10^{-24}$  | (50, 90, 73, 93)    | 44      | 9       | 19      |
| 1.1                           | $0.39 \times 10^{-24}$  | (29, 67, 45, 69)    | 32      | 2       | 9       |
| 1.2                           | $0.36 \times 10^{-24}$  | (13, 39, 21, 41)    | 33      | 2       | 7       |
| 1.3                           | $0.33 \times 10^{-24}$  | (4, 15, 9, 19)      | 22      | 0       | 10      |
| 1.4                           | $0.30 \times 10^{-24}$  | (4, 8, 4, 11)       | 13      | 0       | 6       |
| 1.5                           | $0.28 \times 10^{-24}$  | (1, 6, 2, 7)        | 8       | 0       | 10      |

$^a$ 1170 Kpc is the distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.
$^b$ Calculated using (5) and substituting the parameter values $\alpha = 0.1$ and $f_0 = 1500$ Hz and a distance given by the first column.
$^c$ These 4-tuples are detection efficiencies with FAR= (0.1%, 0.7%, 0.2%, 1%) that were obtained on full resolution maps. The second, third and forth entries are to be compared with the ANN, SVM and CSC results respectively.
$^d$ Highest training efficiency (98%) with parameter values: momentum=0.9, learning rate=0.02. True positive 97%. False positive: 0.7%
$^e$ Highest training efficiency (98%) with parameter values: $C = 10^3$, True positive: 96%. False positive: 0.2%
$^f$ Highest training efficiency (95%) with parameter values: $d_1 = 100, C = 10^3$. True positive: 92%. False positive: 1%.
TABLE III. Detection efficiencies for injections with waveform parameters $\alpha = 0.1$ and $f_o = 1500 \text{ Hz}$. The training was performed with 11350 noise maps and 11350 injection maps. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_o \leq 1600$ and $h$ values distributed in the range of $2.00 \times 10^{-24} = 10^{-23.7} \leq h \leq 10^{-21.2} = 6.31 \times 10^{-24}$. The signal amplitudes that lie within this range are in blue text while the distance at which the conventional algorithm detects 50% with FAR = 0.1% is in red text.

| Distance ($\times 1170 \text{ Kpc}$) | Signal amplitude ($h$) | Conventional (%) | ANN (%) | SVM (%) | CSC (%) |
|---------------------------------|-----------------|----------------|--------|--------|--------|
| 0.1                             | $4.33 \times 10^{-23}$ | (100, 100, 100, 100) | 100    | 100    | 100    |
| 0.2                             | $2.16 \times 10^{-23}$ | (100, 100, 100, 100) | 100    | 100    | 100    |
| 0.3                             | $1.44 \times 10^{-23}$ | (100, 100, 100, 100) | 100    | 100    | 100    |
| 0.4                             | $1.08 \times 10^{-23}$ | (100, 100, 100, 100) | 100    | 100    | 100    |
| 0.5                             | $8.65 \times 10^{-24}$ | (100, 100, 100, 100) | 100    | 100    | 100    |
| 0.6                             | $7.21 \times 10^{-24}$ | (100, 99, 100, 100) | 100    | 100    | 100    |
| 0.7                             | $6.18 \times 10^{-24}$ | (100, 98, 98, 98) | 999    | 999    | 999    |
| 0.8                             | $5.41 \times 10^{-24}$ | (98, 98, 98, 98) | 999    | 989    | 989    |
| 0.9                             | $4.81 \times 10^{-24}$ | (76, 98, 98, 98) | 979    | 929    | 949    |
| 1.0                             | $4.33 \times 10^{-24}$ | (50, 97, 98, 98) | 909    | 909    | 889    |
| 1.1                             | $3.93 \times 10^{-24}$ | (29, 85, 87, 94) | 889    | 909    | 779    |
| 1.2                             | $3.61 \times 10^{-24}$ | (13, 63, 67, 79) | 929    | 939    | 709    |
| 1.3                             | $3.33 \times 10^{-24}$ | (4, 41, 55, 64) | 779    | 799    | 689    |
| 1.4                             | $3.09 \times 10^{-24}$ | (4, 21, 25, 38) | 749    | 749    | 679    |
| 1.5                             | $2.88 \times 10^{-24}$ | (1, 9, 13, 24) | 699    | 709    | 559    |
| 1.6                             | $2.70 \times 10^{-24}$ | (0, 19, 23, 36) | 459    | 449    | 379    |
| 1.7                             | $2.55 \times 10^{-24}$ | (0, 19, 23, 30) | 409    | 429    | 309    |
| 1.8                             | $2.40 \times 10^{-24}$ | (0, 16, 16, 19) | 459    | 499    | 359    |
| 1.9                             | $2.28 \times 10^{-24}$ | (0, 27, 31, 37) | 359    | 299    | 309    |
| 2.0                             | $2.16 \times 10^{-24}$ | (0, 17, 21, 30) | 269    | 239    | 259    |
| 2.1                             | $2.06 \times 10^{-24}$ | (0, 13, 15, 22) | 339    | 279    | 239    |
| 2.2                             | $1.97 \times 10^{-24}$ | (0, 20, 24, 29) | 309    | 239    | 279    |
| 2.3                             | $1.88 \times 10^{-24}$ | (0, 29, 33, 41) | 309    | 299    | 269    |
| 2.4                             | $1.80 \times 10^{-24}$ | (0, 16, 22, 33) | 189    | 169    | 169    |
| 2.5                             | $1.73 \times 10^{-24}$ | (0, 20, 26, 33) | 309    | 199    | 209    |
| 2.6                             | $1.66 \times 10^{-24}$ | (0, 3, 4, 11) | 89     | 69     | 89     |
| 2.7                             | $1.60 \times 10^{-24}$ | (0, 1, 2, 9) | 169    | 119    | 199    |
| 2.8                             | $1.55 \times 10^{-24}$ | (0, 8, 10, 16) | 159    | 129    | 179    |
| 2.9                             | $1.49 \times 10^{-24}$ | (0, 4, 5, 8) | 269    | 189    | 209    |
| 3.0                             | $1.44 \times 10^{-24}$ | (0, 1, 3, 7) | 159    | 79     | 229    |

$^a$ 1170 Kpc is the distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.

$^b$ Calculated using $[5]$ and substituting the parameter values $\alpha = 0.1$ and $f_0 = 1500 \text{ Hz}$ and a distance given by the first column.

$^c$ These 4-tuples are detection efficiencies with FAR= (0.1%, 4%, 5%, 10%) that were obtained on full resolution maps. The second, third and forth entries are to be compared with the ANN, SVM and CSC results respectively.

$^d$ Highest training efficiency (88%) with parameter values: momentum=0.9, learning rate=0.02. True positive: 91%. False positive: 4%.

$^e$ Highest training efficiency (89%) with parameter values: C=10^5. True positive: 88%. False positive: 5%.

$^f$ Highest training efficiency (83%) with parameter values: $d_o=100$, C= 10^4. True positive: 77%. False positive: 10%.

$^g$ This is the 3.5 Mpc distance at which the M82 supernova exploded in January 2014.
FIG. 4. These are the detection efficiencies for the \((f_o = 1500 \text{ Hz}, \alpha = 0.1)\) waveform. This waveform produces the most powerful signal that the 1998 model predicts. This plot demonstrates that (when compared for the same FAR) the MLAs performance is at least as good as that of the conventional algorithm. At the 50% false dismissal rate (FDR), the ANN shows an increase of \(\sim 20\%\) in the detection distance - from 1.5Mpc (of the conventional algorithm dash-dot blue line) to 1.8Mpc. The SVM shows an increase of \(\sim 16\%\) - from 1.55Mpc (of the conventional algorithm dash-dot green line) to 1.8Mpc. The CSC shows an increase of \(\sim 10\%\) - from 1.6Mpc (of the conventional algorithm dash-dot red line) to 1.75Mpc.
TABLE IV. Detection efficiencies for injections with waveform parameters $\alpha = 0.1$ and $f_o = 1500$ Hz. The training was performed with 11350 noise maps and 11350 injection maps. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_o \leq 1600$ and $h$ values distributed in the range of $10^{-24.0} \leq h \leq 10^{-23.7} = 2.00 \times 10^{-24}$. The signal amplitudes that lie within this range are in blue text while the distance at which the conventional algorithm detects 50% with FAR = 0.1% is in red text.

| Distance ($\times 1170$ Kpc) | Signal amplitude $(h)$ | Conventional (%) | ANN (%) | SVM (%) | CSC (%) |
|----------------------------|-----------------------|------------------|---------|---------|---------|
| 0.4 | $1.08 \times 10^{-23}$ | (100, 100, 100, 100) | 100 | 100 | 100 |
| 0.5 | $6.53 \times 10^{-24}$ | (100, 100, 100, 100) | 100 | 100 | 100 |
| 0.6 | $7.21 \times 10^{-24}$ | (100, 98, 98, 98) | 100 | 100 | 100 |
| 0.7 | $6.18 \times 10^{-24}$ | (100, 97, 97, 97) | 100 | 100 | 98 |
| 0.8 | $5.41 \times 10^{-24}$ | (98, 98, 98, 98) | 97 | 98 | 95 |
| 0.9 | $4.81 \times 10^{-24}$ | (76, 98, 98, 98) | 98 | 100 | 93 |
| 1.0 | $4.33 \times 10^{-24}$ | (50, 98, 98, 98) | 93 | 98 | 91 |
| 1.1 | $3.93 \times 10^{-24}$ | (29, 96, 96, 98) | 93 | 98 | 83 |
| 1.2 | $3.61 \times 10^{-24}$ | (13, 92, 92, 96) | 96 | 99 | 92 |
| 1.3 | $3.33 \times 10^{-24}$ | (4, 65, 65, 73) | 91 | 93 | 82 |
| 1.4 | $3.09 \times 10^{-24}$ | (4, 50, 50, 64) | 94 | 92 | 81 |
| 1.5 | $2.88 \times 10^{-24}$ | (1, 37, 37, 49) | 92 | 90 | 78 |
| 1.6 | $2.70 \times 10^{-24}$ | (0, 46, 46, 52) | 68 | 72 | 61 |
| 1.7 | $2.55 \times 10^{-24}$ | (0, 44, 44, 54) | 70 | 70 | 61 |
| 1.8 | $2.40 \times 10^{-24}$ | (0, 35, 35, 51) | 77 | 72 | 63 |
| 1.9 | $2.28 \times 10^{-24}$ | (0, 49, 49, 57) | 55 | 61 | 51 |
| 2.0 | $2.16 \times 10^{-24}$ | (0, 43, 43, 55) | 55 | 59 | 56 |
| 2.1 | $2.06 \times 10^{-24}$ | (0, 41, 41, 52) | 62 | 61 | 61 |
| 2.2 | $1.97 \times 10^{-24}$ | (0, 42, 42, 50) | 61 | 64 | 59 |
| 2.3 | $1.88 \times 10^{-24}$ | (0, 50, 50, 57) | 61 | 61 | 59 |
| 2.4 | $1.80 \times 10^{-24}$ | (0, 45, 45, 53) | 52 | 54 | 50 |
| 2.5 | $1.73 \times 10^{-24}$ | (0, 42, 42, 48) | 50 | 49 | 52 |
| 2.6 | $1.66 \times 10^{-24}$ | (0, 28, 28, 35) | 55 | 37 | 44 |
| 2.7 | $1.60 \times 10^{-24}$ | (0, 21, 21, 40) | 76 | 46 | 58 |
| 2.8 | $1.55 \times 10^{-24}$ | (0, 24, 24, 33) | 67 | 42 | 50 |
| 2.9 | $1.49 \times 10^{-24}$ | (0, 22, 22, 37) | 61 | 48 | 59 |
| 3.0 | $1.44 \times 10^{-24}$ | (0, 14, 14, 26) | 64 | 42 | 51 |
| 3.1 | $1.40 \times 10^{-24}$ | (0, 36, 36, 40) | 47 | 43 | 41 |
| 3.2 | $1.35 \times 10^{-24}$ | (0, 34, 34, 44) | 41 | 39 | 41 |
| 3.3 | $1.31 \times 10^{-24}$ | (0, 34, 34, 44) | 51 | 42 | 43 |
| 3.4 | $1.27 \times 10^{-24}$ | (0, 54, 54, 57) | 40 | 41 | 40 |
| 3.5 | $1.24 \times 10^{-24}$ | (0, 42, 42, 57) | 41 | 36 | 46 |
| 3.6 | $1.20 \times 10^{-24}$ | (0, 41, 41, 47) | 43 | 46 | 49 |
| 3.7 | $1.17 \times 10^{-24}$ | (0, 44, 44, 48) | 43 | 44 | 46 |
| 3.8 | $1.14 \times 10^{-24}$ | (0, 54, 54, 63) | 48 | 53 | 55 |
| 3.9 | $1.11 \times 10^{-24}$ | (0, 38, 38, 48) | 43 | 43 | 43 |
| 4.0 | $1.08 \times 10^{-24}$ | (0, 42, 42, 48) | 35 | 36 | 44 |
| 4.1 | $1.05 \times 10^{-24}$ | (0, 32, 32, 39) | 30 | 26 | 40 |
| 4.2 | $1.03 \times 10^{-24}$ | (0, 21, 21, 40) | 37 | 30 | 44 |
| 4.3 | $1.00 \times 10^{-24}$ | (0, 23, 23, 32) | 34 | 33 | 40 |
| 4.4 | $9.83 \times 10^{-25}$ | (0, 24, 24, 38) | 34 | 41 | 52 |
| 4.5 | $9.61 \times 10^{-25}$ | (0, 19, 19, 31) | 41 | 33 | 48 |

*a* 1170 Kpc is the distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.

*b* Calculated using (5) and substituting the parameter values $\alpha = 0.1$ and $f_o = 1500$Hz and a distance given by the first column.

*c* These 4-tuples are detection efficiencies with FAR=(0.1%, 18%, 22%, 36%) that were obtained on full resolution maps. The second, third and forth entries are to be compared with the ANN, SVM and CSC results respectively.

*d* Highest training efficiency (68%) with parameter values: momentum=0.9, learning rate=0.02. True positive: 72%. False positive: 18%.

*e* Highest training efficiency (64%) with parameter values: $C=10^5$. True positive: 61%. False positive: 22%.

*f* This is the 3.5 Mpc distance at which the supernova event rate (in the Milky Way neighborhood) is 1 every 1-2 years.
FIG. 5. These are the detection efficiencies for the \((f_0 = 1500 \text{ Hz, } \alpha = 0.1)\) waveform. This waveform produces the most powerful signal that the 1998 model predicts. At the 50% FDR, both the ANN and the SVM algorithms show an increase of \(\sim 75\%\) in the detection distance - from 1.6Mpc (of the conventional algorithm dash-dot blue and green lines) to 2.8Mpc. The CSC shows no increase - both dash-dot and solid red lines stay at 50% up to distances of \(\sim 2.9\text{Mpc}\). 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 1 every 1-2 years.
TABLE V. Detection efficiencies for injections with waveform parameters $\alpha = 0.01$ and $f_0 = 1100$ Hz. The training was performed with 11350 noise maps and 11350 injection maps. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_0 \leq 1600$ and $h$ values distributed in the range of $2.00 \times 10^{-24} = 10^{-23.7} \leq h \leq 10^{-23.2} = 6.31 \times 10^{-24}$. The signal amplitudes that lie within this range are in blue text while the distance at which the conventional algorithm detects 50% with FAR = 0.1% is in red text.

| Distance ($\times 133$ Kpc) $^a$ | Signal amplitude ($h$) $^b$ | Conventional (%) | ANN (%) $^d$ | SVM (%) $^e$ | CSC (%) $^e$ |
|-------------------------------|----------------------------|------------------|--------------|-------------|-------------|
| 0.1                           | $1.50 \times 10^{-23}$     | (100, 100, 100, 100) | 100          | 100         | 100         |
| 0.2                           | $7.51 \times 10^{-24}$     | (100, 100, 100, 100) | 100          | 100         | 100         |
| 0.3                           | $5.00 \times 10^{-24}$     | (100, 100, 100, 100) | 99           | 100         | 100         |
| 0.4                           | $3.75 \times 10^{-24}$     | (100, 100, 100, 100) | 86           | 93          | 80          |
| 0.5                           | $3.00 \times 10^{-24}$     | (100, 100, 100, 100) | 69           | 71          | 65          |
| 0.6                           | $2.50 \times 10^{-24}$     | (100, 100, 100, 100) | 58           | 63          | 57          |
| 0.7                           | $2.14 \times 10^{-24}$     | (98, 100, 100, 100)  | 43           | 47          | 43          |
| 0.8                           | $1.88 \times 10^{-24}$     | (92, 100, 100, 100)  | 38           | 50          | 37          |
| 0.9                           | $1.67 \times 10^{-24}$     | (79, 100, 100, 100)  | 24           | 28          | 22          |
| 1.0                           | $1.50 \times 10^{-24}$     | (50, 91, 92, 93)     | 20           | 27          | 21          |
| 1.1                           | $1.36 \times 10^{-24}$     | (23, 69, 70, 75)     | 24           | 7           | 14          |
| 1.2                           | $1.25 \times 10^{-24}$     | (4, 41, 45, 55)      | 27           | 11          | 18          |
| 1.3                           | $1.15 \times 10^{-24}$     | (1, 32, 35, 44)      | 14           | 12          | 16          |
| 1.4                           | $1.07 \times 10^{-24}$     | (1, 19, 21, 25)      | 15           | 18          | 20          |
| 1.5                           | $1.00 \times 10^{-24}$     | (0, 10, 13, 18)      | 14           | 7           | 21          |

$^a$ $133$ Kpc is the distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.

$^b$ Calculated using (5) and substituting the parameter values $\alpha = 0.01$ and $f_0 = 1100$Hz and a distance given by the first column.

$^c$ These 4-tuples are detection efficiencies with FAR=(0.1%, 4%, 5%, 10%) that were obtained on full resolution maps. The second, third and forth entries are to be compared with the ANN, SVM and CSC results respectively.

$^d$ Highest training efficiency (88%) with parameter values: momentum=0.9, learning rate=0.02. True positive: 91%. False positive 4%.

$^e$ Highest training efficiency (89%) with parameter values: C=$10^5$. True positive: 88%. False positive 5%.

$^f$ Highest training efficiency (83%) with parameter values: $d_1=100$, C=$10^4$. True positive: 77%. False positive 10%.
TABLE VI. Detection efficiencies for injections with waveform parameters $\alpha = 0.01$ and $f_0 = 1100$ Hz. The training was performed with 11350 noise maps and 11350 injection maps. The latter were produced with $10^{-3} \leq \alpha \leq 10^{-1}$, $600 \leq f_0 \leq 1600$ and $h$ values distributed in the range of $10^{-24.0} \leq h \leq 10^{-23.7} = 2.00 \times 10^{-24}$. The signal amplitudes that lie within this range are in blue text while the distance at which the conventional algorithm detects 50% with FAR=0.1% is in red text.

| Distance ($\times 133$ Kpc) | Signal amplitude ($h$) | Conventional (%) | ANN (%) | SVM (%) | CSC (%) |
|----------------------------|------------------------|------------------|---------|---------|---------|
| 0.1                        | $1.50 \times 10^{-23}$ | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.2                        | $7.51 \times 10^{-24}$ | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.3                        | $5.00 \times 10^{-24}$ | (100, 100, 100, 100) | 100     | 100     | 100     |
| 0.4                        | $3.75 \times 10^{-24}$ | (100, 100, 100, 100) | 91      | 97      | 85      |
| 0.5                        | $3.00 \times 10^{-24}$ | (100, 100, 100, 100) | 90      | 92      | 75      |
| 0.6                        | $2.50 \times 10^{-24}$ | (100, 100, 100, 100) | 85      | 89      | 79      |
| 0.7                        | $2.14 \times 10^{-24}$ | (98, 100, 100, 100) | 76      | 81      | 68      |
| 0.8                        | $1.88 \times 10^{-24}$ | (92, 100, 100, 100) | 75      | 77      | 66      |
| 0.9                        | $1.67 \times 10^{-24}$ | (79, 100, 100, 100) | 63      | 62      | 57      |
| 1.0                        | $1.50 \times 10^{-24}$ | (50, 95, 95, 95) | 58      | 59      | 56      |
| 1.1                        | $1.36 \times 10^{-24}$ | (23, 83, 83, 86) | 58      | 44      | 47      |
| 1.2                        | $1.25 \times 10^{-24}$ | (4, 74, 74, 80) | 77      | 49      | 59      |
| 1.3                        | $1.15 \times 10^{-24}$ | (1, 52, 52, 62) | 69      | 41      | 49      |
| 1.4                        | $1.07 \times 10^{-24}$ | (1, 45, 45, 54) | 61      | 46      | 58      |
| 1.5                        | $1.00 \times 10^{-24}$ | (0, 30, 30, 43) | 60      | 39      | 50      |
| 1.6                        | $9.38 \times 10^{-25}$ | (0, 30, 30, 40) | 52      | 27      | 37      |
| 1.7                        | $8.83 \times 10^{-25}$ | (0, 21, 21, 30) | 40      | 25      | 39      |
| 1.8                        | $8.34 \times 10^{-25}$ | (0, 33, 33, 43) | 43      | 39      | 45      |
| 1.9                        | $7.90 \times 10^{-25}$ | (0, 32, 32, 43) | 40      | 25      | 38      |
| 2.0                        | $7.51 \times 10^{-25}$ | (0, 24, 24, 33) | 42      | 33      | 43      |

\(^a\) Distance at which the conventional algorithm detects 50% of the signals with FAR=0.1%.

\(^b\) Calculated using (5) and substituting the parameter values $\alpha = 0.01$ and $f_0 = 1100$Hz and a distance given by the first column.

\(^c\) These 4-tuples are detection efficiencies with FAR=(0.1%, 18%, 22%, 36%) that were obtained on full resolution maps. The second, third and forth entries are to be compared with the ANN, SVM and CSC results respectively.

\(^d\) Highest training efficiency (68%) with parameter values: momentum=0.9, learning rate=0.02. True positive: 72%. False positive: 18%.

\(^e\) Highest training efficiency (64%) with parameter values: C=10^5. True positive: 61%. False positive: 22%.

\(^f\) Highest training efficiency (60%) with parameter values: $d_1=200$, C=10^5. True positive: 62%. False positive: 36%.
FIG. 6. These are the detection efficiencies for the \((f_0 = 1100 \text{ Hz}, \alpha = 0.01)\) waveform. This signal was proven (in our previous study) to be detectable only at distances that cover the Milky Way. This signal is approximately monochromatic for the durations our sensitivity studies were designed. At the 50% false dismissal rate (FDR), the ANN shows an increase of \(\sim 20\%\) in the detection distance - from 175Kpc (of the conventional algorithm dash-dot blue line) to 210Kpc. The SVM shows no increase while the CSC shows a very small increase.
FIG. 7. This is one of the noise ft-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. 8. The highest training efficiency for the MLAs was achieved with resolution reduction by a factor of 100 per axis, (Fig. 3). This reduced 10 × 50 resolution ft-map corresponds to the full resolution noise map in Fig. 4. 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.

FIG. 9. This is an injection added to the noise ft-map shown in Fig. 4. 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. 5. This makes it is easier to see the injections in the reduced resolution maps rather than the full resolution ft-maps.

FIG. 10. This reduced 10 × 50 resolution ft-map corresponds to the full resolution map in Fig. 4. Despite the 10000 times reduced resolution as compared to the ft-map of Fig. 6, 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. 3. However, for the parameter estimation algorithms we still use the full resolution ft-maps.
VII. CONCLUSIONS

- **Pipeline suitability**: 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 not only for 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.

- **Computational efficiency**: The most computationally expensive part of this study was the production of the one set of 11350 noise ft-maps and the 3 sets of 11350 injection ft-maps (each set requires up to 10 GB of memory and up to 1 week on a 50 node cluster). The 3 sets of injections examined the 3 different ranges of values for \( h \) (those correspond to 3 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.

- **Training/testing speeds**: Once we have the method (that is presented in this paper) the training of the CSC method requires 10 minutes, the training of the SVM method requires about 30 minutes while the training of the ANN method requires about 8 hours. After the training is done the decision making about the presence of a signal or not takes about 2 seconds for 100 ft-maps. The MLAs are much faster when it comes to the decision making process than the conventional algorithm is (that takes up to 5 minutes for one ft-map).

- **Detection performance**: Comparing table II to table III and table V to table VI, we see that when the training is performed with injections at distances (marked in blue) shorter than the distance, \( d_{red} \) (marked in red) at which the conventional algorithm has a 50% success rate, the MLAs are not efficient enough and do not outperform the conventional algorithm. This is because the MLA training sets (for tables II and V) did not include injections corresponding to distances as long as \( d_{red} \) or beyond. This is what we included in the training sets for the MLAs whose results are shown in tables III and VI. The latter show that when the MLAs are trained with signals injected at distances a little shorter than \( d_{red} \) up to distances 1.5-2 times longer than the latter, then the MLAs performance is at least as good as that of the conventional algorithm. Training the MLAs with injections at distances shorter than \( d_{red} \) was to ensure that the MLAs can detect signals injected at distances \( 0.7 - 0.8 \) that of \( d_{red} \), and training the MLAs with injections at distances longer than \( d_{red} \) was done in order to push the limits of the MLAs and see how much (if at all) they can outperform the conventional algorithm.

- **Low detection efficiency**: for the (0.01, 1100 Hz) waveform. Our suspicion for the low detection efficiencies for the second waveform (weakest signal) we tested is on the resolution reduction factor of \( 10^{-2} \) we used. The plot in Fig[3] was obtained on a study with the strongest signals (0.1, 1500 Hz). We have not tested whether the weaker signals have maximum training efficiencies at a different resolution reduction than the one we used in this study. This needs further investigation.

- **False alarm rates**: In our study FARs of 4-10% (for tables III and V) and 18-36% (for tables IV and VI) are considered very high, however, a more carefully chosen training set may result in lower FARs. 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 \( \alpha \) 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 FAR is by exploring different topologies in the neural network architecture. For SVM and CSC we may introduce a cost function to suppress FAR to acceptable values.

- **Search optimization**: There are many ways that we can further optimize the MLAs specifically designed for the search of r-modes 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% \([25, 26]\). 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.

- **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 conventional algorithm that is generically designed for the detection of any type of signal. Our current method involves the production of at least 10000 ft-maps (that may be overlapping), 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 is the more we may increase the training efficiencies of the MLAs.
• **Future developments:** Future developments include optimization of the current methods as well as the use of other supervised machine learning algorithms such as random forests [27]. 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 [28] which appears to be very promising for image classification.

[1] 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. Reports on Progress in Physics 72, 076901 (2009).
[2] 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).
[3] N. Andersson, and K. D. Kokkotas, International Journal of Modern Physics D 10, 381 (2001).
[4] V. N. Vapnik and V. Vapnik, Statistical learning theory (Wiley New York, 1998).
[5] S. Haykin, Neural Networks Journal 2 (2004).
[6] W. C. Ho and D. Lai, The Astrophysical Journal 543, 386 (2000).
[7] L. Lindblom, B. J. Owen, and S. M. Morsink, Phys. Rev. Lett. 80, 4843 (1998).
[8] O. P. Panagopoulos, V. Pappu, P. Xanthopoulos and P. M. Pardalos, Omega, 10.1016/j.omega.2015.05.009 (2015).
[9] A. Mytidis, M. Coughlin and B. Whiting, arXiv:astro-ph/1505.03191 (2015).
[10] B. J. Owen, L. Lindblom, C. Cutler, et al. 1998, Phys.Rev.D 58, 084020
[11] E. Thrane, S. Kandhasamy, C. D. Ott, et al., Phys.Rev.D 83, 083004 (2011).
[12] E. Thrane and M. Coughlin, arXiv:1308.5292 [astro-ph.IM] (2013).
[13] I. Jolliffe, Principal component analysis (Wiley Online Library, 2002).
[14] I. M. Johnstone and D. M. Titterington, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367, 4237 (2009).
[15] K. P. Murphy, Machine learning: a probabilistic perspective (MIT Press, 2012).
[16] C. M. Bishop, Pattern Recognition and Machine Learning (Springer, 2006).
[17] D. R. Wilson and T. R. Martinez, Neural Networks 16, 1429 (2003).
[18] D. Goldfarb and A. Idnani, Mathematical Programming 27, 1 (1983).
[19] R. Fletcher, Practical methods of optimization (John Wiley &Sons, 2013).
[20] J. Platt et al., Technical report msr-tr-98-14, Microsoft Research (1998).
[21] C. Chang and C. Lin, ACM Transactions on Intelligent Systems and Technology (TIST) 2, 27 (2011).
[22] J. Platt et al., Advances in kernel methods and support vector learning 3 (1999).
[23] R. Vidal, Y. Ra and S. Sastry, Pattern Analysis and Machine Intelligence, IEEE Transactions on 27, 1945 (2005)
[24] J. Laaksonen, Artificial Neural Networks ICANN’97, (Springer, 1997) pp.637-642
[25] D. Branch and G. A. Tammann, Annual Review of Astronomy and Astrophysics 30, 359 (1992)
[26] B. P. Schmidt, R. P. Kirshner, R. G. Eastman, M. M. Phillips, N. B. Suntzeff, M. Hamuy, J. Maza, R. Aviles, Astrophys. J. 432, 42 (1994)
[27] L. Breiman, Machine Learning 45, 5 (2001)
[28] A. Krizhevsky, I. Sutskever, G. E. Hinton, Advances in neural information processing systems (2012) pp.1097-1105