Application of a novel machine learning algorithm in gravitational wave transient noise identification

Ruaridh O'Donnell

E-mail: 1002205o@student.gla.c.uk

Abstract. Due to the low amplitude of gravitational waves, a gravitational wave signal can become masked by noise sources in the detector. A large portion of this noise is transient and of high amplitude. It can be confused with real gravitational wave signals. This project is an investigation into whether a recent machine learning algorithm (NuPIC) would be of use in classifying noise transients. A detailed description of how this algorithm functions is given. The algorithm was trained on a sine Gaussian model and tested to see how well it could differentiate the trained signal from background Gaussian noise. Up to an SNR of 5.1 it was able to differentiate the signal well, but for higher levels of noise it could not. Notes were made on how the algorithm deals with noise. These could be helpful in future work.

1. Overview

The aim of this work was to look at the possible use of a a novel unsupervised machine learning algorithm in the characterisation of gravitational wave detector noise. The algorithm, NuPIC is used commercially in detecting patterns in time series data. Gravitational wave detectors produce a lot of transient high amplitude noise in their output. This is often correlated with data that measures other aspects of the detector. It was thought that this algorithm could detect these patterns.

The algorithm was tested to see how well it could differentiate a known signal from noise. More work is needed to verify if the algorithm can work with real data.

2. Background On Gravitational Waves and Detectors

Advances in astronomy typically follow advances in detector technology. Telescopes tuned to wavelengths of light outside the visible spectrum have produced many discoveries beyond what was known only from visible light telescopes. Gravitational waves are predicted to be emitted by many astronomical bodies. If they can be detected then they provide a new method of discovery that is substantially different from electromagnetic radiation.

Detecting gravitational waves generates many technical challenges and thus the detectors are complex. This section will cover the relevant feature of the detectors for this project and also cover some of the ways in which machine learning has been used in this field in the past.

2.1. Gravitational wave detectors

The gravitational wave detectors in use today are fundamentally Michelson interferometers [1]. The mirrors at either end of the arms are mounted on test masses that move when gravitational waves pass. The output that measures the wave can be thought of as a time series of the power output of the interferometer. In reality modern advanced detectors (such as Advanced LIGO) are considerably more complex but the details do not need to be understood.

The waves detectable from earth have an extremely small effect on the test masses. The detectors need to have advanced noise removal so much of the work in building the detectors has gone into this. However despite the efforts much noise still remains in the signal. This noise can be broadly split into Gaussian noise and glitches. Where a glitch means is any high amplitude transient noise. The glitches can be mistaken for real gravitational wave signals.

2.2. Past attempts at using machine learning methods in gravitational wave searches

Machine learning methods are starting to be used in the field of gravitational wave searches. This section contains a summary of some recent efforts. It should be noted that the algorithms used here are substantially different in operation to NuPIC, the algorithm used in this project.

Inferring Core Collapse Supernova Physics with Gravitational Waves [2] There are different possibilities for how supernova evolve. Each produces a different pattern in the gravitational wave signal. However the waveforms vary enough in each class that that cross-correlation would be impractical. So this paper proposes a method for categorising a signal measured in LIGO into one of the three supernova types using principle component analysis and Bayesian statistics. Principle component analysis has some similarities to the way NuPIC operates.

Noise Artefact Removal Using Machine learning with Gravitational Waves [3] Glitches occur frequently enough that they show up in concurrence between two detectors. This causes problems when trying to detect gravitational waves across multiple detectors. Glitches often often occur in correlation with other readings taken from the detector. Hence automated machine learning methods can be used to distinguish them from real gravitational wave signals.

Data was split into two categories. The first; glitches that weren't gravitational waves, i.e. glitches that didn't have any coincidence in other detectors. This was composed of all glitches that were measured, with the assumption that this set contained no gravitational signals. The other was clean data composed of random samples from when the gravitational wave channel was quiet.

Similar performance was achieved using several different algorithms, suggesting that any improvements would be from including additional data in the classification.

Application of Artificial Neural Network to Search for Gravitational-Wave Signals Associated with Short Gamma-Ray Bursts [4] A fairly basic application of artificial neural networks to categorise glitches.

3. NuPIC: A Novel Machine Learning Algorithm

The Numenta Platform for Intelligent Computing (NuPIC) is the machine learning algorithm used in this project. It was developed privately by the company Numenta for use in analysis of streaming data. As of January 2014 it is in use in a commercial product "Grok" but the core algorithm has been open sourced [5].

This section will go over the main features of this algorithm, followed by a more detailed overview of it's inner workings. This will be illustrated with a simple example. Finally the specific implementation details used in this project will be covered as well as a short analysis of the noise characteristics of the algorithm.

3.1. Backround

Machine learning algorithms automatically learn patterns in collections of data. This automatic discovery of the underlying statistics in data make them useful across a wide class of domains. Most algorithms operate in a similar manner. They are trained on a collection of data until the patterns are discovered, and then they are used to evaluate new data (that was not in the training set). For example, a collection of vectors grouped together into classes can be used by a KNN classifier. A new vector can be evaluated by this algorithm to see which class it belongs to. There is a wide variety in how machine learning algorithms carry out this process. NuPIC was modelled on a theory of how the neocortex (a part of the mammalian brain) functions. This theory is called Hierarchical Temporal Memory (HTM) [6].

The neocortex is the part of the brain associated with higher intelligent thinking and long term memory. It consists of a sheet across the surface of the brain, approximately 2 mm thick, composed of around six layers. Different areas of the sheet carry out different functions (image processing, language, higher level thought, etc). The neocortex is has very regular structure of cells across these different areas. Vernon Mountcastle propositioned that despite the different functions, the neocortex is running the same algorithm across all areas. This is the basis of NuPIC. It is the very first part of an implementation of this algorithm.

3.2. Main features of NuPIC

Most machine learning algorithms have a training phase, where they process the data and learn its statistics. Once this phase is over the algorithm does not change. NuPIC operates in a slightly different manner that has more in common with biological brains. There is no distinct training phase, instead the data is fed in as a sequence and learning is continuous. For example a time series is fed in sample by sample, with learning happening after every sample.

NuPIC is also unsupervised. Supervised machine learning algorithms try to evaluate data against predefined labels. A common example would training on a collection of images of animals, each labeled with the type of animal. Then the algorithm would evaluate a new image of an animal and come up with its type. NuPIC instead categorises data into categories it chooses. This is how biological brains operate.

As well as learning on each sample, NuPIC forms predictions. This is a major part of the background HTM theory. They happen all the time in the brain and play a role in feedback, stability of representations, robustness to noise and expectedness of input. The import features of prediction here are its role in selecting context. When data is fed into NuPIC, it is classified, and a prediction of the next input is formed. This prediction is then used to help classify the next input. This process is detailed in the next section. Prediction allows one to say how unexpected the input is, or how anomalous it is. This anomaly detection was used in the glitch detection.

Internally NuPIC is a neural network. However it the details of this network are more complex than neural networks commonly used.

3.3. Algorithm details

This section borrows heavily from the description in the Numenta white paper which contains a more thorough explanation of the algorithms complete with pseudocode [7]. NuPIC undergoes three steps each time a sample of data is fed in. These are:

(i) Form a sparse distributed representation of the input
(ii) Form a representation of the input in the context of previous inputs
(iii) Form a prediction based on the current input in the context of previous inputs

3.3.1. Step 1: Form a sparse distributed representation of the input This step has two subsets. First the input (which in this case could be a sample form a time series, a real number) is converted to a binary vector. This is called encoding. It is not part of the core algorithm but it is necessary to convert the data type of the input into a format that can be used in subsequent steps. It follows that there are different encoders for different data types. For the case of real numbers as used in this project the "scalar encoder" was used. A value is converted to a binary vector with a section of on bits in a background of off bits. See fig **??** for details. The dimension of the vector and the number of on bits are parameters that do not change.

The next step is to convert this binary vector to a small set of active *columns* contained in a larger set of inactive ones. Consider the structure in fig **??**. The input binary vector from the previous step is input at the bottom. Then, in a process called spatial pooling, a set of columns become active (usually about 2% of the total number of columns). This set of columns is the sparse distributed representation of the input vector. SDRs have many useful properties which explain why this step is necessary. For example to differentiate each SDR in a set of SDRs it is not necessary to compared each bit. It is possible to check only a few and still get a low probability of mismatch. This reduces the computations that must be performed. Further, this property allows superpositions of two or more SDRs to be formed by or-ing their bits together. The chance of two random SDR overlapping significantly is extremely small. This property proves useful in step 3 (section 3.3.3).

Referring back to the structure diagram (fig **??**), each column has a number of synapses. connecting it to a large portion of the input space. Each synapse can be either active or inactive (on or off). An active synapse, permits the information to pass from the input to its column. In the spatial pooling process the following happens: For each column you add up the input bits connected to the column by active synapses. This results in an integer "score" for each column. Then, using this score, the top few columns are chosen to be made active. With the number chosen so that 2% of columns become active. This process is illustrated in fig **??**. Each column can be thought of as corresponding to a "pattern" in the input. The set of active columns represents the input in terms of these "patterns". Much like a vector can be decomposed into a combination of orthogonal basis vectors.

At this step learning occurs but it will not be covered here in detail. The learning process modifies the synapses. Each synapse has an associated real number named *permanence*. This is modified during learning and it is this number that decides wether the synapse should be active or inactive.

3.3.2. Step 2: Form a representation of the input in the context of previous inputs This step activates a set of *cells* within each of the columns activated in the last step. Each column is composed of a number of cells which can either be active or inactive. The active columns represent the input, whereas the active cells will represent the input in the context of previous inputs. This is important as an input can mean different things in different contexts. This is illustrated in the Numenta White Paper: In the spoken sequence of words "I ate a pear" and "I have eight pears" the words "ate" and "eight" sound exactly the same; they are the same input. Yet, given the context, it is clear they mean different things.

Cells are activated as follows: In every active column, each cell is activated if the cell was in a *predicting* state in the time step before. If a column is active but none of its cells are in a predicting state, then all the cells are made active. This process is shown in fig 1. Predicting cells are explained in the next step. When all the cells in a column become active this represents an input when the context is not known. Since each cell represents a different context, they are all activated.

3.3.3. Step 3: Form a prediction based on the current input in the context of previous inputs
The previous step referred to cells being in a *predicting* state. This step calculates which cells should be in the predicting state. Note that the result here will be used in the next time step of the algorithm.

Cells are chosen to become predicting in a manner similar to the way columns are chosen to be active in step 1. The details are not as important, but each cell has a collection of synapses (of the same type mentioned in step one) connecting it to other cells. A score for each cell is calculated by summing active cells that are connected by active synapses. If a score crosses a threshold then the cell becomes predicting. The result of steps 2 and 3 is that cell activations follow each other in learnt sequences. Synapses are formed from a cell to other cells that commonly become active on the time step before. So if two inputs commonly follow one another, A then B. Cells representing input B will form connections to cells representing input A. Hence when input A occurs, and the corresponding cells become active, the cells corresponding to input B become predicting.

This behaviour can be used to measure how unexpected an input is. The *anomaly score* is calculated equation 1. Where A_t is the set of active columns at time t ($|A_t|$ is the size of this set), and P_{t-1} is the set of columns with predicting cells at time step $t-1$.

$$\text{anomalyScore} = \frac{|A_t| - |P_{t-1} \cap A_t|}{|A_t|} \tag{1}$$

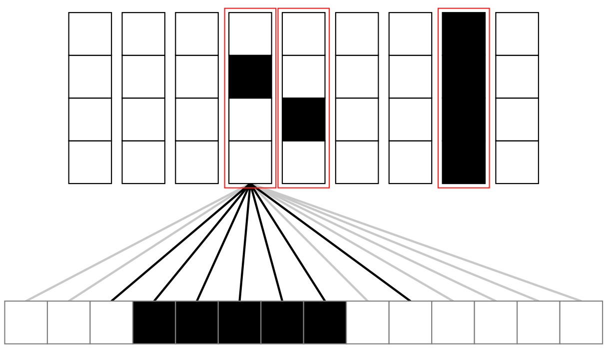

Figure 1. Step 2: Here the cells that were predicting and in an active column become active. Note that in the column with no predicting cells, all the cells became active.

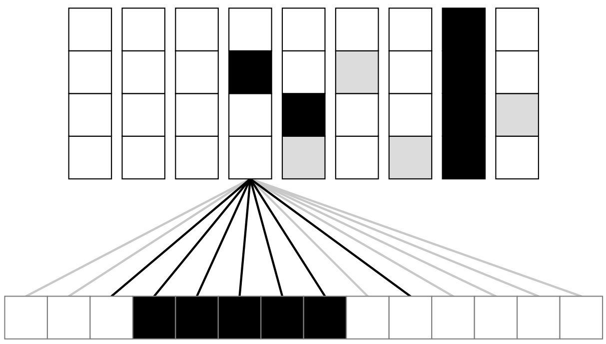

Figure 2. Step 3: This step calculates which cells to make predicting for the next time step. Note that none of the synapses for the cells have been shown.

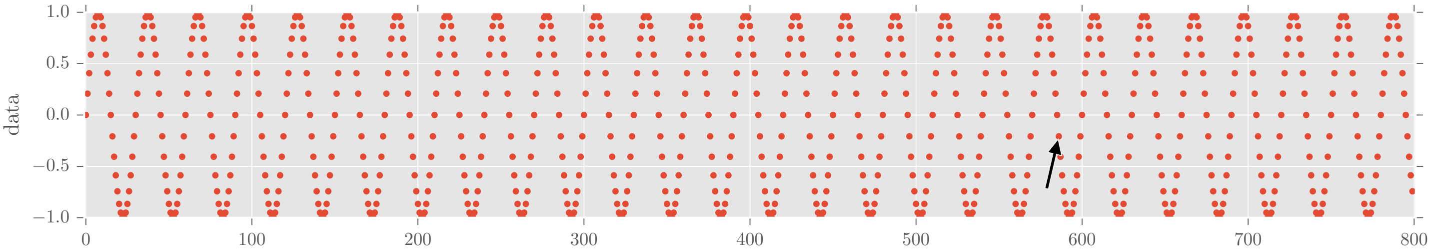

Figure 3. Data sampled from a sine wave. The arrow indicates sample 585.

3.3.4. The classifier The above three steps represent the core algorithm. However to form actual predictions of the next numerical input, a classifier is used. This classifier uses basic probabilistic methods to relate active cells to numerical input. As an aside, it is possible to generate a numerical prediction of the next input from the predicting cells. However Numenta has found that a classifier produces better results.

3.4. Case study: A sine wave

This section provides an example, showing the algorithm operating on some simple sine wave data. It will follow through the steps of the previous section showing the algorithm working at each step. The data that will be used was sampled from a sine wave at frequency of 30 samples per wave. It is shown in fig 3.

The algorithm starts in a newly initialised state with no patterns learnt. The data is fed in sample by sample with the algorithm going through all the steps of the previous section on each iteration. Each sample, the algorithm adjusts its synapses and gradually learns to predict the pattern. In this case, with a pattern that repeats exactly, the predictions become exact. What follows is the algorithm operating on the 585^{th} sample, which is after the data has been learnt. (indicated by the arrow in fig 3).

Step 1 Sample number 585 is -0.2079. This is first encoded into a binary vector. This vector is set to be 242 bits wide with 21 on bits. Since the range of the sine wave is $[-1, 1]$ the value -0.2079 corresponds to roughly the middle. This width of vector is typical in real use of the algorithm.

The next step is to activate the columns. There are 2048 columns, which is again typical. Of these 2048, 40 are chosen to be active. The active columns have the indices:

20	72	114	399	415	447	534	543	550	572
596	664	764	774	894	982	1003	1050	1079	1088
1144	1158	1194	1200	1223	1282	1304	1435	1448	1540
1593	1680	1777	1832	1856	1900	1965	1974	1981	2019

Step 2 Next, cells in each of the active column are chosen to be activated. In this case each column has 32 cells, which is the typical number. One cell in each column becomes active. Note that this value in the time series happens twice per wave. Once when the wave is going up, once when it is going down. Each will share the same set of active columns but have a different set of active cells. This is were the anomaly score is calculated. It is the number of columns that become active but contained no predicting cells. It is expressed as a percentage between 0 and 1. In this case it is 0.

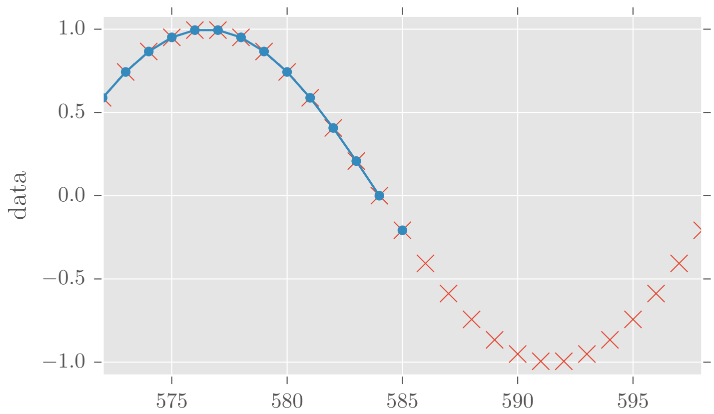

Figure 4. The plot shows the predicted value of the 585$^{\text{th}}$ data sample. The red marks are the data samples. The blue dots are the prediction at each step. The prediction for sample 585 is shown disconnected from previous predicted values.

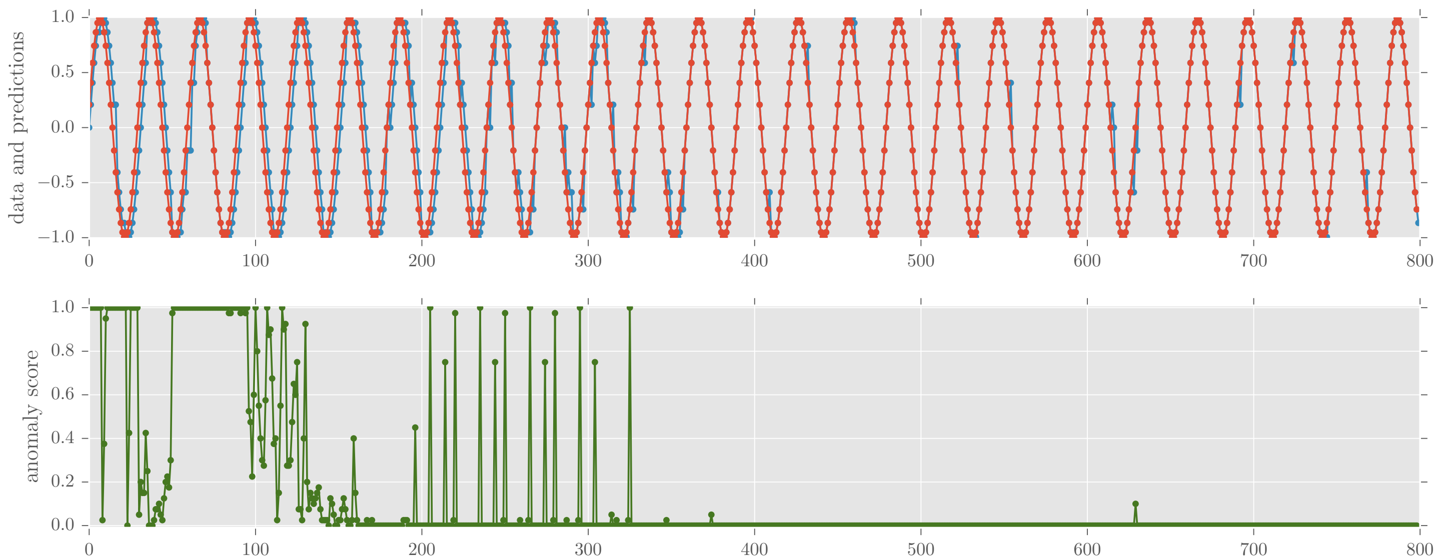

Figure 5. These plots show the algorithm running over all the data from the start. The top plot shows the prediction made at each sample with the data samples overlaid. Red is the data, blue is the predicted values. The plot below shows the anomaly score calculated at each step.

Step 3 Next the predicting cells are calculated. This produces 40 of them, which is the same number of active cells. With more complicated branching data there will usually be more predicting cells. These predicting cells will be used to activate the cells for the next data sample.

The classifier then outputs the prediction for the next value, based on what cells are active. This is -0.4067 and is shown in fig 4. This completes one step of NuPIC. Next the process would be repeated for the next sample, number 586.

Figure 5 shows the whole process starting from the newly initialised algorithm. As NuPIC learns to predict the pattern the anomaly score drops. Also note that initially the classifier predicts the last seen value.

3.5. Implementation details

The experiments were carried out using the following:

- nupic software version 0.1.3dev commit a485fa5
- 2048 columns
- 32 cells per column
- no swarming was used in setting the other parameters

3.6. Noise characteristics

Data from gravitational wave detectors has Gaussian noise that disrupts the signal. This noise can cause problems for NuPIC. Consider the case where the algorithm has been learning on some data and learning has stabilised. The algorithm will have learnt the patterns as best it can. If more of this data is fed in with added Gaussian noise, some points will be perturbed form their true position and will be different from what the algorithm expects them to be. This will cause the anomaly score to be high for those points. If there is enough noise then it will become impossible to tell if the algorithm recognises the data as the anomaly score will be continuously high. Since gravitational wave data is noisy it is useful to understand how NuPIC counteracts noise. Some brief investigations were done. It was found that there are two main mechanisms:

Input overlap between similar values If a sample from a time series is perturbed from its true value by a small amount (relative to the amplitude of the signal) then the binary vector representation of the perturbed sample will be close to the binary vector representation of the true sample (with the closeness measured by hamming distance). This closeness will result in a similar set of columns becoming active, which results in a similar set of cells becoming active. So the algorithms internal representation of the perturbed input will be close (by hamming distance) to the representation of the true input.

Multiple predictions If the algorithm learns in a noisy environment then it will learn that the signal can take many values, not just the true values. For each sample it will form multiple predictions about what the next input will be. This gives a higher chance of the next input being predicted, even if it perturbed form the true value.

4. Transient Identfication

This purpose of this section was to investigate wether NuPIC could identify a learned transient embedded in a background of noise. If NuPIC was trained on transients of interest and could recognise them, then it could pick them out of a long duration signal.

The process was as follows. The algorithm would be trained on a sample transient. After training the learning would be turned off. Then data would be fed in, containing the trained transients and other regions of noise. Error metrics describing how well NuPIC recognised the data could be extracted. These would be high for the noisy regions and low for trained transients that the algorithm recognises. Learning is disabling to stop the algorithm from learning any more patterns in the data. The error metrics used were an average error between the predicted signal and the actual data and the anomaly score detailed in equation 1.

4.1. description

The sample transient used in training (fig 6) was a a sine Gaussian with parameters listed in table 1. Data points close to 0 were trimmed from either end end of the sample (5 removed from the front, 6 from the end). Training was carried out by running NuPIC on the sample transient 200 times until the synapse permanences had stabilised. Learning was then disabled.

phase	5.6 rad
frequency	2 Hz
sample rate	$\frac{1}{32}$ Hz
duration	1 s

Table 1. Parameters defining the sine Gaussian used for training.

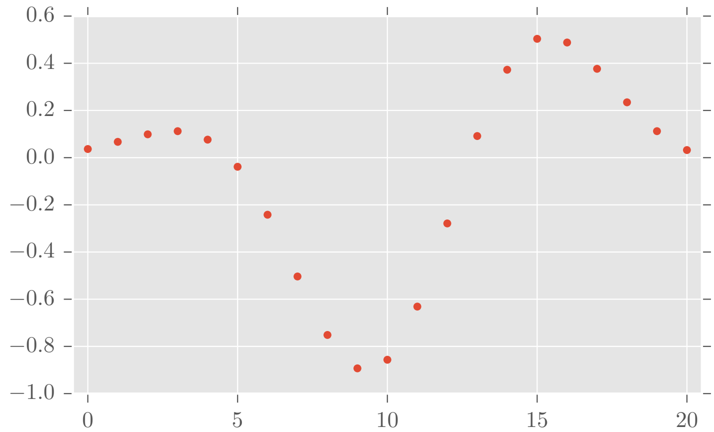

Figure 6. The training data.

Testing was carried out by adding Gaussian noise to a signal and calculating error metrics.The signals used were flat repeating 0 (of the same length as the training data (21 samples)) and the training data.

To calculate the errors metrics, the following process was repeated for each signal. Standard deviations of the noise to add to the signal were chosen to be 0.00, 0.05 ,0.10 ,0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50. For each level, Gaussian noise was generated and added to the signal. The signal plus noise was fed into the trained algorithm. An average error metric was calculated from the predictions generated. An average anomaly score was also calculated. This was repeated with another instance of noise (of the same standard deviation) 100 times and the average errors and average anomaly scores were averaged. This resulted in one average error and one average anomaly score (with error bars for these values) for one level of the added noise. This process was repeated for each level of noise listed above and the results plotted (fig 7 and fig 8).

The average error was calculated as the average of the absolute value of the difference between the predicted value and the actual data. The anomaly score was calculated as detailed in equation 1.

4.2. Results

The results of the testing process detailed in the last section are shown in figures 7 and 8. The anomaly score was the most effective in distinguishing the two signals. It was able to distinguish the signals up to an added noise of standard deviation 0.15. An anomaly score of 1 corresponds to an input that was entirely unpredicted. A score of 0 corresponds to an input that was completely predicted. As the noise added to the training data increases the average anomaly increases as more of the individual data points are unpredicted. The flat signal has a score of 1 initially as a sequence of 0s do not appear anywhere in the training data. The score drops initially as the noise increases as there is a higher likelihood of sequences that occurred in the

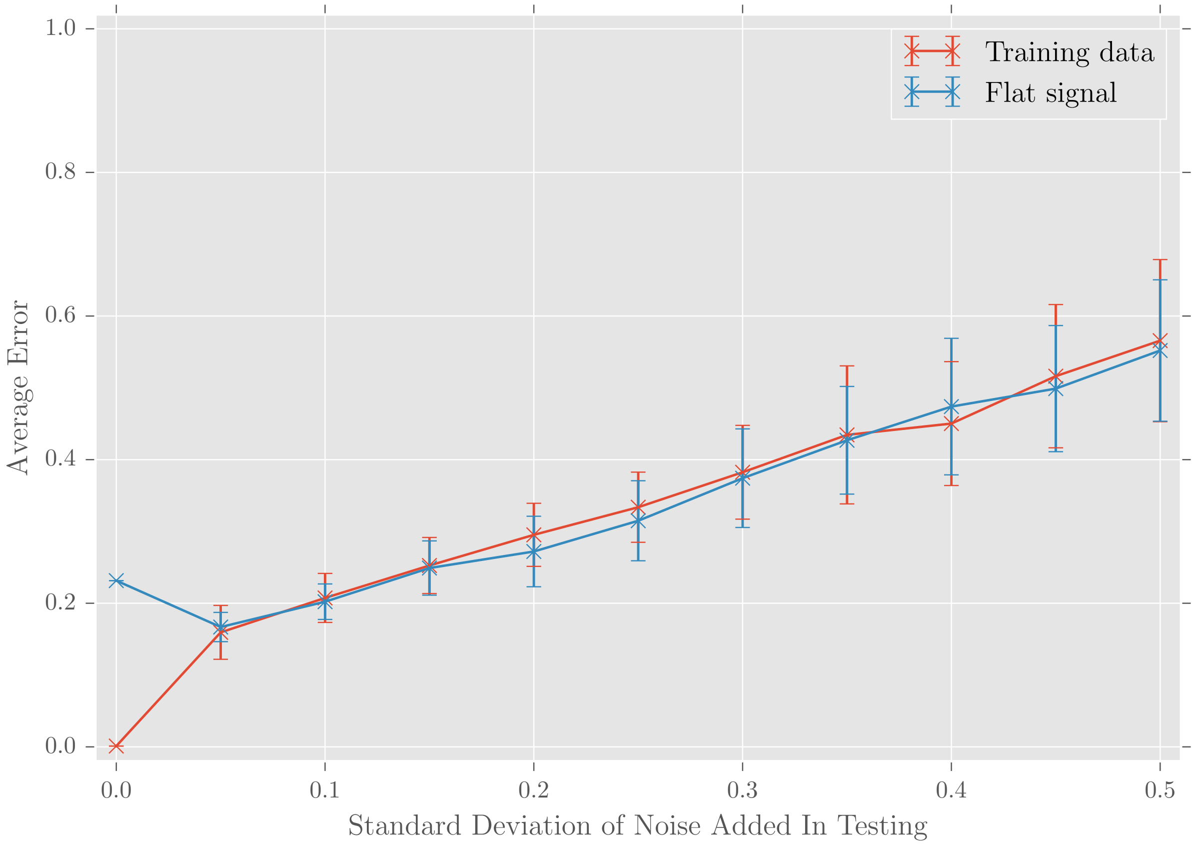

Figure 7. The average anomaly score calculated when testing on two different signals (flat signal and the training data) for different levels of added noise.

training data occurring.

4.3. Issues

One issue found was the tendency for the algorithm to generalise well. Initially the algorithm was trained on the sample transient without the trimming mentioned in section 4.1. This version had a sequence of points close to 0 at either end. The result was that when the algorithm was tested, it was able to predict the flat signal well. This led to it being unable to differentiate the flat signal from the training data. Since the algorithm had seen sequences of 0 in training, it was able to predict them when they occurred in testing.

This was verified by training the algorithm on the untrimmed sample transient and then looking at the sets of active cells. The set of cells that became active during the 0 region of the sample transient matched completely with the set of cells that became active during testing on the flat signal. This shows that the algorithm's internal representation of the 0 region was the same as its internal representation of the flat signal. Hence the 0 regions were causing the unexpectedly high accuracy in the predictions.

4.4. Discussion

The results shown show how well the algorithm identifies a repeating 0 signal. A more complete analysis would investigate how well the algorithm differentiates the trained data from a wide variety of signals. This would give a much clearer picture of how the algorithm would behave

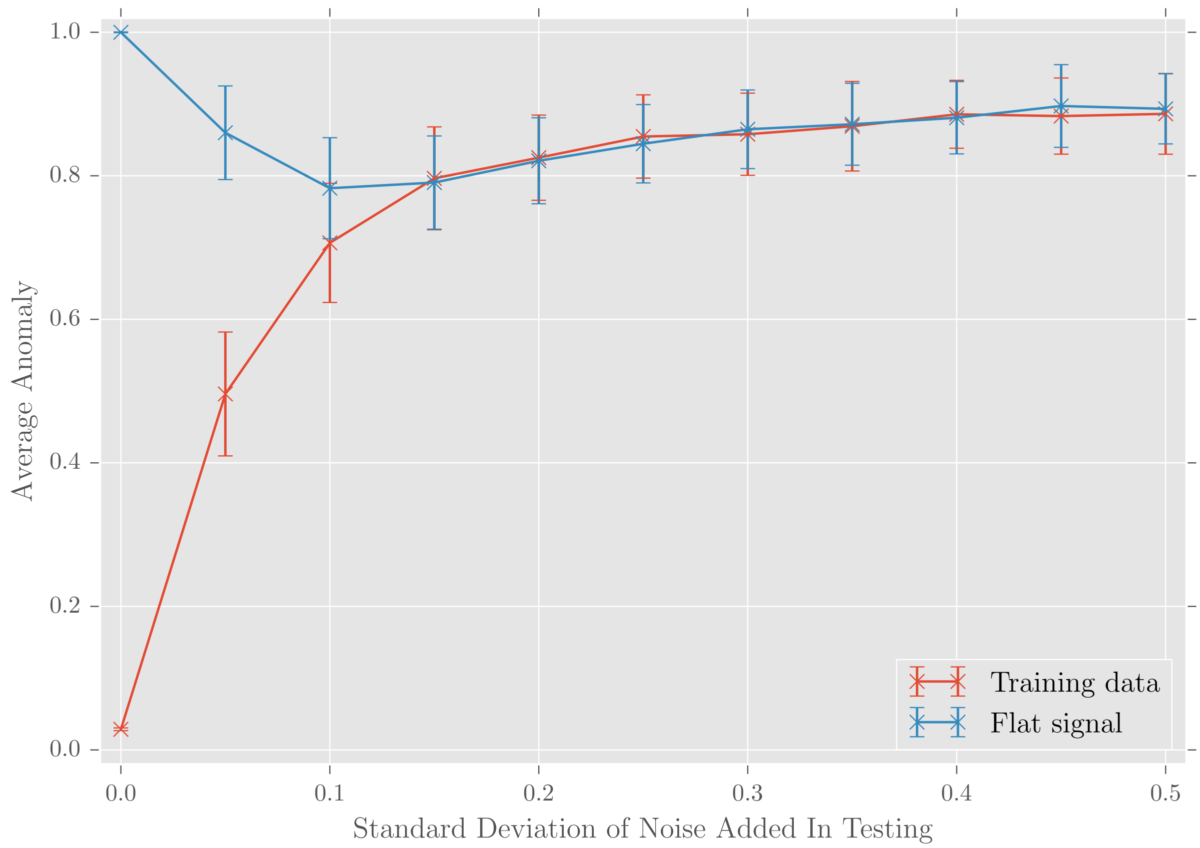

Figure 8. The average error calculated when testing on two different signals (flat signal and the training data) for different levels of added noise.

with real data. Further it would allow the performance to be compared to techniques such as matched filtering for identifying signals.

5. Additional Material: Using Gravitational Wave Detector Data

Given that the algorithm is commercially used in unsupervised learning of real world data, it was of interest to see how it performed on real data taken from the LIGO observatory. The algorithm was run on some channel data however picking channels to obtain data from and then obtaining this data proved to be difficult.

6. Conclusion

Some Text [1]

References

[1] Saulson P R 1994 *Fundamentals of interferometric gravitational wave detectors* (Singapore ; River Edge, N.J: World Scientific) ISBN 9810218206

[2] Logue J, Ott C D, Heng I S, Kalmus P and Scargill J 2012 *arXiv:1202.3256 [astro-ph, physics:gr-qc]* arXiv: 1202.3256 URL `http://arxiv.org/abs/1202.3256`

[3] Biswas R, Blackburn L, Cao J, Essick R, Hodge K A, Katsavounidis E, Kim K, Kim Y M, Bigot E O L, Lee C H, Oh J J, Oh S H, Son E J, Vaulin R, Wang X and Ye T 2013 *Physical Review D* **88** ISSN 1550-7998, 1550-2368 arXiv: 1303.6984 URL `http://arxiv.org/abs/1303.6984`

[4] Kim K, Harry I W, Hodge K A, Kim Y M, Lee C H, Lee H K, Oh J J, Oh S H and Son E J 2014 *arXiv:1410.6878 [astro-ph, physics:gr-qc]* arXiv: 1410.6878 URL `http://arxiv.org/abs/1410.6878`

[5] The Numenta Commnity 2015 Nupic code repository URL `https://github.com/numenta/nupic`

[6] Hawkins J and Blakeslee S 2005 *On intelligence* (New York: Henry Holt and Co.) ISBN 0805078533 9780805078534

[7] Numenta, Inc 2011 HTM whitepaper URL `http://numenta.com/learn/hierarchical-temporal-memory-white-paper.html`