Dynamics at the Horsetooth Volume 1, 2009.

Analysis of Electroencephologram Data Using Time-Delay Embeddings to Reconstruct Phase Space

Nicholas Rohrbacker
Department of Mathematics
Colorado State University rohrback@math.colostate.edu

Report submitted to Prof. P. Shipman for Math 540, Fall 2009

Abstract. In this paper, the necessary embedding dimension for electroencephologram(EEG) data is calculated. The tools used include a 1981 theorem from Floris Takens, and a subsequent extension of that theorem in 1991 given by Casdagli, Sauer and Yorke. These theorems give a method of reconstructing the phase space of the original system up to diffeomorphism using time-delays. If d is the original dimension of the phase space, the theorems require that at least $2d + 1$ time-delays are used in order to guarantee reconstruction of the phase space. This embedding dimension is calculated using false nearest neighbors.

Keywords: Takens, Embedology, Time Delay, Embedding, False Nearest Neighbors.

Contents

1 Introduction

Analysis of observed data requires a tool set in order to obtain useful results from the observations. In general, the experimenter cannot hope to obtain noise-free data, nor can they possibly obtain the sometimes infinite number of observations needed to perfectly duplicate the system or object from which the data originated. Pattern analysis tools generally seek to identify structure, or information, from data that is either not apparent from initial analyses of the data or lie in dimensions well beyond visual comprehension.

In this paper, a 1981 theorem of Floris Takens regarding time-delay embeddings, and its subsequent extension in 1991 by Casdagli, Sauer, and Yorke is used to perform some initial analysis on electroencephologram(EEG) data. EEG data is obtained through electrodes placed on the scalp of an individual that measure the electrical impulses generated by the neurons in a neighborhood of the electrode. The readings are generally taken over time, and the collected data represents a time series of these readings. The afore mentioned theorems provide a way to embed the observed data from these electrodes into a diffeomorphic copy of their original phase space. Once the embedding is complete, then certain characteristics of the system that generated the data can be calculated.

In the second section, the theorems will be stated and explored. The terms from the theorems will be defined, and the connection between these theorems and observed data will be made. Lastly, a simple example to demonstrate how the theorems work will be given. The third section gives a few more details about the specific EEG data used for this project. The fourth section will discuss how the embedding dimension for observed data is calculated. Then the embedding dimension for the EEG data will be found using increasing numbers of data points. Finally, a comparison between the embedding dimension results for unaltered data, and data that has been filtered using a moving average will be performed. The fifth section gives the MATLAB code for the algorithm used.

2 Takens Theorem and Embedology

Takens' Theorem (1981) in its most general form is stated as follows.

Theorem 1 ***(Takens 1981)*** *Let M be a compact manifold of dimension m. For pairs (ϕ, y), with $\phi \in Diff^2(M)$, $y \in C^2(M, \mathbf{R})$, it is a generic property that the map $\Phi_{(\phi,y)} : M \to \mathbf{R}^{2m+1}$, defined by*

$$\Phi_{(\phi,y)}(x) = (y(x), y(\phi(x)), \ldots, y(\phi^{2m}(x)))$$

is an embedding.

In this setting, y is referred to as a measurement function, and corresponds to the function that takes the value of the dynamical system at time t, to an observable value in $\mathbf{R}$. Generic is defined as open and dense. The proof of this theorem provided by Takens in 1981 is restrictively steeped in Differential Topology [1]. However, a 2006 paper by J.P. Huke provides an in-depth, constructive proof of the theorem which along the way takes great care to illuminate the importance of all assumptions made in the statement of the theorem [2].

On the surface, Takens' theorem provides a way to gather information about a dynamical system's phase space given that the mapping $t \to y(\phi_t(x))$ is known, or in most experimental cases, observed. The power of Takens' theorem is in its simplicity, namely, that taking consecutive observations in sufficient number will guarantee, in a generic sense, that one has reconstructed, up to diffeomorphism, the original dynamics of the system. This reconstruction is, of course, limited by the number and quality of the observations to be used.

The work of Sauer, Yorke, and Casdagli in their 1991 paper "Embedology" extends the notion of generic to one that is at the same time more familiar and useful [3]. This is in reaction to the fact that there are many generic sets that can have arbitrarily small Lebesgue measure. Another problem is that the space of smooth maps is infinite-dimensional and does not have an obvious generalization of probability one from finite-dimensional spaces. To begin to solve this problem, they define the notion of prevalence as follows.

Definition 1 *A Borel subset S of a normed linear space V is* ***prevalent*** *if there is a finite-dimensional subspace E of V such that for each $v \in V$, $v + e$ belongs to S for (Lebesgue) almost every $e \in E$.*

With this definition in place, it is possible to extend the works of both Whitney and Takens to guarantee an embedding with probability one. As examples, consider the following theorems. The first extends the well known theorem of Whitney that states that the set of functions that embed a compact smooth manifold A of dimension d into $\mathbf{R}^{2d+1}$ is an open and dense set in the C^1-topology of maps [3]. The next two theorems are extensions of Takens' theorem to the notion of prevalence, and are provided by Casdagli, Sauer and Yorke [3].

Theorem 2 ***(Whitney Embedding Prevalence Theorem).*** *Let A be a compact smooth manifold of dimension d contained in $\mathbf{R}^k$. Almost every smooth map $\mathbf{R}^k \to \mathbf{R}^{2d+1}$ is an embedding of A.*

Definition 2 *If g is a diffeomorphism of an open subset U of $\mathbf{R}^k$ and $h : U \to \mathbf{R}$ is a function, define the* ***delay coordinate map*** *$F(h, g) : U \to \mathbf{R}^n$ by*

$$F(h, g)x = (h(x), h(g(x)), \ldots, h(g^{n-1}(x)))$$

The next theorem will require a brief introduction to the definition of the box-counting dimension of a compact set. The box-counting dimension of a compact set A in $(R)^k$ is defined to be

$$boxdim(A) = lim_{\epsilon\to 0}\frac{log(N(\epsilon))}{-log(\epsilon)},$$

where $N(\epsilon)$ is the number of cubes of side length epsilon necessary to cover A. If this limit does not exist, one must turn to the upper or lower box-counting dimension by taking the lim sup, or lim inf respectively.

Theorem 3 *Let g be a diffeomorphism on an open subset U of $\mathbf{R}^k$, and let A be a compact subset of U with boxdim(A)= d, and let $n > 2d$ be an integer. Assume that for every positive integer $p \leq n$, the set A_p of periodic points of period p satisfies boxdim(A_p)< $\frac{p}{2}$, and that the linearization Dg^p for each of these orbits has distinct eigenvalues. Then for almost every smooth function h on U, the delay coordinate map $F(h, g) : U \to \mathbf{R}^n$ is one-to-one on A, and an immersion on each compact subset C of a smooth manifold contained in A.*

However, in most instances, the time series, or data that is used in the theorems presented are found through experimental observations, and inherantly, will contain some noise. If this noise is significant enough, then the reconstructed state space could be significantly different from the original state space. To account for this, the following theorem was proven.

Theorem 4 ***(Filtered Delay Embedding Prevalence Theorem)*** *Let U be an open subset of $\mathbf{R}^k$, g be a smooth diffeomorphism on U, and let A be a compact subset of U, boxdim(A_p) $= d$. For a positive integer $n > 2d$, let B be an $n \times w$ matrix of rank n. Assume g has no periodic points of period less than or equal to w. Then for almost every smooth function h, the delay coordinate map $F(B, h, g)x = BF(h, g)x$ is one-to-one on A and an immersion on each closed subset C of a smooth manifold contained in A.*

In many instances, B represents a matrix that will invoke a moving average of the coordinates, in order to smooth out any noise between several points. It can, however, take on more complicated forms. In one application, the right singular vectors of the reconstructed data points are used in order to more accurately determine the original dimension of the state space.

When these techniques are used for analyzing observed data, the function h is generally defined to be the actual observed variable, so if it is assumed that the data comes from a dynamical system $\mathbf{x}(n) = F(\mathbf{x}(n-1))$, then $h(\mathbf{x}(n)) = s(n)$ where $s(n)$ is the observed value of the signal at time $t = n$. For the diffeomorphism, g, the function is chosen that takes a vector $\mathbf{x}$, to the vector one time-delay, T, later. This time-delay T varies from system to system depending on the evolution rule over time. As an example, choosing T to be one microsecond when sampling temperature variation in a region of a state would be oversampling. The evolution rule would not have a chance to play itself out in this short amount of time. There are prescriptions for choosing this time-delay in a data-driven manner using a technique called average mutual information [5]. After determining an optimal time-delay, an arbitrary component in the constructed time-delay vector of a time series starting at the vector $\mathbf{x}$ is given by $h(g^k(\mathbf{x}(n))) = h(\mathbf{x}(n + k * T))$. In terms of the observed data, one would have the same component as $s(n + k * T)$, where $s(n)$ corresponds to the observation recorded for the vector $\mathbf{x}(n)$

As a simple example of the technique, consider a cosine wave between $-\pi$ and π as seen projected onto the $cos(t)$-axis, or in general the $s(t)$-axis where $s(t) = cos(t)$. In this representation, the graph flows between 1 and -1 and there are many points whose derivatives are pointing in opposite directions, but are in very close proximity, entirely due to the fact that they are represented in one dimension. If the data points from the $s(t) = cos(t)$ curve are taken every tenth of a second, one can form a time series that is of the form $(cos(0), cos(.1), cos(.2), ..., cos(k * .1), ..., cos(n * .1))$. If a one-step time delay is used, then one can form the set of 2-tuples $[(cos(-\pi), cos(-\pi+.1)), (cos(-\pi+.1), cos(-\pi + .2)), ..., (cos(-\pi + (n-1) * .1), cos(-\pi + n * .1))]$,where $n < \frac{2\pi}{.1}$. The result is an ellipse as is seen in figure 1. This ellipse, while not identical to the classical cosine curve in two dimensions, is however, diffeomorphic to said curve.

3 EEG Data

The data for this project was acquired through the website of professor Charles Anderson from the department of computer science at Colorado State University [4]. The data was created and collected in order to support the construction of Brain-Computer Interfaces (BCIs). BCIs are hardware and software systems that analyze the signals collected through multiple electrodes placed on the scalp of an individual. The purpose of the BCIs is to correctly classify these signals by mental process, thereby making communication with individuals who have lost voluntary muscle control possible.

The time series of EEG data used in this project represents data from one individual, performing one mental task. The data is sampled at 250 Hz for 10 seconds for a total of 2500 data points. The task performed is the baseline task of rest, which means the subject is not performing any of the mental tasks used as comparisons. These tasks can include visualization of three dimensional objects, multiplying two numbers together, picturing someone writing on a chalk board and many

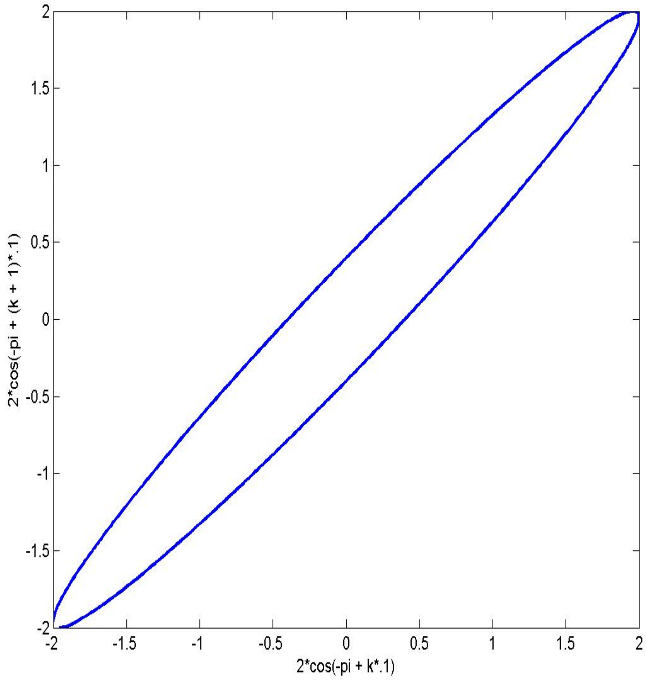

Figure 1: $S(t) = 2 * cos(t)$ as seen embedded in two dimensions using $((2 * cos(-\pi + k * .1), 2 * cos(-\pi + (k + 1) * .1)$

other mental exercises. Since reconstruction of phase space and calculation of certain qualitative features of said phase space is the goal of this paper, and not classification, this will be an adequate data source.

4 Reconstruction of Phase Space

Given a time series of observed data, in general, the dynamics of the original system are unknown. Therefore, the dimension d of the manifold from which the system originated is unknown, and hence so is $2d + 1$. This requires that the dimension be calculated from the data. This can be accomplished using global false nearest neighbors [5]. Global false nearest neighbors effectively determines whether two nearby points in a lower dimension are close due to the original dynamics of the system in its original higher dimensional space, or if they have been placed close to one another by repeated projections to lower dimensions.

The general algorithm for finding false nearest neighbors is as follows:

1. Construct d-dimensional vectors from the observed data using a delay embedding.
2. For each vector, find its nearest neighbor in d-dimensional space using Euclidean distance.
3. Compare the distance between the vectors in d-dimensional space to the distance between the vectors when embedded in dimension $d + 1$.
4. If the absolute value of this difference is above a given threshhold, then the two vectors are assumed to be false nearest neighbors.

If $s(0), s(1), s(2), \ldots, s(n)$ is our observed set of points, then let

$$y(k) = [s(k), s(k+1), \ldots, s(k+(d-1))]$$

be the d-dimensional vector constructed using a time delay starting at s(k). It is clear that the incremental distance between the vector $y(k)$ and its nearest neighbor in d dimensional space, $y_n(k)$, when moved to dimension $d+1$ is represented by the quantity $|s(k+d) - s_n(k+d)|$. If the distance between the vectors in d-dimensional space is represented by D_d, then the relative change in distance can be represented by the following formula [5].

$$\frac{|s(k+d) - s_n(k+d)|}{D_d}$$

The value at which the above formula is deemed to yield a false nearest neighbor is somewhat arbitrarily chosen at .75. This means that the relative difference in the final component when the points are moved one dimension higher, must represent 75 percent of the original distance between the two points.

Since EEG data is inherently noisy, an attempt at filtering the data was made to see if it made an impact on the embedding dimension. To filter the data, a simple moving average filter was applied to the original data. As can be seen in the figures below, this did not make a large difference in the calculated embedding dimension. However, it did smooth the graph as it approached zero percent false nearest neighbors.

In the following figures, the percent of false nearest neighbors is plotted versus the embedding dimension using increasing numbers of observations. In all cases, once the embedding dimension hits approximately 22, the percent of false nearest neighbors ceases to improve. Therefore, one can conclude that the data has been unfolded sufficiently in 22-dimensional space, and no further time-delays are necessary.

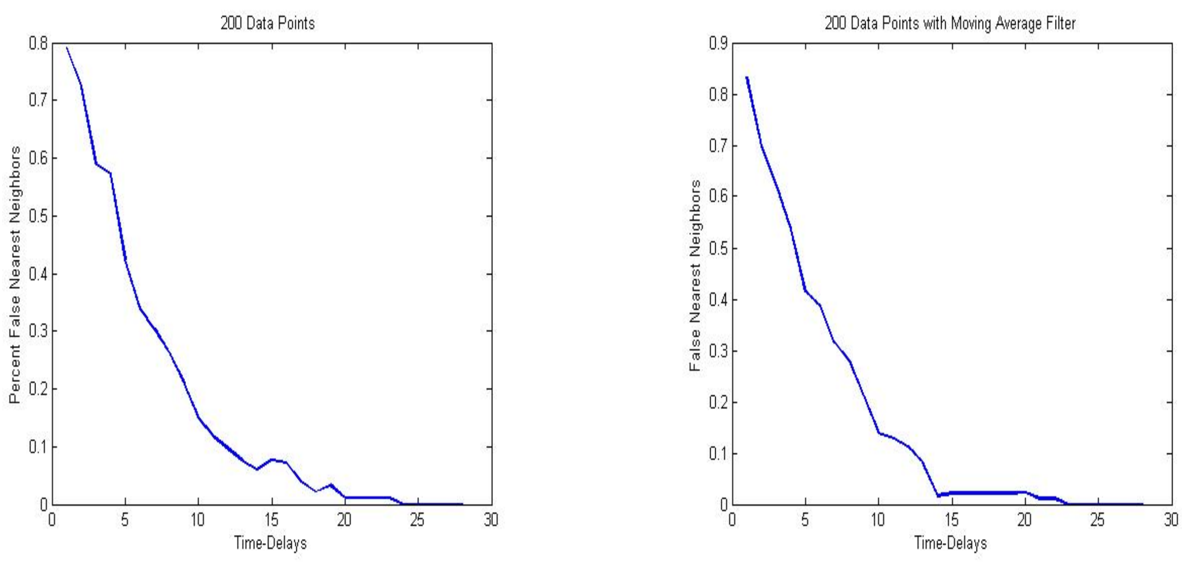

Figure 2: Percentage of False Nearest Neighbors Using 200 Observations Plotted vs Embedding Dimension. Non-filtered data is used in the first graph, filtered in the second.

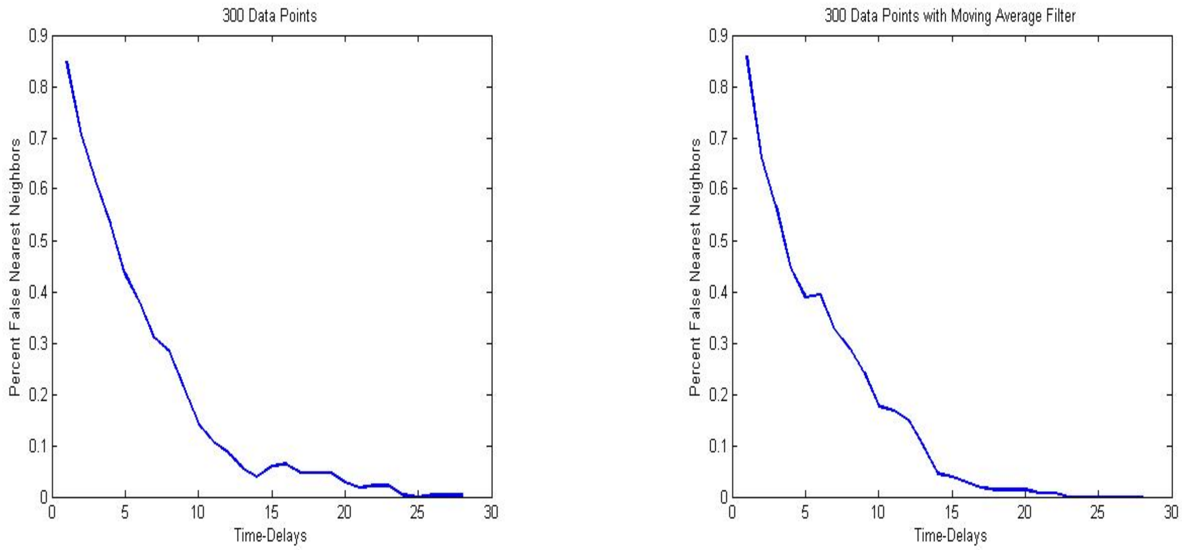

Figure 3: Percentage of False Nearest Neighbors Using 300 Observations Plotted vs Embedding Dimension. Non-filtered data is used in the first graph, filtered in the second.

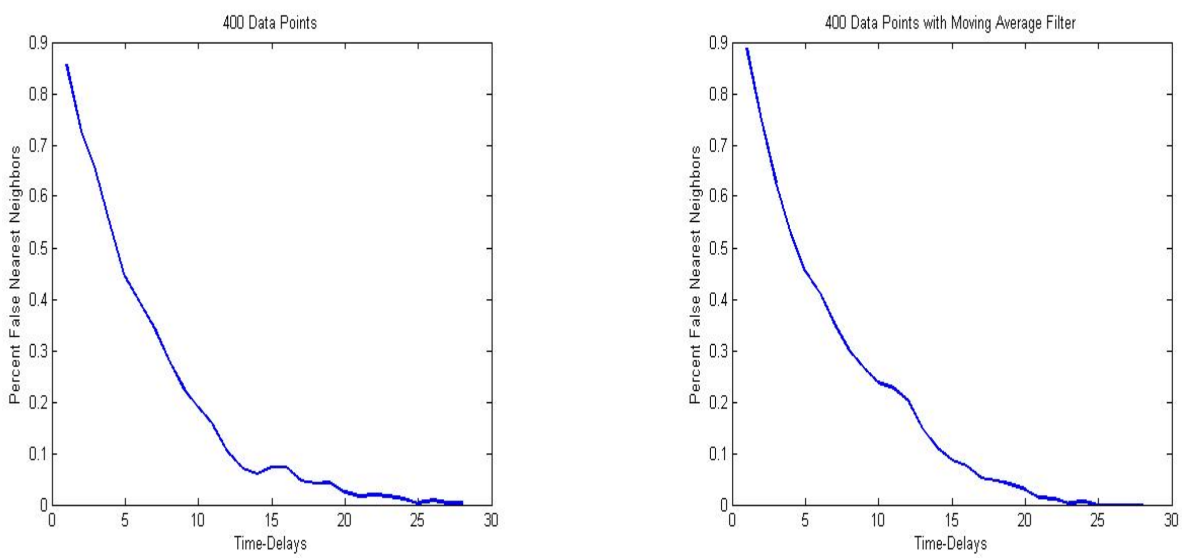

Figure 4: Percentage of False Nearest Neighbors Using 400 Observations Plotted vs Embedding Dimension. Non-filtered data is used in the first graph, filtered in the second.

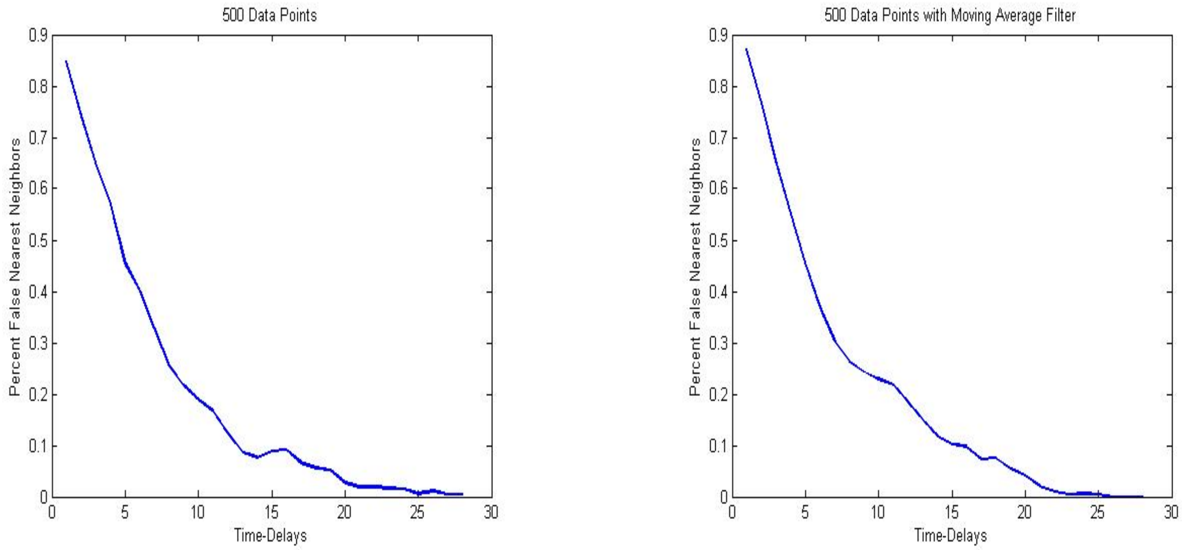

Figure 5: Percentage of False Nearest Neighbors Using 500 Observations Plotted vs Embedding Dimension. Non-filtered data is used in the first graph, filtered in the second.

5 Conclusions

This paper has explored the use of time-delay embeddings in the analysis of observed data. In particular, the application of the theorems of Takens and Casdagli, Sauer and Yorke was demonstrated on EEG data. Also, the technique of false nearest neighbors was used to calculate the embedding dimension required by the afore mentioned theorems.

While EEG data is generally assumed to be quite noisy, the embedding dimension calculated in this paper would indicate otherwise. In data with significant amounts of noise distributed throughout the data, it is common to see extremely high embedding dimensions, or to see the percent of false nearest neighbors never reach zero percent [5]. The calculations performed in this paper would suggest that the data is not as noisy as once thought, or that the noise to signal ratio is small. If this is in fact the case, then further investigation into the true signal to noise ratio of the EEG data is merited. However, it is possible that the structure of the noise in the data has simply caused the algorithm to increase the embedding dimension, but still reach a zero percent nearest neighbor measure. In this case, next steps would include methods of separating the noise from the true signal, and re-calculating the embedding dimension to see if it drops significantly.

6 Code

```
% Read in data from file and then take only task1 from one node.
% The resulting array is called dat.
tic
cd('C:\Users\Nicholas Rohrbacker\Documents\Nick Work');
load eegdata;
eegdataload;
numdat = 500;
udim = 60;
ma = 9;
```

```
numdat2 = numdat + (ma - 1)/2;
dat = task1(1,1:numdat);
dat = dat - mean(dat);
dat2 = task1(1,1:numdat2);
dat2 = dat2 - mean(dat2);
dat3 = zeros(1,numdat);
j = 1;
for m = (((ma - 1)/2) + 1):(numdat2 - ((ma - 1)/2) - 1)
    dat3(1,j) = (sum(dat2(1,(m-(ma - 1)/2):m + (ma -1)/2)))/9;
    j = j + 1;
    j
end



%Since the data must lie in at least dimension
%2d + 1, the minimum embedding dimension is 3.  A holds the vectors in i
%space for the unfiltered data, A1 holds the filtered data.
d1 = zeros(1,udim);
 for i = 3:1:udim
      A = zeros(i,numdat - i);
      A1= zeros(i,numdat - i);
      d = zeros(2,numdat - i);
      d1 = zeros(2,numdat - i);




    %t loops through each value in the
    %dat array that has at least i values beyond the t
    %position
     for t = 1:numdat - i
           A(:,t) = dat(1,t:t+i - 1);
           A1(:,t) = dat3(1,t:t + i - 1);
     end

     %Calculate pairwise distances to find the nearest neighbor of each
     %point in i space.
     %This set of code finds the nearest neighbor for each point.
     h_tot = zeros(numdat - i,numdat - i);
     h_tot1 = zeros(numdat - i,numdat - i);
     for ii = 1:numdat - i
        h_dist = sqrt(sum((A - repmat(A(:,ii),1,numdat - i)).^2));
        h_tot(:,ii) = h_dist';
        h_dist1 = sqrt(sum((A - repmat(A(:,ii),1,numdat - i)).^2));
        h_tot1(:,ii) = h_dist';
     end

     h_tot = h_tot + diag(100000*ones(numdat - i,1),0);
```

```
    h_tot1 = h_tot1 + diag(100000*ones(numdat - i,1),0);

    for ii = 1:numdat - i;
        dist = h_tot(ii,ii);
        dist1 = h_tot1(ii,ii);
        for jj = 1:numdat - i;
            if h_tot(ii,jj) < dist
                dist = h_tot(ii,jj);
                d(1,ii) = dist;
                d(2,ii) = jj;
            end
            if h_tot1(ii,jj) < dist1
                dist1 = h_tot1(ii,jj);
                d1(1,ii) = dist1;
                d1(2,ii) = jj;
            end
        end
    end




    %Now check the relationship of the nearest
    %neighbor in i space to the
    %the distance when the points are moved to i + 1 space.
    numfalse = 0;
    numfalse1 = 0;
    for k = 1:numdat - i
        test = abs(dat(1,k + i) - dat(1,d(2,k) + i))/(d(1,k));
        test1 = abs(dat3(1,k + i) - dat3(1,d1(2,k) + i))/(d1(1,k));
        if test > .75
            numfalse = numfalse + 1;
        end
        if test1 > .75
            numfalse1 = numfalse1 + 1;
        end
    end

    percentfalse = numfalse/(numdat - i)
    percentfalse1 = numfalse1/(numdat - i)
    numdat - i
    i
    d2(1,i) = percentfalse;
    d3(1,i) = percentfalse1;



end
plot(d2(1,3:udim),'LineWidth',2,'MarkerSize',10)
```

```
figure;
plot(d3(1,3:udim),'LineWidth',2,'MarkerSize',10)
toc


t = -pi:.01:pi;
y = 2*cos(t);

for i = 1:609
    y2(1,i) = y(1,i);
    y2(2,i) = y(1,i + 20);
end
plot(y2(1,:),y2(2,:),'LineWidth',2,'MarkerSize',10)
```

References

[1] Takens, F. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics* 898, 366-, 1981.

[2] Huke, J.P. Embedding Non-Linear Dynamical Systems: A Guide to Takens Theorem. *Manchester Institute of Mathematical Sciences* 26, , 2006.

[3] Sauer T., Yorke J., Casdagli M. Embedology. *Journal of Statistical Physics* 65,579-616 , 1991.

[4] Website of Charles Anderson, Professor of Computer Science, Colorado State University http://www.cs.colostate.edu/eeg/eegSoftware.html, accessed 11/01/09

[5] Abarbanel H. *Analysis of Observed Chaotic Data* Springer-Verlag, 1996.