

# EEG Group

Research Paper

By:

Anthony Hampton

Tony Nuth

Miral Patel

## **Tables of Contents**

Introduction.....	Page 3
Methodology.....	Page 6
Analysis.....	Page 9
Appendix.....	Page 11
References.....	Page 14
Code .....	Page 15

## Introduction

EEG Lab was created in 1997. It is a set of data processing functions and was first released on the Internet by Scott Makeig. In 2000, Arnaud Delorme designed a graphic interface on the top of these functions along with some of his own artifact removal functions, and released the first version of the EEGLAB. In 2003, Delorme and Makeig joined effort to release the first stable and fully documented version of EEGLAB. In 2004, EEGLAB was awarded funding by the NIH for continued development of research software.[6]

EEG stands for electroencephalogram. An EEG is an amplified recording of the electrical waves sweeping across the brain's surface. The ways that these are collected are by 40 separate electrodes that are placed on the skull. An EEG is basically the sum of electrical activity cause by neurons in the human brain that dictate the actions of the body. An EEG is composed of 40 epochs, an epoch for every electrode attached. Each epoch is an electrical value at minimum 101 time points sometimes more.

There are three types of brainwave activity based upon that status of the person it's being collected from. The types are Tonic Spontaneous, and Phasic. Tonic is the baseline for brain activity. From the tonic pattern it is easy to see changes in state. Spontaneous is an uncontrolled action such as the heart beating

or breathing in and out. They are irrelevant and task-related events in the brain.

Phasic is the response to stimulation. phasic is a reaction to the environment.

There are also 4 types of EEG's that can be gathered from a person. These four are Beta, Alpha, Theta, and Delta. Beta is a highly alert status, Alpha is a relaxed state, Theta is from when the person is drowsy, and finally a Delta pattern can be gathered when the subject is in deep sleep. For our project were looking at the Beta pattern.

The data was collected from 19 different volunteers collected by Thomas Young. The subjects were placed in a room that minimized electrical interference and asked to push a button on their left or right when instructed to. The subjects were told clearly what button was to be press before they had to push it. The main problem with EEG's which is why we need such analysis is because the noise and interference are great. The noise disrupts the data and we need to use mathematical processes to draw adequate conclusions from the data.

This project is very interesting and relevant. The research being done by our group is furthering the knowledge of neuroscience and understanding of the human brain. This project further aids understanding of the human brain.

# Methodology

While evaluating the EEG data we used a number of mathematical techniques that can give us information that we can use to better understand the EEG data. The techniques used are Time Series Analysis, Expectation Maximization clustering, and wavelets. All of these mathematical techniques have their own specifications and give back data into usable and understandable format. The definitions and purposes of each of the functions used in the projected will be described in further detail in the next few paragraphs.

We first will discuss using Time Series Analysis and described the purpose, history and thought process behind the idea of using the technique. A time series is a collection of observations of well defined data items obtained through measurements over time. “During the last 25 years Time Series Analysis has become one of the most important and widely used branches of Mathematical Statistics. Its fields of application range from **neurophysiology...**” (1) This is important because it gives an idea of how widely and success time series analysis can be. The idea of choosing to use Time Series Analysis is almost simplistic; Given that the way EEG data is represented it makes perfect sense to use Time Series Analysis. In Fact, it seems the idea of Time Series Analysis is perfect for evaluating EEG data.

To test our subjects we used OSB - Optimal Subsequence Bijection. OSB is an algorithm that determines the optimal subsequence bijection between two sequences of real numbers. The purpose of this algorithm is to determine the optimal subsequence bijection of

Time series dataset. We can find solution easily by finding cheapest path, i.e. DAG- Directed Acyclic Graph. Sometimes we get elements that are noisy and may affect the results. We used OSB because it skips outlier elements and get significant results for real numbers. When we compare OSB with DTW – Dynamic Time Warping, OSB is always not better than DTW but OSB gives effective matching results for real numbers.

The second topic for discussion is (Expectation Maximization) clustering. While trying to figure out what type data we have we need some form of a pattern recognition formula, or algorithm. That service is provided to us by Expectation Maximization. Since we have a number of points distributed, which can be approximated by a mixture of Gaussian distributions. (The Comparison of two time series performed results in a histogram which follows Gaussian distribution.)

If the number of events is very large, then the Gaussian distribution function may be used to describe physical events. The Gaussian distribution is a continuous function which approximates the exact binomial distribution of events.

The Gaussian distribution shown is normalized so that the sum over all values of  $x$  gives a probability of 1. The nature of the Gaussian gives a probability of 0.683 of being within one standard deviation of the mean. The mean value is  $a=np$  where  $n$  is the number of events and  $p$  the probability of any integer value of  $x$  (this expression carries over from the binomial distribution). The standard deviation expression used is also that of the binomial distribution.

The Gaussian distribution is also commonly called the "normal distribution" and is often described as a "bell-shaped curve".

<http://hyperphysics.phy-astr.gsu.edu/Hbase/math/gaufcn.html>)

We wanted to estimate the parameters of each Gaussian distribution. The Expectation Maximization algorithm provides the framework for performing this objective. The Expectation Maximization algorithm "alternates between performing an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated". (2)

While performing the task of evaluating the data we also needed a way to represent the data as a function, while taking away data points that will not hurt our overall objective, of keeping a useful model. Wavelets become very useful in this area. "Wavelets are mathematical functions that cut up data into different frequency components, and then study each component with a resolution matched to its scale. They have advantages over traditional Fourier methods in analyzing physical situations where the signal contains discontinuities and sharp spikes."(3) Because EEG data contains this very idea of uniqueness, wavelets become the superior option. In particular, we used discrete 1-D wavelet transform. The discrete 1-D wavelet transform performs single level one dimensional wavelet decomposition with respect

to either a particular wavelet or particular wavelet decomposition filters that we specified. It returns a new model or representation of a model for further use.

When dealing with the various topics that were mention above, we have to step through a data flow and what we did step by step.

- 1) The first step is acquiring one of the subjects from the EEG data file (which is a matrix 40 X 101 X 198) but some may vary.
- 2) We call the function emcluster (while inside of emcluster)
- 3) We initialize the model. (model is a 40 X101 X198 matrix)
- 4) Compare models using time series analysis.
- 5) We then normalized by row
- 6) We weight model at each epoch.
- 7) We output the data.
- 8) Then we use wavelets to create a new model while in emcluster and repeat this process for 5 iterations.
- 9) Then emcluster ends. The output is a sequence of 5 numbers in 4 categories

## **Conclusion of the Analysis of Results**

We used OSB to obtain our results. OSB stands for Optimal Subsequence Bijection. It is an algorithm that determines the optimal Subsequence bijection between two sequences of real numbers. We tested total 10 subjects. We applied that to our results that we derived from the EEG Dataset. Our results consist of two sequences  $a$  and  $b$  then find subsequences  $a'$  of  $a$  and  $b'$  of  $b$  so that  $a'$  matches best with  $b'$ . It is explained in detail in further discussion.

Our results are divided in to two main parts: Cluster precision and Cluster recall. Cluster Precision means  $x\%$  of time that you will cluster an epoch correctly. In our results, total we have 4 clusters (5 numbers in each), two for left button and two right button pressed. To find **precision**, first we chose two numbers which matches best, one from left button pressed cluster and one from right button pressed cluster. (chose two numbers so that the difference between two is higher than other numbers) . Below is the result that we got from one subject.

ltypeout1: [75 76 27 21 22] =  $a$

rtypeout1: [77 72 24 22 24] =  $b$

ltypeout2: [25 24 73 79 78] =  $a'$

rtypeout2: [23 28 76 78 76] =  $b'$

First two rows are sequences/cluster  $a$  and  $b$  and other two are subsequences/cluster  $a'$  for  $a$  and  $b'$  for  $b$ . Each consists of 5 different numbers so we can easily chose two numbers that matches best.

**For example:** left button cluster: 76, right button cluster: 72 (from  $a$  and  $b$ ) We applied formula:  $76 / (76+72) = 0.51351351 = 51\%$

- To find right button cluster you apply same formula.

Cluster recall means y% of time that you will cluster a known left or right buttons. To find **cluster recall**, chose two numbers from subsequences. One from left cluster **a** and one from right cluster **b'**. Chose two numbers so that the difference between two is higher than other numbers.

**For example:** left button cluster: 76, right button cluster: 28 (from **a** and **b'**) Now, we applied the formula so:  $76 / (76+28) = 0.73076923 = 73\%$

- To find **right cluster recall** apply same formula but chose from **b** and **a'**. As we were given total of 19 subjects, we were able to test only 10 subjects. For other 9 subjects we ran in to an unknown error which we were not able to solve. It was difficult to find the error as our code worked for 10 subjects and not for other 9. According to that our error rate was approximately 48%. So can conclude that our results were provably better than guessing but not perfect. The results for all subjects are included in the Appendix.

## Appendix

SUBJECT NUMBER	NAME	RESULT	PRECISION & RECALL	%
1	ABBM	ltypeout1: [60 54 59 60 61]	left cluster precision	0.530973451
		rtypeout1: [68 51 56 53 55]	left cluster recall	0.571428571
		ltypeout2: [39 45 40 39 38]	right cluster precision	0.541176471
		rtypeout2: [31 48 43 46 44]	right cluster recall	0.576086957
2	BTBM	ltypeout1: [75 76 27 21 22]	left cluster precision	0.513513514
		rtypeout1: [77 72 24 22 24]	left cluster recall	0.730769231
		ltypeout2: [25 24 73 79 78]	right cluster precision	0.538461538

		rtypeout2: [23 28 76 78 76]	right cluster recall	0.75
3	BYBM	ltypeout1: [38 42 49 56 50]	left cluster precision	0.546296296
		rtypeout1: [36 57 59 49 39]	left cluster recall	0.50862069
		ltypeout2: [61 57 50 43 49]	right cluster precision	0.575757576
		rtypeout2: [63 42 40 50 60]	right cluster recall	0.538461538
4	CMBM	ltypeout1: [73 65 61 63 64]	left cluster precision	0.543478261
		rtypeout1: [86 73 72 75 72]	left cluster recall	0.51369863
		ltypeout2: [61 69 73 71 70]	right cluster precision	0.514492754
		rtypeout2: [56 69 70 67 70]	right cluster recall	0.515384615
5	GDBM	ltypeout1: [61 52 57 57 58]	left cluster precision	0.519685039
		rtypeout1: [66 52 54 56 54]	left cluster recall	0.628571429
		ltypeout2: [39 48 43 43 42]	right cluster precision	0.541666667
		rtypeout2: [33 47 45 43 45]	right cluster recall	0.64893617
6	KTBM	ltypeout1: [60 75 78 75 74]	left cluster precision	0.516556291
		rtypeout1: [59 75 73 73 70]	left cluster recall	0.513157895
		ltypeout2: [80 65 62 65 66]	right cluster precision	0.528571429
		rtypeout2: [85 69 71 71 74]	right cluster recall	0.525179856
7	LCBM	ltypeout1: [64 55 56 54 53]	left cluster precision	0.539130435
		rtypeout1: [51 58 59 57 62]	left cluster recall	0.52991453
		ltypeout2: [40 49 48 50 51]	right cluster precision	0.578947368
		rtypeout2: [55 48 47 49 44]	right cluster recall	0.569892473

8	MBBM	ltypeout1: [61 52 51 43 47]	left cluster precision	0.544303797
		rtypeout1: [60 53 46 36 43]	left cluster recall	0.614285714
		ltypeout2: [9 18 19 27 23]	right cluster precision	0.534482759
		rtypeout2:[7 14 21 31 24]	right cluster recall	0.462686567
9	MVBM	ltypeout1: [42 29 31 28 30]	left cluster precision	0.56
		rtypeout1:[38 23 25 22 26]	left cluster recall	0.583333333
		ltypeout2: [20 33 31 34 32]	right cluster precision	0.540540541
		rtypeout2: [17 32 30 33 29]	right cluster recall	0.564102564
10	SBBM	ltypeout1: [141 82 120 120 121]	left cluster precision	0.55033557
		rtypeout1: [135 67 116 119 121]	left cluster recall	0.719298246
		ltypeout2:[19 78 40 40 39]	right cluster precision	0.62745098
		rtypeout2: [32 100 51 48 46]	right cluster recall	0.779069767

## References

1. Journal of Time Series Analysis, M. B. Priestley, Publishing 2008, blackwellpublishing/Blackwell Synergy, May 1, 2008, <http://www.blackwellpublishing.com/journal.asp?ref=0143-9782&site=1>
2. Christopher M. Bishop, (2006) Pattern Recognition and Machine Learning, Springer, ISBN 0-387-31073-8. Sergios Theodoridis, Konstantinos Koutroumbas, (2006) Pattern Recognition (3rd edition), Elsevier, ISBN 0-12-369531-7.
3. IEEE Computational Science and Engineering, Summer 1995, vol. 2, num. 2, published by the IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, USA

4. Optimal Subsequence Bijection by Latecki, Longin Jan Wang, Qiang Koknar-Tezel, Suzan Megalooikonomou, Vasileios  
[http://www.ieeexplore.ieee.org/xpl/freeabs\\_all.jsp?isnumber=4470210&number=4470291&count=115&index=80](http://www.ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=4470210&number=4470291&count=115&index=80)
  
5. Credited to Longin Latacki, Thomas Young, Jack Shelley Tremblay and Tezel Suzan.
  
6. EEGLAB Revision History, March 20, 2008, <http://www.sccn.ucsd.edu/eeglab/revisions.html>
  
7. Andrew Blake, Bill Freeman, "Learning and Vision: Generative Methods"-ppt ICCV 2003 October 12, 2003

## **Code used in Project**

### **interpolation.m**

```
function [xi] = interpolation(interp)
```

```
%code for interpolation standard
```

```
xi = 1:101;
```

```
x = 1:2:101;
```

```
% get the y value
```

```
y = interp(1:51);
```

```
% does the interpolation
```

```
values = interp1(x,y,xi);
```

---

### **dwtWavelet.m**

```
function [ca2] = dwtWavelet(s)
```

```

% Perform single-level discrete wavelet transform of s by haar.
[ca1,cd1] = dwt(s,'haar');
%subplot(311); plot(s); title('Original signal');
%subplot(323); plot(ca1); title('Approx. coef. for haar');
%subplot(324); plot(cd1); title('Detail coef. for haar');

% For a given wavelet, compute the two associated decomposition
% filters and compute approximation and detail coefficients
% using directly the filters.
[Lo_D,Hi_D] = wfilters('haar','d');
% [ca1,cd1] = dwt(s,Lo_D,Hi_D);

% Perform single-level discrete wavelet transform of s by db2
% and observe edge effects for last coefficients.
% These extra coefficients are only used to ensure exact
% global reconstruction.
[ca2,cd2] = dwt(s,'db2');
%subplot(325); plot(ca2); title('Approx. coef. for db2');
%subplot(326); plot(cd2); title('Detail coef. for db2');
% returns approx coeff

```

## **emcluster.m**

```

% 20 lines of code
% //referances
% 40 channels
% 101 timepoints
% 40*101 is an epoch
% 2 is the number of models
% 198 number of epochs can be 398 for some subjects
function [output] = emcluster(eegset,type)

% 1. initialize model
% 2. compare models using tsDAG jump4 for loop use size funtion of eeg 3rd dimation each
epoch and compare to each model nested for loop.
% 3. create weight matrix
% 4. normalize row wise from em cluster
% 5. each mode wieghted * each epoch * wieght
% 6. model 1 * EEG model 1 40*101*198*2
% 7. models 1 and 2 come from dwt wavelet/ interpolation code

```

```

% 8. give 2 epochs for tsDAGjump4
% 1st step take 99 average them together model 1, 2nd 99 model 2
% 2nd step just compare
%
% *a bunch of random number with the same range and create
% call wrapper for anthonys code, mirals had code to it and works

% 1 I'm unsure of names for functions so i'll just put in generic names until i get the right ones

model1 = mean(eegset(:,1:99),3);
model2 = mean(eegset(:,100:198),3);
sigma = 1000000;
m(:,1) = model1;
m(:,2) = model2;

w = zeros(198,2);

for iter = 1:5
iter
  for i = 1:2
    for j = 1:198
      [d, pathcost, indxrow, indxcol, jumpcost] = tsDAGjump4(m(:,i), eegset(:,j));
      w(i,j) = normpdf(d,0, sigma)/ normpdf(0,0, sigma);
    end
  end
end

for k = 1:198
  w(k,:) = w(k,+)/sum(w(k,:));
end
outlist1 = find(w(:,1)>w(:,2));
outlist2 = find(w(:,2)>=w(:,1));
output.ltypeout1(iter) = length(find(type(outlist1)==1));
output.rtypeout1(iter) = length(find(type(outlist1)==2));
output.ltypeout2(iter) = length(find(type(outlist2)==1));
output.rtypeout2(iter) = length(find(type(outlist2)==2));

%output []
%output = wrapper(matrix)
%w(:,) = wrapper (w(:,))

for x=1:2 % for each model
  for y= 1:198 % each epoch
    % adj(i,channel,timePT)= wl(timePT) * EEG(timePT, channel)

```

```

        adj(:,:,y) = w(y,x) * eegset(:,:,y);
    end
    %output []
    %output = wrapper(matrix)
    m(:,:,x) = wrapper(adj);

    end
end

```

---

### **reducedataset.m**

```

clear accept
clear type

data=EEG.data;
data=mean(data,3);
subdata=data(13,:)-data(15,:);
imppartsubdata=subdata(251:375);
[maxval, maxidx] = max(abs(imppartsubdata));
for i=1:size(EEG.data,3)
    accept(i)=EEG.epoch(i).eventaccept(1);
    type(i)=EEG.epoch(i).eventepochtype(1);
end

if (iscell(accept))
    accept=cell2mat(accept);
end

if (iscell(type))
    type=cell2mat(type);
end

maxdiffdata=EEG.data(:,maxidx+251-50:maxidx+251+50,find(accept));

```

---

### **wrapper.m**

```

function [output] = wrapper(input)
%takes in a 40 by 101 matrix
%and outputs the dwt and interpotations

%returns the mean with the average

```

```

WiAverage = mean(input,3);
output=zeros(40,101);
temp = [];
for i = 1:40
    temp=dwtWavelet(WiAverage(i,:));
    output(i,:)=interpolation(temp);
end

%return output

```

---

### **tsDAGjump4.m**

```

% Longin Jan Latecki, latecki@temple.edu, June 2006
%
% Parameters:
% t1 - the first (query) time series. A row vector. t1 must be shorter
%     than t2
% t2 - the second (target) time series. A row vector. t2 must be longer
%     than t1
%
% Optional Parameters:
% warpwin - the warping window, same as that used in Dynamic Time Warping
% queryskip - sets a restriction on how many consecutive elements of
%     sequence t1 can be skipped in the matching
% targetskip - sets a restriction on how many consecutive elements of
%     sequence t2 can be skiped in the mataching
% jumpcost - the cost of jump in t1 used to balance the similarity of
%     elements. The number of rows we jump is mulitplied by jumpcost
%
% Return values:
% d - a similarity measure between t1 and t2
% pathcost - the cost of the "path" between t1 and t2
% indxrow - the index of elements in a substring of t1 that best match t2
% indxcol - the index of elements in a substring of t2 that best match t1
%
% Example:
% t1=[ 1 2 8 12 6 8];
% t2=[ 1 2 9 3 3 5 9];
% [d,pathcost,indxrow,indxcol,jumpcost] = tsDAGjump4(t1,t2)
%
% Important hint!
% Insert 0 as first and last elememets in t1 and t2 to be able to skip over
% the true first and last elements

```

```

function [d,pathcost,indxrow,indxcol,jumpcost] =
tsDAGjump4(t1,t2,warpwin,queryskip,targetskip,jumpcost)
    m=length(t1);
    n=length(t2);
    %if warpwin is not given then
    if ~exist('warpwin')
        warpwin = Inf;
    end
    %if targetskip is not given then
    if ~exist('queryskip')
        queryskip = Inf;
    end
    %if queryskip is not given then
    if ~exist('targetskip')
        targetskip = Inf;
    end

    matx = zeros(m,n);
    % create a difference matrix, each entry i,j contains differences of
    % corresponding elements i from t1 and j from t2
    for r = 1:m

        for c = 1:n

            matx(r,c) = norm(t2(:,r)-t1(:,c)).^2;
        end
    end

    %computing the jumpcost
    if ~exist('jumpcost')
        statmatx=min(matx');
        jumpcost=mean(statmatx) + std(statmatx);
    end

    % calling the main function
    [pathcost,indxrow,indxcol] = findpathDAG(matx,warpwin,queryskip,targetskip,jumpcost);

    d = sqrt(sum((t1(indxrow)-t2(indxcol)).^2))/length(indxrow); %Euclidean distance

    %{
    % plots the result

```

```

%%comment out for real tests!!!!
figure;
hold on;
axis off;
l1 = length(indxrow); %length of indxrow = length of indxcol
shifft1=5*std(t1);
plot(t1+shifft1,'r', 'LineWidth',2); % plots the query shape time series with red solid line
shifft2=0.1*std(t2);
plot(t2+shifft2, 'LineWidth',2); % plots the target shape time series with blue solid line
for i=1:l1;
    x = [indxrow(i), indxcol(i)];
    y = [t1(indxrow(i))+shifft1, t2(indxcol(i))+shifft2];
    plot(x,y)
end
hold off;
%}
return

%-----
% function [pathcost,indxrow,indxcol] =
findpathDAG(matx,warpwin,queryskip,targetskip,jumpcost)
% find path with. min. cost
% input: difference matrix
% output: the cost of cheapest path, and indices of the cheapest path
% The cheapest path is computed on DAG
%-----
function [pathcost,indxrow,indxcol] =
findpathDAG(matx,warpwin,queryskip,targetskip,jumpcost)

[m,n]=size(matx);
weight=Inf(m,n); %the weight of the actually cheapest path
camefromcol=zeros(m,n); %the index of the parent col where the cheapest path came from
camefromrow=zeros(m,n); %the index of the parent row where the cheapest path came from
stepcount=ones(m,n); %counts the number of steps (corresponding pairs)

weight(1,:)=matx(1,:); %initialize first row

for i=1:m-1 %index over rows
    for j=1:n-1 %index over columns
        if abs(i-j)<=warpwin %difference between i and j must be smaller than warpwin
            stoprowjump=min([m, i+queryskip]);
            for rowjump=i+1:stoprowjump %second index over rows
                stopk=min([n, j+targetskip]);

```

```

        for k=j+1:stopk %second index over columns
            newweight = ( weight(i,j) + matx(rowjump,k) + sqrt((rowjump-i-1)^2+(k-j-1)^2)*jumpcost) ; %we favor shorter jumps by multiplying with (rowjump-i-1)
            if (1/(stepcount(i,j))*weight(rowjump,k) >= (1/(stepcount(i,j)+1))*newweight
                weight(rowjump,k) = newweight;
                camefromrow(rowjump,k)=i;
                camefromcol(rowjump,k)=j;
                stepcount(rowjump,k)=stepcount(i,j)+1;
            end
        end
    end
end
end
end
end
weight=weight./stepcount;

% collecting the indices of points on the cheapest path
% mincol is the index of the col of the last point on the cheapest path
[pathcost,mincol]=min(weight(m,:)); % pathcost: minimal value
minrow=m;
indxcol=[];
indxrow=[];
while (minrow>0 && mincol>0)
    indxcol=[ mincol indxcol];
    indxrow=[ minrow indxrow];
    mincoltemp=camefromcol(minrow,mincol);
    minrow=camefromrow(minrow,mincol);
    mincol=mincoltemp;
end
return

```

```

%=====

```

### **%HOW TO RUN THIS PROGRAM**

```

%=====

```

```

% a=[1 2 3 7 12 3 10];
% b=[12 5 3 6 7 15 8];
% [d,pathcost,indxrow,indxcol,jumpcost] = tsDAGjump4(a,b);
% a(indxrow)
% b(indxcol)
% [d,pathcost,indxrow,indxcol,jumpcost] = tsDAGjump4(a,b);
% a=[a; 12 34 3 40 21 12 7]
% b=[b; 12 34 3 40 21 12 7]
% [d,pathcost,indxrow,indxcol,jumpcost] = tsDAGjump4(a,b)

```

%