CS 229: Dimensionality-Reduction of Neural Data

Zuley Rivera Alvidrez

Introduction

In the experiments in the Shenoy lab, monkeys make reaching movements while neural data is simultaneously recorded from 96 electrodes. Our ultimate goal is to understand how the population activity of the neural signals relates to the monkey's behavior. Recent advances in neural recording have allowed us to collect data from many neurons simultaneously, however, data analysis techniques have not yet been modified to be able to take advantage of this. What is the best way to represent the relevant information encoded in populations of neural activity, on an instant-by-instant basis? Recent work in this field suggests that the information encoded in the activity of the population of neurons can be represented in a lower-dimensional space. In this project, we use machine learning techniques to help us evaluate an experimental hypothesis involving neural plans encoded by the activity of neurons in the premotor cortex of monkeys.

Background

Let's say you're in the forest, hunting for your dinner. There are a number of different animals you might choose, but you have only one spear. In the distance, coming towards you, you see two animals. As the animals get closer, you instinctively prepare yourself to throw your spear. How do you decide which animal to target? Do you simultaneously plan throws to both locations where the animals may emerge, thereby allowing the maximum flexibility based on the incoming sensory stimuli? Or do you plan your throw to one location, then the other, and alternate back and forth as it becomes clearer which animal is the better target?

We are interested in understanding how the brain forms plans about competing actions, on an instant-by-instant basis. Instead of studying hunters choosing their prey, our lab studies monkeys making sensory-guided reaching movements to targets on a computer screen (see Figure 1). While a monkey plans these reaches, we record the activity of neurons in the motor and premotor (PMd) cortex of his brain. Using experimental data from a more complex task, we seek a method that will afford a better description and predictions than are possible with the current methods used in this field.

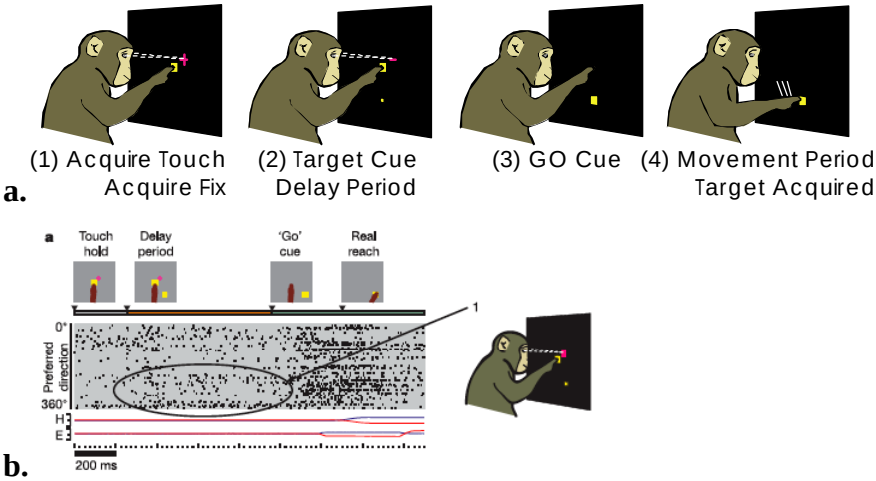


Figure 1. Schematic of delayed reach task. (a) A monkey is trained to sit in front of a computer screen and reach to targets at locations specified by a yellow square. The monkey is rewarded for touching this target within a fixed time window after a go cue is given. (b) Neural activity is recorded while the monkey performs this task. The top panels indicate the different epochs of the task, and the black tick marks represent the spiking activity of the population of cells recorded, sorted by tuning direction. The encircled region contains spikes tuned to the upcoming reach location.

Behavioral task

The data we will use for this project comes from a modified version of the delayed reach task (see Figure 2). In this new task, the 2-target task, two possible target locations are shown on the computer screen. After a delay period, one possible target is removed and the go cue is given after a second delay. Since the monkey is not rewarded unless he touches the target fast enough, he cannot wait until the targets are disambiguated to make his plan.
During this delay period, where is the monkey planning to reach?

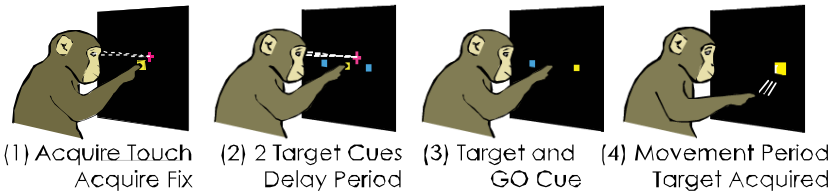


Figure 2. Schematic of 2-target task. This task is similar to the delayed reach task, but includes delay period during which the monkey does not know to which target he will be cued to reach. At the end of the delay period, the 2 potential targets are disambiguated and the go cue is given.

Hypotheses

Specifically, we want to know whether the monkey is planning a reach to both targets at once, or whether he is changing his plan back and forth between plans to each target. There is evidence in the literature that would support either of these options (Cisek & Kalaska, 2002, evidence of co-coding two motor plans; Horwitz & Newsome, 2001, evidence for flipping plans).

By representing the firing rate (activity level) of each cell as a point in a multidimensional space, for each point in time, we can plot out how the monkey's plan trajectory evolves through time. In Fig. 3a, each axis represents the firing rate of one neuron, in the delayed reach task. The point near the origin is the baseline activity of the cells recorded, and as time elapses, the firing rates of the cell create the path towards the grey cloud. When the firing rates of the population fall into the grey cloud, the monkey is ready to make the movement corresponding to this subspace, here a rightwards reach. This "optimal subspace hypothesis" is a common way our group thinks about preparatory motor activity, and is described in Churchland et al., 2006.

In the 2-target task, if the monkey is planning to both targets at once, we should see a trajectory that looks similar to that shown in Fig. 3b. Conversely, if the monkey is flipping his plan between targets, we should see a trajectory in this space that looks more similar to the cartoon in Fig. 3c.

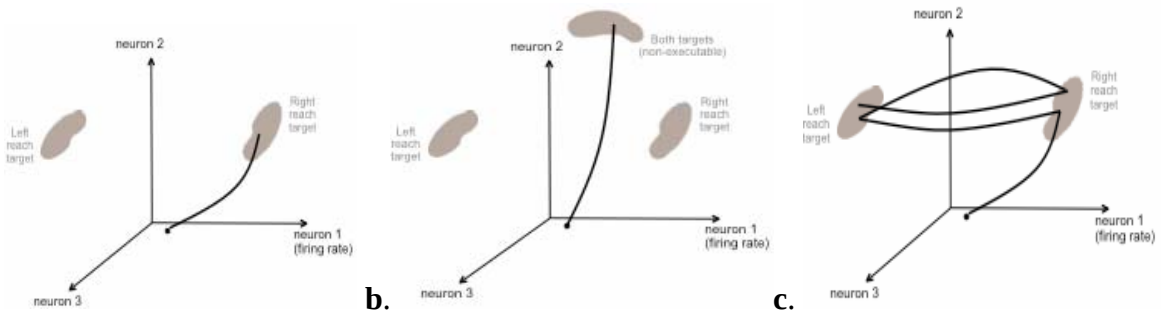


Figure 3. Cartoons of the optimal subspace hypothesis. (a) Plan trajectory in firing rate space, for the delayed reach task. (b) Expected plan trajectory for the delay period of the 2-target task, if the monkey is planning reaches to both target locations simultaneously. (c) Expected plan trajectory for the delay period of the 2-target task, if the monkey is flipping his plan between both target locations.

Dimensionality Reduction Techniques

In this project, we use linear dimensionality reduction techniques. The use of popular non-linear methods (LLE and Isomap) was investigated previously and their performance was found to be inadequate when compared to the performance of linear methods such as principal component analysis (PCA). Linear methods have the advantage of reliability and low computational complexity. They are guaranteed to show genuine properties of the original data and to produce axes that are meaningful in that they are linear combinations of the original axes.

PCA: Principal component analysis (PCA) is one of the most popular and well understood methods for dimensionality reduction. The data is projected into a set of uncorrelated variables called the principal components. Each principal component accounts for as much of the variability of the data as possible. The disadvantage of the technique is that it does not learn a generative model for the data, and one cannot compute the likelihood of the data.

Factor Analysis and Sensible Principal Component Analysis: Factor Analysis (FA) and Sensible Principal Component Analysis (SPCA), have a similar underlying static model of the data:

$$y = Cx + \mathbf{v} \qquad x \sim N(\mathbf{0}, \mathbf{I}) \qquad \mathbf{v} \sim N(\mathbf{0}, R) \qquad \Rightarrow y \sim N(\mathbf{0}, CC^T + R)$$

where $\boldsymbol{x}$ is the underlying or latent state vector, $\boldsymbol{y}$ is the vector of observations, and $\boldsymbol{v}$ is the observation noise with covariance matrix $\boldsymbol{R}$. The key assumption of FA is that the matrix $\boldsymbol{R}$ is diagonal. That is, it puts the variance unique to each coordinate in $\boldsymbol{R}$

and the correlation structure in ***C***. SPCA, however, assumes that ***R*** is a multiple of the identity matrix $\alpha \boldsymbol{I}$, where α represents the global noise level.

Both models are learned using the EM algorithm (see Roweis, 1999):

E-step: $Q_i(x^{(i)}) = p(x^{(i)} \mid y^{(i)}; C, R) = N(\beta y^{(i)}, \mathbf{I} - \beta C)\,|_{x^{(i)}}$ *where* $\beta = C^T(CC^T + R)^{-1}$

Let $\overline{X}$ be the expected value of the latent state vectors X found in E-step. Then the M-step becomes:

M-step: $C^{new} = Y\overline{X}^T\Sigma^{-1}; \quad R^{new}{}_{ii} = \left(YY^T - C\overline{X}Y^T\right)_{ii} / n; \quad where\ \Sigma = n\mathbf{I} - n\beta C + \overline{XX}^T$

where Y is the *p* x *n* mean-subtracted matrix of n observations of p variables**.**

The EM algorithm for SPCA is the same, except that R is now computed as αI where

$$\alpha \longleftarrow \sum_{j}^{p} \left(YY^T - C\overline{X}Y^T\right)_{jj} / p$$

Weighted PCA: While PCA and SPCA had been previously found to produce results that best reflect the expected trajectories of the neural data, both methods suffer from sensitivity to outliers. Furthermore, being unsupervised methods, they do not allow for incorporation of known labels or structure of the data into the projection algorithm. Because the directions of the principal components that maximize the variance are not necessarily the same as the directions that maximize the separation between classes, PCA does not always perform well when used as a classifier. In order to address both shortcomings, the concept of weighted PCA has been recently proposed (Koren, 2004). The objective of PCA can be expressed as finding the p-dimensional projection that maximizes the sum of the squares of the Euclidian distances in the p-dimensional space between vectors i and j of the training set.

$$\sum_{i<j} \left(\mathrm{dist}_{ij}^{p}\right)^2$$

Using the concept of Laplacian matrices and basic linear algebra, this objective can be generalized to the weighed sum:

$$\sum_{i<j} d_{ij} \left(\mathrm{dist}_{ij}^{p}\right)^2$$

Where d_{ij} are the elements of the "dissimilarity" Laplacian matrix. Laplacian matrix is a symmetric positive-semi-definite matrix with zero sums of rows and columns. By adequately defining this weight matrix, one can incorporate labels (supervised PCA), and/or reduce the sensitivity to outliers by making the weights inversely proportional to the Euclidian distance (normalized PCA). The n by b Laplacian matrix can be easily constructed by setting:

$$L_{ij}^{d} = \begin{cases} \sum_{j=1}^{n} d_{ij} & i = j \\ -d_{ij} & i \neq j\,. \end{cases}$$

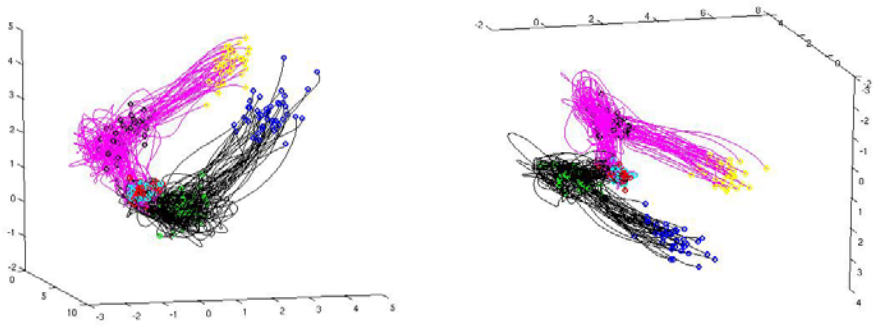
The new objective function can still be solved optimally using an eigenvalue decomposition approach.

Results

The EM algorithm for factor analysis was implemented and applied to neural data for two specific target locations for the simple delay-reach task of fig. 1. The same projections were found using SPCA. The trajectories obtained using both methods are shown in Figure 4. While, it is expected that factor analysis which assumes different noise variances per coordinate will perform better than SPCA, which assumes equal variances, the trajectories for this data set were not significantly different. Figure 5 (a,b, and c) shows the trajectories obtained for the modified task including an ambiguity period. The planning end-point clusters corresponding to the two targets shown in the screen are presented in the figure as well as the neural trajectories during the ambiguous period in which the monkey does not know to which target he will be reaching. For two sets of targets (Fig. 5a, and 5b), the trajectories appear to be in a region in between the two target clusters. For a third pair of target locations (Fig. 5c), the trajectories are clearly biased towards one of the target clusters. Figure 5d shows the trajectories when the monkey's guess of the target location is wrong. Once the target is disambiguated, the monkey corrects its neural plan.

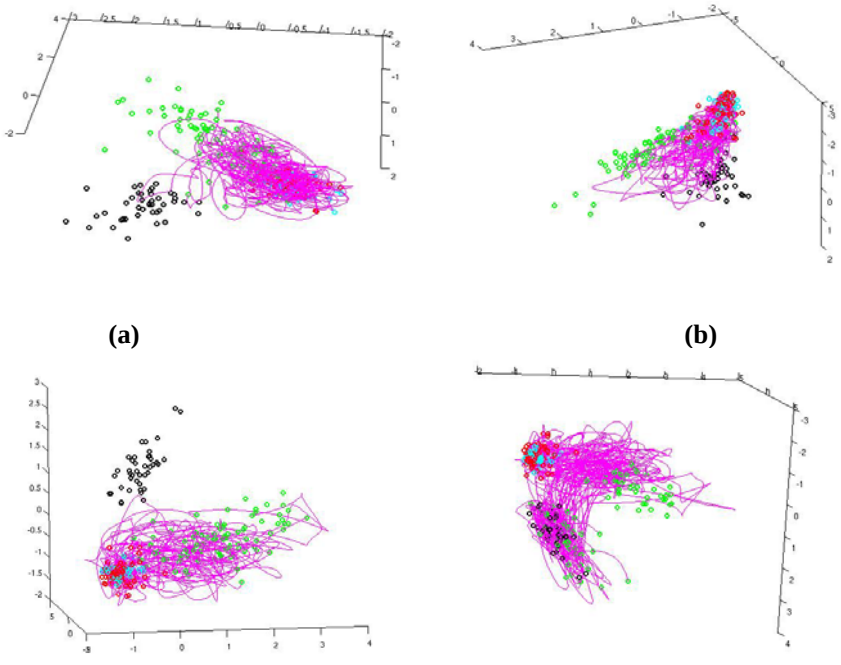
In order to understand whether something interesting was occurring in the trajectories shown in Figures 5a and 5b, the endpoints of the trajectories at the end of the ambiguous period were projected to a 3-dimensional space using PCA and weighted PCA. If the monkey is planning then the end-points should be separable from the baseline state of the trajectories. Furthermore, if the plan takes into account the location of the targets presented, the end-points of the two sets of trajectories should be themselves separable. Figure 6a, shows that the PCA projection is dominated by outliers in the data. A projection using supervised PCA with

binary weights (weights of vectors belonging to the same set of targets were set to zero, and those belonging to different target sets were set to one) was produced, but as shown in Figure 6b, it still suffers from high sensitivity to outliers. The results of normalized PCA, which is designed to reduce the effect of outliers, are shown in Figure 6c. The mean of the clusters is different for the two sets of targets, and also separated from the mean of the baseline state. Finally, maximum separation was obtained with a mixture of normalized and supervised PCA, as shown in Figure 6d. Figure 6e shows the result of projecting the data for all sets of targets using normalized and supervised PCA. The projected end-points show three classes which correspond to the pairs of targets that were presented together. These results support the idea that the monkey is planning, and its plan is different depending on the targets presented.



(a) **(b)**

Figure 4. Neural trajectories for simple delay-reach task for two target locations projected into a three-dimensional space using SPCA(a) and Factor Analysis (b). The initial state is shown in red and blue dots for each target, the end-points of the planning period are shown in black and green, and the end-points at the movement onset are shown in blue and yellow.



(a) **(b)**

(c) **(d)**

Figure 5. Neural trajectories for the task with ambiguity period. The planning clusters corresponding to the pair of targets presented are shown in green and black dots. The pink traces are the trajectories followed during the ambiguous period for the three possible locations of the two targets presented (a, b and c). Panel (c) suggests monkey is planning to reach to a particular target. Panel (d) shows the trajectories when the target chosen by the monkey is not the one that he is instructed to reach. The monkey corrects his plan.

Conclusions

The objective of this study was to provide insight into the neural mechanisms of decision-making by the examination of the neural activity of a monkey that is presented with a choice. For two sets of targets locations, it appears that while the monkeys planning activity is different from baseline, it is not biased towards any of the targets, supporting the idea that the monkey is planning to both targets simultaneously. For the third set of target locations, the monkey's neural plan is clearly biased towards one of the

targets. Current work included correlating the neural data with behavioral measurements, such as reaction time, in order to provide more insight to the results obtained.

Different projection methods were employed to evaluate the results. It was found that for this particular dataset factor analysis and SPCA produced very similar results. This is likely due to the fact that most of the neurons included exhibited similarly high firing rates and hence, similar noise variances (assuming a Poisson model in which variance and mean are the same). For data recorded from neurons with more different firing rates, it is expected that factor analysis will outperform SPCA by assuming different variance for each neuron. Our results also show that weighted PCA can be used to overcome the shortcomings of simple PCA with the same advantages of linear projection methods.

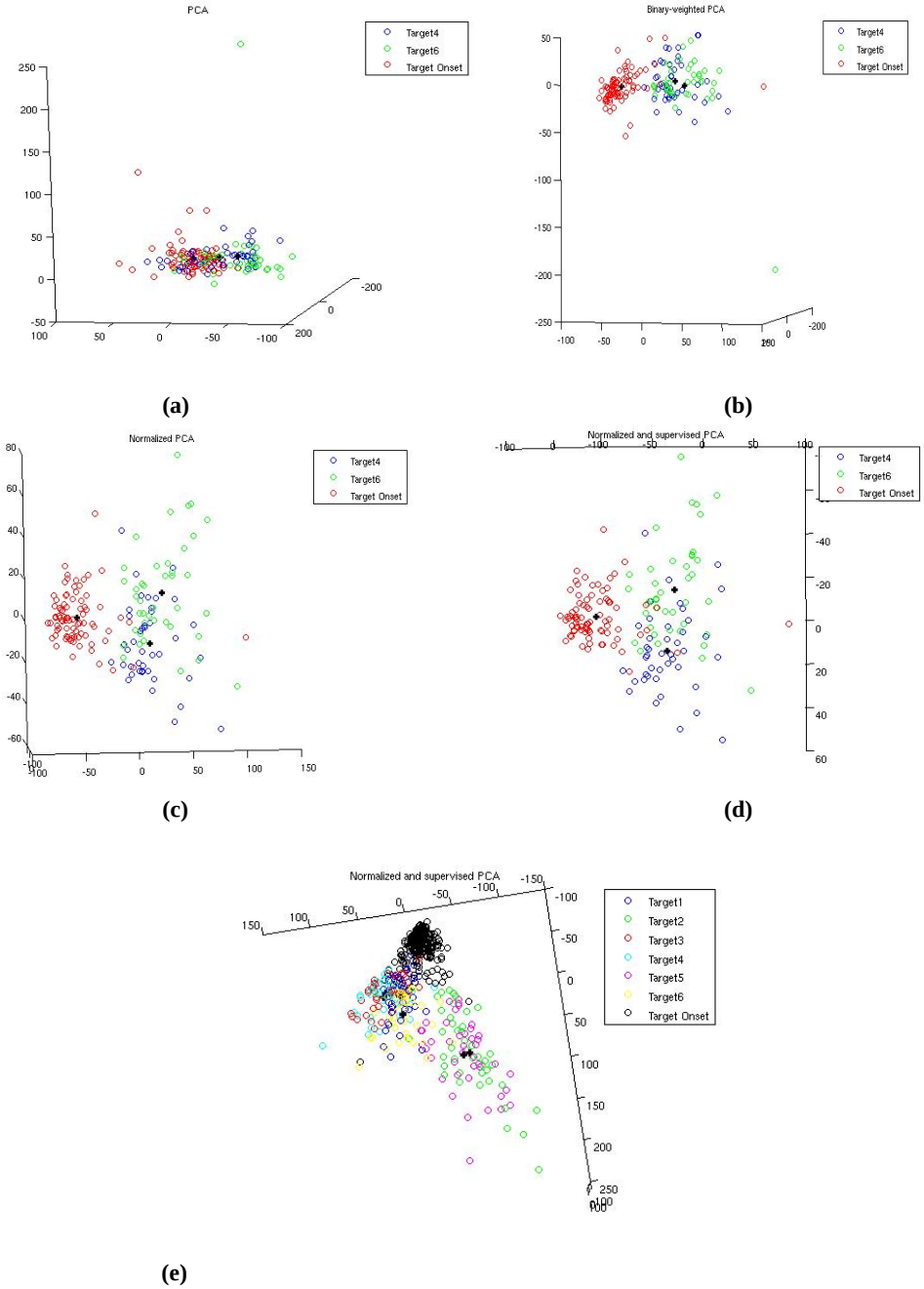


Figure 6. (a-d) End-point analysis of the trajectories corresponding to Figs. 5a (green dots) and b (blue dots) obtained using PCA (a), weighted or supervised PCA with binary weights (b), normalized PCA (c), normalized and supervised PCA(d). The black dots represent the mean of each cluster. The red dots are the initial state. Panel e shows the output of normalized and supervised PCA when all of the

trajectory endpoints are provided. The clustering method produced three clusters (green and purple, yellow and blue, and cyan and red) corresponding to targets that were shown together.

Acknowledgements

This work was done in collaboration with Rachel Kalmar (background, motivation), Afsheen Afshar (discussions, help with data processing), and various other students from Shenoy's lab who contributed with data collection and analysis.

References

Afshar A and Cunningham JC. Using Embedding Algorithms to Find a Low Dimensional Representation of Neural Activity During Motor Planning CS229 Final Project, Fall 2005.

Churchland MM, Yu BM, Ryu SI, Santhanam G, Shenoy KV (2006). Neural variability in premotor cortex provides a signature of motor preparation. J Neurosci. Apr 5;26(14):3697-712.

Cisek and Kalaska. Simultaneous encoding of multiple potential reach directions in dorsal premotor cortex. J Neurophysiol. 87(2):1149-54.

Horwitz and Newsome (2001). Target selection for saccadic eye movements: Prelude activity in the superior colliculus during a direction discrimination task. J. Neurophysiol. 86: 2543-2558.

Koren Y. and Carmel L. (2004). Robust Linear Dimensionality Reduction. IEEE Transactions on Visualization and Computer Graphics. 10(4): 459- 470.

Roweis S. and Ghahramani Z. (1999). A unifying review of linear Gaussian models. Neural Computation 11(2): 305-345.

CODE

The Matab code produced for this project built on the extensive code for data processing produced by several students in Shenoy's lab. Furthermore, plotting routines, preprocessing of neural data sets, and various other tasks utilized code written by Agsheen Afshar and John Cunningham. Such code is not included here for length considerations. Also not included is code that was only slightly modified. Additional or complete code can be provided upon request.

```
% This matlab script produces three-dimensional projections using SCPA or PCA of the
neural data saved in specified mat files for the time period specified (planning,
ambiguous, all, etc). It also performs end-point analysis using PCA, supervised PCA,
normalized PCA and supervised/normalized PCA.

option =2;
loadData = 0;
addpath('forMark');

%load data
if(loadData)
  load('matFiles/R_12_09.mat');
  load('matFiles/bu_12_09.mat');
end

%order data according to target location
OR = TrialOrderArrayGrid(Rnew);

%get the target coordinates
for i = 1:max(size(OR));
  targets(i) = OR(i).R(1).TrialParams.targetAngularDirection;
  tar_cor(:,i) = [OR(i).R(1).TrialParams.target1X ; OR(i).R(1).TrialParams.target1Y];
  end
%clear OR

if(option ==1)

%specify which targets are to be projected as well as the neural parameters of the data
%that is to be projected
tars1 = [2 5];
tars2 = [2];
colorCue1  = [0];
colorCue2 = [400];
%delay1= [100];
%delay2=[100];

%get neural data satisfying the parameters specified.
tp = [Rnew.TrialParams];
cc = [tp.timeColorCue];
td = [tp.timeDelay];
tt = [tp.targetAngularDirection];

%build new R based on specs
%valid1 = (cc == colorCue1) & (td == delay1) & (ismember(tt,targets(tars1))) ;
%valid2 =  (cc == colorCue2) & (td == delay2) & (ismember(tt,targets(tars2))) ;

valid1 = (cc == colorCue1)  & (ismember(tt,targets(tars1))) ;
valid2 =  (cc == colorCue2) & (ismember(tt,targets(tars2))) ;
```

```
R_new = Rnew(valid1 | valid2);
tp = [R_new.TrialParams];
cc = [tp.timeColorCue];
for i=1:max(size(R_new))
  if (cc(i) == colorCue2)
    R_new(i).TrialParams.target1X = R_new(i).TrialParams.target1X+1;
  end
end

OR_new = TrialOrderArrayGrid(R_new);
clear t_coor;
for i = 1:max(size(OR_new));
  t_coor(:,i) =[ OR_new(i).R(1).TrialParams.target1X;
OR_new(i).R(1).TrialParams.target1Y];
end


%print Stats
fprintf (['Found ', sprintf('%d', size(R_new,2)), ' trials out of ', sprintf('%d',
size(Rnew,2)), '\n']);
fprintf(['For specs 1, found: ', sprintf('%d', sum(valid1)),' \n']);
fprintf(['For specs 2, found: ', sprintf('%d', sum(valid2)),' \n']);

%Get data for projection
tartmp = 1:max(size(OR_new));
fprintf('\n Getting Data... \n');
[data, tBegs, tGo, color_data, tBounds,R] = funcMarkData(OR_new, bestUnits, 'tars',
tartmp , 'DistoPeriMove',1);
fprintf('done.\n');

figure(1);
ax1 = 1;

fprintf('Projecting onto lower dimensions...\n');
%specifying the option ‘dofa’ produces projections using factor analysis. The
%default option is SPCA
[Y, newBounds, allTrajs, color_trunc,PM,mu] = funcRunProjectz(data, tBegs, tGo,
color_data, tBounds, ax1,'shouldPlot',1);



elseif(option == 2)
%End point Analysis

%specify targets
tars1= [2 5 ];
tars2 = [2];

%Get data:

colorCue1  = [-999];
colorCue2 = [400];
delay= [90];
tp = [Rnew.TrialParams];
cc = [tp.timeColorCue];
td = [tp.timeDelay];
```

```
tt = [tp.targetAngularDirection];

%build new R based on specs
%valid1 = (cc == colorCue1) & (td == delay) & (ismember(tt,targets(tars1))) ;
%valid2 =  (cc == colorCue2) & (td == delay) & (ismember(tt,targets(tars2))) ;

valid1 = (cc == colorCue1)  &  (td <= delay) &(ismember(tt,targets(tars1))) ;
valid2 =  (cc == colorCue2)  & (td <= delay) & (ismember(tt,targets(tars2))) ;

R_new = Rnew(valid1 | valid2);
tp = [R_new.TrialParams];
cc = [tp.timeColorCue];
for i=1:max(size(R_new))
  if (cc(i) == colorCue2)
R_new(i).TrialParams.target1X = R_new(i).TrialParams.target1X+1;
end
end

OR_new = TrialOrderArrayGrid(R_new);
for i = 1:max(size(OR_new));
  t_coor(:,i) =[ OR_new(i).R(1).TrialParams.target1X;
OR_new(i).R(1).TrialParams.target1Y];
end

%find correct order

%print Stats
fprintf (['Found ', sprintf('%d', size(R_new,2)), ' trials out of ', sprintf('%d',
size(Rnew,2)), '\n']);
fprintf(['For specs 1, found: ', sprintf('%d', sum(valid1)),' \n']);
fprintf(['For specs 2, found: ', sprintf('%d', sum(valid2)),' \n']);

%Get data for projection
tartmp = 1:max(size(OR_new));
fprintf('\n Getting Data... \n');
[data, tBegs, tGo, color_data, tBounds,R] = funcMarkData(OR_new, bestUnits, 'tars',
1:max(size(OR_new)) , 'onlyDisAmb',1);
fprintf('done.\n');

figure(1);
ax1 = 1;

fprintf('Projecting onto lower dimensions...\n');
[PMw, PMn, PMnw, PCA, X] = pointWPCA(data,tBounds, color_data, 1:max(size(OR_new)), 2);

end


% Get some RTs
%

clear meanRT;
for i = 1:max(size((OR_new)))
  meanRT(i) = mean([OR_new(i).R.marksRT]);
  end
```

%FACTOR ANALYSIS EM ALGORITHM

```
% This function performs Factor Analysis on the data matrix given.
% See Roweis, 1997
%
% varargin:
%   shouldPlot (1 if should plot)
%

function [PM, R_out] = facan2(data,k, varargin)

  shouldPlot = 0;
  assignopts(who, varargin);

  % note k = d, the desired embedding dimension
  % note p = n, the size of the input dimension
  p = size(data,1);
  n = size(data,2);
  Y = data ;
  tol_likelihood = 1e-7; % 1e-5 is safe and thorough   % EM algorithm
  C = 1*rand(p,k);
  R = diag(rand(1,p)); %initialize diagonal matrix to random values

  likelihood = 100;
  converged = 0;
  iter = 0;
  learning = [];

%iterate EM
  while(converged == 0 && iter<250)
    iter = iter + 1;
    fprintf(['Factor Analysis EM iteration: ',sprintf('%d',iter),'; likelihood value:
',sprintf('%d',likelihood),'\n']);
    % E step
    Beta = C'*(inv(C*C'+R));
    X_hat = Beta*Y;
    V= eye(k)-Beta*C;
    delt = Y*X_hat';
    gam = X_hat*X_hat'+n*V;
    % M step
    C_new = delt*inv(gam);
    R_new = diag(diag(Y*Y'/n-C_new*delt'/n));
 %   C_new
 %   R
    % check for convergence

    S = C_new*C_new'+R_new;

    likelihood_new = sum(-1/2*(sum(Y.*(inv(S)*Y),1) + sum(log(svd(S))) + log(2*pi))); %
assume y_i are iid --> log likelihood of whole training set; for comparison with proper
likelihoods, this equation needs revision
```

```
    if (abs((likelihood_new-likelihood)/likelihood)<tol_likelihood)
      converged = 1;
    end
    % update model parameters
    C_old = C;
    C = C_new;
    R_old =R;
    R = R_new;
    likelihood_old = likelihood;
    likelihood = likelihood_new;
    learning(iter)=likelihood;
  end
  R_out = R;

 PM= C'*inv(C*C'+R); %Beta is the projection matrix
 PM = PM';

  % have now learned the model y~N(0,C*C' + eps*I), so we can evaluate test data.
  % plot learning curve
 % PC = C'*inv(C*C'+eps1*eye(p));
 % PC =PC';
  if shouldPlot == 1
    figure;
    plot(learning);
    xlabel('EM Iteration');
    ylabel('Log Likelihood');
    title('Learning Curve of SPCA EM Algorithm');
  end
```

```
%Weighted PCA

%For as many targets as desired. Finds projection that maximizes distance between
target endpoints (1)
%or distance between startpoints and target endpoints (2)
function [PMw, PMn, PMnw, PCA, X] = pointWPCA(datatrunc, tBounds, color_data,tars,
option);
  mu = repmat(mean(datatrunc,2),1,size(datatrunc,2));
%Just retain end-points
n = size(datatrunc,2); %num of vectors
m = size(datatrunc,1); %num of vars
X = (datatrunc-mu)'; % nxm



if (option ==1)
%pick only endpoints
X = X(tBounds(3,:),:); % X is now trials xm
n= max(size(tBounds));

%figure out how many trials per target from color_data
p = 0;
for i=1: max(size(tars))
  ta(i) = sum((color_data(1,:)==i));
  targ(i) = find(tBounds(2,:)== ta(i)+p);
```

```
  p= p+ta(i);

end
Lap = ones(n,n);
ind1=[1 targ(1:end-1)+1];
ind2=targ;


  sta= cumsum(ta);


%give weights of zero to data from same class (same target)
for i=1:max(size(targ))

 Lap(ind1(i):ind2(i),ind1(i):ind2(i))= 0;

end

%Ensure Laplace conditions
for i=1:n
  Lap(i,i)=0;
  Lap(i,i) = -sum(Lap(i,:));
end

%Solve using SVD decomposition

SS = X'*Lap*X;
[U,D,V2] = svd(SS);
P = V2(:,1:3);

PM = P;
Yn = X*P;
Tr = (datatrunc-mu)'*P;
colors = [ 'b' 'g' 'r' 'c' 'm' 'y' 'k'];

figure(1);
for i=1:max(size(tars))

plot3(Yn(ind1(i):ind2(i),1), Yn(ind1(i):ind2(i),2),Yn(ind1(i):ind2(i),3), [colors(i)
'o'] )
hold on
st{i} = ['target' num2str(tars(i))];
end


hold off
legend(st);
%plotEmbedded(Y,newBounds, color_trunc);

%Do regular PCA

Sig2 = X'*X;
%[V3,D] = eig(Sig2);
[U, S, V3] =svd(Sig2);
PCA = V3(:,1:3);
YPCA = X*PCA;
```

```
figure(3);
for i=1:max(size(tars))
  %plot3(Yn(1:t1,1), Yn(1:t1,2),Yn(1:t1,3), 'go' )
  plot3(YPCA(ind1(i):ind2(i),1), YPCA(ind1(i):ind2(i),2),YPCA(ind1(i):ind2(i),3),
[colors(i) 'o'] )
  hold on

end

hold off

%Ntrials = 40;
Ntrials = max(size(tBounds));
tBounds2 = tBounds(:,1:Ntrials);
Tr2 = Tr(1:tBounds2(end,end),:);
colort = color_data(:,1:tBounds2(end,end));

%plotEmbedded(Tr2',tBounds2,colort);


elseif (option ==2) %pick starting points, put them in one class and separate from
other endpoints


  X2 = X(tBounds(2,:),:); % X is now trials xm
  X = [X2 ; X(tBounds(1,:),:)]; %append starting pts
  n= max(size(tBounds))*2;

  %figure out how many trials per target from color_data
  p = 0;
  for i=1: max(size(tars))
    ta(i) = sum((color_data(1,:)==i));
    targ(i) = find(tBounds(2,:)== ta(i)+p);
    p= p+ta(i);
  end

  sta= cumsum(ta);

%Give weights of zero to vectors from same class
  Lap = ones(n,n);
  ind1=[1 targ(1:end-1)+1 targ(end)+1];
  ind2=[targ n];
  for i=1:max(size(ind1))
    Lap(ind1(i):ind2(i),ind1(i):ind2(i))= 0;

  end

  %Enforce Laplacian conditions
  for i=1:n
    Lap(i,i)=0;
    Lap(i,i) = -sum(Lap(i,:));
  end


%solve using SVD decompostions
  SS = X'*Lap*X;
  [U,D,V2] = svd(SS);
```

```
  P = V2(:,1:3);
  PMw = P;

  Yw = X*P;
  Tr = (datatrunc-mu)'*P;
  colors = [ 'b' 'g' 'r' 'c' 'm' 'y' 'k'];
  Yn = Yw;
  meani=[];

%plot data and compute means of clusters
  figure(1);
  for i=1:max(size(ind1))
    meani(i,:) =mean( Yw(ind1(i):ind2(i),:),1);
    plot3(Yn(ind1(i):ind2(i),1), Yn(ind1(i):ind2(i),2),Yn(ind1(i):ind2(i),3),
[colors(i) 'o'] )
    hold on
    if(i<= length(tars))
      st{i} = ['Target' num2str(tars(i))];
    else
      st{i} = 'Target Onset';
    end
  end
  legend(st);
  title ('Binary-weighted PCA');
  plot3(meani(:,1), meani(:,2), meani(:,3), '+k','LineWidth',3)
  hold off

  Tr = (datatrunc-mu)'*P;
% Ntrials = 2;
  Ntrials = max(size(tBounds));
  tBounds2 = tBounds(:,1:Ntrials);
  Tr2 = Tr(1:tBounds2(end,end),:);
  colort = color_data(:,1:tBounds2(end,end));

  %plotEmbedded(Tr2',tBounds2,colort);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
  %Do regular PCA

  Sig2 = X'*X;
  [U, S, V3] =svd(Sig2);
  PCA = V3(:,1:3);
  YPCA = X*PCA;

  figure(2);
  for i=1:max(size(ind1))
    meani(i,:) =mean( YPCA(ind1(i):ind2(i),:),1);
    plot3(YPCA(ind1(i):ind2(i),1), YPCA(ind1(i):ind2(i),2),YPCA(ind1(i):ind2(i),3),
[colors(i) 'o'] )
    hold on
    if(i<= length(tars))
      st{i} = ['Target' num2str(tars(i))];
    else
      st{i} = 'Target Onset';
    end
  end
  legend(st);
```

```
  title('PCA');
  plot3(meani(:,1), meani(:,2), meani(:,3), '+k','LineWidth',3)
  hold off
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %Do normalized PCA
  clear dist Lap

  t= 1; %decay factor

  dist = 1./((distance(X',X')).^2);
 Lap = dist;

%Enforce Laplacian conditions
  for i=1:n
    Lap(i,i)=0;
    Lap(i,i) = -sum(Lap(i,:));
  end

  %solve using SVD
  SS = X'*Lap*X;
  [U,D,V2] = svd(SS);
  P = V2(:,1:3);
  PMn = P;
  Yn = X*P;

%plot

  figure(3);
  for i=1:max(size(ind1))
    meani(i,:) =mean( Yn(ind1(i):ind2(i),:),1);
    plot3(Yn(ind1(i):ind2(i),1), Yn(ind1(i):ind2(i),2),Yn(ind1(i):ind2(i),3),
[colors(i) 'o'] )
    hold on
    if(i<= length(tars))
      st{i} = ['Target' num2str(tars(i))];
    else
      st{i} = 'Target Onset';
    end
  end
  legend(st);
  title('Normalized PCA');
  plot3(meani(:,1), meani(:,2), meani(:,3), '+k','LineWidth',3)

  hold off
%do normalized plus binary-weighted PCA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
Lap = dist;
for i=1:max(size(ind1))
  Lap(ind1(i):ind2(i),ind1(i):ind2(i))= 0;
end
for i=1:n
  Lap(i,i)=0;
  Lap(i,i) = -sum(Lap(i,:));
end

SS = X'*Lap*X;
[U,D,V2] = svd(SS);
```

```
P = V2(:,1:3);
PMnw = P;
Yn = X*P;


figure(4);
meani = [];
for i=1:max(size(ind1))
  meani(i,:) =mean( Yn(ind1(i):ind2(i),:),1);
  plot3(Yn(ind1(i):ind2(i),1), Yn(ind1(i):ind2(i),2),Yn(ind1(i):ind2(i),3), [colors(i)
'o'] )
  hold on
  if(i<= length(tars))
    st{i} = ['Target' num2str(tars(i))];
  else
    st{i} = 'Target Onset';
  end
%  plot3(meani(1), meani(2), meani(3), '+k','LineWidth',3)
end
legend(st);
title('Normalized and supervised PCA');
  plot3(meani(:,1), meani(:,2), meani(:,3), '+k','LineWidth',3)
hold off


Tr = (datatrunc-mu)'*PMnw;
Ntrials = max(size(tBounds));
tBounds2 = tBounds(:,1:Ntrials);
Tr2 = Tr(1:tBounds2(end,end),:);
colort = color_data(:,1:tBounds2(end,end));

plotEmbedded(Tr2',tBounds2,colort);


%OPTION 3 %separate targets gradually with time
elseif(option==3)
  Lap =ones(n,n);

%figure out how many trials per target
p = 0;
for i=1: max(size(tars))
  ta(i) = sum((color_data(1,:)==i));
  targ(i) = find(tBounds(2,:)== ta(i)+p);
  p= p+ta(i);

end
tr = size(targ); %assume for now two targets


for i = 1:targ(1)

  %get trial i matrix

  tsi= tBounds(1,i);
  tei = tBounds(3,i);
  Mi = X(tsi:tei,:);
  ni = size(Mi,1);
```

```
%  Lap(tsi:tei,tsi:tei)= intT*ones(ni,ni);
  for j= (targ(1)+1):targ(2)
    tsj= tBounds(1,j);
    tej = tBounds(3,j);
    Mj = X(tsj:tej,:);
    nj = size(Mj,1);
    Mit = Mi;
    %Matrices must have same dimensions
    if(ni <nj)
      Mj = Mj(1:ni,:);
      nt = ni;
    else
      Mit = Mit(1:nj,:);
      nt= nj;
    end

    %compute distance between trials
    %dtr = 1000*(sum((abs(Mit-Mj).^2),2).^(1/2)).^(1.0);

    dtr = exp(.1*(1:nt));
    %dtr = 1000*(sum((abs(Mit-Mj).^2),2).^(1/2)).^(-1.0);
    %dtr = 1e4*ones(1,nt);
    mat = ones(nt,nt) - eye(nt) + diag(dtr);
    Lap(tsi:tsi+nt-1,tsj:tsj+nt-1)= mat;
    Lap(tsj:tsj+nt-1,tsi:tsi+nt-1)= mat;
  end
end


%Laplace conditions satisfied:
for i=1:n
  Lap(i,i)=0;
  Lap(i,i) = -sum(Lap(i,:));
end


SS = X'*Lap*X;
[U,D,V2] = svd(SS);
P = V2(:,1:3);
WPCA = P;
Y = P'*X';
% plotEmbedded(Y,tBounds,color_data);
Tr = (datatrunc-mu)'*P;
Ntrials = max(size(tBounds));
%  Ntrials = max(size(trials));
tBounds2 = tBounds(:,1:Ntrials);
Tr2 = Tr(1:tBounds2(end,end),:);
colort = color_trunc(:,1:tBounds2(end,end));

plotEmbedded(Tr2',tBounds2,colort);


end
```