Detecting Temporal Relations of Signals in a Visuomotor Task Using Support Vector Machines

EEL 6825  Pattern Recognition

Advisor: Prof. Clint Slatton, Prof. Mingzhou Ding, Prof. Onur Seref

Jiang Lu, Shuang Liu, Yuelu Liu
April 30, 2008
Abstract
In this project, we apply the support vector machine (SVM) classifiers to the problem of detecting the timing relations of stimulus- and response-related processes in a visuomotor task performed by a macaque monkey. The stages of the visuomotor task are i) the initial response of the visual cortex, ii) categorical discrimination of the visual stimuli and iii) appropriate response for the visual stimuli. We arrange the data in three possible combinations of classes: right vs. left, line vs. diamond, and go vs. no-go. We apply the SVM classifier to the recorded datasets and a 10-fold cross-validation is done at each time point to derive the classification accuracy. The time points at which the accuracy is over 60% are points when the local field potentials differ in two classes, and thus should be regarded the time interval for a certain stage. Our experimental results show that the SVM classifier can properly detect the time for each stage in the visuomotor task.

Introduction
A major goal of cognitive neuroscience is to understand how the different sensory and motor processes of the cerebral cortex are integrated to achieve goal-directed behavior. An effective method for the study of large-scale sensorimotor integration is the analysis of timing relations of stimulus- and response-related processes in distributed cortical areas. The spatiotemporal dynamics of the visual cortical processing and the interaction between the visual sensory cortex and motor cortex has been extensively studied in cognitive neurophysiology. In the past, the majority of these studies are on visual onset latency focusing on information processing within the visual system and ignoring the integration with the motor system. Hence, simultaneous recordings of cortical activity from visual, motor and executive cortical areas can provide essential information to explore the integration of the visual and motor processes on a large scale. However, the techniques used in the analysis of cortical activity usually involve first order statistical measures and linear statistical models. It is well known that the cortical activities and their interactions have a highly non-linear nature. Furthermore, the ubiquity of the activities in the nervous system and the highly connected structure of massive neural networks in the brain require methods that can incorporate simultaneous recordings in an appropriate multidimensional domain, rather than studying individual recordings independently. Hence, we choose SVM as our major tool to explore the temporal relations of different stimulus- and response-related processes in a visuomotor task.

Support vector machines (SVMs) have been one of the most recent research areas in machine learning. During the past seven years, SVMs have been applied very broadly within the field of computational biology [8]. The main motivation that suggests the use of SVMs in computational biology is that SVMs are known to behave well compared to other statistical or machine learning methods when the input data sets are noisy, highly non-linear, and high-dimensional. As I stated in the above paragraph, cortical activities and interactions are highly non-linear and noisy* in nature in the first place. Hence, we would say that simple and fast linear classifiers are not sufficient to do the classification task. In the second place, multiple electrodes are usually used to record signals from all parts of the brain to accurately record the activities in the nervous system, and this multidimensional data should be studied jointly to explore the internal correlations of the whole process. Since the complexity of

* By saying “noisy”, we mean that the signal part itself is highly random. This does not indicate that the observation noise power is high. Because the data is non-linear and non-Gaussian, we cannot simply apply a Kalman filter to do the denoising. A particle filter may be a good suboptimal choice here, but since we want to preserve the “ongoing activity” of the brain, we choose not to do the denoising in our project.
SVMs depends on the number of support vectors, rather than the dimensionality of the transformed space, we can feed data recorded by multiple channels to the SVMs without incurring huge amount of computation. A third characteristic of the SVM is its uniqueness in solution. Unlike standard neural network training, the convex quadratic programming problem in SVM training is guaranteed to find the global best separating hyperplane [2]. In fact, our experimental results show that SVM classifier works well in classifying different local field potentials (LFPs) corresponding to different stimuli and response types.

**Experimental Datasets**

The experiments to acquire the data used in our project took place in the Laboratory of Neuropsychology at the NIMH between 1984-1988. The data used in our project, as well as some previous studies were collected from an adult macaque monkey. The monkey was well trained to perform a go/no-go visuomotor pattern discrimination task before the electrodes were placed into its brain. Figure 1 shows the locations of the electrodes used in the study. The local field potential (LFP) data was collected during a number of sessions. Each session consists of 1000 trials on average. The data were band-pass filtered between 1 and 100 Hz and digitized at 200 Hz. For each trial, data were recorded for 900 ms after the monkey initiated the trial and truncated to have 90 ms prior to stimulus onset and 510 ms after stimulus onset.

The visual stimuli were created via lines and diamonds using eight squares as shown in Figure 2. The stimuli were referred to as right slanted line and right slanted diamond in the first row, and left slanted line and left slanted diamond in the second row. For each session, the go stimulus was chosen to be either both lines or both diamonds, and never any other possible combination of the four stimulus types. Note that, when the two line stimuli and the two diamond stimuli are superimposed onto each other separately, the resulting two shapes are identical. Therefore, the monkey has to distinguish a line from a diamond by correctly realizing the placement of at least two squares.

The trials were initiated by monkeys’ manually pressing a lever and keeping it pressed. The initiation was followed by a random amount of time, uniformly distributed between 120 and 2200 ms, before the stimulus appears on the screen for exactly 100 ms. The monkey was expected to respond to a stimulus within the 500 ms following the stimulus onset. The release of the lever was the correct go response, and keeping it pressed was the correct no-go response. The correct go responses were rewarded with a small amount of water, whereas the correct no-go responses were not rewarded. In our study, only correct go and correct no-go trials are considered. For more details on data, please refer to [5]

**Theories and Methods**

**Support Vector Machine Classifiers**

Considering the multi-dimensional nature of the problem and the nonlinear behavior of the brain, SVM classification is used on each time point to detect the temporal relationships of the onset of LFPs. In standard SVM formulation, the separating hyperplane maximizes the margin between the two classes. Consider a data set composed of \( n \) pattern vectors \( x_i \in \mathbb{R}^d \) with class labels \( y_i \in \{-1,1\} \). Then, the solution to the following quadratic linearly constrained problem is the optimal hyperplane
with the maximum margin.

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2$$  \hfill (1a)

subject to \( y_i((w \cdot x_i) + b) \geq 1 - \xi_i \quad i = 1, \ldots, n \)  \hfill (1b)

In practice, we often solve the dual representation of the above quadratic programming problem,

$$\max \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$  \hfill (2a)

subject to \( \sum_{i=1}^{n} \alpha_i y_i = 0 \)  \hfill (2b)

$$0 \leq \alpha_i \leq C$$  \hfill (2c)

From the solution of (2), the classification function for new points can be written as

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i (x_i \cdot x)$$  \hfill (3)

A new point is classified to the positive class if \( f(x) > 0 \) and to the negative class if \( f(x) < 0 \). The relationship between features and class labels can be nonlinear. In SVM, an embedding of the data to a higher dimensional space where a linear separation of data is possible is done implicitly by kernel functions, \( K(x_i, x_j) \). Replacing the inner product by the kernel function everywhere in the training and testing algorithm, the algorithm will happily produce a support vector machine which lives in a feature space within which a linear separation is achievable. We use the Gaussian Kernel in our project, which is the most commonly used kernel.

$$K(x_i, x_j) = \exp \left( -\frac{\|x_i - x_j\|^2}{\sigma} \right)$$  \hfill (4)

**Our Experimental Methods**

The visuomotor process studied in our project is naturally hypothesized to include three stages according to their temporal relationship: i) initial response of the visual cortex, ii) discrimination of the visual stimuli and iii) the appropriate motor response. During the first stage, we should expect certain regions of the brain act differently to a left slanted stimulus and a right slanted stimulus. In stage two, the macaque monkey has to first discriminate the categorical difference between the stimulus, and then make a correct go or nogo decision accordingly. In the third stage, a go and nogo decision will induce different responses in the motor cortex to conduct the proper go and nogo movements. In this project, we apply the SVM algorithm to the recorded data sets at each time point and estimate the onset time point of the LFPs for the above mentioned three stages in the neocortex of a macaque monkey’s brain during the visuomotor task. This onset time estimation is based on the SVM’s output accuracy at each time point over the entire time span of the recordings. For instance, if at a certain time point, certain regions of the brain react differently for some reason, say left vs. right slanted stimuli, when the data points at this very time point are used to do the cross validation by the SVM, we should expect the classification accuracy rate is relatively high at this point, and vice versa.

In order to create suitable training datasets, we separate the original datasets into 3 combinations each containing two different classes (see Table 1). We pick out trials coming from sessions \( i = 1, 2 \) with right line, right diamond, left line, and left diamond and mark them as \( s_{11}^1, s_{12}^1, s_{21}^2, \) and \( s_{22}^2 \), respectively. Then we form three different sets of data with respect to our interest out of the above
four types of trials. The possible combinations are:

1) RIGHT vs. LEFT
   \[ S^+ = s_1^1 \cup s_2^1 \cup s_3^1 \cup s_4^2 \text{ vs. } S^- = s_1^1 \cup s_2^1 \cup s_3^2 \cup s_4^2 \]

2) LINE vs. DIAMOND
   \[ S^+ = s_1^1 \cup s_2^1 \cup s_3^1 \cup s_4^2 \text{ vs. } S^- = s_1^2 \cup s_2^1 \cup s_3^2 \cup s_4^2 \]

3) GO vs. NO-GO
   \[ S^+ = s_1^1 \cup s_2^1 \cup s_3^1 \cup s_4^1 \text{ vs. } S^- = s_1^2 \cup s_2^1 \cup s_3^1 \cup s_4^1 \]

Note that the above sets of data must come from two sessions within which a line pattern corresponds respectively to a go and no-go response. For example, trials marked as \( s_1^1 \) come from a session in which line pattern corresponds to a go response and trials marked as \( s_2^1 \) come from another session in which line pattern corresponds to a no-go response. The reason for doing this is to make sure that while we are testing the LFP differences corresponding to line vs. diamond, we want to cancel out the go and no-go patterns’ contribution to classification accuracy, and vice versa. This is very important in conducting our experiments.

Actually, in our experiments, we used 10 sessions of recordings each containing roughly 500 correct go and no-go trials to form the above mentioned training and testing sets. Among these 10 sessions, in five sessions, line stimuli are a go cue whereas in the other five sessions, diamond stimuli are a go cue. For all 10 selected sessions, 14 electrodes are implanted into the macaque monkey’s cortex, thus the recorded data is 14-dimensional in our experiments. At each time point \( t \), recordings from \( n \) trials containing one possible combination of two class shown in Table 1 and 14 channels constitute the \( n \times 14 \) data matrix. We then employ 10-fold cross validation to obtain the classification accuracy for the data matrix at each time point. An accuracy percentage around 50% is equivalent to pure chance and hence we should expect to detect at least 60% or more to conclude any separation between the classes listed in Table 1.

**Experimental Results and Analysis**
A difficult issue when applying the SVM algorithm to do the classification is to choose its parameters. In our experiments, a Gaussian RBF kernel is enough to classify data well at certain time points, thus we do not need to design any new kernel functions tailored to solve the problem. However, we still need to decide two parameters: the penalty \( C \) and the width of the Gaussian kernel \( \sigma \). In [6], Hsu et al. suggests a grid-search on \( C \) and \( \sigma \) using cross-validation. However, although this method is relatively powerful in finding best \((C, \sigma)\) pairs, it is very time consuming. Upon our observation, the SVM’s classification accuracy increases monotonically with \( C \) and when \( C \) is big, the accuracy remains roughly unchanged. Thus we let \( C = 100 \), and only choose \( \sigma \) to maximize the classification accuracy. We first let \( \sigma = 2^{-6}, 2^{-5}, \ldots, 2^{3} \), and choose the one that maximize the classification accuracy. Then we conduct a finer search around the chosen \( \sigma \) value to find the optimal value. In our experiment, we chose \( \sigma = 0.1 \) when detecting the onset time of LFPs in stage one and for the rest two stages, we chose \( \sigma = 0.5 \).

The original datasets are normalized to the range of \([-1, +1]\). The normalization is done only within each session, that is, we divide our data from all trials within one session by the maximal value of that session. The reason for doing so is that the monkey only perform one session of the task every day, and after each session, the electrodes are removed from the brain. Thus normalizing across multiple
sessions might cause different sessions to contribute differently to the classification and cause bias to the classification accuracy. Figure 3 shows the recorded signals for every channel of all trials in one session. The signals shown in the figures are the original ones without normalization, and we can see that they are generally very “noisy” because of the ongoing activities of the brain. However, these ongoing activities are also information bearing signals, thus we cannot “filter” them out in our experiments.

Figure 4 shows the SVM classification accuracy over time for datasets combined in the “Right vs. Left” version. The SVM output accuracy curve is smoothed by a 3-point moving average filter. The zero time point shown on the figure is the time when the stimuli are shown on the screen. Judging from the figure, the highest rate of classification achieved at 125 ms after the stimulus shown on the screen. The time point at which the accuracy rate surpasses 60% for the first time is roughly 103 ms, and at about 90 ms, the accuracy starts to increase dramatically. This suggests that the brain’s earliest responses to different visual stimuli probably happen at about 90 ms. Then at 103 ms after the stimulus, the difference in response patterns in the prestriate and striate cortexes corresponding to left and right slanted stimuli is significant enough for the SVM to generate an classification accuracy above 60%. The width of the first accuracy peak is approximately 52ms (the time span during which the accuracy is greater than 60%). This means that the first stage should last no shorter than 52ms. An interesting thing in Figure 3 is that the second highest peak in accuracy occurs at around 300 ms, which might indicate the difference in LFPs measured in the motor cortex corresponding to go and no-go responses. This is probably because of the imbalance in the number of go and no-go trials across all 10 sessions.

Figure 5 shows the SVM classification accuracy over time for datasets combined in the “Go vs. No-Go” version. The peak in accuracy rate is achieved at 345 ms with 75.59%. Since the overall accuracy in this experiment is relatively high (the lowest is about 58%), we choose 68%, 10% higher than the lowest, as our threshold for determining whether a significant discrimination is recorded by the electrodes. At about 240 ms, the accuracy increases dramatically and at about 300 ms, the accuracy reaches 68%.

Figure 6 shows the SVM classification accuracy over time for datasets combined in the “Line vs. Diamond” version. Unlike the first two cases, categorical discrimination requires higher level cortical activity and is more complicated than the previous cases. Generally speaking, 3 peaks can be seen from the plot at 130, 175, and 275 ms, respectively. The first two peaks at 130 ms and 175 ms are agreeing with the fact that “both inferior temporal sites also had early stimulus-related differences in LU and the differences persisted for at least 100ms” described by Ledberg et al. [4]. The third high peak at 275 ms is probably correlated to the difference in making a go or no-go decision according to “Line” and “Diamond” stimuli.

Figure 7 shows the unfiltered accuracy rate plot for “Line vs. Diamond” combination over time. We see that the second peak is slightly shifted ahead to 155 ms from 175 ms in the previous figure. But still we can see that the accuracy remains relatively high within the interval of 100-190 ms and then the accuracy rise high again from approximately 220 to 275 ms.
Conclusions and Future Works
From the experimental results, the SVM successfully detected the time intervals for the three stages in the visuomotor task. For stage one the estimated time span is 103ms to 155 ms after the stimulus. The second stage lasts relatively long, and includes two sub-stages: the first sub-stage starts at about 100 ms and ends at 190 ms; the second sub-stage starts at about 220 ms and ends at about 300 ms. And the third stage starts at about 270 ms and last till 510 ms.

For future works, we should apply a feature selection process over the recorded signals and to see at which time points, which channels contribute more to the classification. We could weight each channel according to its contribution to classification over time to explore the spatial relations among the signals across the brain. A general guideline of the spatial and temporal relations of activities in the entire nervous system can thus be provided automatically by SVM.

References

Tables and Figures
Table 1 Stages of the Visuomotor Task and Their Corresponding Training Sets

<table>
<thead>
<tr>
<th>Stage</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial response of the visual cortex</td>
<td>Left vs. Right stimulus</td>
</tr>
<tr>
<td>Discrimination of the visual stimuli</td>
<td>Line vs. Diamond stimulus</td>
</tr>
<tr>
<td>The appropriate motor response</td>
<td>Go vs. No-go stimulus</td>
</tr>
</tbody>
</table>
Figure 1. Approximate placement of electrodes in the monkey’s brain

Figure 2. Visual Stimuli used in the visuomotor task. In the actual experiments, only the black squares were shown. Here the outlines of the squares are shown for comparison.
Figure 3. The recorded signals for every channel of all trials in one session. The signals shown in the figures are the original ones without normalization, and we can see that they are generally very “noisy” because of the ongoing activities of the brain.
Figure 4. The SVM classification accuracy over time for datasets combined in the “Right vs. Left” version. The SVM output accuracy curve is filtered by a 3 point moving average filter to generate a smoother look. The parameters for the SVM are: C=10, Gaussian Radial Basis Function Kernel, Sigma=10 (Gamma=0.1). Highest rate of classification achieved at 125 ms after the stimulus shown on the screen. The time point at which the accuracy rate surpasses 60% is roughly 103 ms. The width of the first peak is approximately 52 ms (the time span in which the accuracy is greater than 60%).

Figure 5. The SVM classification accuracy over time for datasets combined in the “Go vs. No-Go” version. The output accuracy vector is filtered by a 3-tap moving average filter to make less spiky-looking. The parameters for the SVM used in this experiment are listed as follows: C=100, Gaussian Radial Basis Function Kernel, Sigma=2 (Gamma=0.5). The peak in classification accuracy is achieved at 345 ms with 75.59%. Since the overall accuracy is relatively high in this experiment (the lowest is about 58%), we choose 68%, 10% higher than the lowest, as our threshold for determining whether a significant discrimination is recorded by the channels. Hence, the ERPs for “go” and “nogo” responses are approximately at 300 ms measured from the figure, which is in accordance to previous studies.
Figure 6. The SVM classification accuracy over time for datasets combined in the “Line vs. Diamond” version. The parameters chosen here for the SVM are: $C=100$, and $\gamma=0.5$ ($\sigma=2$), Gaussian Kernel. Generally speaking, 3 peaks can be shown from the plot within time interval 0 to 300 msec at 130, 175, and 275 msec, respectively. The first two peaks at 130 msec and 175 msec are agreeing with the fact that “both inferior temporal sites also had early stimulus-related differences in LU and the differences persisted for at least 100 msec” shown in the paper “Large-Scale Visuomotor Integration in the Cerebral Cortex” by Ledberg et al. The third high peak at 275 msec is probably correlated to the difference in making a decision according to “Line” and “Diamond” stimuli.

Figure 7. The unfiltered accuracy rate plot for “Line vs. Diamond” combination over time. We see that the second peak is slightly shifted ahead to 155 ms from 175 ms in the previous figure. But still we can see that the accuracy remains relatively high within the interval of 100-190 ms and then the accuracy rise high again from approximately 220 to 275 ms.
Appendix: Matlab Code for the Experiments  
A. Detecting time course of stage 1.

```matlab
% Read in all data from 10 sessions, separate them according to right and left slanted stimuli types
clear all;
load lu62;
RightData62=cat(1,D.data(1).M,D.data(2).M);
LeftData62=cat(1,D.data(3).M,D.data(4).M);
data62=cat(1,RightData62,LeftData62);
max62=max(max(max(abs(data62))));
RightData62=RightData62/max62;
LeftData62=LeftData62/max62;
load lu63;
RightData63=cat(1,D.data(1).M,D.data(2).M);
LeftData63=cat(1,D.data(3).M,D.data(4).M);
data63=cat(1,RightData63,LeftData63);
max63=max(max(abs(data63)));
RightData63=RightData63/max63;
LeftData63=LeftData63/max63;
load lu64;
RightData64=cat(1,D.data(1).M,D.data(2).M);
LeftData64=cat(1,D.data(3).M,D.data(4).M);
data64=cat(1,RightData64,LeftData64);
max64=max(abs(data64));
RightData64=RightData64/max64;
LeftData64=LeftData64/max64;
load lu65;
RightData65=cat(1,D.data(1).M,D.data(2).M);
LeftData65=cat(1,D.data(3).M,D.data(4).M);
data65=cat(1,RightData65,LeftData65);
max65=max(abs(data65));
RightData65=RightData65/max65;
LeftData65=LeftData65/max65;
load lu66;
RightData66=cat(1,D.data(1).M,D.data(2).M);
LeftData66=cat(1,D.data(3).M,D.data(4).M);
data66=cat(1,RightData66,LeftData66);
max66=max(abs(data66));
RightData66=RightData66/max66;
LeftData66=LeftData66/max66;
load lu72;
RightData72=cat(1,D.data(1).M,D.data(2).M);
LeftData72=cat(1,D.data(3).M,D.data(4).M);
data72=cat(1,RightData72,LeftData72);
```

max72 = max(max(max(abs(data72))));
RightData72 = RightData72/max72;
LeftData72 = LeftData72/max72;
load lu73
RightData73 = cat(1,D.data(1).M,D.data(2).M);
LeftData73 = cat(1,D.data(3).M,D.data(4).M);
data73 = cat(1,RightData73,LeftData73);
max73 = max(max(max(abs(data73))));
RightData73 = RightData73/max73;
LeftData73 = LeftData73/max73;
load lu74
RightData74 = cat(1,D.data(1).M,D.data(2).M);
LeftData74 = cat(1,D.data(3).M,D.data(4).M);
data74 = cat(1,RightData74,LeftData74);
max74 = max(max(max(abs(data74))));
RightData74 = RightData74/max74;
LeftData74 = LeftData74/max74;
load lu75
RightData75 = cat(1,D.data(1).M,D.data(2).M);
LeftData75 = cat(1,D.data(3).M,D.data(4).M);
data75 = cat(1,RightData75,LeftData75);
max75 = max(max(max(abs(data75))));
RightData75 = RightData75/max75;
LeftData75 = LeftData75/max75;
load lu76
RightData76 = cat(1,D.data(1).M,D.data(2).M);
LeftData76 = cat(1,D.data(3).M,D.data(4).M);
data76 = cat(1,RightData76,LeftData76);
max76 = max(max(max(abs(data76))));
RightData76 = RightData76/max76;
LeftData76 = LeftData76/max76;
RightData = cat(1,RightData62,RightData63,RightData64,RightData65,...
               RightData66,RightData72,RightData73,RightData74,RightData75,RightData76);
LeftData = cat(1,LeftData62,LeftData63,LeftData64,LeftData65,LeftData66,...
              LeftData72,LeftData73,LeftData74,LeftData75,LeftData76);

RightData = cat(1,RightData62,RightData72);
LeftData = cat(1,LeftData62,LeftData72);
[ntrl1,nchnl1,junk]=size(RightData);
[ntrl2,nchnl2,junk]=size(LeftData);
RL_Data = cat(1,RightData,LeftData);
[ntrl,nchnl,junk]=size(RL_Data);
label = ones(1,ntrl);
label(ntrl1+1:ntrl1)=1;
accu_rate=zeros(1,102);
accu_rate1=zeros(1,102);
index=randperm(ntrl);
L=length(index);
testPercent=10;
tic;
overall_accu=[];
for tp=1:17
    temp_Data=RL_Data(:,:,tp);
    actlTstLbl_total=[];
    cmptdLbl_total=[];
    for i=1:10
        ind_test=index(((i-1)*fix(testPercent/100*L)+1):i*fix(testPercent/100*L));
        %ind_train=index(fix(testPercent/100*length(index)):
        %    length(index));
        index1=index(((i*fix(testPercent/100*L)+1):L));
        if i==1
            ind_train=index1;
        else if i<=9
            ind_train=horzcat(index(1:(i-1)*fix(testPercent/100*L)),index1);
        else
            ind_train=index(1:(i-1)*fix(testPercent/100*L));
        end
    end
    testData=temp_Data(ind_test,:);
    actlTstLbl=label(ind_test);
    trainData=temp_Data(ind_train,:);
lblTrain=label(ind_train);
%testData=testData';
%trainData=trainData';
actlTstLbl=actlTstLbl';
lblTrain=lblTrain';
actlTstLbl_total=[actlTstLbl_total;actlTstLbl];
actlTstLbl_total=[actlTstLbl_total;actlTstLbl];
model=svmtrain(lblTrain,trainData,'-c 10 -g 0.1');
[cmptdLbl,accuracy,prob]=svmpredict(actlTstLbl,testData,model);
overall_accu(i)=accuracy(1,1);
    cmptdLbl_total=[cmptdLbl_total;cmptdLbl];
end
accu_rate(tp)=sum(overall_accu)/10;
count=0;
for i=1:L
    if cmptdLbl_total(i)==actlTstLbl_total(i)
count=count+1;
end
end
accu_rate1(tp)=count/L;
end
toc;

B. Detecting time course for stage 3.

%Read in all data from 10 sessions, separate them according to GO and NOGO response types

clear all;
load lu62;
GO62=cat(1,D.data(2).M,D.data(4).M);
NOGO62=cat(1,D.data(1).M,D.data(3).M);
data62=cat(1,GO62,NOGO62);
max62=max(max(max(abs(data62))));
GO62=GO62/max62;
NOGO62=NOGO62/max62;
load lu63;
GO63=cat(1,D.data(2).M,D.data(4).M);
NOGO63=cat(1,D.data(1).M,D.data(3).M);
data63=cat(1,GO63,NOGO63);
max63=max(max(max(abs(data63))));
GO63=GO63/max63;
NOGO63=NOGO63/max63;
load lu64;
GO64=cat(1,D.data(2).M,D.data(4).M);
NOGO64=cat(1,D.data(1).M,D.data(3).M);
data64=cat(1,GO64,NOGO64);
max64=max(max(max(abs(data64))));
GO64=GO64/max64;
NOGO64=NOGO64/max64;
load lu65;
GO65=cat(1,D.data(2).M,D.data(4).M);
NOGO65=cat(1,D.data(1).M,D.data(3).M);
data65=cat(1,GO65,NOGO65);
max65=max(max(max(abs(data65))));
GO65=GO65/max65;
NOGO65=NOGO65/max65;
load lu66;
GO66=cat(1,D.data(2).M,D.data(4).M);
NOGO66=cat(1,D.data(1).M,D.data(3).M);
data66=cat(1,GO66,NOGO66);
max66=max(max(max(abs(data66))));
GO66 = GO66 / max66;
NOGO66 = NOGO66 / max66;
load lu72
GO72 = cat(1, D.data(1).M, D.data(3).M);
NOGO72 = cat(1, D.data(2).M, D.data(4).M);
data72 = cat(1, GO72, NOGO72);
max72 = max(max(abs(data72)));
GO72 = GO72 / max72;
NOGO72 = NOGO72 / max72;
load lu73
GO73 = cat(1, D.data(1).M, D.data(3).M);
NOGO73 = cat(1, D.data(2).M, D.data(4).M);
data73 = cat(1, GO73, NOGO73);
max73 = max(max(abs(data73)));
GO73 = GO73 / max73;
NOGO73 = NOGO73 / max73;
load lu74
GO74 = cat(1, D.data(1).M, D.data(3).M);
NOGO74 = cat(1, D.data(2).M, D.data(4).M);
data74 = cat(1, GO74, NOGO74);
max74 = max(max(abs(data74)));
GO74 = GO74 / max74;
NOGO74 = NOGO74 / max74;
load lu75
GO75 = cat(1, D.data(1).M, D.data(3).M);
NOGO75 = cat(1, D.data(2).M, D.data(4).M);
data75 = cat(1, GO75, NOGO75);
max75 = max(max(abs(data75)));
GO75 = GO75 / max75;
NOGO75 = NOGO75 / max75;
load lu76
GO76 = cat(1, D.data(1).M, D.data(3).M);
NOGO76 = cat(1, D.data(2).M, D.data(4).M);
data76 = cat(1, GO76, NOGO76);
max76 = max(max(abs(data76)));
GO76 = GO76 / max76;
NOGO76 = NOGO76 / max76;
GO_Data = cat(1, GO62, GO63, GO64, GO65, GO66, GO72, GO73, GO74, GO75, GO76);
NOGO_Data = cat(1, NOGO62, NOGO63, NOGO64, NOGO65, NOGO66, NOGO72, NOGO73, NOGO74, NOGO75, NOGO76);

%GO_Data = cat(1, GO62, GO72);
%NOGO_Data = cat(1, NOGO62, NOGO72);
[ntrl1,nchnl1,junk]=size(GO_Data);
[ntrl2,nchnl2,junk]=size(NOGO_Data);
GNG_Data=cat(1,GO_Data,NOGO_Data);
[ntrl,nchnl,junk]=size(GNG_Data);
label=ones(1,ntrl);
label(ntrl1+1:ntrl)=-1;
accu_rate=zeros(1,120);
accu_rate1=zeros(1,120);
index=randperm(ntrl);
L=length(index);
testPercent=10;
tic;
overall_accu=[];
for tp=1:120
    temp_Data=GNG_Data(:,:,tp);
    actlTstLbl_total=[];
    cmptdLbl_total=[];
    for i=1:10
        ind_test=index(((i-1)*fix(testPercent/100*L)+1):i*fix(testPercent/100*L));
        %ind_train=index(fix(testPercent/100*length(index)):
        %    length(index));
        index1=index(((i*fix(testPercent/100*L)+1):L));
        if i==1
            ind_train=index1;
        else if i<=9
            ind_train=horzcat(index(1:(i-1)*fix(testPercent/100*L)),index1);
        else
            ind_train=index(1:(i-1)*fix(testPercent/100*L));
        end
        testData=temp_Data(ind_test,:);
        actlTstLbl=label(ind_test);
        trainData=temp_Data(ind_train,:);
        lblTrain=label(ind_train);
        testData=testData';
        trainData=trainData';
        lblTrain=lblTrain';
        actlTstLbl=actlTstLbl';
        actlTstLbl_total=[actlTstLbl_total;actlTstLbl];
        model=svmtrain(lblTrain,trainData,'-c 100 -g 0.5');
        [cmptdLbl,accuracy,prob]=svmpredict(actlTstLbl,testData,model);
        overall_accu=overall_accu+accuracy;
        fprintf(1,'overall_accu= %f
',overall_accu);
    end
end
...
overall_accu(i)=accuracy(1,1);
cmptdLbl_total=[cmptdLbl_total;cmptdLbl];
end
accu_rate(tp)=sum(overall_accu)/10;
count=0;
for i=1:L
    if cmptdLbl_total(i)==actlTstLbl_total(i)
        count=count+1;
    end
end
accu_rate1(tp)=count/L;
end
toc;

C. Detecting time course for stage 2.
clear all;
load lu62;
Line62=cat(1,D.data(1).M,D.data(3).M);
Diam62=cat(1,D.data(2).M,D.data(4).M);
data62=cat(1,Line62,Diam62);
max62=max(max(max(abs(data62))));
Line62=Line62/max62;
Diam62=Diam62/max62;
load lu63;
Line63=cat(1,D.data(1).M,D.data(3).M);
Diam63=cat(1,D.data(2).M,D.data(4).M);
data63=cat(1,Line63,Diam63);
max63=max(max(max(abs(data63))));
Line63=Line63/max63;
Diam63=Diam63/max63;
load lu64;
Line64=cat(1,D.data(1).M,D.data(3).M);
Diam64=cat(1,D.data(2).M,D.data(4).M);
data64=cat(1,Line64,Diam64);
max64=max(max(max(abs(data64))));
Line64=Line64/max64;
Diam64=Diam64/max64;
load lu65;
Line65=cat(1,D.data(1).M,D.data(3).M);
Diam65=cat(1,D.data(2).M,D.data(4).M);
data65=cat(1,Line65,Diam65);
max65=max(max(max(abs(data65))));
Line65=Line65/max65;
Diam65=Diam65/max65;
load lu66;
Line66=cat(1,D.data(1).M,D.data(3).M);
Diam66=cat(1,D.data(2).M,D.data(4).M);
data66=cat(1,Line66,Diam66);
max66=max(max(max(abs(data66))));
Line66=Line66/max66;
Diam66=Diam66/max66;
load lu72;
Line72=cat(1,D.data(1).M,D.data(3).M);
Diam72=cat(1,D.data(2).M,D.data(4).M);
data72=cat(1,Line72,Diam72);
max72=max(max(max(abs(data72))));
Line72=Line72/max72;
Diam72=Diam72/max72;
load lu73;
Line73=cat(1,D.data(1).M,D.data(3).M);
Diam73=cat(1,D.data(2).M,D.data(4).M);
data73=cat(1,Line73,Diam73);
max73=max(max(max(abs(data73))));
Line73=Line73/max73;
Diam73=Diam73/max73;
load lu74;
Line74=cat(1,D.data(1).M,D.data(3).M);
Diam74=cat(1,D.data(2).M,D.data(4).M);
data74=cat(1,Line74,Diam74);
max74=max(max(max(abs(data74))));
Line74=Line74/max74;
Diam74=Diam74/max74;
load lu75;
Line75=cat(1,D.data(1).M,D.data(3).M);
Diam75=cat(1,D.data(2).M,D.data(4).M);
data75=cat(1,Line75,Diam75);
max75=max(max(max(abs(data75))));
Line75=Line75/max75;
Diam75=Diam75/max75;
load lu76;
Line76=cat(1,D.data(1).M,D.data(3).M);
Diam76=cat(1,D.data(2).M,D.data(4).M);
data76=cat(1,Line76,Diam76);
max76=max(max(max(abs(data76))));
Line76=Line76/max76;
Diam76=Diam76/max76;
Line_Data=cat(1,Line62,Line63,Line64,Line65,Line66,...
   Line72,Line73,Line74,Line75,Line76);
Diam_Data=cat(1,Diam62,Diam63,Diam64,Diam65,Diam66,...  
    Diam72,Diam73,Diam74,Diam75,Diam76);

%Line_Data=cat(1,Line62,Line72);
%Diam_Data=cat(1,Diam62,Diam72);
[ntrl1,nchnl1,junk]=size(Line_Data);
[ntrl2,nchnl2,junk]=size(Diam_Data);
LDM_Data=cat(1,Line_Data,Diam_Data);
[ntrl,nchnl,junk]=size(LDM_Data);
label=ones(1,ntrl);
label(ntrl1+1:ntrl)=-1;
accu_rate=zeros(1,120);
accu_rate1=zeros(1,120);
index=randperm(ntrl);
L=length(index);
testPercent=10;
tic;
overall_accu=[];
for tp=1:120
    temp_Data=LDM_Data(:,:,tp);
    actlTstLbl_total=[];
    cmptdLbl_total=[];
    for i=1:10
        ind_test=index(((i-1)*fix(testPercent/100*L)+1):i*fix(testPercent/100*L));
        ind_train=index((fix(testPercent/100*length(index))):...
            length(index));
        index1=index(((i*fix(testPercent/100*L)+1):L));
        if i==1
            ind_train=index1;
        else if i<=9
            ind_train=horzcat(index(1:(i-1)*fix(testPercent/100*L)),index1);
        else
            ind_train=index(1:(i-1)*fix(testPercent/100*L)));
        end
        testData=temp_Data(ind_test,:);
        actlTstLbl=label(ind_test);
        trainData=temp_Data(ind_train,:);
        lblTrain=label(ind_train);
        testData=testData';
        %trainData= trainData';
        accu_rate(i,1)=sum(testData==actlTstLbl)/length(actlTstLbl);
        %accu_rate1(i,1)=length(lblTrain)/length(trainData);
    end
end
overall_accu
model = svmtrain(lblTrain, trainData, '-c 100 -g 0.5');
[cmptdLbl, accuracy, prob] = svmpredict(actlTstLbl, testData, model);
overall_accu(i) = accuracy(1,1);
cmptdLbl_total = [cmptdLbl_total; cmptdLbl];
end
accu_rate(tp) = sum(overall_accu) / 10;
count = 0;
for i = 1:L-5
    if cmptdLbl_total(i) == actlTstLbl_total(i)
        count = count + 1;
    end
end
accu_rate1(tp) = count / L;
end
toc;