Table of Contents

HW 3: D' and population tuning curves

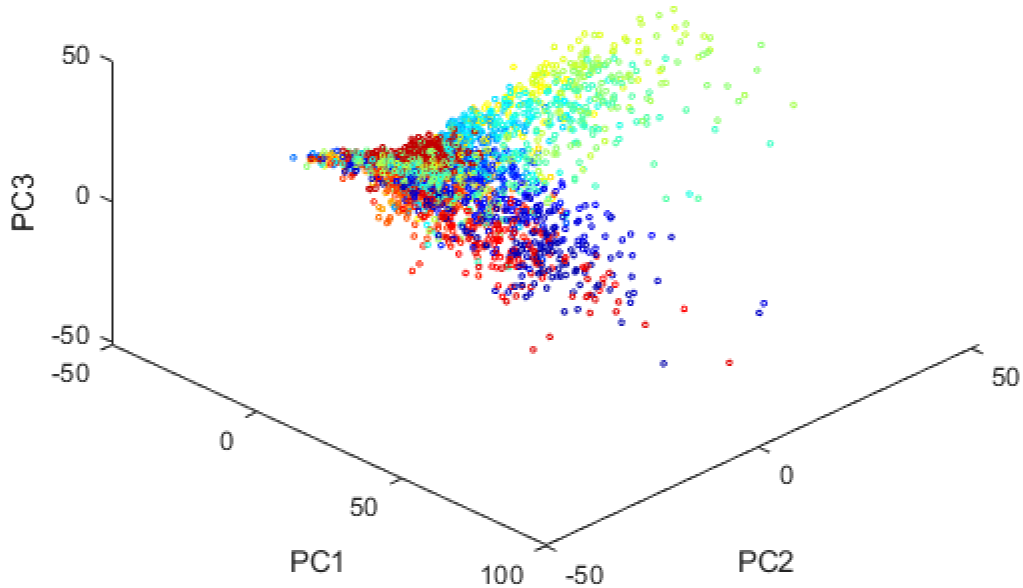
```
clc  %clears all
clear all
close all
tic
load('ori32_M160825_MP027_2016-12-15')

data = stim.resp;
stimlist = stim.istim;

stimuli = unique(stimlist); % istim= stimulus identity of each data,
 returns all unique stimuli
number = length(stimuli); %states how many unique stimuli there are
neurons = size(stim.resp,2); % how many  neurons are there?
```

Step 1: Perform PCA on neural response data

[COEFF, SCORE, LATENT] = pca(X); <-- here X is the full dataset (all 11000 neurons) SCORE is the T matrix from lecture: the data in PCA space

```
% a. perform PCA on data, keep only first 3 PCs
zsc = zscore(data); %Zscores the data

[COEFF, SCORE, LATENT] = pca(zsc); %performs PCA on the z scored data
PCs = SCORE(:,1:3); %creates a matrix of just the first three PCAs'


% b. plot single-trial response data in 3-D PCA space
% grab separate colors for each stimulus:
plotcolors = colormap(jet(number)); % assigns a color to numbers 1-33
 that will be used to color each stimulus
len = length(stimlist); %finds the length of the stimuli

figure(1), hold on
%plots each PCs but each stimulus direction is a different color
for t = 1:len
    %plots all three PC's with respective color markers to each
 stimuli

 plot3(SCORE(t,1),SCORE(t,2),SCORE(t,3),'color',plotcolors(stimlist(t),:),'marker'
end
```

```
view(45,45) %changes view to appear in 3 dimensions
xlabel('PC1')
ylabel('PC2')
zlabel('PC3')
```



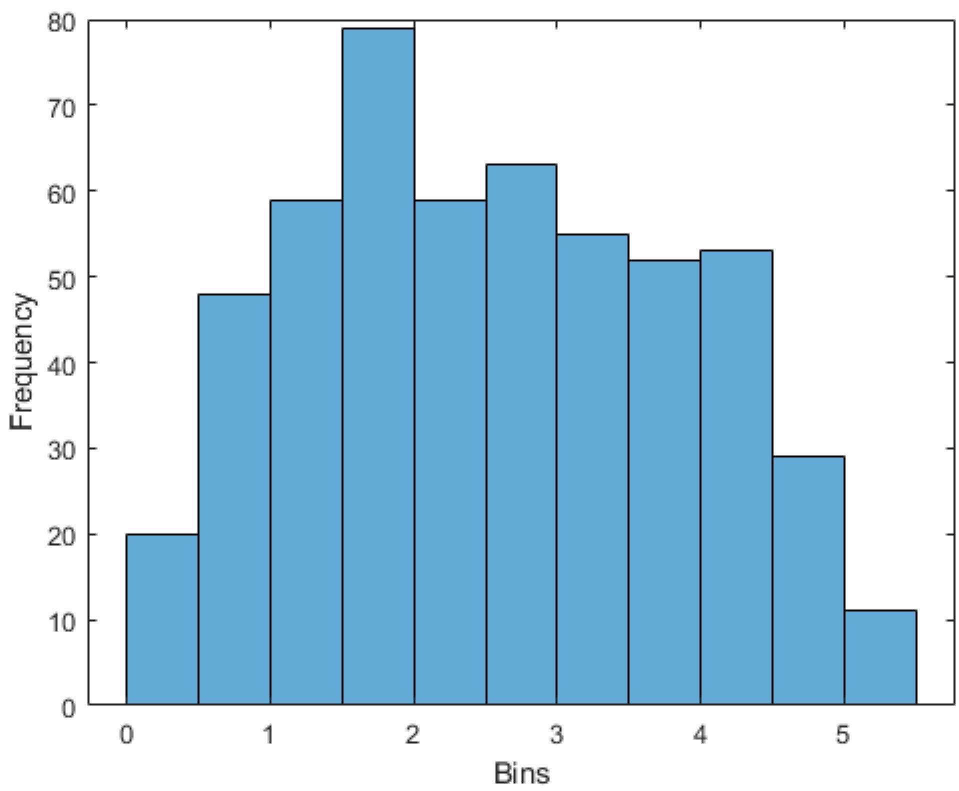
Step 2. Compute d' values for all pairs of clusters

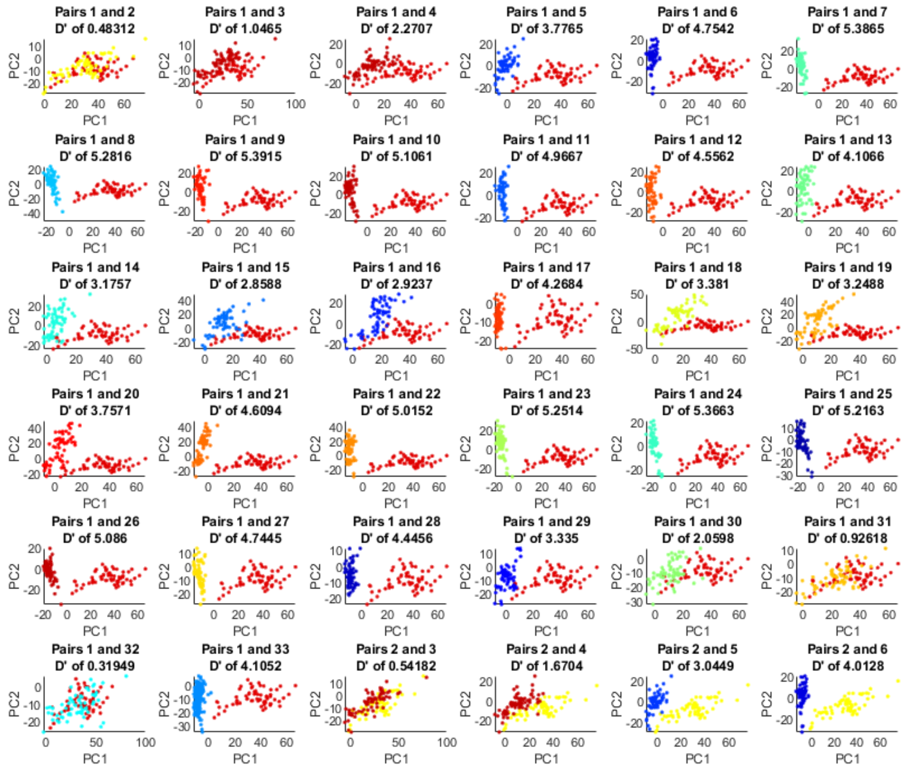
(Each cluster is composed of all of the responses to a single stimulus, e.g. stimulus 1

```
C = nchoosek([1:number],2); % get all possible combinations of stimuli
 orientations
ncomb = size(C,1); %number of combinations
Dprime = NaN(ncomb,1); %initializes d'

% a. Compute D' values for each pair

for k = 1:ncomb

    x = C(k,1); %first number in the pair of combinations
    y = C(k,2); % second number in the pair of combinations
    C1 = find(stimlist == x);%finds all trials where stimulus is equal
 to k orientation in the first column of C
    C2 = find(stimlist == y);%finds all trials where stimulus is equal
 to k orientation in the second column of C
```

```
    u1 = [mean(SCORE(C1,1)), mean(SCORE(C1,2)),
 mean(SCORE(C1,3))]; %finds averages  for PC1, PC2, and PC3 at C1
    u2 = [mean(SCORE(C2,1)), mean(SCORE(C2,2)),
 mean(SCORE(C2,3))]; %finds averages for PC1, PC2, and PC3 at C2
    %good lord I forgot matrix math and dot products
    %[2,1]*{2*1] = 2*2+1*1
    Vvector = [(u2(1,1) - u1(1,1)),(u2(1,2) - u1(1,2)),(u2(1,3)-
u1(1,3))]; %Difference in the means of u1 and u2
    Z1 = [SCORE(C1,1), SCORE(C1,2), SCORE(C1,3)]; %indexes the
 real data of same stimuli for all three PC groups given the first
 combination vector
    Z2 = [SCORE(C2,1), SCORE(C2,2), SCORE(C2,3)]; %indexes the
 real data of same stimuli for all three PC groups given the second
 combination vector

    leng = length(Z1); % length of Z1 rows
    for t = 1:leng %iterates for every real z scored data of the same
 stimulus combination to do a dot product each time
        Dot1(t) = (dot(Z1(t,:),Vvector)); % dot product of a zscored
 data and its PCs means
        Dot2(t) = (dot(Z2(t,:),Vvector)); % dot product of a zscored
 data of specific stimuli and its PCs means
    end

    sigma1 = var(Dot1); %variation of the dot products for stimuli
 Combination 1
    sigma2 = var(Dot2); %variation of the dot products for stimuli
 combination 2

    avgPC1 = mean(Dot1); %average dot product for stimuli Combination
 1
    avgPC2 = mean(Dot2); %average dot product for stimuli Combination
 2

    Dprime(k) = abs(avgPC1-avgPC2) / (sqrt(0.5 * (sigma1 +
 sigma2))); % calculated D'
end

% b. Plot histogram of D' values
figure(2)
histogram(Dprime)
xlabel('Bins')
ylabel('Frequency')
% c. Plot first 36 pairs of clusters in PCA space, (6 x 6 subplots)
% and title each plot with pair # and d' value in order to prove
% that clusters that are visually far apart ALSO have high d' values

figure(3)
for k = 1:36
     x = C(k,1); %repeats indexing from part a for the 36 loop
     y = C(k,2);
     C1 = find(stimlist == x);%finds all trials where stimulus is
 equal to k in the first column of C
```

```
    C2 = find(stimlist == y);%finds all trials where stimulus is
equal to k in the second column of C

    subplot(6,6,k)
    hold on

plot(SCORE(C1,1),SCORE(C1,2),'.','color',plotcolors(stimlist(C(k,1)),:)) %plots
the first and second PC against each other from the first combination
number (1 for first 33)

plot(SCORE(C2,1),SCORE(C2,2),'.','color',plotcolors(stimlist(C(k,2)),:)) %plots
the first and second PC against each other from the second
combination number
    xlabel('PC1')
    ylabel('PC2')
    title({
       ['Pairs ',num2str(C(k,1)),' and ',num2str(C(k,2))]
       ['D'' of ',num2str(Dprime(k))]
       });

    hold off
    set(gcf,'Position',[100 100 1000 800])
end
```





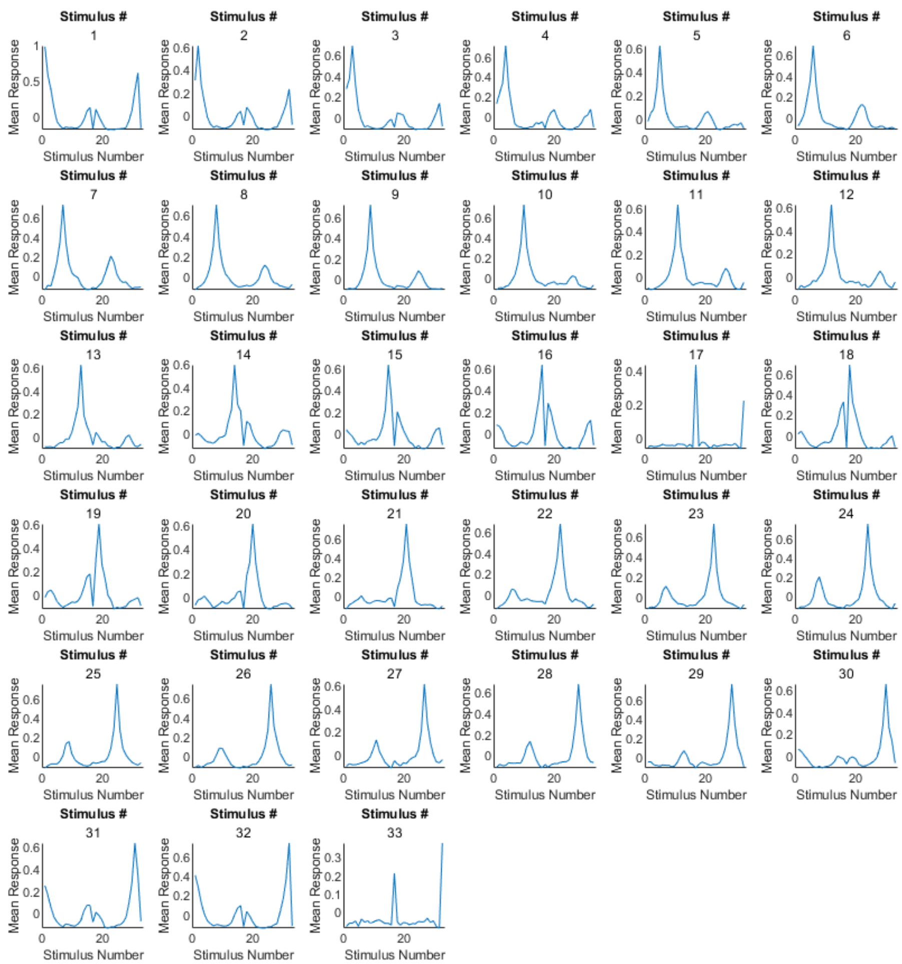
Step 4. Plot a population-level tuning curve for each orientation direction

This will be the AVERAGE response of the population for a particular stimulus

```
% a. First compute tuning curves for each neuron

TC = NaN(number,neurons); %initializes TC

for k = 1:number
    TC(k,:) = mean(zsc(stimlist == k,:));
end

% b. Then organize responses by the PD of each cell
for t = 1:neurons %iterates through every stimulus
    MaxResp = max(TC(:,t)); % max response at each orientation
    PD(t) = find(TC(:,t) == MaxResp);  % finds the stimulation that
 creates the max response of zscored data
end

% c. Plot population tuning curves for each direction
 %find all neurons tuned to the specific stimulus
figure(4)
```

```
for k = 1:number
    TunedDirection = TC(:,PD==k); %indexes where TC is equal to a
 stimulis direction
    AvgTD = mean(TunedDirection,2); %averages the response of each
 direction of stimulus
    subplot(6,6,k)
    hold on
    plot(AvgTD);
    title('Stimulus #',num2str(k));
    xlabel('Stimulus Number')
    ylabel('Mean Response')
    set(gcf,'Position',[100 100 1000 1000])
    xlim([0,34])
end
toc % times the code to make sure it ran all the way and give data on
 the run time
```

Elapsed time is 20.901620 seconds.



Published with MATLAB® R2020b