BE 521: Homework 5

Vision

Spring 2019

42 points

Due: Thursday, 02/27/2020 11:59pm

Objective: Visual responses and likelihood

John Talley
Collaborators: Joseph Iwasyk

V1 Dataset

In this homework, you will work with data from 18 cells recorded from mouse primary visual cortex (also known as V1). Cells in this area are responsive to specific angles. Hence, a common stimulation paradigm is to show the subject a sinusoidal grating drifting at a specific angle (see Figure 1).
This data was collected and graciously provided by Daniel Denman in the Contreras Lab, University of Pennsylvania. The file `mouseV1.mat` contains two variables: `neurons`, a cell array representing all the times that each of the 18 cells fired a spike during the approximately 7 minute long experiment, and `stimuli`, which provides the time (first column, in milliseconds) that a given angle (second column) was presented in this experiment. Note that each stimulus in `stimuli` is presented for exactly 2 seconds, after which a gray screen is presented for approximately 1.5 seconds (therefore each trial is approximately 3.5 seconds in duration.)

1 Stimulus Response (11 pts)

In this section, you will explore the response of the cells to different stimulus angles.

1. How many unique grating angles, m, are there in `stimuli`? (1 pt)

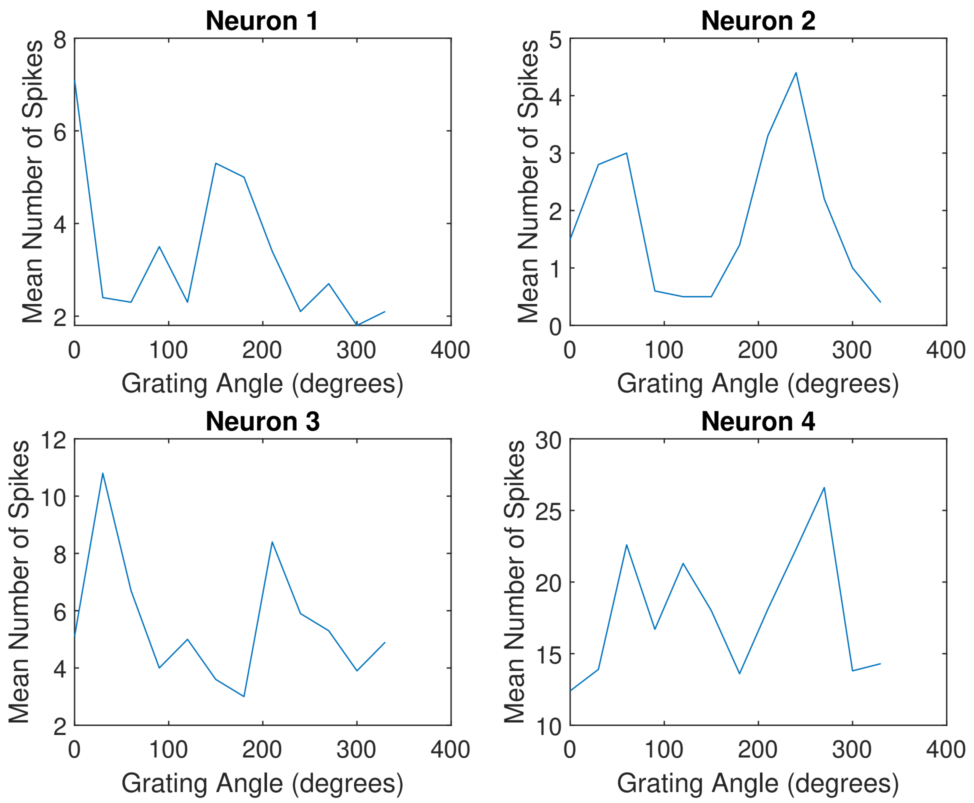
```
% stimuli(:,1) = time stamp (ms) grating angle was presented
% stimuli(:,2) = grating angles presented
% neurons = 18 cell array of all times each neuron fired in 7 minute trial
% period

load('mouseV1.mat');
grating_angles = unique(stimuli(:,2));
length(grating_angles)

% 12 UNIQUE GRATING ANGLES
```

```
ans =

    12
```

2. A *tuning curve* is frequently used to study the response of a neuron to a given range of input stimuli. To create tuning curves for this data, calculate the average number of spikes each cell fires in response to each grating angle. Store the result in an $18 \times m$ dimensional matrix, where each element represents the response of a single neuron to a particular input stimulus angle, with each neuron assigned a row and each angle assigned a column. In a 2×2 Matlab subplot, plot the tuning curve for the first four cells. Place the stimulus angle on the x-axis and the number of spikes on the y-axis. (6 pts)

   Create 18 neuron x 12 grating angles matrix

```matlab
stimulation_ts = stimuli(:,1);
mean_spikes  = zeros(1,length(grating_angles));
tuning_matrix = zeros(18,length(grating_angles));

for n = 1:length(neurons)
    for j = 1:length(grating_angles)
        idx = find(stimuli(:,2) == grating_angles(j)); % find indices of applied stimuli for each neuron
        stim_ts = stimulation_ts(idx);
        num_spikes = zeros(1,length(stim_ts));
        for i = 1:length(stim_ts) % look at 3.500 second window for each stimulus
            num_spikes(i) = sum(neurons{n} >= stim_ts(i) & neurons{n} <= stim_ts(i) + 3500); % create array
        end
        mean_spikes(j) = mean(num_spikes); % mean spikes for neuron n in response to grating angle j
    end
    tuning_matrix(n,:) = mean_spikes; % mean number spikes at each grating angle for neuron n
end

figure;
subplot(2,2,1);
plot(grating_angles, tuning_matrix(1,:));
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 1');
hold on;
subplot(2,2,2);
plot(grating_angles, tuning_matrix(2,:));
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 2');
subplot(2,2,3);
plot(grating_angles, tuning_matrix(3,:));
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 3');
subplot(2,2,4);
plot(grating_angles, tuning_matrix(4,:));
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 4');
```

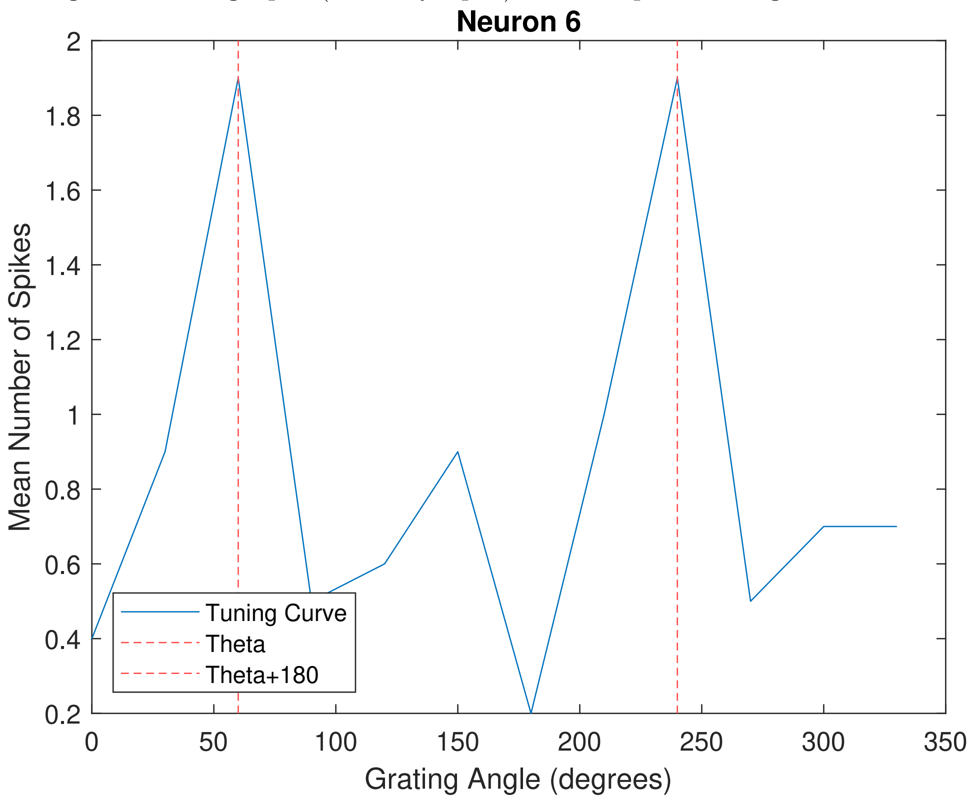


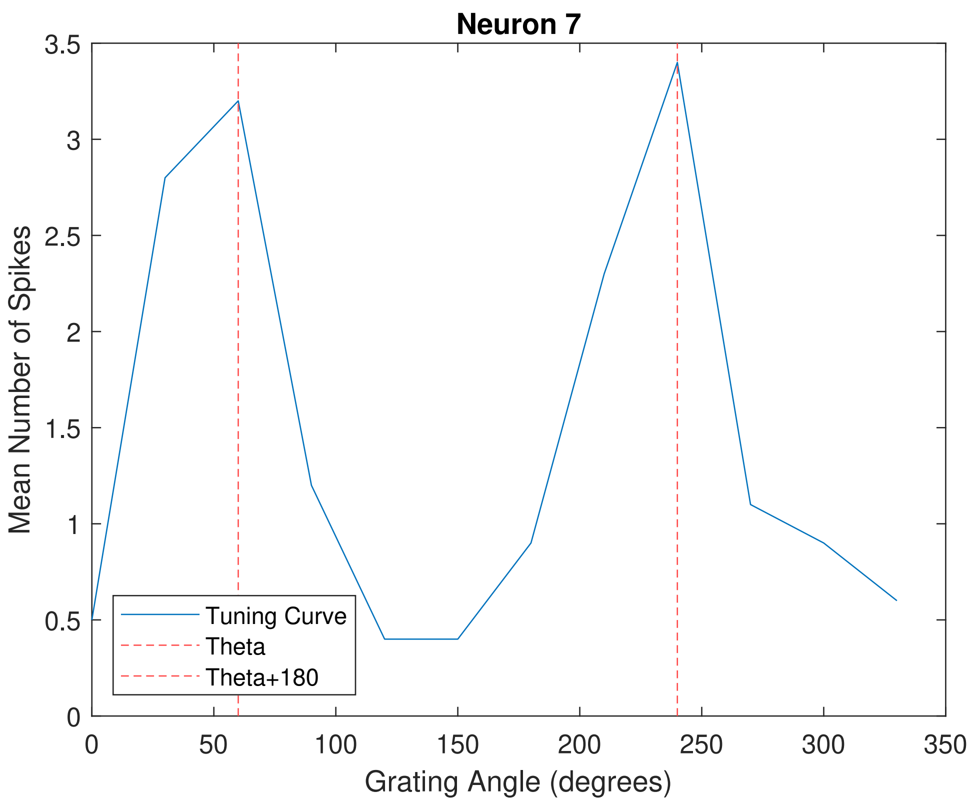
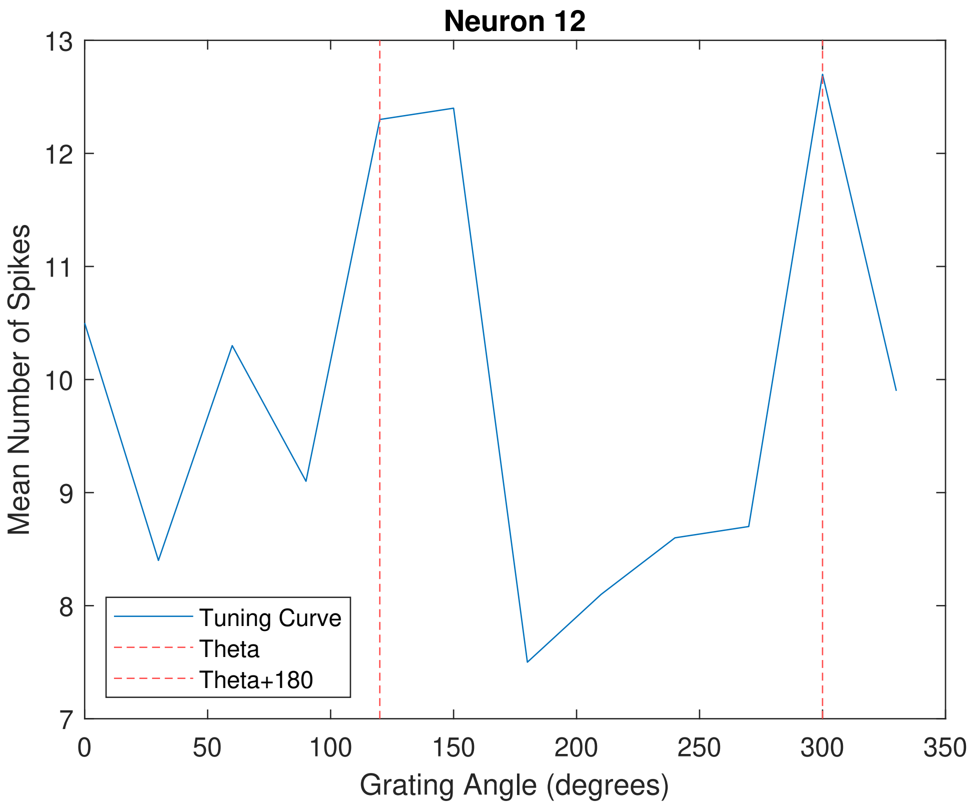
(a) Look through the tuning response curves of each of the 18 cells. How reasonable is it to assume that the response of a given cell to angle θ is the same as its response to angle $\theta + 180$? Include at least a few tuning curves to back up your answer. (2 pts)

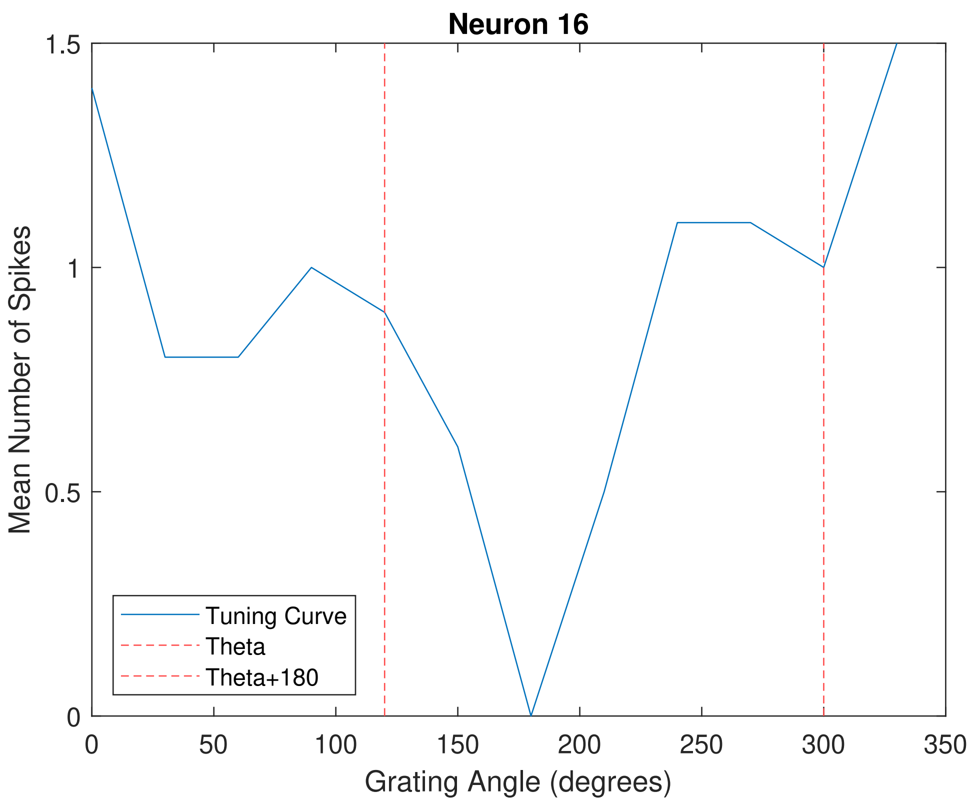
```
% view all 18 tuning curves

figure;
plot(grating_angles, tuning_matrix(6,:));
xline(60,'--r');
xline(240,'--r');
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 6');
legend('Tuning Curve','Theta','Theta+180','Location','Southwest');

figure;
plot(grating_angles, tuning_matrix(7,:));
xline(60,'--r');
xline(240,'--r');
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 7');
legend('Tuning Curve','Theta','Theta+180','Location','Southwest');

figure;
plot(grating_angles, tuning_matrix(12,:));
xline(120,'--r');
xline(300,'--r');
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 12');
```

```
legend('Tuning Curve','Theta','Theta+180','Location','Southwest');

figure;
plot(grating_angles, tuning_matrix(16,:));
xline(120,'--r');
xline(300,'--r');
xlabel('Grating Angle (degrees)');
ylabel('Mean Number of Spikes');
title('Neuron 16');
legend('Tuning Curve','Theta','Theta+180','Location','Southwest');

% Yes.  Tuning curves for neurons 6, 7, 12, and 16 show a few good examples
% of the spike response at angle theta being equal (or nearly equal) to the
% response at angle theta + 180.
```

Yes. Tuning curves for neurons 6, 7, 12, and 16 show a few good examples of the spike response at angle theta being equal (or nearly equal) to the response at angle theta + 180.




Neuron 7
Mean Number of Spikes
Grating Angle (degrees)
Tuning Curve
Theta
Theta+180
0
0.5
1
1.5
2
2.5
3
3.5
0
50
100
150
200
250
300
350

Neuron 12
Mean Number of Spikes
Grating Angle (degrees)
Tuning Curve
Theta
Theta+180
7
8
9
10
11
12
13
0
50
100
150
200
250
300
350



(b) Does this assumption have any physiological justification (given what you know about the types of cells in V1)? (2 pts)

Given that these are visual cortex cells, it does make sense physiologically that the neuronal responses are the same at angle theta and theta + 180. For example, if you take a linear horizontal drift grating and re-orient it 180 degrees, it will be visually processed the same way. This is a sinusoidal drifting grating, and mathematically, the sine(theta) = abs(sin(theta+180)). If you rotate any of the three displayed grating lines at the beginning of this problem set by 180 degrees, the orientation overlaps its original position and is indistinguishable in the mouse's primary visual cortex.

2 Neural Decoding (31 pts)

Suppose we would like to work backwards - that is, for a given neural response, can we predict what the grating angle was? This process is called "neural decoding," and is especially of interest to the BCI motor control community (as we'll see in a later homework). In this section, we will try out an approach which is detailed in Jazayeri & Movshon 2006 [1]. The method we will use involves finding the maximum likelihood of the data.

Here, the data is the number of spikes s_i that cell i fires when the subject sees a stimulus with grating angle θ. One way to think about our likelihood function is to ask the question "given a stimulus angle θ, how many spikes would I expect this cell to fire?" We can represent this number of spikes s_i using a Poisson process with parameter $f_i(\theta)$ for a stimulus θ, where f_i represents neuron i's tuning function. A Poisson distribution is often used to model count data that occurs at a constant rate, and in this case the rate is given

[1] A copy of this paper is included if you are curious, but we walk you through the method in this homework.

by $f_i(\theta)$. In other words, our likelihood function $L_i(\theta)$ for each neuron i is the probability $p(s_i|\theta)$ of neuron i firing s_i spikes for a given value of θ. The idea in this method is to calculate the log likelihood[2] function of each neuron and then add them all together to get the log likelihood function of the entire population of (n) neurons. We often work with the *log* likelihood because it allows adding of probabilities instead of multiplying, which can lead to numerical problems.

$$
\begin{aligned}
p(s_i|\theta) &\sim Pois(f_i(\theta)) = \frac{f_i(\theta)^{s_i}}{s_i!} e^{-f_i(\theta)} && \text{(Poisson probability density)} \\
\text{Ł}_i(\theta) &= p(s_i|\theta) && \text{(Likelihood of a given neuron firing at } s_i\text{)} \\
\text{Ł}(\theta) &= \prod_{i=1}^{n} p(s_i|\theta) && \text{(Joint likelihood of all n neurons)} \\
\log \text{Ł}(\theta) &= \sum_{i=1}^{n} \log L_i(\theta) = \sum_{i=1}^{n} \log p(s_i|\theta) && \text{(Take log)} \\
&\propto \sum_{i=1}^{n} s_i \log f_i(\theta) && \text{(evaluation of PDF and simplifying)}
\end{aligned}
$$

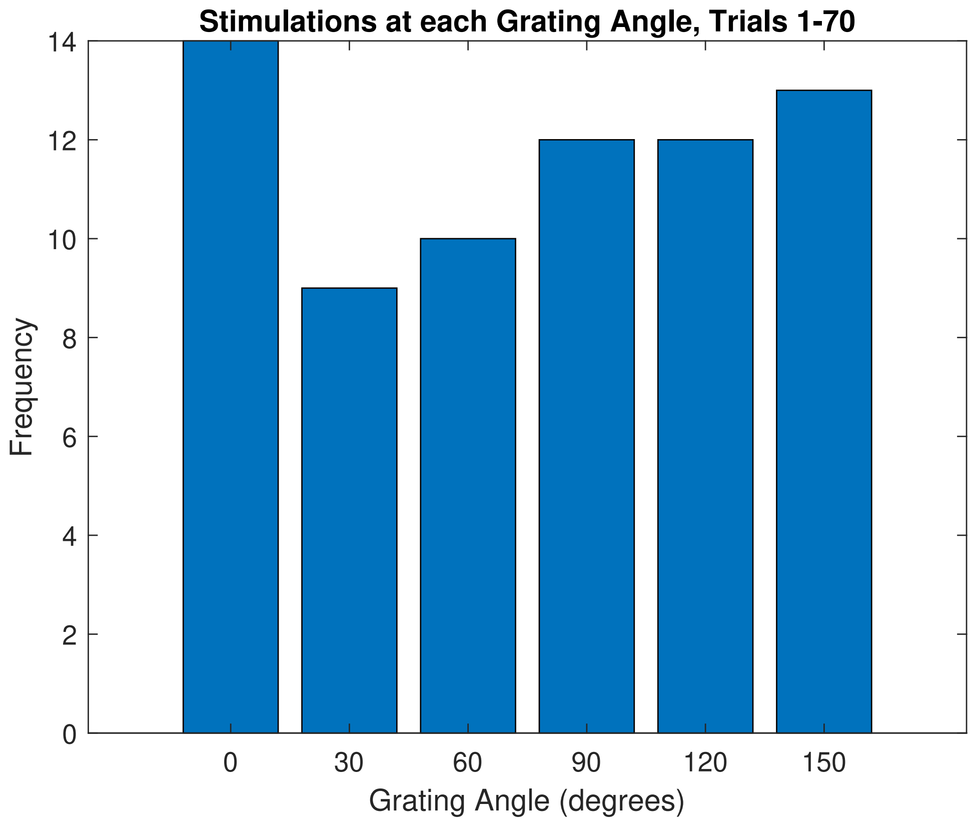
Thus, we can define the log likelihood for each neuron i as the log of its tuning curve $f_i(\theta)$ times the number of spikes s_i it fires for a particular stimulus θ, and the population log likelihood is simply the summation across all cells. This tells us that, given a set of tuning curves $f_i(\theta)$, we can compute the likelihood of observing our data s. But we already have those tuning curves for each cell from question 1.2, so all we need to know for a new (hidden) stimulus is how many spikes each neuron fires. Let $\mathbf{s}$ be the n-dimensional column vector of the number of spikes each cell fires after the subject is presented with a new stimulus θ' and let $\mathbf{F}$ be the $n \times m$ matrix representing the tuning curves of each neuron at each of the m stimuli (for us, m is the number of stimuli between 0 and 150 degrees because we assume that all neurons respond equally to θ and $\theta + 180$ degrees.) We can then compute the log likelihood of the new stimulus θ' easily using the inner product of $\mathbf{s}$ and $\mathbf{F}$: `L = s'*log(F)`.

1. Compute the matrix `F` by recalculating the tuning curves you calculated in question 1.2 using only the **first 70** trials (this is akin to our "training" data). You will use the remaining 50 trials (as "testing" data) to make predictions. Make a histogram of the number of stimulation angles for the first 70 trials to ensure that each angle (0 to 150) is presented at least a few times. (4 pts)

```
% compute log likelihood L of stimulus theta' based on response data
% re-compute matrix F as training data

stimulation_ts = stimuli(1:70,1);
mean_spikes  = zeros(1,length(grating_angles)/2);
tuning_matrix_train1 = zeros(18,length(grating_angles)/2); % rating angles 0 to 150
tuning_matrix_train2 = zeros(18,length(grating_angles)/2); % rating angles 180 to 330

angle_counts1 = zeros(1,6);
angle_counts2 = zeros(1,6);

for n = 1:length(neurons)

    for j = 1:6
        idx = find(stimuli(1:70,2) == grating_angles(j)); % find indices of applied stimuli for each neuron
        angle_counts1(j) = length(idx); % count the number of times each angle shows up in stimulations 1:70
        stim_ts = stimulation_ts(idx);
        num_spikes = zeros(1,length(stim_ts));
        for i = 1:length(stim_ts) % look at 3.500 second window for each stimulus
            num_spikes(i) = sum(neurons{n} >= stim_ts(i) & neurons{n} <= stim_ts(i) + 3500); % create array
```

[2] log = natural log unless otherwise specified

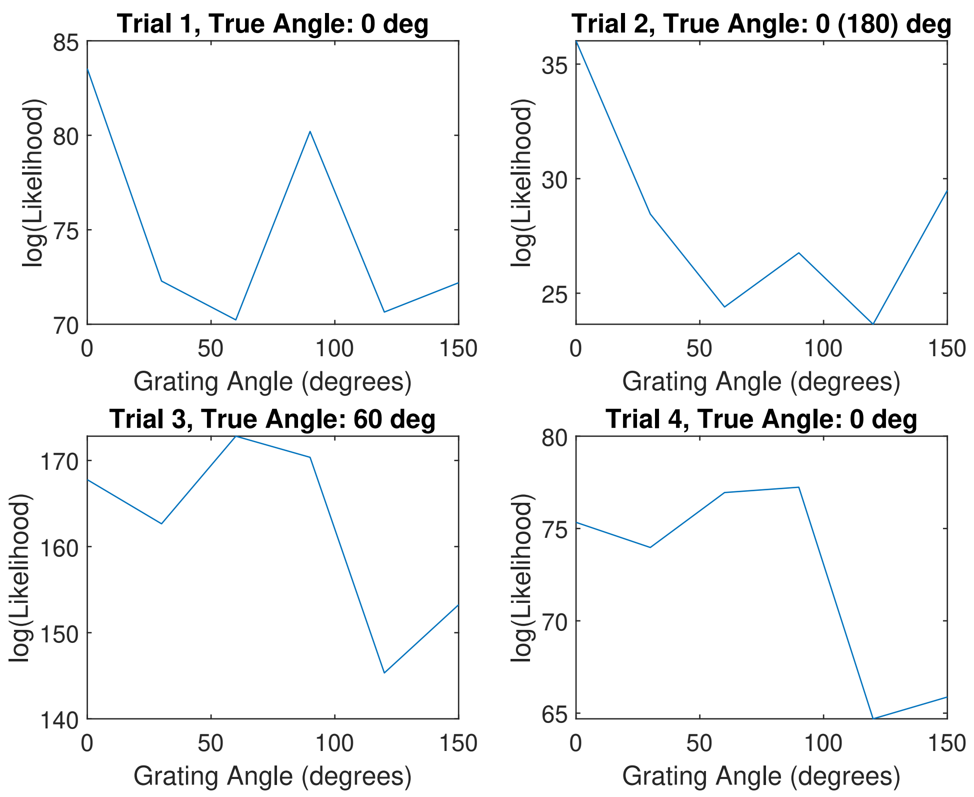
```
        end
        mean_spikes(j) = mean(num_spikes); % mean spikes for neuron n in response to grating angle j
    end
    tuning_matrix_train1(n,:) = mean_spikes; % mean number spikes at each grating angle for neuron n

    for j = 7:length(grating_angles) % angles 180 to 330
        idx = find(stimuli(1:70,2) == grating_angles(j)); % find indices of applied stimuli for each neuron
        angle_counts2(j-6) = length(idx); % count the number of times each angle shows up in stimulations 1:
        stim_ts = stimulation_ts(idx);
        num_spikes = zeros(1,length(stim_ts));
        for i = 1:length(stim_ts) % look at 3.500 second window for each stimulus
            num_spikes(i) = sum(neurons{n} >= stim_ts(i) & neurons{n} <= stim_ts(i) + 3500); % create array
        end
        mean_spikes(j-6) = mean(num_spikes); % mean spikes for neuron n in response to grating angle j
    end
    tuning_matrix_train2(n,:) = mean_spikes; % mean number spikes at each grating angle for neuron n
end

% Average the two training matrices so that theta is paired with theta+180
% mean spikes for each cell

F = (tuning_matrix_train1 + tuning_matrix_train2)/2;

% Histogram of angles, treaing 180-330 as 0-150 in pairs, respectively
angle_counts = angle_counts1 + angle_counts2;
figure;
bar(grating_angles(1:6), angle_counts);
xlabel('Grating Angle (degrees)');
ylabel('Frequency');
title('Stimulations at each Grating Angle, Trials 1-70');
```



2. For the 50 “testing” trials, compute a $n \times 50$ matrix `S` where each row represents the number of spikes one neuron fired in response to a each of the 50 trials. With this, you can easily compute the log likelihood functions for all the trials at once with the command: `L_test = S’*log(F)`. (Hint: add a small number to `F` to avoid taking the log of 0)

   (a) Plot the likelihood functions for the first four testing trials in a 2×2 subplot. In the title of each plot, give the trial number (1, 2, 3, or 4) and the true stimulation angle. Make sure to label your axes correctly. (5 pts)

```
% Create matrix S

S = zeros(18,50);
stimulation_ts = stimuli(71:120,1);
numSpikeEvents = zeros(1,50);

for n = 1:length(neurons)
    for k = 1:length(stimulation_ts)-1
%         numSpikeEvents(k) = sum(neurons{n} >= stimulation_ts(k) & neurons{n} <= stimulation_ts(k) + 3
        numSpikeEvents(k) = sum(neurons{n} >= stimulation_ts(k) & neurons{n} <= stimulation_ts(k+1));
    end
    numSpikeEvents(50) = sum(neurons{n} >= stimulation_ts(k) & neurons{n} <= stimulation_ts(k) + 3500);
    S(n,:) = numSpikeEvents;
end

% COMPUTE LIKELIHOOD AND PLOT FOR FIRST 4 TRIALS
trueAngle1 = stimuli(71,2);
trueAngle2 = stimuli(72,2);
trueAngle3 = stimuli(73,2);
trueAngle4 = stimuli(74,2);

L_test = S'*log(F);
figure;
subplot(2,2,1);
plot(grating_angles(1:6), L_test(1,:));
xlabel('Grating Angle (degrees)');
ylabel('log(Likelihood)');
title('Trial 1, True Angle: 0 deg');
subplot(2,2,2);
plot(grating_angles(1:6), L_test(2,:));
xlabel('Grating Angle (degrees)');
ylabel('log(Likelihood)');
title('Trial 2, True Angle: 0 (180) deg');
subplot(2,2,3);
plot(grating_angles(1:6), L_test(3,:));
xlabel('Grating Angle (degrees)');
ylabel('log(Likelihood)');
title('Trial 3, True Angle: 60 deg');
subplot(2,2,4);
plot(grating_angles(1:6), L_test(4,:));
xlabel('Grating Angle (degrees)');
ylabel('log(Likelihood)');
title('Trial 4, True Angle: 0 deg');
```



(b) How well do these four likelihood functions seem to match the true stimulation angle? Explain in a few sentences. (3 pts)

These four likelihood functions accurately predicted the true stimulation angle for testing trials 1, 2, and 3 as 0, 180, and 60 degrees. However, it wrongly predicted trial 4 as an angle of 90 degrees instead of the true stimulation angle of 0 degrees. However, teh likelihood function was still close on trial 4, predicting the log(Likedlihood) of 9 degrees as ˜77 and for 0 degrees as ˜75, so the model was not far off the true stimulation angle.

(c) Compute the maximum likelihood estimate (MLE) for each of the 50 trials. This is another way of asking which angle θ has the highest probability.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \; L(\theta) = \arg\max_{\theta} \; \log L(\theta)$$

In what percentage of the 50 trials did your MLE correctly predict the stimulation angle [0-150]? (5 pts)

```
max_angle_idx = zeros(1,50);
predict_max_angle = zeros(50,1);
for i = 1:50
    max_angle_idx(i) = find(L_test(i,:) == max(L_test(i,:)));
    predict_max_angle(i) = grating_angles(max_angle_idx(i));
end

true_angle = stimuli(71:120,2);
for i = 1:length(true_angle)
    if true_angle(i) > 150
        true_angle(i) = true_angle(i) - 180; % convert to 0-150 scale from original input angles
    end
end
```

```
pct_accuracy = 100*(sum(true_angle == predict_max_angle))/50;

% 38 percent accuracy in predicting the true stimulation angle based on the
% max likelihood of each stimulation angle in each trial.
```

38 percent accuracy in predicting the true stimulation angle based on the max likelihood of each stimulation angle in each trial.

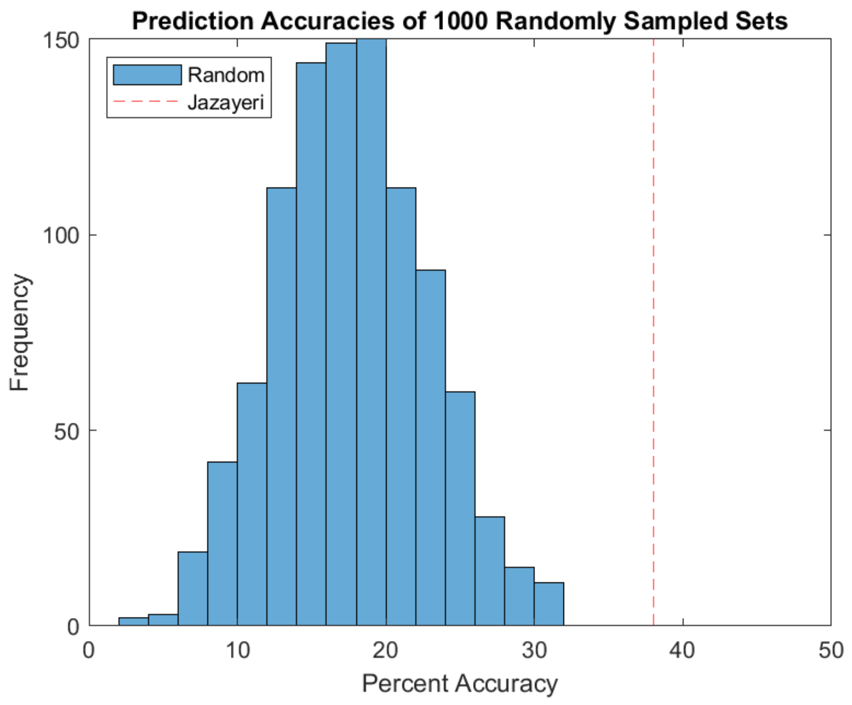
(d) In a few sentences, discuss how well this method worked for predicting the input angle from the response of the 18 neurons. What might you change in the experimental protocol to try and get a better prediction? (3 pts)

This method did not work particualrly well in predicting the true input stimulation angle from the response of 18 neurons and the max likelihood estimate, correctly predicting the input angle in 38This accuracy is better than random prediction, which we would assume to yield a 16.7are multiple ways in which the experimental protocol could have improved the accuracy of this model. This experiment conducted trials on only one mouse, and examined the responses of only 18 neurons. In a better experimental protocol, we may have examined 3 animals over multiple 7 minute response trials across different days in order to statistically validate the predicted spike event responses to each input angle. If possible, it would also be useful to include more neurons in the recording.

3. It is important to show that your findings are not a result of chance. One way to demonstrate this is using a "permutation test." Here, we will perform a permutation test by randomly reassigning new grating angles to the 50 test responses and then calculating how often the new grating angles match the true stimulation angles.

(a) Simulate the chance prediction (the "null" distribution) by randomly reassigning the stimulation angles 1000 times. For each permutation, calculate the percent accuracy of each label. Create a histogram of the 1000 accuracy measurements for the null distribution. Add a red vertical line with the accuracy calculated from using the Jazayeri method above. (4 pts)

```
angle_set = stimuli(:,2); % create set of angles

for i = 1:length(angle_set)
    if angle_set(i) > 150
        angle_set(i) = angle_set(i) - 180; % convert to 0-150 scale from original input angles
    end
end

accuracies = zeros(1,1000);

for p = 1:1000
    rand_angles = randsample(angle_set,50);
    accuracies(p) = 100*(sum(true_angle == rand_angles))/50;
end

figure;
histogram(accuracies, 15);
xlabel('Percent Accuracy');
ylabel('Frequency');
xline(pct_accuracy,'--r');
xlim([0 50]);
title('Prediction Accuracies of 1000 Randomly Sampled Sets');
legend('Random','Jazayeri','Location','Northwest');
```



(b) Is the null distribution what you expected? Explain. (1 pt)

Yes, the null distribution is as we would expect. The null distribution is generally a normal bell curve centered around 16-17 percent. Because we are randomly predicting an input stimulation an angle from the set of 0,30,60,90,120, and 150, it makes sense that a random sample would be correct about 1/6th of the time when iterated over 1000 samples.

(c) What is the probability that your actual accuracy measurement comes from the null-distribution? That is, calculate the fraction of permutation samples with accuracy *more extreme* relative to the mean of the null distribution (less than your measurement if your value is less than the mean, or more than your measurement if your value is more than the mean). (2 pts)

```
pval = sum(accuracies > pct_accuracy)/length(accuracies);

% On this trial, there was a 0 percent probability that the null distribution
% prediction accuracies would be better than that of Jazayeri's Method
% (38 percent) over 1000 randomly sampled sets of predictions.
```

On this trial, there was a 0 percent probability that the null distribution prediction accuracies would be better than that of Jazayeri's Method (38 percent) over 1000 randomly sampled sets of predictions.

(d) What would the probability be if your accuracy had been 25%? (1 pt)

```
pval_25 = sum(accuracies > 25)/length(accuracies);
% If the accuracy had been 25%, the probability would be 0.0640 based on
% this random sample of 1000 sets of 50 prediction angles.
```

If the accuracy had been 25this random sample of 1000 sets of 50 prediction angles.

(e) What is this value typically called? (1 pt)

This value is typically called a p-value in statistics.

4. The tuning curves and neuron responses to a new trial were calculated using the number of spikes each neuron fired after the stimulation. But what if a particular neuron just happens to fire a lot and fires even more when it gets a stimulus to which it's tuned? Those neurons could "swamp" the log likelihood estimate given in Equation 1 by virtue of just having a larger average s_i and $f_i(\theta)$. How might we correct for this type of behavior? Suggest a possible method. (2 pts)

   If a particular neuron happens to fire a lot and fires even more when it receives the input stimulus to which it is tuned, there are ways to normalize the spike event data to maintain an accurate log likelihood estimate. One way to do so may be to z-score the data when receiving an input stimulus as compared to the baseline. Experimentally, this would mean recording a baseline measure of the neuronal firing rate without stimulus, then recording the neuronal firing rate for each neuron in the field of view upon receiving the stimulation inputs over a session (120 trials of about 3.5 seconds each, here). For each neuron we could the calibrate the firing rate by z-scoring (or a similar statistical normalization method) the experimental spikes compared to the baseline, and then use these values as the inputs for our log likelihood function.