Assignment 3/4

Anna Hekkink
Student number: 45964345
Tutorial group number: 3104

1. You are getting married tomorrow.
Responses: R (it rains), DR (it doesn't rain)
Stimulus: BoM Prediction of rain, P (Predicted rain), NP (no Predicted rain)

Information we are given:

$P(R) = 5/365$ *
$P(P \mid R) = 0.9$
$P(P \mid DR) = 0.1$

* we choose to ignore leap years.

To calculate the probability of it raining tomorrow, given the BoM's prediction, we want: $P(R \mid P)$.

Using Baye's theorm : $P(R \mid P) = \dfrac{P(P \mid R)\,P(R)}{P(P)}$

Using the sum rule: $P(P) = \sum_r P(P \mid r)\,P(r)$

$= P(P \mid R)P(R) + P(P \mid DR)P(DR)$

So then, $P(R \mid P) = \dfrac{P(P \mid R)P(R)}{P(P \mid R)P(R) + P(P \mid DR)P(DR)}$

$= \dfrac{0.9 \times 5/365}{(0.9 \times 5/365) + \left(0.1 \times \frac{360}{365}\right)} = 0.1111$

Hence, the probability that it will rain on your wedding tomorrow is 0.1111, i.e. 11.11% chance. This is a relatively low chance of rain, so may not support the decision to rent a marquee. It would depend on the cost of the marquee and the risk you are willing to take.

2. Consider the case of Bayesian Inference for the case $L = (s - s_{bayes})^2$. So, we aim to minimize:

$$\int (s - s_{bayes})^2 \, P(s|r) \, ds$$

w.r.t. s_{bayes}.

We find the optimal estimate of s_{bayes} by differentiating w.r.t. s_{bayes} and setting equal to 0.

$$0 = \frac{d}{ds_{bayes}} \left[\int (s^2 - 2 s s_{bayes} + s_{bayes}^2) \, P(s|r) \, ds \right]$$

$$0 = \frac{d}{ds_{bayes}} \left[\int s^2 P(s|r) \, ds - 2 s_{bayes} \int s \, P(s|r) \, ds + s_{bayes}^2 \int P(s|r) \, ds \right]$$

Note $\int P(s|r) \, ds = 1$. Then, by differentiating, we get:

$$0 = -2 \int s \, P(s|r) \, ds + 2 s_{bayes}$$

$$\Rightarrow s_{bayes} = \int s \, P(s|r) \, ds$$ as required $\square$

3. consider the joint distribution :

X\Y	0	1
0	1/3	1/3
1	0	1/3

Entropy def: $H = -\sum_{r} P(r) \log_2 (P(r))$.

Find $H(X) = -\left(P(X=0)\log_2 (P(X=0)) + P(X=1)\log_2(P(X=1))\right)$

$P(X=0) = 2/3$, $P(X=1) = 1/3$. Plug these in to get:

$H(X) = -\left(2/3 \log_2 (2/3) + 1/3 \log_2 (1/3)\right)$

$= 0.9183$ bits

Similarly, $H(Y) = -\left(P(Y=0)\log_2 (P(Y=0)) + P(Y=1)\log_2(P(Y=1))\right)$

$P(Y=0) = 1/3$, $P(Y=1) = 2/3$

$H(Y) = -\left(1/3 \log_2 (1/3) + 2/3 \log_2 (2/3)\right)$

$= 0.9183$ bits

$$H(X|Y) = -\sum_{X=0,1}\sum_{Y=0,1} P(Y)\, P(X|Y) \log_2 (P(X|Y))$$

$$= -\Big(P(Y=0)\left[P(X=0|Y=0)\log_2 (P(X=0|Y=0)) + P(X=1|Y=0)\log_2(P(X=1|Y=0))\right]$$
$$+ P(Y=1)\left[P(X=0|Y=1)\log_2 P(X=0|Y=1) + P(X=1|Y=1)\log_2(P(X=1|Y=1))\right]\Big)$$

$P(X|Y) =$

X\Y	0	1
0	1	1/2
1	0	1/2

$P(Y=0) = 1/3$

$P(Y=1) = 2/3$.

Plug these values in to get ...

$$H(X|Y) = -\left(\tfrac{1}{3}\left[1\cdot\log_2(1) + 0\right] + \tfrac{2}{3}\left[\tfrac{1}{2}\log_2(\tfrac{1}{2}) + \tfrac{1}{2}\log_2(\tfrac{1}{2})\right]\right)$$

$$= -\left(0 + \tfrac{2}{3}\cdot(-1)\right)$$

$$= \tfrac{2}{3} \text{ bits}$$

$H(Y|X)$ can be found similarly.
Instead use:

$P(X=0) = 2/3$

$P(X=1) = 1/3$

$P(Y|X) =$

X\Y	0	1
0	1/2	1/2
1	0	1

$$H(Y|X) = -\left(\tfrac{2}{3}\left[\tfrac{1}{2}\log_2(\tfrac{1}{2}) + \tfrac{1}{2}\log_2(\tfrac{1}{2})\right] + \tfrac{1}{3}\left[0 + 1\cdot\log_2(1)\right]\right)$$

$$= -\left(\tfrac{2}{3}(-1) + 0\right)$$

$$= 2/3 \text{ bits}$$

$$H(X,Y) = -\sum_{(x,y)} P(x,y)\log_2(P(x,y))$$

$$= -\left(\tfrac{1}{3}\log_2(\tfrac{1}{3}) + \tfrac{1}{3}\log_2(\tfrac{1}{3}) + \tfrac{1}{3}\log_2(\tfrac{1}{3}) + 0\right)$$

$$= -\log_2(\tfrac{1}{3})$$

$$= 1.5850 \text{ bits}$$

Mutual information, $I_m(X,Y) = H(x) - H(X|Y)$

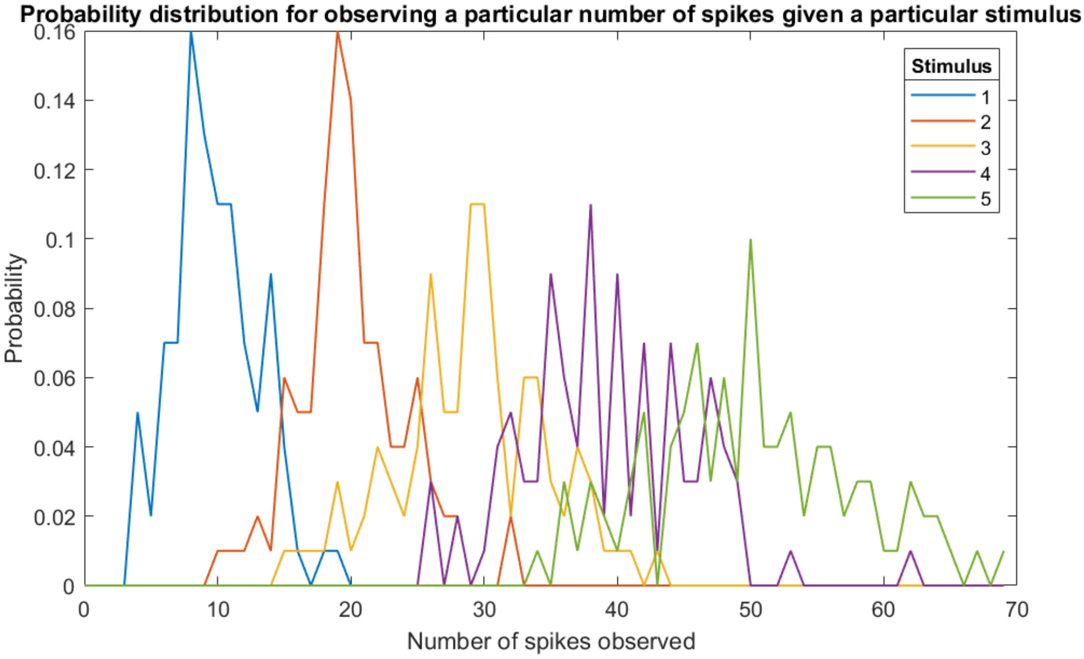
$$I_m(X,Y) = 0.9183 - 2/3$$

$$= 0.2516 \text{ bits}$$

MATLAB Questions

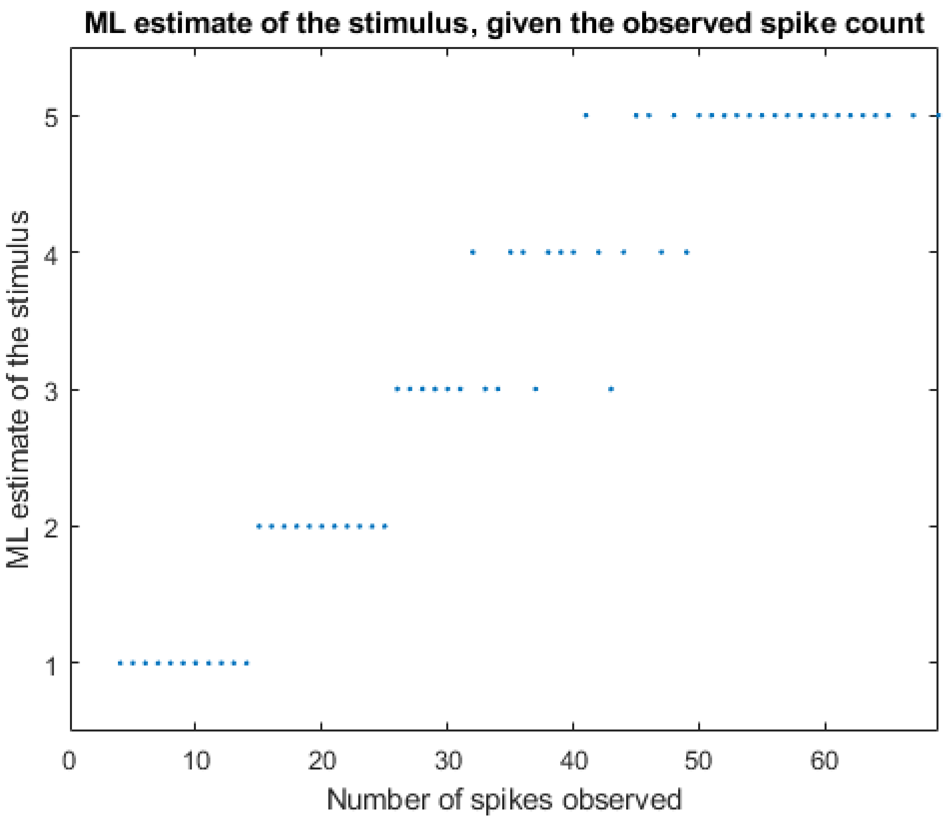
```
clear;
% QUESTION 1

% Case with 5 stimuli
% Set up parameter values
T = 1000;                                              % 1000 ms = 1 sec
rateStim = [0.01, 0.02, 0.03, 0.04, 0.05];           % Rates in spikes/ ms
numStimPres = [100, 100, 100, 100, 100]; % All stimuli presented 100 times
binSize = 1;                                            % 1 ms time bins
seed = 1;

[trains] = generatetrains(T, binSize, numStimPres, rateStim, seed);


% QUESTION 2

% Set up an array to store no. of spikes in each spike train
numberOfStimuli = length(numStimPres);
r = numberOfStimuli;        % no. of rows = no. of stimuli
c = max(numStimPres);       % no. cols = max no. of stimuli presentations
spikeSums = zeros(r,c);

% Calculate spike count for each spike train
for i = 1:r
    for j = 1:c
        spikeSums(i,j) = sum(trains(i,j,:));
    end
end

% Determine max no. spikes occuring in any train,so find max of whole array
maxN = max(spikeSums, [], 'all');



% QUESTION 3

% Set up array pns to store the probabilities of observing a particular no.
% of spikes, n, given a particular stimulus, s
pns = zeros(numberOfStimuli, (maxN +1));

for i = 1:numberOfStimuli
    for n = 1:(maxN +1)
        % Look at row corresponding to stimulus i
        x = spikeSums(i,:);
        % Find all the values which equal n-1
        % (Note use of n-1 instead of n to account for possibility of zero)
        y = find( x == (n-1));
        % Probability of n = no. times n occured / total no. stim pres
        p = length(y)/numStimPres(i);

        % Record this probability in pns
        pns(i, n) = p;
    end
end
```

```
figure(1)
plot(0:maxN, pns, 'LineWidth', 1)
title('Probability distribution for observing a particular number of spikes given a
particular stimulus')
lgd = legend({'1','2','3','4','5'});
title(lgd, 'Stimulus')
xlabel('Number of spikes observed')
ylabel('Probability')
```



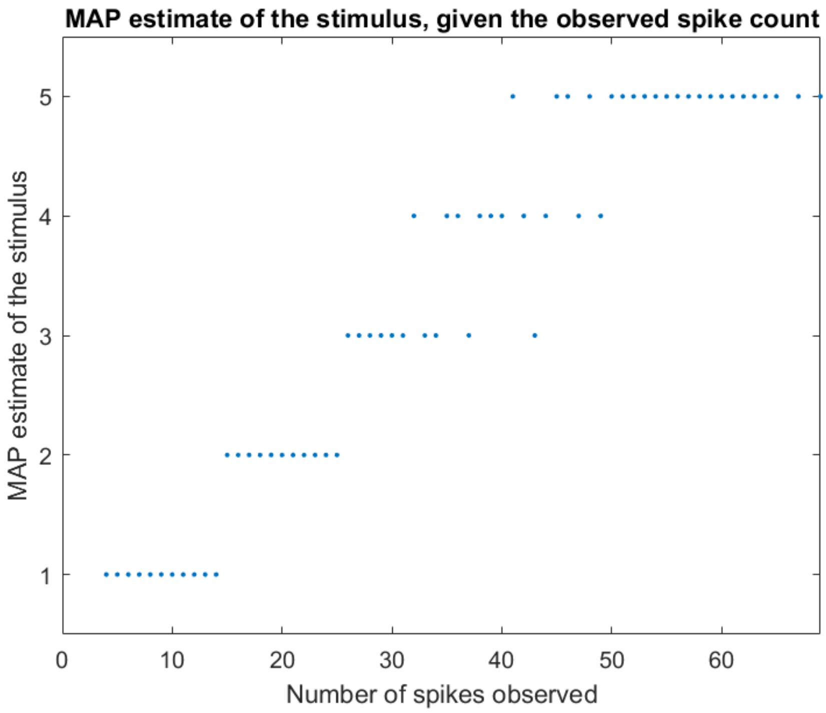

```
% QUESTION 4

% Set up vector for ML Estimate
mlEstimate = zeros(1,(maxN +1));

% ML estimate S_ml is the stimulus that maximises p(n|S_ml)
% So for each column of pns, we find the max value and return the
% corresponding stimulus.

% Note that the (m+1)th element of mlEstimate contains the MLE of the
% stimulus,  given that m spikes occured.

for n = 1:maxN+1
    % Find max in column n and return index i
    % Index i corresponds to the ML estimate, S_ml
    % (Note that if two stimuli have the same probability, matlab
    % automatically picks the smaller stimulus, as requested in assignment)
    [M, i] = max(pns(:,n));
    if M == 0
        mlEstimate(n) = NaN;
```

```
    else
    mlEstimate(n) = i;
    end
end

% Plot of the MLE for the stimulus, given the observed spike count

% Note: I am choosing to plot the data as points rather than a line because
% I think it displays the information more effectively and makes sure no
% data is lost due to the fact some inputs are NaN

figure(2)
subplot(1,1,1)
plot(0:maxN, mlEstimate, 'Marker', '.', 'LineStyle', 'none')
title('ML estimate of the stimulus, given the observed spike count')
ylabel('ML estimate of the stimulus')
xlabel('Number of spikes observed')
xlim([0, maxN])
ylim([0.5, 5.5])
yticks([1, 2, 3, 4, 5])
```

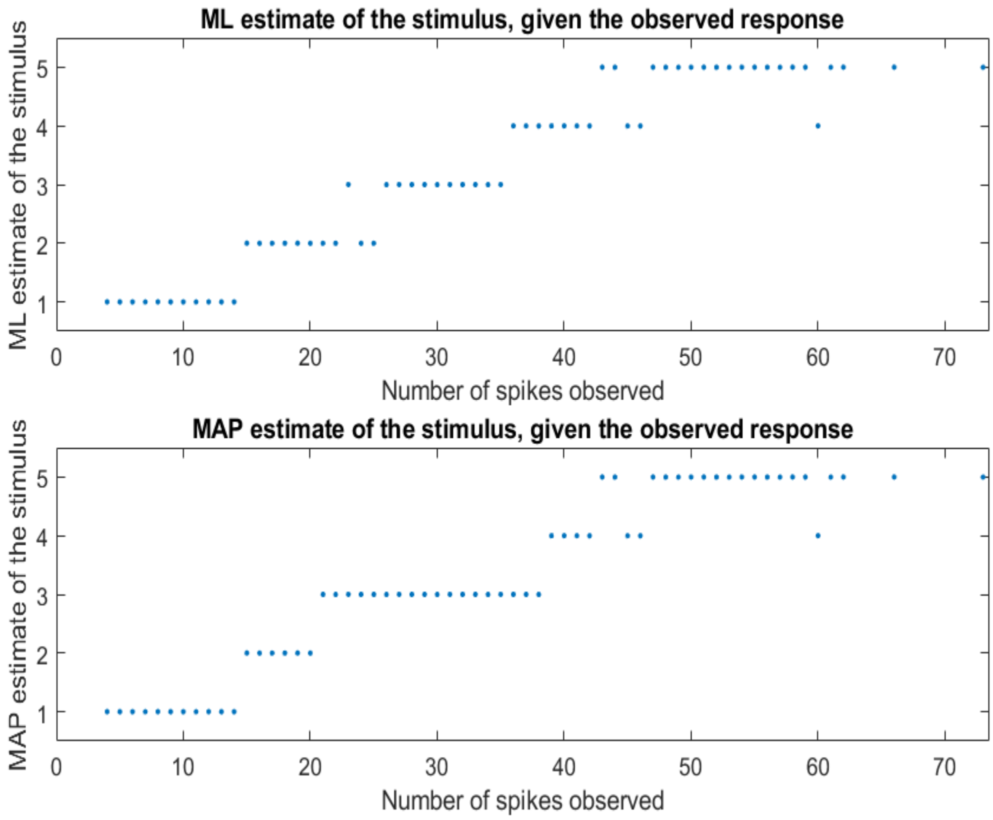



```
% QUESTION 5

% Set up vector for MAP Estimate
mapEstimate = zeros(1,(maxN +1));

% To calculate the MAP Estimate, we need the probs of each stimulus, P(s)
```

```
ps = numStimPres./sum(numStimPres);

% Set up array for storing P(s_map|n) that is P(n|s_map)*P(s_map)
% i.e. psn = pns * ps
psn = zeros(numberOfStimuli, (maxN +1));
for i = 1:numberOfStimuli
    for n = 1:(maxN+1)
        psn(i,n) = pns(i,n) * ps(i);
    end
end

% MAP estimate S_map is the stimulus that maximises p(S_map|n)
% So for each column of psn, we find the max value and return the
% corresponding stimulus.

for n = 1:maxN+1
    % Find max in column n and return index i
    % Index i corresponds to the MAP estimate, S_map
    [M, i] = max(psn(:,n));
    if M == 0
        mapEstimate(n) = NaN;
    else
        mapEstimate(n) = i;
    end
end

% Plot of the MAP estimate for the stimulus, given the observed spike count
figure(3)
% Again, plotting just the points
subplot(1,1,1)
plot(0:maxN, mapEstimate, 'Marker', '.', 'LineStyle', 'none')
title('MAP estimate of the stimulus, given the observed spike count')
ylabel('MAP estimate of the stimulus')
xlabel('Number of spikes observed')
xlim([0, maxN])
ylim([0.5, 5.5])
yticks([1, 2, 3, 4, 5])

% Notice that this is the same plot as the one for the ML estimate, since
% in this case all the stimuli are equally likely
```



```
% QUESTION 6

% Plots
```



EXPLANATION OF THE DIFFERENCES

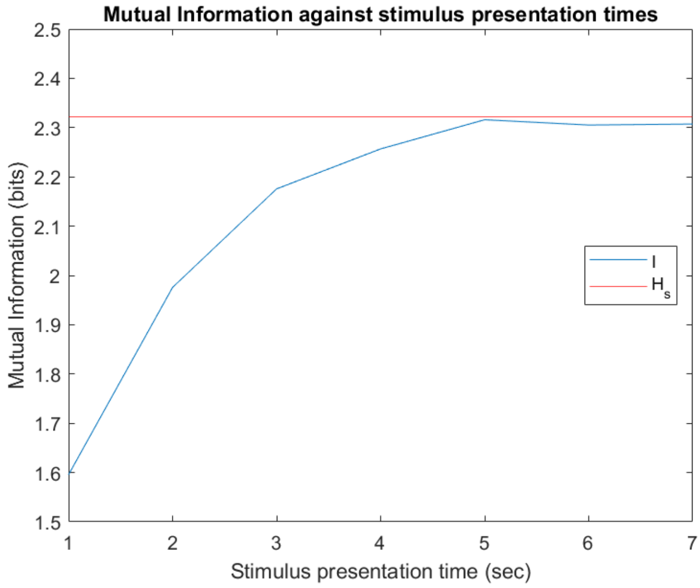
There is no significant difference between the ML estimate plots, because the ML estimate does not take in to account the probability distribution of the stimulus, hence it keeps the same shape. The only noticeable difference is that the points are now less spread out, with less overlap in the ranges of spike count values which predict each stimulus. This is simply because we are running an extra 300 simulations of the experiment, and so the results are becoming more accurate.

The major difference in MAP plots is that the MAP estimate now favours stimulus 3, since it takes in to account the probability distribution of the stimulus, and since stim 3 is now presented four times as often as the other stimuli, it's probability is higher. On the graph, this has resulted in more observed spike counts giving an estimate of stimulus 3. This has also caused the MAP estimate to become a smoother function with less overlap in the range of values (of the spike count) for when either stimuli 2 or 3 were chosen and equally less overlap between stimuli 3 and 4.

```
% QUESTION 7

% Entropy in spike count response , H , is equal to:
% H = - sum( p(n) * log2( p(n) ), with n = 0:maxN

% To find p(n), we use p(n) = sum( p(n|s) * p(s) ), with s = 1:5

% Set up vector to store p(n)
pn = zeros(1,(maxN + 1));
for n = 1: (maxN+1)
    A = zeros(1, numberOfStimuli);
    for i = 1:5
        A(i) = ps(i) * pns(i, n);
    end
    pn(n) = sum(A);
end

% Set up vector for the values which need to be summed
B = zeros(1,(maxN+1));

for n = 1: (maxN+1)
    % Preventing NaN from log2(0)
    if pn(n) == 0
        B(n) = 0;
    else
        B(n) = pn(n) * log2( pn(n) );
    end
end

% Compute Entropy, H
H = (-1) * sum(B)
% Output:   H = 5.7485 bits

%H_noise = sum( sum( p(s) * p(n|s) * log2(p(n|s)) )), with s=1:5, n=0:maxN
a = zeros(numberOfStimuli, 1);

for i = 1:5
    b = zeros((maxN+1),1);
    for n = 1:maxN
        if pns(i,n) == 0
            b(n) = 0;
        else
            b(n) = ps(i) * pns(i,n) * log2(pns(i,n));
        end
    end
    a(i) = sum(b);
end

% Compute noise entropy, H_noise
H_noise = -1 * sum(a)
% Output: H_noise = 4.1530

% Calculate mutual information, I = H - H_noise
I = H - H_noise
% Output: I = 1.5955
```

Values of mutual information, I, at the requested presentation times, T.

Pres. Time	Mutual Information
1	1.5955
2	1.9757
3	2.1757
4	2.2568
5	2.316
6	2.3052
7	2.3072



The mutual information increases with the stimulus presentation times until about T = 5 sec, at which point the mutual information plateaus at roughly the value I = 2.3 bits. We then calculated that H_stim = 2.3219, and so it appears that as T increases, the mutual information, I, tends towards H_stim, and it reaches very close to this value by T = 5 sec. (See the attached graph, I've added a red horizontal line to indicate the value of H_stim).

This is interesting, beacuse we know that mutual information, I, is symmetric, i.e. I = H(n) - H(n|s) = H(s) - H(s|n). Hence if I is reaching the value of H(s), it means H(s|n) is zero, i.e. n exactly predicts s. So when we present the stimulus for at least 5 sec, the observed spike count will almost exactly predict the stimulus that was presented.