KTH Royal Institute of Technology

DD2434
Machine Learning Advanced Course

Year 2017 - 2018

Assigment 2

Individual report

Picó Oristrell Sandra 940531 – 2308 sandrapo@kth.se

Professors :
Herman Pawel
Kjellström Hedvig
Lagergren Jens

January 31, 2018

Contents

1 Dependences in a Directed Graphical Model

Important note: In order to get a **E** in this assignment I decided to complete Task 2.1,2.3 and 2.6.

Considering the following Directed Acyclic Graph (DAG) shown below, answer the next two questions:

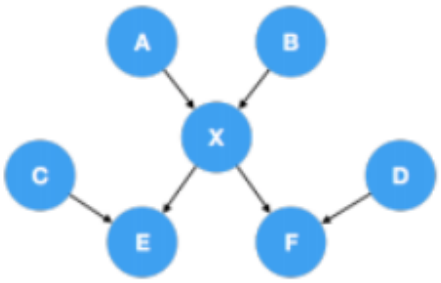


Figure 1: The DAG

***Question 1**: Which pairs of variables, not including X, are dependent conditioned on X?*

In the figure attached above we can see a Directed Acyclic Graphical Model or also called Bayesian Network. In this kind of models, the arrow represents that one node has direct influence over the other. For example, in the graph of this exercise we can state that the node D has direct influence over F.

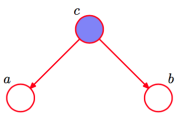
Another term that is important to define is the *conditional independence* term. As Bishop say [1], conditional independence is an important concept for probability distributions over multiple variables. An example of two variables that are conditionally independent is the following:

If:

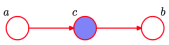
$$p(a|b,c) = p(a|c) \tag{1}$$

then, we state that *a* is conditionally independent of *b* given *c*.

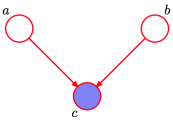
At the same time, there are different graph structures that verify the property described above. However, as explained in [1], there are three specific configurations that can represent the property mentioned. That ones are the following ones:



(a) C as a tail-to-tail node



(b) C as a head-to-tail node



(c) C as a head-to-head node

Figure 2: Conditional independence property in C node

It is also important to state that the mathematical notation used to define conditional independence property is:

$$a \perp\!\!\!\perp b|c \tag{2}$$

For this reason, based on all explained above, the pairs of variables that appear in the graph that are dependent conditioned on X (not including X) is only one: [A,B].

$$A \not\perp\!\!\!\perp B|X$$

However, whether conditioned on X or not, we can also find two other pairs of variables that are dependent:

$$E \not\perp\!\!\!\perp C|X \qquad F \not\perp\!\!\!\perp D|X$$

To sum up the question, we can then state that the pairs of variables that dependent conditioned on X (not including X) are:

- (A,B)
- (E,C)
- (F,D)

***Question 2**: Which pairs of variables, not including X, are dependent, not conditioned on X?*

In order to analyze the graph in a proper way, we are going to study all the different pairs of variables by steps.

First of all, based on all the information already explained in *question 1*, the node X is *tail-to-tail* respect to E and F because the node is connected to the tails of the two arrows.For this reason, if we write the conditional distribution of E and F, given X, we are going to obtain the conditional independence property. However, this question is related to be *not conditioned on X*. So, if we are in the situation where X is not observed, we can test to see if E and F are independent by marginalizing over X. This condition will imply the following:

$$E \not\perp\!\!\!\perp F|\varnothing \tag{3}$$

where $\varnothing$ means not being observed anything. Based on the justification above, we can state that the pair [E,F] are dependent, not conditioned on X.

To continue, now we are going to analyze the *head-to-tail* nodes. In the graph, we find the four pairs of variables that have this configuration related to X: [A,F] , [B,E], [A,E] and [B,F]. Therefore, based on the same reasoning that above, we can state:

$$A \not\perp\!\!\!\perp F|\varnothing \qquad B \not\perp\!\!\!\perp E|\varnothing \qquad A \not\perp\!\!\!\perp E|\varnothing \qquad B \not\perp\!\!\!\perp F|\varnothing$$

And, finally, as we did in the previous question,we can also state that E and F are dependent to C and D respectively, either conditioned or not conditioned on X.

$$E \not\perp\!\!\!\perp C|\varnothing \qquad F \not\perp\!\!\!\perp D|\varnothing$$

To sum up, based on all explained above, the pairs of variables that appear in the graph that are dependent not conditioned on X(not including X) are:

- (E,F)
- (A,F)
- (B,E)
- (B,F)
- (A,E)
- (E,C)
- (F,D)

2 The Sum-HMM

***Question 3**: Implement the Sum-HMM, i.e., write your own code for it.*

I attached the code in the appendix section.

***Question 4**: Provide data generated using at least three different sets of categorical dice distri- butions that provide reasonable tests for the correctness of your program .*

***Question 5**:Motivate your test and why the result of it indicates correctness.*

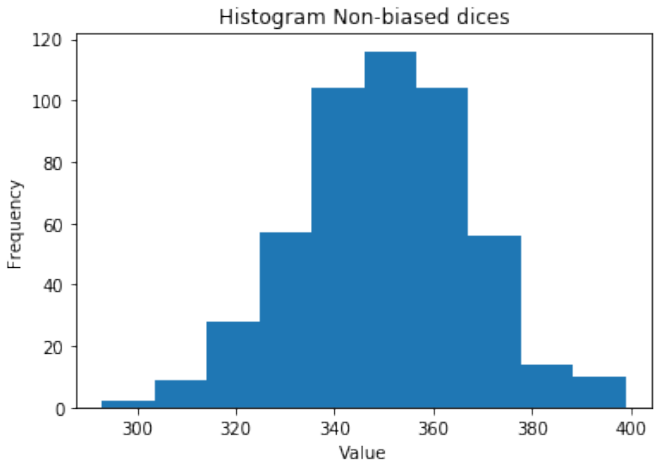
I will answer question 4 and question 5 together. In these questions, in order to check if the program is working properly, we are going to test different cases. Specifically, the tests that we want to check are the following ones:

- Non-biased dice.
- Medium biased dice.
- Total biased dice.

We have to take into account that the original problem was simplified. We are going to consider that the two groups of tables are going to have the same categorical distribution,i.e behave in the same manner. Also, the number of players will be 500 and the number of tables will be 100 for all the cases tested.

2.0.1 Non-biased dices

In the non-biased dices, as the name indicates, we will have the same probability per each number. In other words, $p(number) = \frac{1}{6}$.



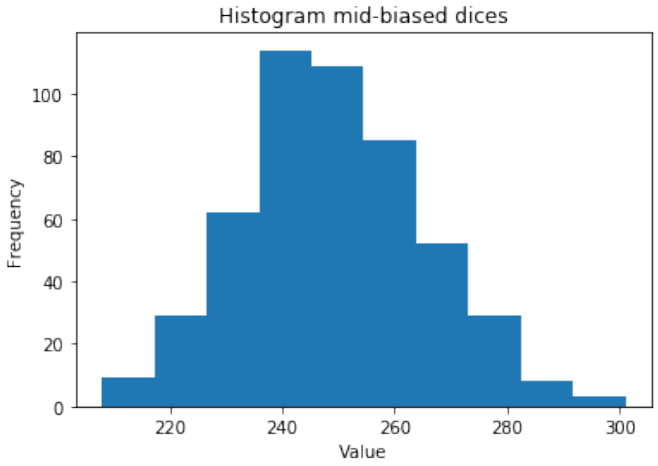
The previous figure reveal the result obtained in the case described. Concretely, we can state that is similar to the Normal distribution. The reason behind that is the central limit theorem. As the number of players is high enough, the random variable tends to behave as the Normal distribution.

Moreover, if we analyze the mean of the distribution we realize that is in the "middle" by the simple fact that we are studying the case where the two groups of tables have uniform distribution.

2.0.2 Medium biased dice

This case, we will have different distributions per each group of tables. The first group will behave as a uniform distribution and the second group will act with a probability of 0.5 for the numbers 1 and 2.

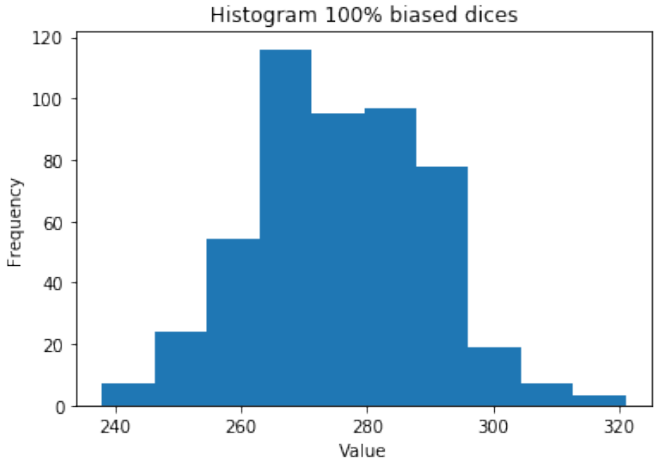
The result of this case is:



If we compare the result obtained in the previous section (2.0.1) with the one obtained now we are going to realize that the mean is displaced. The reason is the distribution for the second group of tables.

2.0.3 Total biased dice

In this case, the second group of tables will only obtain the number 2. In other words, the probability of having a 2 will be $\frac{6}{6}$. However, regarding the first group of tables we will have a uniform distribution. That is the reason why we obtain the following histogram moved to the left side:



3 Simple VI

Consider the model defined by the following equations:

Likelihood function given by:

$$p(\mathcal{D}|\mu,\tau) = (\frac{\tau}{2\pi})^{\frac{N}{2}} exp - \frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2 \tag{4}$$

Conjugate prior distributions for μ and τ given by:

$$p(\mu|\tau) = \mathcal{N}(\mu|\mu_0,(\lambda_0\tau)^{-1}) \tag{5}$$

$$p(\tau) = Gam(\tau|a_0,b_0) \tag{6}$$

where Gam is the gamma distribution defined by:

$$Gam(\lambda|a,b) = \frac{1}{(a)}b^a\lambda^{a-1}exp(-b\lambda). \tag{7}$$

***Question 8**:Implement the VI algorithm for the variational distribution in Equation (10.24) in Bishop.*

In order to implement the VI algorithm for the variational distribution we need to understand how the algorithm works.In other words, understand the theory in a proper way. For this reason, I will explain a little bit the formulas used. However, the whole python implementation is attached in the appendix section.

The final goal is to infer the posterior $p(\mu,\tau|D)$ where $\tau = \frac{1}{\sigma^2}$ represents the precision. At the same time, we need to take into consideration the conjugate prior distributions described previously in the description of this section.

In order to infer the posterior, we are going to use a factorized variational approximation. This approximation is follows by the Equation (10.24) stated in Bishop.

$$q(\mu,\tau) = q_\mu(\mu)q_\tau(\tau) \tag{8}$$

Then, in order to obtain the approximation we will need to compute two differentiated steps: updating $q_\mu(\mu)$ and updating $q_\tau(\tau)$.

In order to find the optimal form for $q_\mu(\mu)$ we need to compute the following equations:

$\log q_\mu(\mu) = -\frac{\mathbb{E}_{q\tau}[\tau]}{2}(\lambda_0(\mu-\mu_0)^2 + \sum_{n=1}^{N}(x_n-\mu_0)^2) + const$

After completing the square we can also understand that $q_\mu(\mu)$ is a Gaussian. All the steps followed can be stated through the code.

On the other hand, to achieve the optimal form of $q_\tau(\tau)$ we will need to compute the next equation. Here, we are going to recognize the log of a Gamma distribution.

$\log q_\tau(\tau) = (a_0-1)\log\tau - b_0\tau + \frac{N}{2}\log\tau - \frac{\tau}{2}\mathbb{E}_{q\mu}(\lambda_0(\mu-\mu_0)^2 + \sum_{n=1}^{N}(x_n-\mu_0)^2) + const$

After finishing all the previous steps and computing the parameters $a_N, b_N, \mu_N, \lambda_N$ using the equations also found in Bishop, we are going to have the optimal distributions for $q_\tau(\tau)$ and $q_\mu(\mu)$. Then, after doing an initial guess for $\mathbb{E}_{q\tau}[\tau]$ and computing again $q_\tau(\tau)$ we are going to extract $\mathbb{E}_{q\mu}[\mu]$ and $\mathbb{E}_{q\mu}[\mu^2]$ and compute the distribution $q_\lambda(\lambda)$.

***Question 9**:Describe the exact posterior*

To describe the exact posterior we are going to start with the likelihood stated in the description of this task.

$$p(\mathcal{D}|\mu,\tau) = (\frac{\tau}{2\pi})^{\frac{N}{2}} exp - \frac{\tau}{2}\sum_{n=1}^{N}(x_n-\mu)^2. \tag{9}$$

We can re-write this equation through adding the sample mean ($\bar{x}$) and the sample covariance (s). Applying that, we are going to express the likelihood as:

$$p(\mathcal{D}|\mu,\tau) = (\frac{\tau}{2\pi})^{\frac{N}{2}} exp(-\frac{\tau}{2}(N(\mu-\bar{x})^2+Ns)) \tag{10}$$

Where $\bar{x}$ and *s* are defined as:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{11}$$

$$s = \frac{1}{N}\sum_{n=1}^{N}(x_n-\bar{x})^2 \tag{12}$$

Therefore, as the description also indicates, the conjugate prior distributions follow Gaussian-Gamma distribution.That is why the joint probability will be:

$$p(\mu,\tau) = p(\mu|\tau)p(\tau) = \frac{b^a\sqrt{\lambda_0}}{\Gamma(a)\sqrt{2\pi}}\tau^{a_0-\frac{1}{2}} exp(-b_0\tau) exp(-\frac{\lambda_0\tau(\mu-\mu_0)^2}{2}) \tag{13}$$

The fact of having a Gaussian likelihood and also, as defined before, a prior of Gaussian-Gamma implies that the posterior distribution will be also a Normal-Gamma distribution. We can state that combining the likelihood described and the priors mentioned:

$$p(\mu,\tau|D) \propto \tau^{N/2+a_0-1/2} exp\left(-\tau(b_0-\tfrac{1}{2}Ns)\right) exp\left\{-\tfrac{\tau(\lambda_0(\mu-\mu_0)^2+N(\bar{x}-\mu)^2)}{2}\right\}$$

Then, if we complete the squares in the last exponential, we are going to obtain:

$$\lambda_0(\mu-\mu_0)^2+N(\bar{x}-\mu)^2 = \lambda_0\mu^2+\lambda_0\mu_0^2-2\lambda_0\mu\mu_0+N\bar{x}+N\mu^2-2N\bar{x}\mu = ... =$$
$$= (\lambda_0+N)\left(\mu-\tfrac{\lambda_0\mu_0+N\bar{x}}{\lambda_0+N}\right)^2+\tfrac{\lambda_0 N(\bar{x}-\mu_0)^2}{\lambda_0+N}$$

The next step is using the previous result and substitute it in the posterior formula.

$$p(\mu,\tau|D) \propto \tau^{N/2+a_0-1/2} exp\left\{-\tau\left(\tfrac{1}{2}Ns+b_0+\tfrac{\lambda_0 N(\bar{x}-\mu_0)^2}{\lambda_0+N}\right)\right\}$$
$$x exp\left\{-\tfrac{\tau}{2}(\lambda_0+N)\left(\mu-\tfrac{\lambda_0\mu_0+N\bar{x}}{\lambda_0+N}\right)^2\right\}$$

Moreover, from the previous equation we can also discover the parameters of the posterior distribution. As said before, the posterior will be also a Normal-Gamma distribution.

$$\mu_{posterior} = \frac{\lambda_0\mu_0+N\bar{x}}{\lambda_0+N} \tag{14}$$

$$\lambda_{posterior} = \lambda_0+N \tag{15}$$

$$a_{posterior} = a_0+\frac{N}{2} \tag{16}$$

$$b_{posterior} = b_0 + \frac{1}{2}\left(Ns + \frac{\lambda_0 N(\bar{x} - \mu_0)^2}{\lambda_0 + N}\right) \tag{17}$$

***Question 10**:Compare the variational distribution with the exact posterior. Run the inference for a couple of interesting cases and describe the difference.*

The main goal of this question is to compare the variational distribution with the exact posterior. However, it is also important to identify a couple of interesting cases to test and extract conclusions from them. Specifically, I will comment three cases/plots.

The main intention for plotting three different cases is to realize about the importance of the number of data points. We are going to see how the variational distribution is improving as the number of samples is increasing. Concretely, the first case will test using a N = 20, the second case using a N = 100 and the other case, using a N = 900.

The variational distribution improves with the number of samples by the simple fact that, as the prior is not good, we need more datapoints to achieve a good variational distribution.

However, in order to understand how the variational inference algorithm work, I also plot four images in each case. Each of the images represent different moments in the execution of the algorithm (a,b,c,d). The first image is representing the initial factorized approximation. The second and the third image are referring to the moment after updating $q_\mu(\mu)$ and $q_\tau(\tau)$ respectively. To conclude, the forth image is the one obtained at the end of the algorithm with 20 iterations.

It has to be also take into account that the plots attached below are for a simple example with the following parameters:

$$\tau = 2.2 \qquad \mu = 1 \qquad \lambda_0 \text{ and } \mu_0 = 1 \qquad a_0 \text{ and } b_0 = 0.01$$

With this initialization we are close to have non-informative priors. Obviously, if we try values closer to the real posterior we are going to have better results in the VI. The reason behind that is because it will imply easier convergence.

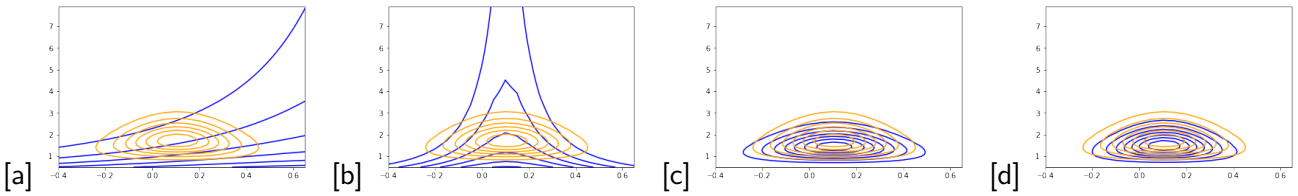


Figure 3: Case 1: Four plots(a,b,c,d) corresponding to the evolution of the VI algorithm using N = 20.(X axis = mu, Y axis = tau). The estimated posterior is represented by the color blue and computed through the VI algorithm, while the exact posterior is plotted in orange.The number of iterations used in the algorithm is 20. (a) Initial factorized approximation, (b) Iteration 1 after updating $q_\mu(\mu)$, (c) Iteration 1 after updating $q_\tau(\tau)$ and (d) Last iteration.

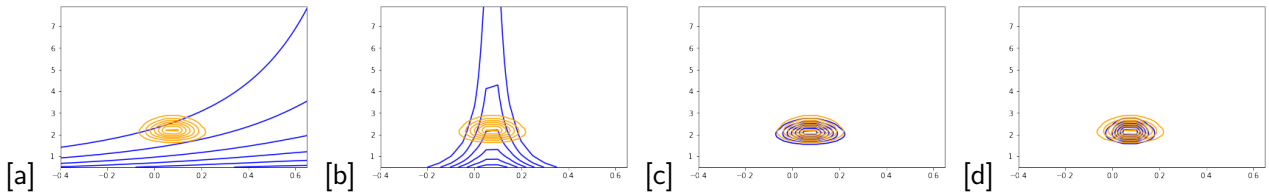


Figure 4: Case 2: Four plots corresponding to the evolution of the VI algorithm using N = 100.(X axis = mu, Y axis = tau). The estimated posterior is represented by the color blue and computed through the VI algorithm, while the exact posterior is plotted in orange.The number of iterations used in the algorithm is 20. (a) Initial factorized approximation, (b) Iteration 1 after updating $q_\mu(\mu)$, (c) Iteration 1 after updating $q_\tau(\tau)$ and (d) Last iteration.

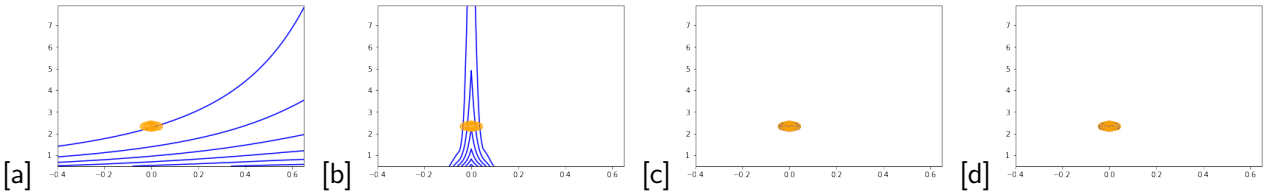


Figure 5: Case 3: Four plots corresponding to the evolution of the VI algorithm using N = 900. (X axis = mu, Y axis = tau). The estimated posterior is represented by the color blue and computed through the VI algorithm, while the exact posterior is plotted in orange.The number of iterations used in the algorithm is 20. (a) Initial factorized approximation, (b) Iteration 1 after updating $q_\mu(\mu)$, (c) Iteration 1 after updating $q_\tau(\tau)$ and (d) Last iteration.

4 Variational Inference

***Question 15**:Present the algorithm written down in a formal manner (using both text and mathematical notation, but not pseudo code).*

To sum up all the information described in the description of this task we can state that the Cartesian Matrix Model (CMM) is defined as:

- R rows distribution $\mathcal{N}(\mu_r, \lambda_r^{-1}) : 1 \leq r \leq R$:,each variance λ_r^{-1} is known and each μ_r has a prior distribution $\mathcal{N}(\mu, \lambda^{-1})$
- C column distribution $\mathcal{N}(\xi_r, \tau_c^{-1}) : 1 \leq c \leq C$ each variance τ_c^{-1} is known and each ξ_c has a prior distribution $\mathcal{N}(\xi, \tau^{-1})$
- Hyper-parameters known.
- Matrix S is generated by $S_{rc} = X_r + Y_c$
- Use Variational Inference to obtain variational distribution :

$$q(\mu_1, ..., \mu_R, \xi_1, ..., \xi_C) = \prod_r q(\mu_r) \prod_c q(\xi_c) \tag{18}$$

To continue, we can define the distribution S_{rc} through the sum of the two Gaussian variables. The reason behind is because we know that from a sum of gaussians we are going to achieve another gaussian.

$$S_{rc} \sim \mathcal{N}(\mu_r + \xi_c, \lambda_r^{-1} + \tau_c^{-1}). \tag{19}$$

Then, we can define the joint distribution as:

$$p(S, \mu_1, ..., \mu_R, \xi_1, ..., \xi_C) = p(S|\mu_1, ..., \mu_R, \xi_1, ..., \xi_C) p(\mu_1, ..., \mu_R) p(\xi_1, ..., \xi_C) = ... \tag{20}$$

$$\prod_{r=1}^{R} \prod_{c=1}^{C} \mathcal{N}(S_{rc}|\mu_r + \xi_c, \lambda_r^{-1} + \tau_c^{-1}) \prod_{r=1}^{R} \mathcal{N}(\mu_r|\mu, \lambda_r^{-1}) \prod_{c=1}^{C} \mathcal{N}(\xi_c|\xi, \tau^{-1}). \tag{21}$$

Also, we need to state that in order to obtain the previous joint distribution the assumption of rows and column distributions being independent has been made.

Then, in order to obtain the optimized factors, we are going to compute the log form of the join probability.

$$lnp(S, \mu_1, ..., \mu_R, \xi_1, ..., \xi_C) \tag{22}$$

Moreover, we are going to define the overall variance as $\beta_{rc}^{-1} = \lambda_r^{-1} + \tau_c^{-1}$

Based on what we said,the log form of the join probability is equal to:

$\sum_{c=1}^{C}\sum_{r=1}^{R}\ln\mathcal{N}(S_{rc}|\mu_r+\xi_c,\lambda_r^{-1}+\tau_c^{-1}) + \sum_{r=1}^{R}\ln\mathcal{N}(\mu_r|\mu,\lambda^{-1}) + \sum_{c=1}^{C}\ln\mathcal{N}(\xi_c|\xi,\tau^{-1}) =$

$\sum_{c=1}^{C}\sum_{r=1}^{R}(\frac{1}{2}\ln\beta_{rc} - \frac{1}{2}\ln(2\pi) - \frac{\beta_{rc}}{2}(S_{rc}-\mu_r-\xi_c)^2) + \sum_{r=1}^{R}(\frac{1}{2}\ln\lambda - \frac{1}{2}\ln(2\pi) - \frac{\lambda}{2}(\mu_r-\mu)^2)$

$+ \sum_{c=1}^{C}(\frac{1}{2}\ln\tau - \frac{1}{2}\ln 2\pi - \frac{\tau}{2}(\xi_c-\xi)^2)$

And then, we are able to define ln q*(μ_r) and ln q*(ξ_c). First of all we are going to calculate ln q*(μ) and then the ln q*(ξ_c).

In order to calculate the first optimized factor we are going to compute the expectation of the log joint probability over all parameters(except of μ_r).

$$
\begin{aligned}
lnq_r^*(\mu_r) &= E_{i\neq r}[lnp(\mu_1,...,\mu_R,\xi_1,...,\xi_C,S)] + const = \\
&= E_{i\neq r}\left[ln\left(\prod_{r=1}^{R}\prod_{c=1}^{C}N(S_{rc}|\mu_r+\xi_c,\lambda_r^{-1}+\tau_c^{-1})\prod_{r=1}^{R}N(\mu_r|\mu,\lambda^{-1})\prod_{c=1}^{C}N(\xi_c|\xi,\tau^{-1})\right)\right] + const = \\
&= E_{i\neq r}\left[\sum_{r=1}^{R}\sum_{c=1}^{C}lnN(S_{rc}|\mu_r+\xi_c,\lambda_r^{-1}+\tau_c^{-1}) + \sum_{r=1}^{R}lnN(\mu_r|\mu,\lambda^{-1}) + \sum_{c=1}^{C}lnN(\xi_c|\xi,\tau^{-1})\right] + const =
\end{aligned}
\tag{23}
$$

And then, calculating the normal distributions:

$$
\begin{aligned}
&= E_{i\neq r}\left[\sum_{r=1}^{R}\sum_{c=1}^{C}-\frac{(S_{rc}-\mu_r-\xi_c)^2}{2(\lambda_r^{-1}+\tau_c^{-1})} + \sum_{r=1}^{R}-\frac{(\mu_r-\mu)^2}{2\lambda^{-1}} + \sum_{c=1}^{C}-\frac{(\xi_c-\xi)^2}{2\tau^{-1}}\right] + const \\
&= -\frac{1}{2}E_{i\neq r}\left[\sum_{r=1}^{R}\sum_{c=1}^{C}\frac{(S_{rc}-\mu_r-\xi_c)^2}{\lambda_r^{-1}+\tau_c^{-1}} + \sum_{r=1}^{R}\lambda(\mu_r-\mu)^2 + \sum_{c=1}^{C}\tau(\xi_c-\xi)^2\right] + const \\
&= -\frac{1}{2}E_{i\neq r}\left[\sum_{c=1}^{C}\left(\frac{(S_{rc}-\mu_r-\xi_c)^2}{\lambda_r^{-1}+\tau_c^{-1}} + \lambda(\mu_r-\mu)^2\right)\right] + const
\end{aligned}
\tag{24}
$$

To continue, we are only keeping the μ_r-dependent terms and taking into account $a_{rc} = \frac{1}{\lambda_r^{-1}+\tau_c^{-1}}$ we can continue with the calculus of the squares:

$$
\begin{aligned}
&= -\frac{1}{2}E_{i\neq r}\left[\sum_{c=1}^{C}a_{rc}(S_{rc}^2 - 2S_{rc}(\mu_r+\xi_c) + \mu_r^2 + 2\mu_r\xi_c + \xi_c^2) + \lambda(\mu_r^2 - 2\mu\mu_r + \mu^2)\right] + const \\
&= -\frac{1}{2}E_{i\neq r}\left[\sum_{c=1}^{C}(a_{rc}+\lambda)\mu_r^2 + 2(a_{rc}(\xi_c - S_{rc}) - \mu\lambda)\mu_r\right] + const \\
&= -\frac{1}{2}\left[\left(\sum_{c=1}^{C}(a_{rc}+\lambda)\right)\mu_r^2 + 2\left(\sum_{c=1}^{C}(a_{rc}(E[\xi_c] - S_{rc}) - \mu\lambda)\right)\mu_r\right] + const \\
&= -\frac{1}{2}\left(\sum_{c=1}^{C}a_{rc} + C\lambda\right)\mu_r^2 - \left(\sum_{c=1}^{C}a_{rc}(E[\xi_c] - S_{rc}) - C\mu\lambda\right)\mu_r + const
\end{aligned}
\tag{25}
$$

With all the expressions stated above, we can conclude:

$$\mu_r \sim N(a_r, b_r) \tag{26}$$

$$b_r = \left(\sum_{c=1}^{C} a_{rc} + C\lambda \right)^{-1} \tag{27}$$

$$a_r = b_r \left(\sum_{c=1}^{C} a_{rc}(S_{rc} - E[\xi_c]) + C\mu\lambda \right) \tag{28}$$

$$\xi_c \sim N(a_c, b_c) \tag{29}$$

$$b_c = \left(\sum_{r=1}^{R} a_{rc} + R\tau \right)^{-1} \tag{30}$$

$$a_c = b_c \left(\sum_{r=1}^{R} a_{rc}(S_{rc} - E[\mu_r]) + R\xi\tau \right) \tag{31}$$

Appendices

A Code question 3

```
import numpy as np
import matplotlib.pyplot as plt

def Draw(total_sum):

    plt.title("Histogram mid-biased dices")
    plt.hist(total_sum)
    plt.xlabel("Value")
    plt.ylabel("Frequency")
    plt.show()

#Number of players and number of tables for all the experiments.
num_players = 500
num_tables = 100


#Define all the possible distributions.
uniform_distribution = [1./6,1./6,1./6,1./6,1./6,1./6]
mid_bias_distribution = [0.5,0.5,0,0,0,0]
biased_distribution = [0,1,0,0,0,0] #In this case the probability of the number 2 will ...
    be 6/6.

#HMM

# Create transition matrix ( probabilities of going to different groups of tables (1/4 ...
    and 3/4))
A = [[0.25, 0.75], [0.75, 0.25]]
#Define the emission matrix.
B = [uniform_distribution, mid_bias_distribution]
#Initial conditions. We can be either in one group of tables or in the other.
```

```
pi = [0.5, 0.5]


# Alpha-pass algorithm
total_sum = np.zeros(num_players)
for player in range(num_players):
    pi = [0.5, 0.5]
    output_distribution = np.dot(pi, B)
    total_sum[player] = total_sum[player] + np.random.choice(a=[1, 2, 3, 4, 5, 6], ...
        p=output_distribution)
    for timestep in range(1,num_tables):
        pi = np.dot(pi,A)
        output_distribution = np.dot(pi,B)
        total_sum[player] = total_sum[player] + np.random.choice(a=[1, 2, 3, 4, 5, 6], ...
            p = output_distribution)

Draw(total_sum)
```

B Code question 10

```
#Import libraries
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import sys

#Create the data
def GenerateData(mu,tau,N):
    x = np.random.normal(loc=mu, scale=tau**(-0.5), size=N)
    return x

#Prior distribution of tau will be the Gaussian-Gamma distribution.
def normal_gamma(mean, precision, shape, rate):
    normal = stats.norm(loc=mean, scale=(precision*range_tau)**(-.5))
    gamma = stats.gamma(a=shape, scale=1.0/rate)
    return [[normal.pdf(u)[t]*gamma.pdf(range_tau[t])
    for u in range_mu] for t in range(len(range_tau))]

#VI algorithm. Following Bishop equations.
def VIalgorithm(a_init,b_init,mu_init,lambda_init,N,data,its):
    sample_mean = np.mean(x)
    sample_var = np.var(x)
    a_N = a_init
    b_N = b_init
    vi_posterior= []
    for i in range(20):
        mean_tau = a_N/b_N
        mu_N = (lambda_init*mu_init+N*sample_mean)/(lambda_init+N)
        lambda_N = (N+lambda_init)*mean_tau
        vi_posterior1 = normal_gamma(mu_N,lambda_N,a_N,b_N)
        mean_mu = mu_N
        mean_mu2 = mu_N**2 + 1/lambda_N
        a_N = a_init+(N+1)/2
        b_N = ...
            b_init+.5*(mean_mu2*(N+lambda_init)-2*mean_mu*(N*sample_mean+lambda_init*mu_init)+lambda
        vi_posterior2 = normal_gamma(mu_N,lambda_N,a_N,b_N)
        if (i < 1):
            vi_posterior.append(vi_posterior1)
            vi_posterior.append(vi_posterior2)
    vi_posterior.append(vi_posterior2)
    return vi_posterior
```

```
41
42 #Initialize the prior parameters.
43 #In that case, non-informative priors.
44 def priorInit():
45     mu_init = 1
46     lambda_init = 1
47     a_init = 0.01
48     b_init = 0.01
49     return a_init,mu_init,lambda_init,b_init
50
51 #Calculate the exact posterior.
52 def ExactPosterior(mu_init,lambda_init,a_init,b_init,N,data):
53     sample_mean = np.mean(data)
54     sample_var = np.var(data)
55     mu_posterior = (lambda_init*mu_init+N*sample_mean)/(lambda_init + N)
56     lambda_posterior = lambda_init+N
57     a_posterior = a_init + N/2
58     b_posterior = b_init + 0.5*(N*sample_var + ...
           (lambda_init*N*(sample_mean-mu_init))/(lambda_init+N))
59     real_posterior = normal_gamma(mu_posterior,lambda_posterior,a_posterior,b_posterior)
60     return real_posterior
61
62 def DrawGraphs(prior,real_posterior,vi_posterior):
63     fig = plt.figure()
64     plt.contour(prior,colors = 'blue')
65     plt.contour(real_posterior,colors ='orange')
66     plt.show()
67     for j in range(0,3):
68         fig = plt.figure()
69         plt.contour(vi_posterior[j],colors='blue')
70         plt.contour(real_posterior,colors='orange')
71         plt.show()
72
73 def Main():
74
75     #Parameters
76     mu = 0
77     tau = 2.2
78     N_data = 100
79     iterations = 20
80     data = GenerateData(mu,tau,N_data)
81     #To compute the graphs
82     range_mu = np.arange(-.4,0.7,0.05)
83     range_tau = np.arange(0.5,8,0.1)
84     sample_mean = np.mean(data)
85     sample_var = np.var(data)
86     a_init,mu_init,lambda_init,b_init = priorInit()
87     prior = normal_gamma(mu_init,lambda_init,a_init,b_init)
88     exact_posterior = ExactPosterior(mu_init,lambda_init,a_init,b_init,N_data,data)
89     infer_posterior = ...
           VIalgorithm(a_init,b_init,mu_init,lambda_init,N_data,data,iterations)
90     DrawGraphs(prior,exact_posterior,infer_posterior)
91
92 if __name__ == "__main__":
93     Main()
```

References

[1] C. M.Bishop, "Pattern recognition and machine learning," 2009.