COMP0080 Graphical Models
Assignment 1

Daniel May
ucabdd3@ucl.ac.uk

Olivier Kraft
ucabpk0@ucl.ac.uk

Daniel O'Hara
daniel.o'hara.20@ucl.ac.uk

Stephen O'Brien
stephen.o'brien.20@ucl.ac.uk

Umais Zahid
umais.zahid.16@ucl.ac.uk

November 17, 2020

Contents

1 Question 1

$$\text{Three cards} \begin{cases} \text{both sides white (W)} \\ \text{both sides black (B)} \\ \text{one side each colour} \end{cases}$$

If the upper side of a randomly drawn card is black, what is distribution of lower side? By Bayes' Theorem,

$$P(Lower = W|Upper = B) = \frac{P(Lower = W, Upper = B)}{P(Upper = B)}$$

We know that:

$$P(Lower = W, Upper = B) = \frac{1}{6} \text{ (only 1 card has one side of each colour)}$$
$$P(Upper = B) = \frac{1}{2} \text{ (3 of 6 possible card faces are black)}$$

Plugging these in,

$$P(Lower = W|Upper = B) = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3}$$
$$\Rightarrow P(Lower = B|Upper = B) = 1 - \frac{1}{3} = \frac{2}{3}$$

Hence, given that the upper face is black, the lower face has distribution

$$Lower = \begin{cases} \text{White} & \text{with probability } 1/3 \\ \text{Black} & \text{with probability } 2/3 \end{cases}$$

2 Question 2

(a)

(a).1

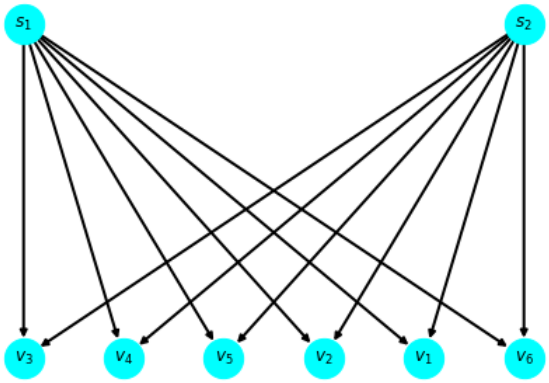


Figure 1: The representation of the belief graph for the explosions, s_1 and s_2, and the measurements $\{v_i\}_{i=1}^n$ (With $n = 6$ in this case).

First, we must make independence assumptions about the explosions and sensors. These can be seen in fig. [1]. We note that the locations, e_1 and e_2, of the two explosions are independent of each other. We also note that the reading on each of the sensors are dependent on the locations of both of the explosions.

From this belief network, we can write the joint pdf as

$$P(s_1, s_2, \bar{v}) = P(s_1)P(s_2)\prod_{i=1}^{n} P(v_i|s_1, s_2) \quad \text{with} \quad \bar{v} = \{v_i\}_{i=1}^n$$

For this problem, we assume the measurements from the sensors are distributed as follows where $d_i(1)$ and $d_i(2)$ are the distances from the i^{th} sensor to the 1^{st} and 2^{nd} explosions respectively.

$$v_i \sim \mathcal{N}\left(\mu = \frac{1}{d_i^2(1) + 0.1} + \frac{1}{d_i^2(2) + 0.1}, \sigma = 0.2\right)$$

This means that

$$P(v_i|s_1, s_2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2\sigma^2}\left(v_i - \frac{1}{d_i^2(1)+0.1} + \frac{1}{d_i^2(2)+0.1}\right)^2}$$

We ultimately seek $P(s_1|\bar{v})$. To get this we must first calculate $P(s_1, s_2|\bar{v})$ and then marginalise out s_2.

$$P(s_1|\bar{v}) = \sum_{s_2} P(s_1, s_2|\bar{v})$$

From Bayes' Theorem we have

$$P(s_1, s_2|\bar{v}) = \frac{P(s_1, s_2, \bar{v})}{\sum_{s_1,s_2} P(s_1, s_2, \bar{v})} = \frac{P(s_1)P(s_2)\prod_{i=1}^{n} P(v_i|s_1, s_2)}{\sum_{s_1,s_2} P(s_1)P(s_2)\prod_{i=1}^{n} P(v_i|s_1, s_2)}$$

Marginalising out s_2 we get

$$P(s_1|\bar{v}) = \frac{\sum_{s_2} P(s_1)P(s_2)\prod_{i=1}^{n} P(v_i|s_1,s_2)}{\sum_{s_1,s_2} P(s_1)P(s_2)\prod_{i=1}^{n} P(v_i|s_1,s_2)}$$

Since we assume a uniform prior on the locations of the explosions, we must set

$$P(s_1) = P(s_2) = \frac{1}{m}$$

Where m is the number of possible discretized explosion locations. Thus we have

$$P(s_1|\bar{v}) = \frac{\sum_{s_2} \prod_{i=1}^{n} P(v_i|s_1,s_2)}{\sum_{s_1,s_2} \prod_{i=1}^{n} P(v_i|s_1,s_2)}$$

This distribution was calculated numerically and can be visualised in fig.[2].

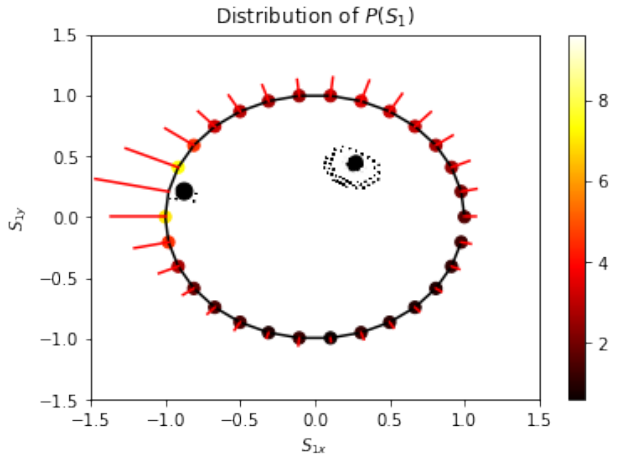


Figure 2: The posterior distribution $P(S|V)$. The dots each represent the a potential location for the explosion. The dots are scaled according to the calculated posterior distribution. The red lines represent the observed sensor values, as does the colours of the sensors.

(a).2

We now have the following hypotheses:

$\mathcal{H}_1$: The hypothesis that there is only one explosion
$\mathcal{H}_2$: The hypothesis that there are two explosions

We take a Bayesian modelling approach to this question. The posterior distribution looks like

$$P(\bar{v}|s_k,\mathcal{H}_k) = \frac{P(\bar{v}|s_k,\mathcal{H}_k)P(s_k|\mathcal{H}_k)}{P(\bar{v}|\mathcal{H}_k)}$$

It is important to note here that

$$s_k = \begin{cases} s & \text{under } \mathcal{H}_1 \\ (s_1, s_2) & \text{under } \mathcal{H}_2 \end{cases}$$
$$P(s_k|\mathcal{H}_k) = \begin{cases} \frac{1}{m} & \text{under } \mathcal{H}_1 \\ \frac{1}{m^2} & \text{under } \mathcal{H}_2 \end{cases}$$

We get the denominator $P(\bar{v}|\mathcal{H}_k)$ by marginalising out s_k as follows. We note here that $P(s_k|\mathcal{H}_k)$ is constant $\forall$ s_k due to the uniform nature of the prior and can therefore be factored out of the sum.

$$P(\bar{v}|\mathcal{H}_k) = P(s_k|\mathcal{H}_k) \sum_{s_k} P(\bar{v}|s_k, \mathcal{H}_k)$$

Under $\mathcal{H}_k$, each v_i is mutually conditionally independent given the location(s) of the explosion(s). Thus

$$P(\bar{v}|\mathcal{H}_k) = P(s_k|\mathcal{H}_k) \sum_{s_k} \prod_{i=1}^{n} P(v_i|s_k, \mathcal{H}_k)$$

This was calculated numerically in python and the desired value was found to be

$$\log P(\bar{v}|\mathcal{H}_2) - \log P(\bar{v}|\mathcal{H}_1) = 735.68$$

(a).3

From Bayes' Theorem we have

$$P(\mathcal{H}_k|\bar{v}) = \frac{P(\bar{v}|\mathcal{H}_k)P(\mathcal{H}_k)}{P(\bar{v})}$$

We can create a metric to measure the likelihood of one hypothesis over another as follows

$$\frac{P(\mathcal{H}_2|\bar{v})}{P(\mathcal{H}_1|\bar{v})} = \frac{\frac{P(\bar{v}|\mathcal{H}_2)P(\mathcal{H}_2)}{P(\bar{v})}}{\frac{P(\bar{v}|\mathcal{H}_1)P(\mathcal{H}_1)}{P(\bar{v})}}$$

The larger this value is, the more likely $\mathcal{H}_2$ is over $\mathcal{H}_1$. If we have no prior preference, that is $P(\mathcal{H}_1) = P(\mathcal{H}_2) = \text{const.}$, we are left with

$$\frac{P(\mathcal{H}_2|\bar{v})}{P(\mathcal{H}_1|\bar{v})} = \frac{P(\bar{v}|\mathcal{H}_2)}{P(\bar{v}|\mathcal{H}_1)}$$

Taking the log of both sides, we get

$$\log\left(\frac{P(\mathcal{H}_2|\bar{v})}{P(\mathcal{H}_1|\bar{v})}\right) = \log P(\bar{v}|\mathcal{H}_2) - \log P(\bar{v}|\mathcal{H}_1)$$

Since $\log(x)$ is a monotonically increasing function for $x \in \mathbb{R}^+$, we can say that the larger $\log P(\bar{v}|\mathcal{H}_2) - \log P(\bar{v}|\mathcal{H}_1)$ is, the more likely $\mathcal{H}_2$ is over $\mathcal{H}_1$. In other words, the more likely there are two explosions rather than one. In the case of this question, it was found that it was extremely likely that there were two explosions as a value of 735.68 was recorded.

(a).4

The computational cost would be $O(m^K)$, where m is the total number of possible explosion locations considered. This is because a table containing each possible combination of explosions must be considered. This table is of dimension $\underbrace{m \times m \times ... \times m}_{\text{K times}}$ so each additional explosion considered raises the complexity by a factor of m.

(b)

(b).1

The posterior distribution $P(s_1|\bar{v})$ was calculated in the same manner as in part 'a'. The only difference is that we assumethe measurements from the sensors are distributed as follows where $d_i(1)$ and $d_i(2)$ are the distances from the i^{th} sensor to the 1^{st} and 2^{nd} explosions respectively.

$$v_i \sim \mathcal{N}\left(\mu = \frac{0.5}{d_i^2(1)+0.1} + \frac{0.5}{d_i^2(2)+0.1}, \sigma = 0.2\right)$$

This means that

$$P(v_i|s_1,s_2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2\sigma^2}\left(v_i - \frac{0.5}{d_i^2(1)+0.1} + \frac{0.5}{d_i^2(2)+0.1}\right)^2}$$

Just like before, the joint distribution $P(s_1, s_2|\bar{v})$ was calculated and then s_2 was marginalised out. This distribution was numerically calculated by discretizing the possible explosion locations and a visualisation can be seen in fig [3].

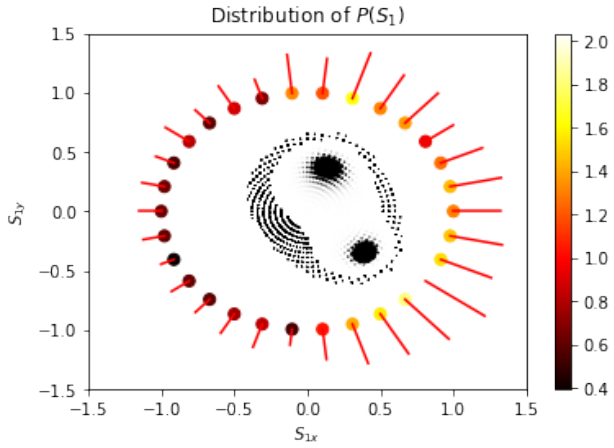


Figure 3: The posterior distribution $P(S|V)$. The dots each represent the a potential location for the explosion. The dots are scaled according to the calculated posterior distribution. The red lines represent the observed sensor values, as does the colours of the sensors.

(b).2

The value of $\log P(\bar{v}|\mathcal{H}_2) - \log P(\bar{v}|\mathcal{H}_1)$ was calculated numerically just like before and the result was

$$\log P(\bar{v}|\mathcal{H}_2) - \log P(\bar{v}|\mathcal{H}_1) = 15.9659$$

(b).3

It would be more difficult as using the mean of the two explosions rather than the sum halves the observed value of the sensor. However, the noise term/error of each sensor remains of constant variance so has a larger relative effect compared to the previous method of adding the two explosions' magnitude.
If there were two locations in close proximity, the averaging method's sensors would look very similar to a single explosion in that area but the summing method's sensors would clearly show that the magnitude was too large to be a single explosion. Therefore, it makes it more difficult to determine both the location and number of explosions.

(c) Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from scipy.stats import norm
import networkx as nx
from networkx.algorithms import bipartite
os.chdir("C:/Users/Steph/OneDrive/Desktop/UCL/Graphical Models")
EarthquakeExerciseData = pd.read_csv("EarthquakeExerciseData.txt",header = None)
EarthquakeExerciseMeanData = pd.read_csv("EarthquakeExerciseMeanData.txt",header = None)
Data1 = np.array(EarthquakeExerciseData)
Data2 = np.array(EarthquakeExerciseMeanData)
s = 2000
n = 30
sd = 0.2


B = nx.DiGraph()
B.add_nodes_from(["$s_2$", "$s_1$"], bipartite=1)
B.add_nodes_from(["$v_1$", "$v_2$", "$v_3$", "$v_4$", "$v_5$", "$v_6$"], bipartite=0)
B.add_edges_from([("$s_1$", "$v_1$"), ("$s_1$", "$v_2$"), ("$s_1$", "$v_3$"),
                  ("$s_1$", "$v_4$"), ("$s_1$", "$v_5$"), ("$s_1$", "$v_6$")])
B.add_edges_from([("$s_2$", "$v_1$"), ("$s_2$", "$v_2$"), ("$s_2$", "$v_3$"),
                  ("$s_2$", "$v_4$"), ("$s_2$", "$v_5$"), ("$s_2$", "$v_6$")])
top = nx.bipartite.sets(B)[0]
pos = nx.bipartite_layout(B, top,align='horizontal')
nx.draw(B,pos,with_labels=True,node_color='cyan',node_size=800,node_outline=True,width=2)

def value(x, y, xsensor, ysensor):
    return (0.1 + (xsensor-x)**2 + (ysensor-y)**2)**(-1)
s = 2000
rate = 25
sd = 0.2
x = np.zeros(s)
y = np.zeros(s)
for i in range(s):
    theta = rate*2*np.pi*i/s
    r = i/s
    x[i] = r*np.cos(theta)
    y[i] = r*np.sin(theta)

v = np.zeros((s, n))
x_sensor, y_sensor = np.zeros(n), np.zeros(n)
```

```
for sensor in range(n):
    theta_sensor = 2*np.pi*sensor/n
    x_sensor[sensor] = np.cos(theta_sensor) # coords of sensors
    y_sensor[sensor] = np.sin(theta_sensor) # coords of sensors
    for k in range(s):
        #D-value for x, y for sensor k
        v[k, sensor] = value(x[k], y[k], x_sensor[sensor], y_sensor[sensor])

D = np.zeros((s, s, n))
for i in range(30):
    for j in range(s):
        for k in range(s):
            #D-value for 2 explosions - V * V.T
            D[j,k,i] = v[j,i] + v[k,i]

p = np.ones((s, s)) # likelihoods of explosion at e1, e2 given D
for i in range(n):
    p[:, :] *= norm.pdf(D[:,:,i], Data1[i], sd)
psum = p.sum(axis = 1)
pscale = psum/psum.max()

plt.scatter(x_sensor, y_sensor, marker = "o",s=50,  c = Data1, cmap = "hot")
plt.colorbar()
plt.plot(x_sensor, y_sensor, color = "black")
plt.scatter(x, y, s = pscale*100,zorder = 0, color = "black")
plt.ylim(-1.5, 1.5)
plt.xlim(-1.5, 1.5)
plt.ylabel("$S_{1y}$")
plt.xlabel("$S_{1x}$")
plt.title("Distribution of $P(S_{1})$")
datanorm = 0.5*Data1/Data1.max()
for i in range(n):
    theta = 2*np.pi *i/n
    plt.plot([x_sensor[i], x_sensor[i] + datanorm[i]*np.cos(theta)],
             [y_sensor[i], y_sensor[i] + datanorm[i]*np.sin(theta)], color = "red")
# H1
ph1 = np.ones(s)
for i in range(n):s
    ph1 *= norm.pdf(v[:,i], Data1[i], sd)
ph1sum = ph1.sum()
ph1scale = ph1/ph1.max()
np.log(psum.sum()/2000) - np.log(ph1sum)
#### Q2.2####
def value(x, y, xsensor, ysensor):
    return 0.5 *(0.1 + (xsensor-x)**2 + (ysensor-y)**2)**(-1)
s = 2000
rate = 25
sd = 0.2
x = np.zeros(s)
y = np.zeros(s)
for i in range(s):
    theta = rate*2*np.pi*i/s
    r = i/s
    x[i] = r*np.cos(theta)
```

```
    y[i] = r*np.sin(theta)
v = np.zeros((s, n))
x_sensor, y_sensor = np.zeros(n), np.zeros(n)
D = np.zeros((s, s, n))
def value(x, y, xsensor, ysensor):
    return 0.5 *(0.1 + (xsensor-x)**2 + (ysensor-y)**2)**(-1)
for sensor in range(n):
    theta_sensor = 2*np.pi*sensor/n
    x_sensor[sensor] = np.cos(theta_sensor) # coords of sensors
    y_sensor[sensor] = np.sin(theta_sensor) # coords of sensors
    for k in range(s):
        #D-value for x, y for sensor k
        v[k, sensor] = value(x[k], y[k], x_sensor[sensor], y_sensor[sensor])
for i in range(30):
    for j in range(s):
        for k in range(s):
            #D-value for 2 explosions - V * V.T
            D[j,k,i] = v[j,i] + v[k,i]
D = np.zeros((s, s, n))
for i in range(30):
    for j in range(s):
        for k in range(s):
            #D-value for 2 explosions - V * V.T
            D[j,k,i] = v[j,i] + v[k,i]

p = np.ones((s, s)) # likelihoods of explosion at e1, e2 given D
for i in range(n):
    p[:, :] *= norm.pdf(D[:,:,i], Data2[i], sd)
psum = p.sum(axis = 1)
pscale = psum/psum.max()
plt.scatter(x_sensor, y_sensor, marker = "o",s=50,  c = Data2, cmap = "hot")
plt.colorbar()
plt.scatter(x, y, s = pscale * 100,zorder = 0, color = "k")
plt.ylabel("$S_{1y}$")
plt.xlabel("$S_{1x}$")
plt.title("Distribution of $P(S_{1})$")
plt.ylim(-1.5, 1.5)
plt.xlim(-1.5, 1.5)
datanorm = 0.5*Data2/Data2.max()
for i in range(n):
    theta = 2*np.pi *i/n
    plt.plot([x_sensor[i], x_sensor[i] + datanorm[i]*np.cos(theta)],
             [y_sensor[i], y_sensor[i] + datanorm[i]*np.sin(theta)], color = "red")
ph1 = np.ones(s)
for i in range(n):
    ph1 *= norm.pdf(2*v[:,i], Data2[i], sd) #remove average affect
ph1sum = ph1.sum()
ph1scale = ph1/ph1.max()
np.log(psum.sum()/2000) - np.log(ph1sum)
```

3 Question 3

(a)

Let S_0 be the time (in minutes) between T_0 and 21:05. We can tabulate the probability distribution for $D_i < S_0$ using the intervals provided:

T_0	S_0	$P(D_i < S_0)$
21:00	5	0.8
20:55	10	0.9
20:50	15	0.97
20:45	20	0.99
N/A	∞	1

In order for the group to catch the train, all friends need to arrive within S_0 of the agreed meeting time. Since the D_i's are independent, the probability that all friends arrive within S_0 is equal to the product of the probabilities that $D_i < S_0, \forall i = 1, 2, ..., N$.

Meeting at the latest possible time is equivalent to finding the smallest value for S_0 such that:

$$P((D_1 < S_0) \cap (D_2 < S_0) \cap ... \cap (D_n < S_0)) \geq 0.9$$
$$\prod_{n=1}^{N} P(D_i < S_0) \geq 0.9 \text{ (by independence)}$$
$$P(D_i < S_0)^N \geq 0.9$$

Therefore with N friends, we wish to find the smallest S_0 such that $P(D_i < S_0) \geq (0.9)^{1/N}$.

- For $N = 3$:
  - The condition for S_0 becomes:
    $$P(D_i < S_0) \geq 0.9654$$
  - The smallest value of S_0 for which this condition is met is 15.
  - Therefore: $T_0(3) = 20{:}50$.

- For $N = 5$:
  - The condition for S_0 becomes:
    $$P(D_i < S_0) \geq 0.9791$$
  - The smallest value of S_0 for which this condition is met is 20.
  - Therefore: $T_0(5) = 20{:}45$.

- For $N = 10$:
  - The condition for S_0 becomes:
    $$P(D_i < S_0) \geq 0.989$$
  - The smallest value of S_0 for which this condition is met is 20.
  - Therefore: $T_0(10) = 20{:}45$.

(b)

Notation: $Z_i = 1$ denotes punctual. $Z_i = 0$ denotes not punctual.

We can update our probability distribution table as follows:

T_0	S_0	$P(D_i < S_0 \mid Z_i = 1)$	$P(D_i < S_0 \mid Z_i = 0)$
21:00	5	0.8	0.7
20:55	10	0.9	0.8
20:50	15	0.97	0.9
20:45	20	0.99	0.95
N/A	∞	1	1

The probability of missing the train is then:

$$\begin{aligned} P(\text{miss}) &= P(\text{at least 1 person delayed}) \\ &= 1 - P(\text{each person arrives within } S_0) \\ &= 1 - \prod_{i=1}^{N} P(D_i < S_0) \\ &= 1 - \prod_{i=1}^{N} \sum_{z} P(D_i < S_0, Z_i) \\ &= 1 - \prod_{i=1}^{N} \sum_{z} P(D_i < S_0 \mid Z_i) P(Z_i) \\ &= 1 - (\sum_{z} P(D_i < S_0 \mid Z_i) P(Z_i))^N \end{aligned}$$

- For $N = 3$, $S_0 = 15$:
  - Using the values from the probability distribution, we obtain:

$$P(\text{miss}) = 1 - (\frac{2}{3} \cdot 0.97 + \frac{1}{3} \cdot 0.9)^3 = 0.152$$

  - Therefore, there is a probability of 0.152 of missing the train.

- For $N = 5$, $S_0 = 20$:
  - Using the values from the probability distribution, we obtain:

$$P(\text{miss}) = 1 - (\frac{2}{3} \cdot 0.99 + \frac{1}{3} \cdot 0.95)^5 = 0.111$$

  - Therefore, there is a probability of 0.111 of missing the train.

- For $N = 10$, $S_0 = 20$:
  - Using the values from the probability distribution, we obtain:

$$P(\text{miss}) = 1 - (\frac{2}{3} \cdot 0.99 + \frac{1}{3} \cdot 0.95)^{10} = 0.210$$

  - Therefore, there is a probability of 0.210 of missing the train.

(c) Bonus

(c).1 Prior distribution

We begin by determining the prior distribution for the number of friends (out of 5) who are not punctual. This is a binomial distribution with parameters $N = 5$ and $p = 1/3$. The probability prob(k) of k friends being non-punctual is therefore equal to:

$$\text{prob}(k) = \binom{N}{k} p^k (1-p)^{N-k}$$

We obtain the following approximate values:

- Prior probability of 0 friend being non-punctual is 0.13.
- Prior probability of 1 friend being non-punctual is 0.33.
- Prior probability of 2 friends being non-punctual is 0.33.
- Prior probability of 3 friends being non-punctual is 0.16.
- Prior probability of 4 friends being non-punctual is 0.04.
- Prior probability of 5 friends being non-punctual is 0.0.

(c).2 Likelihood

Next, we need to determine the likelihood of missing the train given that all friends were asked to arrive by 20:45 (i.e. $S_0 = 20$). The complementary event of missing the train is the scenario where all friends arrive on time. The probability of everyone arriving on time with k friends who are not punctual is equal to the product of:

- all k friends who are not punctual being on time. For each of these friends, the probability of being on time is equal to $P(D_i \leq S_0 | Z_i = 0)$.
- all $N - k$ friends who are punctual being on time. For each of these friends, the probability of being on time is equal to $P(D_i \leq S_0 | Z_i = 1)$.

Using the values for $P(D_i < 20|Z_i)$:

$$P(\text{all on time}) = (P(D_i < 20|Z_i = 0))^k * (P(D_i < 20|Z_i = 1))^{N-k}$$

$$\Rightarrow P(\text{miss}) = 1 - (P(D_i < 20|Z_i = 0))^k * (P(D_i < 20|Z_i = 1))^{N-k}$$

$$\Rightarrow P(\text{miss}) = 1 - (0.95^k * 0.99^{N-k})$$

(c).3 Updating the prior

We can then use Bayes Theorem to compute the posterior distribution of k:

$$P(\text{k friends non-punctual}|\text{miss}) = \frac{\text{p(k friends non-punctual)} * \text{likelihood(miss}|\text{k friends non-punctual)}}{\text{likelihood(miss)}}$$

Considering that the denominator is a normalising constant, we obtain the following approximate values for the posterior distribution of k:

- Posterior probability of 0 friend being non-punctual is 0.06.
- Posterior probability of 1 friend being non-punctual is 0.26.
- Posterior probability of 2 friends being non-punctual is 0.37.

- Posterior probability of 3 friends being non-punctual is 0.24.
- Posterior probability of 4 friends being non-punctual is 0.07.
- Posterior probability of 5 friends being non-punctual is 0.01.

We can observe that the probability distribution has shifted towards higher values of k.

4 Question 4

(a)

(a).1 Notation

We define a_n as the number of season ticket holders attending the n-th match, and a_n as the number of regular fans attending the n-th match. We therefore have:

$$c_n = a_n + b_n$$

(a).2 Distribution of a_n and b_n

a_n follows a binomial distribution with parameters (a, p_a). As a result:

$$E[a_n] = a \times p_a$$
$$Var[a_n] = a \times p_a(1 - p_a)$$

b_n follows a binomial distribution with parameters (b, p_b). As a result:

$$E[b_n] = b \times p_b$$
$$Var[b_n] = b \times p_b(1 - p_b)$$

(a).3 Distribution of c_n

As the sum of two variables with binomial distributions, c_n has a Poisson binomial distribution.

Computing the mean:

$$\begin{aligned} E[c_n] &= E[a_n + b_n] \\ &= E[a_n] + E[b_n] \\ &= \sum_{i=1}^{a} p_a + \sum_{i=1}^{b} p_b \\ &= ap_a + bp_b \end{aligned}$$

Computing the variance:

$$\begin{aligned} Var[c_n] &= Var[a_n + b_n] \\ &= Var[a_n] + Var[b_n] + 2Cov[a_n, b_n] \\ &= Var[a_n] + Var[b_n] \qquad \text{(because } a_n \text{ and } b_n \text{ are independent)} \\ &= a \times p_a(1 - p_a) + b \times p_b(1 - p_b) \end{aligned}$$

Summary:

- The mean of c_n is $ap_a + bp_b$.
- The variance of c_n is $a \times p_a(1 - p_a) + b \times p_b(1 - p_b)$.

(b)

For every ticket holder, let f be the number of fans in the stadium. f can take two values: 1 or 2.

$$\begin{aligned} E[f] &= \sum_{f} fp(f) \\ &= 1 * (1 - p_d) + 2 * p_d \\ &= 1 + p_d \end{aligned}$$

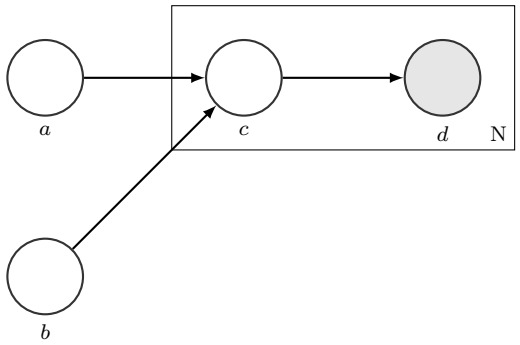
The probability above is the same for all ticket holders. Therefore:

$$\begin{aligned} E[d_n|c_n] &= c_n * E[f] \\ &= c_n(1 + p_d) \end{aligned}$$

The mean of $d_n|c_n$ is $c_n(1 + p_d)$.

(c)

Draw the graphical model corresponding to this setup (the variables are $a, b, c_1, c_2, ..., c_N, d_1, d_2, ..., d_N$). Using plate notation for directed graphs,



(d)

Notation (considering that the subscript n is now used to refer to the last match of the series):

- a_k = number of season ticket holders attending the k-th game
- b_k = number of tickets holders at the k-th game that do not have season ticket
- c_k = number of tickets holders at the k-th game
- d_k = total number of fans at the k-th game (ticket holders plus guests)

(d).1 Probability distribution of a

Based on Bayes' Theorem:

$$p(a|d_1, d_2, ...d_n) = \frac{p(a)p(d_1, d_2, ..., d_n|a)}{p(d_1, d_2, ..., d_n)} \tag{1}$$

- Prior $p(a)$: The prior distribution of $p(a)$ is a discrete uniform distribution. Therefore: $p(a = i) = \frac{1}{16}, \forall i \in [0, 15]$
- Likelihood $p(d_1, d_2, ..., d_n|a)$: Since fans make independent decisions for every match, the joint probability of events $d_1, d_2, ..., d_n$ is equal to the product of their individual probabilities:

$$p(d_1, d_2, ..., d_n|a) = \prod_{i=1}^{n} p(d_i|a) \tag{2}$$

For the k-th match, we can marginalise over c_k, noting that the number of ticket holders attending a game is necessarily in the range from 0 to $a_{max} + b_{max}$:

$$\begin{aligned} p(d_k|a) &= \sum_{c_k=0}^{a_{max}+b_{max}} p(d_k, c_k|a) \\ &= \sum_{c_k=0}^{a_{max}+b_{max}} p(d_k|c_k)p(c_k|a) \qquad \text{(because } d_k \text{ is independent of } a \text{ given } c_k) \end{aligned} \tag{3}$$

Need to compute $p(d_k|c_k)$ and $p(c_k|a)$.

– $p(d_k|c_k)$
If $d_k < c_k$ or $d_k > 2c_k$, then $p(d_k|c_k) = 0$, since every ticket holder is allowed to bring up to one guest, and guests are not allowed in without a ticket holder. If $0 \leq d_k \leq 2c_k$, then the number of guests (i.e. $d_k - c_k$) follows a binomial distribution with parameters c_k and p_d. Therefore:

$$\begin{aligned} p(d_k|c_k) &= \binom{c_k}{d_k - c_k} p_d^{d_k - c_k} (1 - p_d)^{c_k - (d_k - c_k)} \\ &= \binom{c_k}{d_k - c_k} p_d^{d_k - c_k} (1 - p_d)^{2c_k - d_k} \\ &= \binom{c_k}{d_k - c_k} (0.5)^{d_k - c_k} (0.5)^{2c_k - d_k} \\ &= \binom{c_k}{d_k - c_k} (0.5)^{c_k} \end{aligned} \tag{4}$$

– $p(c_k|a)$
In order to compute $p(c_k|a)$, we marginalise over b, which takes values between b_{min} and b_{max}.

$$\begin{aligned} p(c_k|a) &= \sum_{i=b_{min}}^{b_{max}} p(c_k, b = i|a) \\ &= \sum_{i=b_{min}}^{b_{max}} p(c_k|a, b = i)p(b = i|a) \\ &= \sum_{i=b_{min}}^{b_{max}} p(c_k|a, b = i)p(b = i) \qquad \text{(because } b \text{ is independent of } a) \\ &= \frac{1}{201} \sum_{i=b_{min}}^{b_{max}} p(c_k|a, b = i) \qquad \text{(since } p(b = i) = \frac{1}{201} \forall i) \end{aligned} \tag{5}$$

For a given b:

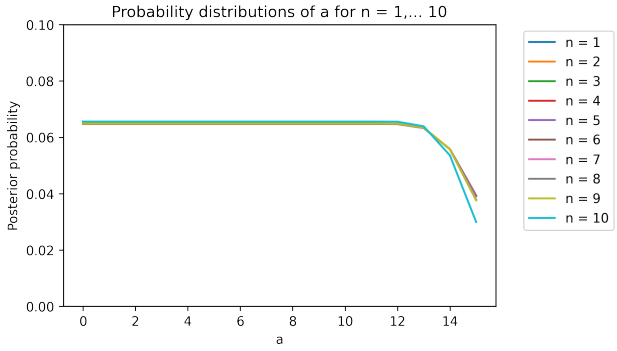
$$p(c_k|a, b = i) = \sum_{a_k=0}^{a} \binom{a}{a_k} p_a^{a_k} (1 - p_a)^{a - a_k} * \binom{b}{c_k - a_k} p_b^{c_k - a_k} (1 - p_b)^{b - c_k + a_k}$$

With the constraint that $0 \leq c_k - a_k \leq b$

- In order to compute the posterior probability after the k-th match, we use Bayesian updating based on the likelihood of d_k given a, with $p(d_k)$ as the normalising constant:

$$p(a|d_k) = \frac{p(a)p(d_k|a)}{p(d_k)} \tag{6}$$

The graph below plots all probability distributions of a after n matches with $n = 1,\ldots 10$.



The MAP/ML estimate for a after the 10-th match is 11. We can observe that a high value of a is consistent with the fact that the variance of the observed values of d is low, suggesting that season ticket holders account for a large ratio of fans in the stadium.

Comment: We can also verify the result for n = 10 in a single step by using the likelihood of the joint event consisting of all observed values of d given the prior distribution of a.

(d).2 Probability distribution of b

Repeating the same approach for b, and marginalising over a this time, we obtain:

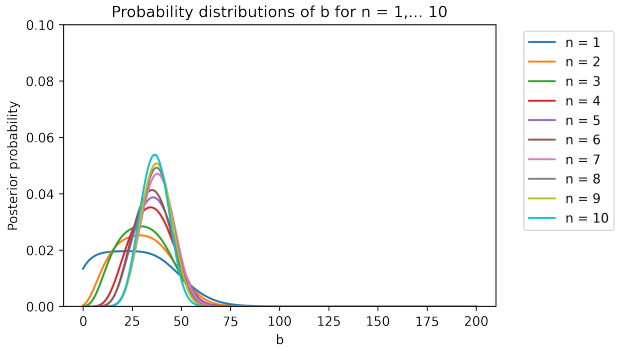
$$p(c_k|b) = \frac{1}{16}\sum_{i=a_{min}}^{a_{max}} p(c_k|b, a = i) \tag{7}$$

For a given a:

$$p(c_k|b, a = i) = \sum_{b_k=0}^{b} \binom{b}{b_k} p_b^{b_k}(1-p_b)^{b-b_k} * \binom{a}{c_k - b_k} p_a^{c_k-b_k}(1-p_a)^{a-c_k+b_k}$$

With the constraint that $0 \leq c_k - b_k \leq a$

We obtain the following plots for the posterior distribution of b after n matches with n = 1,... 10.



The MAP/ML estimate for b after the 10-th match is 36.

(e) Python Code

```
import scipy.special
import numpy as np
import pandas as pd
import math
from matplotlib import pyplot as plt

def prob_d_given_c(d):
    """Compute the probability of a given value of d_k given all potential values of
    ↪  c_k"""
    prob_d_given_c = np.zeros((215+1,1))
    for ck in range(215+1):
        # constraint on d_k is that it cannot be smaller than c_k, nor larger than
        ↪  two times c_k
        if d < ck:
            prob_d_given_c[ck,0] = 0
        elif d > 2*ck:
            prob_d_given_c[ck,0] = 0
        # apply formula for binomial distribution
        else:
            prob_d_given_c[ck,0] = scipy.special.comb(ck,d-ck) * ((0.5)**ck)
    return prob_d_given_c

p_a = 0.99
p_b = 0.3

# We create a matrix that captures the conditional likelihood of all values of c_k
↪  given all potential values of a.
# The element (i,j) of this matrix corresponds to the likelihood of c_k = i, given a
↪  = j.
prob_c_given_a = np.zeros((215+1,15+1))
for a in range(15+1):
    for c_k in range(215+1):
        prob_c_k_given_a = []
        # marginalising over b
        for b in range (200+1):
            prob_c_k_given_b = []
            # consider all potential values of a_k
            for a_k in range(a+1):
                # constraints for a_k
                if a_k > c_k:
                    continue
                elif c_k - a_k > b:
                    continue
                # formula for cases where the constraints are met
                else:
                    prob_c_k_a_k = scipy.special.comb(a,
                    ↪  a_k)*(p_a**a_k)*((1-p_a)**(a-a_k))*scipy.special.comb(b,
                    ↪  c_k-a_k)* \
                            p_b**(c_k-a_k)*(1-p_b)**(b-c_k+a_k)
                    prob_c_k_given_b.append(prob_c_k_a_k)
            prob_c_k_given_b_total = sum(prob_c_k_given_b)
            prob_c_k_given_a.append(prob_c_k_given_b_total)
```

```
        # update matrix for given pairing of c_k and a
        prob_c_given_a[c_k,a] = sum(prob_c_k_given_a)/201

# Use previous prob_d_given_c and prob_c_given_a to compute prob_d_given_a.
# We create a matrix that captures the conditional likelihood of all values of c_k
↪  given all potential values of a.
prob_d_given_a = np.zeros((430+1,15+1))
for d in range(430+1):
    prob_d_given_c_k = prob_d_given_c(d)
    for a in range(15+1):
        result = []
        for i in range(215+1):
            result.append(prob_d_given_c_k[i]*prob_c_given_a[i,a])
        prob_d_given_a[d,a] = sum(result)

d_values = [22, 27, 26, 32, 31, 25, 35, 26, 28, 23]

# Create an array to capture all probability distributions of a, such that the i-th
↪  row of the array corresponds
# to the probability distribution of a after the i-th match. The row 0 corresponds
↪  to the prior uniform distribution.
distribution_a = np.zeros((10+1, 15+1))

# Set row 0 to uniform distribution.
for i in range(15+1):
    distribution_a[0,i] = 1/16

# Populate rows 1-10 based using Bayes theorem.
for i in range(10):
    dk = d_values[i]
    likelihood = []
    # compute the denominator, multiplying the prior of a by the likelihood of d
    ↪  given a,
    # and computing the sum over all values of a
    for a in range(15+1):
        likelihood.append(distribution_a[i,a]*prob_d_given_a[dk,a])
    denominator = sum(likelihood)
    # compute posterior probability
    for a in range(15+1):
        distribution_a[i+1,a] =
        ↪  (distribution_a[i,a]*prob_d_given_a[dk,a])/denominator


# Identifying the MAP/ML estimate of a after 10 matches, using the last row of the
↪  matrix.
np.argmax(distribution_a[-1,:])

for i in range(1, distribution_a.shape[0]):
    plt.plot(distribution_a[i,:])
    plt.ylim(0,0.1)
    plt.xlabel('a')
    plt.ylabel('Posterior probability')
    plt.title(f'Probability distributions of a for n = 1,... 10')
    plt.savefig(f"q4a", dpi=300)
```

```
from matplotlib.backends.backend_pdf import PdfPages
# plotting the probability distributions.
plot_prob = PdfPages('q4a.pdf')
for i in range(distribution_a.shape[0]):
    plot = plt.figure()
    plt.plot(distribution_a[i,:])
    plt.ylim(0,0.1)
    plt.xlabel('a')
    plt.ylabel('Posterior probability')
    plt.title(f'Probability distribution of a after {i} matches')
    plt.show
    plot_prob.savefig(plot)
plot_prob.close()

# Alternative approach. We can verify our result for n=10 in a single step using the
↪  joint probability.

likelihood_joint_given_a = [1 for i in range(16)]

for a in range(15+1):
    for dk in d_values:
        likelihood_joint_given_a[a] *= prob_d_given_a[dk,a]
prior_multiplied_by_likelihood = [likelihood_joint_given_a[a]/16 for a in
↪  range(15+1)]
denominator = sum(prior_multiplied_by_likelihood)
posterior_a = []
for a in range(15+1):
    posterior_a.append(prior_multiplied_by_likelihood[a]/denominator)

np.argmax(posterior_a)

# Repeat the same logic to determine the likelihood of c_k given b, and the
↪  likelihood of d_k given b
# Probability of c_k given b:
prob_c_given_b = np.zeros((215+1,200+1))
for b in range(200+1):
    for c_k in range(215+1):
        prob_c_k_given_b = []
        for a in range (15+1):
            prob_c_k_given_a = []
            for b_k in range(b+1):
                if b_k > c_k:
                    continue
                elif c_k - b_k > a:
                    continue
                else:
                    prob_c_k_b_k = scipy.special.comb(b,
                    ↪  b_k)*(p_b**b_k)*((1-p_b)**(b-b_k))*scipy.special.comb(a,
                    ↪  c_k-b_k)* \
                            p_a**(c_k-b_k)*(1-p_a)**(a-c_k+b_k)
                    prob_c_k_given_a.append(prob_c_k_b_k)
            prob_c_k_given_a_total = sum(prob_c_k_given_a)
            prob_c_k_given_b.append(prob_c_k_given_a_total)
```

```
        prob_c_given_b[c_k,b] = sum(prob_c_k_given_b)/16

prob_d_given_b = np.zeros((430+1,200+1))
for d in range(430+1):
    prob_d_given_c_k = prob_d_given_c(d)
    for b in range(200+1):
        result = []
        for i in range(215+1):
            result.append(prob_d_given_c_k[i]*prob_c_given_b[i,b])
        prob_d_given_b[d,b] = sum(result)

# Create an array to capture all probability distributions of b, such that the i-th
↪  row of the array corresponds
# to the probability distribution of b after the i-th match. The row 0 corresponds
↪  to the prior uniform distribution.
distribution_b = np.zeros((10+1, 200+1))

# Set row 0 to uniform distribution.
for i in range(200+1):
    distribution_b[0,i] = 1/(200+1)

# Populate rows 1-10 based using Bayes theorem.
for i in range(10):
    dk = d_values[i]
    likelihood = []
    # compute the denominator, multiplying the prior of a by the likelihood of d
    ↪  given b,
    # and computing the sum over all values of b
    for b in range(200+1):
        likelihood.append(distribution_b[i,b]*prob_d_given_b[dk,b])
    denominator = sum(likelihood)
    # compute posterior probability
    for b in range(200+1):
        distribution_b[i+1,b] =
        ↪  (distribution_b[i,b]*prob_d_given_b[dk,b])/denominator

# Identifying the MAP/ML estimate of b after 10 matches, using the last row of the
↪  matrix.
np.argmax(distribution_b[-1,:])

for i in range(1, distribution_b.shape[0]):
    plt.plot(distribution_b[i,:])
    plt.ylim(0,0.1)
    plt.xlabel('b')
    plt.ylabel('Posterior probability')
    plt.title(f'Probability distributions of b for n = 1,... 10')
    plt.savefig(f"q4b", dpi=300)

from matplotlib.backends.backend_pdf import PdfPages
# plotting the probability distributions.
plot_prob = PdfPages('q4b.pdf')
for i in range(distribution_b.shape[0]):
    plot = plt.figure()
    plt.plot(distribution_b[i,:])
```

```
    plt.ylim(0,0.1)
    plt.xlabel('b')
    plt.ylabel('Posterior probability')
    plt.title(f'Probability distribution of b after {i} matches')
    plt.show
    plot_prob.savefig(plot)
plot_prob.close()
```