Statistical Machine Learning 2018
Assignment 2
Deadline: 28th of October 2018

Christoph Schmidl
s4226887
c.schmidl@student.ru.nl

Mark Beijer
s4354834
mbeijer@science.ru.nl

November 4, 2018

Exercise 1 - weight 3

The financial services department of an insurance company receives numerous phone calls each day from people who want to make a claim against their policy. Most claims are genuine, however about 1 out of every 6 are thought to be fraudulent. To tackle this problem the company has installed a trial version of a software voice-analysis system that monitors each conversation and gives a numerical score z between 0 and 1, depending on allegedly suspicious vocal intonations of the customer. Unfortunately, nobody seems to know anymore how to interpret the score in this particular version of the system...

Tests revelealed that the conditional probability density of z, given that a claim was valid ($c = 1$) or false $c = 0$ are

$$\begin{aligned} p(z|c=0) &= \alpha_o(1-z^2), \\ p(z|c=1) &= \alpha_1 z(z+1). \end{aligned}$$

1.1

Compute the normalization constants α_0 and α_1. How does the z score relate to the validity of the claim? What values for z would you expect when the claim is valid / false?

Answer:

We integrate over all possible values of z, which has to equal 1, for it is sure that give c some value of z will be chosen.

$$\begin{aligned} 1 &= \int_0^1 \alpha_0(1-z^2)dz && (1) \\ &= \alpha_0 z - \frac{\alpha_0}{3}z^3\Big|_0^1 && (2) \\ &= \left(\alpha_0 - \frac{\alpha_0}{3}\right) = \frac{2}{3}\alpha_0 && (3) \\ a_0 &= \frac{3}{2} && (4) \end{aligned}$$

And for the next constant:

$$
\begin{aligned}
1 &= \int_0^1 \alpha_1 z(z+1)dz && (5)\\
&= \frac{\alpha_1}{3}z^3 + \frac{\alpha_1}{2}z^2\Big|_0^1 && (6)\\
&= \frac{1}{3}\alpha_1 + \frac{1}{2}\alpha_1 = \frac{5}{6}\alpha_1 && (7)\\
\alpha_1 &= \frac{6}{5} && (8)
\end{aligned}
$$

From the plot at figure 1 we see that for higher values of z the probabilty p(z|c=1) is higher. Therefore we think that higher values of z correspond with a higher change that the claim is true/not fraudulent.

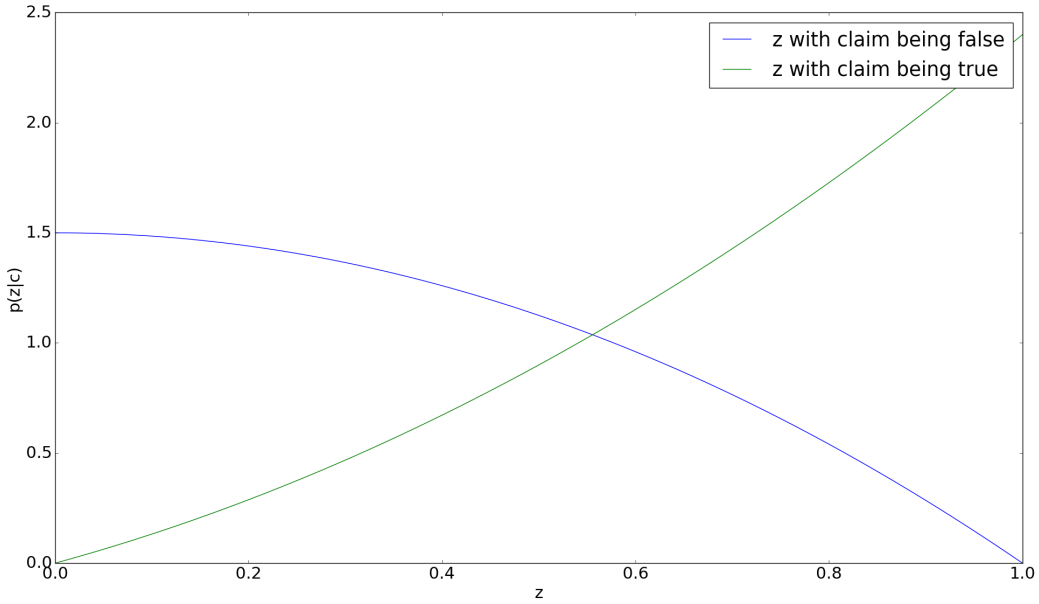


Figure 1: The probabilty p(z|c) plotted against z, with $0 < z < 1$.

1.2

Use the sum and product rule to show that the probability distribution function $p(z)$ can be written as

$$p(z) = \frac{(3z+1)(z+1)}{4} \tag{9}$$

Answer:

For this we need to sum over the possible values of c and add the probabilty of c times the conditional probability over c.

$$
\begin{aligned}
p(z) &= p(c=0)p(z|c=0) + p(c=1)p(z|c=1) && (10)\\
&= \frac{1}{6}\frac{3}{2}(1-z^2) + \frac{5}{6}\frac{6}{5}z(z+1) && (11)\\
&= \frac{3}{12}(1-z^2) + \frac{30}{30}z(z+1) && (12)\\
&= \frac{1}{4}(1-z^2) + z(z+1) && (13)\\
&= \frac{1}{4}(1-z^2) + \frac{4z(z+1)}{4} && (14)\\
&= \frac{4z^2+4z-z^2+1}{4} && (15)\\
&= \frac{3z^2+4z+1}{4} && (16)\\
&= \frac{(3z+1)(z+1)}{4} && (17)
\end{aligned}
$$

1.3

Use Bayes' rule to compute the posterior probability distribution function $p(c|z)$. Plot these distributions in MATLAB as a function of z. How can these posterior probabilities help in making a decision regarding the validity of the claim?

Answer:

Bayes rule states that:

$$p(c|z) = \frac{p(z|c)p(c)}{p(z)} \tag{18}$$

So for both values of c we get:

$$
\begin{aligned}
p(c=0|z) &= \frac{\frac{2}{3}(1-z^2)\frac{1}{6}\cdot 4}{(3z+1)(z+1)} && (19)\\
&= \frac{8}{18}\frac{1-z^2}{(3z+1)(z+1)} && (20)\\
&= \frac{4}{9}\frac{1-z^2}{(3z+1)(z+1)} && (21)\\
&= \frac{4}{9}\frac{(1+z)(1-z)}{(3z+1)(z+1)} && (22)\\
&= \frac{4}{9}\frac{1-z}{3z+1} && (23)\\
p(c=1|z) &= \frac{\frac{6}{5}z(z+1)\frac{5}{6}\cdot 4}{(3z+1)(z+1)} && (24)\\
&= \frac{4z(z+1)}{(3z+1)(z+1)} && (25)\\
&= \frac{4z}{3z+1} && (26)
\end{aligned}
$$

The plot of the distribution can be seen in figure 2.

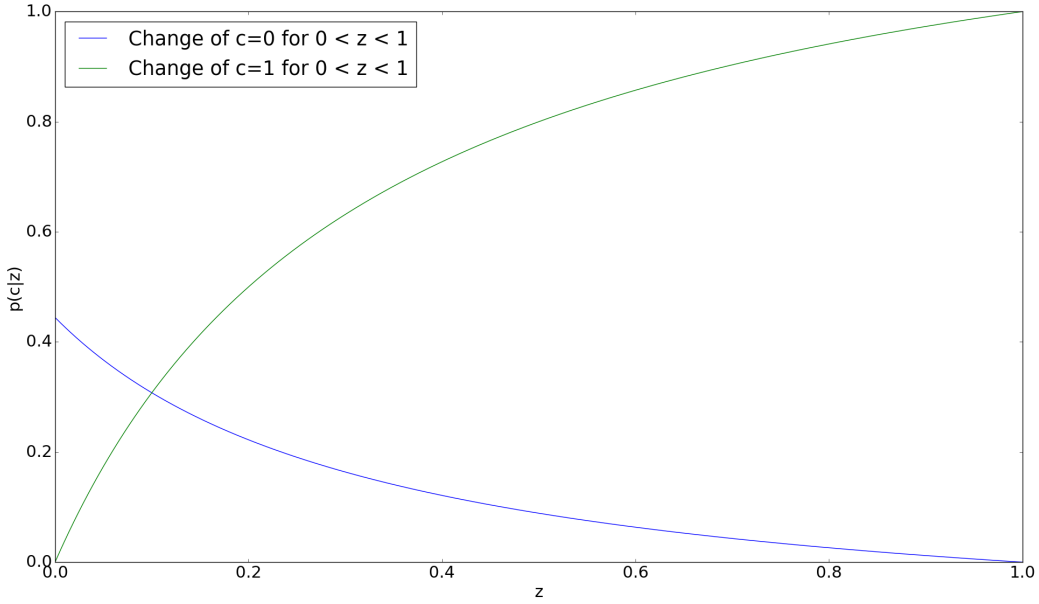


Figure 2: The probabilty p(c|z) plotted against z, with $0 < z < 1$.

1.4

Compute the optimal decision boundary (based on our numerical score z) that minimizes the misclassification rate. For which z should we classify $c = 0$ (false) and for which z should we classify $c = 1$ (valid)? Explain your decision.

Answer:

We take the decision boundary at the point where the two functions, as plotted at Figure 2, interesect. This happens at the point where they equal:

$$\frac{4}{9}\frac{1-z}{3z+1} = \frac{4z}{3z+1} \tag{27}$$

$$\frac{4}{9}(1-z) = 4z \tag{28}$$

$$1-z = 9z \tag{29}$$

$$1 = 10z \tag{30}$$

$$z = 0.1 \tag{31}$$

We have also found the point 0.1 using our program.

1.5

Compute the misclassification rate, given the optimal decision boundary determined previously. Interpret the result you have obtained. Is the z score useful in determining the validity of the claim? Compare this with your prior guess from 1.

Answer:

We integrate the probabilities up to the descision point and from the decision point to 1. Then we compare the change of actually getting the right label vs us putting the decision boundary at 0.1 .
If z is less then 0.1 we classify it as c=0. The actual change is:

$$
\begin{aligned}
p(z<0.1)p(c=0|z<0.1) &= \int_0^{0.1} \frac{4}{9}\frac{1-z}{3z+1}dz && (32)\\
\frac{1-z}{3z+1} &= \frac{4}{3(3z+1)} - \frac{1}{3} && (33)\\
p(z<0.1)p(c=0|z<0.1) &= \frac{4}{9}\int_0^{0.1}\left(\frac{4}{3}\frac{1}{(3z+1)} - \frac{1}{3}\right)dz && (34)\\
&= \frac{4}{9}\left(\frac{4}{9}\ln(3z+1) - \frac{z}{3}\right)\Bigg|_0^{0.1} && (35)\\
&\approx 0.0370 && (36)
\end{aligned}
$$

Now for the other possibility:

$$
\begin{aligned}
p(z<0.1)p(c=1|z<0.1) &= \int_0^{0.1} \frac{4z}{3z+1}dz && (37)\\
\frac{4z}{3z+1} &= \frac{4}{3} - \frac{4}{3}\frac{1}{3z+1} && (38)\\
&= \frac{4}{3}\left(1 - \frac{1}{3z+1}\right) && (39)\\
p(z<0.1)p(c=1|z<0.1) &= \frac{4}{3}\int_0^{0.1} 1 - \frac{1}{3z+1}dz && (40)\\
&= \frac{4}{3}\left(z - \frac{1}{3}\ln(3z+1)\right)\Bigg|_0^{0.1} && (41)\\
&\approx 0.0167 && (42)
\end{aligned}
$$

For both $\ln(1) = 0$, so filling in z = 0 gives us 0.
Now the fraction of the two:

$$
\text{Classification error} = \frac{p(c=1|z<0.1)}{p(c=0|z<0.1)} \approx 45.20\% \quad (43)
$$

Now we do the same for z>0.1. Here we already calculated the indefinite integrals, so we only have to fill in the boundaries:

$$
\begin{aligned}
p(z>0.1)p(c=0|z>0.1) &= \int_{0.1}^{1} \frac{4}{9}\frac{1-z}{3z+1}dz && (44)\\
&= \frac{4}{9}\left(\frac{4}{9}\ln(3z+1) - \frac{z}{3}\right)\Bigg|_{0.1}^{1} && (45)\\
&\approx 0.0887 && (46)
\end{aligned}
$$

And for c=1:

$$
\begin{aligned}
p(z>0.1)p(c=1|z>0.1) &= \int_{0.1}^{1} \frac{4z}{3z+1}dz && (47)\\
&= \frac{4}{3}\left(\frac{1}{3}\ln(3z+1) - z\right)\Bigg|_{0.1}^{1} && (48)\\
&\approx 0467 && (49)
\end{aligned}
$$

$$
\text{Classification error} = \frac{p(c=0|z>0.1)}{p(c=1|z>0.1)} \approx 18.98\% \quad (50)
$$

Now the total classification error is the sum of the two errors, weighted by the probability of z lying in these regions. So let's integrate z:

$$\begin{aligned} p(z) &= \frac{(3z+1)(z+1)}{4} && (51)\\ &= \frac{1}{4}(3z^2+4z+1) && (52)\\ \int p(z)dz &= \frac{1}{4}\left(z^3+2z^2+z+C\right) && (53) \end{aligned}$$

C is the integration constant independent of z.
Now we calculate the integrals:

$$\begin{aligned} p(z<0.1) &= \int_0^{0.1} \frac{(3z+1)(z+1)}{4} \approx 0.03025 && (54)\\ p(z>0.1) &= \int_{0.1}^{1} \frac{(3z+1)(z+1)}{4} \approx 0.96975 && (55) \end{aligned}$$

So the total error is:

$$\text{Misclassification error} = 0.03025 \cdot 45.20\% + 0.96975 \cdot 18.98\% \approx 19.77\% \quad (56)$$

Exercise 2 - weight 2

The government of the United Kingdom has decided to call a referendum regarding the country's European Union membership. The citizens of the UK will be asked the following question at the referendum: "Whould the United Kingdom remain a member of the European Union or leave the European Union?". The European Commission (EC) is interested in the potential outcome of this referendum and has contracted a polling agency to study this issue.
Suppose that a person's vote follows a Bernoulli distribution with paramter θ and suppose the the EC's prior distribution for θ, the proportion of British citizens that would be in favor of leaving the EU, is beta distribution with $\alpha = 90$ and $\beta = 110$.

2.1

Determine the mean and variance of the prior distribution. Plot the prior density function.

Answer:

The beta distribution is defined as:

$$Beta(\mu|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha-1}(1-\mu)^{\beta-1} \quad (57)$$

The mean is calculated as following:

$$\begin{aligned} \mathbb{E}[\mu] &= \int_0^1 \text{Beta}(\mu|\alpha,\beta)\mu d\mu && (58)\\ &= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha}(1-\mu)^{\beta-1}d\mu && (59)\\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int_0^1 \mu^{\alpha}(1-\mu)^{\beta-1}d\mu && (60) \end{aligned}$$

The mean and variance are therefore:

$$\begin{aligned} \mathbb{E}[\mu] &= \frac{\alpha}{\alpha+\beta} = 4.5 \cdot 10^{-1} && (61)\\ var[\mu] &= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \approx 1.23 \cdot 10^{-3} && (62) \end{aligned}$$

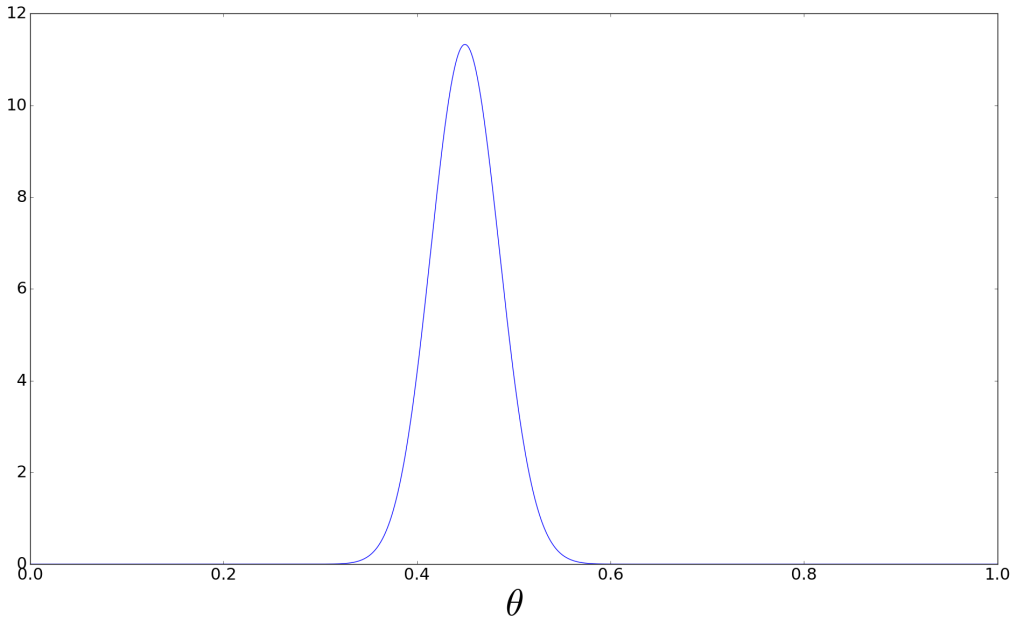


Figure 3: Plot of the beta distrubition for $\alpha = 90$ and $\beta = 110$.

2.2

A random sample of 1000 British citizens is taken, and 60% of the people polled support leaving the European Union. What are the posterior mean and variance for θ? Plot the posterior density function together with the prior density. Explain how the data from the sample changed the prior belief.

Answer:

The posterior probabilty is described as:

$$p(\theta|m,l,\alpha,\beta) = \frac{\Gamma(m+\alpha+l+\beta)}{\Gamma(m+\alpha)\Gamma(l+\beta)}\mu^{m+\alpha-1}(1-\mu)^{l+\beta-1} \tag{63}$$

So the value of α is increased by m and the value of β is increased by l.The value of m is the amount of people in support, the value of m is how many people will vote in favour, while l is how many people will vote against. So the new mean is:

$$\mathbb{E}[\theta] = \frac{\alpha+m}{\alpha+m+\beta+l} = 5.75 \cdot 10^{-1} \tag{64}$$

$$var[\theta] = \frac{(\alpha+m)(\beta+l)}{(\alpha+\beta+l+m)^2(\alpha+\beta+m+l+1)} = 2.03 \cdot 10^{-4} \tag{65}$$

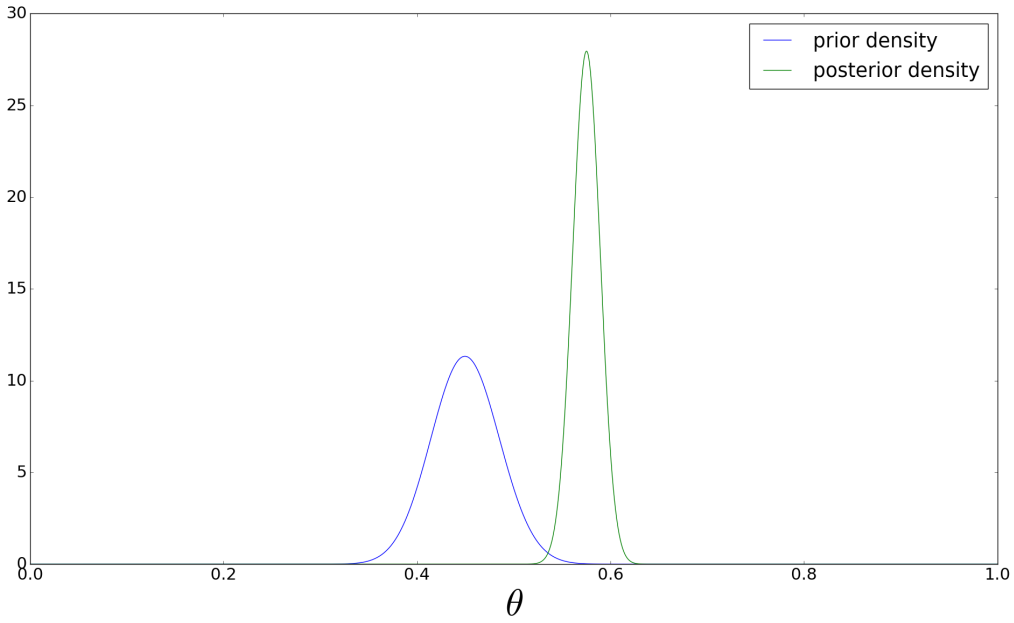


Figure 4: Plot of the prior and posterior probabilty distribution.

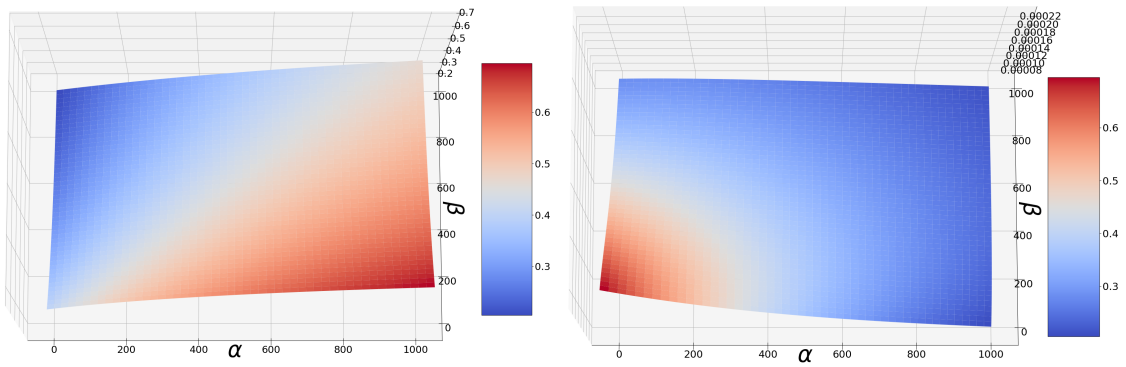
As you can see in the plot, the expected value (mean) of the plot is further to the right. For the new information showed more people supporting the brexit. You can also see that the variance has decreased, since we're more sure about our guess because we have more data.

2.3

Examine the effect of changing the prior hyperparameters (α, β) on the posterior by looking at several other hyperparamter configurations. Which values for α and β correspond to a non-informative prior? What is the interpretation of α and β for the beta prior? What does the choice of α and β in Question 1 tell you about the strength of the prior belief?

Answer:

I took different values of α and β and plotted them against each other:



(a) The mean plotted against α and β

(b) The variance plotted against α and β

Figure 5: The mean and variance given different hyperparamters α and β.

At figure 5 you can see the mean is higher for higher *alpha* and lower β. The variance increases when α or β is lowered. If you have a non-informed prior your guess is somewhere in the middle with a high variance. This corresponds to a low α and β, while $\alpha = \beta$ so the mean will be $\frac{\alpha}{2\alpha} = 0.5$. This makes sense, for if you gather more information you can update your hyperparameters, thus making them larger. So the hyperparameters in the prior tells us what we believe the distribution is likely to be.

The size of α and β tells us about the strenghth of our believe. If they are higher it is as if we had low hyperparameters to start with, but saw lots of evidence suggesting our current state of hyperparamenters. They also give rise to a low variance as seen in figure 5.

2.4

Imagine you are now a reporter for the polling agency and you have been sent on field duty to gather more data. Your mission is to go out on the streets and randomly survey people on their thoughts regarding the upcoming referendum. Given all the available information you have acquired, what is the probability that the first person you talk to will vote 'Leave'?

Hint: Derive the predictive distribution for the next vote using the posterior distribution for θ computed in Question 2. For a reminder on predictive distribution, see subsection 1.2.6 in Bishop, in particular Equation (1.68).

Answer:

Equation (1.68) says:

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int p(t|x, w)p(w|\mathbf{x}, \mathbf{t})dw \tag{66}$$

Exercise 3 - Sequential learning (weight 5)

Part 1 - Obtaining the prior
Consider a four dimenstional variable $[x_1, x_2, x_3, x_4]^T$, distributed according to a multivariate Gaussian with mean $\tilde{\mu} = [1, 0, 1, 2]^T$ and covariance matrix $\tilde{\Sigma}$ given as

$$\tilde{\Sigma} = \left(\begin{array}{cc|cc} 0.14 & -0.3 & 0.0 & 0.2 \\ -0.3 & 1.16 & 0.2 & -0.8 \\ \hline 0.0 & 0.2 & 1.0 & 1.0 \\ 0.2 & -0.8 & 1.0 & 2.0 \end{array}\right)$$

We are interested in the conditional distribution over $[x_1, x_2]^T$, given that $x_3 = x_4 = 0$. We know this conditional distribution will also take the form of a Gaussian:

$$p([x_1, x_2]^T \mid x_3 = x_4 = 0) = \mathcal{N}([x_1, x_2]^T \mid \mu_p, \Sigma_p) \tag{67}$$

for which the mean and covariance matrix are most easily expressed in terms of the (partitioned) precision matrix (see Bishop, §2.3.1).

3.1.1

Use the partitioned precision matrix $\tilde{\Lambda} = \tilde{\Sigma}^{-1}$ to give an explicit expression for the mean μ_p and covariance matrix Σ_p of this distribution and calculate their values. (This distribution will be taken as the prior information for the rest of this exercise, hence the subscript p). You may use the MATLAB command `inv` to calculate the matrix inverses.

Answer:

We can compute the precision matrix by taking the inverse of the given covariance matrix using the Python numpy function `numpy.linalg.inv`. The following python code just gives us this precision matrix.

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


# Exercise 3.1.1

# numpy.linalg.inv(a)
# Compute the (multiplicative) inverse of a matrix.
# Given a square matrix a, return the matrix ainv
# satisfying dot(a, ainv) = dot(ainv, a) = eye(a.shape[0]).
```

```
12
13 # See: https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.linalg.inv.html
14
15 covariance_matrix = np.array([
16         [0.14, -0.3, 0.0, 0.2],
17         [-0.3, 1.16, 0.2, -0.8],
18         [0.0, 0.2, 1.0, 1.0],
19         [0.2, -0.8, 1.0, 2.0]])
20
21 precision_matrix = np.linalg.inv(covariance_matrix)
22
23 print(precision_matrix)
24
25 #[[ 60.   50.  -48.   38. ]
26 # [ 50.   50.  -50.   40. ]
27 # [-48.  -50.   52.4 -41.4]
28 # [ 38.   40.  -41.4  33.4]]
29
30 precision_matrix_aa = np.array([
31         [60, 50],
32         [50, 50]])
33
34 inv_presision_matrix_aa = np.linalg.inv(precision_matrix_aa)
35
36 print(inv_presision_matrix_aa)
37
38 #[[ 0.1  -0.1 ]
39 # [-0.1   0.12]]
```

The precision matrix is therefore:

$$\tilde{\Lambda} = \tilde{\Sigma}^{-1} = \left(\begin{array}{cc|cc} 60 & 50 & -48 & 38 \\ 50 & 50 & -50 & 40 \\ \hline -48 & -50 & 52.4 & -41.4 \\ 38 & 40 & -41.4 & 33.4 \end{array}\right) = \left(\begin{array}{c|c} \tilde{\Lambda}_{aa} & \tilde{\Lambda}_{ab} \\ \hline \tilde{\Lambda}_{ba} & \tilde{\Lambda}_{bb} \end{array}\right)$$

The last part of this equation is the partitioned form of the precision matrix as described in section 2.3.1, equation 2.69 of Bishop:

$$\tilde{\Lambda} = \left(\begin{array}{c|c} \tilde{\Lambda}_{aa} & \tilde{\Lambda}_{ab} \\ \hline \tilde{\Lambda}_{ba} & \tilde{\Lambda}_{bb} \end{array}\right)$$

In section 2.3.1, equation 2.73 of Bishop, it is described how one can conclude the covariance of $p(x_a|x_b)$ by using

$$\Sigma_{a|b} = \Lambda_{aa}^{-1}$$

and in section 2.3.1, equation 2.75, how to conclude the mean by using

$$\begin{aligned} \mu_{a|b} &= \Sigma_{a|b}\{\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\} \\ &= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \end{aligned}$$

Therefore, the conditional covariance is:

$$\Sigma_p = \tilde{\Lambda}_{aa}^{-1} = \begin{pmatrix} 0.1 & -0.1 \\ -0.1 & 0.12 \end{pmatrix}$$

And the conditional mean is:

$$
\begin{aligned}
\mu_p &= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \\
&= \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0.1 & -0.1\\-0.1 & 0.12\end{pmatrix}\begin{pmatrix}-48 & 38\\-50 & 40\end{pmatrix}\left[\begin{pmatrix}0\\0\end{pmatrix} - \begin{pmatrix}1\\2\end{pmatrix}\right] \\
&= \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0.1 & -0.1\\-0.1 & 0.12\end{pmatrix}\begin{pmatrix}-48 & 38\\-50 & 40\end{pmatrix}\begin{pmatrix}-1\\-2\end{pmatrix} \\
&= \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0.2 & -0.2\\-1.2 & 1\end{pmatrix}\begin{pmatrix}-1\\-2\end{pmatrix} \\
&= \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0.2\\-0.8\end{pmatrix} \\
&= \begin{pmatrix}0.8\\0.8\end{pmatrix}
\end{aligned}
$$

3.1.2

[MATLAB] - Create a function that can generate random number pairs, distributed according to the distribution in 67. Initialize your random generator and the draw a single pair

$$\mu_t = [\mu_{t1}, \mu_{t2}]^T \tag{68}$$

from this distribution. (These will be the 'true' means, hence the subscript t).

Hint: you can use the MATLAB function `mvnrnd` (which resides the Statistics toolbox).

Answer:

We used the `numpy.random.multivariate_normal` function to generate the random number pairs.

```python
# Exercise 3.1.2

# numpy.random.multivariate_normal(mean, cov[, size, check_valid, tol])

# Draw random samples from a multivariate normal distribution.

def generate_random_number_pair():
    return np.random.multivariate_normal(
    [0.8, 0.8],
    [[0.1, -0.1], [-0.1, 0.12]])

random_pair = generate_random_number_pair()
print(random_pair)

# [ 0.25822872  1.32385609]
```

By executing the above code we get the following:

$$\mu_t = \begin{pmatrix}0.26\\1.32\end{pmatrix}$$

3.1.3

[MATLAB] - Make a plot of the probability density of the distribution 67.

Hint: Use the MATLAB function `mvnpdf` (which resides in the Statistics toolbox) to calculate the probability density of a multivariate Gaussian random variable. The MATLAB functions `meshgrid` and `surf` may also prove useful.

Answer:

Instead of using the MATLAB `mvnpdf` function, we used the Python library and function `scipy.stats.multivariate_normal`.

```python
# Exercise 3.1.3

x, y = np.mgrid[-0.25:2.25:.01, -1:2:.01]
pos = np.empty(x.shape + (2,))
pos[:, :, 0] = x
pos[:, :, 1] = y
mu_p = [0.8, 0.8]
cov_p = [[0.1, -0.1], [-0.1, 0.12]]
z = multivariate_normal(mu_p, cov_p).pdf(pos)

fig = plt.figure(figsize=(10, 10), dpi=300)
ax = fig.gca(projection='3d')
ax.plot_surface(x, y, z)
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
ax.set_zlabel('$p(x_1, x_2 | x_3=0, x_4=0)$')
plt.savefig('3_1_3.png', bbox_inches='tight', dpi=300)
plt.show()
```

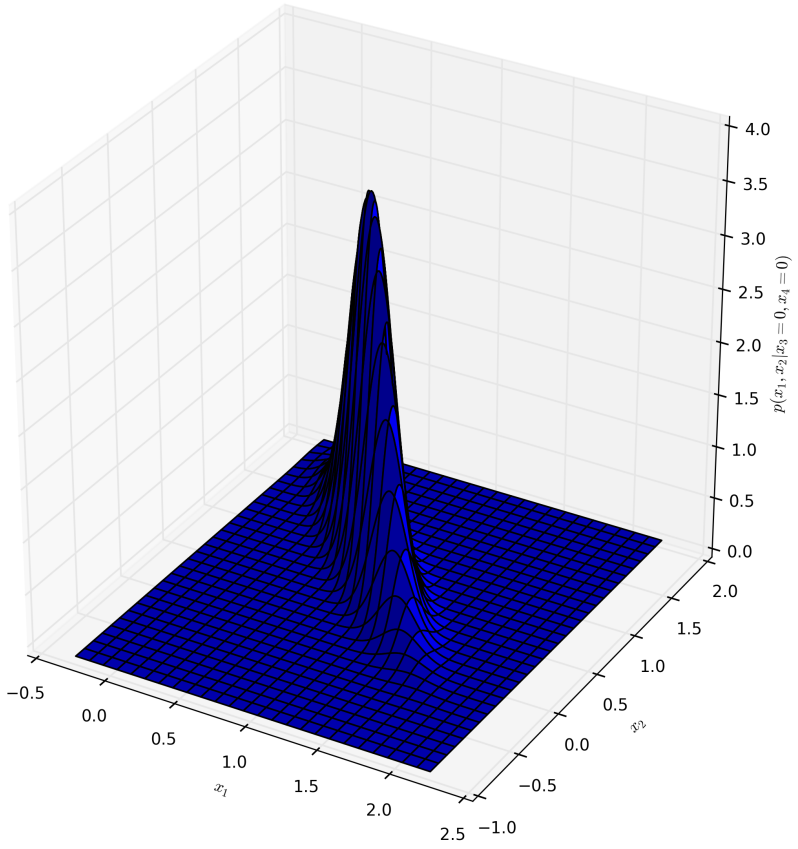


Figure 6: Plot of the probability density

Part 2 - Generating the data
Here we assume we are dealing with a 2d-Gaussian data generating process

$$p(x) = \mathcal{N}(x|\mu, \Sigma) \tag{69}$$

For the mean μ, we will use the value μ_t drawn in **??** in order to generate the data. Subsequently, we will pretend that we do not know this "true" value μ_t of μ, and estimate μ from the data. For the covariance matrix Σ we will use the "true" value

$$\Sigma_t = \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 4.0 \end{pmatrix}$$

to generate the data.

3.2.1

[MATLAB] - Generate at least 1000 data pairs $\{x_i, y_i\}$, distributed according to **??** with $\mu = \mu_t$ and $\Sigma = \Sigma_t$ and save them to a file in plain-text format.

Answer:

```
# Exercise 3.2.1

number_of_datapoints = 1000

covariance_3_2 = np.array([[2.0, 0.8],[0.8, 4.0]])
print(covariance_3_2)

data = np.random.multivariate_normal(
    random_pair,
    covariance_3_2,
    number_of_datapoints)

np.savetxt('ex3_data.txt', data)
```

3.2.2

From now on, we will assume (pretend) the 'true' mean μ_t is unknown and estimate μ from the data. Calculate the maximum likelihood estimate of μ_{ML} and Σ_{ML} for the data, and also an unbiased estimate of Σ (see Bishop, §2.3.4). Compare with the true values μ_t and Σ_t.

Answer:

```
# Exercise 3.2.2

mle_mean = np.mean(data, axis=0) # to take the mean of each col
print(mean)
normalized_data = data - mle_mean
mle_covariance = np.dot(normalized_data.T, normalized_data) / number_of_datapoints
mle_covariance_unbiased =  np.dot(normalized_data.T, normalized_data) / (
    number_of_datapoints - 1)
print(mle_covariance)
print(mle_covariance_unbiased)

# [ 0.24887413  1.39664922]
# [[ 1.91424907  0.90025163]
#  [ 0.90025163  3.91668677]]
# [[ 1.91616523  0.90115278]
#  [ 0.90115278  3.92060738]]
```

The maximum likelihood estimate of μ_{ML} is calculated directly on our generated dataset given the above code and approximates to:

$$\mu_{ML} = \frac{1}{N}\Sigma_{n=1}^{N} x_n = \begin{pmatrix} 0.25 \\ 1.4 \end{pmatrix}$$

The maximum likelihood estimate of the covariance is calculated directly on the data given the above code and approximates to:

$$\Sigma_{ML} = \frac{1}{N}\Sigma_{n=1}^{N}(x_n - \mu_{ML})(x_n - \mu_{ML})^T = \begin{pmatrix} 1.914 & 0.9 \\ 0.9 & 3.917 \end{pmatrix}$$

The unbiased maximum likelihood estimate of the covariance is calculated directly on the data given the above code, was normalized beforehand and approximates to:

$$\Sigma_{ML} = \frac{1}{N-1}\Sigma_{n=1}^{N}(x_n - \mu_{ML})(x_n - \mu_{ML})^T = \begin{pmatrix} 1.916 & 0.9 \\ 0.9 & 3.92 \end{pmatrix}$$

The "true" values were:

$$\Sigma_t = \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 4.0 \end{pmatrix}$$

and

$$\mu_t = \begin{pmatrix} 0.26 \\ 1.32 \end{pmatrix}$$

Therefore we can see that our estimates are close to the true values and that the values of the unbiased maximum likelihood estimate of the covariance do not change much.

Part 3 - Sequential learning algorithms
We will now estimate the mean μ from the generated data and the known variance Σ_t sequentially, i.e., by considering the data points one-by-one.

3.3.1

[MATLAB] - Write a procedure that processes the data points $\{x_n\}$ in the generated file one-by-one, and after each step comptues an updated estimate of μ_{ML}, the maximum likelihood of the mean (using Bishop, eq. 2.126).

Answer:

In section 2.3.5 "Sequential estimation" of Bishop, equation 2.126, a one-by-one estimate of μ_{ML} is described as follows:

$$\mu_{ML}^{(N)} = \mu_{ML}^{(N-1)} + \frac{1}{N}(x_N - \mu_{ML}^{(N-1)})$$

The corresponding python code:

```
# Exercise 3.3.1

mu = 0
for i in range(1, np.size(data, 0)+1):
    mu = mu + 1.0 / i * (data[i-1] - mu)
    print(mu)

# .
# .
# .
# [ 0.2476993   1.39181433]
# [ 0.24847877  1.39458494]
# [ 0.24593138  1.39564682]
# [ 0.24639905  1.39454373]
# [ 0.24649338  1.39428266]
# [ 0.24598495  1.39340938]
# [ 0.2468215   1.39439567]
# [ 0.2458356   1.39225878]
# [ 0.24887413  1.39664922]
```

We can see that at the end of the process the mean converges towards the previousy calculated mean and therefore works correctly.

3.3.2

Now we also use the prior information $p(\mu) = \mathcal{N}(\mu|\mu_p, \Sigma_p)$. From the prior, the generated data and the known variance Σ_t, we will estimate the mean μ.

Work out the details of sequential Bayesian inference (see eq. 2.144) for the mean μ. Apply Bayes' theorem in eq. 2.113 - 2.117 at each step $n = 1, ..., N$ to compute the new posterior mean $\mu^{(n)}$ and covariance $\Sigma^{(n)}$ after a new point (x_n) has arrived from the old posterior step. The first step starts from the original prior **??**.
Note: Do not confuse the posterior $\Sigma^{(n)}$ with the known Σ_t of the data generating process. For some more

hints, see appendix.

Answer:

3.3.3

[MATLAB] - Write a procedure that processes the data points $\{x_n\}$ in the generated file one-by-one, and after each step computes an updated estimate of μ_{MAP} - the maximum of the posterior distribution, using the results of the previous exercise.

Answer:

3.3.4

[MATLAB] - Plot both estimates (ML and MAP) in a single graph (1d or 2d) as a function of the number of data points observed. Indicate the true values $\{\mu_{t1}, \mu_{t2}\}$ as well. Evaluate your result.

Answer:

Hints

Below are some hints for **Exercise 3 - Part 3 - Question 2**.

Bayes'r rule is also valid if earlier acquired information is taken into account. For example, if this is earlier seen data $D_{n-1} = \{x_1, ..., x_{n-1}\}$. Bayes' rule conditioned on this earlier data is

$$P(\mu|x_n, D_{n-1}) \propto P(\mu|D_{n-1})P(x_n|\mu, D_{n-1})$$

Since $D_n = \{x_1, ..., x_n\}$ this is written more conveniently as

$$P(\mu|D_n) \propto P(\mu|D_{n-1})P(x_n|\mu, D_{n-1})$$

If, given the model paramters μ, the probability distribution of x_n is independent of earlier data D_{n-1}, we can further reduce this to

$$P(\mu|D_n) \propto P(\mu|D_{n-1})P(x_n|\mu)$$

You should be able to see the realtion with (2.144) and see in particular that the factor between brackets in (2.144) is to be identified with $P(\mu|D_{n-1})$.
Another important insight is that if $P(\mu|D_{n-1})$ and $P(x_n|\mu)$ are of the form (2.113) and (2.114), i.e., if $P(\mu|D_{n-1})$ is a Gaussian distribution over μ with a certain mean and covariance (you are free to give these any name, e.g. $\mu^{(n-1)}, \Sigma^{(n-1)}$) and if $P(x_n|\mu)$ is also Gaussian with a mean that is linear μ, then you can use (2.116) and (2.117) to compute the posterior $P(\mu|D_n)$, which therefore is also Gaussian.
So it is your task to show this. To do this you have to figure out the mapping of the variables and parameters in the current exercise, i.e., what is the correspondence between $\mu, x_n, \Sigma_t, \mu^{(n-1)}, \Sigma^{(n-1)}$ etc. with x, μ, Λ, A, b, L. Don't forget that some quantities can also be zero or and other may be identity matrices.