Assignment 3, Vingron Part

by Paul Vogler (4979420), Yiftach Kolb (5195763)

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

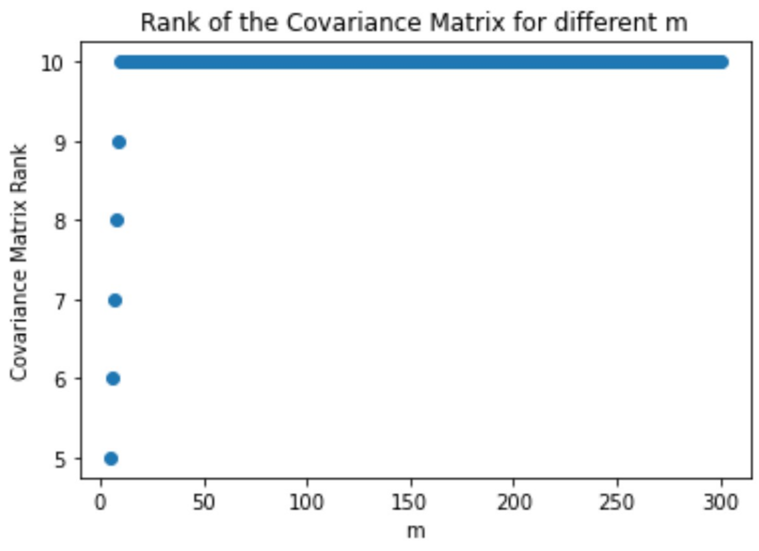
Problem 1 (20 Points; SVD).

Use Python to generate a matrix with m columns and 10 rows, entries should be sampled from Gaussian distribution and made non-negative (by taking absolute value). Let m vary as follows: m ∈ {5, 6, 7, 8, 9, 10, 15, 20, 25, . . . , 300}. Calculate the (estimated) covariance matrix (remember to center your data before using this formula).

```
In [3]: np.random.seed(42)
        mu, sigma = 0, 1
        m_list = list(range(5,301))
        m_ranks = []
        m_cov_matrices = []
        for m in m_list:
            matrix = np.random.normal(mu, sigma, size=(10, m))
            matrix = abs(matrix)
            matrix_centered = matrix - matrix.mean()
            cov_matix = (matrix_centered.T @ matrix_centered) / 9
            m_ranks.append(np.linalg.matrix_rank(cov_matix))
            m_cov_matrices.append(cov_matix)
```

(A) What is the rank of matrix S for different m? Create a scatterplot of the ranks in dependency on m and justify your result.

In [3]:
```
plt.scatter(m_list, m_ranks)
plt.xlabel("m")
plt.ylabel("Covariance Matrix Rank")
plt.title("Rank of the Covariance Matrix for different m")
plt.show()
```



Result:

The Rank of the Covariance Matrix is dependent on the m vs rowsize 10. For m <= 10 the rank is equal to m and equal to the matrix size. For m > 10 the rank can only be 10 at maximum, because of the 10 rows.

(B) Calculate the Singular Value Decomposition (SVD) of the matrix S.

In [12]:
```
m_u, m_w, m_vt = [],[],[]
for cov_matix in m_cov_matrices:
    u, w, vt = np.linalg.svd(cov_matix)
    m_u.append(u)
    m_w.append(np.diag(w))
    m_vt.append(vt)
```

(C) Compute the pseudoinverse S+ based on the SVD and the procedure explained in the slides. Do not use np.linalg.pinv or similar for this task.

In [13]:
```
# A+ = V*W^(-1)*U.T
m_S_plus = []
for u, w, vt in zip(m_u, m_w, m_vt):
    S_plus = vt.T @ np.linalg.inv(w) @ u.T
    m_S_plus.append(S_plus)
```

(D) Calculate the product S · S+. By definition, this should be an identity matrix for nonsingular matrices.

In [14]:
```
m_identities = []
m_is_identity = []
for m, S_mat, S_plus in zip(m_list, m_cov_matrices, m_S_plus):
    identity = np.identity(m)
    is_indent = S_mat @ S_plus
    m_identities.append(is_indent)
    is_indent_rounded = np.around(is_indent, decimals=10)
    m_is_identity.append(np.all(np.equal(is_indent_rounded, identit
y)))
print("For the following m, The S*S+ product is the identity matrix
 (when rounding to 10 decimals):")
print(np.array(m_list)[np.where(m_is_identity)[0]])
```

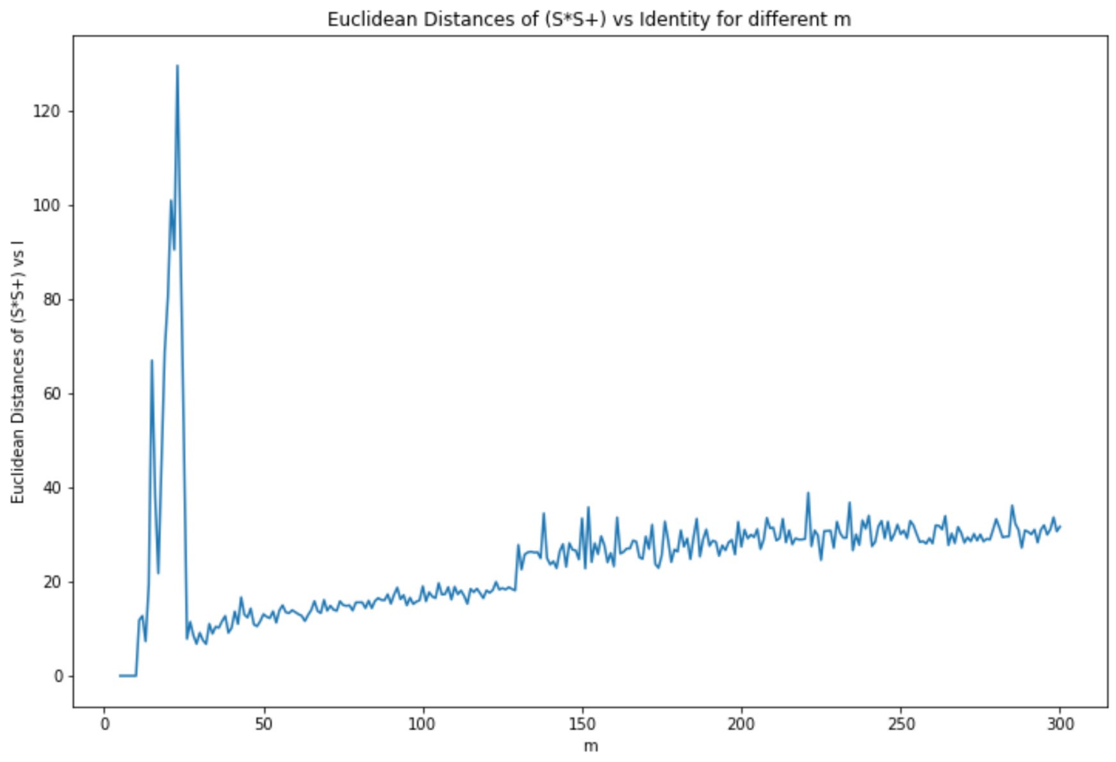
```
For the following m, The S*S+ product is the identity matrix (when
rounding to 10 decimals):
[ 5  6  7  8  9 10]
```

(E) Calculate the Euclidean distance between S · S+ and the identity matrix of rank m: Im.

In [15]:
```
m_dists = []
for m, ident in zip(m_list , m_identities):
    identity = np.identity(m)
    m_dists.append(np.linalg.norm(ident-identity))
```

(F) Plot this Euclidean distance versus m. Interpret your results.

In [16]:
```
plt.figure(figsize=(12, 8))
plt.plot(m_list, m_dists)
plt.xlabel("m")
plt.ylabel("Euclidean Distances of (S*S+) vs I")
plt.title("Euclidean Distances of (S*S+) vs Identity for different m")
plt.show()
```

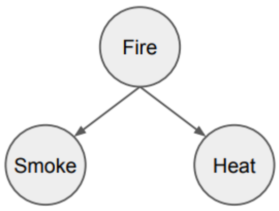


Result:

The euclidean distances are 0 for small m (probably the non-singular ones up to 10. For the slightly singular matrices for m > 10, the distance seems to be the most unstable. Then for m > 30 the distance does stabilize but stays significantly unequal to 0 and is slightly increasing.

Problem 2

Consider the following Baysian network with 3 variables: {F=Fire,S=Smoke,H=Heat}.



Variable F represents the existence of a fire (F = 1), S = 1 if we see smoke; H = 1 if we observe heat and vice versa:

- F = Fire (1 = Fire, 0 = No fire)
- S = Smoke (1 = Observed smoke, 0 = No observed smoke)
- H = Heat (1 = Observed heat, 0 = No observed heat)

We know the following probabilities:

- P(F = 1) = 0.1 => P(F = 0) = 0.9
- P(S = 1|F = 1) = 0.9 => P(S = 0|F = 1) = 0.1
- P(S = 1|F = 0) = 0.001 => P(S = 0|F = 0) = 0.999
- P(H = 1|F = 1) = 0.99 => P(H = 0|F = 1) = 0.01
- P(H = 1|F = 0) = 0.0001 = P(H = 0|F = 0) = 0.9999

(A) Write down the joint distribution of the Bayesian network P(F, S, H) using the conditionally independent effects. Are Smoke and Heat somehow independent?

a. $P(F,S,H) = \mathrm{p(F)} * \mathrm{p(S|F)} * \mathrm{p(H|F)}$

b. Smoke and Heat should not be independent, because they share the same parent node Fire. They are only conditionally independent, conditioned on Fire.

(B) Before we observe any data, what is the prior probability that there is no fire?

a. P(F = 0) = 1 - P(F = 1) = 0.9

(C) Now suppose that we observe smoke. Use Bayes' theorem to evaluate the posterior probability that there is no fire given the observation of smoke and compare it to prior probability that there is no fire.

a. $P(F=0|S=1) = \frac{P(F=0,S=1)}{P(S=1)} = \frac{P(S=1|F=0)*P(F=0)}{P(S=1)} = \frac{P(S=1|F=0)*P(F=0)}{P(S=1|F=1)*P(F=1)+P(S=1|F=0)*P(F=0)} = \frac{0.001*0.9}{0.9*0.1+0.001*0.9} \approx 0.00990$

b. The Probability of no fire with 0.9 is magnitudes higher than the probability of no fire given that we observed smoke.

(D) Next suppose that we also check the temperature and find that it is very hot. We have now observed the states of both smoke and heat. Compute the posterior probability that there is fire given the observations of both heat and smoke and compare it to the posterior probability of P(F = 0|S = 1).

a. $P(F=1|S=1,H=1) = \frac{P(S=1,H=1|F=1)*P(F=1)}{P(S=1,H=1|F=1)*P(F=1)+P(S=1,H=1|F=0)*P(F=0)} =$

b. $\frac{P(S=1|F=1)*P(H=1|F=1)*P(F=1)}{P(S=1|F=1)*P(H=1|F=1)*P(F=1)+P(S=1|F=0)*P(H=1|F=0)*P(F=0)} =$

c. $\frac{0.9*0.99*0.1}{0.9*0.99*0.1+0.001*0.0001*0.9} \approx 0.9999989899$

d. Observing both Heat and Smoke, the Fire is almost guaranteed with a probability very close to 1. Heat, Smoke and Fire being 1 are highly correlated, therefore the probability of P(F = 0|S = 1) was very low and the probability of $P(F = 1|S = 1, H = 1)$ is really high.

Problem 3

Consider a Bayesian network with 2 variables and the following structure: A → B. We observed 4 samples for these variables:

A	B
1	1
0	0
1	1
0	0

(A) Calculate the maximum likelihood (ML) estimator of the parameter θ = P(B = 1|A). Recall $\hat{\theta}_{ML} = \max_{\theta} \mathrm{P(D|G)}$, where D =Data, G =Graph(Model).

a. $\theta_{0B} = \mathrm{P(B = 1|A = 0)}$

b. $\theta_{0B} = \mathrm{P(B = 1|A = 0)}$

c. $\theta_{A} = \mathrm{P(A = 1)}$

d. $\mathrm{P(D|G,\theta_A)} = \binom{4}{2} * \theta_A^2 * (1-\theta_A)^{4-2} => 6 * \theta_A^2 * (1-\theta_A)^2 = 0$

   i. $\theta_A = 0 \lor \theta_A = 1$

e. $\mathrm{P(D|G,\theta_{0B})} = \binom{2}{0} * \theta_{0B}^0 * (1-\theta_{0B})^2 => (1-\theta_{0B})^2 = 0$

   i. $\theta_{0B} = 1$

f. $\mathrm{P(D|G,\theta_{1B})} = \binom{2}{2} * \theta_{1B}^2 * (1-\theta_{1B})^0 => \theta_{1B}^2 = 0$

   i. $\theta_{1B} = 0$

(B) Calculate the model evidence P(D|G). The prior distribution of θ is: P(θ) = P(θ|G) ~ β(3, 3).

a. $P(D|G) = \int P(D|\theta, G) * P(\theta|G)\, d\theta = \int P(D|\theta_A, G) * P(\theta_A|G)\, d\theta_A \int P(D|\theta_{0B}, G) * P(\theta_{0B}|G)\, d\theta_{0B} * \int P(D|\theta_{1B}, G) * P(\theta_{1B}|G)\, d\theta_{1B}$

b. For each factor we have the generic integral:

c. $\int \binom{n}{x} * \theta^x * (1-\theta)^{n-x} * \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} * \theta^{\alpha-1} * (1-\theta)^{\beta-1}\, d\theta$

d. $= \int \binom{n}{x} * \theta^{x+\alpha-1} * (1-\theta)^{n-x+\beta-1} * \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\, d\theta$

e. $= \int \binom{n}{x} * \theta^{x+\alpha-1} * (1-\theta)^{n-x+\beta-1} * \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\, d\theta$

f. $= \binom{n}{x} * \frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(\alpha+n+\beta)} * \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$

g. And now we insert the parameter counts:

h. $P(D|G) = \binom{4}{2} * \frac{\Gamma(2+3)\Gamma(4-2+3)}{\Gamma(3+4+3)} * \frac{\Gamma(3+3)}{\Gamma(3)\Gamma(3)} * \binom{2}{0} * \frac{\Gamma(0+3)\Gamma(2-0+3)}{\Gamma(3+2+3)} * \frac{\Gamma(3+3)}{\Gamma(3)\Gamma(3)} * \binom{2}{2} * \frac{\Gamma(2+3)\Gamma(2-2+3)}{\Gamma(3+2+3)} * \frac{\Gamma(3+3)}{\Gamma(3)\Gamma(3)} = 6 * \left(\frac{\Gamma(3+3)}{\Gamma(3)\Gamma(3)}\right)^3 * \frac{\Gamma(5)\Gamma(5)}{\Gamma(10)} * \frac{\Gamma(3)\Gamma(5)}{\Gamma(8)} * \frac{\Gamma(5)\Gamma(3)}{\Gamma(8)} = 6 * \left(\frac{120}{4}\right)^3 * \frac{24^2}{362880} * \frac{48}{5040} * \frac{48}{5040} = \frac{2}{77175} \approx 0.00002$

(C) Calculate the posterior distribution P(θ|D, G) for both estimators (use the prior distribution from before). Again, estimate the parameter θ, but this time using the posterior distribution. Compare them to the ML estimators.

a. The posterior is proportional to $\theta^{x+\alpha-1} * (1-\theta)^{n-x+\beta-1}$
b. $P(\theta_A|G,D) \propto \theta^{2+3-1} * (1-\theta)^{4-2+3-1} => P(\theta_A|G,D) = \beta(5,5)$
c. $\mathrm{P}(\theta_{0\mathrm{B}}|\mathrm{G},\mathrm{D}) \propto \theta^{0+3-1} * (1-\theta)^{2-0+3-1} => P(\theta_{0B}|G,D) = \beta(3,5)$
d. $\mathrm{P}(\theta_{1\mathrm{B}}|\mathrm{G},\mathrm{D}) \propto \theta^{2+3-1} * (1-\theta)^{2-2+3-1} => P(\theta_{1B}|G,D) = \beta(5,3)$
e. We compare the estimators using the expected value of the beta distributions to the ML Estimators:
    i. $\theta_A => E[\beta(5,5)] = \frac{\alpha}{\alpha+\beta} = 0.5 \neq 0 \vee 1 = \hat{\theta}_A{}^{ML}$
    ii. $\theta_{0B} => E[\beta(3,5)] = \frac{3}{8} \neq 1 = \hat{\theta}_{0B}{}^{ML}$
    iii. $\theta_{1B} => E[\beta(5,3)] = \frac{5}{8} \neq 0 = \hat{\theta}_{1B}{}^{ML}$

Problem 4

by Paul Vogler (4979420), Yiftach Kolb (5195763)

Part A Computing the Likelihood

The likelihood is $L(\theta) = binom(10, 8, \theta)$ and the maximum likelihood is the observed success ferquency, which is $\hat{\theta} = 0.8$

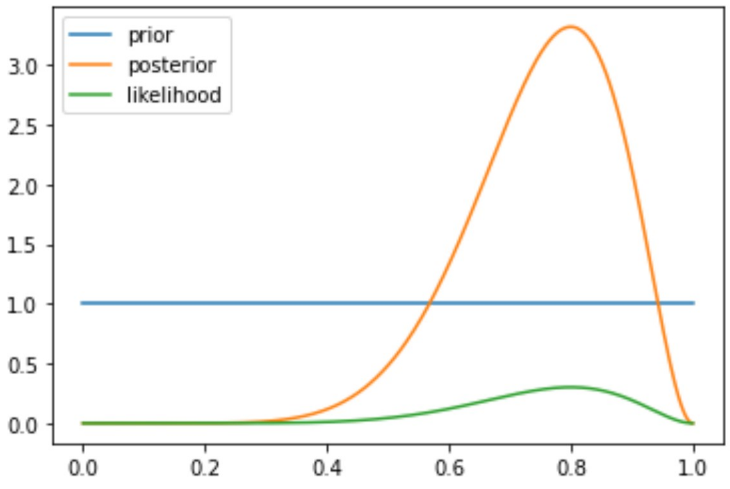
Part B: Posterior

The posterior probability is of a beta distribution and a binomial rv is again beta: $\beta(1 + 8, 1 + 2)$

Part B: Plots

```
In [1]: import numpy as np
        import pandas as pd
        from scipy.stats import beta, binom

        import matplotlib.pyplot as plt

        ## Problem 4
        ### Part A
        likel = lambda t: binom.pmf(8,10,t)

        # the maximum likelihood for binomial rv is
        # the observed success rate, so:
        tmax = 8/10

        ### Part B
        prior = lambda t: beta.pdf(t, 1, 1)

        post = lambda t: beta.pdf(t, 1+8, 1+2)

        xs = np.linspace(0,1,500)

        plt.plot(xs, prior(xs))
        plt.plot(xs,post(xs))
        plt.plot(xs, likel(xs))
        plt.legend(['prior', 'posterior', 'likelihood'])
```

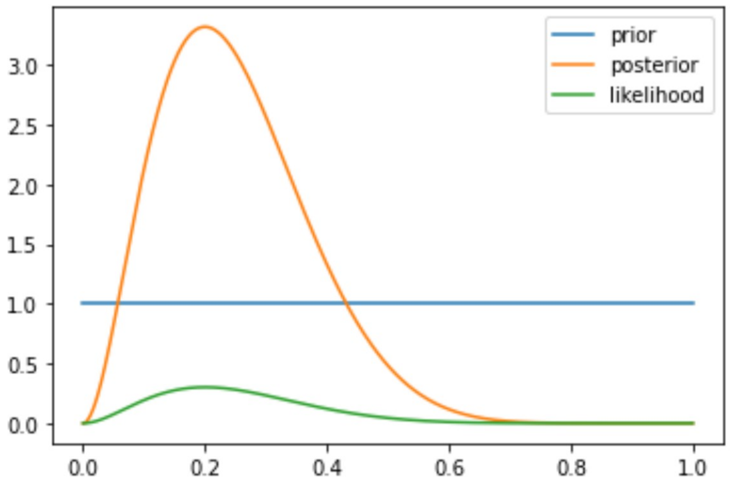
```
Out[1]: <matplotlib.legend.Legend at 0x7fcf5cd77e20>
```



Part D, repeat

In [2]:
```
import numpy as np
import pandas as pd
from scipy.stats import beta, binom

import matplotlib.pyplot as plt

## Problem 4
### Part A
likel = lambda t: binom.pmf(2,10,t)

# the maximum likelihood for binomial rv is
# the observed success rate, so:
tmax = 2/10

### Part B
prior = lambda t: beta.pdf(t, 1, 1)

post = lambda t: beta.pdf(t, 1+2, 1+8)

xs = np.linspace(0,1,500)

plt.plot(xs, prior(xs))
plt.plot(xs,post(xs))
plt.plot(xs, likel(xs))
plt.legend(['prior', 'posterior', 'likelihood'])
```

Out[2]:
```
<matplotlib.legend.Legend at 0x7fcf320e7d90>
```

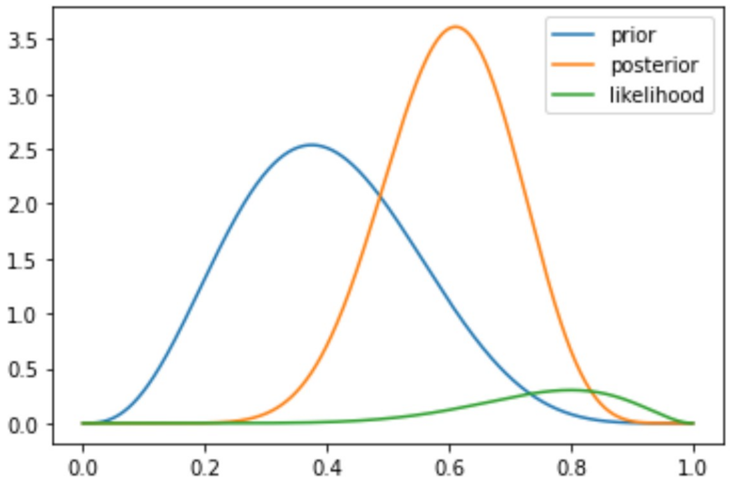


Part E

The Prior is different and consequently the posterior. It still shifts towards the true location, and the beta becomes narrower as the a, b increase.

In [3]:
```python
import numpy as np
import pandas as pd
from scipy.stats import beta, binom

import matplotlib.pyplot as plt

## Problem 4
### Part A
likel = lambda t: binom.pmf(8,10,t)

# the maximum likelihood for binomial rv is
# the observed success rate, so:
tmax = 8/10

### Part B
prior = lambda t: beta.pdf(t, 4, 6)

post = lambda t: beta.pdf(t, 4+8, 6+2)

xs = np.linspace(0,1,500)

plt.plot(xs, prior(xs))
plt.plot(xs,post(xs))
plt.plot(xs, likel(xs))
plt.legend(['prior', 'posterior', 'likelihood'])
```

Out[3]:
```
<matplotlib.legend.Legend at 0x7fcf320e7dc0>
```



In [ ]: