Exercise 1

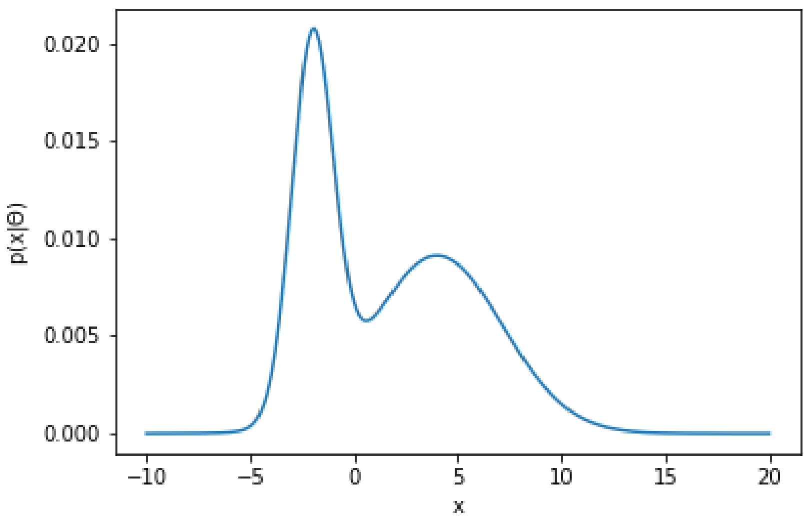
Task A

The PDF for this generative model is set to be the sum of two Gaussian distributions:

$$\begin{aligned} p(x|\Theta) &= \pi_1 \cdot \mathcal{N}(x_n; \mu_1, \sigma_1^2) + \pi_2 \cdot \mathcal{N}(x_n; \mu_2, \sigma_2^2) \\ &= \pi_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x_n - \mu_1)^2}{2\sigma_1^2}\right) + \pi_2 \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x_n - \mu_2)^2}{2\sigma_2^2}\right) \end{aligned}$$

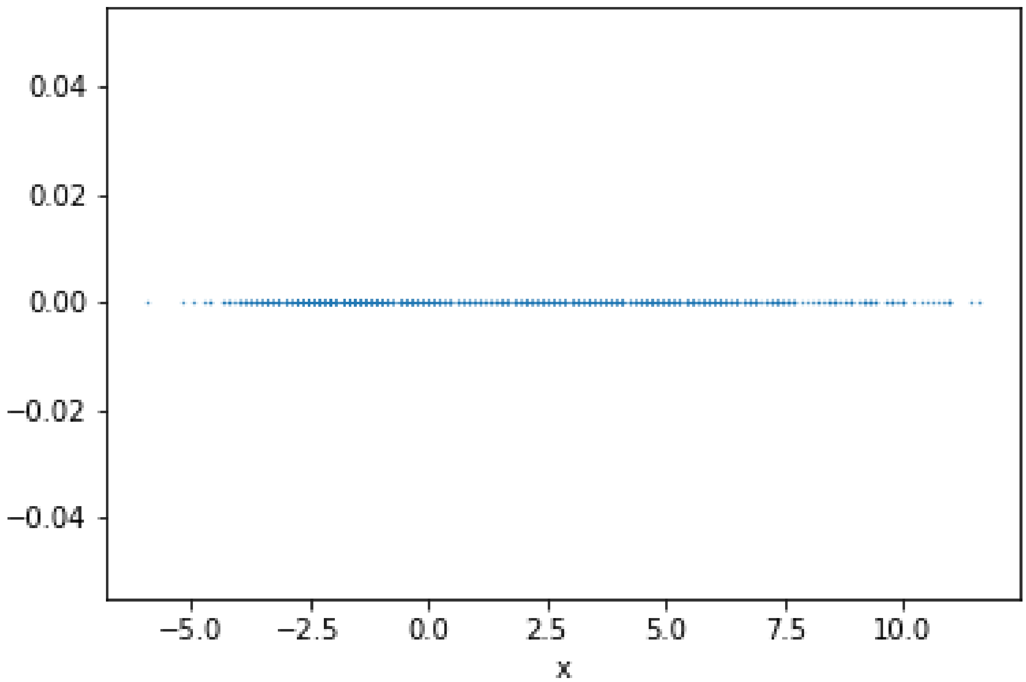
Task B

Using the given parameters, the respective PDF s plotted



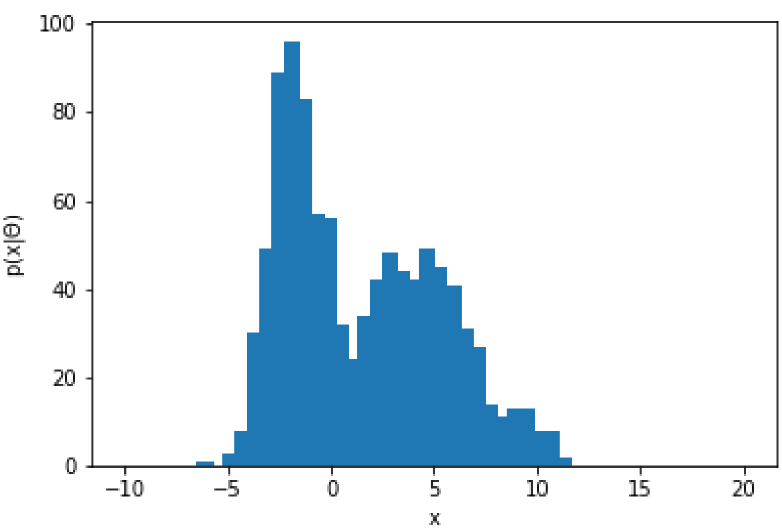
Task C

For this task, $N = 1000$ datapoints are supposed to be plotted from the generative model with the given parameters from *Task B*. This is done using the python library *numpy*, vie the function *numpy.random.choice()*. The generated 1D-distribution looks like:



Task D

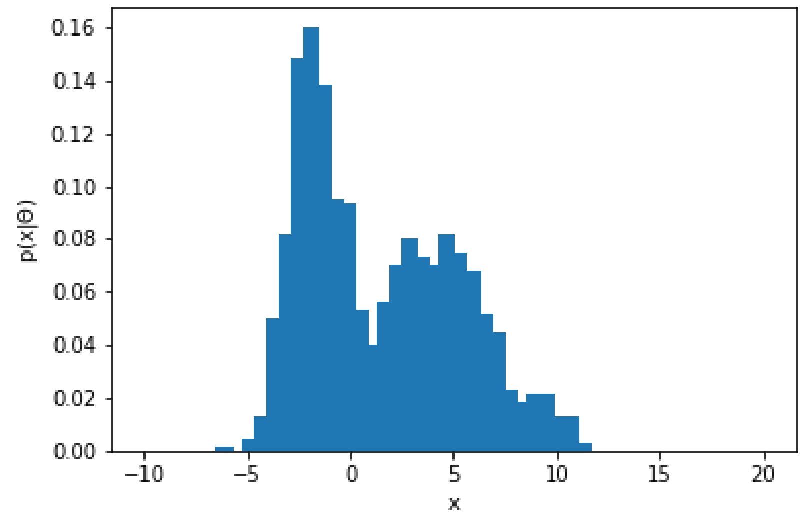
The histogram is computed via by counting the numbers per bin n_i using python (see code attached at the end).



Task E

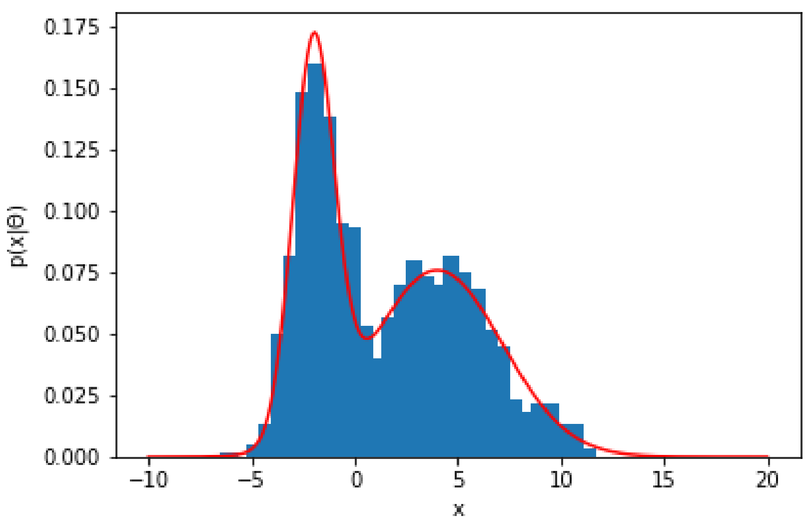
The normalized histogram is computed via the sampling formula.

$$p_x = \frac{n_i}{N \cdot \Delta}$$

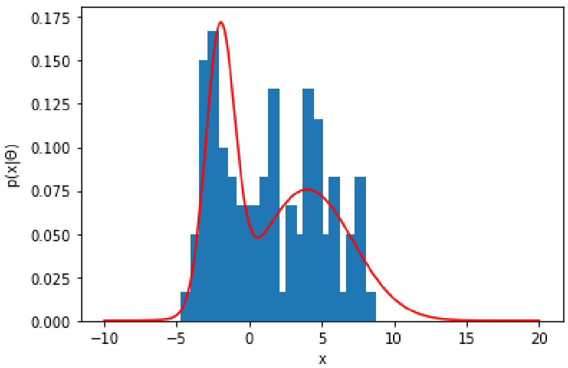


Task F

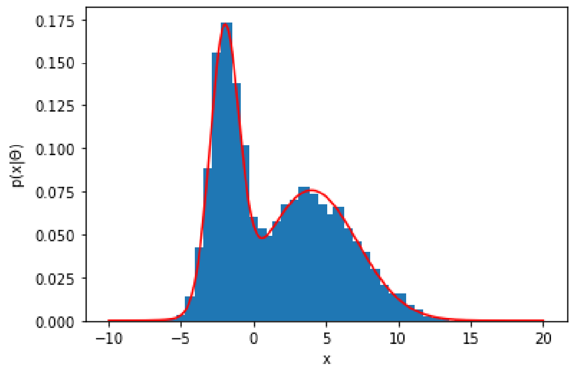
For this task, the PDF derived from *Task B* is added to the plot from *Task E*:



It is observed, that for less datapoints (left) the PDF fits worse and form more points (right), the histogram and PDF get more similar.

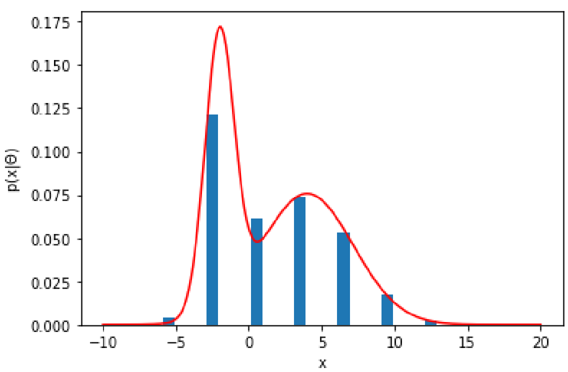


(a) $N = 100$

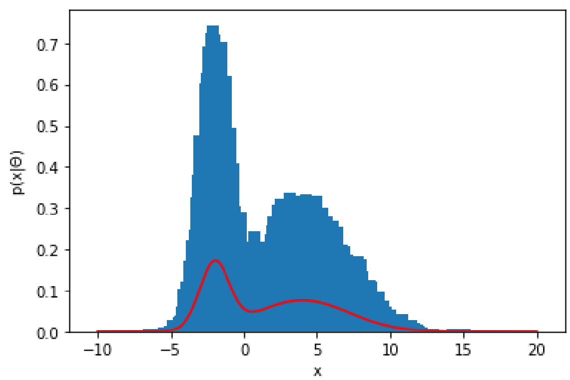


(b) $N = 10000$

By choosing less bins, the histogram seems to match the PDF quite ok, while the histogram clearly overshoots the PDF at higher bin numbers.



(a) $N_{\text{bins}} = 10$



(b) $N_{\text{bins}} = 1000$

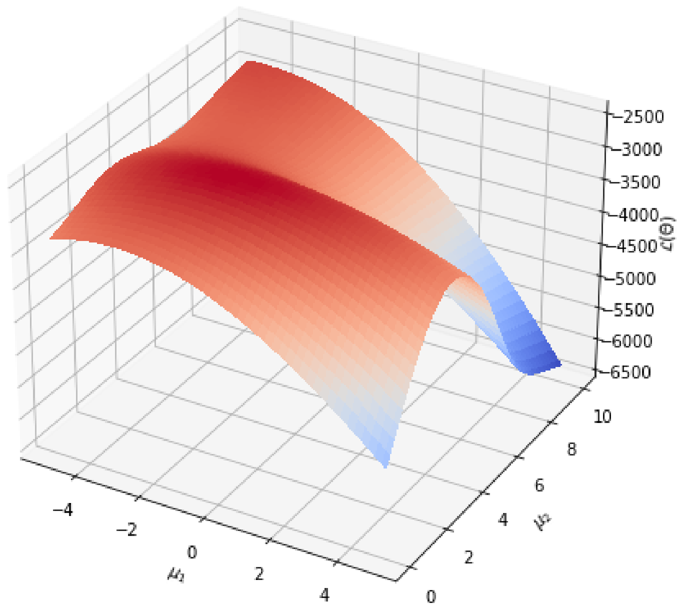
Task G

In this task it is required to compute the log-likelyhood $\mathcal{L}(\Theta)$ for the generated sample and the given parameters. It is computed to be:

$$\mathcal{L}(\Theta) = \sum_n \log\left(p(x_n|\Theta)\right) = -2601.998$$

Task H

The log-likelyhood computed in *Task G* is found to be the highest possible, when changing the parameters for the computation. The figure below shows this behaviour exemplary for changing mean values μ_1 and μ_2. Here its is clearly seen, that the further the parameters deviate from the given parameters from *Task B*, the lower is the log-likelyhood. This is consistent with the lecture, as $\mathcal{L}(\Theta)$ should be maximised to obtain the best fitting parameters.



The code for this exercise is found attached at the end.

Exercise 2

Task A

$$\Theta^{\text{new}} = \text{argmax}_{\Theta} F(q_{\text{new}}, \Theta),$$

$$F(q_{\text{new}}, \Theta) = \sum_n \sum_c q_{(n)}(c) \log\left(\frac{p(\overrightarrow{x}^{(n)}, c|\Theta)}{q_{(n)}(c)}\right),$$

$$\log(p(\overrightarrow{x}^{(n)}, c|\Theta)) = \log(p(\overrightarrow{x}^{(n)}, c|\Theta)) + \log(p(\overrightarrow{x}^{(n)}, c|\Theta)$$

$$= -0.5\log(2\pi) - 0.5\log(\det(\Sigma_c)) - 0.5(\overrightarrow{x}^{(\text{n})} - \overrightarrow{\mu}_c)^{\text{T}}\Sigma_c^{-1}(\overrightarrow{x}^{(n)} - \overrightarrow{\mu}_c) + \log(\pi_c)$$

$$\rightarrow F(q, \Theta) = \sum_n \sum_{c'} q_{(n)}(c')(-0.5\log(2\pi) -$$
$$0.5\log(\det(\Sigma_c)) - 0.5(\overrightarrow{x}^{(\text{n})} - \overrightarrow{\mu}_c)^{\text{T}}\Sigma_c^{-1}(\overrightarrow{x}^{(n)} - \overrightarrow{\mu}_c) + \log(\pi_c)) - \log(q_{(n)}(c')))$$

$$\tfrac{\partial F(q,\Theta)}{\partial \overrightarrow{\mu}_c} = \sum_n \sum_{c'} q_{(n)}(c')\Sigma_{c'}^{-1}(\overrightarrow{x}^{(n)} - \overrightarrow{\mu}_{c'})\delta_{cc'}$$

$$= \sum_n q_{(n)}(c)\Sigma_c^{-1}(\overrightarrow{x}^{(n)} - \overrightarrow{\mu}_c) \overset{!}{=} 0$$

$$\rightarrow \overrightarrow{\mu}_c^{\text{new}} = \frac{\sum_n q(c; \overrightarrow{x}^{(n)}, \Theta^{\text{old}})\overrightarrow{x}^{(n)}}{\sum_n q(c; \overrightarrow{x}^{(n)}, \Theta^{\text{old}})}$$

Code

```python
#%% Import

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import cm


#%% 1B

Theta = pd.DataFrame(index = ["c=1", "c=2"], columns=["pi", "mu", "sigma"])

Theta["pi"] = [0.4, 0.6]
Theta["mu"] = [-2, 4]
Theta["sigma"] = [1, 10]


def Gaussian(x, mu, sigma):
    return 1/np.sqrt(2*np.pi*sigma) * np.exp(-(x-mu)**2/(2*sigma))



x = np.linspace(-10, 20, 250)
dx = x[1] - x[0]
p_x = Theta.loc["c=1", "pi"] * Gaussian(x, Theta.loc["c=1", "mu"], Theta.loc["c=1", "sigma"]) +  Theta.loc["c=2",

plt.plot(x, p_x*dx)
plt.xlabel("x")
plt.ylabel("p(x|$\Theta$)")
plt.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex3_1B.png")
plt.show()

# %% 1C
N=10000

sample = np.random.choice(x, p=p_x/np.sum(p_x), size=N)
plt.scatter(sample, [0]*N, marker=".", s = 0.5)
plt.xlabel("x")
plt.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex_3_1_C.png")
plt.show()

# %% 1D-F

# Histogram via Sampling formula

edges = np.linspace(-10, 20,7)
p_x_i = []
n_i = []

for idx, edge_up in enumerate(edges[1:]):
    edge_low = edges[idx]

    n_i.append(len([i for i in sample if edge_low<=i<edge_up]))
    d_bins = edge_up-edge_low

    p_x_i.append(n_i[-1] / (N * d_bins) )
```

```python
center = (edges[:-1] + edges[1:])/2

# D
plt.bar(center, n_i)
# plt.hist(sample, bins=50, range=[-10,20])
plt.xlabel("x")
plt.ylabel("p(x|$|Theta$)")
plt.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex_3_1_D.png")
plt.show()


#E, F
# plt.hist(sample, bins=50, range=[-10,20], density=True)
plt.bar(center, p_x_i)
plt.xlabel("x")
plt.ylabel("p(x|$|Theta$)")
plt.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex_3_1_E.png")
plt.plot(x, p_x, c="red")
plt.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex_3_1_F.png")
plt.show()

# %% 1G-H

p_x = Theta.loc["c=1", "pi"] * Gaussian(sample, Theta.loc["c=1", "mu"], Theta.loc["c=1", "sigma"]) +  Theta.loc["c

Log_l = np.sum(np.log(p_x))

print(Log_l)


Theta2 = pd.DataFrame(index = ["c=1", "c=2"], columns=["pi", "mu", "sigma"])

Theta2["pi"] = [0.4, 0.6]
Theta2["mu"] = [-1, 5]
Theta2["sigma"] = [1, 7]

# pi1 = np.linspace(0,1,10)
# pi2 = 1-pi1

mu1 = np.linspace(-5,5,100)
mu2 = np.linspace(0,10,100)
ll = pd.DataFrame(index = mu1, columns=mu2)

for m in mu1:
    for m2 in mu2:
        p_x_m = Theta.loc["c=1", "pi"] * Gaussian(sample, m, Theta.loc["c=1", "sigma"]) +  Theta.loc["c=2", "pi"]
        ll.loc[m, m2] = np.sum(np.log(p_x_m))
        # ll.append(np.sum(np.log(p_x_m)))
    print(m)


mm1, mm2 = np.meshgrid(mu1, mu2)
LL = ll.to_numpy(dtype="float64")

fig, ax = plt.subplots(subplot_kw={"projection": "3d"}, figsize=[7.5,7.5])

ax.plot_surface(mm1, mm2, LL, cmap = cm.coolwarm,  linewidth=0, antialiased=False)
ax.set_xlabel("$|mu_1$")
ax.set_ylabel("$|mu_2$")
ax.set_zlabel("$|mathcal{L}(|Theta)$")

ax.scatter(-2, 4, np.max(LL)+10, marker="x")


fig.savefig("/home/johannes/Dokumente/Uni/Ma_3rd Semester/Unsupervised_ML/Homework/Ex_3_LL_plot.png")
```