

# Machine Learning

Summer Semester 2019, Homework 2

Prof. Dr. J. Peters, H. Abdulsamad, S. Stark, D. Koert



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

**Total points: 90 + 15 bonus**

Due date: Friday, 14 June 2019 17:00

Hand in a PDF over Moodle and a printed version to the postbox at (S2/02 | E315)

Name, Surname, ID Number

Steffen Schäfer, 2635897

Peter Nickl, 1941346

---

## Problem 2.1 Bayesian Decision Theory [20 Points]

---

In this exercise, we consider data generated by a mixture of two Gaussian distributions with parameters  $\{\mu_1, \sigma_1\}$  and  $\{\mu_2, \sigma_2\}$ . Each Gaussian represents a class labeled  $C_1$  and  $C_2$ , respectively.

**a) Optimal Boundary [4 Points]**

Explain in one short sentence what Bayesian Decision Theory is. What is its goal? What condition does hold at the optimal decision boundary? When do we decide for class  $C_1$  over  $C_2$ ?

*In Bayesian decision theory a decision is made using the degree of belief in an outcome.*

*It's goal is it to minimize the risk presented in a loss function.*

*At the decision boundary  $p(C_1|x) = p(C_2|x)$  holds true.*

*We decide for  $C_1$  over  $C_2$  if  $P(C_1|x) > p(C_2|x)$*

**b) Decision Boundaries [8 Points]**

If both classes have equal prior probabilities  $p(C_1) = p(C_2)$  and the same variance  $\sigma_1 = \sigma_2$ , derive the decision boundary  $x^*$  analytically as a function of the two means  $\mu_1$  and  $\mu_2$ .

Since the prior are equal following equations holds true we decide for  $C_1$  if:

$$\frac{p(x|C_1)}{p(x|C_2)} > 1 \quad (8)$$

At the point of the decision boundary we can say:

$$p(x|C_1) = p(x|C_2) \quad (9)$$

Both datasets are gaussian distributed.

$$\mathcal{N}(x|\mu_1, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_1)^2\right) \quad (10)$$

$$\mathcal{N}(x|\mu_2, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_2)^2\right) \quad (11)$$

Using equation 10 and 11 in equation 9 we get:

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_1)^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_2)^2\right) \quad (12)$$

$$(x - \mu_1)^2 = (x - \mu_2)^2 \quad (13)$$

$$x = \frac{\mu_1 + \mu_2}{2} \quad (14)$$

## c) Different Misclassification Costs [8 Points]

Assume  $\mu_1 > 0$ ,  $\mu_1 = 2\mu_2$ ,  $\sigma_1 = \sigma_2$  and  $p(C_1) = p(C_2)$ . If misclassifying sample  $x \in C_2$  as class  $C_1$  is three times more expensive than the opposite, how does the decision boundary change? Derive the boundary analytically. (There is no cost for correctly classifying samples.)

We want to minimize following risk:

$$\frac{p(x|C_1)}{p(x|C_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \cdot \frac{p(C_2)}{p(C_1)} \quad (26)$$

With

$$\lambda_{ij} = \lambda(\alpha_i | \alpha_j) \quad (27)$$

The main diagonal of  $\lambda$  so  $\lambda_{ii}$  is the cost for correctly classifying. In our case this is zero, so  $\lambda_{11} = \lambda_{22} = 0$ . Now its stated that missclassifying sample  $x \in C_2$  as class  $C_1$  is three times more expensive than the opposite, meaning that  $\lambda_{12} = 3 \cdot \lambda_{21}$ . We can now use this information for equation 27.

$$\frac{p(x|C_1)}{p(x|C_2)} > \frac{3\lambda_{21}}{\lambda_{21}} \cdot \frac{p(C_2)}{p(C_1)} = 3 \quad (28)$$

Now we can use the equations for the gaussian distribution 10 and 11 and get:

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_1)^2\right) = 3 \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_2)^2\right) \quad (29)$$

$$\exp\left(-\frac{1}{2\sigma^2}(x - \mu_1)^2\right) = 3 \cdot \exp\left(-\frac{1}{2\sigma^2}(x - \mu_2)^2\right) \quad (30)$$

Using the log to get rid of the exponential function.

$$-\frac{1}{2\sigma^2}(x - \mu_1)^2 = \ln(3) - \frac{1}{2\sigma^2}(x - \mu_2)^2 \quad (31)$$

We now use  $\mu_1 = 2\mu_2$  and continue.

$$-\frac{1}{2\sigma^2}(x^2 - 4x\mu_2 + 4\mu_2^2) + \frac{1}{2\sigma^2}(x^2 - 2x\mu_2 + \mu_2^2) = \ln(3) \quad (32)$$

$$-\frac{1}{2\sigma^2}(x^2 - 4x\mu_2 + 4\mu_2^2) + \frac{1}{2\sigma^2}(-x^2 + 2x\mu_2 - \mu_2^2) = \ln(3) \quad (33)$$

Transforming the terms inside the brackets brings us to

$$-2x\mu_2 + 3\mu_2^2 = \ln(3) \cdot (-2\sigma^2) \quad (34)$$

$$-2x\mu_2 = -2\ln(3)\sigma^2 - 3\mu_2^2 \quad (35)$$

After transposing this equation we get:

$$x = \frac{3\mu_2^2 + 2\ln(3) \cdot \sigma^2}{2\mu_2} \quad (36)$$

## Problem 2.2 Density Estimation [30 Points + 15 Bonus ]

In this exercise, you will use the datasets `densEst1.txt` and `densEst2.txt`. The datasets contain 2D data belonging to two classes,  $C_1$  and  $C_2$ .

## a) Gaussian Maximum Likelihood Estimation [10 Points]

Derive the ML estimate for the mean and covariance of the **multivariate** Gaussian distribution. Start your derivations with the function you optimize. Assume that you can collect i.i.d data. (Hint: you can find many matrix identities on the Matrix Cookbook (<https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>) and at [http://en.wikipedia.org/wiki/Matrix\\_calculus](http://en.wikipedia.org/wiki/Matrix_calculus).)

The Multivariate Gaussian Distribution is defined as

$$f_x(\mu|\Sigma) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \quad (59)$$

$\Sigma$  is the Covariance Matrix defined as:

$$\begin{pmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_{yy}^2 \end{pmatrix} \quad (60)$$

The resulting likelihood function of  $f_x(x)$  is:

$$L(\mu, \Sigma) = \prod_{n=1}^N f_x(\mu|\Sigma) \quad (61)$$

We now take the log of the likelihood function.

$$\log L(\mu, \Sigma) = \sum_{n=1}^N \log f_x(\mu|\Sigma) \quad (62)$$

Resulting in the log likelihood function for a multivariate gaussian distribution.

$$\log L(\mu, \Sigma) = \sum_{n=1}^N \log\left(\frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)\right) \quad (63)$$

Now we want to derive the mean and covariance for this distribution. Starting with the mean, we need to maximize the partial equation.

$$\frac{\partial L}{\partial \mu} = \sum_{n=1}^N \underbrace{\log\left(\frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}}\right)'}_0 + \sum_{n=1}^N \left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)' \quad (64)$$

For matrix Calculus we use following rules:

$$\frac{\partial (Xa+b)^T C (Xa+b)}{\partial x} = (C + C^T)(Xa+b)a^T \quad (65)$$

With  $X = x$ ,  $a = 1$ , thus  $a^T = 1$  and  $b = -\mu$ . We get

$$\sum_{n=1}^N \left(\Sigma + (\Sigma^{-1})^T (x-\mu)\right) = 0 \quad (66)$$

Since  $\Sigma$  is a symmetrical Matrix  $\Sigma + \Sigma^T = 2 \cdot \Sigma$  holds true. This matrix is also constant, meaning its independent on the sum and can be written before it.

$$2\Sigma^{-1} \cdot \sum_{n=1}^N (x - \mu) = 0 \quad (67)$$

$$\sum_{n=1}^N x_i = \sum_{n=1}^N \mu \quad (68)$$

Since  $\mu$  is a constant we can write the sum over a constant as  $N \cdot \mu$

$$\sum_{n=1}^N x_i = N \cdot \mu \quad (69)$$

Resulting in the following equation for the mean:

$$\mu = \frac{1}{N} \sum_{n=1}^N x_i \quad (70)$$

Now we want to derive the maximum Likelihood estimate for the covariance of the multivariate Gaussian distribution. Again we Start with Equation (63). For this derivation we will use the following rules for matrix calculus.

$$\frac{\partial}{\partial A} \log(\det(A)) = (A^{-1})^T \quad (71)$$

$$\frac{\partial}{\partial A} x^T A x = x x^T \quad (72)$$

With  $A = \Sigma$  and  $x = (x - \mu)$ . First we split the log of gaussian equation up into three part for easier derivation.

$$\frac{\partial L}{\partial \Sigma} = \underbrace{\log\left(\frac{1}{\sqrt{2\pi}^p}\right)'}_a + \sum_{n=1}^N \left( \underbrace{\log\left(\frac{1}{\sqrt{\det(\Sigma)}}\right)'}_b + \underbrace{\log \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)'}_c \right) \quad (73)$$

For a:

$$\frac{\partial L_a}{\partial \Sigma} = 0 \quad (74)$$

For b:

$$\log\left(\frac{1}{\sqrt{\det(\Sigma)}}\right) = \log\left(\frac{1}{\det(\Sigma)^{\frac{1}{2}}}\right) = \frac{1}{2} \cdot \log\left(\frac{1}{\det(\Sigma)}\right) \quad (75)$$

$$\frac{\partial L_b}{\partial \Sigma} = \frac{1}{2} \cdot \frac{1}{(\Sigma^{-1})^T} = \frac{1}{2} \cdot \Sigma \quad (76)$$

For c:

$$\frac{\partial L_c}{\partial \Sigma} = -\frac{1}{2} (x - \mu) \cdot (x - \mu)^T \quad (77)$$

In total we get the following derivative:

$$\frac{\partial L}{\partial \Sigma} = \sum_{n=1}^N \frac{1}{2} \Sigma - \sum_{n=1}^N \left( \frac{1}{2} (x - \mu) \cdot (x - \mu)^T \right) = 0 \quad (78)$$

Multiplied by  $\frac{1}{2}$ :

$$\sum_{n=1}^N \Sigma = \sum_{n=1}^N ((x - \mu) \cdot (x - \mu)^T) \quad (79)$$

Since  $\Sigma$  is constant we can write  $\sum_{n=1}^N \Sigma = N \cdot \Sigma$ . Resulting in the final equation for the covariance

$$\Sigma = \frac{1}{N} \sum_{n=1}^N (x_i - \mu) \cdot (x_i - \mu)^T \quad (80)$$

#### b) Prior Probabilities [2 Points]

Compute the prior probability of each class from the dataset.

We count for class 1:  $n_1 = 239$  and for class 2:  $n_2 = 761$ . Resulting in following priors

$$p(C_1) = \frac{239}{235 + 761} = 0.239 \quad (83)$$

$$p(C_2) = \frac{761}{235 + 761} = 0.761 \quad (84)$$

## c) Biased ML Estimate [5 Points]

Define the bias of an estimator and write how we can compute it. Then calculate the biased and unbiased estimates of the conditional distribution  $p(x|C_i)$ , assuming that each class can be modeled with a Gaussian distribution. Which parameters have to be calculated? Show the final result and attach a snippet of your code. Do not use existing functions, but rather implement the computations by yourself!

The bias of an estimator  $\Theta$  measures the difference between the expected value of an estimator and its' true value. It is defined as:

$$\text{Bias}(\hat{\theta}) = E(\hat{\theta}) - \theta$$

Assuming a Gaussian distribution, the biased and unbiased estimates of the conditional distributions can be calculated using the following formula:

$$\begin{aligned}\bar{\mathbf{x}}_{\text{unbiased}} &= \frac{1}{N} \sum_{i=1}^n \mathbf{x}_i \\ \hat{\Sigma}_{\text{unbiased}} &= \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \\ \hat{\Sigma}_{\text{biased}} &= \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\end{aligned}$$

We thus obtain the following values for the given dataset.

$$\begin{aligned}\bar{\mathbf{x}}_{1,\text{unbiased}} &= \begin{bmatrix} -0.70681374 \\ -0.81343083 \end{bmatrix} \\ \hat{\Sigma}_{1,\text{unbiased}} &= \begin{bmatrix} 9.05742302 & 2.6841014 \\ 2.6841014 & 3.61145033 \end{bmatrix} \\ \hat{\Sigma}_{1,\text{biased}} &= \begin{bmatrix} 9.01952586 & 2.67287085 \\ 2.67287085 & 3.59633965 \end{bmatrix} \\ \bar{\mathbf{x}}_{2,\text{unbiased}} &= \begin{bmatrix} 3.98534252 \\ 3.98438364 \end{bmatrix} \\ \hat{\Sigma}_{2,\text{unbiased}} &= \begin{bmatrix} 4.18087542 & 0.02761954 \\ 0.02761954 & 2.75658555 \end{bmatrix} \\ \hat{\Sigma}_{2,\text{biased}} &= \begin{bmatrix} 4.1753815 & 0.02758324 \\ 0.02758324 & 2.75296323 \end{bmatrix}\end{aligned}$$

The code is attached.

```
# import necessary Python packages
import os
import numpy as np

# set the working directory
os.chdir("C:/Users/pistl/Desktop/hw2/")

# print the current working directory
os.getcwd()

# load data
densEst1 = np.loadtxt(fname = "C:/Users/pistl/Desktop/hw2/dataSets/densEst1.csv")
densEst2 = np.loadtxt(fname = "C:/Users/pistl/Desktop/hw2/dataSets/densEst2.csv")
```

---

```

def sample_mean(data):
    N = np.size(data,0)
    column_sums = np.sum(data, axis=0)
    means = 1/N * column_sums
    return means

def sample_variance_biased(data,means):
    N = np.size(data,0)
    x = data - means
    variances_biased = 1/N * np.matmul(np.transpose(x),x)
    return variances_biased

def sample_variance_unbiased(data,means):
    N = np.size(data,0)
    denominator = N - 1
    x = data - means
    variances_biased = 1/denominator * np.matmul(np.transpose(x),x)
    return variances_biased

print('ESTIMATORS OF DATASET densEst1\n')
means = sample_mean(densEst1)
print('means\n',means)
print('\nvariance_biased\n',sample_variance_biased(densEst1,means))
print('\nvariance_unbiased\n',sample_variance_unbiased(densEst1,means))

print('-----')

print('ESTIMATORS OF DATASET densEst2\n')
means = sample_mean(densEst2)
print('means\n',means)
print('\nvariance_biased\n',sample_variance_biased(densEst2,means))
print('\nvariance_unbiased\n',sample_variance_unbiased(densEst2,means))

```

---

d) **Class Density [5 Points]**

Using the unbiased estimates from the previous question, fit a Gaussian distribution to the data of each class. Generate a single plot showing the data points and the probability densities of each class. (Hint: use the contour function for plotting the Gaussians.)

e) **Posterior [8 Points]**

In a single graph, plot the posterior distribution of each class  $p(C_i|x)$  and show the decision boundary.

f) **Bayesian Estimation [15 Bonus Points]**

State the generic case of Bayesian linear regression with data  $\langle X, Y \rangle$  and parameters  $\theta$ . What do we assume about the data, the model and the parameters?

Formulate the posterior distribution for your model parameters given the data, i.e.,  $p(\theta|X, Y)$ , and derive its mean and covariance, assuming that the model of the output variable is a Gaussian distribution with a fixed variance.

What do we do when we want to predict a new point?

Which are the advantages of being Bayesian?



---

**Problem 2.3 Non-parametric Density Estimation [20 Points]**


---

In this exercise, you will use the datasets `nonParamTrain.txt` for training and `nonParamTest.txt` for evaluating the performance of your model.

**a) Histogram [4 Points]**

Compute and plot the histograms using 0.02, 0.5, 2.0 size bins of the training data. Intuitively, indicate which bin size performs the best and explain why. Knowing only these three trials, would you be sure that the one you picked is truly the best one? Attach the plot of the histograms and a snippet of your code.

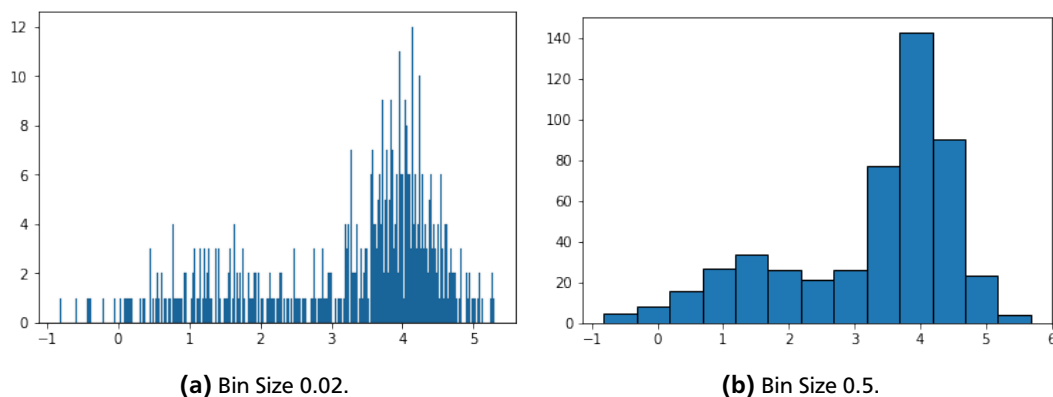
The first figure 1a shows a Histogram of the data with bin size 0.02. The "spicky" behavior and empty bins which are consistend throughout the Histogram, indicate that the bin size has been chosen to small (to much Variance).

In the next Figure 1b you can imagine a trend for the underlying distribution. This indicates a good bin size.

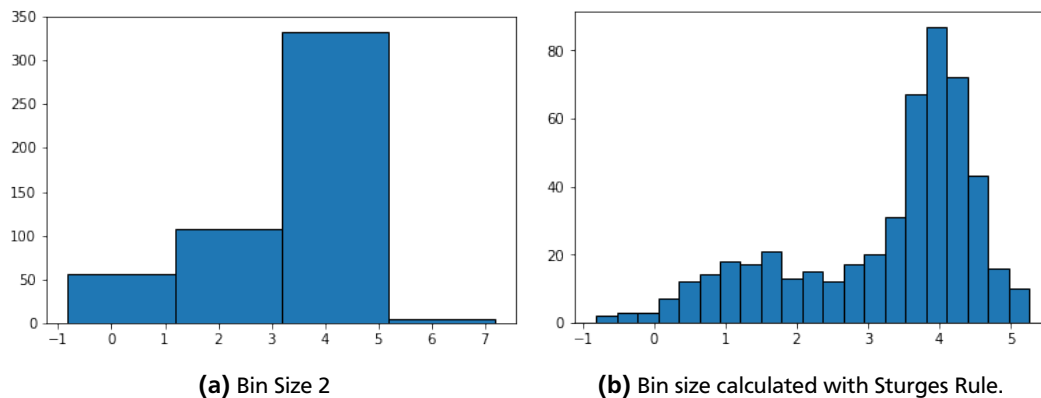
The third histogram 2a also schows a trend of the underlying function, but compared to figure 1b some of details get lost in the representation. Meaning that the bin size has been choosen to big (to much Bias).

Bonus: This last histogram 2b is an histogram with the number of bins determined by Sturges Rule. It's shape is similiar to figure (2), only beeing a bit more smooth. However Sturges rule is critized to smooth the Histogram to much (Hyndman, 1995) and should be only used as a rule of thumb. In this case it adds futher believe into a bin size of about 0.5

However in the end for these four bin sizes, we cant be completly sure that the one choose is the best, but we have atleast good evidence for it.



**Figure 1: First two Histograms**



**Figure 2: Last two Histograms**

Snippet of the code used:

```
import numpy as np
train = np.loadtxt("nonParamTrain.txt");

import matplotlib.pyplot as plt
binwidth = 0.02
bin_sequence = np.arange(min(train), max(train) + binwidth, binwidth)
plt.hist(train, bins=bin_sequence, edgecolor="black", linewidth=0.1);
```

### b) Kernel Density Estimate [6 Points]

Compute the probability density estimate using a Gaussian kernel with  $\sigma = 0.03$ ,  $\sigma = 0.2$  and  $\sigma = 0.8$  of the training data. Compute the log-likelihood of the data for each case, compare them and show which parameter performs the best. Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval  $x \in [-4, 8]$ , attach a snippet of your code and comment the results.

The log-likelihood for the first Gaussian kernel with  $\sigma = 0.03$  is:  $-674.7$ .

The log-likelihood for the second Gaussian kernel with  $\sigma = 0.2$  is:  $-717.0$ .

The log-likelihood for the third Gaussian kernel with  $\sigma = 0.08$  is:  $-795.7$ .

From comparing the log-likelihoods the first kernel should be the best one. But by further observing the graphs it seems that the second kernel is the best suited for the task.

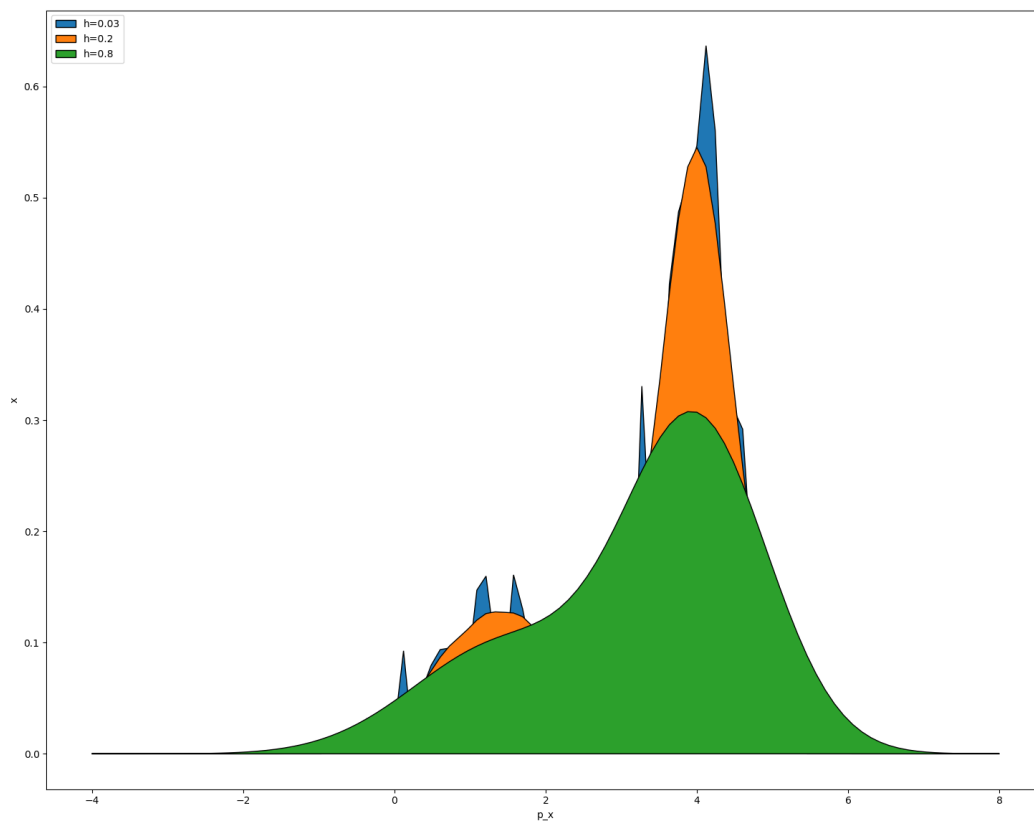


Figure 3: Kernel density estimation for  $\sigma = h = [0.03, 0.2, 0.8]$

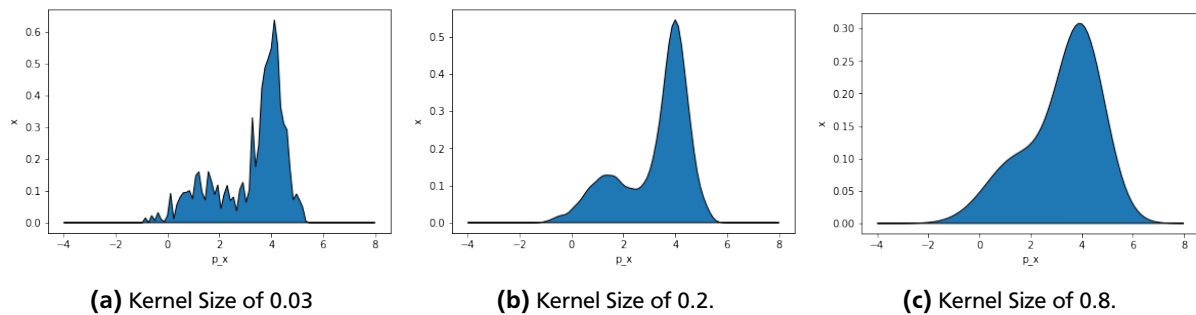


Figure 4: All the plots seperated

Snippet of the code used:

---

```
def gaussian(x,h):
    return np.exp(-x**2/(2*h**2))/(h*np.sqrt(2*np.pi))

N = train.size
X_plot = np.linspace(-4, 8, 100)
sum1 = np.zeros(len(X_plot))
sum2 = np.zeros(len(X_plot))
sum3 = np.zeros(len(X_plot))

for i in range(0, N):
    sum1 += ((gaussian(X_plot - train[i], h=0.03)) / N)

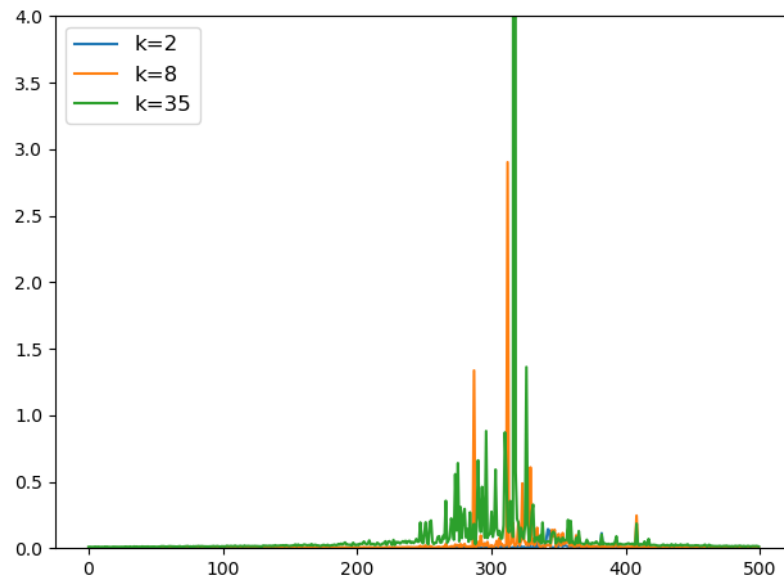
# Plot the result
plt.fill(X_plot, sum1, edgecolor="black", linewidth=1)
plt.xlabel("p_x")
plt.ylabel("x")

# Calculate the likelihood
likelihood_sum = 0
for i in range(0, N):
    likelihood = 0
    for j in range(0, N):
        likelihood += ((gaussian(train[i] -train[j], h=0.03)) / N)
    likelihood = math.log(likelihood)
    likelihood_sum += likelihood
print(likelihood_sum)
```

---

### c) K-Nearest Neighbors [6 Points]

Estimate the probability density with the K-nearest neighbors method with  $K = 2, K = 8, K = 35$ . Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval  $x \in [-4, 8]$ , attach a snippet of your code and comment the results.

Figure 5: K-Nearest Neighbors for  $k = 2, k = 8, k = 35$ 


---

```

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

train = np.loadtxt("nonParamTrain.txt");
k = 35
N = train.size
distances = np.zeros([N, N - 1])
pd = np.zeros(N)
x_axis = np.linspace(-4, 8, N)

# plot density
for i in range(0, N):
    for ii in range(0, N - 1, ):
        distances[i][ii] = abs(train[i] - train[ii])

    distances[i, :] = np.sort(distances[i, :]);
    pd[i] = distances[i][-k] # k nearest neighbor

result = k / (N * abs(x_axis - pd))
plt.plot(result)

# Compute log likelihood
distances = np.zeros([N, N - 1])
pd = np.zeros(N)
for i in range(0, N):
    for ii in range(0, N - 1, ):
        distances[i][ii] = abs(train[i] - train[ii])

    distances[i, :] = np.sort(distances[i, :]);
    pd[i] = distances[i][-k] # k nearest neighbor

likelihood = 0
for i in range(0, N):
    likelihood += math.log(k / (N * abs(train[i] - pd[i])))
print(likelihood)

```

---

d) **Comparison of the Non-Parametric Methods [4 Points]**

Estimate the log-likelihood of the testing data using the KDE estimators and the K-NN estimators. Why do we need to test them on a different data set? Compare the log-likelihoods of the estimators w.r.t. both the training and testing sets in a table. Which estimator would you choose?

*The following table lists all results for the log-likelihoods of the estimators w.r.t both the training and testing sets in a table. The First three rows are for the training set with  $\sigma = [0.03, 0.2, 0.8]$  and  $k = [2, 8, 35]$  and the next three for the test set using the same order of parameterization.*

*Actually from the log-likelihoods it seems that all estimators perform badly. However the Kernel Density Estimate using Gaussian kernel has much better results than the knn-Algorithm so its preferable.*

	KDE	KNN
1.	-675	-2675
2.	-717	-875
3.	-795	-1273
4.	-2812	-14631
5.	-2877	-11115
6.	3192	-4469

---

**Problem 2.4 Expectation Maximization [20 Points]**


---

In this exercise, you will use the datasets `gmm.txt`. It contains data from a Gaussian Mixture Model with four 2-dimensional Gaussian distributions.

a) **Gaussian Mixture Update Rules [2 Points]**

Define the model parameters and the update rules for your model. Specify the E- and M-steps of the algorithm.

b) **EM [18 Points]**

Implement the Expectation Maximization algorithm for Gaussian Mixture Models. Initialize your model uniformly. Generate plots at different iterations  $t_i \in [1, 3, 5, 10, 30]$ , showing the data and the mixture components, and plot the log-likelihood for every iteration  $t_i = 1 : 30$ . Attach a snippet of your code.