Statistical Machine Learning: Exercise 2

Maximum Likelihood Estimation, Parametric and Non-parametric Density Estimation, Expectation Maximization
Total Possible Points: 66

Prof. Marcus Rohrbach, Prof. Simone Schaub-Meyer

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2025

Publication Date: May 25th, 18:10 PM
Due Date: June 8th, 23:59 PM

Note: Many of the concepts required for solving this homework will be introduced in lectures 4, 5, and 6. If you don't know how to do some of these problems by the time this is released, hold on a bit more for that specific problem.

Programming tasks: 3c), 4b), 4c), 6

TEMPLATE: `https://colab.research.google.com/drive/1DfEty2lw1Wx3r7uDYu8qdzrQLQV2O3c-?usp=sharing`

Task 1: Maximum Likelihood Estimation (20 Points)

We assume that samples are drawn i.i.d. from the Gaussian distribution:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}.$$

1a) (7 Points)

Determine the maximum likelihood estimator $\hat{\mu}$ for the unknown mean μ of the Gaussian distribution. (Note: Do not forget to show or argue that the estimator is indeed a maximum and not a minimum or saddle point.)

Solution:

1b) (5 Points)

Determine the maximum likelihood estimator $\hat{\sigma}^2$ for the unknown variance σ^2 for the Gaussian distribution with unknown variance and known mean. You may reuse the derived log-likelihood function from a) and begin your derivation from there.

Solution:

1c) (3 Points)

Determine whether the variance estimator $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$ is unbiased in the case of an unknown mean μ. Assume you can estimate the mean μ with $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and check if:

$$E\left[\hat{\sigma}^2\right] = E\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right] = Var(X)$$

Hint: Use the following identity for x and for $\bar{x}$:

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

Solution:

1d) (5 Points)

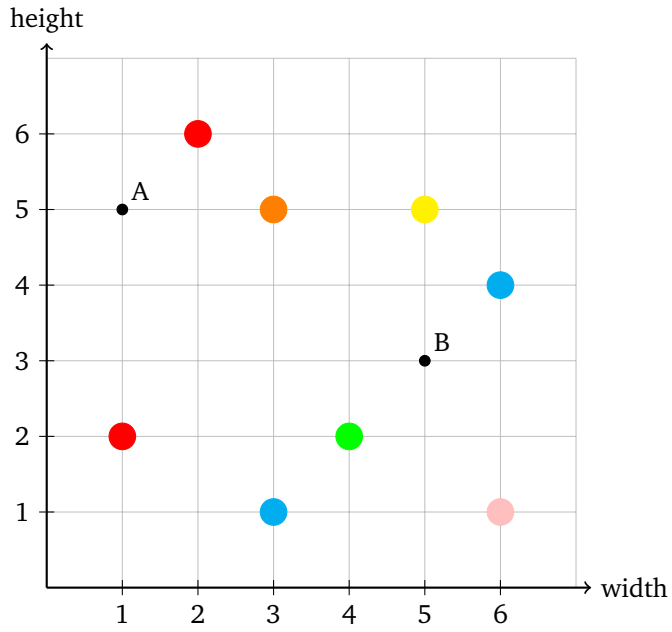
Given the following three samples: $x_1 = 38, x_2 = 40.5, x_3 = 41, x_4 = 42, x_5 = 46$.

Draw or plot the distribution using your derived estimators for the mean and variance and the provided samples. Explain what you calculated in the previous subtask and explain it using the graph you created. Then, think of a real-life scenario with the given samples that demonstrates the importance of the previously calculated maximum likelihood estimator for the unknown mean.

Solution:

Task 2: Non-Parametric Density Estimation (9 Points)

Given the following dataset graph



depicting the fruit colors red, orange, blue, green, pink and yellow with their corresponding width and height features. For example, the feature values of the pink fruit are (6, 1). In this task, we will apply the k-Nearest Neighbor (kNN) method to classify the color of fruit A and B shown in the graph, using a kernel to measure the similarity between data points. The Euclidean distance is defined as $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2$ and is used to define the distance in the feature space. For the entirety of this task, the hyperparameter will be $k = 3$ for the kNN method.

2a) (7 Points)

Assume that the kernel is Gaussian $k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\alpha d(\boldsymbol{x},\boldsymbol{y})^2}$, classify the color of fruit A with $\alpha = 0.2$.

List the nearest neighbors of A and B that are relevant for the classification, including their Euclidean distances and corresponding similarity scores (kernel values). Then, use the kernel values to determine the predicted class for each point.

Solution:

2b) (2 Points)

What problem arises if we use the kernel to weight the neighbors' contributions and set $\alpha = 20$ or higher? How does this affect the classification?

Solution:

Task 3: Bayesian Estimation (16 Points)

Given is a dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$. Each $x_i \in \mathbb{R}$ is drawn independently from a Gaussian distribution with **unknown mean** μ and **known variance** σ^2. That is,

$$x_i \sim \mathcal{N}(\mu, \sigma^2), \quad \text{for } i = 1, \ldots, n$$

We want to estimate the distribution of the mean $p(\mu \mid \mathcal{D})$ using Bayesian Estimation. We define the prior over μ by another Gaussian distribution:

$$\mu \sim \mathcal{N}(\mu_0, \tau^2)$$

3a) (2 Points)

Write the likelihood function $p(\mathcal{D} \mid \mu)$ and the prior $p(\mu)$.

Solution:

3b) (2 Points)

1) Under the Gaussian prior, what type of distribution does the posterior $p(\mu \mid \mathcal{D})$ assume?

2) How is this concept called in the lecture?

Solution:

3c) (12 Points)

Programming Task

Complete the corresponding section of the notebook as part of this task.

In this task, we are going to implement Bayesian estimation for a Gaussian with unknown mean and known variance in Python. The update rules for the mean and variance of our posterior look like this:

$$\mu_n = \frac{\sigma^2 \cdot \mu_0}{\sigma^2 + n \cdot \sigma_0^2} + \frac{\sigma_0^2 \cdot \sum_{i=1}^{n} x_i}{\sigma^2 + n \cdot \sigma_0^2} \tag{1}$$

$$\tag{2}$$

We substitute the sum $\sum_{i=1}^{n} x_i$ with $\sum_{i=1}^{n} x_i = n \cdot \frac{1}{n} \sum_{i=1}^{n} x_i = n \cdot \bar{x} = n \cdot \mu_{MLE}$

$$\mu_n = \frac{\sigma^2 \cdot \mu_0}{\sigma^2 + n \cdot \sigma_0^2} + \frac{\sigma_0^2 \cdot n \cdot \mu_{MLE}}{\sigma^2 + n \cdot \sigma_0^2} \tag{3}$$

$$\sigma_n^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2} \right)^{-1} \tag{4}$$

You are not allowed to import any additional functionality. Begin by loading the file `GaussianBayesianEstimation.npy`. Here are some rough guidelines for the implementation of this task:

- Load the file `GaussianBayesianEstimation.npy` which contains a numpy array
- Initialize the parameters for the Gaussian Likelihood with unknown mean μ and known variance $\sigma^2 = 4$

- Initialize the parameters for the Gaussian Prior with $\mu_0 = 0$ and variance $\sigma_0^2 = 9$. Since we have no concrete prior information about the dataset we initialize a weakly informative prior with high variance.
- Implement the update rules for the parameters of the posterior.
- Compute the parameter values for the posterior as well as the maximum likelihood estimator for the mean for $n \in \{1, 2, 5, 10, 100, 1000\}$, where n is the number of data points, and plot the prior, posterior as well as the maximum likelihood estimate of the mean for each value of n resulting in 6 plots.

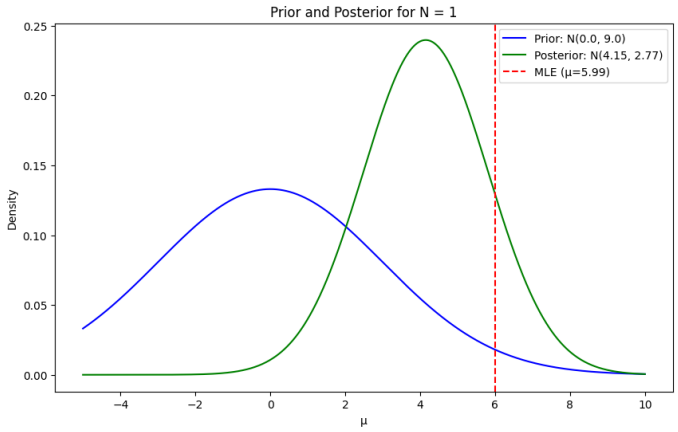


Figure 1: Resulting Plot for n=1

Now using the plots, describe the observed behavior. What are the main differences between maximum likelihood estimation and Bayesian estimation?

Solution:

Task 4: Expectation Maximization for Gaussian Mixture Models (10 Points)

Programming Task

Complete the corresponding section of the notebook as part of this task.

For the entirety of this task, no additional packages are allowed other than the ones already given.

In this task, we would like to implement the Expectation Maximization (EM) algorithm to fit a Gaussian Mixture Model (GMM) to a dataset. You have learnt from the lecture about GMM - modeling data using a sum of individual Gaussian distributions. However, the challenge is estimating the parameters of these Gaussians, as the MLE solution for this is more complex than the case of a single Gaussian. (Lecture 3, Slide 51).

Background about EM. The EM algorithm is a method for solving this problem. EM is an iterative method for finding maximum likelihood solutions for models with latent variables such as GMMs. In GMMs, the latent variables are the hidden mixture/cluster assignments. The algorithm has two main steps, which are repeated until convergence:

1. Expectation step (E-step): In this step, we compute the probabilistic assignment of data points to the probability distributions. In other words, we want to compute the **responsibilities**, i.e., the probability that each data point belongs to each Gaussian component using the *current parameters*.

$$a_{nj} = p(j|x_n) = \frac{\pi_j \mathcal{N}(x_n|\mu_j, \sigma_j)}{\sum_{i=1}^{J} \pi_i \mathcal{N}(x_n|\mu_i, \sigma_i)}.$$

2. Maximization step (M-step): Using the responsibilities computed from the previous E-step, we re-estimate the parameters (means, covariances, and mixing coefficients) of each Gaussian components.

$$\mu_j = \frac{\sum_{n=1}^{N} a_{nj} x_n}{\sum_{n=1}^{N} a_{nj}}, \quad \sigma_j^2 = \frac{\sum_{n=1}^{N} a_{nj}(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} a_{nj}}, \quad \pi_j = \frac{\sum_{n=1}^{N} a_{nj}}{N}.$$

An iteration of the EM algorithm consists of an E-step and a M-step. After each iteration, we evaluate the log likelihood with the new parameter estimates obtained from the last M-step and check for convergence. Below are some additional notes about the algorithm.

- x_n is a data point out of the dataset of N data points.
- In the expectation step, we want to compute the probability that each data point belongs to each Gaussian component. This means there will be $N * J$ a_{nj} values to compute, assuming the dataset contain N points and there are J Gaussian components. Furthermore, because we are working with posterior probabilities, keep in mind that $\sum_{j=1}^{J} a_{nj} = 1$ for each data point x_n.
- If you are unsure what mixing coefficient means, see Lecture 3, Slide 49.
- In the maximization step, the covariances are updated based on the **new means**. In other words, the new means have to be computed before the new covariances.
- You can check out Bishop, Chapter 9.2.2 if you still feel unsure about any details of the EM algorithm.
- You may have noticed that in order to start the first iteration of the algorithm, we need an initial estimate of the parameters for the Gaussian components. For this, we have provided the function `init_EM()`, which will be used in task 4b).

4a) EM Implementation (6 Points)

Implement the Expectation Maximization algorithm for Gaussian Mixture Models from scratch. More specifically, complete the following functions in the notebook:

- `expectation()` for the expectation step,
- `regularize()` for regularizing the covariance matrices in the maximization step,
- `maximization()` for the maximization step,
- `compute_log_likelihood()` for estimating the log likelihood.
- `em_gmm()`, which leverages the previously implemented functions.

For your implementation, note the following:

- To compute the multivariate normal probability density function you can use the function `multivariate_normal` from `scipy.stats`.
- For the regularization, implement the function so that a small value is added to the diagonal entries of the covariance matrix, i.e. $\boldsymbol{\Sigma}_{\text{reg}} = \boldsymbol{\Sigma} + \sigma_{\text{min}}\boldsymbol{I}$. Make sure to use this in your EM implementation.
- For the stopping criterion of `em_gmm()`, check if the log likelihood has converged, i.e., the absolute difference in subsequent values is strictly below a certain threshold ϵ.
- Aside from returning the mixture components and the responsibilities/posteriors, `em_gmm()` should also return all the computed log likelihood values.

Solution:

4b) (4 Points)

We now want to test the implemented algorithm on the provided dataset `clustering_dataset.npy`. Set the number of mixture components to 3 and ϵ to 0.001. Initialize the mixture components using the provided helper function `init_EM()`. Plot the log likelihood values at every iteration. After that, complete `plot_results()` to visualize the final clustering result along with the mixture components in a single plot.

Your visualization should show each data point's corresponding mixture component. To simplify the visualization, from the soft assignment result of EM-GMM, assign each data point to the component with the highest corresponding responsibility/posterior (hard assignment).

Hint: For the mixture components, think about how you can visualize them when you are working with 2D data points. Lecture 1b, Slide 23 can be helpful.

Solution:

Task 5: Fisher's Linear Discriminant (6 Points)

5a) (3 Points)

From the lecture slides, following optimzation problem is known

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2\right)^2$$

Show mathematically why the optimal solution for $\mathbf{w}^*$ grows without bound (i.e., the norm $\|\mathbf{w}\| \to \infty$) if no constraint is placed on $\mathbf{w}$. What does this imply about the nature of this optimization problem?

Solution:

5b) (3 Points)

In this task, at least one answer is true for each multiple-choice question. Each wrong answer results in a minus point. The minimum amount of points is 0. You may explain your choice, if you are unsure with your answer, but note that choosing the wrong answer with a sound explanation will only result in no minus point.

1. Which of the following is true for Fisher's Linear Discriminant in the two-class case?
   - □ Fisher's Linear Discriminant maximizes the distance between the means of the classes while minimizing the variance within each class.
   - □ Fisher's Linear Discriminant assumes equal class-conditional distributions with diagonal covariance.
   - □ Fisher's Linear Discriminant requires i.i.d. data in order to compute a meaningful linear classifier.
   - □ A solution $\mathbf{w}^*$ exists for non-linearly separable data under Fisher's criterion.
2. Suppose the within-class covariance matrix S_W is singular (not invertible). What is the implication for Fisher's Linear Discriminant?
   - □ The data is i.i.d.
   - □ The solution w cannot be computed.
   - □ S_B should be used instead.
   - □ The entries of S_W must be tweaked such that S_W becomes invertible.

Solution:

Task 6: LDA Implementation (5 Points)

Programming Task

Complete the corresponding section of the notebook as part of this task.

In this exercise, you will use the dataset ldaData.txt, containing 100 points. Each row corresponds to a point, where the first and second values are the features of a point respectively. The first 50 points belong to class C1, the remaining 50 belong to class C2.

6a) (5 Points)

Implement the Fisher linear discriminant in Python, i.e., plot the projection line of the data. You are allowed to use built-in functions for computing the mean, the covariance, eigenvalues and eigenvectors.

Solution: