SS 2025
© Prof. Georgia Koppe

Course Coordinator: Andre Herz
andre.herz@iwr.uni-heidelberg.de

Machine Learning Essentials

Exercise Sheet 02

Due: 19.05.2025 11:15

This sheet is about **probabilistic classification** and **logistic regression**. You'll begin by deriving and comparing decision boundaries for generative classifiers, namely **LDA and QDA**. Then, you'll implement and evaluate an LDA classifier for handwritten digit recognition. The next exercise contrasts different methods of **statistical inference** using a darts throwing example, to help you deepen your understanding about how parameters for probabilistic models can be inferred. The last exercise is about the logistic regression model and some of its properties.

Regulations

Please submit your solutions via Moodle in teams of **3** students. The coding tasks must be completed using the template `MLE25_sheet02.ipynb`. Each submission must include **exactly** one file:

- Upload a `.pdf` file containing both your Jupyter notebook and solutions to analytical exercises.

The Jupyter notebook can be exported to pdf by selecting **File** → **Download as** → **pdf** in JupyterLab. If this method does not work, you may print the notebook as a pdf instead. Your analytical solutions can be either scanned handwritten solutions or created using LaTeX.

Exercise 1: Decision Boundaries of Generative Classifiers

Consider a C-class classification problem. Taking on a **probabilistic view**, we assume that y is a discrete random variable that takes on values $k \in \{1, ..., C\}$ out of the C class labels. The goal is then to predict the class label k for any input feature $\boldsymbol{x}$ of the training data. The predicted label $\hat{y}$ is acquired according to a decision rule, given by the **discriminant function** $g_k(\boldsymbol{x})$:

$$\hat{y} = \arg\max_k g_k(\boldsymbol{x})$$

The decision rule partitions the input feature space into C disjoint regions belonging to each of the classes. The classification problem thus comes down to constructing g_k such

that the resulting partition achieves the lowest possible classification error. **Bayesian decision theory** states that the optimal[1] decision rule is given by the **maximum a posteriori (MAP)** decision rule:

$$\hat{y}_{\text{MAP}} = \arg\max_k p(y = k|\boldsymbol{x}) = \arg\max_k p(\boldsymbol{x}|y = k)p(y = k),$$

where $p(y = k)$ is the prior probability for class k. We want to apply a **generative model** to this problem that learns the **class-conditional densities** $p(\boldsymbol{x}|y = k)$ for all k. In this exercise, you'll derive the **decision boundaries** induced by applying the MAP decision rule for Gaussian class-conditional densities in a binary classification problem, and compare the forms obtained for **Linear Discriminant Analysis (LDA)** and **Quadratic Discriminant Analysis (QDA)**.

Tasks

1. In the following, you'll conduct **Gaussian discriminant analysis**, where it is assumed that the class-conditional densities of input features $\boldsymbol{x} \in \mathbb{R}^N$ are N-dimensional Gaussian distributions:

   $$p(\boldsymbol{x}|y = k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

   Applying the logarithm to the MAP decision rule gives a discriminant function

   $$g_k(\boldsymbol{x}) = \log p(\boldsymbol{x}|y = k) + \log \pi_k, \quad \text{where } \pi_k = p(y = k).$$

   (a) **QDA:** Assuming that $\boldsymbol{\Sigma}_A \neq \boldsymbol{\Sigma}_B$, show that the decision boundary between two classes A and B can be written as

   $$\boldsymbol{x}^\top \boldsymbol{\Lambda} \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{x} + b = 0,$$

   by explicitly deriving the expressions for the matrix $\boldsymbol{\Lambda}$, the vector $\boldsymbol{w}$, and the scalar b in terms of the class means $\boldsymbol{\mu}_A$, $\boldsymbol{\mu}_B$, the covariance matrices $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$, and the prior probabilities π_A and π_B. Briefly explain why the presence of the quadratic term indicates a non-linear decision boundary.

   (4 pts.)

   (b) **LDA:** Now assume that $\boldsymbol{\Sigma}_A = \boldsymbol{\Sigma}_B = \boldsymbol{\Sigma}$. Show that in this case, the decision boundary simplifies to a linear function

   $$\boldsymbol{w}^\top \boldsymbol{x} + b = 0.$$

   Explicitly derive and state the expressions for the weight vector $\boldsymbol{w}$ and the bias b in terms of $\boldsymbol{\mu}_A$, $\boldsymbol{\mu}_B$, $\boldsymbol{\Sigma}$, π_A and π_B.

   (3 pts.)

[1] It is optimal in the sense that it minimizes the expected risk (not the empirical risk as in discriminative models) for a 0-1 loss, see next sheet.

2. Now consider a specific two-class problem in $\mathbb{R}^2$ with the following parameters:

$$\boldsymbol{\mu}_A = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \quad \boldsymbol{\mu}_B = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$$\boldsymbol{\Sigma}_A = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_B = \begin{pmatrix} 1.5 & -0.2 \\ -0.2 & 1.5 \end{pmatrix},$$

and equal priors $\pi_A = \pi_B = 0.5$.

   (a) Using your results from part 1, state the decision boundaries (explicitly) for both QDA and LDA. For LDA, use the **pooled covariance matrix** $\boldsymbol{\Sigma} = \frac{1}{2}(\boldsymbol{\Sigma}_A + \boldsymbol{\Sigma}_B)$. Note that you don't have to compute the boundaries by hand here. You may use e.g. Python to do it.

(1 pts.)

   (b) Sketch or plot both decision boundaries in the same coordinate system.

(1 pts.)

   (c) Explain how the difference in covariance matrices of A and B affects the shape and complexity of the QDA boundary compared to the linear LDA boundary, and comment on when one method might be preferred over the other.

(1 pts.)

Exercise 2: Implementing LDA

In this exercise, your goal is to recognize handwritten digits, a classical ML application. To do so, you'll implement and apply an LDA classifier.

Tasks

1. The `digits` dataset consists of 1797 8x8 images with one digit per image. Load the dataset from sklearn, using `sklearn.datasets.load_digits`. Note that the data points are flattened feature vectors for each image, $\boldsymbol{x} \in \mathbb{R}^{64}$, where each pixel's brightness is encoded by an integer ranging from 0 to 16. Visualize an example image of the digit **3** from the data using the `imshow` function of `matplotlib.pyplot` (set the `interpolation` parameter such that each exact pixel value is visible).

(1 pts.)

2. To get a binary classification problem, we'll only work with digits "3" and "9". Filter the dataset such that only these two digits are left. Split this filtered dataset in a training and a test set (#train/#test = 3/2).

(1 pts.)

3. To facilitate data visualization, we would prefer to work with 2-dimensional features. Design and implement a feature transformation

   ```
   features_2d = features2d(x)
   ```

   that maps each data point $\boldsymbol{x}$ to a 2-dimensional feature vector $\boldsymbol{x'} \in \mathbb{R}^2$. The goal of this **dimensionality reduction** is to embed the features in a lower-dimensional space while loosing as little information as possible. Thus, you want to find 2 features that "summarize" each image as accurately as possible. For example, you may choose two pixels that appear most discriminative for distinguishing "3" from "9" (based on average images of each class), or form a linear or non-linear combination of several pixel values. Create a transformed dataset and provide a brief justification for the features you chose.

   (2 pts.)

4. Using the predefined PCA function (principal component analysis, you'll learn about it later in the lecture) again transform the data into 2D. Then, create two scatter plots, displaying your hand-crafted features and the features obtained using PCA. In both plots, color-code the data points according to their class labels. Briefly discuss the separation of classes achieved by your hand-crafted transformation versus the PCA-based transformation.

   (1 pts.)

5. Implement the function:

   ```
   mu, covmat, p = fit_lda(training_features, training_labels),
   ```

   for LDA training. Before fitting your models, filter out any "dead" pixels by removing features with variance less than 0.001 (use `numpy.var()`). For the sample means and the empirical covariance matrix, use

   $$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i:\, y_i = k} \boldsymbol{x}_i, \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{k \in \{-1,1\}} \sum_{i:\, y_i = k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^\top.$$

   Apply the function to your hand-crafted feature set (Task 3), the PCA representation and the full 64-pixel dataset.

   (2 pts.)

6. Now implement the LDA label prediction:

   ```
   predicted_labels = predict_lda(mu, covmat, p, test_features)
   ```

according to the decision rule:

$$\hat{y}_i = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i + b),$$

using

$$\boldsymbol{w} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_{-1}), \quad b = -\frac{1}{2}\Big(\widehat{\boldsymbol{\mu}}_1^\top \widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_{-1}^\top \widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_{-1}\Big) + \log\frac{N_{-1}}{N_1}.$$

Compute and report the training and test error rates for all 3 datasets.

(1 pts.)

7. Create a grid for the 2D feature space (using your hand-crafted features) and plot the LDA decision regions overlaid with:

   - A scatter plot of the training data,
   - The decision boundary,
   - **Bonus** Gaussian isocontours of both class distributions.

   (1 pts.)

8. Using all available 64-dimensional data for both digits (ignoring previous train/test splits, but filtering out the dead pixels), implement and perform 10-fold **cross validation** using your LDA implementation. Report the obtained estimate and its standard error. Then, compare the cross validation estimate of the test error to the one obtained for the single train/test split in task 6.

   (1 pts.)

Exercise 3: Statistical Darts

This exercise compares different **methods of statistical inference** used in ML (and statistical models in general) through a dart throwing experiment. To infer the location of the latent[2] aiming point of a dart thrower (i.e. the location they intend to hit), you'll apply:

- **MLE (maximum likelihood estimation)**,
- **MAP (maximum a posteriori) estimation**, and
- **Bayesian inference**.

[2]In statistics, unobserved quantities that capture underlying factors which influence the observed data are called **latent variables**.

In order to apply statistical inference, we first need to come up with a reasonable **statistical model** for our experiment. In our model, we describe the dart throws $\boldsymbol{x}_i \in \mathbb{R}^2$ as Cartesian coordinates on the dartboard, distributed according to a bivariate Gaussian distribution

$$\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{\text{true}}, \boldsymbol{\Sigma}_{\text{true}}) \quad \text{i.i.d. } \forall i,$$

with an **unknown mean** $\boldsymbol{\mu}_{\text{true}}$ (the hidden aim point) and **known covariance** $\boldsymbol{\Sigma}_{\text{true}}$ (corresponding to the player's precision).

Tasks

1. Complete the function

   `data = simulate_data(mu_true, Sigma_true, n_samples)`,

   that simulates N observed dart throws from the player, creating a dataset $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. As **ground truth** values, we'll choose

   $$\boldsymbol{\mu}_{\text{true}} = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^\top, \quad \boldsymbol{\Sigma}_{\text{true}} = \begin{bmatrix} 0.05 & 0.02 \\ 0.02 & 0.04 \end{bmatrix}.$$

   (1 pts.)

2. To perform MLE, complete the function

   `mu_mle = compute_mle(data)`,

   that returns the maximum likelihood estimator $\hat{\boldsymbol{\mu}}_{\text{MLE}}$ for the unknown parameter $\boldsymbol{\mu}$.

   (1 pts.)

3. In order to perform MAP and/or Bayesian inference, we need a prior distribution for $\boldsymbol{\mu}$. Assuming a standard normal prior around the bullseye:

   $$p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$

   complete the function

   `mu_post, Sigma_post = compute_posterior(data, prior, Sigma_true)`,

   that computes the parameters of the full posterior distribution for $\boldsymbol{\mu}$ given the data and the prior. Then, also complete the function

   `mu_map = compute_map(data, prior, Sigma_true)`,

that assigns the mode of the posterior to `mu_map`.
Hint: The likelihood[3] and prior are both Gaussian,

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\text{true}} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}_{\text{true}}), \quad p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0).$$

Since the Gaussian is a **conjugate prior** to itself, the posterior is also Gaussian with parameters

$$p(\boldsymbol{\mu} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \frac{\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\text{true}} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) p(\boldsymbol{\mu})}{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)} = \mathcal{N}\Big(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}\Big),$$

where

$$\boldsymbol{\Sigma}_{\text{post}} = \Big(\boldsymbol{\Sigma}_0^{-1} + N\, \boldsymbol{\Sigma}_{\text{true}}^{-1}\Big)^{-1}, \quad \text{and} \quad \boldsymbol{\mu}_{\text{post}} = \boldsymbol{\Sigma}_{\text{post}}\Big(\boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 + N\, \boldsymbol{\Sigma}_{\text{true}}^{-1}\, \hat{\boldsymbol{\mu}}_{\text{MLE}}\Big).$$

(1 pts.)

4. Now compute all 3 inferences using the data and execute the provided function

   ```
   visualize_inference(mu_true, mu_mle, mu_map, mu_post, Sigma_post, data)
   ```

   to create a visualization of the posterior and the point estimates given by MLE/MAP on the dart board. For this, assume we observed $N = 5$ throws (you may try different values of N to see what changes). Imagine you are the coach of the player that performed the throws. Which conclusions can you draw from the results? How would the extra information provided by the Bayesian posterior influence your coaching strategy compared to relying on the point estimates?

   (3 pts.)

5. Consider the influence of the prior on the Bayesian inference results. How does the inferred aiming point and its uncertainty change if we chose a different prior (e.g. more concentrated or less informative)? In your discussion, refer to the expressions for $\boldsymbol{\mu}_{\text{post}}$ and $\boldsymbol{\Sigma}_{\text{post}}$ and contrast them to the MLE result.

   (1 pts.)

6. State 2 conditions under which the MAP and the MLE estimates for $\boldsymbol{\mu}$ exactly coincide. Again use the expressions for $\boldsymbol{\mu}_{\text{post}}$ and $\boldsymbol{\Sigma}_{\text{post}}$ for your reasoning.

   (2 pts.)

7. Name 2 other real-world scenarios where it might be important to quantify the uncertainty in a model's estimate (and briefly explain why).

   (1 pts.)

[3] Gaussian distributions are closed under multiplication, thus the likelihood is Gaussian.

Exercise 4: Logistic Regression

In this exercise, you'll derive the logistic regression model and some of its important properties from "first principles". Assume that the class labels $y \in \{0, 1\}$ in a binary classification problem are distributed according to a **Bernoulli distribution** with success probability $p = p(y = 1|x)$:

$$y \sim \text{Ber}(p).$$

For distributions of the **exponential family** (like the Bernoulli distribution), one can define a **generalized linear model (GLM)**. The sense in which these models generalize ordinary linear models is that they model the conditional expectation $\mu = \mathbb{E}_{p(y|x)}[y]$ as a function of a linear predictor $\boldsymbol{w}^\top \boldsymbol{x}$ via a **link function** g:

$$g^{-1}(\boldsymbol{w}^\top \boldsymbol{x}) = \mu.$$

In logistic regression, the canonical link function is the **logit function** (also called **log odds**), defined as:

$$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right) = \boldsymbol{w}^\top \boldsymbol{x}.$$

Solving for μ gives the **sigmoid function**:

$$\mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})},$$

which models the probability p. The sigmoid function has some nice computational properties (see lecture) and ensures that the predicted probabilities lie in the interval $(0, 1)$. Another motivation for this particular form is that if one assumes Gaussian class-conditional densities with equal covariances (like in LDA), then the **Bayes-optimal** classifier is a posterior of the form of $p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$, which is exactly the logistic regression model.

Tasks

1. Derive the log-likelihood for an i.i.d. dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ under the Bernoulli model.

   (1 pts.)

2. Show that

   (a) the negative log-likelihood for logistic regression is convex. State briefly why convexity is an important property for the optimization algorithm (e.g. gradient descent).

   (5 pts.)

(b) maximizing the log-likelihood derived in Task 2.1 is equivalent to minimizing the **average binary cross-entropy loss** over the dataset $\mathcal{D}$:

$$\mathcal{L}_{BCE}(\boldsymbol{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\Big(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\Big) + (1-y_i)\log\Big(1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\Big)\right].$$

(1 pts.)

3. (a) Show that for a linearly separable data set, the MLE solution for the logistic regression model is obtained by finding a vector w whose decision boundary $\boldsymbol{w}^\top \boldsymbol{x} = 0$ separates the classes and then taking the magnitude of $\boldsymbol{w}$ infinity. In other words: show that logistic regression fails to converge under MLE for linearly separable data.
Hint: Consider a sequence of weight vectors $\boldsymbol{w}_k = k \cdot \boldsymbol{w}'$ where $\boldsymbol{w}'$ defines a separating hyperplane. What happens to the log-likelihood as $k \to \infty$?

(2 pts.)

(b) What can we do to mitigate this?

(1 pts.)