Machine Learning and Physics
Winter Semester 2025/26

Exercise Sheet 2

Due: 03.11.25 14h00

Christof Gehrig, Gerrit Gerhartz

General Regulations.

- Please hand in your solutions in groups of two (preferably from the same tutorial group).
- Your solutions to theoretical exercises can be either handwritten notes (scanned), or typeset using LaTeX. For scanned handwritten notes, please ensure they are legible and not blurry.
- For the practical exercises, always provide the (commented) code as well as the output, and don't forget to explain/interpret the latter. Please submit a single PDF that includes both the exported notebook and your solutions to the theoretical exercises.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group of two. Specify all names of your group in the submission.
- You can find all the data in the GitHub Repository.

0 Python Environment

We added a file `requirements.txt` to the GitHub repository. You can use it to build the python environment using your preferred package manager. We recommend uv (`https://docs.astral.sh/uv/`). To create a virtual environment and install dependencies, run:

```
uv venv
source .venv/bin/activate  # On Windows: .venv\Scripts\activate
uv pip install -r requirements.txt
```

You can also use `conda` or `venv` using the `requirements.txt` file.

1 Uniform Manifold Approximation and Projection

In this exercise, we will implement a simplified version of UMAP. The simplified version minimizes the energy

$$\mathcal{E}(\{\mathbf{z}_i\}) = \sum_{(i,j)\in E} \Phi_{\text{attr}}\left(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2\right) + \sum_{(i,j)\in R} \Phi_{\text{rep}}\left(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2\right),$$

where E is the set of edges in the symmetrized kNN graph, and R is a set of randomly sampled pairs of points. The attractive and repulsive potentials are given by

$$\Phi_{\text{attr}}(d^2) = \log(1 + d^2), \quad \Phi_{\text{rep}}(d^2) = c \cdot \frac{1}{1 + d^2}.$$

Use $k = 15$ for the kNN graph, $|R| = 5N$ for the number of random pairs (N is the number of data points), and $c = 10$ for the repulsion strength.

(a) Vibe-code this simplified version of UMAP. Try not to code a single line! You can use the GitHub Copilot agent inside your IDE (e.g., using the integration in VS Code) instead of asking a chatbot like ChatGPT. Let the agent implement the following steps:

- Load data
- Build symmetrized kNN graph
- Extract edges

- Do force directed layout from random initialization using gradient descent with a decreasing learning rate. Use analytical derivatives which are hard-coded, not automated differentiation. Resample the pairs in the repulsive term in each iteration.

Read "your" code and try and understand it line by line. Verify its mathematical correctness.

(6 pts)

(b) Apply UMAP to the jet data from the first sheet. Compare these results with those obtained using PCA. (2 pts)

2 Kernel Density Estimation

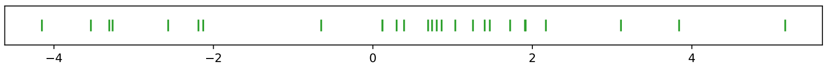

Figure 1: "Grass plot" of the one-dimensional data points.

You are given n points in one dimension, see Figure 1. We want to estimate the underlying density using kernel density estimation (KDE) with a radial basis function (RBF) kernel given by

$$k(\mathbf{x} - \mathbf{x}_n; w) = \frac{1}{w\sqrt{2\pi}} \exp\left(-\frac{(\mathbf{x} - \mathbf{x}_n)^2}{2w^2}\right).$$

The parameter w specifies the bandwidth.

(a) What does the density estimate look like if you choose an RBF kernel with a bandwidth that is very small / too large? Sketch the outcomes by hand (or compute them, if you prefer). For this particular data set, what kernel size bandwidth would you recommend? (3 pts)

(b) Suggest a scheme (words or pseudo code) to adjust the bandwidth locally. (3 pts)

(c) **Bonus:** Describe in words or pseudocode how an implementation of KDE using the fast fourier transform works. Does that work for high-dimensional data? Why? (3 bonus pts)

3 Linear regression: σ^2 Estimation and Heteroscedastic Noise

(a) Focusing on a single point $(y_n, \mathbf{x}_n)$, our linear regression model simplifies to

$$y_n = \boldsymbol{\beta}^T \mathbf{x}_n + \varepsilon_n.$$

If we assume that $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$, this is equivalent to the assumption that $y_n \sim \mathcal{N}(\boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2)$. The logarithm of $p(y_n \mid \boldsymbol{\beta}, \sigma^2)$ is known as the log-likelihood. Having observed N data points, this formulation generalizes to a sum of log-likelihoods and we can learn $\boldsymbol{\beta}$ by maximizing the logarithm of

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{n=1}^{N} \log \mathcal{N}\left(y_n \mid \boldsymbol{\beta}^T \mathbf{x}, \sigma^2\right).$$

Estimating σ^2 then analogously consists of finding the $\hat{\sigma}^2$ that maximizes this log-likelihood given the estimates $\hat{\boldsymbol{\beta}}$, i.e.,

$$\hat{\sigma}^2 = \arg\max_{\sigma^2} \sum_{n=1}^{N} \log \mathcal{N}\left(y_n \mid \hat{\boldsymbol{\beta}}^T \mathbf{x}, \sigma^2\right).$$

Solve this and relate the result to the SSQ (sum of squares) residual formulation from the lecture.

(3 pts)

(b) The standard formulation of linear regression is of homoscedastic noise, i.e. the variances of the observation noise is independent of $\mathbf{x}$. A generalization is to have a data point dependent variance on the observation noise, i.e., we have

$$y_n = \boldsymbol{\beta}^T \mathbf{x}_n + \varepsilon_n,$$

with $\mathbb{E}[\varepsilon_n] = 0$ and $\mathrm{var}[\varepsilon_n] = \sigma_n^2$, which is known as *heteroscedastic noise*. Give the sum-of-squares problem in that case and derive mean structure of the $\hat{\boldsymbol{\beta}}$. (3 pts)