Exercise 1

Deadline: 6.11.2024 16:00.
Ask questions in discord to #ask-your-tutor

In this exercise, you will implement and test classical generative modelling methods.

Regulations

Please implement your solutions in form of *Jupyter notebooks* (`*.ipynb` files), which can mix executable code, figures and text in a single file. They can be created, edited, and executed in the web browser, in the stand-alone app JupyterLab, or in the cloud via Google Colab and similar services.

Create a Jupyter notebook `generative-baselines.ipynb` for your solution and export the notebook to HTML as `generative-baselines.html`. Zip all files into a single archive `ex01.zip` and upload this file to MaMPF before the given deadline.
Moreover, please set your **Anzeigename/display name** and **Name in Uebungsgruppen/name in tutorials** in MaMPF to your real name, which should be identical to your name in `muesli` and make sure you **join the submission** of your team via the invitation code before the submission deadline. Check out https://mampf.blog/handing-in-homework-assignments for instructions. Also note that joining a submission takes a while when you do it for the first time (later it will be just a single click), so don't wait until the last minute before the deadline.

1 Two-dimensional data

Use the function `sklearn.datasets.make_moons()` to create 2-dimensional training data sets of varying sizes. Implement and train the following models (do not use pre-defined models and training algorithms from sklearn!) :

1. a two-dimensional histogram
2. a single Gaussian
3. a Gaussian mixture model (GMM)
4. a kernel density estimator (KDE) with squared exponential kernel

Implement the maximum mean discrepancy (MMD) metric with squared exponential and inverse multi-quadratic kernels for evaluation. Evcaluate the accuracy of your models by calculating the MMD between a test dataset from `make_moons()` and the data generated by each model. Visualize the accuracies as a function of model hyperparameters (histogram: bin size, GMM: number of components, KDE: kernel bandwidth) and training set size. Comment on your findings.

For a number of representative models (both good and bad ones), create two 2D plots that (i) visualize the numerical values of the learned density, and (ii) visualize a generated dataset from the model. Comment on model strengths and weaknesses. Bonus: Add some representation of the model solution to your plots (e.g. the grid of the histogram, some selected mixture components of the GMM).

2 Higher-dimensional data

Repeat the same tasks with the digits dataset (`sklearn.datasets.load_digits()`). Use the models and algorithms from sklearn this time. You may consider sklearn's KDtrees for speeding up computations in GMMs and KDE. Replace histograms (which do not scale to higher dimensions) with density forests, e.g. using the code from https://pypi.org/project/quantile-forest/ or https://github.com/kfritsch/density_forest.

Again, check model accuracy by MMD and visualize generated data for some representative models (you do not need to visualize the numerical density values – this is hard in 64 dimensions). In addition, train a `sklearn.ensemble.RandomForestClassifier` on the original dataset to distinguish the 10 digit classes. Use this classifier to check for the models which create recognisable output if the 10 digits are generated in equal proportions.