Exam project for PML 2023/2024

The exam project consists of parts A and B, reflecting different topics from the course. You are expected to work on the project in groups of 2-4 students. To avoid any misunderstandings, please read the following sections about requirements and practicalities carefully before you begin.

Technical requirements and practicalities

Report length The project should be documented in a brief report (one report per group), which will determine your final grade in the course. The report should consist of maximum 6 pages in a reasonable font-size, including figures. You are allowed to include additional text, results and figures as appendices (appendices do not count to your total page number).

Report style We recommend using this template for your report: https://da.overleaf.com/latex/templates/style-and-template-for-preprints-arxiv-bio-arxiv/fxsnsrzpnvwc.

Should I include code in my report? It might make sense to include short snippets - but in general we do not expect you to include a lot of code in the report. You can optionally include it in an appendix, but the most useful approach is to provide a link to a relevant github repository.

Contribution from group members If members of the group contributed unequally to the different parts of the project, you should state this in the report, by specifying for each part the contribution percentage from the different members. If nothing is stated, we will assume that all members contributed equally. *Please try to avoid distributing the sub-questions of the project among the group members* - ***ideally, each group members should be involved in all aspects of the project.***

Handing in The reports should be handed in as a PDF through Digital Exam (eksamen.ku.dk), which will become available over the next week.

Questions The next two weeks, the exercise sessions (Thursday 13:00-16:00) will be dedicated to the projects. We prefer that you save your questions for the in-class sessions, but if something urgent comes up, you are welcome to start a thread on the Absalon Discussion Board, and we'll try to answer it there. Please do not send us emails with questions about the project (since we then end up answering the same questions many times).

A Diffusion-based generative models

In the assignment for week 6 we coded a simple diffusion model on MNIST. In this first part of the project, you will explore different variations/extensions of this basic model.

A.1 Variations/extensions of the basic MNIST diffusion model

Below are some ideas for things you could explore (although feel free to choose other extensions if you want). For each variation/extension, implement it and compare its performance to the original model from the exercise session (a working solution to the exercise can be found on Absalon under the module for week 6 (`mnist_ddpm_solution.ipynb`)).

- In Ho et al, 2020[1], the authors state that we can design our neural network to predict μ, or the noise level ϵ. They also very briefly mention the possibility of predicting x_0 (which they say works less well). Implement and compare these approaches - which works best?

- Our estimates for the ELBO are quite noisy, because we approximate several expectations with a single-sample Monte Carlo estimate. Experiment with ways to reduce this variance.

  One strategy is to try out the approach mentioned in the "Low-discrepency sampler" section in appendix I.1 in the "Variational Diffusion Models" paper[2] (they describe it for the continuous case, but can you design a similar strategy for the case with discrete time steps?).

  Another strategy you could try is to use importance sampling to sample the time step t, as proposed in the "Improved denoising diffusion probabilistic models" paper[3]. Does reducing the variance make the model train faster?

- Implement a continuous-time version of the diffusion model, either by following the SDE approach by Yang Song et al[4], or the approach in the "Variational Diffusion Models" paper[5]. Does the continuous version provide any benefits on its own, in terms of training time or image quality?

  You can experiment with some of the unique features of the continuous formulation - e.g. different noising processes, or the ODE-based deterministic reverse process (which can also give you exact likelihoods - relevant for quantitative comparisons below).

[1] Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33, pp.6840-6851. https://arxiv.org/abs/2006.11239

[2] Kingma, D., Salimans, T., Poole, B. and Ho, J., 2021. Variational diffusion models. Advances in neural information processing systems, 34, pp.21696-21707. https://arxiv.org/abs/2107.00630

[3] Nichol, A.Q. and Dhariwal, P., 2021. Improved denoising diffusion probabilistic models. In International conference on machine learning, PMLR. https://arxiv.org/abs/2102.09672

[4] https://yang-song.net/blog/2021/score/

[5] Kingma, D., Salimans, T., Poole, B. and Ho, J., 2021. Variational diffusion models. Advances in neural information processing systems, 34, pp.21696-21707. https://arxiv.org/abs/2107.00630

- So far, we have considered only models generating samples from the entire distribution. How can we get diffusion models to generate samples *conditionally* - e.g. sampling only MNIST images corresponding to the digit 4?

  There are (at least) two ways of achieving this in diffusion models - either through "Classifier guided diffusion", or through "Classifier-free guidance". The two approaches are explained in blog posts by Lilian Weng[6] and Sander Dieleman[7]. Implement one or both of these techniques to make it possible to generate images of specific digits. If you implement both strategies, assess which ones performs best.

Please use the template code we provided (`ddpm_template.ipynb` or `mnist_ddpm_solution.ipynb`) as the basis for your implementations to ensure that the models are run under similar conditions (and to make it easy to compare the changes made). If you prefer, you are of course welcome to move the code from the Jupyter notebook to a standard python file.

A.2 Quantitative comparisons

In addition to visual comparisons, you should find some way to quantify the difference in performance between the different model variants. You can do this based on established metrics such as the FID or the Inception score, but also consider whether you could make a comparison based on likelihood. Are there any issues with evaluating likelihoods for diffusion models, and are there ways that these can be overcome? Do you observe that improvement in likelihood scores generally correlate with improvements in image quality?

Deliverables:

1. Implementation of one or more extensions/variations of the basic MNIST diffusion model. For each, briefly explain the theoretical background for the modification, and sketch what changes were necessary to make in the template code to make this work (you can do this in the appendix if you run out of space).

2. Compare the performance of the models to the basic model through visual inspection.

3. Quantify the performance of the models compared to the basic model. Include a discussion of using likelihood as a metric.

Note the A.1 question allows you to set your ambition level to match your group size. If your group has 2 members, you are not expected to do more than 1-2 variation/extension in A.1. On the other hand, if there are 4 members in your group, we expect you to explore 3-4 different variations/extensions (depending on the difficulty of each of them).

[6] https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
[7] https://sander.ai/2022/05/26/guidance.html

B Function fitting with Constraints

In some real world applications, learned models need to fulfill additional constraints. For example, in physics, simulations must obey energy conservation laws. When replacing the exact physics with learned models from data, we have to ensure that these laws hold, in order for the simulation not to diverge. In this exercise, we will consider integral constraints of the form

$$q = \int_0^1 f(x)\, dx \ ,$$

that is, the learned function f must integrate to q over the integral $[0, 1]$. Your task is to extend Gaussian Processes to integrate these integral constraints. As an example case, we will consider learning the function

$$g(x) = -\left(\sin(6\pi x)\right)^2 + 6x^2 - 5x^4 + \frac{3}{2}$$

on the interval $x \in [0, 1]$ using the integral constraint $q = 2$.

B.1 Fitting a standard GP

We will first consider the simpler problem of fitting a Gaussian Process without the additional constraint. This will form your baseline. You will first implement a standard GP model using the maximum a-posteriori estimate of hyper parameters and compare that to the sampled GP using NUTS.

As dataset, we assume that observations are given by:

$$y_i = g(x_i) + \epsilon,\ \epsilon \sim \mathcal{N}(0, 0.01) \ ,$$

where the observations are the grid $x_i = \frac{i-1}{\ell-1},\ i = 1 \ldots, \ell$ with $\ell = 30$. Partition the set of points randomly into a set of 20 points for training and 10 for evaluation. Follow the steps below:

1. Select a suitable model with your own choice of kernel. Identify the parameters of the model and decide which parameters are fixed and which are variable (you need ≥ 2 variable parameters). We will refer to the variable parameters as θ. For each parameter, pick a suitable prior distribution and implement the model (or use the GP implemented in Pyro) as well as a function implementing $\log p(y, \theta | X)$.

2. Compute the maximum a-posteriori estimate θ^* (e.g., using grid search or gradient descent) and evaluate the posterior log-likelihood of the test set on the fitted GP using θ^*.

3. Use NUTS to sample from the posterior. Check the quality of the MCMC sampling using diagnostics (Arviz). Use the diagnostics to choose the hyper-parameters of the sampling (such as the number of warmup samples).

4. Compute the approximate posterior likelihood of the samples in the test set using 500 sampled values of θ.

5. Repeat this procedure for 20 different generated datasets for both approaches and compare the obtained test likelihoods.

Deliverables:

- A mathematical description of the chosen probabilistic model $p(\theta, y|X)$ and a reasoning behind the chosen kernels and priors.
- An analysis of your obtained sample quality for the models
- State the equation of the approximate posterior likelihood you implemented.
- Compare the likelihoods of the model using either MAP or approximate posterior likelihood and report mean and standard deviation. Can you argue whether one of your models performed better than the other?

Hint: If it appears that Pyro ignores your setting of a kernel parameter, consider cleaning up old variables with `pyro.clear_param_store()`.

B.2 Learning with Integral Constraints

We will now extend the previous approach to include an integral constraint. Since exact integration is infeasible, we will instead approximate it using Trapezoidal rule that is computed on a grid of points X with elements $x_i = \frac{i-1}{\ell-1},\ i = 1\ldots,\ell$ as

$$q \approx \hat{q} = \sum_{i=1}^{\ell} w_i f(x_i), \quad w_i = \begin{cases} \frac{1}{2\ell-2}, & \text{for } i \in \{1, \ell\} \\ \frac{1}{\ell-1} & \text{otherwise} \end{cases}.$$

To model f, we will assume the GP prior $f \sim \mathcal{GP}(0, k(\cdot,\cdot))$, where k is the kernel with parameters from the MAP estimate from B.1.Follow the derivation steps:

1. Derive the probability distribution of the random variable $(\hat{q}, f)|X$ and show that it is normal distributed. Hint: Write the vector $(\hat{q}, f)$ as a linear transformation of f and use properties of the multivariate normal distribution. You can use the weight vector w to simplify the result.
2. Derive the probability distribution of the random variable $f|X, \hat{q}$. Assuming k is a universal kernel, does the covariance matrix of this distribution have full rank?
3. Plot five samples from $f|X, \hat{q}$ for $\ell = 101$ and different choices for $\hat{q} \in \{0, 5, 10\}$
4. Consider that you obtain a dataset $\mathcal{D}$ with three input-label pairs (x_i, y_i) following $y_i = g(x_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.01)$. The exact dataset is:

   $$\mathcal{D} = \{(0, 1.46), (0.25, 0.93), (0.5, 2.76)\}$$

   Compute the posteriors $f|\hat{q}, \mathcal{D}$ and $f|\mathcal{D}$ (i.e., the GP prediction with and without constraint after observing $\mathcal{D}$) and visualize the distribution. As parameters, use $\ell = 101$ and $\hat{q} = 2$.

Hints: derive your models using the rules for the normal distribution. The Pyro GP class is not easy to use here. For sampling, you can use the eigenvalue decomposition to obtain a decomposition $\Sigma = AA^T$ of the covariance matrix.

Deliverables:

- Derivations and resulting mean and covariance matrix for $(\hat{q}, f)|X$ and $f|X, \hat{q}$.
- Discussion on whether the covariance matrix of the distribution $f|X, \hat{q}$ has full rank.
- Three plots visualizing the samples from $f|X, \hat{q}$. Compare the different results and discuss their differences.
- Two plots visualizing the posteriors $f|\hat{q}, \mathcal{D}$ and $f|\mathcal{D}$. For this, overlay a plot of $g(x)$, a scatter plot of $\mathcal{D}$, and plots of $m(X)$ as well as $m(X) \pm 1.95\sqrt{v(X)}$, where m and v are the vectors of means and variances of the random variables at the locations x_i, $i = 1, \ldots, \ell$ of the GP posterior, respectively. Compare the plots and discuss your observations.