02477 Bayesian Machine Learning 2025: Assignment 3

This is the last assignment out of three in the Bayesian machine learning course 2025. The assignment is a group work of 3-5 students (please use the same groups as in assignment 1,2 if possible) and hand in via DTU Learn). The assignment is **mandatory**. The deadline is **4th of May 23:59**.

Part 1: Fully Bayesian inference for Gaussian process regression

In this part, we will extend our Gaussian process regression analysis of the bike sharing dataset (from week 5) to fully Bayesian inference on hyperparameter level. Use the code below (Fig. **??**) to load and preprocess the data .

```
# load data from disk
data = jnp.load('./data_exercise5b.npz')
X = data['day']
y = np.log(data['bike_count'])

# remove mean and scale to unit variance
ym, ys = jnp.mean(y), jnp.std(y)
y = (y-ym)/ys
```

Figure 1: Code for loading and preprocessing the bike sharing dataset.

The Gaussian process regression model for the dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ is given below:

$$y_n = f(x_n) + \epsilon_n, \tag{1}$$

where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and $f(x) \sim \mathcal{GP}(0, k(x, x'))$ for $k(x, x') = \kappa^2 \exp\left(-\frac{\|x-x'\|_2^2}{2\ell^2}\right)$.

We will impose the following prior distributions on the hyperparameters:

$$\begin{aligned}\kappa &\sim \mathcal{N}_+(0, 1)\\ \ell &\sim \mathcal{N}_+(0, v)\\ \sigma &\sim \mathcal{N}_+(0, 1),\end{aligned}$$

where $v > 0$ is a positive constant (which you will determine in the next task) and $\mathcal{N}_+(m, v)$ is the half-normal distribution. These assumptions lead to the following joint distribution

$$\begin{aligned}p(\boldsymbol{y}, \boldsymbol{f}, \sigma, \kappa, \ell) &= p(\boldsymbol{y}|\boldsymbol{f}, \sigma^2)p(\boldsymbol{f}|\kappa, \ell)p(\kappa)p(\ell)p(\sigma)\\ &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K})\mathcal{N}_+(\kappa|0, 1)\mathcal{N}_+(\ell|0, v)\mathcal{N}_+(\sigma|0, 1).\end{aligned}$$

In this exercise, we want to impose a prior that prevent the lengthscale from becoming too large.

Task 1.1: Choose a value for v such that the prior probability of observing a lengthscale larger than 100 is approximately 1%, i.e. $p(\ell > 100) \approx 0.01$.

Hints: You can do this in several ways, e.g. numerically or analytically

Task 1.2: Determine the marginalized distribution $p(\boldsymbol{y}, \sigma, \kappa, \ell)$.

The next goal is to approximate the posterior distribution over the hyperparameters, i.e. $p(\kappa, \ell, \sigma|\boldsymbol{y})$, using a Metropolis-sampler. Define $\theta = \{\kappa, \ell, \sigma^2\}$ to be the set of hyperparameters of the model. Since the scale of the hyperparameters are quite different, we will use an anisotropic proposal distribution:

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = \mathcal{N}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}, \boldsymbol{\Sigma}) \quad \text{for} \quad \boldsymbol{\Sigma} = \frac{1}{2}\begin{bmatrix}1 & 0 & 0\\ 0 & 100 & 0\\ 0 & 0 & 0.01\end{bmatrix}. \tag{2}$$

That is, the proposed step-size will generally be larger for the lengthscale dimension and so on.

Task 1.3: Implement a Metropolis sampler using the proposal distribution in eq. (2) for generating samples from the posterior $p(\kappa, \ell, \sigma|\boldsymbol{y})$. Run 4 chains for 10000 iterations each.

Hint: The half-normal distribution can be implemented as follows

```
from scipy.stats import norm
def log_halfnormal(x):
    return jnp.log(2) + norm.logpdf(x, 0, 1)
```

Hint: The parameters $\kappa, \ell, \sigma > 0$ are strictly positive parameters, and therefore, we need to be careful that our MCMC chain will jump to invalid parameter configuration. One way to achieve this is to implement the log joint function to return `-jnp.inf` whenever one or more hyperparameters are less than or equal to zero.

Task 1.4: Plot the trace for each parameter and report the convergence diagnostics $\hat{R}$ and S_{eff} for each parameter. Discard warm-up samples and report the number of samples discarded.

Task 1.5: Estimate and report the posterior mean for each hyperparameter. Report the MCSE for each estimate.

Task 1.6: Estimate a 95% posterior credibility interval for each hyperparameter.

Part 2: Regression modelling using mixture of experts

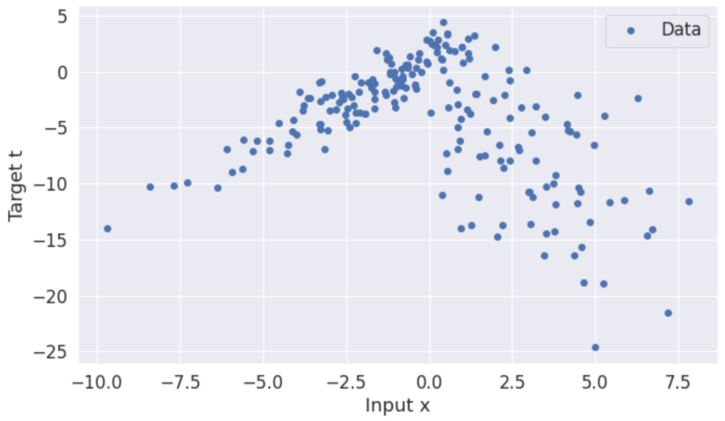

Figure 2: Dataset for regression

Consider the dataset for regression given in Figure 2. It is evident from the plot that there is a strong non-linear dependency between the target variables y_n and the input variables x_n. We also observe that the noise variance seems to depend on the input x. Consequently, assuming $y_n = y(x_n) + e_n$, where e_n is independent and identically distributed (i.i.d) noise would not be appropriate.

In this part, we will work with a so-called *Mixture of experts* (MoE) model for regression. The core idea is to model the dataset using several submodels, which are 'experts' in their own region of the input space. For example, the dataset in Figure 2 could be well-represented using two linear models: a linear model with positive slope and relative low noise variance for the 'left half' of the dataset and another linear model with negative slope and larger noise variance on the 'right half' of the dataset. Our goal is to simultaneously learn these two linear models as well as when to use which model.

To construct this model, we will consider two different linear models and then introduce a latent binary variable for each data point, $z_n \in \{0, 1\}$, to control which of the two linear models that should explain the data point y_n. That is, if $z_n = 0$ we assume y_n should be explained a linear model with parameters $\boldsymbol{w}_0$ and if $z_n = 1$, then y_n should be explained by a linear model with weights $\boldsymbol{w}_1$.

We model each z_n using Bernoulli distributions as follows

$$p(z_n|\pi_n) = \text{Ber}(z_n|\pi_n), \tag{3}$$

where $\pi_n = \sigma(\boldsymbol{v}^T\boldsymbol{x}_n)$ is the probability of $z_n = 1$, $\sigma(\cdot)$ is the logistic sigmoid function and $\boldsymbol{v}$ is a parameter vector to be estimated. That is, we basically model the latent z_n variable using a logistic regression model. We can set up a conditional likelihood as follows

$$p(y_n|x_n, z_n, \boldsymbol{w}_0, \boldsymbol{w}_1, \sigma_0^2, \sigma_1^2) = \begin{cases} \mathcal{N}(y_n|\boldsymbol{w}_1^T\boldsymbol{x}_n, \sigma_1^2) & \text{if} \quad z_n = 1, \\ \mathcal{N}(y_n|\boldsymbol{w}_0^T\boldsymbol{x}_n, \sigma_0^2) & \text{if} \quad z_n = 0, \end{cases} = \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T\boldsymbol{x}_n, \sigma_{z_n}^2) \tag{4}$$

where we also allow the noise variance to depend on z_n. We complete the model with generic priors

$$\tau, \sigma_0^2, \sigma_1^2 \sim \mathcal{N}_+(0, 1) \tag{5}$$
$$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \tau^2\boldsymbol{I}) \tag{6}$$
$$z_n|\boldsymbol{v} \sim \text{Ber}(\sigma(\boldsymbol{v}^T\boldsymbol{x}_n)) \tag{7}$$
$$y_n|z_n \sim \mathcal{N}(\boldsymbol{w}_{z_n}^T\boldsymbol{x}_n, \sigma_{z_n}^2), \tag{8}$$

where $\mathcal{N}_+(0,1)$ is the half-normal distribution. This leads to a joint model of the form

$$p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \boldsymbol{z}, \tau, \sigma_0, \sigma_1) = \left[\prod_{n=1}^{N} \mathcal{N}(y_n | \boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2) \text{Ber}(z_n | \sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\right] \mathcal{N}(\boldsymbol{w}_0 | \boldsymbol{0}, \tau^2 \boldsymbol{I})$$
$$\mathcal{N}(\boldsymbol{w}_1 | \boldsymbol{0}, \tau^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{v} | \boldsymbol{0}, \tau^2 \boldsymbol{I}) \mathcal{N}_+(\tau^2 | 0, 1) \mathcal{N}_+(\sigma_0 | 0, 1) \mathcal{N}_+(\sigma_1 | 0, 1). \tag{9}$$

The purpose is now to fit this model to the dataset in Figure 2 using MCMC. The dataset can be found on DTU Learn as loaded as follows:

```
data = jnp.load('./data_assignment3.npz')
x, y = data['x'], data['t']
```

To avoid handling the intercepts in the linear models explicitly, we will use the convention $\boldsymbol{x_n} = [x_n,\ 1]$.

Task 2.1: Marginalize out each z_n from to joint model in eq. (9) to obtain a joint distribution, where the likelihood for each observation is a mixture of two Gaussian distributions.

Hints: Use the sum rule to marginalize out z_n.

Task 2.2: Implement a Python function to evaluate the marginalized log joint distribution

Hints: Let θ denote all the parameters of the model, i.e. $\theta = \left\{\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau, \sigma_0^2, \sigma_1^2\right\}$, then implement a function that takes θ and returns $\ln p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \tau, \sigma_0, \sigma_1)$.

Task 2.3: Run a Metropolis-Hasting sampler to infer all parameters. Explain the settings you used (number of iterations, proposal distribution etc).

Hints: Feel free to use the Metropolis-Hasting code from the exercises. Plot the trace for all parameters to assess convergence (convergence diagnostics might be tricky due to the multimodality of the model). Compute the posterior mean of the weights $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}$, and plot the resulting function on top of the data as a sanity check. Make sure to initialize the sampler using a valid parameter vector, i.e. positive scale parameters $\sigma_0, \sigma_1, \tau > 0$.

Task 2.4: Report the posterior mean and 95% credibility intervals for all parameters.

Task 2.5: For $x \in [-12, 12]$, plot posterior predictive distribution for $p(\pi^* | \boldsymbol{y}, \boldsymbol{x}^*)$, $p(y^* | \boldsymbol{y}, \boldsymbol{x}^*, z^* = 0)$ and $p(y^* | \boldsymbol{y}, x^, z^* = 1)$ on top of the data.

Hints: For inspiration on how to plot these distributions using samples, see the exercise on Gibbs sampling for change point detection.

Task 2.6: For $x \in [-12, 12]$, plot posterior predictive distribution for $p(y^* | \boldsymbol{y}, \boldsymbol{x}^*)$