02477 Bayesian Machine Learning 2024: Assignment 3

This is the last assignment out of three in the Bayesian machine learning course 2024. 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 **5th 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. 1) to load and preprocess the data.

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

# remove mean and scale to unit variance
ym, ys = np.mean(y), np.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

Solution

For Gaussian, we know that approximately 68% of the probability mass fall within one standard deviation from the mean, approximately 95% within two standard deviations and approximately 99.7% within three deviations. Hence, we expect solution for v to be close to $100^2/3$. In fact,

$$\ell \sim \mathcal{N}_+(0, 100^2/3) \Rightarrow P(\ell > 100) \approx 0.003 \tag{2}$$

If we want more precision:

$$p(\ell > 100) = \int_0^x \mathcal{N}_+(\ell|0, v)\mathrm{d}\ell = 0.01 \tag{3}$$

which can be solved via the error function or via the by inverting the CDF of the standardized Gaussian Φ

$$x = \Phi^{-1}(0.995) \approx 2.58,$$

where we use 0.995 to account for the fact that the regular Gaussian is two-sided, whereas the half-Gaussian is one-sided. Hence,

$$v = \left(\frac{100}{2.58}\right)^2 \approx 1507.182 \tag{4}$$

End of solution

Task 1.2: Compute the distribution $p(\boldsymbol{y}, \sigma, \kappa, \ell)$ by marginalizing out $\boldsymbol{f}$

Solution

We have

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

We now recognize the integral $\int \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K})\mathrm{d}\boldsymbol{f}$ as the definition of the model evidence for a GP, and therefore, we know solution is $p(\mathbf{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \mathbf{K} + \sigma^2\boldsymbol{I})$, e.g. by eq. (18.74) in Murphy2. Hence,

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

End of solution

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 $\theta = \left\{\kappa, \ell, \sigma^2\right\}$ 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:

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}) = \mathcal{N}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{k-1}, \boldsymbol{\Sigma}) \quad \text{for} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}. \tag{5}$$

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. (5) for generating samples from the posterior $p(\kappa, \ell, \sigma|\boldsymbol{y})$. Run 4 chains for 10000 iterations each. Discard the warm-up samples and report the number of samples discarded.

The first step is the implement a function for evaluating the logarithm of the joint distribution form the previous task:

```
# prepare gp object
gp = GaussianProcessRegression(X_train, y_train, StationaryIsotropicKernel(squared_exponential))

# specify prior params
param_names = ['kappa', 'ell', 'sigma']
prior_std_devs = np.array([1, 100/2.58, 1])

# implement log target
def log_target(theta):
    if theta[0] < 0 or theta[1] <0  or theta[2] <0:
        return -np.Inf

    # prior contribution
    log_prior_kappa = np.log(2) + norm.logpdf(theta[0], 0, prior_std_devs[0])
    log_prior_ell = np.log(2) + norm.logpdf(theta[1], 0, prior_std_devs[1])
    log_prior_sigma = np.log(2) + norm.logpdf(theta[2], 0, prior_std_devs[2])

    # likelihood contribution
    log_lik = gp.log_marginal_likelihood(*theta)

    # sum and return
    return log_lik + log_prior_kappa + log_prior_ell + log_prior_sigma
```

and then we can set up a Metropolis sampler with the specified proposal distribution as follows

```
# generate initial values from the prior (shape: num_chains x params)
np.random.seed(123)
theta_init = np.abs(np.random.normal(0, prior_std_devs, size=(4, 3)))

# run sampler without excluding warmup
theta_samples, accept_rates = metropolis_multiple_chains(log_target, 3, 4, tau=
np.array([1, 10, 0.1]), num_iter=10000, theta_init=theta_init, warm_up=0)
```

Task 1.4: Plot the trace for each parameter and report the convergence diagnostics $\hat{R}$ and S_{eff} for each parameter.

Solution

We plot the traces for each chain for each parameter

```
fig, ax = plt.subplots(1, 3, figsize=(20, 5))
for i in range(3):
    ax[i].plot(theta_samples[:, :, i].T)
    ax[i].set(xlabel='Iterations', ylabel=param_names[i], title=f'Trace for {param_names[i]}')
```

Figure 4 shows the trace for the 4 chains of MCMC. By visual inspection, all chains appears to have (roughly) mixed after approximately 500 iterations, but to be conservative, we discard the first 2000 samples as warm up samples, but less would probably suffice.

End of solution

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

Solution

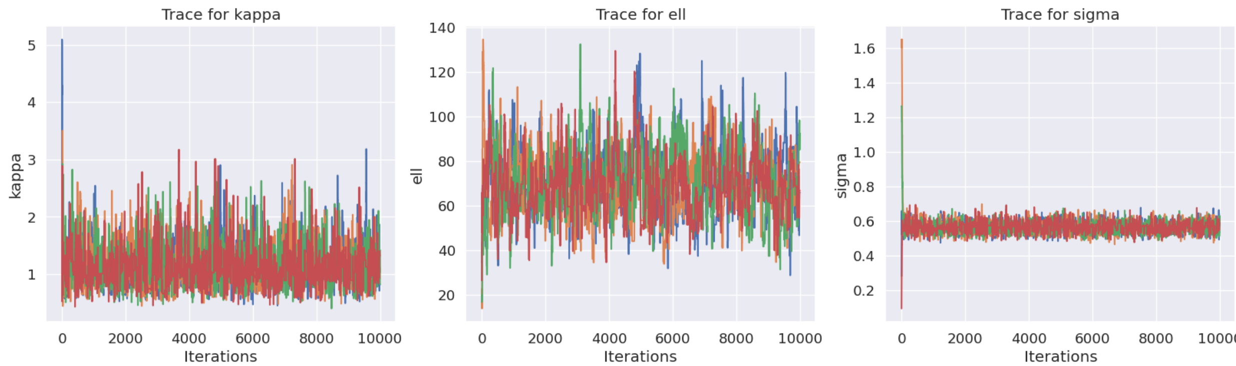

Figure 2: Trace of MCMC plot

First, we discard the warm-up samples and check the convergence diagnostics

```
# discard warmup
theta_samples = theta_samples[:, 2000:, :]

# compute diagnostics
Rhat = compute_Rhat(theta_samples)
S_eff = compute_effective_sample_size(theta_samples)

# print
for i in range(3):
    print(f'{param_names[i]:10s} Rhat={Rhat[i]:4.3f}, S_eff={S_eff[i]:4.3f}')
```

which yields

$$\hat{R}(\kappa) = 1.01 \qquad S_{\text{eff}}(\kappa) = 463$$
$$\hat{R}(\ell) = 1.04 \qquad S_{\text{eff}}(\ell) = 158$$
$$\hat{R}(\sigma) = 1.00 \qquad S_{\text{eff}}(\sigma) = 913$$

where we see that the $\hat{R}$-statistics is below 1.1 for all parameters. Then we merge the chains and evaluate the posterior mean and the MCSEs

```
merged_chains = theta_samples.reshape((-1, 3))
chain_mean = np.mean(merged_chains, axis=0)
chain_std = np.std(merged_chains, axis=0)

MCSE = 1/np.sqrt(S_eff) * chain_std

for i in range(3):
    print(f'{param_names[i]:10s}Posterior mean: {chain_mean[i]:4.2f} (MCSE = {MCSE[i]:4.3f})')
```

which yields

$$\mathbb{E}\left[\kappa|\boldsymbol{y}\right] \approx 1.12 \qquad \text{MCSE}(\kappa) \approx 0.02$$
$$\mathbb{E}\left[\ell|\boldsymbol{y}\right] \approx 69.80 \qquad \text{MCSE}(\ell) \approx 1.13$$
$$\mathbb{E}\left[\sigma|\boldsymbol{y}\right] \approx 0.56 \qquad \text{MCSE}(\sigma) \approx 0.00$$

End of solution

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

Solution

Using the posterior samples, we compute the requested interval using the `numpy.percentile`-function as follows

```
lower, upper = np.percentile(merged_chains, [2.5, 97.5], axis=0)
for i in range(3):
    print(f'{param_names[i]:10s}95% Interval: [{lower[i]:3.2f}, {upper[i]:3.2f}]')
```

which yields

$$
\begin{aligned}
&p(0.62 < \kappa < 1.93|\boldsymbol{y}) \approx 0.95 \\
&p(44.63 < \ell < 99.63|\boldsymbol{y}) \approx 0.95 \\
&p(0.51 < \sigma < 0.62|\boldsymbol{y}) \approx 0.95
\end{aligned}
$$

End of solution

Part 2: Variational inference, KL-divergences and entropy

Variational inference is a flexible tool approximating posterior distributions $p(\boldsymbol{w}|\boldsymbol{y})$, where $\boldsymbol{w} \in \mathbb{R}^D$ is a set of parameters to be estimated given some data $\boldsymbol{y}$. The central object in variational inference is the evidence lower bound (ELBO) given by

$$
\mathcal{L}[q] \equiv \mathbb{E}_q[\ln p(\boldsymbol{y}, \boldsymbol{w})] - \mathbb{E}_q[\ln q(\boldsymbol{w})]. \tag{6}
$$

The first term of the ELBO depends both on the model $p(\boldsymbol{w}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})$ and the variational approximation $q(\boldsymbol{w})$. The second term only depends on the variational approximation q and is equivalent to the *entropy* of $q(\boldsymbol{w})$, i.e. $\mathcal{H}[q] \equiv -\mathbb{E}_q[\ln q(\boldsymbol{w})]$. In this assignment, we will study the significance and interpretation of the entropy term in the context of mean-field Gaussian variational families

$$
q(\boldsymbol{w}) = \prod_{i=1}^{D} q(w_i) = \prod_{i=1}^{D} \mathcal{N}(w_i|\mu_i, \sigma_i^2). \tag{7}
$$

That is, all members in our variational family $\mathcal{Q}$ can be written as in eq. (7), where $\left\{\mu_i, \sigma_i^2\right\}_{i=1}^{D}$ are the *variational parameters.*

For many variational families, including mean-field Gaussians, the entropy $\mathcal{H}[q(\boldsymbol{w})]$ can be computed analytically, and it is the purpose of the next few tasks to derive this quantity analytically.

Task 2.1: Assuming a mean-field Gaussian variational family, show that the entropy of the variational approximation $q(\boldsymbol{w})$ is equal to the sum of the marginal entropies, i.e. $\mathcal{H}[q(\boldsymbol{w})] = \sum_{i=1}^{D} \mathcal{H}[q(w_i)]$.

Hints: Use the definition of entropy and insert the mean-field family from eq. (7). Use the fact that the distribution $q(\boldsymbol{w})$ factorizes to simplify the expression and obtain the desired result.

Solution We have

$$
\begin{aligned}
\mathcal{H}\left[q(\boldsymbol{w})\right] &= -\mathbb{E}_{q(\boldsymbol{w})}\left[\ln q(\boldsymbol{w})\right] && (8)\\
&= -\mathbb{E}_{\prod_{i=1}^{D} q(w_i)}\left[\ln \prod_{i=1}^{D} q(w_i)\right] && (9)\\
&= -\mathbb{E}_{\prod_{i=1}^{D} q(w_i)}\left[\sum_{i=1}^{D} \ln q(w_i)\right] && (10)\\
&= -\sum_{i=1}^{D} \mathbb{E}_{\prod_{i=1}^{D} q(w_i)}\left[\ln q(w_i)\right] && (11)\\
&= -\sum_{i=1}^{D} \mathbb{E}_{q(w_i)}\left[\ln q(w_i)\right] && (12)\\
&= \sum_{i=1}^{D} \mathcal{H}\left[q(w_i)\right] && (13)
\end{aligned}
$$

Task 2.2: Assuming a mean-field Gaussian variational family, derive the analytical expression for the entropy of the marginal distribution $q(w_i)$ is $\mathcal{H}\left[q(w_i)\right] = \frac{1}{2}\log(2\pi\sigma_i^2) + \frac{1}{2}$.

Hints: Marginal distributions are easy to obtain for mean-field families, i.e. the approximate marginal distribution of w_j is simply $q(w_j) = \mathcal{N}(w_j|\mu_j, \sigma_j^2)$.

Solution We have

$$
\begin{aligned}
\mathcal{H}\left[q(w_i)\right] &= -\mathbb{E}_{q(w_i)}\left[\ln q(w_i)\right] && \text{(Definition of entropy)}\\
&= -\mathbb{E}_{\mathcal{N}(w_i|\mu_i,\sigma_i^2)}\left[\ln \mathcal{N}(w_i|\mu_i,\sigma_i^2)\right] && \text{(Insert densities)}\\
&= -\mathbb{E}_{\mathcal{N}(w_i|\mu_i,\sigma_i^2)}\left[\ln \frac{1}{\sqrt{2\pi\sigma_i^2}}\left(-\frac{1}{2\sigma_i^2}(w_i-\mu_i)^2\right)\right] && (14)\\
&= -\mathbb{E}_{\mathcal{N}(w_i|\mu_i,\sigma_i^2)}\left[-\frac{1}{2}\ln(2\pi\sigma_i^2) - \frac{1}{2\sigma_i^2}(w_i-\mu_i)^2\right] && (15)\\
&= \frac{1}{2}\ln(2\pi\sigma_i^2) + \frac{1}{2\sigma_i^2}\mathbb{E}_{\mathcal{N}(w_i|\mu_i,\sigma_i^2)}\left[(w_i-\mu_i)^2\right] && \text{(Linearity of expectations)}\\
&= \frac{1}{2}\ln(2\pi\sigma_i^2) + \frac{\sigma_i^2}{2\sigma_i^2} && \text{(Definition of variance)}\\
&= \frac{1}{2}\ln(2\pi\sigma_i^2) + \frac{1}{2}. && (16)
\end{aligned}
$$

Task 2.3: Assuming a mean-field Gaussian variational family, show that $\mathcal{H}\left[q(\boldsymbol{w})\right] = \frac{1}{2}\sum_{i=1}^{D}\ln(2\pi e\sigma_i^2)$.

Hints: Combine the results from the previous two tasks. The constant e is the base of the natural logarithm.

Solution

$$\mathcal{H}\left[q(\boldsymbol{w})\right] = \sum_{i=1}^{D} \mathcal{H}\left[q(w_i)\right] \tag{17}$$

$$= \sum_{i=1}^{D} \left[\frac{1}{2}\ln(2\pi\sigma_i^2) + \frac{1}{2}\right] \tag{18}$$

$$= \sum_{i=1}^{D} \left[\frac{1}{2}\left(\ln(2\pi\sigma_i^2) + 1\right)\right] \tag{19}$$

$$= \frac{1}{2}\sum_{i=1}^{D} \left[\left(\ln(2\pi\sigma_i^2) + \ln(e)\right)\right] \tag{20}$$

$$= \frac{1}{2}\sum_{i=1}^{D} \ln(2\pi e\sigma_i^2) \tag{21}$$

This result implies that the ELBO objective for mean-field Gaussian families can be written as follows

$$\mathcal{L}\left[q\right] \equiv \mathbb{E}_q\left[\ln p(\boldsymbol{y}, \boldsymbol{w})\right] + \frac{1}{2}\sum_{i=1}^{D} \ln(2\pi e\sigma_i^2). \tag{22}$$

When fitting our variational approximation $q(\boldsymbol{w})$ by maximizing eq. (22), the two terms above will each 'favor' different solutions. Next, we will investigate which types of distributions $q(\boldsymbol{w})$ that are 'encouraged' or favored by the entropy term.

Task 2.4: Compute the partial derivative of the entropy term with respect to σ_j^2, i.e. $\frac{\partial}{\partial\sigma_j^2}\mathcal{H}\left[q(\boldsymbol{w})\right]$, and argue that the gradient is always strictly positive.

Solution The second term (the entropy) will encourage the approximation to have larger variances since the entropy increases when the variance of the approximation increases. We can also see this by computing the gradient of the entropy wrt. σ_j^2:

$$\frac{\partial}{\partial\sigma_j^2}\mathcal{H}\left[q(\boldsymbol{w})\right] = \frac{\partial}{\partial\sigma_j^2}\frac{1}{2}\sum_{i=1}^{D} \ln(2\pi e\sigma_i^2) \tag{23}$$

$$= \frac{\partial}{\partial\sigma_j^2}\frac{1}{2}\sum_{i=1}^{D} \ln(2\pi e\sigma_i^2) \tag{24}$$

$$= \frac{1}{2}\sum_{i=1}^{D} \frac{\partial}{\partial\sigma_j^2}\ln(2\pi e\sigma_i^2) \tag{25}$$

$$= \frac{1}{2}\frac{\partial}{\partial\sigma_j^2}\ln(2\pi e\sigma_j^2) \tag{26}$$

$$= \frac{1}{2}\frac{\partial}{\partial\sigma_j^2}\left[\ln(2\pi e) + \ln(\sigma_j^2)\right] \tag{27}$$

$$= \frac{1}{2}\left[\frac{\partial}{\partial\sigma_j^2}\ln(2\pi e) + \frac{\partial}{\partial\sigma_j^2}\ln(\sigma_j^2)\right] \tag{28}$$

$$= \frac{1}{2\sigma_j^2} \tag{29}$$

$$> 0 \tag*{(Since $\sigma_j^2 > 0$)}$$

Task 2.5: Which types of mean-field Gaussian distributions $q(\boldsymbol{w})$ are encouraged by the entropy-term?

Hint: When maximizing the ELBO using gradient-based methods, each of the two terms of the ELBO will contribute to the gradient. What would happen to q if the first term in the ELBO was not present?

Solution Since the partial is strictly positive for all variance parameters, it means that the gradient contribution from the entropy term will always push the distributions towards greater variances.

Next, we will rewrite the ELBO to enable a different interpretation of the objective. To do this, we will focus on a broad class of models for supervised learning

$$p(\boldsymbol{y}, \boldsymbol{w}) = p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{w})p(\boldsymbol{w}), \tag{30}$$

where $p(y_n|\boldsymbol{w})$ is the likelihood for the n'th data point and $p(\boldsymbol{w})$ is the prior distribution. This model family includes regression, classification, neural networks, Gaussian processes etc.

Task 2.6: Assume a probabilistic model of the form in eq. (30)**. Show that the ELBO can be written as in eq.** (31)**, where the KL-divergence is now between the approximation $q(\boldsymbol{w})$ and the prior $p(\boldsymbol{w})$.**

$$\mathcal{L}\left[q\right] = \sum_{n=1}^{N} \mathbb{E}_q\left[\ln p(y_n|\boldsymbol{w})\right] - \mathrm{KL}\left[q(\boldsymbol{w})||p(\boldsymbol{w})\right] \tag{31}$$

Solution

We have

$$\begin{aligned}
\mathcal{L}\left[q\right] &\equiv \mathbb{E}_q\left[\ln\left[\prod_{n=1}^{N} p(y_n|\boldsymbol{w})p(\boldsymbol{w})\right]\right] - \mathbb{E}_q\left[\ln q(\boldsymbol{w})\right] && (32)\\
&= \mathbb{E}_q\left[\left[\sum_{n=1}^{N} \ln p(y_n|\boldsymbol{w})\right] + \ln p(\boldsymbol{w})\right] - \mathbb{E}_q\left[\ln q(\boldsymbol{w})\right] && (33)\\
&= \sum_{n=1}^{N} \mathbb{E}_q\left[\ln p(y_n|\boldsymbol{w})\right] + \mathbb{E}_q\left[\ln p(\boldsymbol{w})\right] - \mathbb{E}_q\left[\ln q(\boldsymbol{w})\right] && (34)\\
&= \sum_{n=1}^{N} \mathbb{E}_q\left[\ln p(y_n|\boldsymbol{w})\right] + \mathbb{E}_q\left[\ln \frac{p(\boldsymbol{w})}{q(\boldsymbol{w})}\right], && (35)\\
&= \sum_{n=1}^{N} \mathbb{E}_q\left[\ln p(y_n|\boldsymbol{w})\right] - \mathbb{E}_q\left[\ln \frac{q(\boldsymbol{w})}{p(\boldsymbol{w})}\right], && (36)\\
&= \sum_{n=1}^{N} \mathbb{E}_q\left[\ln p(y_n|\boldsymbol{w})\right] - \mathrm{KL}\left[q(\boldsymbol{w})||p(\boldsymbol{w})\right] && (37)
\end{aligned}$$

Task 2.7: Which types of approximate distributions $q(\boldsymbol{w})$ are encouraged by the KL-term in eq. (31)**?**

Hint: When maximizing the ELBO using gradient-based methods, each of the two terms of the ELBO will contribute to the gradient. What would happen to q if the first term (the entire sum) in the ELBO was not present? When is the KL-divergence minimized?

Solution

The KL-divergence KL $[q(\boldsymbol{w}||p(\boldsymbol{w})]$ is minimized when $q(\boldsymbol{w}) = p(\boldsymbol{w})$, and therefore, the KL term will encourage the posterior to be similar to the prior.

Task 2.8: Which mean-field Gaussian distributions $q(w)$ are encouraged by the first term, i.e. the expected log likelihood? Argue why if you can.

Solution

Informally, the first term, i.e. $\sum_{n=1}^{N} \mathbb{E}_q \left[\ln p(y_n|\boldsymbol{w})\right]$ will encourage the model to fit the data as well as possible by pushing $q(\boldsymbol{w})$ towards $\mathcal{N}(\boldsymbol{w}_{ML}, \epsilon \boldsymbol{I})$, where $\epsilon \to 0$. That is, it will encourage $q(\boldsymbol{w})$ to "be" the maximum likelihood estimator.

There are several ways to make more formal arguments, which is beyond the scope of this course. But for the curios, we sketch a proof below in the following. We argue that 1) the expected log likelihood is always less than or equal to the maximum of the log likelihood function and 2) the expected log likelihood attains its maximum when $q(\boldsymbol{w})$ is a Dirac delta function centered on the maximum likelihood estimator, i.e. $q(\boldsymbol{w}) = \delta\left(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{ML}}\right)$. First, the log likelihood of the model is given by

$$L(\boldsymbol{w}) \equiv \sum_{n=1}^{N} \ln p(y_n|\boldsymbol{w}) \tag{38}$$

and the maximum likelihood estimator is defined

$$\boldsymbol{w}_{\mathrm{ML}} \equiv \arg\max_{\boldsymbol{w}} L(\boldsymbol{w}) = \arg\max_{\boldsymbol{w}} \sum_{n=1}^{N} \ln p(y_n|\boldsymbol{w}) \tag{39}$$

which implies

$$\sum_{n=1}^{N} \ln p(y_n|\boldsymbol{w}) = L(\boldsymbol{w}) \leq L(\boldsymbol{w}_{\mathrm{ML}}) \quad \Rightarrow \quad q(\boldsymbol{w})L(\boldsymbol{w}) \leq q(\boldsymbol{w})L(\boldsymbol{w}_{\mathrm{ML}}) \tag{40}$$

for all $\boldsymbol{w} \in \mathbb{R}^D$.

$$\begin{aligned}
\sum_{n=1}^{N} \mathbb{E}_q \left[\ln p(y_n|\boldsymbol{w})\right] &= \mathbb{E}_q \left[\sum_{n=1}^{N} \ln p(y_n|\boldsymbol{w})\right] && \text{(Linearity)} \\
&= \int q(\boldsymbol{w})L(\boldsymbol{w})\mathrm{d}\boldsymbol{w} && \text{(Def. of expectation)} \\
&\leq \int q(\boldsymbol{w})L(\boldsymbol{w}_{\mathrm{ML}})\mathrm{d}\boldsymbol{w} && \text{(Monotonicity of integrals)} \\
&= L(\boldsymbol{w}_{\mathrm{ML}}) \int q(\boldsymbol{w})\mathrm{d}\boldsymbol{w} && (41) \\
&= L(\boldsymbol{w}_{\mathrm{ML}}). && (42)
\end{aligned}$$

Thus, the expected log likelihood is bounded by the log likelihood function evaluated at the maximum likelihood estimator. Furthermore, the maximum value is attained in the limit, where q is centered on $\boldsymbol{w}_{\mathrm{ML}}$ and all posterior variances go to zero:

$$\lim_{\epsilon \to 0} \mathbb{E}_{\mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{\mathrm{ML}}, \epsilon \boldsymbol{I})} \left[L(\boldsymbol{w})\right] = \lim_{\epsilon \to 0} \int \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{\mathrm{ML}}, \epsilon \boldsymbol{I})L(\boldsymbol{w})\mathrm{d}\boldsymbol{w} = \int \delta(\boldsymbol{w} - \boldsymbol{w}_{\mathrm{ML}})L(\boldsymbol{w})\mathrm{d}\boldsymbol{w} = L(\boldsymbol{w}_{\mathrm{ML}}). \tag{43}$$

Thus, combining the answers for task 1.7 and 1.8 we see that we can interpret the distribution $q(\boldsymbol{w})$ that maximizes the ELBO as a compromise between fitting the data well and being similar to the prior.

End of solution

Task 2.9: Derive the analytical expression (i.e solve the integral) for the KL-divergence KL$[q_1||q_2]$ between univariate Gaussian distrbitions $q_1(w) = \mathcal{N}(w|\mu_1, \sigma_1^2)$ and $q_2(w) = \mathcal{N}(w|\mu_2, \sigma_2^2)$.

Solution

We have

$$\begin{aligned}
&\mathrm{KL}[q_1||q_2] = \mathrm{KL}[\mathcal{N}(w|\mu_1,\sigma_1^2)||\mathcal{N}(w|\mu_2,\sigma_2^2)] && (44)\\
&= \mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[\log\frac{\mathcal{N}(w|\mu_1,\sigma_1^2)}{\mathcal{N}(w|\mu_2,\sigma_2^2)}\right] && \text{(Def. of KL)}\\
&= \mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[\log\mathcal{N}(w|\mu_1,\sigma_1^2)\right] - \mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[\mathcal{N}(w|\mu_2,\sigma_2^2)\right] && \text{(Properties of log.)}\\
&= \mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[-\frac{1}{2}\log(2\pi\sigma_1^2) - \frac{1}{2\sigma_1^2}(w-\mu_1)^2\right] - \mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[-\frac{1}{2}\log(2\pi\sigma_2^2) - \frac{1}{2\sigma_2^2}(w-\mu_2)^2\right] && (45)\\
&= -\frac{1}{2}\log(2\pi\sigma_1^2) - \frac{1}{2\sigma_1^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_1)^2\right] + \frac{1}{2}\log(2\pi\sigma_2^2) + \frac{1}{2\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)^2\right] && (46)\\
&= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_1^2) - \frac{1}{2\sigma_1^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_1)^2\right] + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(\sigma_2^2) + \frac{1}{2\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)^2\right] && (47)\\
&= \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - \frac{1}{2\sigma_1^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_1)^2\right] + \frac{1}{2\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)^2\right] && (48)\\
& && (49)
\end{aligned}$$

We now recognize that $\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_1)^2\right]$ is the definition of the variance of q_1 such that

$$\begin{aligned}
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - \frac{1}{2\sigma_1^2}\sigma_1^2 + \frac{1}{2\sigma_1^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)^2\right] && (50)\\
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - \frac{1}{2} + \frac{1}{2\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)^2\right] && (51)\\
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[(w-\mu_2)\right]^2\right] && (52)
\end{aligned}$$

We can now expand the square and use linearity

$$\begin{aligned}
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[w^2 + \mu_2^2 + 2\mu_2 w\right]\right] && (53)\\
&= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}\left[\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[w^2\right] + \mu_2^2 + 2\mu_2\mathbb{E}_{\mathcal{N}(w|\mu_1,\sigma_1^2)}\left[w\right]\right]\right] && (54)\\
& && (55)
\end{aligned}$$

Finally, inserting the first and second moment of w yields

$$\begin{aligned}
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}\left[\mu_1^2 + \sigma_1^2 + \mu_2^2 - 2\mu_2\mu_1\right]\right] && (56)\\
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}\left[(\mu_1-\mu_2)^2 + \sigma_1^2\right]\right] && (57)\\
\mathrm{KL}[q_1||q_2] &= \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - 1 + \frac{1}{\sigma_2^2}(\mu_1-\mu_2)^2 + \frac{\sigma_1^2}{\sigma_2^2}\right] && (58)
\end{aligned}$$

End of solution

Part 3: Regression modelling using mixture of experts

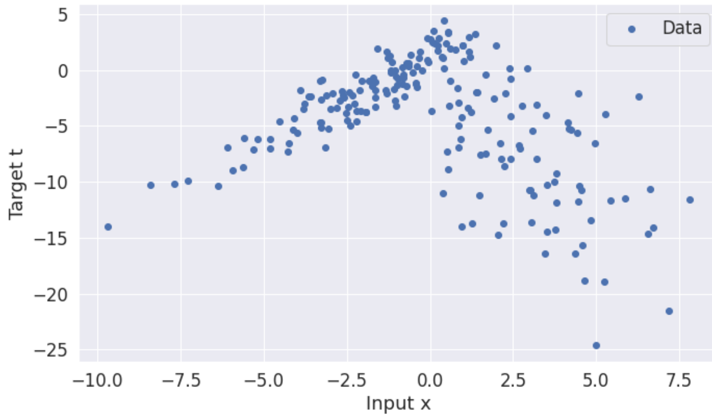

Figure 3: Dataset for regression

Consider the dataset for regression given in Figure 3. 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 3 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

$$p(z_n|\pi_n) = \text{Ber}(z_n|\pi_n), \tag{59}$$

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

$$p(y_n|x_n, z_n, \boldsymbol{w}_0, \boldsymbol{w}_1, \sigma_0^2, \sigma_1^2) = \begin{cases} \mathcal{N}(y_n|\boldsymbol{w}_1^T\boldsymbol{x}_n, \sigma_1^2) & \text{if} \quad z_n = 1, \\ \mathcal{N}(y_n|\boldsymbol{w}_0^T\boldsymbol{x}_n, \sigma_0^2) & \text{if} \quad z_n = 0, \end{cases} = \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T\boldsymbol{x}_n, \sigma_{z_n}^2) \tag{60}$$

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

$$\tau, \sigma_0^2, \sigma_1^2 \sim \mathcal{N}_+(0, 1) \tag{61}$$

$$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \tau^2\boldsymbol{I}) \tag{62}$$

$$z_n|\boldsymbol{v} \sim \text{Ber}(\sigma(\boldsymbol{v}^T\boldsymbol{x}_n)) \tag{63}$$

$$y_n|z_n \sim \mathcal{N}(\boldsymbol{w}_{z_n}^T\boldsymbol{x}_n, \sigma_{z_n}^2), \tag{64}$$

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

$$p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \boldsymbol{z}, \tau, \sigma_0, \sigma_1) = \left[\prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2)\text{Ber}(z_n|\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\right] \mathcal{N}(\boldsymbol{w}_0|\boldsymbol{0}, \tau^2 \boldsymbol{I})$$
$$\mathcal{N}(\boldsymbol{w}_1|\boldsymbol{0}, \tau^2 \boldsymbol{I})\mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \tau^2 \boldsymbol{I})\mathcal{N}_+(\tau^2|0,1)\mathcal{N}_+(\sigma_0|0,1)\mathcal{N}_+(\sigma_1|0,1). \tag{65}$$

The purpose is now to fit this model to the dataset in Figure 3 using MCMC. The dataset can be found on DTU Learn as loaded as follows:

```
data = np.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 3.1: Marginalize out each z_n from to joint model in eq. (65) 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.

Solution

Denoting $p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1) = \mathcal{N}(\boldsymbol{w}_0|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}(\boldsymbol{w}_1|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}_+(\tau^2|0,1)\mathcal{N}_+(\sigma_0|0,1)\mathcal{N}_+(\sigma_1|0,1)$ and using the sum rule, we get

$$\begin{aligned}
p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \boldsymbol{z}, \tau, \sigma_0, \sigma_1) &\overset{(a)}{=} \sum_{\boldsymbol{z}} p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \boldsymbol{z}, \tau, \sigma_0, \sigma_1) && (66)\\
&\overset{(b)}{=} \sum_{\boldsymbol{z}} \left[\prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2)\text{Ber}(z_n|\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\right] p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1) && (67)\\
&\overset{(c)}{=} \left[\sum_{\boldsymbol{z}} \left[\prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2)\text{Ber}(z_n|\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\right]\right] p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1) && (68)\\
&\overset{(d)}{=} \left[\prod_{n=1}^{N} \sum_{z_n} \mathcal{N}(y_n|\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2)\text{Ber}(z_n|\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\mathrm{d}z_n\right] p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1) && (69)\\
&\overset{(e)}{=} \left[\prod_{n=1}^{N} (1-\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\mathcal{N}(y_n|\boldsymbol{w}_0^T \boldsymbol{x}_n, \sigma_0^2) + \sigma(\boldsymbol{v}^T \boldsymbol{x}_n)\mathcal{N}(y_n|\boldsymbol{w}_1^T \boldsymbol{x}_n, \sigma_1^2)\right] p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1), && (70)
\end{aligned}$$

where in (a) we use the sum rule, in (b) we substitute in the distributions concerning $\boldsymbol{z}$, in (c) we use the fact that $p(\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{v}, \tau^2, \sigma_0, \sigma_1)$ is independent of $\boldsymbol{z}$, in (d) we use the fact that $\mathbb{E}[xy] = \mathbb{E}[x]\,\mathbb{E}[y]$ when x, y are independent and finally, in (e) we compute the expectations with respect to each z_n. Therefore,

$$\begin{aligned}
&p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \boldsymbol{z}, \tau, \sigma_0, \sigma_1) && (71)\\
&= \left[\prod_{n=1}^{N} (1-\sigma(\boldsymbol{v}^T \boldsymbol{x}_n))\mathcal{N}(y_n|\boldsymbol{w}_0^T \boldsymbol{x}_n, \sigma_0^2) + \sigma(\boldsymbol{v}^T \boldsymbol{x}_n)\mathcal{N}(y_n|\boldsymbol{w}_1^T \boldsymbol{x}_n, \sigma_1^2)\right] && (72)\\
&\mathcal{N}(\boldsymbol{w}_0|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}(\boldsymbol{w}_1|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}(\boldsymbol{v}|\boldsymbol{0}, \tau^2\boldsymbol{I})\mathcal{N}_+(\tau^2|0,1)\mathcal{N}_+(\sigma_0|0,1)\mathcal{N}_+(\sigma_1|0,1) && (73)
\end{aligned}$$

Task 3.2: 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 $\ln p(\boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_0, \boldsymbol{v}, \tau, \sigma_0, \sigma_1)$. The half-normal distribution can be implemented as follows

```
from scipy.stats import norm
def log_halfnormal(x):
    if x < 0: # negative values not supported
        return -np.Inf
    else:
        return np.log(2) + norm.logpdf(x, 0, 1)
```

Solution There are many ways to implement the log joint, e.g.

```
from scipy.stats import norm

sigmoid = lambda x: 1./(1 + np.exp(-x))

def halfnormal(x):
    if x < 0:
        return 0
    else:
        return 2*norm.pdf(x, 0, 1)

def log_halfnormal(x):
    if x < 0:
        return -np.Inf
    else:
        return np.log(2) + norm.logpdf(x, 0, 1)

def unpack(params):
    w1 = params[:2]
    w2 = params[2:4]
    v = params[4:6]
    tau = params[6]
    sigma1 = params[7]
    sigma2 = params[8]
    return w1, w2, v, tau, sigma1, sigma2

def log_likelihood(X, y, params):
    w1, w2, v, tau, sigma1, sigma2 = unpack(params)

    g = X@v
    f1 = X@w1
    f2 = X@w2

    pi = sigmoid(g)

    if tau > 0 and sigma1 > 0 and sigma2 > 0:
        # could be implemented numerically more robust via logsumexp
        return np.log((1-pi)*norm.pdf(y, f1, sigma1) + pi*norm.pdf(y, f2, sigma2)).sum()
    else:
        return -np.Inf

def log_prior(params):
    w1, w2, v, tau, sigma1, sigma2 = unpack(params)
    log_p = log_halfnormal(tau) + log_halfnormal(sigma2) + log_halfnormal(sigma1)
    log_p += norm.logpdf(w1, 0, tau).sum() + norm.logpdf(w2, 0, tau).sum() + norm.logpdf(v, 0,
```

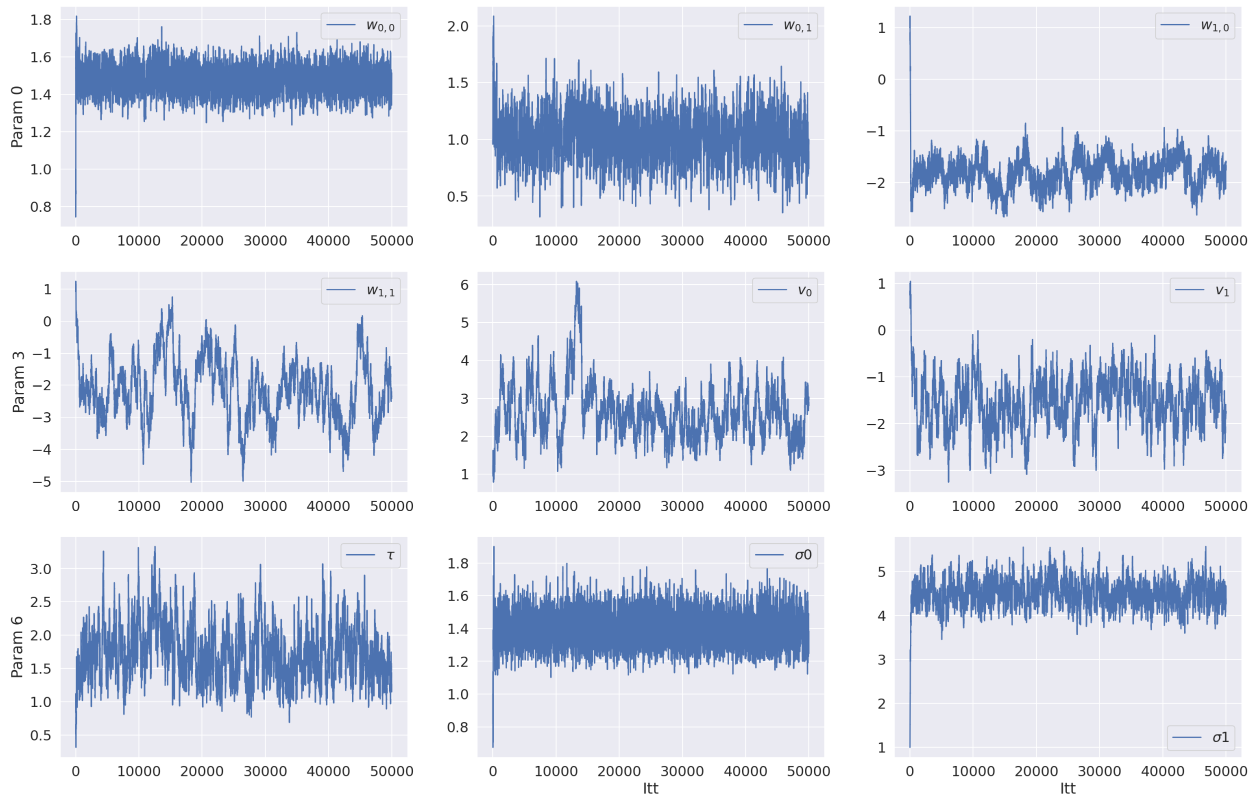

Figure 4: Trace of MCMC run for all 9 parameters.

```
    return log_p

log_joint_data = lambda X, y, params: log_likelihood(X, y, params)  + log_prior(params)
log_joint = lambda p: log_joint_data(X, y, p)
```

Task 3.3: 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* $\sigma_0, \sigma_1, \tau > 0$.

Solution

Figure 4 plots the trace of all 9 parameters of the model obtained using 50000 iterations with a Gaussian proposal distribution with variance 10^{-1}.

Task 3.4: Report the posterior mean and 95% credibility intervals for all parameters.

Solution

We can estimate means and intervals as follows

```
params_samples_after_warmup = params_samples[25000:, :]
for i in range(num_params):
    # compute mean
    m = np.mean(params_samples_after_warmup[:, i])
    # intervals
    lower, upper = np.percentile(params_samples_after_warmup[:, i], [2.5, 97.5])
    # print
    print(f'{param_names[i]:15s} Mean = {m:+4.3f}\tInterval = [{lower:+4.3f}, {upper:+4.3f}]')
```

which yields

```
w_{0,0}         Mean = +1.471     Interval = [+1.342, +1.601]
w_{0,1}         Mean = +0.984     Interval = [+0.613, +1.369]
w_{1,0}         Mean = -1.735     Interval = [-2.289, -1.251]
w_{1,1}         Mean = -2.502     Interval = [-4.518, -0.536]
v_{0}           Mean = +2.379     Interval = [+1.590, +3.138]
v_{1}           Mean = -1.547     Interval = [-2.735, -0.725]
\tau            Mean = +1.747     Interval = [+1.003, +2.719]
\sigma0         Mean = +1.399     Interval = [+1.237, +1.589]
\sigma1         Mean = +4.500     Interval = [+3.904, +5.150]
```

Task 3.5: 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.

Solution

Given the set of posterior samples, we can first compute the posterior samples of $\pi^*_{(i)}$, $y^*|z^* = 0$, $y^*|z^* = 1$, where (i) indicated the i'th sample:

$$\pi^*_{(i)} = \sigma(\boldsymbol{v}_{(i)}^T \boldsymbol{x}^*) \tag{74}$$

$$y^*_{(i)}|z^* = 0 \sim \mathcal{N}\left(\left(\boldsymbol{w}_0^{(i)}\right)^T \boldsymbol{x}^*, \left(\sigma_0^{(i)}\right)^2\right) \tag{75}$$

$$y^*_{(i)}|z^* = 1 \sim \mathcal{N}\left(\left(\boldsymbol{w}_1^{(i)}\right)^T \boldsymbol{x}^*, \left(\sigma_1^{(i)}\right)^2\right) \tag{76}$$

$$\tag{77}$$

and then compute the means and intervals using the `numpy.percentile`-function.

```
# extract samples for plotting p(pi^*|y, x^*) using pi^* = sigma(v^T x^*)
w0_samples = params_samples_after_warmup[:, 0:2].T
w1_samples = params_samples_after_warmup[:, 2:4].T
v_samples = params_samples_after_warmup[:, 4:6].T
sigma0_samples = params_samples_after_warmup[:, 7]
sigma1_samples = params_samples_after_warmup[:, 8]

# prep inputs
xp = np.linspace(-12, 12, 1000)
Xp = np.column_stack((xp, np.ones(len(xp))))

# compute p(pi^*|y, x^*)
pi_star = sigmoid(Xp@v_samples)
```

```
pi_star_lower, pi_star_upper = np.percentile(pi_star, [2.5, 97.5], axis=1)
pi_star_mean = np.mean(pi_star, axis=1)

# compute p(y^*|y, x^*, z^*=0)
y_star_z0 = Xp@w0_samples + np.random.normal(0, sigma0_samples)
y_star_z0_lower, y_star_z0_upper = np.percentile(y_star_z0, [2.5, 97.5], axis=1)
y_star_z0_mean = np.mean(y_star_z0, axis=1)

# compute p(y^*|y, x^*, z^*=1)
y_star_z1 = Xp@w1_samples + np.random.normal(0, sigma1_samples)
y_star_z1_lower, y_star_z1_upper = np.percentile(y_star_z1, [2.5, 97.5], axis=1)
y_star_z1_mean = np.mean(y_star_z1, axis=1)

# plot
fig, ax = plt.subplots(1, 2, figsize=(25, 10))
ax[0].plot(xp, pi_star_mean, 'b-', label='$p(\pi^*|\mathbf{y}, x^*)$')
ax[0].fill_between(xp, pi_star_lower, pi_star_upper, color='b', alpha=0.3)
ax[1].plot(xp, y_star_z0_mean, 'g-', label='$p(y^*|\mathbf{y}, x^*, z^*=0)$')
ax[1].fill_between(xp, y_star_z0_lower, y_star_z0_upper , color='g', alpha=0.3)
ax[1].plot(xp, y_star_z1_mean, 'r-', label='$p(y^*|\mathbf{y}, x^*, z^*=1)$')
ax[1].fill_between(xp, y_star_z1_lower, y_star_z1_upper , color='r', alpha=0.3)


for i in range(2):
    ax[i].legend()
    ax[i].set(xlabel='x')
ax[0].plot(x, 0.7 + 0.05*y, 'o', alpha=0.5, label='Shifted and scale data')
ax[1].plot(x, y, 'o', alpha=0.5, label='data')
ax[0].legend()
ax[1].legend()
```

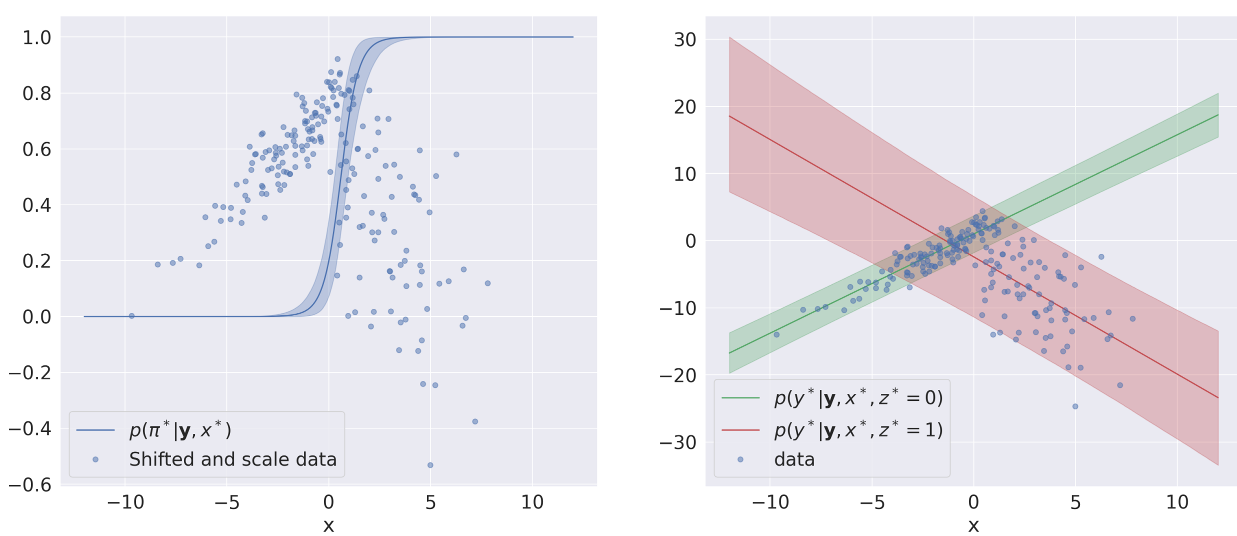

Figure 5: Plots for task 2.5

From the plot in Figure 5, we can see that the "change point" of the data roughly matches the position

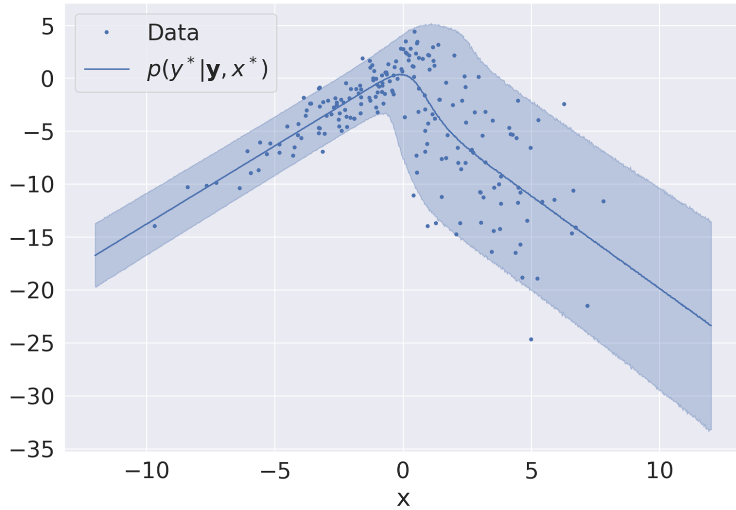

Figure 6: Plot for task 2.6

where $\pi^* = 0.5$ as expected (note that the data has been shifted and scale in the left panel to make this more clear). We can also that two conditional linear models appear to fit the left and right most part of the data nicely.

Task 3.6: For $x \in [-12, 12]$, plot posterior predictive distribution for $p(y^*|\boldsymbol{y}, \boldsymbol{x}^*)$

Solution

To compute the posterior predictive distribution, we repeat the same procedure as above, except that we now sample z^* as well.

```
# sample y^* based on posterior samples
def predictive_sample(params):
    w1_, w2_, v_, tau_, sigma1_, sigma2_ = unpack(params)

    pi = sigmoid(Xp@v_)
    z = np.random.binomial(1, pi)
    y1 = np.random.normal(Xp@w1_, sigma1_)
    y2 = np.random.normal(Xp@w2_, sigma2_)
    return (1-z)*y1 + z*y2

# collect statistics
y_samples = np.vstack([predictive_sample(p) for p in params_samples_after_warmup])
y_mean = y_samples.mean(0)
y_lower, y_upper = np.percentile(y_samples, [2.5, 97.5], axis=0)
# plot
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
ax.plot(x, y, '.', label='Data')
ax.plot(xp, y_mean, 'b-', label='$p(y^*|\mathbf{y}, x^*)$')
ax.fill_between(xp, y_lower, y_upper, alpha=0.3, color='b')
ax.set(xlabel='x')
ax.legend()
```

Task 3.7: Suppose we were analyzing another dataset requiring three experts rather two. Describe how you would adapt the model and write the corresponding probabilistic model in the same format as in eq. (61)–(64)**.**

Hints: A softmax function is required somewhere.

Solution

If the dataset required three experts, we would first need three linear models parametrized by $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$, and three noise variances $\sigma_0^2, \sigma_1^2, \sigma_2^2$. We would also need to allow the variables z_n to takes values $z_n \in \{0, 1, 2\}$. The 'natural' extension for the prior on z_n would therefore be a categorial distribution on z_n parametrized by a softmax-function (similar to the multi-class classification model in exercise 6). To use the softmax-construction, we would also need to introduce $\boldsymbol{V} = [\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2] \in \mathbb{R}^{3 \times D}$, i.e. one for each expert. Therefore, one solution would be

$$\tau, \sigma_0^2, \sigma_1^2, \sigma_2^2 \sim \mathcal{N}_+(0, 1) \tag{78}$$

$$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3 \sim \mathcal{N}(\boldsymbol{0}, \tau^2 \boldsymbol{I}) \tag{79}$$

$$z_n | \boldsymbol{V}, \boldsymbol{x}_n \sim \text{Cat}(\text{softmax}(\boldsymbol{V} \boldsymbol{x}_n)) \tag{80}$$

$$y_n | z_n \sim \mathcal{N}(\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n, \sigma_{z_n}^2), \tag{81}$$