Probabilistic Inference Coursework
Monte-Carlo Markov Chain
Variational Inference

Mark van der Wilk & Luca Grillotti

1 Introduction

1.1 About this coursework

In this coursework, we will see how to perform probabilistic predictions in the case of:

- A Logistic Regression
- A Gaussian Process Regression

by using two different types of algorithms:

- A Monte-Carlo Markov Chain method: Metropolis-Hastings
- A Black-Box Variational Inference

1.2 Provided code and examples

Skeleton Code You are provided a skeleton of the code you will have to complete in this coursework. It is simply an extended version of a solution to the previous coursework.

All the code you will have to complete is in the folder `distribution_prediction/`.

Also, a definition of the **sigmoid** generalised to matrix inputs is provided in the file `utils.py`.

Examples All the files you have to complete contain an example of prediction visualisation at the end. Those examples will only work once you have finished implementing all the functions in the file.

For more information on how to launch these examples, please refer to the `README.md` on GitLab.

2 Monte-Carlo Markov Chain

2.1 Logistic Regression

We will start by simply trying to make predictions in a simple 2-classes classification problem. In the next section, we will apply the same algorithm to Gaussian Process predictions.

In this "toy example", we take 2 classes $\mathcal{C}_1$ and $\mathcal{C}_2$. And we would like to predict $p(\mathcal{C}_1|x)$ (we suppose that $p(\mathcal{C}_2|x) = 1 - p(\mathcal{C}_1|x)$))

Our data points are in $\mathbb{R}^2$ and use a Bernoulli likelihood on the predictions, parameterised by $\theta = [\theta_1, \theta_2]^T$:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \text{Ber}(y|\sigma(\boldsymbol{\theta}^T \boldsymbol{x})) \tag{1}$$

The corresponding posterior distribution is:

$$p(\boldsymbol{\theta}|X, \boldsymbol{y}) = \frac{\prod_{n=1}^{N} p(y_n|\sigma(\boldsymbol{\theta}^T \boldsymbol{x}))p(\boldsymbol{\theta})}{p(\boldsymbol{y}|X)} \tag{2}$$

were we suppose the prior is given by: $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; 0, \sigma^2_{prior}\boldsymbol{I})$

And if we want to calculate the predictive distribution at one point, we need to compute:

$$p(y^*|X, \boldsymbol{y}, \boldsymbol{x}^*) = \int p(y^*|\boldsymbol{\theta}, \boldsymbol{x}^*)p(\boldsymbol{\theta}|X, y)\mathrm{d}\boldsymbol{\theta} \tag{3}$$

There is no closed form for that predictive distribution, but we can estimate it with a Monte-Carlo sampling:

$$p(y^*|X, \boldsymbol{y}, \boldsymbol{x}^*) \approx \frac{1}{S}\sum_{s=1}^{S} p(y^*|\boldsymbol{\theta}_s, \boldsymbol{x}^*),\ \boldsymbol{\theta}_s \sim p(\cdot|X, y) \tag{4}$$

However, sampling from the posterior distribution as required is not straightforward, as there is no closed-form solution for the marginal likelihood $p(\boldsymbol{y}|X)$.

Different classes of algorithm exist for approximating such distribution. In this coursework, we will explore two of them: **Metropolis-Hastings** (a Monte-Carlo Markov Chain method) and **Blackbox Variational Inference**.

In Metropolis Hastings, we try to simulate the sampling from a distribution p, by using a *function* $\widetilde{p} = Zp$ which is easy to evaluate. The procedure also depends on a proposal density $q(\cdot|\boldsymbol{x}^{(t)})$, which is used to determine the next candidates samples for p.

In our case, we choose the symmetric proposal density: $q(\cdot|\boldsymbol{x}^{(t)}) = \mathcal{N}(\cdot|\boldsymbol{x}^{(t)}, \sigma_q^2\boldsymbol{I})$

Task 1 Implement the function `get_log_upper_proba_distribution` in the file `metropolis_hastings_logistic.py`. That function should return the value of $\widetilde{p}$ at the given parameters.

Task 2 Implement the function `metropolis_hastings` in the file `metropolis_hastings_logistic.py`. That function performs several steps of the Metropolis-Hastings algorithm, and yields its results at every step.

Remark: That function is a **generator**: At each step, it yields several variables x1, x2 and so on... So that we can later use it this way:

```
for x1, x2, *_ in get_log_upper_proba_distribution_gp(...):
    # Use x1 and x2 Here
```

Task 3 You can now implement the prediction function `get_predictions` in the file `metropolis_hastings_logistic.py`. That function uses Eq. 4 to calculate $p(\mathcal{C}_1|X, \boldsymbol{y}, \boldsymbol{x}^*) = p(y^* = 1|X, \boldsymbol{y}, \boldsymbol{x}^*)$ for every point given as input.

Visualisation In this section, for every sampled hyper-parameter θ_i the visualisation (with `interaction=True`) shows the evolution of the predictions when S is progressively increased.

If you put `interaction=False`, then only the final prediction will be shown (once S has reached its maximal value).

An example of final prediction may be seen on figure 1

2.2 Gaussian Process

In this section, we will perform predictions on a Gaussian Process regression task. The considered kernel k is the sum of a gaussian kernel and a linear kernel:

$$k(x, x') = \left[\sigma^2_{gaussian} \exp\left(-\frac{(x-x')^2}{2l^2}\right)\right] + \left[\sigma^2_{offset} + \sigma^2_{linear}(x-c)(x'-c)\right] \tag{5}$$

As a consequence, here, $\boldsymbol{\theta}$ is a vector of 6 elements:

$$\boldsymbol{\theta} = \begin{bmatrix} \log \sigma_{gaussian} \\ \log l \\ \log \sigma_{noise} \\ \log \sigma_{linear} \\ \log \sigma_{offset} \\ c \end{bmatrix} \tag{6}$$

We had seen that, in order to make a prediction with a Gaussian Process, we need to compute:

$$p(f(\boldsymbol{x}^*)|\boldsymbol{y}, X) = \int p(f(\boldsymbol{x}^*)|\boldsymbol{y}, X, \theta) p(\theta|\boldsymbol{y}, X) \mathrm{d}\theta \tag{7}$$

However, as before, there is no closed form for that integral. So we will need to approximate a sampling from the posterior distribution $p(\theta|\boldsymbol{y}, X)$.

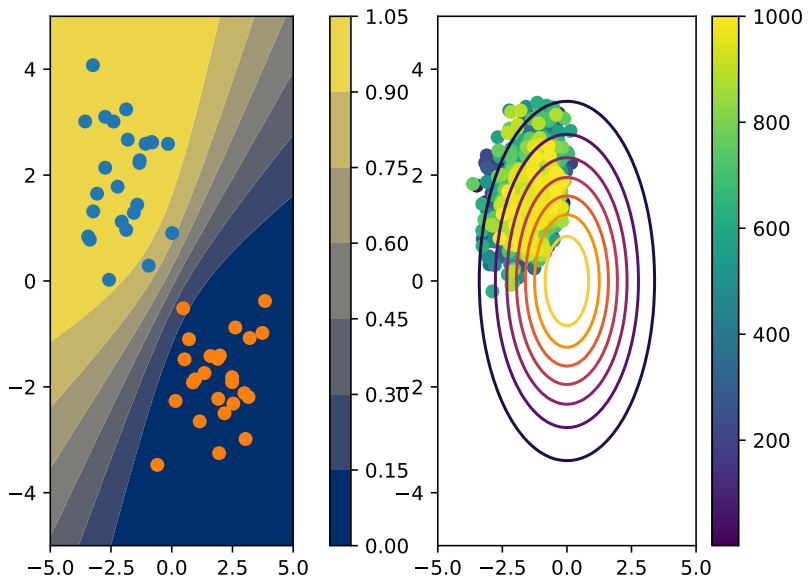

Figure 1: On the left: Contour plot showing the final predictions for the logistic regression; On the right: Simulated samples from the posterior generated by the Metropolis-Hastings algorithm (the contour plot shows the prior distribution on θ).

Task 4 Implement the function `get_log_upper_proba_distribution_gp` in the file `metropolis_hastings_gp.py`. That function should return the value of $\widetilde{p}$ at the given parameters.

Task 5 Implement the function `metropolis_hastings_gaussian_process` in the file `metropolis_hastings_gp.py`. That function performs several steps of the Metropolis-Hastings algorithm, and yields its results at every step.

Visualisation In this section, for every sampled hyperparameter θ_i the visualisation shows one sampled function from the gaussian process $\mathcal{GP}(\theta_i)$ (all these functions are already implemented), see Figure 2

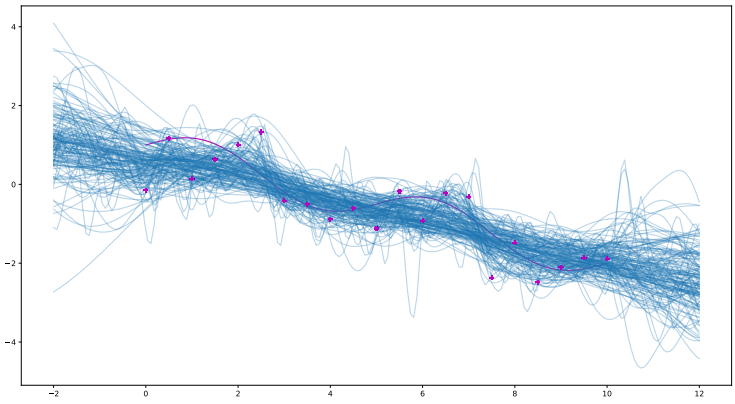

Figure 2: Sampled functions for each θ_i generated by the Metropolis-Hastings Algorithm

3 Black Box Variational Inference

3.1 Automatic Gradients

In this section, we will avoid computing the gradients by hand. Instead, we will perform automatic differentiation with the `JAX` library. All the features you will need for completing this coursework are described here (you will only need to use the `grad` function).

3.2 Logistic Regression

In this section, we will suppose our parameters $\boldsymbol{\theta}$ are sampled from a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. And we will try to find the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ maximising the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \mathbb{E}_{\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})] - \mathrm{KL}[\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \| p(\boldsymbol{\theta})] \tag{8}$$

where, we use $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \sigma^2_{prior}\boldsymbol{I})$ for the prior on $\boldsymbol{\theta}$.

Also, instead of working with $\boldsymbol{\Sigma}$, we will work with the corresponding Cholesky matrix $\boldsymbol{A}$. This is a lower-triangular matrix such that $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$.

Our goal in this part is to solve the following optimisation problem:

$$\arg\max_{\boldsymbol{\mu}, \boldsymbol{A}} F(\boldsymbol{\mu}, \boldsymbol{A}) = \mathcal{L}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{A}\boldsymbol{A}^T)) \tag{9}$$

Task 6 Implement the function `kl_div` in the file `blackbox_vi_logistics.py`, computing the KL-divergence between:

- The approximated posterior distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{A}\boldsymbol{A}^T)$
- The prior distribution on the $\boldsymbol{\theta}$: $\mathcal{N}(\mathbf{0}, \sigma^2_{prior}\boldsymbol{I})$

You may find the following formula useful:

$$\begin{aligned}&\mathrm{KL}[\mathcal{N}(\cdot|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\|\mathcal{N}(\cdot|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)] \\ &= \frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - d + \mathrm{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right]\end{aligned} \tag{10}$$

where d is the size of $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

Task 7 Implement the function `expected_log_likelihood` in the same file as before. That function computes an approximation of the expected log-likelihood, by using the re-parameterisation trick:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{\epsilon})] \tag{11}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S}\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{\epsilon}_s), \boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{12}$$

Task 8 Now you can complete the definition of the function `variational inference_logistics` which performs the Variational Inference entire procedure. In that function, you will just need to use the other functions to compute the values of:

- $\boldsymbol{\epsilon}_s$, for all $s \in [\![1, S]\!]$
- $(\boldsymbol{\nabla}_{\boldsymbol{A}} F)(\boldsymbol{\mu}, \boldsymbol{A})$
- $(\boldsymbol{\nabla}_{\boldsymbol{\mu}} F)(\boldsymbol{\mu}, \boldsymbol{A})$

Visualisation The Figure 3 shows the visualisation you should get after having implemented the functions above.

3.3 Gaussian Process

In this section, we will study the same Gaussian Process regression problem as in 2.2. We also use the same notations as in 3.2.

As in 3.2, our goal is to maximise the Evidence Lower Bound with a Gaussian Distribution. The only difference is that now $\boldsymbol{\mu}$ is a vector of size 6, and $\boldsymbol{A}$ is a lower triangular matrix of shape 6×6

Task 9 Implement the function `kl_div` in the file `blackbox_vi_gp.py`, computing the KL-divergence between:

- The approximated posterior distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{A}\boldsymbol{A}^T)$
- The prior distribution on the $\boldsymbol{\theta}$: $\mathcal{N}(\mathbf{0}, \sigma^2_{prior}\boldsymbol{I})$

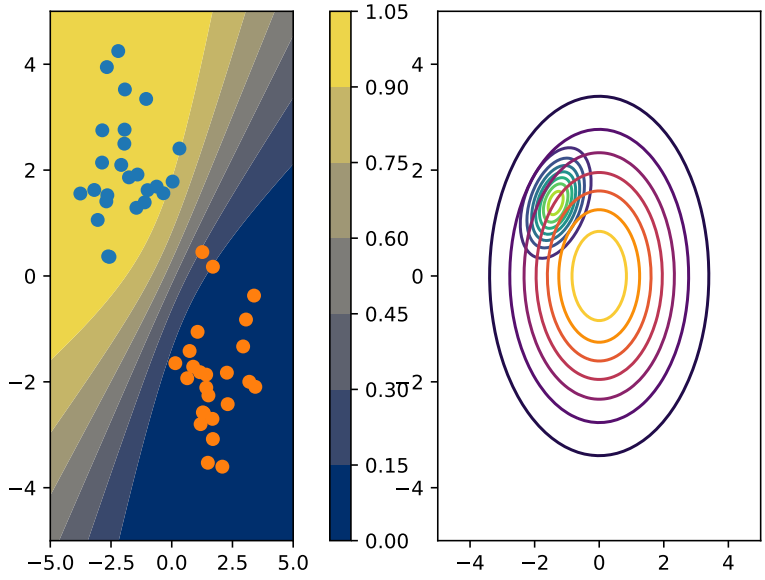

Figure 3: On the left: Contour plot showing the final predictions for the logistic regression; On the right: Prior distribution $\mathcal{N}(\mathbf{0}, \sigma^2_{prior}\boldsymbol{I})$ and approximated posterior distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}^*, \boldsymbol{A}^*\boldsymbol{A}^{*T})$

Task 10 In the same file, implement the function `expected_log_marginal likelihood`. That function computes an approximation of the expected log-marginal likelihood of the Gaussian Process, by using the re-parameterisation trick:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}[\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\mu}+\boldsymbol{A}\boldsymbol{\epsilon})] \tag{13}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S}\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\mu}+\boldsymbol{A}\boldsymbol{\epsilon}_s), \boldsymbol{\epsilon}_s \sim \mathcal{N}(\mathbf{0},\boldsymbol{I}) \tag{14}$$

Task 11 Complete the definition of the function `variational_inference_gp` which performs the Variational Inference entire procedure. In that function, you will just need to use the other functions to compute the values of:

- $\boldsymbol{\epsilon}_s$, for all $s \in [\![1, S]\!]$
- $(\boldsymbol{\nabla}_{\boldsymbol{A}} F)(\boldsymbol{\mu}, \boldsymbol{A})$
- $(\boldsymbol{\nabla}_{\boldsymbol{\mu}} F)(\boldsymbol{\mu}, \boldsymbol{A})$

Visualisation The Figure 4 shows the visualisation you should get after having implemented the functions above.

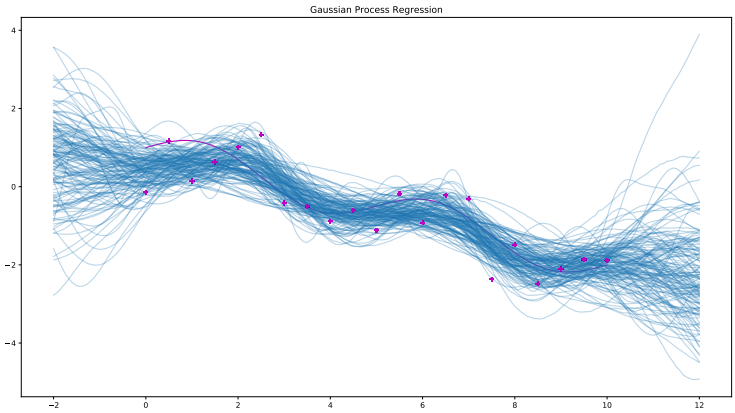

Figure 4: Sampled functions for several $\boldsymbol{\theta}_i$ sampled from the approximated posterior distribution $\mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{A}^*\boldsymbol{A}^{*T})$