MAD 2021-22, Assignment 6

Bulat Ibragimov, Kim Steenstrup Pedersen

January 4, 2022

Optional assignment: This assignment is optional, cannot be handed in for comments, and will not be graded. The purpose is to give you a chance to work with the material of the last week of the course as preparation for the exam. Reference solutions will be made available to you in Absalon on Friday 14 January 2022 at 08:00.

Monte Carlo Integration

Exercise 1 (Sampling from a Gaussian distribution, 2 points). Lets consider the case of a univariate Gaussian (Normal) distributed random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ with density $p(x|\mu, \sigma^2)$. We will use sampling to estimate expectation values for some selected functions of X, specifically we will consider the mean $E(X)$, kurtosis $\mathrm{Kurt}(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3$, and the entropy $H(X) = E[-\log(p(x|\mu, \sigma^2))]$. Here we use the Gaussian distribution, because we can easily sample directly from it, and we can prove analytically that these expectations are $E(X) = \mu$, $\mathrm{Kurt}(X) = 0$, and $H(X) = \ln(\sigma\sqrt{2\pi e})$ (where ln refers to the natural logarithm and $e = \exp(1)$), which we can use to investigate the quality of our sampling-based estimates.

a) (1 point) Implement a Python function that performs Monte Carlo integration (for any choice of μ and σ^2) of the three expectation functions mean, kurtosis, and entropy. We recommend that you use either `numpy.random.randn` or `numpy.random.Generator.normal`.

b) (1 point) Use your Python function to get approximate values for the three expectation functions stated above for the Gaussian distribution with parameter values $\mu = 2$ and $\sigma^2 = 4$. For each expectation function make a plot of the estimated value as a function of the number of samples N using an appropriate range of N.

Deliverables. a) Include your function as a code snippet in the report, b) include the three plots and comment on what you see from these.

Solution:
See the code and figures in `solution.py`.

Exercise 2 (Rejection sampling, 5 points). Lets consider a univariate random variable $X \in \mathbb{R}$ which has a distribution for which we only know the probability density function (PDF) up to a normalization constant, i.e. the PDF is proportional to

$$p(x) \propto \exp\left(-|x-2|\right) \ . \tag{1}$$

We will use sampling to estimate expectation values for some functions of X as in the previous question. But this time, we will consider the mean $E(X)$, variance $\mathrm{Var}(X) = E[(X - E(X))^2]$, and skewness $\mathrm{Skew}(X) = E\left[\left(\frac{X-E(X)}{\sqrt{\mathrm{Var}(X)}}\right)^3\right]$. But since we are not sure which distribution it is, we will be using rejection sampling using a Gaussian distribution as proposal distribution $q(x) = \mathcal{N}(\mu, \sigma^2)$.

a) (1 point) Plot the unnormalized distribution $\tilde{p}(x) = \exp\left(-|x-2|\right)$ and the proposal $k \cdot q(x) = k \cdot \mathcal{N}(\mu, \sigma^2)$ in the same graph, and argue what are appropriate values for μ, σ, and k in order to use the proposal distribution in a rejection sampling algorithm. Can you choose the parameters such that $k \cdot q(x) \geq \tilde{p}(x)$ for all x?

b) (2 points) Write a Python function that implements the rejection sampling algorithm for the distribution described by (1) using your choice of parameters μ, σ, and k. We recommend that you use either `numpy.random.randn` or `numpy.random.Generator.normal` as well as `numpy.random.Generator.random`.

c) (1 point) Implement a Python function that performs Monte Carlo integration of the three expectation functions mean, variance, and skewness by using samples generated by the function from b).

d) (1 point) Use your Python functions to get approximate values for the three expectation functions stated above for the distribution described by (1). For each expectation function make a plot of the estimated value as a function of the number of samples N using an appropriate range of N.

Deliverables. a) Include the plot of $\tilde{p}(x)$ and $k \cdot q(x)$ as well as arguments for and the values you choose for μ, σ, and k, b) include your function as a code snippet in the report, c) include your function as a code snippet in the report, d) include the three plots and comment on what you see from these.

Solution:

See the code and figures in `solution.py` for actual solution.

The distribution is actually a Laplace distribution Laplace(a, b) with parameters $a = 2$ and $b = 1$ - see definition at e.g. `https://en.wikipedia.org/wiki/Laplace_distribution`. And the theoretical expectations of the three functions are $E(x) = a = 2$, $\text{Var}(X) = 2b^2 = 2$, and $\text{Skew}(X) = 0$.

In a), appropriate choices for the parameters are $\mu = 2$ (same as the mean of the Laplace distribution), $\sigma \geq 2$, and $k = \sqrt{2\pi\sigma}$ (the normalization constant of the Gaussian, which makes both functions equal at $x = \mu$ making the rejection gap as small as possible).

In a), I do not expect that students to realize this, but I add it for you and myself: The Chebyshev inequality theorem tells us that for a random variable X with mean μ and variance σ^2, we have for any choice of the scalar $a > 0$,

$$P\left(|X - \mu| \geq a\right) \leq \frac{\sigma^2}{a^2}$$

That is, the tail probability of X is bounded from above by the variance. We therefore, need to pick the proposal distribution such that its variance is larger than or equal to that of the $p(x)$ distribution. Since I know the theoretical variance is $\text{Var}(X) = 2$, we can use this theorem to argue for $\sigma \geq 2$. However, we do not know the variance of $p(x)$ (we have not solved (c) yet), but by looking at the plot you can see that you need a variance ≥ 2 (look at the tails of $p(x)$ and $q(x)$.

Bayesian Inference and Sampling

We will in the following exercises use the binary response classification model introduced in R&G Section 4.2, pages 138 – 141. We have created a similar data set that you find in the file `binary_classes.csv` (comma separated file format). The rows represents data points and the first two columns represents input variables $\mathbf{x} = (x_0, x_1)$ and the last column is the binary target variable t (class label). We provide a function `loaddata` for reading the data in `A6_Bayes.py`.

Exercise 3 (**MAP point estimate, 4 points).** In this exercise, we will implement the MAP point estimate approach to the binary classification problem as discussed in R&G Section 4.3, pages 141 – 146.

a) (2 point) Write a Python function that implements the Newton-Raphson optimization method to find the maximum $\hat{\mathbf{w}}$ of the posterior distribution for the binary response model. The gradient and Hessian matrix for the unnormalized log-posterior needed for this procedure is available in the book.

b) (1 point) Apply your function from a) to the binary data set in `binary_classes.csv`. Start the Newton-Raphson method in $\mathbf{w} = (0, 0)^T$ and use the prior parameter value $\sigma^2 = 10$. You should perform a fixed number of iterations (e.g. 10–100) in the Newton-Raphson method.

c) (1 point) Make a plot of the probability contours of $P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \hat{\mathbf{w}})$ (similar to R&G Figure 4.4 (b)) using the function `contourplot` in `A6_Bayes.py`.

Deliverables. a) Include your function as a code snippet in the report, b) include in the report, the optimal values for $\hat{\mathbf{w}}$, c) include in the report a plot of the probability contours and your reflections on the result.

Solution:

See the code and figures in `solution_MAP.py` for actual solution.

Exercise 4 (**Metropolis-Hastings sampler, 5 points).** In this exercise, we will implement the Metropolis-Hastings sampling approach to the binary classification problem as discussed in R&G Section 4.5, pages 152 – 163.

a) (3 point) Write a Python function that implements the Metropolis-Hastings sampling algorithm for the posterior distribution for the binary response model. Use a Gaussian centered on the previous sample as proposal distribution

$$p(\tilde{\mathbf{w}}_s|\mathbf{w}_{s-1}, \rho^2) = \mathcal{N}(\mathbf{w}_{s-1}, \rho^2\mathbf{I}) \ ,$$

with $\rho^2 = 0.5$ and the identity matrix $\mathbf{I} \in \mathbb{R}^{2\times 2}$. You can use the numpy function `numpy.random.Generator.multvariate_normal` to generate samples from the proposal distribution. You must start the Metropolis-Hastings sampler at the fixed parameter vector $\mathbf{w}_0 = (2.2, 2.5)^T$ (instead of e.g. sampling from the prior). Only keep new samples, that is, whenever the algorithm suggest to keep the current sample, do not add it to your list of generated samples, only use it to generate the next proposal. You do not have to implement starting multiple Markov chains to check convergence or thinning of the samples to ensure that the samples are independent, or other advanced procedures (it is not needed for this problem and our choice of parameter values).

b) (1 point) Apply your function from a) to the binary data set in `binary_classes.csv`. Use the prior parameter value $\sigma^2 = 10$ and generate 10.000 samples (after removing duplicates, you should expect about ≈ 7.800 samples). Make a plot of a subset of the samples (e.g. in the order of 1.000 samples) like R&G Fig. 4.12 (a), page 161 (without the posterior contours).

c) (1 point) Make a plot of the probability contours of the predictive probability $P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2)$ using the random samples generated from your Metropolis-Hastings sampler (similar to R&G Figure 4.12 (e), page 161). You can use the function `contourplot` in `A6_Bayes.py` for this, but you need to write a new `prob(X, w)` function that implements the predictive probability (see the function `sigmoid` in `A6_Bayes.py` for an example). For this function, recall that given a set of M samples $\mathbf{w}_1, \ldots, \mathbf{w}_M$ we can use the approximation

$$P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2) \approx \frac{1}{M}\sum_{s=1}^{M} P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{w}_s) = \frac{1}{M}\sum_{s=1}^{M} \frac{1}{1 + \exp(-\mathbf{w}_s^T \mathbf{x}_{\text{new}})} \ .$$

Deliverables. a) Include your function as a code snippet in the report, b) include the plot of your samples in the report, c) include in the report a plot of the probability contours and your reflections on the result.

Solution:

See the code and figures in `solution_MH.py` for actual solution.