Computer Assignment MT5012 VT2023

Hand in: Monday, May 22, 23:59

The computer assignment consists of three simulation topics covered in the lectures. You can work on the exercises in pairs and use any resources you need to solve them, including Chat-GPT. However, ensure that the report is written in your own words.

For each task, provide a clear explanation of your approach. Additionally, make sure that the code you submit is readable. Please note that unreadable code will not be graded.

If you have any questions you can reach me at: **taariq.nazar@math.su.se**.

Monte-Carlo integration

Extra resources:

- Monte Carlo Simulation & Importance Sampling **link** (*12:45 min*)

Note: *These videos are* ***not*** *necessary to complete this part of the assignment. However, I find them very helpful for building an understanding of these concepts.*

Crude Monte-Carlo integration aims at numerically approximate an expectation/integral $\mathrm{E}[h(X)] = \int h(x) f_X(x)\, dx$ by

1. Simulate a realisation X
2. Compute $h(X)$
3. Repeat the above two steps a large number of times and return the average computed h

The law of large numbers ensures convergence and given finite variance Var(h(X)) we can also us the central limit theorem to approximate the error involved. According to the latter, the magnitude of approximation error will decrease at the rate $N^{-1/2}$, which is painfully slow. In comparison to many other numerical integration techniques, it does however have the advantage of not depending directly on dimensionality of X. Another advantage is that f_X need not be analytically available as long as we can simulate from it.

As a basic example, we can approximate an integral on the unit interval using that $\int_0^1 h(x)\, dx = \mathrm{E}[h(X)]$ if X is uniformly distributed on $[0, 1]$. Let $h(x) = \sin(x)$, we can simulate by the following algorithm

```
X = uniform(N)          # Draw N uniform random variables
h = sin(X)              # Evaluate
mean(h)                 # Sample mean
[Result]: 0.4614475
```

which is reasonably close to the true value $1 - \cos(1) = 0.4596977$.

Importance sampling

Importance sampling generalises crude Monte Carlo by writing

$$\mathrm{E}[h(X)] = \int h(x) f_X(x)\, dx = \int h(x) f_X(x) \frac{f_Y(x)}{f_Y(x)}\, dx = \mathrm{E}\big[\frac{h(Y) f_X(Y)}{f_Y(Y)}\big] = \mathrm{E}[h(Y)w(Y)],$$

which is valid as long as $f_Y(x) > 0$ for all x such that $f_X(x) > 0$ (the property of *absolute continuity*) and where $w(Y) = \frac{f_X(Y)}{f_Y(Y)}$. The expression is then approximated by simulating from Y and averaging simulated $h(y)w(y)$.

There are two main arguments for this construction:

- The random variable/vector Y may be easier to simulate than X,
- the variance $\mathrm{Var}((h(Y)w(Y))$ may be much smaller than $\mathrm{Var}(h(X))$, leading to a smaller Monte-Carlo error.

The name of the method refers to applications of the latter; say we want to approximate $P(X > a) = \int \mathbf{1}(x > a) f_X(x)\, dx$ for a value a far out in the tail of f_X. If we proceed by simulating $X_1, \ldots, X_N$ from f_X, and take the average of $\mathbf{1}(X_i > a)$, we will be spending most of our simulation budget on a region that is *unimportant*; in this case where the integrand $\mathbf{1}(x > a) f_X(x)$ equals zero. We would like to spend more effort on the *important* region $x > a$ in order to increase efficiency/decrease variance.

For illustration, we consider the problem of approximating the probability of a rare event $p = P(X > 10)$ when X has a standard exponential distribution. Of course, Monte Carlo is not needed here since $P(X > 10) = \exp(-10) = 0.000045$. First, the crude Monte-Carlo gives the following based on 10000 draws:

```
X = Exponential(N, rate=1)  # Draw N samples from standard exponential.
h = (X > 10)                # 1 if sample is > 10 else 0
mean(h)                     # Average
[Results]: 0
```

Not a single draw was larger than 10 and the approximation returned is 0. While this is "close" to the true value in an absolute sense, when approximating rare events the relative error is more relevant (i.e. we are concerned whether the probability is 10^{-5} or 10^{-10}, not just that it is close to zero). Hence we want

$$\frac{\hat{p}}{p} - 1$$

rather than $p - \hat{p}$ to be small.

Using importance sampling, we may replace the standard exponential with a distribution that is more likely to excess 10.

Task 1: Implement an importance sampler with $Y \sim Exponential(1/10)$ and compute the relative error. Graph the convergence and a 95% confidence interval of the estimate w.r.t the number of samples with and without importance sampling.

General procedure:

1. Generate samples for Y.
2. Evaluate $h(Y)$.
3. Evaluate $w(Y)$.
4. Estimate $\mathrm{E}[h(Y)w(Y)]$.

Task 2: An "optimal" importance sampling distribution for the above problem is $Y = X + 10$, a standard exponential shifted to the right. Try it. In what sense is it "optimal"?

Default of an insurance company

Consider an insurance company that starts with capital C at year $t = 0$. Each year, it gains $p > 0$ in premiums and loses $X \sim F$ independently of other years. Its capital at the end of year t is thus

$$Z_t = C + tp - \sum_{i=1}^{t} X_i.$$

What is the probability of default within the next T years? Defaulting can be interpreted as not being able to pay for the losses. That is, $Z_t < 0$. Therefore, we can write the probability of default as

$$P(\min_{0<t\leq T} Z_t < 0) = E(\mathbf{1}(\min_{0<t\leq T} Z_t < 0)).$$

Which in plain words is: *What is the probability that the the insurance company will have negative capital for any one of the years $t \in [1, T]$.*

Approximating this probability can be done by crude Monte-Carlo as follows:

1. Generate an outcome(sample) $h(\mathbf{Z})$:
    1. Simulate the losses, $\mathbf{X} = (X_1, \ldots, X_T)$.
    2. Compute the capital, $\mathbf{Z} = (Z_1, \ldots, Z_T)$ using $\mathbf{X}$.
    3. Evaluate if the insurance company has defaulted given the simulated outcome, $h(\mathbf{Z}) = \mathbf{1}(\min_{0<t\leq T} Z_t < 0)$.
2. Average over multiple outcomes of $h(\mathbf{Z})$.

Task 3: Approximate the probability of default using the following parameters; $T = 20$, $C = 15$, $p = 1$ and loss-distributions independent $LogNormal(0, 1/4)$.

Again this will be inefficient if the probability of default is very small. A simple importance sampling alternative is to replace $X_i \sim LogNormal(0, 1/4)$ by $Y_i \sim LogNormal(\mu, 1/4)$, where the aim is to increase the probability of default (but not make it too close to 1).

Task 4: Approximate the probability of default using the same parameters as in previous task, using importance sampling described above. Choose a suitable μ and report what values are appropriate.

Markov chain Monte Carlo

Extra resources:

- Markov Chain Monte Carlo **link** (*12:10 min*)
- Metropolis-Hastings **link** (*18:14 min*)
- Gibbs Sampling **link** (*8:48 min*)

Note: *These videos are* ***not*** *necessary to complete this part of the assignment. However, I find them very helpful for building an understanding of these concepts.*

Simulating directly from a given distribution $X \sim P$ can be difficult, in particular when the dimension of X is high. It turns out to be relatively straightforward to simulate a Markov chain which has P as its stationary distribution though. Less straightforward is to do it in an efficient way! Again we want to approximate $\mathrm{E}[h(X)]$ based on random draws $X_1, \ldots, X_N$. This time however

- $X_1, \ldots, X_N$ form a Markov chain, hence they will in general not be independent
- X_i is only guaranteed to have the same distribution as X in the limit.

Due to dependence, variance of $\bar{h}$ is now

$$\mathrm{Var}(n^{-1}\sum_{i=1}^{N} h(X_i)) = n^{-2}\sum_{i=1} \mathrm{Var}(h(X_i)) + n^{-2}\sum_{i\neq j} \mathrm{Cov}(h(X_i), h(X_j)),$$

where the first term will be approximately $n^{-1}\,\mathrm{Var}(h(X_n))$ as for the independent case. For a Markov chain with strong serial dependence, the second term will dominate the Monte carlo error/variance.

Gibbs sampling

Gibbs sampling is a Markov Chain Monte Carlo (MCMC) algorithm that is used to generate samples from a joint probability distribution by iteratively sampling from the conditional distributions of each variable, given the values of all other variables. This sampling method is particularly useful when the joint distribution is complex and is difficult to directly sample from. Which is typical when the joint distribution has a large number of variables.

To familiarise yourself with this method you will start off by looking at the simplest case. Which is sampling from a multivariate normal correlated variables.

Let (X_1, X_2) follow a bivariate Normal distribution with mean $\mu = (0, 0)$ and

covariance matrix

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

Task 5: Derive the conditional distributions $X_1|X_2 = x_2$ and $X_2|X_1 = x_1$.

Task 6: Implement a Gibbs sampler that samples from the bivariate distribution described above. Let $\rho = 0.8$. Does the Monte Carlo variance depend on ρ? Conclusions?.

The 2D Ising model

The square lattice ising model is a classical model in statistical physics describing interacting magnetic spins. It defines a probability distribution on a $k \times k$ matrix σ(state of the system), with entries $\sigma_{ij} \in \{-1, 1\}^2$, where $i, j \in J_k = \{1, \ldots, k\}^2$, defined by

$$\pi_\sigma = C_\beta \exp\big(-\beta \sum_{i \sim j} \sigma_i \sigma_j\big)$$

where the sum is over all *neighbours*. We say that a pair are neighbours if they are separated by a distance of 1 in the horizontal or vertical direction. For instance $(1, 1)$ and $(1, 2)$ are neighbours but $(1, 1)$ and $(2, 2)$ are not. It can be shown that

$$p(\sigma_i = 1|\{\sigma_j; j \neq i\}) = 1/(1 + \exp(2\beta \sum_{j:i \sim j} \sigma_j)),$$

note that this does not depend on the hard-to-compute normalising constant C_β. Where the sum is over all neighbours.

Task 7: Use Gibbs sampling to simulate the system described above. Present a visualisation of the state dynamics(over time). Let $\beta = 10$.

Note that this method is computationally demanding for large values of k, try it with a small value (e.g. $k = 10$) first and then increase if computer power allows.

Task 8: Simulate and visualise the Ising model for positive and negative values of β and report the result.

Task 9: Note that by symmetry, $P(\sigma_i j = 1) = P(\sigma_i j = -1) = 1/2$. Simulate a long chain (with a modest k) and $\beta = 10$. What is the proportion of values of, say, $\sigma_{(2,2)}$ that equals 1? Conclusion?

A single server queue

A single-server queuing system is completely determined by a set of arrival times $(a_1, \ldots, a_N)$ and a set of service times $(s_1, \ldots, s_2)$. In order to compute $N(t)$, the number of customers in the queue at time t, it is convenient to first compute the exit times. This can be done recursively as follows:

- The first customer exits at time $e_1 = a_1 + s_1$.
- The second customer exits at time $e_2 = \max(a_2, e_1) + s_2$.
- ...
- The N:th customer exits at time $e_N = \max(a_N, e_{N-1}) + s_N$.

that is, if the queue is empty when customer k arrives he/she exits at s_k time-units after arrival. If there are people in the system, he/she exits at s_k time-units after the previous customer.

Task 10: Write a function that returns a vector of exit-times given arrival and service times. Then write a function that that computes the number of people in the queue given arrival and service times.

Task 11: Simulate homogenous Poisson arrival and service times using suitable parameters. Plot the queue length over time.

A queuing process with non-homogenous Poisson arrivals

The homogenous Poisson process used for arrivals above is easily generalised to a non-homogenous one by replacing the uniformly distributed arrival times with e.g. a Beta distribution.

Task 12: Write a function that simulates a queuing system with non-homogenous Poisson arrivals (your choice!) and non-exponential service times (your choice, make them positive though!) and returns the maximum observed queue-length. Run the function 1000 times and visualise the maximum queue-length distribution with a histogram.