Assignment 6

Please follow instructions on the assignments page

About this assignment: Part 1: Working with importance sampling. Part 2: An example of rejection sampling. Part 3: Illustrating the curse of dimensionality. Part 4: MCMC – the algorithm and visualizing the posterior.

1. (*Importance sampling*) As explained in class, *importance sampling* means using samples from a distribution $q(\cdot)$ rather than the target distribution, $p(/cdot)$, and then compensating using the *likelihood ratio*

$$L(x) = \frac{p(x)}{q(x)} \; .$$

A simple application is estimating an expected value in the p distribution using q samples.

$$\mathrm{E}_p[\, u(X)\,] = \int u(x)\, p(x)\, dx = \int u(x)\, L(x)\, q(x)\, dx = \mathrm{E}_q[\, u(X) L(X)\,] \; .$$

This exercise explores importance sampling on the "toy problem" of estimating moments of a standard normal

$$M_{2n} = \mathrm{E}_{\mathcal{N}(0,1)}\left[\, X^{2n}\,\right] = (2n-1)(2n-3)\cdots 3 \; .$$

In this example, p is the standard normal probability density. Take q to be $\mathcal{N}(0,\sigma^2)$. Let $\widehat{M}_{2n,\sigma}$ be the estimate of M_{2n} using importance sampling with q. For fourth, sixth, etc. moments, make a graph (using Monte Carlo) of the variance of $\widehat{M}_{2n,\sigma}$ as a function of σ for a fixed sample size large enough to make reasonably accurate estimates of the variance of the estimator. Using computational experiments and looking at the plots, try to see how the optimal σ^2 for q depends on n. Make plots of the integrand (prefactor omitted)

$$f(x) = x^{2n} e^{-\frac{x^2}{2}} \; .$$

Note (from the plot) $x_* = \arg\max_x f(x)$ and see how this is related to the optimal σ.

Comments:

- There is an explicit formula for $\mathrm{var}(\widehat{M})$ as a function of σ. An AI should be able to find it for you. It just involves integrals of quadratic exponentials, which are known in closed form. I didn't assign these formulas because they have little educational value in this context. Please learn about the variance estimator using computational experiments rather than the formula. In almost any other situation there would be no "explicit formula" alternative to computational experiments.
- Intuitively, it seems natural that σ^2 should be larger than 1. You want samples farther away because the max of f is farther away. Either from the explicit formula or by examining the likelihood ratio, or by computational experiments, you can see that the variance is infinite if $\sigma^2 < 1$.

2. (*An example of rejection sampling*) Write a rejection sampler that produces standard normal samples by rejection from a two-sided exponential. The target distribution is

$$p(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}}$$

The proposal distribution is

$$q(x) = \frac{1}{2} e^{-|x|} \ .$$

Use histogram based density estimation to check that your samples have the density p.

3. (*Curse of dimensionality for sampling*) The *curse of dimensionality* is the general phenomenon that ideas and algorithms that seem reasonable in 2D or 3D are actually very bad in higher dimension. Sometimes "higher" starts with $D = 4$, sometimes intuition from 2D and 3D holds until D is larger. This exercise explores rejection sampling from proposals uniformly distributed in a box as a way to make samples uniformly distributed in a ball. The target distribution is

$$p(x) = \frac{1}{Z} \mathbf{1}_{\|x\|_2 \leq 2} = \frac{1}{Z} \begin{cases} 1 & \text{if } \|x\|_2 \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Except for the normalization constant (which is irrelevant here), p is the *characteristic function* (in mathematical analysis) or *indicator function* (in probability) of the unit ball in D dimensions. Take the proposal distribution to be that X is uniformly distributed in the side-length 2 box in D dimensions

$$q(x) = \frac{1}{2^D} \prod_{k=1}^{d} \mathbf{1}_{|x_k| \leq 1}(x_k) = \frac{1}{2^D} \begin{cases} 1 & \text{if } -1 \leq x_k \leq 1 \text{ for all } k = 1, \cdots, D \\ 0 & \text{otherwise} \end{cases}$$

You can sample q using D independent standard uniforms and mapping the interval $[0, 1]$ to the interval $[-1, 1]$. The rejection algorithm is: generate $X \sim q$, accept if $\|X\|_2 \leq 1$, otherwise reject. The efficiency is determined by the overall acceptance probability

$$Z_D = \Pr_q(\|X\|_2 \leq 1) \ .$$

Write a routine that estimates Z_D as a function of D. Make a semilog plot of the result. You should be able to conclude that the acceptance probability decreases exponentially with D. It should take quite a lot of computing time (on a laptop) to estimate Z_D accurately enough to make a convincing case for this exponential behavior. Do two correctness checks on your code (and include the results in your writeup)

(a) The area of the unit "ball" (disk) in 2D is π and in 3D it's $\frac{4}{3}\pi$. This gives formulas for Z_2 and Z_3. Your AI (or a simple web search) may point you to a formula for the volume in D dimensions. This confirms the exponential badness of rejection sampling that you found in the experiment

(b) Make a plot of the marginal density of X_1 produced by the sampler (or any of the other components) and compare it the the known theoretical density, In 2D, the marginal density is proportional to the length of the vertical segment inside the disk over x_1. In 3D, it's the area of the 2D circle in x_2 and x_3 "above" x_1 inside the ball.

4. (*First experiment with MCMC*) The target density is the exponential quartic

$$p(x) = \frac{1}{Z} e^{-x^4} \ .$$

Sample this using random walk Metropolis with Gaussian proposals with step size r (as described in class). Make a plot of the distribution of samples produced from the Markov chain and check that they match p. Use the samples to estimate

$$M_2 = \mathrm{E}_p\big[X^2 \big] \ .$$

Compute Z and the integral $\int x^2 e^{-x^4} dx$ by numerical quadrature (as explained in class). Calculate and plot the auto-correlation function $\widehat{\rho}(s)$ for various values of the step size parameter.

5. (*MCMC and Bayesian statistics*) A scalar linear *auto-regressive* "process" satisfies the linear noisy recurrence

$$X_{n+1} = a_0 X_n + \cdots + a_k X_{n-k} + \epsilon_n \ .$$

The noise "innovations" (also called "shocks" in economics and finance because they represent responses to shocking new information) are supposed to be independent mean zero Gaussian with common variance σ^2. This process is called $AR(k+1)$, where $k+1$ is the number of lags. The simplest is $AR(1)$, which is $X_{n+1} = a_0 X_n + \epsilon_n$. As background, an $AR(k)$ process is *unstable* if there is a "homogeneous" solution (solution with no shocks) of the form $X_n = z^n$, with $|z| > 1$, which implies

$$z^{n+1} = a_0 z^n + \cdots + a_k z^{n-k} \ .$$

These equations are the same for every n. An "unstable" z would be a root of the *characteristic polynomial*

$$P(x) = z^{k+1} - a_0 z^k - \cdots - a_k \ .$$

If all the roots of the characteristic polynomial $P(z) = 0$ have $|z| < 1$, the recurrence is *stable*. It's easy to tell in practice. Just run the process. If it's unstable, the X_n grow exponentially with n, which is hard to miss in the output. If the X_n just "bounce" randomly without growing, that's a stable process. We always assume the starting values $X_0, \cdots, X_k$ are given and not random. The problem is to estimate the regression coefficients a_j and the noise parameter σ^2 from a dataset $X_0, \cdots, X_N$ (imagining that N is reasonably large and k is not large).

(a) Describe an algorithm for calculating the likelihood function $L(X|\sigma^2, a_0, \cdots, a_k)$ given the data sequence X.

(b) Write a code to sample the posterior with a flat prior and one of the datasets `Assignent6_short.csv` or `Assignent6_long.csv` It should use the algorithm from part (a) to evaluate the likelihood function. Try fits with 1, 2, and 3 lags. If the code is well written, it should be simple to do a run with a specified number of lags. Make triangle plots for the three lag problem. The dimension of the sample space is the number of lags plus 1, for the innovation (ϵ_n) variance. Make a plot of auto-correlation functions of the state variables and observe the time scale on which they decays to zero. The `scipy` package `scipy.signal.correlate` can compute the auto-covariance function. The time scale should depend on the parameters in the MCMC. Make plots of a state variable as a function of the MCMC "time" variable for various values of the MCMC parameters. You need to do this intelligently. If the sequence is too long, it will just look like noise. If the sequence is too short, it's just some random points. Think about using the auto-correlation time to decide how long a sequence to plot. If you start from the beginning, you should see some "burn in" before the time series settles into its long time behavior.

(c) (totally voluntary, only if you're curious) Use the `emcee` sampler of Foreman-Mackey. It should have shorter auto-correlation time.

Note. This problem is "trivial" and doesn't really need MCMC because it's a linear least squares problem. Nevertheless, many of the issues you see in non-linear problems arise here.