Домашнее задание

4.1 Integral

На занятии мы обсудили метод Monte-Carlo для вычисления произвольных интегралов. Давайте попробуем его применить.

Пусть нам необходимо для некоторой функции f найти $\int f$ по некоторому заданному множеству.

Идея метода M-C состоит в том, чтобы рассмотреть случайную величину $\eta = \frac{f(\xi)}{\rho_\xi(\xi)}$, где ξ — некоторая случайная величина с плотностью ρ_ξ. Матожидание этой случайной величины равно в точности интегралу f по носителю случайной величины ξ. Промоделировав достаточно большое количество реализаций η и осреднив их, мы получим $\gamma^{(N)} = \frac{\eta_1 + \cdots + \eta_N}{N} \to \mathsf{E}\eta = \int_{supp\,\xi} f$ в силу Закона Больших Чисел.

Остается открытым вопрос, как выбрать интегрирующую плотность ρ_ξ. Во-первых, ее носитель должен совпадать с областью интегрирования (или хотя бы содержать его). Во-вторых, мы должны уметь эффективно получать реализации случайной величины ξ и эффективно вычислять плотность ρ_ξ в произвольных точках. В-третьих, мы хотим минимизировать число вычислений целевой функции, следовательно, мы хотим максимально улучшить сходимость.

Как очевидно из неравенства Чебышева, в случае, когда случайная величина η имеет конечную дисперсию, ошибка имеет порядок (не больше, чем) $\frac{\sqrt{\mathsf{D}\,\eta}}{\sqrt{N}}$, следовательно, мы хотим уменьшить дисперсию.

Выражение для дисперсии имеет вид:

$$\mathsf{D}\,\eta = \int \frac{f(x)^2}{\rho_\xi(x)} dx - \left(\int f(x) dx \right)^2,$$

где все интегралы берутся по носителю случайной величины ξ.

Известно, что минимум достигается при $\rho_\xi(x) = \frac{f(x)}{\int |f|}$. На практике использовать этот факт напрямую как правило невозможно, потому что построение такой интегрирующей плотности предполагает вычисление интеграла $\int |f|$, что едва ли менее просто, чем решение исходной задачи.

Однако мы можем рассмотреть интегрирующую плотность, которая ведет себя "похоже" на $|f|$. Особенно важно, чтобы на бесконечности (т.е. при $x \to \pm\infty$) и в полюсах f (т.е. при x: $\lim_{t \to x} f(x) = \pm\infty$) их поведение совпадало в смысле порядка убывания/возрастания (без этого первый интеграл в выражении дисперсии может даже перестать сходиться).

Предлагается (для данного интеграла) построить несколько интегрирующих плотностей, промоделировать процедуру. Убедиться, что процедура корректна (т.е. сходится к истинному значению) и сравнить плотности в смысле RMSE получаемых оценок. Истинное значение интеграла найти с помощью детерминированных методов, например, с помощью функции `integrate()` в `R` или `scipy.integrate` в Python.

Для $N = 10, 100, \ldots, 10^6$ изобразить на одном графике зависимость RMSE $\sim N$ (в логарифмической шкале) для всех выбранных плотностей. *Прикладывайте график к письму, пожалуйста, я не смогу запустить каждую программу.*

Опционально: сравнить плотности с точки зрения посчитанных теоретически дисперсий, сопоставить результаты.

Собственно, интегралы (можно выбрать любой один):

1. $\int_0^\infty \cos x \cdot e^{-x^2}\, dx$

2. $\int_0^1 (\sin x)^{-3/4}\,dx$

Напоминаю, что RMSE это Rooted Mean Squared Error, т.е. корень из среднего квадрата отклонения оценки от истинного значения:

$$\mathrm{RMSE}_\gamma^{(\mathrm{M})} = \sqrt{\mathrm{MSE}_\gamma^{(M)}} = \sqrt{\frac{(\gamma_1 - I)^2 + \cdots + (\gamma_M - I)^2}{M}} \approx \sqrt{\mathsf{E}(\gamma - I)^2} = \mathrm{RMSE}_\gamma,$$

где γ — случайная величина, оценка неизвестного значения, γ_i — полученные независимые реализации оценок (в данном случае, оценки интеграла, каждая из которых получена с помощью отдельной серии моделирования) I — точное значение (в данном случае, интеграла).

То есть здесь мы *оцениваем* значение RMSE с помощью многократного моделирования и осреднения (такой вот двойной Монте-Карло, да).

M можно взять 100 – 1000, этого достаточно для получения устойчивого поведения RMSE.

4.2 Two Problems

Рассмотрим две небольших задачки:

Задача. Устройство состоит из k блоков. Время работы каждого блока до отказа — случайная величина t_i с распределением $Exp(\lambda_i)$, t_i совместно независимы. Устройство отказывает при выходе из строя любого из блоков. Найти распределение величины t — времени работы устройства до отказа.

Задача. В уездном городе N есть один кольцевой автобусный маршрут, по которому ходит k автобусов. Все автобусы едут в одну сторону, с одной и той же скоростью и каждый проходит маршрут ровно за один час. При этом все автобусы выходят на линию случайно (и независимо), поэтому в произвольный момент времени каждый из может находится равновероятно в любой точке маршрута.

Найти распределение времени ожидания ближайшего автобуса и время ожидания последнего (k-го по порядку) автобуса. Назвать это распределение. *Для желающих: найти время ожидания i-го по порядку автобуса.*

Будут ли независимы время ожидания первого и последнего автобуса как случайные величины?

Обе эти задачи можно (и нужно) решить теоретически. Это несложно. После чего, нужно промоделировать происходящее в цикле и получить выборку из случайных величин ("времен"). Затем по этой выборке построить гистограмму или ядерную оценку плотности (Kernel Density Estimation) и на ее фоне нарисовать плотность распределения, полученную теоретически. Сравнить.

Имейте ввиду, что при малом объеме выборки гистограмма очень сильно отличается от реальной плотности, а ядерная оценка это вообще весьма мутная вещь (хоть и выглядит красиво), поэтому на практике отвергать гипотезу о совпадении распределений надо достаточно осторожно.

Рекламная пауза. Для рисования всяких околостатистических графиков в Python я рекомендую пакет `seaborn`.