Глава 2

Занятие второе. О генерировании случайных величин, векторов и более сложных объектов, о том, как случайное помогает неслучайному, о прелестях Монте-Карло и отжиге и том, как полезно владеть эргодической теорией для цепей Маркова

2.1 Как генерировать случайные величины

В рамках второго занятия мы будем говорить о том, как создавать случайные объекты, а также немного поговорим о некоторых не связанных с теорией вероятностей проблемах, в которых владение нашим аппаратом может существенно помочь.

2.1.1 Метод обратной функции

Начнем мы с простого факта, который, вероятно, известен большей части из вас: *если мы хотим сгенерировать величину X с функцией распределения $F(x)$, то нужно взять величину Y, имеющую равномерное на отрезке $[0,1]$ распределение, и положить $X = F^{-1}(Y)$.*

Этот принцип работает для любой функции $F(x)$, включая разрывные, нужно лишь правильно определить обратную функцию: $F^{-1}(y) = \inf\{x : F(x) \geq y\}$.

Вопрос 1. Как доказать этот факт?

Таким образом, чтобы сгенерировать случайную величину с распределением F нам достаточно:
1) датчика равномерных $R[0,1]$ чисел,
2) функции G, обратной к функции распределения F.

Пример 1. Функция распределения бернуллиевской случайной величины с параметром

p будет ступенчатой

$$F(x) = \begin{cases} 0, & x < 0, \\ 1-p, & x \in [0,1), \\ 1, & x \geq 1. \end{cases}$$

Тогда

$$F^{-1}(y) = \begin{cases} 0, & y \in [0, 1-p], \\ 1, & y \in (1-p, 1]. \end{cases}$$

Следовательно, рассматривая случайную величину $Y \sim R[0,1]$ и полагая $X = 1$ при $Y > 1-p$ и $X = 0$ при $Y \leq 1-p$, мы получим бернуллиевскую величину. Конечно, догадаться до такого преобразования можно и без всяких обратных функций.

Пример 2. Чтобы получить величину с ф.р.

$$F_X(x) = \begin{cases} x^2, & x \in [0,1], \\ 0, & x \in [0,1], \end{cases}$$

мы должны взять $\sqrt{R}$, где $R \sim R[0,1]$.

Вопрос 2. А как сгенерировать величину, имеющую ф.р.

$$F(x) = \begin{cases} 0, & x < 0, \\ (2 - e^{-x})/2, & x \geq 0 \end{cases}$$

Этот метод называется *методом обратной функции*.

2.1.2 Метод выбора с отклонением

Метод обратной функции позволяет сгенерировать случайную величину, однако на практике может оказаться трудозатратным считать функцию распределения и еще более затратно обратную функцию. В таком случае можно использовать другие методы, например, так называемый *метод выбора с отклонением* (в англоязычной литературе он называется Rejection или Acceptance-Rejection method):
Допустим X имеет плотность $f(x)$ (или дискретное распределение $f(x) = \mathbf{P}(X = x)$), а мы умеем генерировать величину Y с плотностью $g(x)$ (вероятностью $g(x) = \mathbf{P}(Y = x)$), причем $f(x)/g(x) \leq c$. Тогда мы можем:
1) Сгенерировать величину Y с плотностью (распределением) $g(x)$.
2) При $Y = y$ с вероятностью $f(y)/(cg(y))$ положить $X = y$, иначе повторить 1), 2).

Вопрос 3. Почему полученный метод действительно работает а) в дискретном случае б) в абсолютно-неперерывном случае?

Пример 3. Для генерирования случайной величины со значением от 1 до 7 мы можем трижды бросить монетку и сопоставить полученным исходам числа от 1 до 8. Тогда $f(x) = 1/7$ при $x = 1, ..., 7$, $g(x) = 1/8$, $x = 1, ..., 8$. Следовательно, $f(x) \leq 8g(x)/7$ при любом x. Таким образом, мы должны сгенерировать величину Y с помощью трех

бросков монеты, если полученное значение y будет из набора $\{1, ..., 7\}$, то мы оставляем y с вероятностью $f(y)/(cg(y)) = 1$, а если $y = 8$, то мы оставляем y с вероятностью $f(y)/(cg(y)) = 0$.
Иначе говоря, при выпадении 1,...,7 мы заканчиваем эксперимент с таким результатом, а при выпадении 8 перебрасываем три броска монеты.

Задача 1. Для построения величины с плотностью $f(x) = 20x(1-x)^3$, $0 < x < 1$ воспользоваться величиной с плотностью $g(x) = 1$, $0 < x < 1$.

Вопрос 4. Какое c можно взять в этом случае?

2.2 От одномерного случая к многомерному

Умея генерировать случайные величины, мы можем сгенерировать вектор с независимыми компонентами, имеющими заданные функции распределения. Для этого, как видим, нам достаточно иметь датчик независимых $R[0,1]$ величин. А как же моделировать вектор, чьи компоненты зависимы?

2.2.1 Метод выбора с отклонением в многомерном случае

Отметим, что в процедуре метода выбора с отклонением нет никаких отсылок к размерности объекта, поэтому его без каких-либо модификаций можно использовать для генерирования дискретных или абсолютно-непрерывных векторов.

Пример 4. Для выбора случайной точки в единичном круге можно выбрать равномерную точку в квадрате $[-1,1]\times[-1,1]$ и перегенерировать результат, если точка не попала в круг.

Вопрос 5. Почему этот алгоритм соответствует методу выбора с отклонением?

2.2.2 Метод условных распределений

Здесь нам поможет условное распределение. Идея такова — для получения вектора $(X_1, ..., X_n)$ мы:
1) Генерируем X_1 одним из указанных выше методов, например, как $F^{-1}(R_1)$, где $R_1 \sim R[0,1]$;
2) Смотрим на X_1 как на фиксированный параметр x_1 и генерируем X_2 на основе условной функции распределения $G(x) = F_{X_2|X_1}(x|x_1) = P(X_2 \le x|X_1 = x_1)$, $X_2 = G^{-1}(R_2)$, где $R_2 \sim R[0,1]$;
3) Смотрим на X_1, X_2 как на фиксированные параметры x_1, x_2 и генерируем X_3 как $H^{-1}(R_3)$, где $H(x) = F_{X_3|X_1,X_2}(x|x_1,x_2)$ и так далее.

Пример 5. Смоделируем вектор с совместной плотностью

$$f_{X,Y}(x,y) = \begin{cases} e^{-y} & 0 < x < y, \\ 0, & \text{иначе} \end{cases}$$

Найдем сперва плотность $f_X(x)$ и ф.р. $F_X(x)$. Вспомним, что плотность компоненты X двумерного вектора (X, Y) находится путем интегрирования по второй переменной: при $x > 0$

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dy = \int_x^{\infty} e^{-y}dy = e^{-x},\ F_X(x) = 1 - e^{-x}.$$

Следовательно, X генерируется как $-\ln(1 - R_1)$, где $R_1 \sim R[0, 1]$ (на самом деле можно брать $-\ln R_1$, поскольку $1 - R_1$ распределена также как R_1).
Условная плотность $Y|X$ при $0 < x < y$ равна

$$f_{Y|X}(y|x) = \frac{f_{X,Y}(x, y)}{f_X(x)} = e^{-y}/e^{-x} = e^{x-y},$$

откуда

$$F_{Y|X}(y|x) = \int_x^y e^{x-z}dz = 1 - e^{-(y-x)},$$

при $0 < x < y$. Следовательно, получив значение x величины $-\ln R_1$ мы можем сгенерировать Y как $x - \ln R_2$, где $R_2 \sim R[0, 1]$ и не зависит от R_1.
Таким образом, вектор $(-\ln R_1, -\ln(R_1R_2))$ будет иметь требуемое распределение.

Задача 2. Сгенерировать точку в единичном круге с помощью метода условных распределений .

Задача 3. Более сложная задача — сгенерировать случайную точку в 10-мерном шаре. Почему здесь нерационально использовать метод выбора с отклонением?
Указание: Объем n-мерного шара есть

$$V_n = \frac{\pi^{n/2}}{\Gamma(n/2 + 1)}R^n.$$

Подсказка: Если вам показалось, что задача слишком сложная, то попробуйте сферические координаты

2.2.3 Gibbs sampler

Казалось бы, мы разобрались с проблемой генерирования случайных величин и векторов. Однако, нередко мы сталкиваемся со случаем, когда явный вид распределения труден для вычисления, но известны условные плотности $f_{X_i|X_1,...,X_{i-1},X_{i+1},...X_n}(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$. Примеры такого рода задач будут приведены ниже. Оказывается, что это довольно эффективно можно сделать с помощью алгоритма Gibbs sampler (в русской википедии можно найти его под названием "гиббсовское семплирование", но это название не кажется мне удачным). Обоснование работы этого алгоритма напрямую связано с алгоритмом Метрополиса-Гастингса, разобранным ниже. Для упрощения восприятия эти рассуждения помещены в конец этого листочка.

Итак, нам нужно сгенерировать вектор $(X_1, ..., X_n)$, где мы знаем условные плотности. 1) Выберем некоторый начальный вектор $(X_{1,1}, ..., X_{1,n})$, который вообще может получиться из нашего распределения.

2) Выберем случайный индекс d из множества $\{1, ..., n\}$.
3) Положим $X_{2,i} = X_{1,i}$ при $i \neq d$. Величину $X_{2,d}$ сделаем случайной с плотностью $f_{X_d|X_1,...,X_{d-1},X_{d+1},...,X_n}(x_d|x_1, ..., x_{d-1}, x_{d+1}, ..., x_n)$ на основе $X_{1,i}$ при $i \neq d$.
4) Повторим процедуру 2)-3) для нового вектора и так далее.

Утверждается, что вектор $(X_{m,1}, ..., X_{m,n})$ сходится по распределению к требуемому вектору $(X_1, ..., X_n)$ при $m \to \infty$. Выбирая достаточно большое m, мы получим вектор практически с нужным распределением.

Пример 6. Предположим, что мы хотим взять наугад выборку из N точек в единичном круге, удаленных друг от друга не менее чем на расстояние r. Мы не способны выписать для данной задачи совместную плотность нашего вектора. Но условные плотности устроены довольно просто: если мы знаем все точки, кроме одной, то оставшаяся точка распределена равномерно на исходном круге за вычетом кругов радиуса r вокруг остальных точек.

Gibbs Sampler предлагает нам на начальном шаге взять какой-то набор подходящих точек, например, бросить первую точку как попало, вторую в оставшееся множество и так далее.
На каждом следующем шаге мы берем имеющийся набор $(x_1, ..., x_N)$, выбираем случайный индекс i а затем меняем x_i на точку, равномерно распределенную на множестве, являющемся разностью исходного круга и кругов радиуса r вокруг x_j, $j \neq i$. После достаточного числа итераций мы получим искомую выборку.

Вопрос 6. Почему полученный на начальном шаге начальный вектор еще не будет иметь нужного распределения?

Задача 4. Попробуйте реализовать эту процедуру на R или Python с N = 50, r = 0.1.

2.3 Алгоритм Метрополиса-Гастингса или как полезно владеть эргодической теоремой.

Этот параграф предназначен для тех, кто хочет сдавать эту тему на продвинутом уровне.

2.3.1 Описание алгоритма

В случае объектов более сложной структуры, мы можем столкнуться с тем, что задача их генерации напрямую достаточно трудоемка. Как, например, сгенерировать случайный связный граф на n вершинах, если мы даже слабо представляем сколько их?

Разберем более продвинутый алгоритм, называемый алгоритмом Метро'полиса-Гастингса (Николас Метрополис — один из основоположников применения методов Монте-Карло, о которых мы поговорим чуть позже).

Алгоритм базируется на построении марковской цепи, сходящейся к данному распределению. В данном разделе мы опишем сам алгоритм и методику его применения, а математическую подоплеку разберем в приложении. Возможно, вам полезнее начать именно с него.

Итак, пусть мы хотим построить случайную величину, вектор или другой случайный объект, у которого распределение $\pi(x)$ в дискретном случае или плотность $\pi(x)$ в непрерывном случае с точностью известно до множителя (например, если мы хотим сгенерировать равномерное распределение, то это $\pi(x)$ равно 1 с точностью до множителя). Предположим, что мы найдем эргодическую марковскую цепь X_n с переходной матрицей P (в дискретном случае) или переходными плотностями $P(x, y)$ (в непрерывном случае), имеющую эргодическое распределениеπ. Тогда задав произвольное начальное распределение, в силу эргодичности мы спустя достаточно большое число итераций получим нужно нам распределение π. Как же найти такую марковскую цепь?

Достаточным условием для эргодического распределения является условие обратимости

$$\pi(x)P(x, y) = \pi(y)P(y, x).$$

Давайте предположим, что у нас есть некоторая переходная матрица (плотность) Q, возможно не являющаяся требуемой. Пусть для заданных x, y

$$\pi(x)Q(x, y) > \pi(y)Q(y, x).$$

Давайте попробуем исправить ее так, чтобы стало выполняться условие обратимости. Для этого надо уменьшить $Q(x, y)$. Добавим к переходной вероятности $Q(x, y)$ некоторый множитель $\alpha(x, y)$, чтобы тождество выполнялось (иначе говоря, раньше вы выполняли переход из x в y с вероятностью (плотностью)Q(x,y), а теперь будем с вероятностью $Q(x, y)\alpha(x, y)$). Очевидно,

$$\alpha(x, y) = \frac{\pi(y)Q(y, x)}{Q(x, y)\pi(x)}.$$

Положим $\alpha(y, x)$ равным 1. В случае же, когда

$$\pi(x)Q(x, y) < \pi(y)Q(y, x),$$

имеет смысл взять $\alpha(x, y) = 1$,

$$\alpha(y, x) = \frac{\pi(x)Q(x, y)}{Q(y, x)\pi(y)}.$$

Новая матрица перехода (переходная плотность) $P(x, y) = Q(x, y)\alpha(x, y)$ при $x \neq y$, дополненная в дискретном случае соответствующим $P(x, x)$ (то есть тем, который дополнит вероятности до суммы 1), будет искомой цепью. Таким образом, наш алгоритм приобретает вид:

0) Запасемся какой-либо матрицей перехода (переходной плотностью) Q.

1) Положим $X_0 = x_0$, где x_0 — произвольная случайная величина (вектор, объект), ко-

торый может получиться при нашем распределении.
2) Пусть на n-ом шаге у нас есть $X_n = x_n$. Сделаем шаг марковской цепью с матрицей перехода (переходной плотностью) Q из x_n. Иначе говоря, в дискретном случае разыграем величину с распределением $P(Y_n = y) = Q(x_n, y)$, в непрерывном — с аналогичной плотностью. Получим случайную величину $Y_n = y$.
3) Найдем $\alpha(x_n, y)$ по приведенным выше формулам, разыграем бернуллиевскую случайную величину Z_n, равную 1 с вероятностью $\alpha(x_n, y)$ и 0 с вероятностью $1-\alpha(x_n, y)$. Если $Z_n = 1$, то положим $X_{n+1} = y$, иначе $X_{n+1} = x_n$. Генерировать Z_n можно как в первом примере.
4) Повторим этапы 2)-4).
Спустя достаточно большое число итераций n, мы получим нужный нам случайный объект.

2.3.2 Частные случаи

Остается ответить на вопрос "Как разжиться исходной матрицей Q?". С одной стороны, алгоритм будет работать при любом выборе Q' при котором из любого состояния можно попасть в любое. С другой стороны, если $\alpha(x, y)$ будут маленькими, то мы подолгу будем застревать в состояниях и скорость сходимости будет малой. Желательно, чтобы $\alpha(x, y)$ были не меньше 0.5.
Приведем примеры выбора Q:
А) В дискретном случае удобным методом является следующий. Введем на состояниях (т.е. значениях нашей величины, вектора, объекта) соотношение соседства так, чтобы любое состояние было связано с любым цепочкой соседей. Обозначим через $N(x)$ число соседей у вершины x. Положим $Q(x, y) = 1/N(x)$, если y — сосед x и 0 иначе. Тогда алгоритм приобретет следующий вид:
1) Выберем какое-то состояние x_0 и положим $X_0 = x_0$.
2) Пусть на n-ом шаге $X_n = x_n$. Выберем произвольного соседа y вершины x_n.
3) Положим $\alpha(x, y) = \min\left(\frac{\pi(x)N(y)}{\pi(y)N(x)}, 1\right)$. Сгенерируем случайную величину $R \sim R[0, 1]$. Если $R < \alpha(x, y)$, то $X_{n+1} = y$, иначе $X_{n+1} = x_n$.
4) Повторим процедуру 2)-4).
После n итераций, где n — достаточно велико, мы получим требуемый объект.

Пример 7. Представим себе, что мы, например, хотим сгенерировать наугад выбранную перестановку из N элементов, обладающую свойством, скажем, $\sum_{k=1}^{N} k\sigma(k) > C$. Тогда мы хорошо представляем соотношение между p_i (вероятностями того, что это будет именно i-ая перестановка) — они все равны. При этом мы слабо представляем количество и структуру рассматриваемых перестановок.
Возьмем отношение соседства, назвав соседними перестановки, отличающиеся транспозицией.
1) Выберем любую такую перестановку, например, тождественную, заведомо удовлетворяющую условию.

2) На каждом следующем шаге выберем одну из соседних перестановок j к имеющейся i. Это можно сделать просто выбирая произвольную транспозицию и умножая ее на нашу перестановку. Если полученная перестановка будет удовлетворять условию $\sum_{k=1}^{N} k\sigma(k) > C$, то оставляем ее, а иначе выбираем новую транспозицию и так далее.
3) Подсчитаем $\alpha(i,j)$ как $\min\left(\frac{N(j)}{N(i)}, 1\right)$, где $N(k)$ — число соседних с k перестановок. Мы можем подсчитать $N(j)$ для конкретной j довольно быстро, просто перебрав все возможные транспозиции (их не так много — C_N^2 штук).
Сгенерируем случайную величину $R \sim R[0,1]$. Если $R < \alpha(i,j)$, то переходим в j, иначе остаемся в i.
4) Повторяем процедуру 2)-4).

Вопрос 7. Почему в рамках прошлого примера, вообще говоря, генерировать перестановки до тех пор, пока не получим перестановку нужного вида (сродни Acceptance-Rejection методу) — плохая идея?

Задача 5. Попробуйте реализовать эту процедуру на R или Python с $N = 20$, $C = 200$, сделав 20 итераций.

B)Случайное блуждание по состояниям.
Пусть наши состояния — числа или векторы. Выберем какое-нибудь распределение $F(x)$ на наших числах или векторах. Будем использовать матрицу Q, соответствующую сдвигу на случайный вектор с распределением F. Тогда алгоритм приобретет вид:
1) Выберем любое число (вектор) в качестве X_0
2) Пусть на n-ом шаге $X_n = x_n$. Сгенерируем случайную величину (вектор) Z_n с распределением F. При $Z_n = z_n$ положим $y = x_n + z_n$.
3) Подсчитаем $\alpha(x_n, y)$ как $\min\left(\frac{\pi(y)F(-z_n)}{\pi(x_n)F(z_n)}, 1\right)$ (особенно удобно, если F симметрична, то есть $F(z) = F(-z)$). Сгенерируем случайную величину $R \sim R[0,1]$. Если $R < \alpha(x_n, y)$, то переходим в $X_{n+1} = y$, иначе $X_{n+1} = x_n$.
4) Повторяем процедуру 2)-4).

C) Независимый выбор состояния.
Будем брать $q(x,y) = g(y)$, где g — некоторая плотность. Тогда
1) Выберем любое число или вектор.
2) Пусть на n-ом шаге $X_n = x_n$. Сгенерируем случайную величину (вектор) с распределением G.
3) Подсчитаем $\alpha(x_n, y)$ как $\min\left(\frac{\pi(y)g(x_n)}{\pi(x_n)g(y)}, 1\right)$. Сгенерируем случайную величину $R \sim R[0,1]$. Если $R < \alpha(x_n, y)$, то переходим в $X_{n+1} = y$, иначе $X_{n+1} = x_n$.
4) Повторяем процедуру 2)-4).

D) Metropolis-Hastings Acceptance-Rejection Method.
Пусть мы генерируем случайную величину (вектор) с плотностью, пропорциональной $f(x)$. Допустим, что мы умеем генерировать случайную величину с плотностью $g(x)$ и пусть c — некоторая константа. В Acceptance-Rejection Method мы требовали, чтобы

$f(x) < cg(x)$, а теперь мы допускаем, что это верно не при всех x.
Используем метод Метрополиса-Гастингса с независимым выбором состояния, соответствующим обычному Rejection Method. То есть:
1) Выберем любое число (вектор) в качестве X_0
2) Пусть на n-ом шаге $X_n = x_n$. Сгенерируем Y_n следующим образом:
2а) Выберем Z с плотностью $g(x)$. Разыграем $R \sim R[0,1]$.
2б) Если $R_1 < f(Z)/(cg(Z))$, то $Y_n = Z$, иначе повторим эксперимент 2а), 2б).
3) При $Y_n = y$ найдем $\alpha(x_n, y)$ по формуле

$$\alpha(x_n, y) = \begin{cases} 1, & f(x_n) < cg(x_n), \\ \frac{cg(x_n)}{f(x_n)}, & f(x_n) \geq cg(x_n),\ f(y) < cg(y), \\ \min\left(\frac{f(y)g(x_n)}{f(x_n)g(y)}, 1\right), & f(x_n) \geq cg(x_n),\ f(y) \geq cg(y). \end{cases}$$

Сгенерируем $R_2 \sim R[0,1]$. При $R_2 < \alpha(x_n, y)$ положим $X_{n+1} = y$, иначе $X_{n+1} = x_n$.
4) Повторяем процедуру 2)-4)

Вопрос 8. Почему такой выбор α соответствует алгоритму Метрополиса-Гастингса?

Задача 6. Построить выборку из $\mathcal{N}(\mu, \Sigma^2)$, $\mu = (1, 2)$,

$$\Sigma^2 = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1. \end{pmatrix}.$$

Напомню, что двумерное нормальное распределение имеет плотность

$$f(\vec{x}) = \frac{1}{\sqrt{2\pi \det \Sigma^2}} \exp\left(-\frac{1}{2}(\vec{x} - \mu)^t \Sigma^{-2} (\vec{x} - \mu)\right).$$

а) Использовать метод B), в качестве z_n использовать случайный равномерный вектор из $[-1, 1]^2$.
б) Использовать метод D) с $c = 0.9$ и g — двумерным нормальным распределением с вектором средних (1, 2) и матрицей ковариации

$$\Sigma^2 = \begin{pmatrix} 2 & 0 \\ 0 & 2. \end{pmatrix}.$$

2.4 Случайности против неслучайных задач

2.4.1 Метод Монте-Карло

Итак, мы набрались знаний о моделировании случайных объектов. Давайте теперь посмотрим на то, как это применяется в тех разделах математики, которые не связаны с теорией вероятностей. В ряде приложений, не имеющих непосредственного отношения к случайности, бывает удобно получать приближенный ответ с помощью вероятностных методов. Такого рода методы носят название методы Монте-Карло (в честь одноименного

района Монако, знаменитого своими игорными заведениями). Классическим примером такого рода задачи служит приближенный подсчет интеграла.

Итак, предположим, что нам нужно подсчитать приближенно интеграл $I = \int_a^b f(x)dx$, где f — заданная функция.
Наиболее простым методом является метод прямоугольников, предлагающий заменить интеграл на сумму

$$\sum_{k=1}^{n} f(\tilde{x}_k)(x_{k+1} - x_k) = \frac{(b-a)}{n} \sum_{k=1}^{n} f(\tilde{x}_k),$$

где x_k — разбиение отрезка $[a, b]$ на равные части, а $\tilde{x}_k = (x_{k+1} + x_k)/2$ — отмеченные точки. Погрешность такого метода $O(1/n)$ для любой гладкой f и $O(1/n^2)$ для дважды гладкой f.

Метод Монте-Карло предлагает взамен взять

$$\widehat{I} = \frac{b-a}{n} \sum_{k=1}^{n} f(X_k),$$

где X_k — н.о.р. $R[a, b]$. Применяя к $f(X_k)$ ЗБЧ, мы видим, что $\widehat{I}$ сходится к

$$(b-a)Ef(X) = (b-a) \int_a^b f(x) \frac{1}{b-a} dx = I,$$

причем в силу ЦПТ $\hat{I}$ будет асимптотически нормальной I,

$$\sqrt{n} \frac{\hat{I} - I}{\sigma} \xrightarrow{d} Z \sim \mathcal{N}(0, 1),$$

где $\sigma^2 = (b-a)^2 Df(X)$. Таким образом, погрешность метода Монте-Карло будет порядка $O(n^{-1/2})$.

На данной задаче мы не видим особенных преимуществ замены детерминированных точек $\tilde{x}_k$ на случайные X_k — точки попадают более разреженно и вместо возможной погрешности $O(n^{-2})$ мы получаем погрешность $O(n^{-1/2})$, да еще и случайную.

Однако, во-первых, мы не налагаем никаких условий на гладкость или даже непрерывность f, нам нужна лишь ЦПТ, то есть интегрируемость квадрата нашей функции по Лебегу. Во-вторых, выгода становится заметной, когда мы рассматриваем многомерную задачу

$$\int_A f(x_1, ..., x_k)dx_1...dx_k$$

Тогда метод прямоугольников для той же погрешности $O^(n^{-2})$ заставит нас взять разбиение на n^k кубиков, т.е. сделать порядка $O(n^k)$ операций суммирования. В то же время при оценивании погрешности метода Монте-Карло мы не опираемся на размерность и для той же погрешности $O(n^{-2})$ нам понадобится $O(n^4)$ операций суммирования. При больших размерностях выгода налицо.

Пример 8. Оценим

$$\int_{0<x_1<x_2<\ldots<x_6} \prod_{i=1}^{n} x_i^{a_i} e^{-x_6} dx_1 \ldots dx_6.$$

Аналогично примеру 3, мы можем заметить, что $e^{-x_6} I_{0<x_1<\ldots<x_6}$ — плотность вектора, распределенного также как $(-\ln R_1, -\ln R_1 R_2, \ldots, -\ln R_1 \ldots R_6)$. Таким образом, величина

$$\frac{1}{n} \sum_{i=0}^{n-1} \prod_{j=1}^{6} (-\ln \prod_{k=1}^{j} R_{6i+k})^{a_j}$$

будет давать погрешность порядка $n^{-1/2}$. Численное интегрирование с помощью приближения простой фигурой давало бы погрешность только порядка $n^{-1/3}$.

Недостатком метода Монте-Карло является то, что случайные точки могут оказаться достаточно разреженными в какой-либо области. Поэтому зачастую метод Монте-Карло для интегрирования используют вместе с расслоением — сперва область разбивают на несколько подобластей, а затем в каждой из них генерируют выборку. Это гарантирует, что точки не будут концентрироваться в отдельной области, а будут относительно равномерно распределяться по отдельным блокам.

2.4.2 Метод отжига или как не застревать в локальных экстремумах при численной максимизации

Этот параграф предназначен для тех, кто хочет сдавать эту тему на продвинутом уровне.

Другой неслучайной задачей, для которой удобно использовать случайные методы является нахождение максимума функции. Мы рассмотрим метод, автором которого является Николас Метрополис, называемый методом имитации отжига. Наиболее простой невероятностный метод нахождения максимума функционала f заключается в следующем: стартуя с некоторой начальной точки x_0, по определенному правилу $F(i, x)$ на i-м шаге от X_i переходить к $y = F(i, x_i)$, если $f(y) \geq f(x_i)$ или оставаться в x_i иначе.

Однако, если попадем при этом в локальный максимум, то может оказаться затруднительно выбраться из него. При использовании метода имитации отжига мы задаем также некоторую функцию температуры $T : \mathbb{N} \to \mathbb{R}$. При высокой температуре мы будем двигаться более активно и можем уйти из точки x_i в y даже если $f(y) < f(x_i)$. Итак, формальная запись алгоритма такова:
1) Находим $s = F(i, s_i)$.
2) Если $f(s) < f(s_i)$, то берем $s_{i+1} = s_i$,
3) Если $f(s) \geq f(s_i)$, то берем $s_{i+1} = s$ с вероятностью $p(T(i), s_i, s)$.
Для использования метода нам нужно задать функцию температуры T и функцию p. Обычно в качестве последней используют функцию $e^{-(f(s_i)-f(s))/T(i)}$.

Этот метод позволяет нам не застревать в локальных минимумах, вылетая из них засчет температуры. Функцию температуры $T(i)$ берут убывающей, остановка алгоритма проис-

ходит, когда температура T опустится ниже какой-то заданной границы. Увеличивая скорость убывания T, мы снижаем устойчивость оценки к попаданиям в локальные минимумы, но сокращаем количество итераций. Популярные температуры $T(i) = T(0)/\ln(1+i)$ (больцмановский отжиг), $T(i) = T(0)/(1+i)$ — отжиг Коши.
Для случая дискретных множествах часто вводят отношение соседства (т.е. каждой точке соспоставляют множество соседних с ней) и в качестве $F(s_i)$ берут случайного соседа.

Пример 9. Решим задачу расстановки на доске 40×40 40 ферзей так, чтобы они не били друг друга. Будем рассматривать в качестве пространства расстановки, в которых никакие два ферзя не стоят на одной вертикали (горизонтали), т.е. перестановки σ. Ферзей мы ставим в точки $(i, \sigma(i))$. В качестве функции $f(\cdot)$ расстановки возьмем количество конфликтов (бьющих друг друга фигур) при данной расстановке ферзей. Соседними расстановками назовем те две, которые получаются путем замены двух ферзей (i, j), (k, l) на ферзей (i, l), (k, j) (т.е. соответствующая перестановка отличается от нашей транспозицией).
В качестве $p(T(i), s_i, s)$ используем $e^{-(f(s_i)-f(s))/(T(i)f(s_i))}$.
На контрольных тестах такой метод за 2000 итераций находил подходящую расстановку.

Задача 7. Реализовать описанную задачу на R или Python.

2.5 Приложения

Следующие параграфы требуются для того, чтобы не оставлять белых пятен в математическом представлении о происходящем.

2.5.1 О методе Метрополиса-Гастингса с математической точки зрения: Дискретный случай

Более подробно про цепи Маркова можно прочитать, например, в главе XV "Марковские цепи" тома 1 книги Феллера "Введение в теорию вероятностей и ее приложения".

Разберем подробно случай конечных дискретных цепей Маркова, а потом укажем аналогичные результаты для остальных случаев.

Определение 1. Случайная последовательность X_n, принимающая значения из конечного или счетного множества S, называется марковской цепью, если

$$P(X_n = i_n | X_{n-1} = i_{n-1}, ..., X_0 = i_0) = P(X_n = i_n | X_{n-1} = i_{n-1})$$

при любых $i_0, ..., i_n \in S$. Если к тому же эта вероятность не зависит от n, то цепь называют однородной.

Содержательно, условие означает, что следующее состояние X_n зависит от прошлого только через X_{n-1}, то есть если X_{n-1} фиксированно, то X_n не зависит от $X_0, ..., X_{n-2}$.

Будем считать, что $S = \{1, ..., N\}$ (это вопрос просто нумерации состояний). Тогда матрица $P = (p_{i,j})$ называется переходной матрицей цепи. Вместе с вектором $p = (P(X_0 = 1), ..., P(X_0 = N))$, , она полностью определяет распределение цепи. В частности, легко убедиться, что

$$(P(X_n = 1), ..., P(X_n = N)) = pP^n.$$

Определение 2. Говорят, что из состояния i следует состояние j, если при каком-то n $P(X_n = j|X_0 = i) > 0$ (иначе говоря, возможно, находясь в состоянии i, через некоторое время оказаться в j). Состояния i, j сообщаются, если из i следует j, а из j следует i. Цепь называется неразложимой, если все состояния в ней сообщаются.

Определение 3. Неразложимая цепь называется непериодической, если наибольший общий делитель длин замкнутых путей в этой цепи равен 1. Иначе говоря, есть несколько путей из каких-либо состояний в себя, длины которых взаимно просты.

Оказывается, что справедлива следующая эргодическая теорема:

Теорема 1. *Если конечная цепь Маркова является неразложимой и непериодической, то при любом начальном распределении, вероятности $P(X_n = i)$ сходятся к некоторым π_i, $i = 1, ..., N$ при $n \to \infty$. При этом $\vec{\pi} = (\pi_1, ..., \pi_N)$ является единственным решением системы уравнений $\pi P = \pi$.*
Вектор $\vec{\pi}$ называется эргодическим распределением.

Для нас эта теорема играет ключевую роль. Она говорит, что если мы найдем такую переходную матрицу П, являющуюся неразложимой и непериодической, что π будет ее стационарным распределением (то есть $\pi P = \pi$), то какое распределение не задай в качестве начального, спустя достаточное количество шагов мы получим распределение, близкое к π. Более того, скорость сходимости достаточно велика, можно показать, что вероятности приближаются к эргодическому распределению с экспоненциальной скоростью.

Как же искать такую матрицу P? Давайте сначала найдем другое условие, достаточное для того, чтобы π было стационарным распределением. Пусть X_n — цепь Маркова, причем начальное распределение $p = (p_1, ..., p_N)$ стационарно. Тогда $P(X_n = i) = p_i$ в силу указанного выше соотношения $(P(X_n = 1), ..., P(X_n = N)) = pP^n$. Рассмотрим последовательность $X_1, X_2, ...$ в обратном порядке. Это также будет марковская цепь с матрицей перехода

$$q_{i,j} = P(X_1 = j|X_2 = i) = \frac{P(X_2 = i, X_1 = j)}{P(X_2 = i)} = \frac{P_{j,i}\pi_j}{\pi_i}.$$

Назовем цепь обратимой, если

$$\pi_i P_{i,j} = \pi_j P_{j,i}$$

при всех i, j, то есть если $q_{i,j} = p_{i,j}$. Оказывается, что справедлива следующая теорема:

Теорема 2. *Если для некоторого распределения p выполнено условие сбалансированности*

$$p_i P_{i,j} = p_j P_{j,i},$$

то цепь обратима, причем p является ее стационарным распределением.

Доказательство. Действительно, рассмотрим цепь с матрицей перехода P и начальным распределением p. Тогда

$$(pP)_j = \sum_{i=1}^{N} p_i P_{i,j} = \sum_{i=1}^{N} p_j P_{j,i} = p_j,$$

поскольку сумма матрицы перехода по строке равна 1. Значит, p — стационарное распределение и цепь обратима. □

Значит, если мы подберем для заданного распределения p такую матрицу перехода P, что выполнено условие сбалансированности и P будет неразложима и непериодична, то марковская цепь с матрицей перехода P с течением времени будет сходиться по распределению к стационарному распределению. Именно это и предлагает делать алгоритм Метрополиса-Гастингса. Заметим, что наличие $\alpha(x, y)$ позволяет повторно оставаться в том же состоянии, что и на прошлом шаге, что делает цепь непериодической. При этом неразложимость исходной цепи Q влечет неразложимость и цепи из алгоритма Метрополиса-Гастингса.

2.5.2 О методе Метрополиса-Гастингса с математической точки зрения: Непрерывное множество состояний

Аналогичную конструкцию можно построить в непрерывном случае.

Определение 4. Марковским процессом X_n с дискретным временем и множеством состояний S (теперь уже не обязательно конечным или счетным) будем называть такую последовательность, что

$$P(X_n \in A | X_{n-1} = x_{n-1}, ..., X_0 = x_0) = P(X_n \in A | X_{n-1} = x_{n-1}).$$

Здесь $P(X_n \in A | X_{n-1} = x_{n-1}) = E(I_{X_n \in A} | X_{n-1} = x_{n-1})$ — условное математическое ожидание. Функция $P(X_n \in A | X_{n-1} = x_{n-1}) = p(x_{n-1}, A, n)$ называется переходной функцией, если $S \subset \mathbb{R}$ и существует плотность $f_{X_n|X_{n-1}}(y|x)$, то она называется переходной плотностью $P_{x,y,n}$. Если переходная функция (плотность) не зависит от n, то цепь называется однородной.

Рассмотрим стационарное распределение с плотностью p, то есть

$$\int_x p(x) P(x, y) dx = p(y).$$

Обратимой назовем цепь, для которой

$$P(x,y)p(x) = P(y,x)p(y).$$

Опять же условие сбалансированности влечет за собой обратимость цепи и стационарность распределения.

Остается сформулировать эргодическую теорему для цепей Маркова с непрерывным множеством состояний. Например, справедлива следующая теорема:

Теорема 3. *Пусть $P(x,y)$ — непрерывная положительная переходная плотность, а множество состояний S есть отрезок. Тогда X_n сходится к единственному стационарному распределению.*

Тем самым, метод Метрополиса-Гастингса будет работать и в случае марковских процессов с континуальным множеством состояний. Общая теория позволяет работать и в неабсолютно-непрерывном случае, но это останется вне нашего обзора.

2.5.3 Gibbs sampler как частный случай алгоритма Метрополиса-Гастингса

Обсудим также почему алгоритм Gibbs sampler укладывается в изложенную теорию. Действительно, в этом случае мы рассматриваем цепь с переходной плотностью из $\vec{x} = (x_1, ..., x_n)$ в $\vec{y} = (x_1, .., x_{k-1}, z, x_{k+1}, .., x_n)$

$$P(\vec{x},\vec{y}) = \frac{1}{n} f_{X_k|X_1,..,X_{k-1},X_{k+1},...X_n}(z|x_1,..,x_{k-1},x_{k+1},...,x_n).$$

Проверим, что она удовлетворяет условию сбалансированности с нужной нам $p(\vec{x}) = f_{X_1,...,X_n}(\vec{x})$. Действительно,

$$P(\vec{x},\vec{y})p(\vec{x}) = \frac{1}{n} \frac{f_{X_1,..,X_n}(x_1,..,x_{k-1},z,x_{k+1},...,x_n)}{f_{X_1,..,X_{k-1},X_{k+1},...,X_n}(x_1,...,x_{k-1},x_{k+1},...,x_n)} f_{X_1,...,X_n}(x_1,...,x_n).$$

При этом $P(\vec{y},\vec{x})p(\vec{y})$ даст ту же формулу с точностью до перестановки множителей. Следовательно, построенная цепь является сбалансированной с нужной плотностью.

2.6 Ответы на вопросы

Этот раздел для тех, кто подумал над вопросами, но не смог ответить или остался не вполне уверен в ответе.

Ответ 1. Как доказать, что $X = F^{-1}(Y)$, где $Y \sim R[0,1]$, имеет функцию распределения F?

Запишем

$$P(X \le x) = P(F^{-1}(Y) \le x) = P(Y \le F(x)) = F(x),$$

где последнее тождество следует из того, что $Y \sim R[0,1]$. Тот факт, что $\{y : F^{-1}(y) \le x\} = \{y : y \le F(x)\}$ напрямую вытекает из определения обратной функции.

Ответ 2. Как сгенерировать величину, имеющую ф.р.

$$F_X(x) = \begin{cases} 0, & x < 0, \\ (2 - e^{-x})/2, & x \ge 0. \end{cases}$$

По сути задача равносильна нахождению функции, обратной к заданной. Это

$$G(y) = \begin{cases} 0, & y \in [0, 1/2], \\ -\ln(2 - 2y), & y \in [1/2, 1]. \end{cases}$$

Значит сгенерировать X можно с помощью формулы

$$\begin{cases} -\ln(2 - 2Y), & Y > 1/2, \\ 0, Y \le 1/2 \end{cases},$$

где $Y \sim R[0,1]$.

Ответ 3. • Почему метод выбора с отклонением в дискретном случае действительно дает величину с нужным распределением?

Положим A_i — событие, что i-ое генерирование случайной величины Y выдаст нам удачный результат (то есть не случится отклонения сгенерированного значения Y). Тогда

$$\mathbf{P}(Y = x, A_1) \times \frac{f(x)}{cg(x)} = \frac{f(x)}{c}, \quad \mathbf{P}(A_1) = \frac{1}{c}\sum_x f(x) = \frac{1}{c}, \quad \mathbf{P}(Y = x | A_1) = f(x).$$

Аналогично вероятность того, что алгоритм завершится на k попытке на числе x при условии, что не завершился до этого есть

$$\mathbf{P}(X = x, A_k | \overline{A}_{k-1}, ..., \overline{A}_1) = \frac{f(x)}{c}, \quad \mathbf{P}(X = x | A_k \overline{A}_{k-1}, ..., \overline{A}_1) = f(x).$$

Следовательно,

$$\mathbf{P}(X=x)=\sum_{k=1}^{\infty}\mathbf{P}(X=x,A_k,\overline{A}_{k-1},...,\overline{A}_1)=\sum_{k=1}^{\infty}\mathbf{P}(X=x|A_k\overline{A}_{k-1},...,\overline{A}_1)\times$$
$$\mathbf{P}(A_k\overline{A}_{k-1},...,\overline{A}_1)=f(x)\sum_{k=1}^{\infty}\mathbf{P}(A_k\overline{A}_{k-1},...,\overline{A}_1)=f(x).$$

- Почему метод выбора с отклонением в абсолютно-неперерывном случае действительно дает величину с нужной плотностью?

  Рассуждения полностью аналогичны, но вместо $\mathbf{P}(X=x)$ мы рассматриваем $\mathbf{P}(X\in[x,x+\delta))/\delta$, где δ устремится к нулю. Указанное выражение стремится к плотности $f(x)$. Остается записать формулы

$$\lim_{\delta\to 0}\frac{1}{\delta}\mathbf{P}(X\in[x,x+\delta)|A_k,\overline{A}_{k-1}...,\overline{A}_1)=f(x),$$
$$\frac{\mathbf{P}(X\in[x,x+\delta))}{\delta}=\frac{1}{\delta}\sum_{k=1}^{\infty}\mathbf{P}(X\in[x,x+\delta)|A_k\overline{A}_{k-1}...\overline{A}_1).$$

  Здесь есть нюанс, связанный с необходимостью поменять предел и ряд, который впрочем не представляет особой сложности, поскольку событие $\mathbf{P}(\overline{A}_{k-1}...\overline{A}_1)$ имеет вероятность, стремящуюся к нулю при $k\to\infty$.

Ответ 4. Какое c взять для построения величины с плотностью $f(x)=20\ x(1-x)^3$, $0<x<1$ с помощью Acceptance-Rejection Method на основе равномерной плотности $g(x)$?

Нам нужно такое c, что $f(x)\le cg(x)$. Возьмем

$$c=\sup\frac{f(x)}{g(x)}=\sup 20x(1-x)^3.$$

Дифференцируя $f(x)=x(1-x)^3$ получим $f'(x)=(1-x)^3-3x(1-x)^2=(1-4x)(1-x)^2$. Производная обращается в 0 на отрезке при $x=1/4$, $x=1$, при этом при $x<1/4$ функция возрастает, а после убывает. Значит максимум функции будет при $x=1/4$ и

$$c=20\cdot\frac{1}{4}\cdot\left(1-\frac{3}{4}\right)^3=\frac{135}{64}.$$

Ответ 5. Почему генерировать случайный вектор, равномерно распределенный на единичном круге с помощью метода выбора с отклонением можно просто бросая точку в квадрат $[-1,1]\times[-1,1]$, пока она не попадет в круг?

Плотность $f(x,y)$ равномерного брошенного в круг вектора есть $1/\pi$ при $x^2+y^2\le 1$ и 0 иначе. Плотность $g(x,y)$ равномерно брошенной в квадрат точки — $1/4$ при $|x|,|y|\le 1$ и 0 иначе. Тем самым, $f(x,y)\le 4g(x,y)/\pi$. Аналогично примеру с тремя бросками монеты, это приводит алгоритм к описанной схеме — перегенерировать вектор, пока он не попадет

в круг.

Ответ 6. Почему, чтобы сгенерировать случайный (равномерный) набор из N точек в единичном круге, удаленных друг от друга по меньшей мере на r, нельзя бросать эти точки одна за одной, запрещая каждой следующей попадать на расстояние меньше чем r от первой из них.

Случайный набор, удовлетворяющий условию, очевидно симметрично зависим, то есть первая точка распределена также как вторая и так далее. Если бы указанная процедура приводила к требуемой раскладке, то получилось бы, что раз первая точка равномерно распределена на круге, то и все остальные тоже. Однако нетрудно видеть, что уже вторая точка брошенная так точка неравномерно распределена. Действительно, представьте, что $r = 1$. Тогда вторая точка очень редко будет попадать в маленькую окрестность центра круга, ведь для этого надо, чтобы а) первая точка попала на самую границу б) вторая точка попала в искомую окрестность. Между тем, в окрестность точки на границе вторая точка будет попадать куда чаще, для этого достаточно, чтобы а) первая точка попала в множество удаленных от нашей окрестности на расстояние более чем один точек (это более половины круга), б) вторая точка попала в искомую окрестность. В первом случае вероятность сильно меньше, чем во втором при одинаковых площадях окрестностей. Это противоречит равномерности.

Ответ 7. Всего перестановок $N!$. Скажем, для $N = 20$ это больше 2^{1018}. При этом при больших C множество искомых перестановок может быть достаточно малым, например, состоять из нескольких перестановок. Тем самым, нам придется проводить огромное число экспериментов, прежде чем выпадет хоть одна искомая перестановка. Между тем для успешной работы приведенного алгоритма Метрополиса-Гастингса вполне достаточно порядка N итераций.

Как же удается за такое маленькое число шагов сделать выбор из такого большого множества? Представим себе, что мы просто генерировали бы случайную перестановку тем же методом. На каждом шаге алгоритма мы выбираем одну из потенциальных $N(N-1)/2$ перестановок, за N шагов в качестве потенциальных кандидатов нашего выбора побывают все $N!$ штук.

Ответ 8. Почему в Acceptance-Rejection Metropolis-Hastings методе следует выбирать

$$\alpha(x_n, y) = \begin{cases} 1, & f(x_n) < cg(x_n), \\ \frac{cg(x_n)}{f(x_n)}, & f(x_n) \geq cg(x_n),\ f(y) < cg(y), \\ \min\left(\frac{f(y)g(x_n)}{f(x_n)g(y)}, 1\right), & f(x_n) \geq cg(x_n),\ f(y) \geq cg(y). \end{cases}$$

Давайте сперва разберемся с тем, как распределено Z. Пусть $C = \{x : f(x) < cg(x)\}$. Исходя из того, как работает Acceptance-Rejection метод, $f_Z(z)$ с точностью до константы имеет вид

$$\begin{cases} f(z), z \in C, \\ cg(z), z \notin C. \end{cases}$$

Эта величина означает для нас $Q(x,y)$. Значит,

$$\alpha(x,y) = \min\left(\frac{\pi(x)Q(x,y)}{Q(y,x)\pi(y)}, 1\right) = \begin{cases} \frac{f(x)f(y)}{f(y)f(x)}, & x,y \in C, \\ \min\left(\frac{f(x)cg(y)}{f(y)f(x)}, 1\right), & x \in C, y \notin C, \\ \frac{f(x)f(y)}{cg(x)f(y)}, & x \notin C, y \in C, \\ \min\left(\frac{f(x)cg(y)}{f(y)cg(x)}, 1\right), & x \notin C, y \notin C. \end{cases}.$$