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

Как генерировать случайные величины.

В рамках первого 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.

Начнем мы с простого факта, который, вероятно, известен большей части из вас: если $X \sim R[0,1]$, то $F^{-1}(X)$ имеет ф.р. $F(x)$. Это утверждение справедливо для любой (в том числе и дискретной) функции распределения F.

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, т.е. $G(x) = \inf\{u : F(u) \geq x\}$.

Пример 1. Чтобы получить бернуллиевскую величину с параметром p, мы можем взять 0 при $R < 1-p$ и 1 при $R \geq 1-p$, где $R \sim R[0,1]$. Иначе говоря, $I_{R \geq 1-p}$ будет нужной нам величиной.

Пример 2. Чтобы получить величину с ф.р. $F_X = x^2$, $x \in [0,1]$, мы можем брать $\sqrt{R}$, $R \sim R[0,1]$.

Вопрос 2. А как сгенерировать величину, имеющую ф.р. $0, x < 0$, $(2 - e^{-x})/2$, $x \geq 0$?

Этот метод называется методом обратной функции. В целом он решает большую часть проблем, связанным с генерированием случайных величин, однако на практике может оказаться труднозатратным считать функцию распределения и еще более затратно обратную функцию. В таком случае можно использовать другие методы, например, так называемый метод отклонения (Rejection method).

Допустим X имеет плотность $f(x)$, а мы умеем строить величину с плотностью $g(x)$, причем $f(x)/g(x) \leq c$. Тогда можно сгенерировать величину с плотностью $g(x)$ и с вероятностью $f(x)/(cg(x))$ взять ее в качестве X, а в противном случае отвергнуть наше значение и сгенерировать новую случайную величину.

Вопрос 3. Почему полученный метод действительно дает величину с нужной плотностью?

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

Вопрос 4. Какое c можно взять в этом случае?

Метод отклонения предлагает сгенерировать величины R_1, R_2 и если $R_2 < f(R_1)/c$ брать R_1, а если нет, то генерировать R_3 и аналогичную операцию проводить с ней и так далее.

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 достаточность иметь датчик независимых $R[0,1]$ величин. А как же моделировать вектор, чьи компоненты зависимы? Здесь нам поможет условное распределение.

Идея такова — для получения вектора $(X_1, ..., X_n)$ мы сначала сгенерируем X_1, затем найдем условную ф.р. $F_{X_2|X_1}(x|y) = P(X_2 \leq x | X_1 = y)$, подставим вместо y полученное нами значение X_1 и сгенерируем X_2. Затем аналогичным образом сгенерируем X_3 с помощью $F_{X_3|X_2,X_1}$ и так далее.

Пример 4. Смоделируем вектор с совместной плотностью $f_{X,Y}(x,y) = e^{-y}$ при $0 < x < y$, 0 иначе. Найдем сперва плотность $f_X(x)$ и ф.р. $F_X(x)$. При $x > 0$

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

Следовательно, X генерируется как $-\ln(1 - R_1)$, где $R_1 \sim R[0,1]$ (на самом деле можно брать $-\ln R_1$, поскольку $1 - R_1 \stackrel{d}{=} R_1$).

Условная плотность $Y|X$ при $0 < x < y$ равна

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

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 $-\ln R_1, -\ln(R_1 R_2)$ будет иметь требуемое распределение.
Упражнение 1. Сгенерировать точку в единичном круге.
Упражнение 2. Более сложная задача — сгенерировать случайную точку в 10-мерном шаре. Почему нерационален такой метод: бросим точку в описанный около этого шара кубик 2) если точка не попала в шар, сгенерируем ее еще раз?
Объем n-мерного шара есть

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

Подсказка (светло-серым шрифтом): Если вам показалось, что задача слишком сложная, то попробуйте сферические координаты

Алгоритм Метрополиса-Гастингса или как полезно владеть эргодической теоремой.

Казалось бы, мы разобрались с проблемой генерирования случайных величин и векторов. Однако, 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. Если вы плохо знакомы с теорией марковских цепей (нам понадобится определение марковской цепи, понятия матрицы перехода, эргодичности и эргодического распределения), то прочитайте, например, главу XV "Марковские цепи" тома 1 книги Феллера "Введение в теорию вероятностей и ее приложения". Следующий материал можно использовать и без этого, но понимание теории прояснит почему этот алгоритм работает.

Итак, предположим сперва, что мы хотим разыграть величину X с дискретным распределением со значениями $\{x_1, ..., x_N\}$ и вероятностями $\{p_1, ..., p_N\}$. При этом разрешается знать $p_1, ..., p_N$ с точностью до мультипликативной константы, поскольку мы будем использовать метод, зависящий только от отношения p_i. Чуть ниже будут приведены примеры, почему это зачастую важно.

Метод заключается в следующем:
1) Выберем некоторую переходную матрицу Q (матрицу с суммой по строкам 1 из неотрицательных элементов), соответствующую неразложимой цепи Маркова с N состояниями. Иначе говоря, матрица Q должна быть такой, что для любых $1 \le k \le N$, $1 \le l \le N$ найдется такая цепочка $i_0, ..., i_m$, что $i_0 = k$, $i_m = l$, $Q_{i_j, i_{j+1}} > 0$.
2) Положим $\alpha_{i,j} = \min(p_j Q_{j,i}, p_i Q_{i,j})/(p_i Q_{i,j})$.
3) Введем $P_{i,j} = Q_{i,j}\alpha_{i,j}$, $i \ne j$, $P_{i,i} = Q_{i,i} + \sum_{k \ne i} Q_{i,k}(1 - \alpha_{i,k})$.
4) Сгенерируем цепь Маркова с матрицей P и произвольным начальным состоянием.

Шаг 4) можно делать непосредственно, сгенерировав начальную величину X_0, а затем на каждой шаге, зная $X_n = i$, разыгрывать X_{n+1} с вероятностями $P(X_{n+1} = k) = P_{i,j}$. Однако более удобным зачастую является второй вариант: при $X_n = i$ разыгрывать Y с вероятностью $P(Y = k | X_n = i) = Q_{i,k}$, а при $Y = k$ разыгрываем $X_{n+1} = i$ с вероятностью $1 - \alpha_{i,k}$ или делаем $X_{n+1} = k$ с вероятностью $\alpha_{i,k}$.
Вопрос 5. Почему этот метод даст нужную нам вероятность перехода $P_{i,j}$?

Оказывается, что цепь X_n эргодическая, величины $P(X_n = k)$ сходится к p_k. Таким образом, мы можем подождать достаточно большое число шагов n, взять X_n (допустим оно равно k) и положить в качестве сгенерированного значения X x_k.
Упражнение 3. Докажите, что если $p_i P_{i,j} = p_j P_{j,i}$ при всех i, j, то цепь Маркова с матрицей перехода

P имеет стационарное распределение p. Выведите отсюда, что рассматриваемая цепь действительно эргодична с эргодическим распределением p

Пример 5. Зачем же нам нужен такой сложный алгоритм?
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, но не очень представляем число рассматриваемых перестановок.

При этом мы можем взять отношение соседства (возьмем соседними перестановки, получающиеся друг из друга транспозицией), взять матрицу Q, соответствующую равновероятному переходу в одну из соседних перестановок, обладающих нашим свойством. Тогда $\alpha(i,j)$ будет $min(N(i), N(j))/N(j)$, где $N(k)$ — число соседних с k перестановок. В соответствии с этим мы можем сгенерировать X_n.

Итак,
1) На нулевом шаге берем некоторую перестановку, удовлетворяющую нужному свойству, например, тождественная перестановка имеет наибольшее значение $\sum_{k=1}^{N} k\sigma(k)$.
2) На n-ом шаге выбираем любую из соседних с ней, обладающих нашим свойством (перебирая все соседние перебором всех транспозиций (r,s)). Считаем число соседей и число соседей этой соседней, определяем $\alpha_{i,k}$, выбираем, остаться ли в исходной перестановке или перейти в эту соседнюю. 3) После большого числа n итераций распределение случайной перестановки будет близко с равномерной на нашем множестве.

Упражнение 4. Попробуйте реализовать эту процедуру на R с $N = 20$, $C = 200$.

Исходя из эргодической теоремы мы имеем право применять алгоритм Метрополиса-Гастингса только для дискретных величин X. Однако по факту с некоторыми ограничениями тем же методом можно генерировать и непрерывные случайные величины или векторы, повсеместно заменяя вероятности на плотности.

Пример 6. Одним из вариантов использования алгоритма Хастингса-Метрополиса является так называемый Gibbs Sampler (иногда это переводят как "Гиббсовское семплирование", но это ужасно), используемый для генерирования случайного вектора $X = (X_1, ..., X_n)$ с распределением $P(\vec{x})$, опять же, известным с точностью до константы.

Тогда мы можем считать соседними с вектором $(x_1, ..., x_n)$ векторы $(x_1, ..., x_{i-1}, \tilde{x}_i, x_{i+1}, ..., x_n)$ при различных i, $\tilde{x}_i$.

Алгоритм, таким образом, предписывает нам случайно разыграть номер i изменяемой координаты от 1 до n, а затем в соответствии с вероятностью $P(X_i = \tilde{x}_i | X_j = x_j, j \neq i)$ разыграть новое значение $\tilde{x}_i$ этой координаты. При этом для векторов $\vec{x}, \vec{y}$, отличающихся только одной компонентой i имеем

$$Q_{\vec{x},\vec{y}} = \frac{1}{n} \frac{P(\vec{X} = \vec{y})}{P(X_j = x_j, j \neq i)},$$

а в остальных случаях 0. Подставляя наше значение q в алгоритм Хастингса-Метрополиса, имеем

$$\alpha_{\vec{x},\vec{y}} = \frac{\min(P(\vec{X} = \vec{y})Q_{\vec{y},\vec{x}}, P(\vec{X} = \vec{x})Q_{\vec{x},\vec{y}})}{P(\vec{X} = \vec{x})Q_{\vec{x},\vec{y}})} = 1.$$

Таким образом, вторая часть шага 4) алгоритма в нашем случае не нужна и мы просто ограничиваемся переходом в случайного соседа. Этот же алгоритм используют и для непрерывного случая, когда вектор обладает плотностью.

Пример 7. Предположим, что мы хотим взять наугад выборку из N точек в круге, удаленных друг от друга не менее чем на расстояние r.
Gibbs Sampler предлагает нам сперва взять какой-то набор таких точек. На каждом 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.

Упражнение 5. Попробуйте реализовать эту процедуру на R с N = 50, r = 0.1.

Метод Монте-Карло или случайности против неслучайных задач.

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

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

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

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

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 $O(n^{-1/2})$.

Как видим, 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 $O(n^{-1/2})$, да еще и с какой-то случайной скоростью сходимости.

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

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

Аналогично предыдущему примеру мы можем заметить, что $e^{-x_6} I_{0<x_1<...<x_6}$ — плотность вектора, рас-

пределенного также как $(-\ln R_1, -\ln R_1 R_2, ..., -\ln R_1 .... R_6)$. 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 приближение на нем достигается лишь с определенной вероятностью, нам может не повезти и при конкретном n точность будет неудовлетворительной. 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.

Отжигаем или как не застревать в локальных экстремумах при численной максимизации

Второй 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— стартуя с некоторой начальной точки x_0, по определенному правилу $F(i,x)$ на i-ом 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 $s_{i+1} = s$,
3) Если $f(s) \geq f(s_i)$, то берем $s_{i+1} = s$ с вероятностью $p(T(i), s_i, s)$ (обычно $e^{-(f(s_i)-f(s))/T(i)}$).

Этот метод позволяет нам не застревать в локальных минимумах, вылетая из них засчет температуры. Функцию температуры $T(i)$ берут убывающей, остановка алгоритма происходит, 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 на 40 40 ферзей. Будем рассматривать в качестве пространства расстановки, в которых никакие два ферзя не стоят на одной вертикали (горизонтали), т.е. перестановки σ (тогда ферзей мы ставим в точки $(i, \sigma(i))$. В качестве функции $f()$ расстановки возьмем количество конфликтов (бьющих друг друга) 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 $p(T(i), s_i, s)$ возьмем $e^{-(f(s_i)-f(s))/(T(i)f(s_i))}$.
На контрольных тестах такой метод за 2000 итераций находил подходящую расстановку.
Упражнение 6. Реализуйте этот алгоритм.