Продвинутое семплирование

На этой неделе мы попробуем посмотреть на цепи Маркова и диффузию Ланжевена немного в другом контексте. Мы посмотрим, как можно использовать известное нам для семплирования значений случайных величин.

17.1 Зачем нужно семплирование?

Семлирование – это симуляция значений случайных величин. Прежде всего, мы знаем огромное количество алгоритмов, где в каком-то месте нужно накинуть случайность: от симуляции моделей до обучения и инференса диффузионных моделей типа StableDiffusion. С другой стороны, нам часто интересно получать Монте-Карло семплы из распределений для того, чтобы оценивать матожидания и соответственно интегралы в высокой размерности, где квадратурные формулы не работают.

Оказывается, в каких-то простых и даже не очень простых случаях мы уже умеем генерировать семплы.

Пример 17.1. *Равномерное распределение на отрезке (или на конечном множестве). На самом деле очень неочевидный и непростой концептуально метод с точки зрения того, чтобы поверить, что работает.*

Есть алгоритмы (они называются линейные конгруэнтные генераторы, LCG), которые упрощённо основаны на следующей идее. Рассмотрим единичную окружность: отрезок $[0, 2\pi)$, на котором по любому вещественному значению из $\mathbb{R}$ берётся остаток $x \bmod 2\pi$. Если взять произвольную стартовую точку x_0 и добавлять сдвиги

$$x_i = x_{i-1} + \Delta,$$

то получится детерминированная последовательность точек. Оказывается, что для любого рационального Δ множество различных x_i в последовательности конечно. Но если задать Δ иррациональным, то это множество будет бесконечным и плотным (в любом открытом множестве найдётся как минимум одна точка из траектории). Такая система будет эргодической [5]: вне зависимости от того, откуда мы начали, после некоторого момента семплы будут выглядеть очень похоже на то, как будто они пришли из распределения $U[0, 2\pi]$. Более того, корреляция между типа-семплами будет убывать со временем, то есть, можно в том числе получать и наборы близкие к независимым.

Компьютер пока не умеет численно задавать иррациональные числа, но можно задать достаточно сложные рациональные типа

$$\Delta = 10^8 \frac{27474919237474883664711}{2374748009847484474747873487 48}$$

так, что в результате различных точек будет очень много и их обход будет очень сложным и непредсказуемым после взятия остатка. Для численного моделирования этого достаточно. При этом эта система тоже эргодическая: при больших i значения x_i будто приходят из дискретного равномерного распределения. Стартовое значение x_0 называется обычно seed (наверняка вы его видели в питру) и математически это прямой аналог элементарного исхода $\omega \in \Omega$.

Пример 17.2. *(Обратная функция распределения) В одномерном случае очень часто есть универсальный рецепт семплирования из произвольной плотности $f(x)$. Для этого задают*

$$F^{-1}(t) = \inf\{x : \ F(x) \geq t\}.$$

Такое определение подходит даже в случае, когда строго обратной функции нет. Тогда если взять $U \sim U[0,1]$, то $F^{-1}(U) \sim f$. Таким образом можно семплировать, например, экспоненциальные и гамма-величины.

Пример 17.3. *(Гауссовское распределение) Равномерное распределение также открывает дорогу для семплирования гауссовского. Оказывается (после полярной замены и некоторых манипуляций), что если $U \sim [0,1]$, то*

$$X = \sqrt{-2\ln U}\sin(2\pi U), \ Y = \sqrt{-2\ln U}\cos(2\pi U)$$

являются двумя независимыми гауссовскими величинами $\mathcal{N}(0,1)$. Отсюда можно просто получить $Z = \mu + \sigma X$, который будет иметь распределение $\mathcal{N}(\mu, \sigma^2)$.

Пример 17.4. *(Гауссовский вектор) Если мы умеем генерировать значения независимых $\mathcal{N}(0,1)$, то мы можем получать семплы векторов из*

$$X \sim \mathcal{N}(0, I).$$

Чтобы получить $Y \sim \mathcal{N}(\mu, \Sigma)$, достаточно найти любую матрицу A такую, чтобы получилось $AA^T = \Sigma$ и тогда

$$AX + \mu \sim \mathcal{N}(\mu, \Sigma).$$

Один из самых популярных методов для поиска A – это разложение Холецкого $\Sigma = AA^T$, где A – нижнетреугольная.

Но это примеры, не выходящие за классическую теории вероятности. На самом деле мы знаем более сложные примеры вероятностных моделей.

Пример 17.5. *(Гауссовская смесь) Гауссовская смесь это двухшаговая вероятностная модель, в которой задаётся набор средних $\mu_1, .., \mu_K$, набор ковариационных матриц $\Sigma_1, .., \Sigma_K$ и набор весов $\alpha_1, .., \alpha_K$, которые суммируются в 1 и неотрицательные. С точки зрения плотности это сумма гауссовских плотностей $\mathcal{N}(\mu_k, \Sigma_k)$ с весами α_k. Рецепт семплирования здесь очень простой:*

1. *Семплируется k из распределения на конечном множестве (вероятность – это вес компонента смеси α_k);*
2. *Семплируется $X \sim \mathcal{N}(\mu_k, \Sigma_k)$.*

Пример 17.6. *(Гауссовский процесс) Гауссовский процесс нам хорошо знаком, мы научились его семплировать на самой первой неделе.*

Можно ли взять из головы произвольную плотность $f(x)$ и научиться получать семплы? Что особенно актуально для апостериорных распределений Байесовских методов: что если мы знаем форму плотности, но она не интегрируется в 1 ? Есть более-менее универсальный рецепт: Монте-Карло на Марковских цепях (Markov Chain Monte Carlo, MCMC).

17.2 Цепь Маркова

Давайте немного вспомним классические цепи Маркова с конечным числом состояний. Это процесс в дискретном времени с конечным числом N возможных значений. Такой процесс задаётся стартовым распределением μ_0 и переходной матрицей $P = (p(i, j))$. Можно себе представлять некоторого агента, который с вероятностью $\mu_0(i)$ оказывается в начальный момент в состоянии i и на каждом шаге по времени совершает независимые переходы $i \to j$ с вероятностью $p(i, j)$.

Мы помним, что есть эргодические цепи, где есть единственное *инвариантное распределение*

$$\pi : \ \pi P = \pi,$$

к которому сходится при $t \to \infty$ распределение $\mu_0 P^t$, вероятности нахождения в момент t в состоянии i. Важный момент состоит в том, что со временем цепь вообще будто забывает всю историю и в реальности коррелированные семплы последовательных состояний X_t можно до какого-то предела рассматривать как почти независимые и ковариационная функция быстро убывает с ростом расстояния между семплами. К примеру, для них есть закон больших чисел (для него достаточно эргодичности) и даже разные центральные предельные теоремы [12].

Основная идея методов MCMC – построить цепь Маркова, которая

- Будет эргодической;
- В качестве стационарного распределения будет иметь то, из которого нужно семплировать.

В этом случае наш алгоритм будет следующим:

1. Стартовать процесс из какого-то стартового распределения $X_0 \sim \mu_0$;
2. подождать до момента T....
3. Собирать семплы $X_{T+1}, ..., X_{T+k}, ...$

Или же таким:

1. Стартовать N процессов из какого-то стартового распределения $X_0 \sim \mu_0$;
2. подождать до момента T....
3. Собрать выборку из последних значений N цепей.

Единственный вопрос состоит в том, как именно построить по заданному распределению π такую цепь Маркова (переходные вероятности P). А в более общем (не конечном и не дискретном) случае нужно построить правильную последовательность переходных ядер P_h переходов за время h, в этом случае условие стационарности меры π перепишется так: для любого $A \in \mathcal{F}$ при вычислении распределения после времени h

$$(\pi \mathrm{P}_h)(A) = \int_A \mathrm{P}_h(x, A) d\pi(x)$$

распределение меняться не должно:

$$\pi \mathrm{P}_h = \pi.$$

17.3 Как построить подходящую цепь?

Пока попробуем поработать в конечном случае (с дискретным временем и конечным числом состояний). На самом деле, даже здесь с практической точки зрения MCMC часто будет полезен. Пример ниже приводит в своём тексте известный учёный в области теории вероятности Перси Диаконис [7].

Предположим, что нам приходит зашифрованное сообщение типа

рассеаэдтынпрзнцыпкнЁарезва

и нам известно, что оно получено путём какой-то замены символов кириллицы и знака пробела. Это называется подстановочным кодом (substitution code), частный случай – это

код Цезаря, который получается просто сдвигом алфавита на m символов. Если мы хотим сломать код по-грубому (брутфорсом), то нам придётся перебрать порядка $N!$ различных расшифровок, что даже при небольшом алфавите типа 35 букв составляет

$$35! = 10333147966386144929666651337523200000000 \approx 10^{40}.$$

Допустим, мы знаем язык сообщения (русский). Тогда для русского можно найти в сети Интернет таблицу частот би-грамов (последовательностей двух символов) либо большой русский текст и посчитать её вручную. Помимо этого можно пытаться дополнительно использовать любую другую языковую статистику и даже скоры языковой модели. Исходя из неё можно предположить модель текста: каждый следующий символ текста генерируется независимо в сответствии с вероятностями би-грама kv. Тогда для любого сообщения за линейное время можно вычислить его вероятность:

$$\mathbb{P}\left(\text{абб}\right) = \mathbb{P}\left(\text{б} \mid \text{б}\right)\mathbb{P}\left(\text{б} \mid \text{а}\right)\mathbb{P}\left(\text{а}\right),$$

где переходные вероятности можно взять из таблицы би-грамов и поделить на частоту символа, а частоту символа – из статистики встречаемости букв русского языка.

Естественная идея тогда следующая. Будем начинать с произвольной расшифровки (таблицы, показывающей какую букву в какую нужно перевести при расшифровке) и цепь Маркова будет перебирать расшифровки, переходя в случайную равномерно с вероятностью $1/N!$. Такая цепь эргодическая, но предельное распределение у неё равномерное. То есть, такой перебор – это брутфорс, причём даже хуже изначального.

Чтобы исправить ситуацию, мы можем использовать уже выясненную информацию о языке. Мы не можем перебрать все расшифровки, но мы можем вычислить вероятность текста при условии, что расшифровка дана. Поэтому задача сводится по сути к поиску моды распределения с $N!$ значениями и вероятностями

$$\pi_i = \mathbb{P}\left(\sigma_i \mid text\right) = \frac{\mathbb{P}\left(text \mid \sigma_i\right)}{\sum_{j=1}^{N!}\mathbb{P}\left(text \mid \sigma_j\right)}.$$

Нормировочная константа нам недоступна, но мы попробуем построить цепь Маркова, перебирающую расшифровки(перестановки символов) $\sigma \in S^N$; если эта цепь будет иметь заданное инвариантное распределение и будет эргодической, но после некоторого прогрева мы будем генерировать семплы из распределения π. Далее вычислить моды легко: нужно всего лишь выбрать самые частые и посмотреть на текст. Либо выбрать топ-10 расшифровок, чтобы защититься от риска плохой оценки статистики языка. По сути задача у нас такая: есть K состояний (у нас $K = N!$), есть распределение π на K состояниях и нам нужно построить цепь Маркова (то есть, переходную матрицу P), которая была бы эргодической и качестве инвариантного распределения имела бы π.

Начнём с того, что нам нужен более простой критерий инвариантности π для матрицы P. По опредению инвариантное распределение – это такое, что

$$\pi P = \pi$$

или

$$\sum_{i=1}^{K} \pi_i p_{ij} = \pi_j.$$

Это также иногда называют условием *баланса*. Его можно упростить, получив несложное достаточное условие.

Утверждение 17.1. *Если выполнено условие детального баланса*

$$\pi_i p_{ij} = \pi_j p_{ji},$$

то выполнено условие баланса (и, следовательно, π – инвариантное распределение).

▷ Просуммируем обе части по j:

$$\pi_i = \sum_{j=1}^{K} \pi_j p_{ji}, \ -$$

это уравнение баланса. □

Условие детального баланса называют также *обратимостью по времени* (time reversibility). Это имеет вполне чёткий физический смысл: цепь, запущенная по шагам $t_1, .., t_n$ будет иметь те же распределения, что цепь, запущенная с развёрнутым ядром в обратную сторону по времени (КАРТИНКА).

Теперь нам нужна эргодичность. Вспомним, что для эргодичности цепи тоже есть достаточно простые и проверяемые условия.

Утверждение 17.2. *Цепь Маркова эргодическая, если она*

- *Несократимая (из каждого состояния в каждое можно дойти);*
- *Апериодическая (НОД времён всех возможных возвратов для каждого состояния равен 1).*

Теперь остаётся явно предложить цепь, удостовериться, что она эргодическая и имеет нужное инвариантное распределение. Один из вариантов – это процедура Метрополиса-Гастингса (1953[16]? говорят, было известно раньше). Мы строим двухэтапную процедуру для одного шага по времени. На каждом шаге t

1. Генерируем предложение(proposal) $X' \sim q(\cdot \mid X_t)$;
2. С вероятностью $a(X' \mid X_t)$ принимается предложение и $X_{t+1} = X'$, иначе отвергается и $X_{t+1} = X_t$.

Заметим, что мы получили всё ещё цепь Маркова, причём такая цепь уже точно апериодическая, так как мы закладываем возможность остаться на месте. Ещё мы можем потребовать несократимость, если цепь с ненулевой вероятностью сможет дойти до любого состояния. Чтобы завершить алгоритм, нужно подобрать распределение пропоузала и вероятность принятия. В качестве пропоузала мы, как изначально хотели, можем взять, к примеру, просто случайное состояние из равномерного распределения на K состояниях. Вероятность принятия поможет получить уравнения детального баланса.

Итак, мы строим цепь с переходными вероятностями

$$p(j \mid i) = q(j \mid i)a(j \mid i),$$

так как подброшенная монета на шаге принятия/отвержения не зависит от сгенерированного пропоузала. Уравнение детального баланса требует, чтобы

$$\pi_i q(j \mid i)a(j \mid i) = \pi_j q(i \mid j)a(i \mid j)$$

или по-другому

$$\frac{\pi_i}{\pi_j} = \frac{q(i \mid j)a(i \mid j)}{q(j \mid i)a(j \mid i)}.$$

Метрополис предложил такой вариант:

$$a(i \mid j) = 1 \wedge \frac{\pi_i q(j \mid i)}{\pi_j q(i \mid j)}.$$

Заметим, что если $a(i \mid j) < 1$ то $a(j \mid i) = 1$ и наоборот, поэтому уравнение детального баланса будет выполнено.

Подытожим алгоритм **RWMH**, Random Walk Metropolis-Hastings.

1. $X' \sim Q(\cdot|X_t)$, в качестве Q берём равномерное распределение на всех состояниях.
2. С вероятностью
$$a(X' \mid X_t) = 1 \wedge \frac{\pi_{X'} q(X_t \mid X')}{\pi_{X_t} q(X' \mid X_t)}$$
задаётся $X_{t+1} = X'$, иначе $X_{t+1} = X_t$.

Этот алгоритм несовершенен и можно его в разные стороны улучшать*. Например, мы не использовали знание текущего состояния в пропоузале (можно, к примеру, предлагать состояния из окрестности текущего, чтобы обход был более локальным). Ещё пропоузал можно делать более сложным, но всё это по большей части более специфичные детали.

**У меня такой брутфорс с русским не заработал за* 30 *минут времени, но по интернету у кого-то завелось. Думаю, проблема в очень долгом прогреве и секрет в правильной априорной информации для пропоузала.*

17.4 В непрерывном случае

Как мы можем обобщить подход на случай, когда нам нужно семплировать из данной плотности $f(x)$? Оказывается, что как и раньше нам нужна цепь Маркова в дискретном времени, которая будет несократимой (в смысле, что из любого состояния траектория из точки x проходит с ненулевой вероятностью любую окрестность любой точки y за конечное время), апериодической (цепь вернётся в любую окрестность старта за конечное время, но без периода). Здесь нам, как и раньше, помогает уравнение детального баланса, которое для случая, когда переходное ядро имеет плотность $p(y|x)$ и желаемое распределение имеет плотность $f(x)$, можно записать как

$$f(x)p(y|x) = f(y)p(x|y).$$

Пример 17.7. *Поэтому если мы находимся в $\mathbb{R}^d$ и плотность $f(x)$ ненулевая на всём пространстве, то мы можем построить непрерывную версию* ***RWMH(cont)****, к примеру, так:*

1. *$X' \sim Q(\cdot|X_t)$, в качестве Q берём гауссовское распределение $\mathcal{N}(X_t, \sigma^2)$.*
2. *С вероятностью*
$$a(X' \mid X_t) = 1 \wedge \frac{f(X')q(X_t \mid X')}{f(X_t)q(X' \mid X_t)}$$
задаётся $X_{t+1} = X'$, иначе $X_{t+1} = X_t$.

Мы могли бы взять пропоузал $\mathcal{N}(0, \sigma^2)$, но проблема такого подхода была бы в том, что если большая часть вероятностной массы $f(x)$ находится далеко от 0, то все пропоузалы бы с очень высокой вероятностью отвергались и мы бы ничего не добились. С другой стороны такая идея случайного блуждания позволяет уверенно обойти всю плоскость в надежде, что когда-то мы точно зайдём в область, где $f(x)$ ощутимо больше нуля (КАРТИНКА). Вообще алгоритмы типа RWMH дают несколько узкий взгляд на задачу. Мы сумели построить цепь с нужными свойствами, но для пропоузала никак не используется знание плотности $f(x)$. Это приводит к тому, что имеющаяся информация используется неэффективно. Проблема алгоритмов MH в том, что они в теории дают эргодическую цепь, но почти нет гарантий, что она обязательно сойдётся за разумное время.

В ядре алгоритма MCMC находится эргодическая цепь. Но не так давно мы с помощью уравнения Фоккера-Планка узнали, что инвариантное распределение есть у модели диффузии Ланжевена. Например, у диффузии

$$dX_t = \nabla \ln f(X_t)dt + \sqrt{2}dW_t$$

инвариантным распределением будет плотность $C_0 f(x)$ (даже в случае дискретизации по методу Эйлера), с точностью до нормировочной константы то, что нужно.

Пример 17.8. *Первый простой подход – просто запустить диффузию с помощью метода Эйлера, подождать и потом собирать семплы. Это приводит к алгоритму* ***ULA*** *(Unadjusted Langevin Algorithm).*

Но есть нюанс. Как показывают исследования, дискретизация не проходит бесследно[9]: с постоянным шагом по времени алгоритм ULA асимптотически смещён относительно настоящей стационарной плотности и смещение зависит от шага. Впрочем, если брать переменный шаг, этот эффект можно попробовать починить, но это непросто с точки зрения подбора параметров [8]. В общем, это недостаток схемы, который вместе со временем прогрева делает её сложной в использовании. С другой стороны, что мешает использовать диффузию в качестве пропоузала в схеме Метрополиса-Гастингса? После дискретизации у нас останется Марковская цепь, пропоузал больше не будет симметричным, хотя условие детального баланса будет выполняться. При этом в силу самой природы метода Метрополиса-Гастингса, инвариантное распределение будет таким, как надо.

Пример 17.9. *Второй способ приводит к алгоритму* ***MALA*** *(Metropolis-Adjusted Langevin Algorithm).*

1. *Сгенерировать пропоузал из ULA, формально*

$$X' \sim \mathcal{N}(X_{t_k} + \nabla \ln f(X_{t_k}), 2\Delta t_k).$$

2. *С вероятностью*

$$a(X' \mid X_t) = 1 \wedge \frac{f(X')q(X_t \mid X')}{f(X_t)q(X' \mid X_t)}$$

задаётся $X_{t_{k+1}} = X'$, *иначе* $X_{t_{k+1}} = X_{t_k}$.

Преимущества MALA в том, что мы ликвидировали асимптотическое смещение ULA, а кроме того улучшили эффективность: цепи (во всяком случае теоретически) нужно меньше времени на прогрев, чем в случае ULA. Но ULA отметать полностью не стоит. Что будет, если мы стартанём и

Наконец, в непрерывном случае есть ещё один способ, который относится к MCMC, но формально не совсем такой. Это Гамильтонов Монте-Карло **(HMC)**. Методы HMC полностью детерминированные и используют правильно собранную систему обыкновенных дифференциальных уравнений типа

$$\dot{x} = p,\ \dot{p} = g(x)$$

Если система гамильтонова, то есть, существует функция $H(x, p)$ такая, что эта система имеет вид

$$\dot{x} = \nabla_p H,\ \dot{p} = -\nabla_x H.$$

Это история из самой классической физики и очень хорошо изученные системы с позиции математической физики в недавнее время [26]. Про такие системы известно, что они эргодические и имеют инвариантную меру. Например, если задать гамильтониан

$$H(x,p) = U(x) + \frac{1}{2}p^T V^{-1} p,$$

то инвариантная мера будет $e^{-U(x)}$. С точки зрения физики, первый член – это потенциальная энергия системы, а второй – кинетическая.

Пример 17.10. *(дописать про техническую сторону) В теории и на практике это достаточно неплохой подход, но он страдает от некоторых недостатков:*

1. *Чувствительность к стартовому условию: может не повезти и ждать сходимости нужно долго.*

2. *Схема дискретизации важна. Популярный стандартный выбор – схема leapfrog, но есть более сложные варианты (например, No U-turn).*

3. *Ещё динамика рекуррентна в том смысле, что если ждать достаточно долго, то система постарается вернуться в окрестность старта. Чтобы получить достаточно хаотическое поведение, нужно подбирать параметры и добавлять случайность в разных местах, например задавать поле скоростей с использованием плотности $f(x)$ и дополнительной случайности.*

Всё это чинится разными техническими улучшениями HMC и сам базовый алгоритм HMC сильно отличается от того, чтобы просто как есть использовать обыкновенное дифференциальное уравнение. Отдельного внимания заслуживает то, что можно применить шаг Метрополиса-Гастингса, и как именно, это отдельная история [23, 6].

17.5 Ещё альтернативы MCMC

MCMC – неплохой универсальный рецепт семплирования в высокой размерности в случае, когда дана известная плотность $f(x)$. Для эргодических средних формально зависимых наблюдений есть законы больших чисел, то есть, MCMC можно использовать для вычисления матожиданий. Более того, других способов в общий ситуации почти нет. Всё же есть достаточно много недостатков:

(-) Нужно прогревать цепь, ждать сходимости к инвариантному распределению. Сложно понять, когда именно это происходит. И можно ждать очень долго.

(-) Очень много дополнительной инженерии для того, чтобы всё заработало, как хотелось бы и добиваться быстрой сходимости.

Всё это приводит к вычислительной неэффективности: большое количество арифметики проходит впустую во время прогрева. По этой причине стал развиваться генеративный подход: вместо того, чтобы (к примеру, с помощью ММП) оценить вероятностную модель и думать, как из неё семплировать при известном распределении, пытаются изначально строить такую модель, которая бы явно включала в себя явно блок генерации, который при обученной модели мог бы быстро генерировать семплы.

Это немного другая постановка: теперь у нас нет явно плотности $f(x)$, но есть данные, семплы из неё. Генеративные модели пытаются что-то узнать из данных, чтобы сразу генерировать семплы из распределения, близкого к $f(x)$. Среди главных моделей здесь выступают

1. **Гауссовские смеси** (да, это простейшая генеративная модель). Эта модель оценивается EM-алгоритмом и семплирование происходит по очень простой схеме.

2. **Гауссовские процессы** (да, это генеративная модель, но немного сложнее). Она оценивается с помощью метода максимального правдоподобия и потом можно просто генерировать траектории как гауссовские векторы.

3. **Фильтр Калмана**. В известных пределах отлично себя показывает на симуляции финансовых данных для задач, связанных с биржей. Его можно обучать в том числе в онлайн-режиме с помощью EM и генерировать итеративно с известными функциями перехода и ковариационными матрицами.

4. **Вариационный автоэнкодер (VAE).** Это модель, которая имеет функцию вложения в скрытое пространство (энкодер) и функцию реконструкции из скрытого представления в пространство данных (декодер). Две эти функции правильным образом параметризуются и обучаются на задачу хорошей реконструкции и хорошей аппроксимации распределения. После обучения для генерации нужно сгенерировать вектор стандартного гауссовского шума и прогнать через декодер.

5. **Нормализующие потоки (NF).** Это модель, немного похожая на VAE, но энкодер $f(x)$ явно обратим, поэтому декодер задаётся как $f^{-1}(y)$.

6. **Генеративно-состязательные сети (GAN).** В этой модели генератор – это функция, которая гауссовский стандартный вектор должна перевести в экземпляр, похожий на данные. Ещё есть блок дискриминатора, который по данному экземпляру классифицирует фейк (сгенерировано генератором) и не фейк (оеальный семпл данных). По итогам обучения имеется модель генератора, которая тоже способна быстро генерировать семплы.

7. **Диффузионные модели.** Здесь после приглашённой лекции добавить нечего :). Тоже модель, которая призвана отойти от модели семплирования из готовой плотности.

Разумеется, у всех этих подходов есть свои проблемы и преимущества, но отчасти их развитие мотивировано в том числе недостатками MCMC для задач с данной выборкой наблюдений и явно неизвестной плотностью.

[14] M. G. Kendall and A. Bradford Hill. The analysis of economic time-series-part i: Prices. *Journal of the Royal Statistical Society. Series A (General)*, 116(1):11–34, 1953.

[15] Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125–144, 1976.

[16] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 06 1953.

[17] Bernt Oksendal. *Stochastic Differential Equations (3rd Ed.): An Introduction with Applications*. Springer-Verlag, Berlin, Heidelberg, 1992.

[18] Paul A. Samuelson. Proof that properly discounted present values of assets vibrate randomly. *The Bell Journal of Economics and Management Science*, 4(2):369–374, 1973.

[19] O. A. Stepanov. Kalman filtering: Past and present. an outlook from russia. (on the occasion of the 80th birthday of rudolf emil kalman). *Gyroscopy and Navigation*, 2(2):99–110, Apr 2011.

[20] R. L. Stratonovich. Conditional markov processes. *Theory of Probability & Its Applications*, 5(2):156–178, 1960.

[21] Holbrook Working. A random-difference series for use in the analysis of time series. *Journal of the American Statistical Association*, 29(185):11–24, 1934.

[22] L.E. Zachrisson. On optimal smoothing of continuous time kalman processes. *Information Sciences*, 1(2):143–172, 1969.

[23] Guangyao Zhou. Metropolis augmented hamiltonian monte carlo. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022.

[24] Н. В. Рекнер М. Шапошников С. В. Богачев, В. И. Крылов. *Уравнения Фоккера – Планка – Колмогорова*. Институт компьютерных исследований, 2013.

[25] Ширяев А.Н. Булинский А.В. *Теория случайных процессов*. М:ФИЗМАТЛИТ, 2005.

[26] Тиморин В.А. *Геометрия гамильтоновых систем и уравнений с частными производными*. Москва : ВШЭ, 2017.

[27] Синай Я.Г. Коралов Л.Б. *Теория вероятностей и случайные процессы*. МЦНМО, 2013.

[28] Р.Л. Стратонович. *Условные марковские процессы и их применение к теории оптимальъного управления*. Московский государственный университет, 1966.

[29] А.Н. Ширяев. *Основы стохастической финансовой математики*. МЦНМО, 2016.