СЭМПЛИРОВАНИЕ В МАШИННОМ ОБУЧЕНИИ

Сергей Николенко

СПбГУ — Санкт-Петербург

4 апреля 2024 г.

Random facts:

- 4 апреля — Международный День Интернета, празднующийся в день преставления (смерти) святого Исидора Севильского, покровителя учеников и студентов, создавшего первую в истории энциклопедию «Etymologiae» в 20 томах; кроме того, дата 4.04 создаёт аллюзию на ошибку HTTP 404
- 4 апреля 1147 г. Юрий Долгорукий устроил пир в честь своего союзника князя Новгород-Северского Святослава Ольговича; проходило пиршество в малоизвестном городке Москва, и поэтому её и упомянули в Ипатьевской летописи
- 4 апреля 1786 г. Екатерина II издала указ о запрещении «пускать шары в предупреждение пожарных случаев и несчастных приключений», чем сильно замедлила развитие воздухоплавания в России, а 4 апреля 1919 г. в Италии была открыта пассажирская авиалиния Рим — Неаполь на дирижаблях
- 4 апреля 1581 г. бывший пират Фрэнсис Дрейк завершил кругосветное путешествие; Елизавета I взошла на борт его судна «Пеликан» и посвятила Дрейка в рыцари
- 4 апреля 1833 г. магазин Смирдина начал продавать первое полное издание «Евгения Онегина»

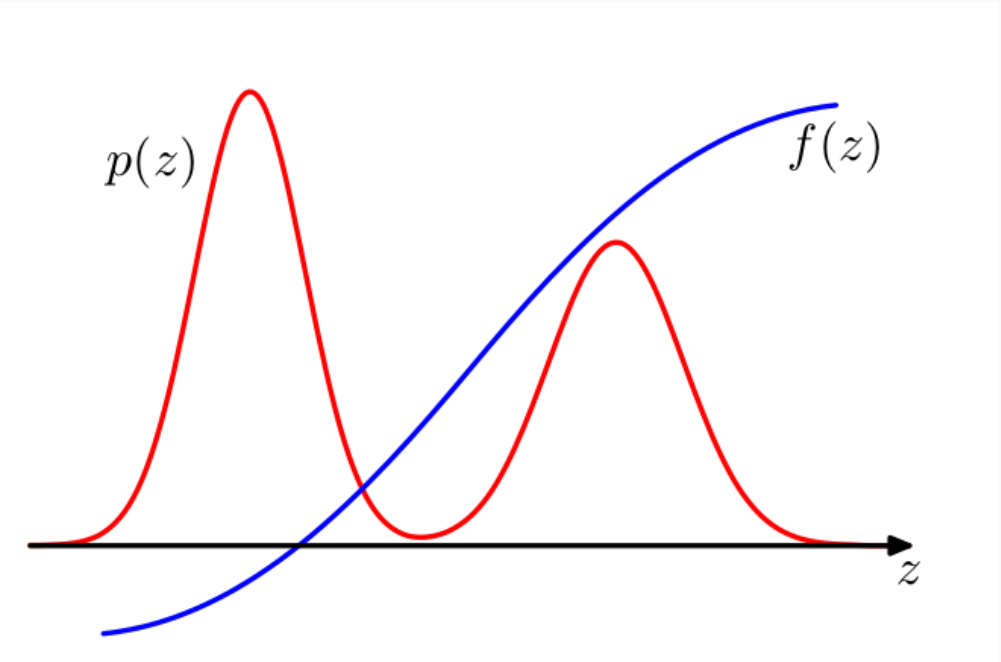
Постановка задачи сэмплирования

Почему проблема

- Пусть у нас есть некоторое вероятностное распределение.
- Как с ним работать? Как, например, его симулировать?
- Мы не всегда можем приблизить (как по методу Лапласа) распределение каким-нибудь известным так, чтобы всё посчитать в явном виде.
- Например, в кластеризации: мультимодальное распределение с кучей параметров, что с ним делать?

Ожидания

- Однако часто задача не в том, чтобы что-то сделать с самой плотностью, а в том, чтобы считать ожидания функций:



Постановка задачи

- Пусть имеется некое распределение $p(x)$.
- Задача 1: научиться генерировать сэмплы $\{x^{(r)}\}_{r=1}^{R}$ по $p(x)$.
- Задача 2: научиться оценивать ожидания функций по распределению $p(x)$, т.е. научиться оценивать интегралы вида

$$E_p[f] = \int p(x)f(x)dx.$$

Постановка задачи

- Мы будем обычно предполагать, что x — это вектор из $\mathbb{R}^n$ с компонентами x_n, но иногда будем рассматривать дискретные множества значений.
- Функции f — это, например, моменты случайных величин, зависящих от x.
- Например, если $t(x)$ — случайная величина, то её среднее — это $E_p[t(x)]$ $(\int p(x)t(x)dx)$, а её дисперсия равна $E_p[t^2] - (E_p[t])^2$.
- И мы предполагаем, что явно вычислить не получается — слишком сложная функция p.

Ожидания и сэмплинг

- Мы будем заниматься только сэмплингом, потому что задача оценки ожиданий функций легко решится, если мы научимся делать сэмплинг.
- Как она решится?

Ожидания и сэмплинг

- Мы будем заниматься только сэмплингом, потому что задача оценки ожиданий функций легко решится, если мы научимся делать сэмплинг.
- Как она решится?
- Нужно взять сэмплы $\{x^{(r)}\}_{r=1}^{R}$ и подсчитать

$$\hat{f} = \frac{1}{R} \sum_{r} f(x^{(r)}).$$

- Ожидание $\hat{f}$ равно $E_p[f]$, а дисперсия убывает обратно пропорционально R.

Monte Carlo EM

- Пример применения: вспомним, где мы часто вычисляем ожидания – в алгоритме EM, на E-шаге:

$$Q\left(\theta, \theta^{\text{old}}\right) = \int p(\mathbf{Z} \mid \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{Z}, \mathbf{X} \mid \theta) d\mathbf{Z}.$$

- Давайте приблизим:

$$Q\left(\theta, \theta^{\text{old}}\right) \approx \frac{1}{R} \sum_{r=1}^{R} \ln p\left(\mathbf{Z}^{(r)}, \mathbf{X} \mid \theta\right) d\mathbf{Z}.$$

- А потом будем это приближение оптимизировать; получится *Monte Carlo EM*.
- Пример ещё проще: байесовские предсказания – это ожидания известных функций по сложному апостериорному распределению, и посчитать их руками обычно сложно.

Сэмплирование известных функций

Сэмплирование известной плотности

- Как сэмплировать из известного распределения? Ну скажем, нормального?
- Предположим, что умеем сэмплировать $z \in [0, 1]$ равномерно, `rand`.
- Какое будет распределение $p(y)$ у $y = f(z)$?

Сэмплирование известной плотности

- Как сэмплировать из известного распределения? Ну скажем, нормального?
- Предположим, что умеем сэмплировать $z \in [0, 1]$ равномерно, `rand`.
- Какое будет распределение $p(y)$ у $y = f(z)$?

$$p(y) = p(z) \left| \frac{\mathrm{d}z}{\mathrm{d}y} \right|, \text{ и тут } p(z) = 1.$$

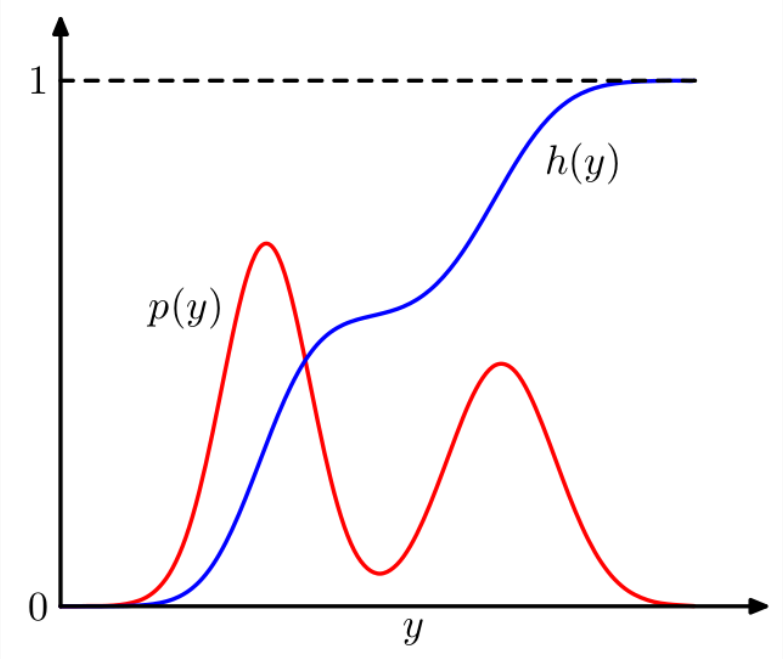
- Нам надо выбрать $f(z)$ так, чтобы получилось заданное $p(y)$. Это что за $f(z)$ будет?

Сэмплирование известной плотности

- Надо выбрать

$$z = h(y) = \int_{-\infty}^{y} p(\hat{y})\mathrm{d}\hat{y},$$

неопределённый интеграл от $p(y)$.



- Т.е. надо как-то найти $h^{-1}(z)$.

Сэмплирование известной плотности

- Например, что будет для экспоненциального распределения

$$p(y) = \lambda e^{-\lambda y}, \text{ где } y \in [0, \infty)?$$

Сэмплирование известной плотности

- Например, что будет для экспоненциального распределения

$$p(y) = \lambda e^{-\lambda y}, \text{ где } y \in [0, \infty)?$$

- Будет интеграл от нуля, $h(y) = 1 - e^{-\lambda y}$, и $y = \frac{1}{\lambda} \ln(1 - z)$.
- То же и с многомерными распределениями, только теперь якобиан

$$p(y_1, \dots, y_M) = p(z_1, \dots, z_M) \left| \frac{\mathrm{d}(z_1, \dots, z_M)}{\mathrm{d}(y_1, \dots, y_M)} \right|.$$

Сэмплирование гауссиана

- А как гауссиан сэмплировать?

Сэмплирование гауссиана

- А как гауссиан сэмплировать?
- Преобразование Бокса-Мюллера: сначала сгенерируем (z_1, z_2) равномерно из единичного круга, потом посчитаем

$$y_1 = z_1\sqrt{\frac{-2\ln r^2}{r^2}},\ y_2 = z_2\sqrt{\frac{-2\ln r^2}{r^2}},\ \text{где } r^2 = z_1^2 + z_2^2.$$

- Тогда совместное распределение

$$p(y_1, y_2) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}y_1^2}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}y_2^2}.$$

- Всё понятно?..

Выборка с отклонением

Что же сложного в сэмплинге?

- Мы предполагаем, что дана функция $p^*(x)$, которая отличается от $p(x)$ только нормировочной константой $Z = \int p^*(x)dx$: $p(x) = p^*(x)/Z$.
- Почему трудно делать сэмплинг?
- Во-первых, мы обычно не знаем Z; но это не главное.
- Главное — обычно правильные сэмплы p^* часто попадают туда, где p^* велика. А как определить, где она велика, не вычисляя её *везде*?

Дискретизация пространства

- Простейшая идея: давайте дискретизуем пространство, вычислим p^* на каждом участке (пусть она гладкая), потом будем брать дискретные сэмплы, зная все вероятности (это нетрудно).
- Сколько же будет дискретных участков?
- Главная проблема — обычно велика размерность x. Например, если разделить каждую ось на 20 участков, то участков будет 20^n; а n в реальных задачах может достигать нескольких тысяч...
- Иными словами, такой подход никак не работает.

Пример: сколько в озере нефти?

- Перед вами — участок, под которым залежи нефти (да хоть подземное озеро нефти).
- Вам нужно определить, сколько её тут.
- Вы можете проводить замер в каждой конкретной точке, чтобы определить глубину слоя в этой точке.
- Проблема в том, что значительная часть общего объёма нефти может быть сосредоточена в глубоких, но узких каньонах.
- И это только размерность два. :)

Равномерное сэмплирование

- Может быть, всё-таки получится решить хотя бы вторую задачу?
- Давайте брать сэмплы $\{x^{(r)}\}_{r=1}^{R}$ *равномерно* из всего пространства, затем вычислять там p^* и нормализовать посредством $Z_R = \sum_{r=1}^{R} p^*(x^{(r)})$.
- Тогда $\hat{f}$ можно будет оценить как

$$\hat{f} = \frac{1}{Z_R} \sum_{r=1}^{R} f(x^{(r)}) p^*(x^{(r)}).$$

- В чём проблема?

Равномерное сэмплирование

- Да в том же самом.
- Обычно значительная часть p^* сосредоточена в очень небольшой части пространства.
- Вероятность попасть в неё за R равномерно выбранных сэмплов тоже экспоненциально мала (например, если по каждой оси вероятность попасть $1/2$, и всё независимо, то получится вероятность 2^{-n}).
- Так что даже вторую задачу решить не получится.

Суть

- Но что-то всё-таки делать надо.
- Выборка с отклонением — rejection sampling.
- Наше предположение теперь в том, что у нас есть q^*, которое мы можем сэмплировать и про которое мы знаем константу c, такую, что

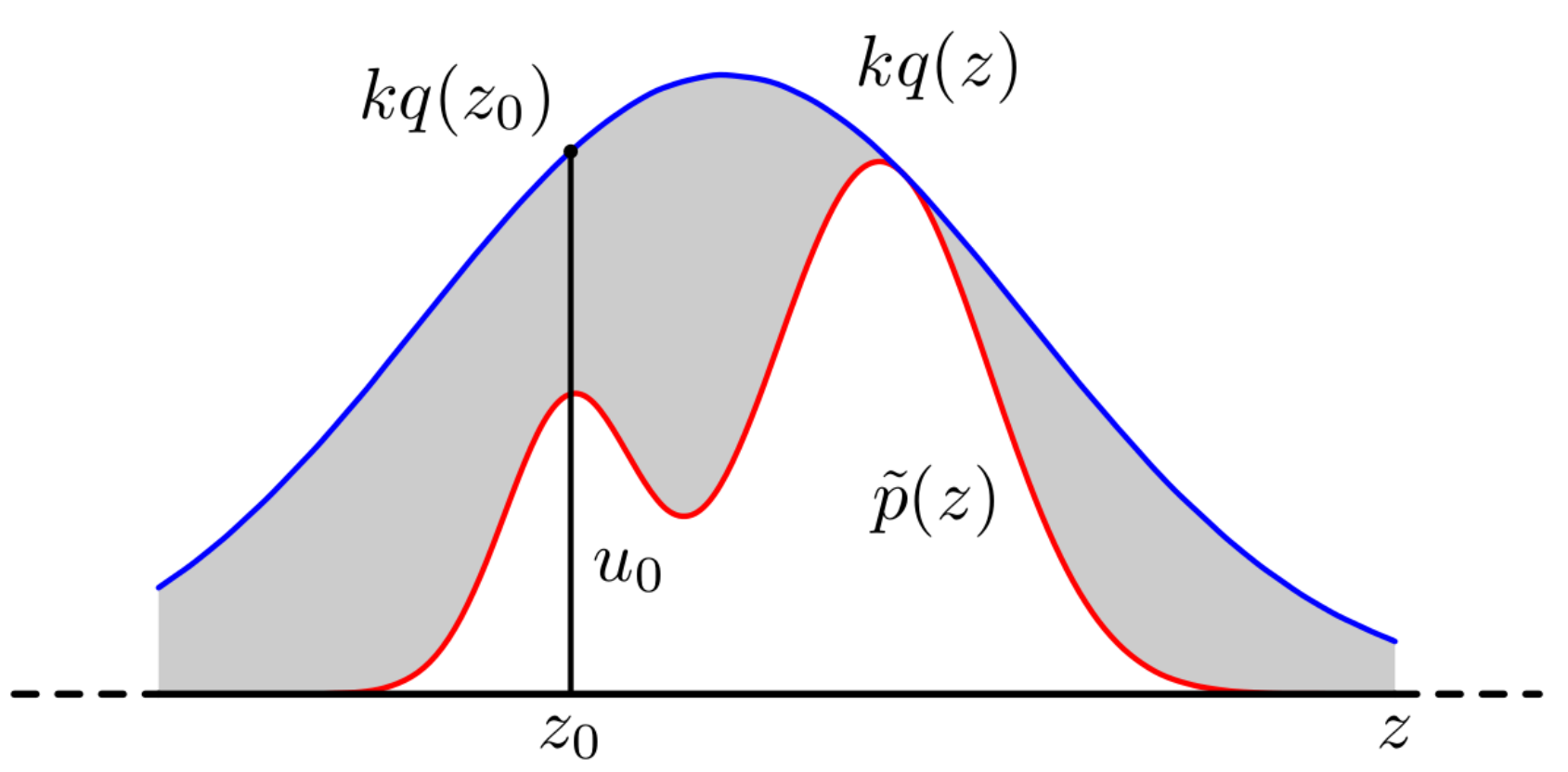
$$\forall x \quad cq^*(x) > p^*(x).$$

- Тогда мы сумеем сэмплировать p.

Алгоритм формально

- Взять сэмпл x по распределению $q^*(x)$.
- Выбрать случайное число u равномерно из интервала $[0, cq^*(x)]$.
- Вычислить $p^*(x)$. Если $u > p^*(x)$, x отклоняется (отсюда и название), иначе добавляется в сэмплы.

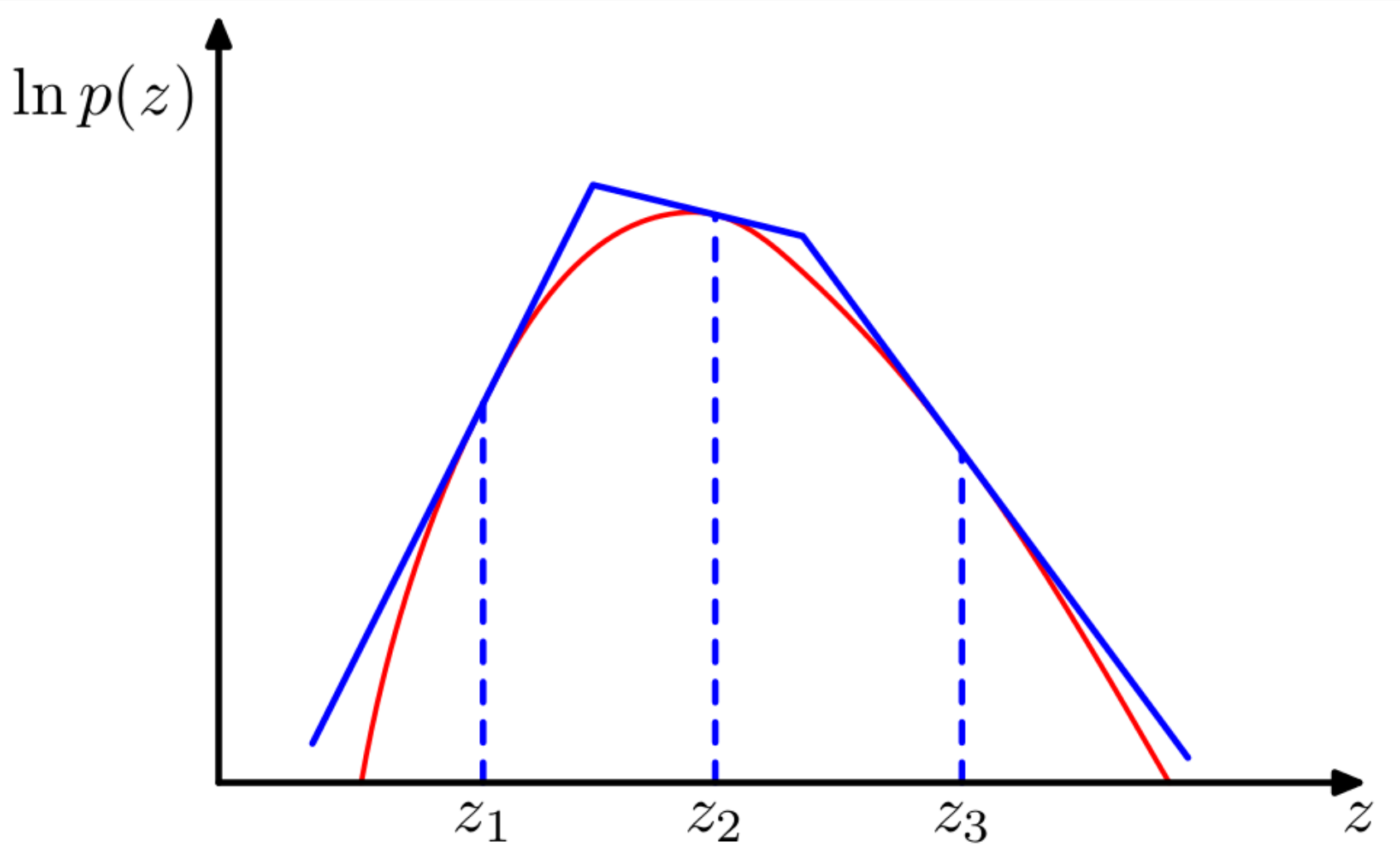
Выборка с отклонением



Обоснование

- Алгоритм работает, потому что выбирает точки $[x, u]$ равномерно из области под графиком $p^*(x)$, а это и значит, что получатся сэмплы p^*.
- Вариант – *адаптивная выборка*: если мы можем точнее определить $q(x)$, например построить её как многогранник, касающийся выпуклой (как правило, лог-выпуклой – и многогранник в логарифмическом пространстве) плотности распределения.

Адаптивная выборка с отклонением



Сэмплирование по значимости

Сэмплирование в графических моделях

- Вариант выборки с отклонением можно применить к направленным графическим моделям.
- Сэмплировать без evidence – тривиально.
- Сэмплировать с evidence можно так: сделаем сэмпл, если наблюдаемые переменные не сошлись, выкинем.
- Для ненаправленных не так просто, да и для направленных не сработает, если наблюдаемых много.

Проблемы

- Как и у предыдущего алгоритма, у выборки с отклонением начинаются проблемы в больших размерностях.
- Суть проблемы та же, что в предыдущем случае, а выражается она в том, что c будет очень большим (экспоненциальным от n), и почти все сэмплы будут отвергаться.

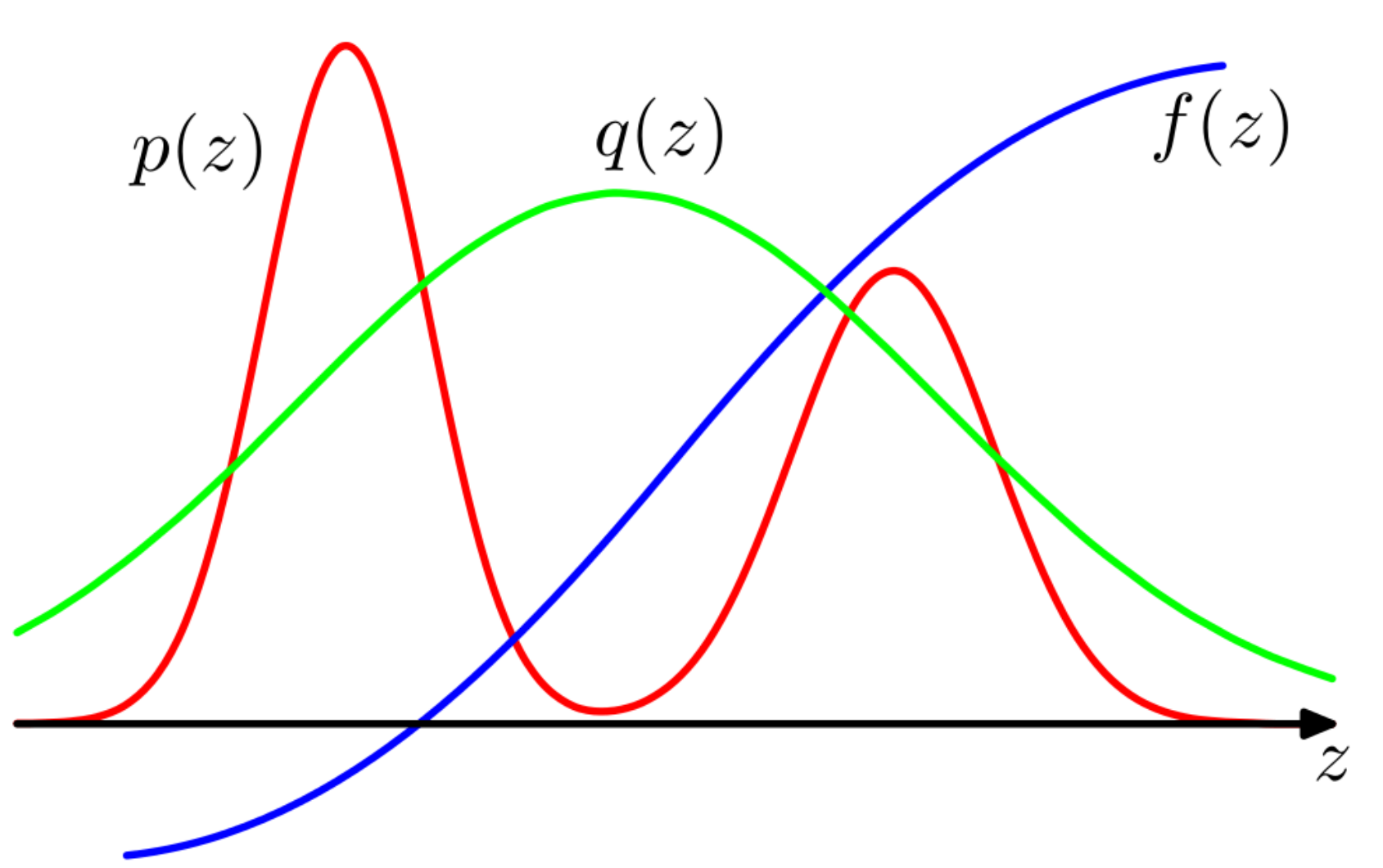
Суть метода

- Выборка по значимости — importance sampling.
- Мы решаем только вторую задачу, а не первую.
- То есть нам нужно брать сэмплы, при этом желательно попадая в зоны, где функция p^* имеет большие значения.

Суть метода

- Предположим, что у нас есть какое-то другое распределение вероятностей q (точнее, q^*), попроще, и мы умеем брать его сэмплы.
- Тогда алгоритм такой: сначала взять выборку по q^*, а затем перевзвесить её так, чтобы получилась всё-таки выборка по p^*.

Выборка по значимости



Вывод

- Мы хотим

$$\mathrm{E}[f] = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \int f(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}q(\mathbf{x})d\mathbf{x} = \approx \frac{1}{L}\sum_r \frac{p(\mathbf{x}^{(r)})}{q(\mathbf{x}^{(r)})}f(\mathbf{x}^{(r)}).$$

- $w_r = p(\mathbf{x}^{(r)})/q(\mathbf{x}^{(r)})$ – веса, с которыми входят сэмплы, но все сэмплы остаются в множестве.

Вывод

- Если у нас не p и q, а p^* и q^*, и $p = \frac{1}{Z_p} p^*$, $q = \frac{1}{Z_q} q^*$, то

$$\mathrm{E}[f] = \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \frac{Z_q}{Z_p} \int f(\mathbf{x}) \frac{p^*(\mathbf{x})}{q^*(\mathbf{x})} q(\mathbf{x}) d\mathbf{x} \approx$$

$$\approx \frac{Z_q}{Z_p} \frac{1}{R} \sum_{r=1}^{R} \frac{p^*(\mathbf{x}^{(r)})}{q^*(\mathbf{x}^{(r)})} f(\mathbf{x}^{(r)}),$$

и Z_q / Z_p можно оценить из тех же сэмплов:

$$\frac{Z_p}{Z_q} = \frac{1}{Z_q} \int p^*(\mathbf{x}) d\mathbf{x} = \int \frac{p^*(\mathbf{x})}{q^*(\mathbf{x})} q(\mathbf{x}) d\mathbf{x} \approx \frac{1}{R} \sum_{r=1}^{R} \frac{p^*(\mathbf{x}^{(r)})}{q^*(\mathbf{x}^{(r)})}.$$

Вывод

- Получаем такой алгоритм:
  1. Взять сэмплы $\{x^{(r)}\}_{r=1}^{R}$ по распределению q^*.
  2. Рассчитать веса

$$w_r = \frac{p^*(\mathbf{x}^{(r)})/q^*(\mathbf{x}^{(r)})}{\sum_m p^*(\mathbf{x}^{(m)})/q^*(\mathbf{x}^{(m)})}.$$

  3. Оценить функцию по формуле

$$\mathrm{E}[f] \approx \frac{1}{R}\sum_{r=1}^{R} w_r f(\mathbf{x}^{(r)}).$$

Обсуждение

- Зачем нужно q? Чем это лучше равномерного распределения?

Обсуждение

- Зачем нужно q? Чем это лучше равномерного распределения?
- Проще говоря, распределение q должно помочь выбрать те участки, на которых имеет смысл сэмплить r.
- Если q хорошее, то может помочь, а если плохое, может только навредить.
- Но есть и более фундаментальные проблемы.

Проблемы

- Во-первых, сэмплер q не должен быть слишком узким.
- Например, если сэмплер гауссиановский с небольшой вариацией, то пики r далеко от центра q вообще никто не заметит.

Проблемы

- Во-вторых, может случиться, что все сэмплы будут напрочь убиты небольшим количеством сэмплов с огромными весами. Это плохо.
- Чтобы показать, как это бывает, давайте перейдём в многомерный случай.

Проблемы

- Пусть есть равномерное распределение r на единичном шаре и сэмплер q — произведение гауссианов с центром в нуле:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2\sigma^2}\sum_i x_i^2}.$$

Упражнение. Найдите среднее и дисперсию расстояния $r^2 = \sum_i x_i^2$ точки, взятой по этому распределению.

Проблемы

- Ответ на упражнение: расстояние будет $N\sigma^2 \pm \sqrt{2N}\sigma^2$ (распределение будет похоже на гауссовское).
- Значит, почти все сэмплы лежат в «типичном множестве», кольце расстоянием около $\sigma\sqrt{N}$ от нуля.

Проблемы

- Тогда большинство сэмплов q будут лежать в интервале

$$\frac{1}{(2\pi\sigma^2)^{n/2}} 2^{-\frac{N}{2} \pm \frac{\sqrt{2N}}{2}},$$

  и ненулевые веса будут иметь значения порядка

$$(2\pi\sigma^2)^{n/2} 2^{\frac{N}{2} \pm \frac{\sqrt{2N}}{2}}.$$

- Это значит, что максимальный вес будет относиться к среднему примерно как $2^{\sqrt{2N}}$, а это очень много.

Сэмплинг в графических моделях

- Варианты выборки по значимости для направленных графических моделей:
  - uniform sampling – фиксируем evidence, выбираем остальные равномерно, вес у сэмпла получается просто $p(\mathbf{x})$, потому что он автоматически сходится с evidence;
  - likelihood weighted sampling – фиксируем evidence, выбираем остальные от родителей к детям из условного распределения $p(x_i \mid \text{pa}(x_i))$, где $\text{pa}(x_i)$ уже зафиксированы; вес тогда будет

$$r(\mathbf{x}) = \prod_{x_i \notin E} \frac{p(x_i \mid \text{pa}(x_i))}{p(x_i \mid \text{pa}(x_i))} \prod_{x_i \in E} \frac{p(x_i \mid \text{pa}(x_i))}{1} = \prod_{x_i \in E} p(x_i \mid \text{pa}(x_i)).$$

Заключение

- Если размерность большая, то у выборки по значимости есть две большие проблемы.
- Во-первых, чтобы получить разумные сэмплы, нужно уже заранее выбрать q так, чтобы оно хорошо аппроксимировало p.
- Во-вторых, даже если их получить, часто может так случиться, что веса у некоторых сэмплов будут слишком велики.
- В общем, для случая многих размерностей это не очень хороший метод.

Марковские методы Монте-Карло

Общая идея

- Алгоритм Метрополиса-Гастингса; суть алгоритма похожа на выборку с отклонением, но есть важное отличие.
- Распределение q теперь будет меняться со временем, зависеть от текущего состояния алгоритма.
- Как и прежде, нужно распределение q, точнее, семейство $q(x'; x^{(t)})$, где $x^{(t)}$ — текущее состояние.
- Но теперь q не должно быть приближением p, а должно просто быть каким-нибудь сэмплируемым распределением (например, сферический гауссиан).
- Кандидат в новое состояние x' сэмплируется из $q(x'; x^{(t)})$.

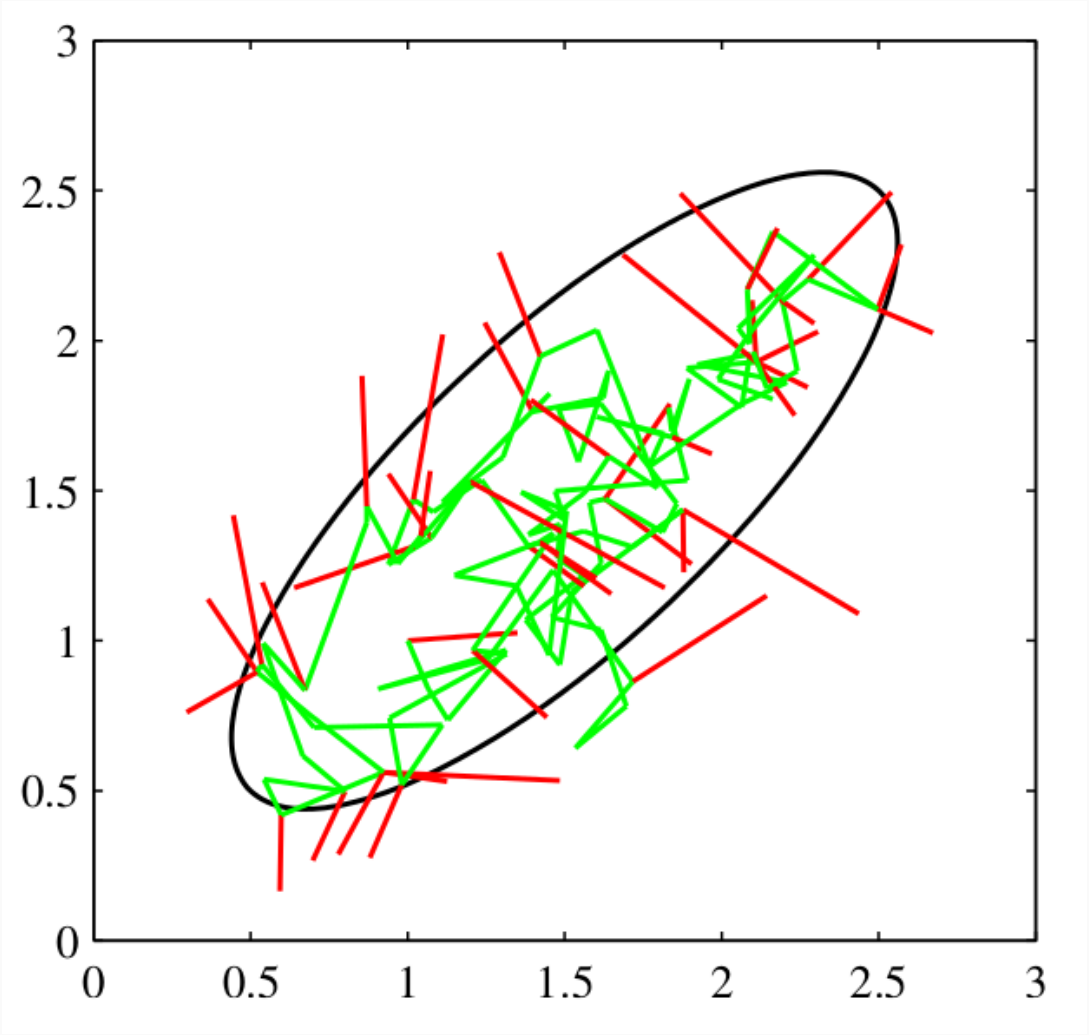
Алгоритм

- Очередная итерация начинается с состояния $x^{(i)}$.
- Выбрать x' по распределению $q(x'; x^{(i)})$.
- Вычислить
$$a = \frac{p^*(x')}{p^*(x^{(i)})} \frac{q(x^{(i)}; x')}{q(x'; x^{(i)})}.$$
- С вероятностью a (1, если $a \geq 1$) $x^{(i+1)} := x'$, иначе $x^{(i+1)} := x^{(i)}$.

Обсуждение

- Суть в том, что мы переходим в новый центр распределения, если примем очередной шаг.
- Получается этакий random walk, зависящий от распределения p^*.
- $\frac{q(x^{(i)};x')}{q(x';x^{(i)})}$ для симметричных распределений (гауссиана) равно 1, это просто поправка на асимметрию.
- Отличие от rejection sampling: если не примем, то не просто отбрасываем шаг, а записываем $x^{(i)}$ ещё раз.

Пример блуждания [Bishop]



Обсуждение

- Очевидно, что $x^{(i)}$ — отнюдь не независимы.
- Независимые сэмплы получаются только с большими интервалами.
- Поскольку это random walk, то если большая часть q сосредоточена в радиусе ϵ, а общий радиус p^* равен D, то для получения независимого сэмпла нужно будет минимум... сколько?
- Рассмотрим одномерное случайное блуждание, где на каждом шаге с вероятностью $1/2$ точка движется влево или вправо на единицу длины. Какое ожидаемое расстояние точки от нуля после T шагов?

Обсуждение

- Ответ на упражнение: ожидаемое расстояние будет $\sqrt{T}$.
- Значит, нам потребуется где-то $\left(\frac{D}{\epsilon}\right)^2$ шагов (и это оценка снизу).
- Хорошие новости: это верно для любой размерности. То есть времени надо много, но нет катастрофы при переходе к размерности 1000.

Когда размерность велика

- Когда размерность большая, можно не сразу все переменные изменять по $q(x'; x)$, а выбрать несколько распределений q_j, каждое из которых касается части переменных, и принимать или отвергать изменения по очереди.
- Тогда процесс пойдёт быстрее, чаще принимать изменения будем.

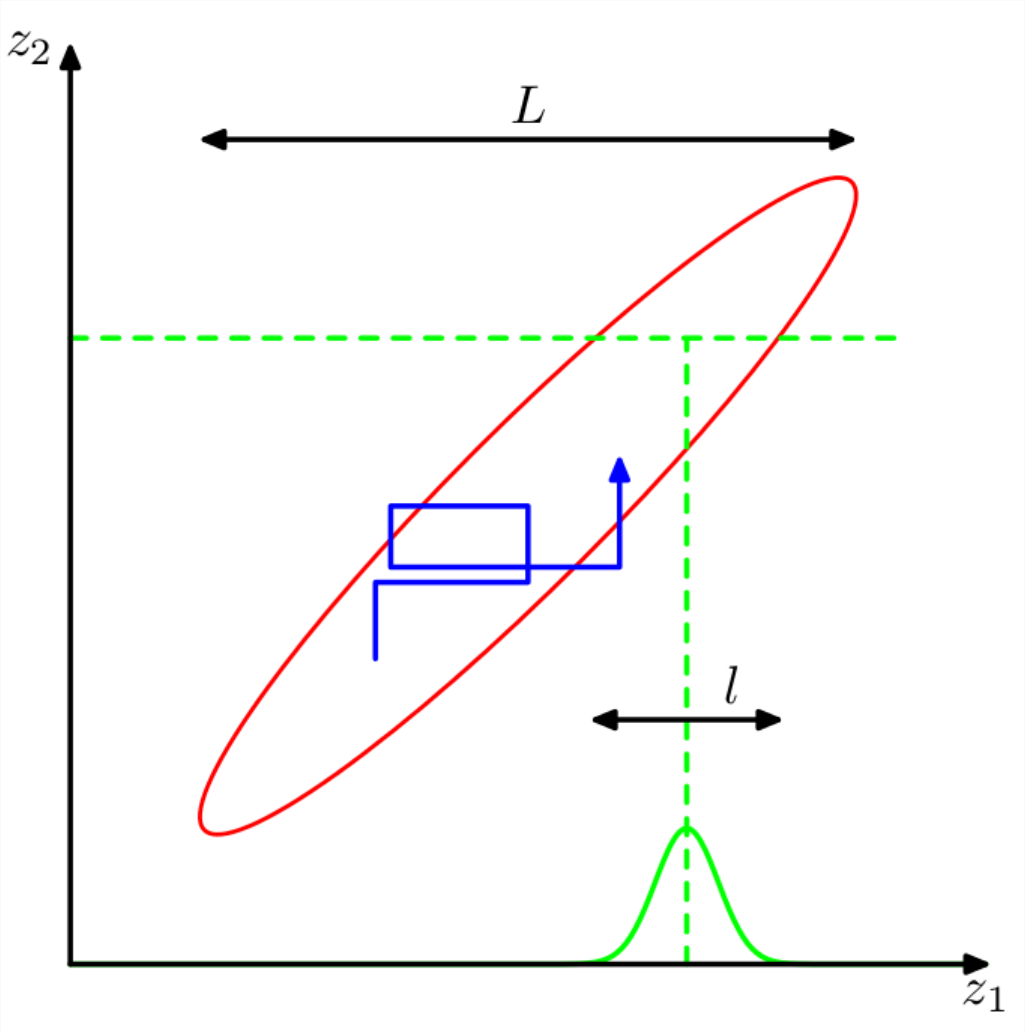
Идея сэмплирования по Гиббсу

- Пусть размерность большая. Что делать?
- Давайте попробуем выбирать сэмпл не весь сразу, а покомпонентно.
- Тогда наверняка эти одномерные распределения окажутся проще, и сэмпл мы выберем.

На двух переменных

- Пусть есть две координаты: x и y. Начинаем с (x^0, y^0).
- Выбираем x^1 по распределению $p(x|y = y^0)$.
- Выбираем y^1 по распределению $p(y|x = x^1)$.
- Повторяем.

Пример [Bishop]



Общая схема

- В общем виде всё то же самое: x_i^{t+1} выбираем по распределению

$$p(x_i | x_1^{t+1}, \dots, x_{i-1}^{t+1}, x_{i+1}^t, \dots, x_n^t)$$

  и повторяем.
- Это частный случай алгоритма Метрополиса (для распределений $q(\mathbf{x}'; \mathbf{x}) = p(x_i' \mid \mathbf{x}_{-i})$, и вероятность принятия получится 1 – упражнение).
- Поэтому сэмплирование по Гиббсу сходится, и, так как это тот же random walk по сути, верна та же квадратичная оценка.

Обсуждение

- Нужно знать $p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$. Это, например, особенно легко знать в байесовских сетях.
- Как будет работать сэмплирование по Гиббсу в байесовской сети?
- Для сэмплирования по Гиббсу не нужно никаких особенных предположений или знаний. Можно быстро сделать работающую модель, поэтому это очень популярный алгоритм.
- В больших размерностях может оказаться эффективнее сэмплить по несколько переменных сразу, а не по одной.

Сэмплирование по Гиббсу

Условие баланса

- Кратко напомним алгоритм Метрополиса-Гастингса
- Свойство баланса в марковских цепях: для p и T

$$\forall x, x' \quad T(x, x')p(x') = T(x', x)p(x).$$

- Т.е. вероятность того, что мы выберем x и дойдём до x', равна вероятности выбрать x' и дойти до x.
- Такие цепи называются *обратимыми* (reversible).
- Если выполняется условие баланса, то $p(x)$ — инвариантное распределение (докажите!).

Метрополис--Гастингс

- Очередная итерация начинается с состояния $x^{(i)}$.
- Выбрать x' по распределению $q(x'; x^{(i)})$.
- Вычислить
$$a(x', x) = \frac{p^*(x')}{p^*(x^{(i)})} \frac{q(x^{(i)}; x')}{q(x'; x^{(i)})}.$$
- С вероятностью $a(x', x)$ (1, если $a \geq 1$) $x^{(i+1)} := x'$, иначе $x^{(i+1)} := x^{(i)}$.

Метрополис--Гастингс

- Условие баланса:

$$p(x)q(x;x')a(x',x) = \min(p(x)q(x;x'), p(x')q(x';x)) =$$
$$= \min(p(x')q(x';x), p(x)q(x;x')) = p(x')q(x';x)a(x,x').$$

- Важный параметр – дисперсия распределения q; она задаёт баланс между частым принятием и быстрым перемещением по пространству состояний.

Обсуждение

- Очевидно, что $x^{(i)}$ — отнюдь не независимы.
- Независимые сэмплы получаются только с большими интервалами.
- Поскольку это random walk, то если большая часть q сосредоточена в радиусе ϵ, а общий радиус p^* равен D, то для получения независимого сэмпла нужно будет минимум... сколько?
- Рассмотрим одномерное случайное блуждание, где на каждом шаге с вероятностью $1/2$ точка движется влево или вправо на единицу длины. Какое ожидаемое расстояние точки от нуля после T шагов?

Обсуждение

- Ответ на упражнение: ожидаемое расстояние будет $\sqrt{T}$.
- Значит, нам потребуется где-то $\left(\frac{D}{\epsilon}\right)^2$ шагов (и это оценка снизу).
- Хорошие новости: это верно для любой размерности. То есть времени надо много, но нет катастрофы при переходе к размерности 1000.

Когда размерность велика

- Когда размерность большая, можно не сразу все переменные изменять по $q(x'; x)$, а выбрать несколько распределений q_j, каждое из которых касается части переменных, и принимать или отвергать изменения по очереди.
- Тогда процесс пойдёт быстрее, чаще принимать изменения будем.

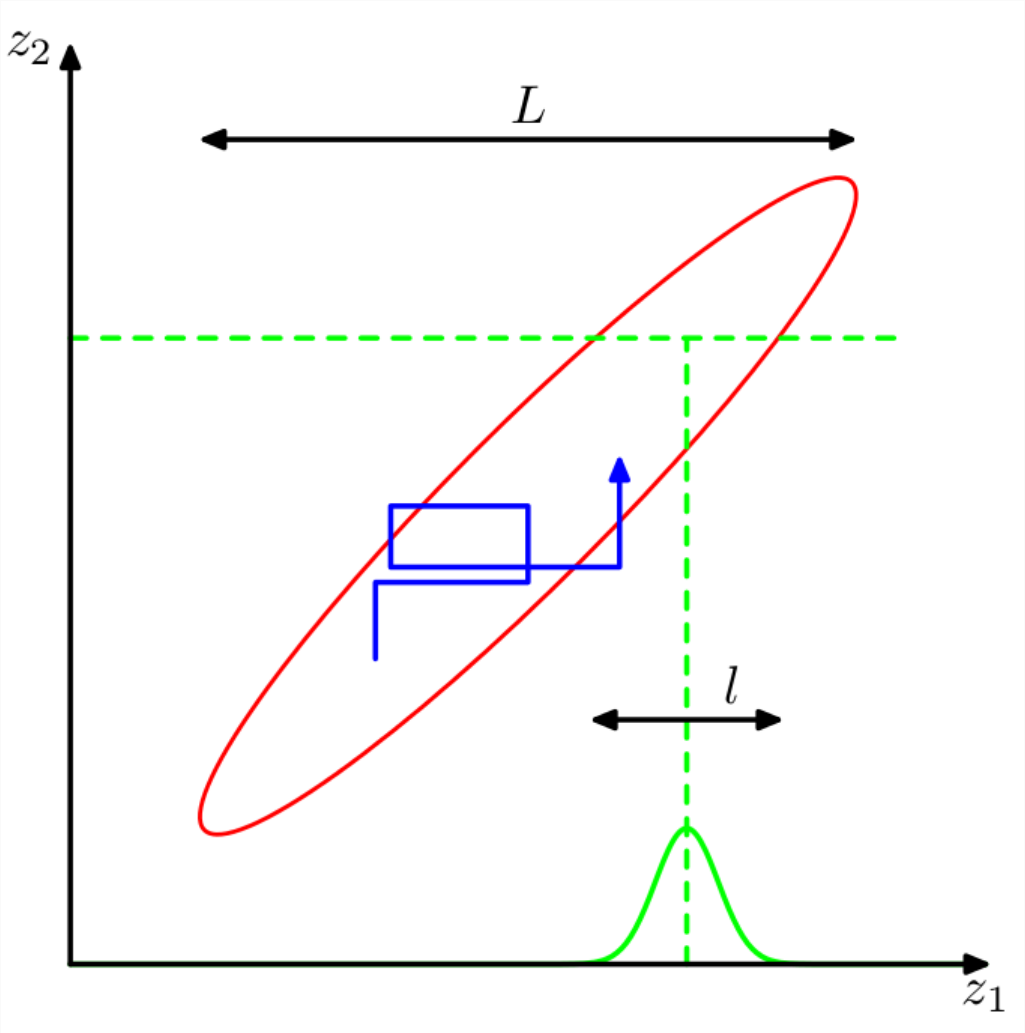
Идея сэмплирования по Гиббсу

- Пусть размерность большая. Что делать?
- Давайте попробуем выбирать сэмпл не весь сразу, а покомпонентно.
- Тогда наверняка эти одномерные распределения окажутся проще, и сэмпл мы выберем.

На двух переменных

- Пусть есть две координаты: x и y. Начинаем с (x^0, y^0).
- Выбираем x^1 по распределению $p(x|y = y^0)$.
- Выбираем y^1 по распределению $p(y|x = x^1)$.
- Повторяем.

Пример [Bishop]



Общая схема

- В общем виде всё то же самое: x_i^{t+1} выбираем по распределению

$$p(x_i | x_1^{t+1}, \dots, x_{i-1}^{t+1}, x_{i+1}^t, \dots, x_n^t)$$

  и повторяем.
- Это частный случай алгоритма Метрополиса (для распределений $q(\mathbf{x}'; \mathbf{x}) = p(x_i' \mid \mathbf{x}_{-i})$, и вероятность принятия получится 1 – упражнение).
- Поэтому сэмплирование по Гиббсу сходится, и, так как это тот же random walk по сути, верна та же квадратичная оценка.

Обсуждение

- Нужно знать $p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$. Это, например, особенно легко знать в байесовских сетях.
- Как будет работать сэмплирование по Гиббсу в байесовской сети?
- Для сэмплирования по Гиббсу не нужно никаких особенных предположений или знаний. Можно быстро сделать работающую модель, поэтому это очень популярный алгоритм.
- В больших размерностях может оказаться эффективнее сэмплить по несколько переменных сразу, а не по одной.

Slice sampling

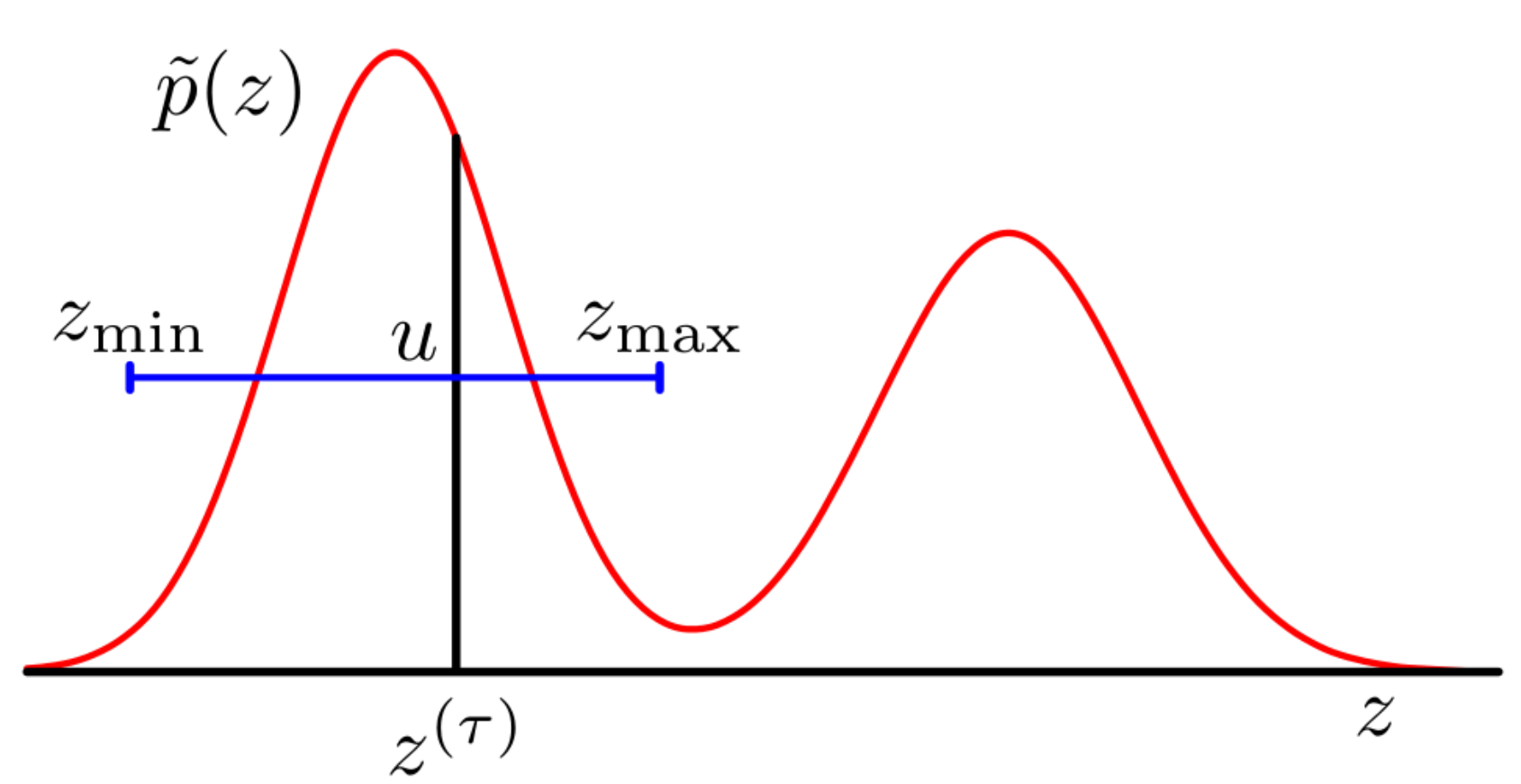
Суть

- Slice sampling — ещё один алгоритм, похожий на алгоритм Метрополиса.
- Это аналог алгоритма Метрополиса, но в нём мы хотим настраивать длину шага («дисперсию») автоматически.

Алгоритм в одномерном случае

- Мы хотим сделать random walk из одной точки под графиком p^* в другую точку под графиком p^*, да так, чтобы в пределе получилось равномерное распределение.
- Вот как будем делать переход $(x, u) \to (x', u')$:
  - Вычислим $p^*(x)$ и выберем u' равномерно из $[0, p^*(x)]$.
  - Сделаем горизонтальный интервал (x_l, x_r) вокруг x.
  - Затем будем выбирать x' равномерно из (x_l, x_r), пока не попадём под график.
  - Если не попадаем, модифицируем (x_l, x_r).
- Осталось понять, как сделать (x_l, x_r) и как его потом модифицировать.

Slice sampling



Дополнения к алгоритму

- Исходный выбор (x_l, x_r):
  - Выбрать r равномерно из $[0, \epsilon]$.
  - $x_l := x - r$, $x_r := x + (\epsilon - r)$.
  - Раздвигать границы на ϵ, пока $p^*(x_l) > u'$ и $p^*(x_r) > u'$.
- Модификация (x_l, x_r): Если x' лежит выше p^*, сокращаем интервал до x'.

Свойства

- В алгоритме Метрополиса нужно было выбирать размер шага. И от него всё зависело квадратично.
- А тут размер шага подправляется сам собой, и эта поправка происходит за линейное время (а то и логарифм).
- В задачах с большой размерностью нужно сначала выбрать (случайно или совпадающими с осями) направление изменения y, а потом проводить алгоритм относительно параметра α в распределении $p^*(x + \alpha y)$.

Спасибо за внимание!