СЭМПЛИРОВАНИЕ III

Сергей Николенко

СПбГУ — Санкт-Петербург

09 декабря 2020 г.

Random facts:

- 9 декабря в ООН — Международный день борьбы с коррупцией, а в России — День героев Отечества, памятная дата, которая будет способствовать «формированию в обществе идеалов самоотверженного и бескорыстного служения Отечеству»
- 9 декабря 536 г. — остготский гарнизон покинул Рим, в город вошёл Велисарий, и так после 60 лет варварского владычества Рим снова стал римским
- 9 декабря — большой день для Михаила Глинки: 9 декабря 1836 г. прошла первая постановка оперы «Жизнь за царя», а 9 декабря 1842 г. — первая постановка оперы «Руслан и Людмила»
- 9 декабря 1953 г. компания «General Electric» объявила об увольнении всех сотрудников-коммунистов
- 9 декабря 1968 г. Дуглас Энгельбарт впервые публично продемонстрировал изобретённые в его лаборатории компьютерную мышь и гипертекст

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.

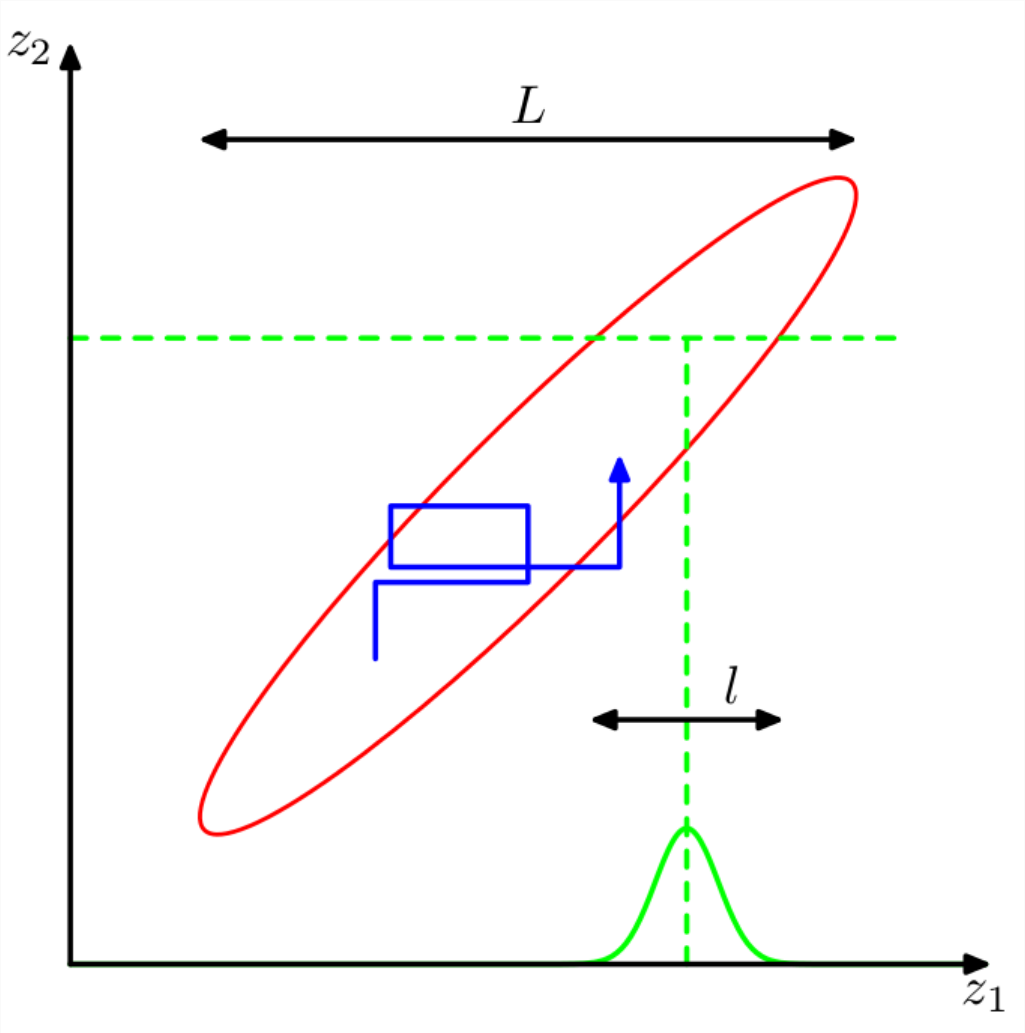
Идея сэмплирования по Гиббсу

- 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

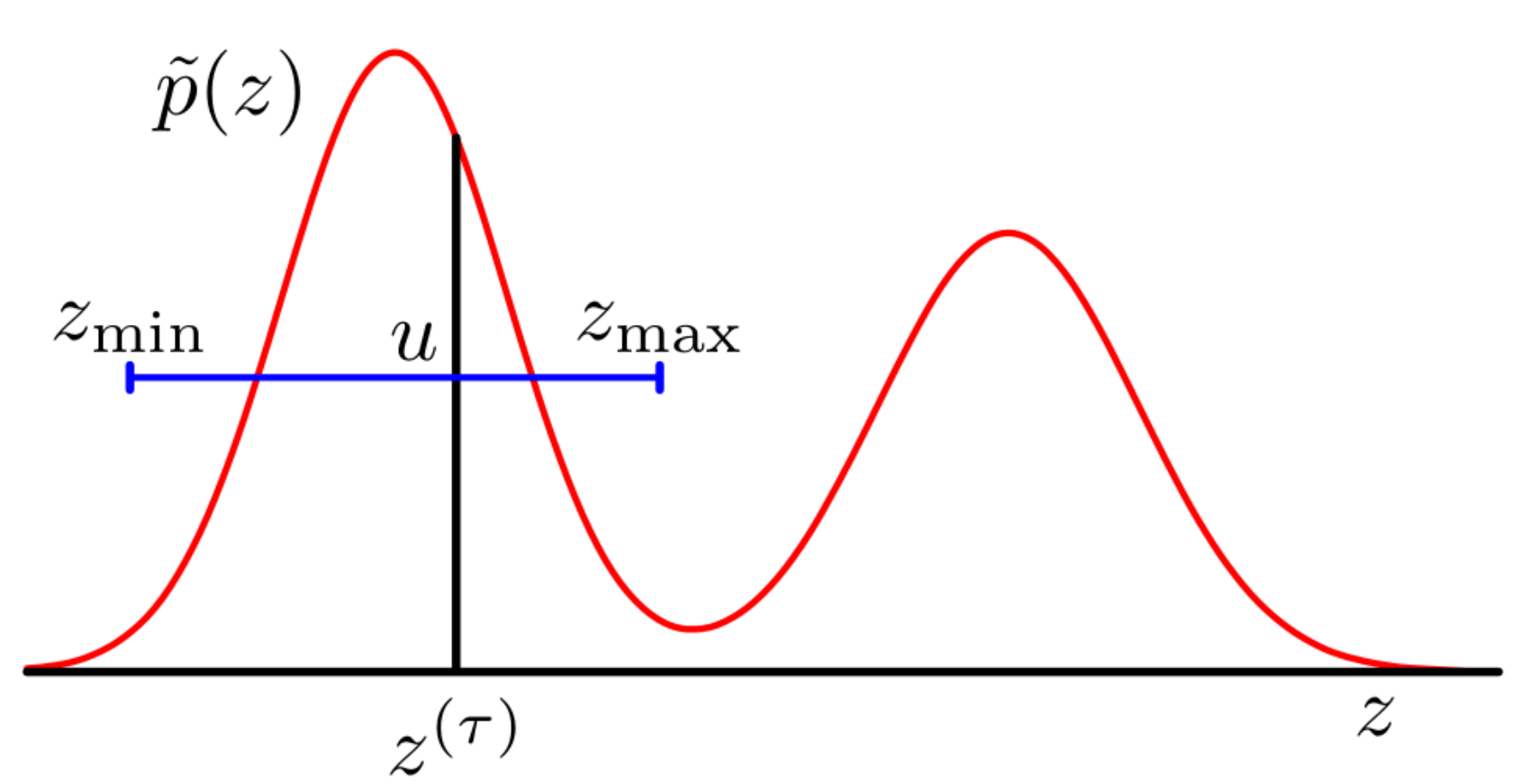
Суть

- 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 $p^*(x + \alpha y)$.

Сэмплирование и гамильтонова механика

Идея

- Рассмотрим ситуацию, когда вероятность можно записать как $p(x) = \frac{1}{Z} e^{-E(x)}$.
- Во многих таких случаях можно вычислить не только $E(x)$, но и градиент $\nabla E(x)$.
- Такую информацию хотелось бы использовать.

Гамильтонова механика

- Займёмся матфизикой: рассмотрим механическую систему.
- Состояние системы описывается обобщёнными координатами q и обобщёнными моментами p (векторные переменные).
- Её общая энергия $H(q, p, t) = V(q, t) + K(p, t)$, где V — потенциальная, K — кинетическая.

Гамильтонова механика

- Тогда система будет описываться гамильтоновыми уравнениями
$$\dot{p} = -\frac{\partial H}{\partial q}, \qquad \dot{q} = \frac{\partial H}{\partial p}.$$
- Гамильтонова механика — это, конечно, то же самое, что лагранжева, но вместо уравнений второго порядка на n переменных получаются уравнения первого порядка на $2n$ переменных.
- Важные для нас свойства: в течение эволюции системы
  1. значение гамильтониана H остаётся постоянным;
  2. объём любой области в пространстве переменных (p, q) сохраняется.

Суть

- Гамильтонов метод Монте-Карло — это вариация метода Метрополиса.
- Пространство поиска x расширяется *моментами* p.
- Благодаря законам сохранения гамильтонова динамика оставляет постоянным совместное распределение $p(x, p)$; применяя эволюцию вдоль гамильтониана, можно ходить далеко по пространству состояний, не меняя распределение; а потом делать несколько «обычных» (гиббсовских, например) шагов, которые уже будут менять H.

Суть

- Введём *гамильтониан* $H(x, p) = E(x) + K(p)$, где K — кинетическая энергия, например $K(p) = \frac{p^T p}{2}$.
- Теперь блуждание осуществляется двумя способами: первый случайно блуждает по пространству моментов (по Гиббсу, например).
- А второй шаг пытается сэмплировать совместную вероятность

$$p_H(x, p) = \frac{1}{Z_H} e^{-H(x,p)} = \frac{1}{Z_H} e^{-E(x)} \frac{1}{Z_H} e^{-K(p)}.$$

- Потом можно будет просто отбросить K и получить сэмплы для $e^{-E(x)}$, потому что тут всё так хорошо разделяется.

Суть

- Мы хотим построить траекторию в пространстве (x, p), на которой H остаётся постоянным, а затем по методу Метрополиса либо принять, либо отклонить этот сэмпл.
- Понятно, что $\dot{x} = p$, а гамильтоновы уравнения нам говорят, что

$$\dot{p} = -\frac{\partial E(x)}{\partial x}.$$

Суть

- Осталось это проинтегрировать. Для этого можно использовать leapfrog technique приближённого интегрирования:

$$\begin{aligned} p_i(t+\tfrac{\tau}{2}) &= p_i(t) - \tfrac{\tau}{2}\,\frac{\partial E}{\partial x_i}\Big|_{x(t)}, \\ x_i(t+\tau) &= x_i(t) + \tfrac{\tau}{m_i} p_i(t+\tfrac{\tau}{2}), \\ p_i(t+\tau) &= p_i(t+\tfrac{\tau}{2}) - \tfrac{\tau}{2}\,\frac{\partial E}{\partial x_i}\Big|_{x(t+\tau)}. \end{aligned}$$

- Дополнительные «половинные» шаги позволяют добиться погрешности второго порядка по τ.

Суть

- Алгоритм делает m leapfrog шагов, потом по методу Метрополиса принимает или отвергает получившуюся точку (проекцию на x).
- То есть если мы можем подсчитывать ∇E, а не только E, мы можем включить эту информацию в наш random walk.
- В результате он будет двигаться более-менее в правильном направлении, и пройденное расстояние $\sqrt{n}$ превратится в n (доказывать уж не будем).

Спасибо!

Спасибо за внимание!