МСМС СЭМПЛИРОВАНИЕ И SIR МОДЕЛИ

Сергей Николенко

СПбГУ — Санкт-Петербург

25 марта 2025 г.

Random facts:

- 25 марта 421 г., по легенде, в день Благовещения Девы Марии римляне, спасшиеся от готов на пустынных островах болотистого побережья, основали Венецию
- 25 марта 1238 г. началась героическая оборона Козельска; за 7 недель монголы потеряли несколько тысяч человек, а затем убили всех козлян, включая грудных детей
- 25 марта 1604 г. Борис Годунов послал казачьего голову Гаврилу Писемского из Сургута и стрелецкого голову Василия Тыркова из Тобольска с заданием основать крепость на берегу реки Томи в татарской земле; так появился Томск
- 25 марта 1918 г. Рада Белорусской Народной Республики провозгласила независимость БНР; впрочем, части Красной армии заняли Минск уже в декабре
- 25 марта 1969 г. Джон Леннон и Йоко Оно начали акцию «В постели за мир» (Bed-In for Peace) против войны во Вьетнаме; семь дней молодожёны приглашали прессу и по 12 часов сидели в постели, призывая к миру
- 25 марта 1975 г. король Саудовской Аравии Фейсал ибн Абдул-Азиз Аль Сауд был убит своим племянником Фейсалом, который мстил за смерть своего брата, застреленного полицейским во время митинга против секуляризации и телевидения

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 $q(x'; x^{(t)})$.

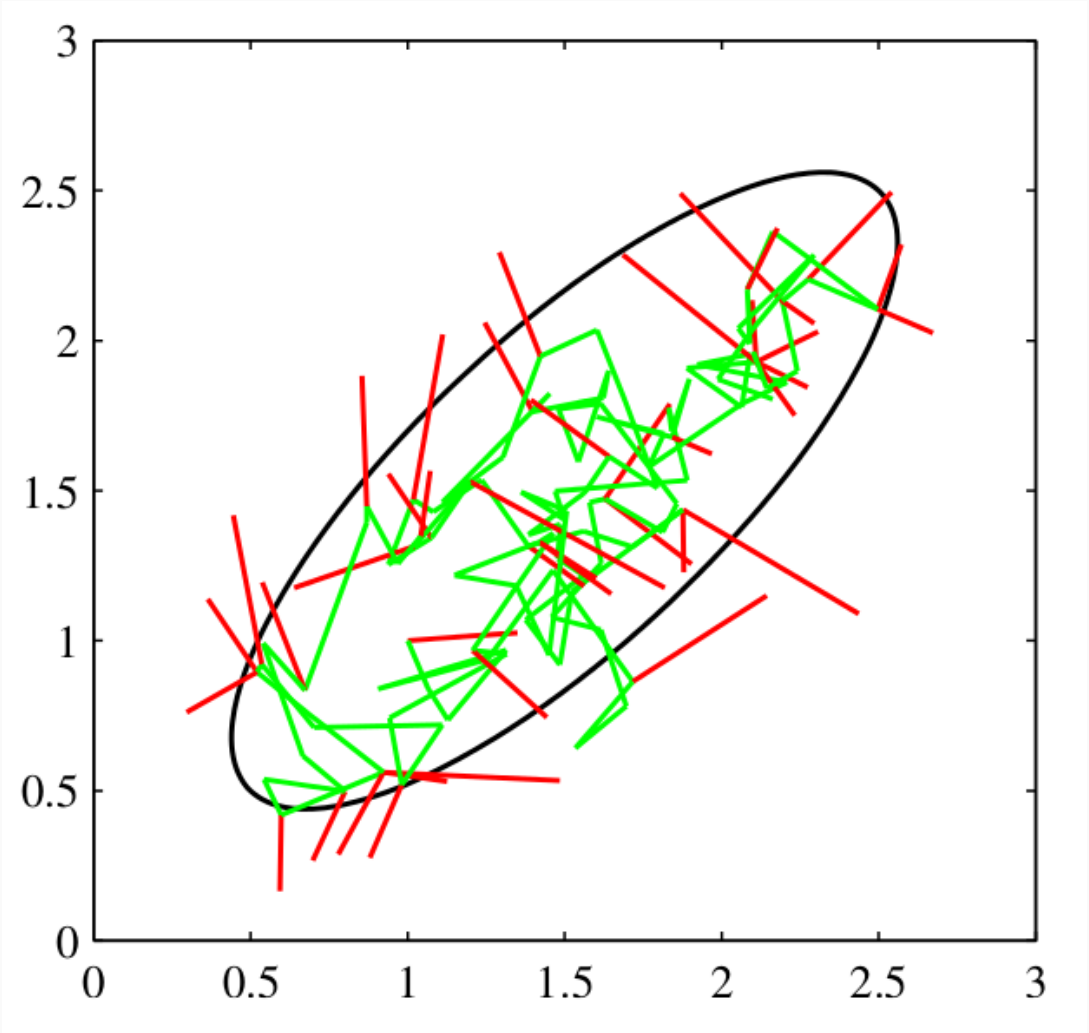
Алгоритм

- 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.

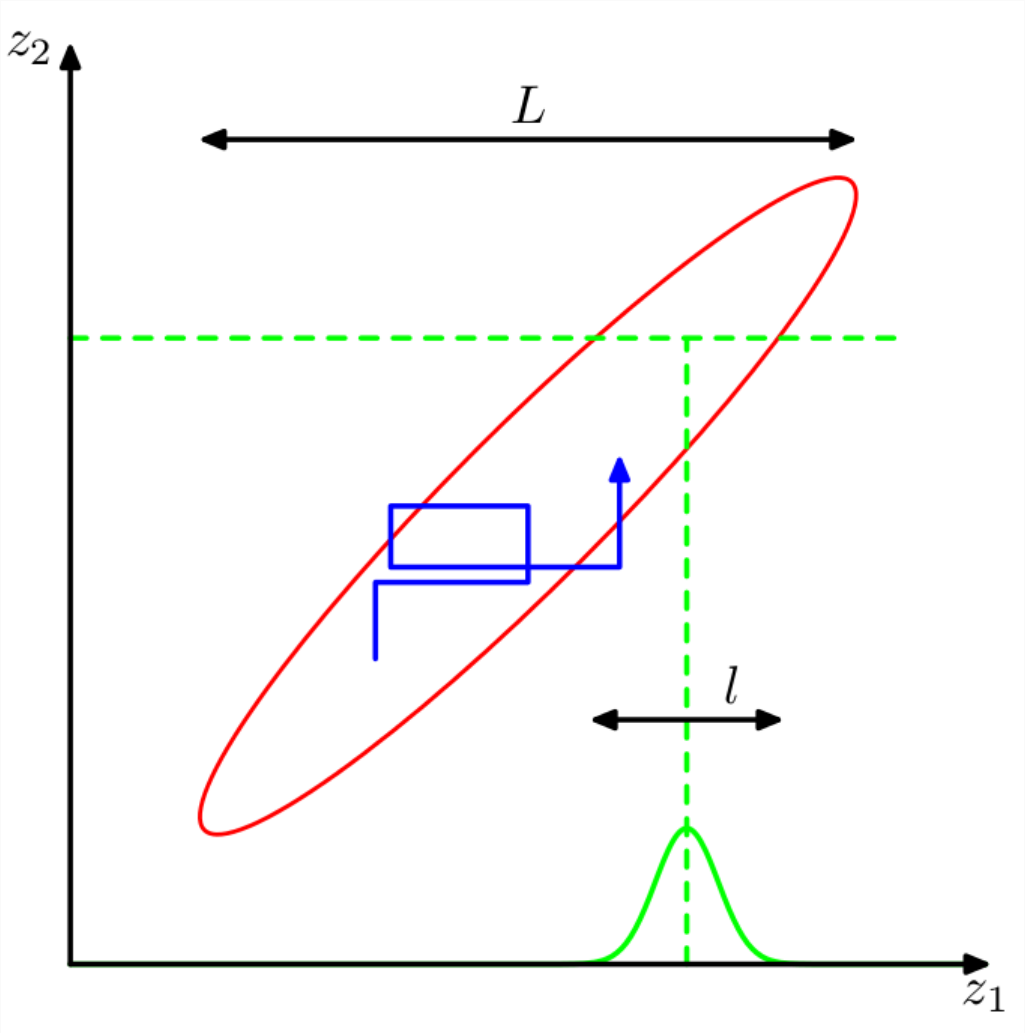
Идея сэмплирования по Гиббсу

- 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.

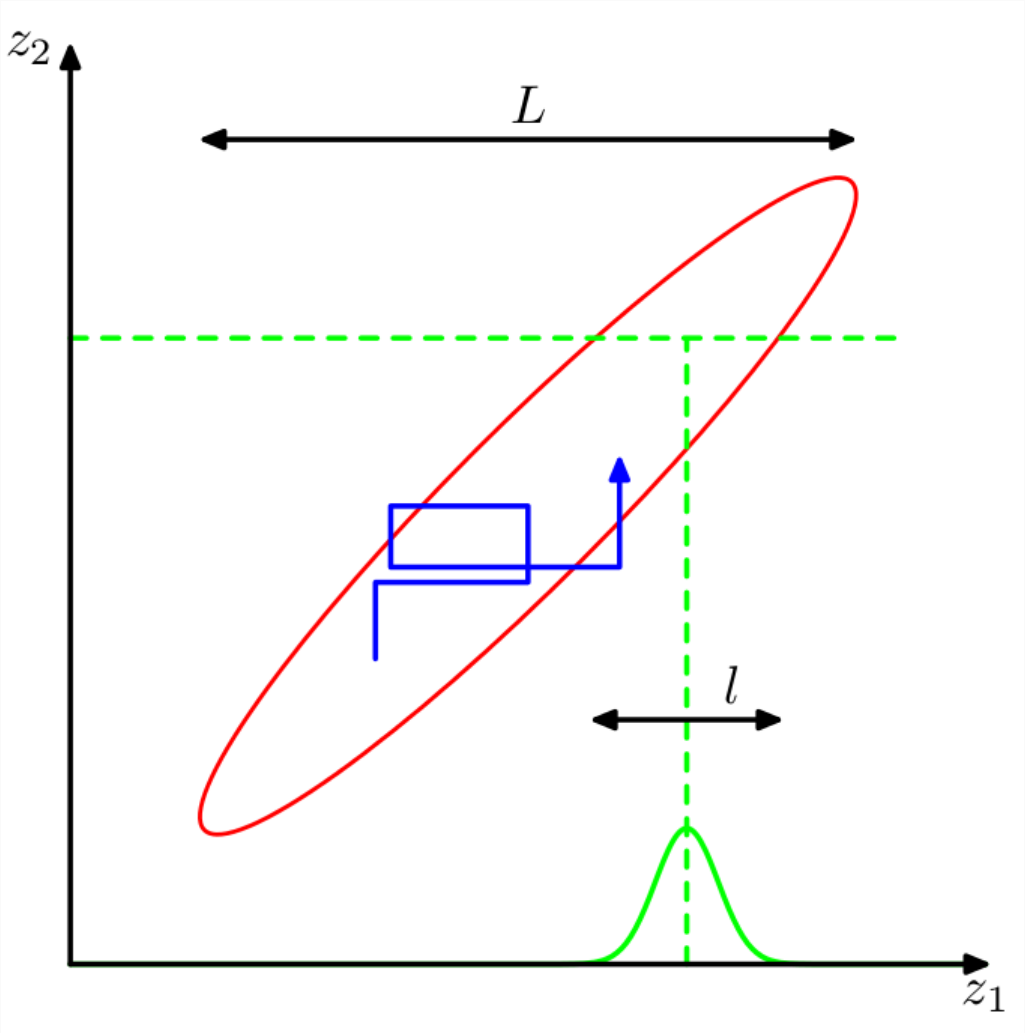
Идея сэмплирования по Гиббсу

- 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

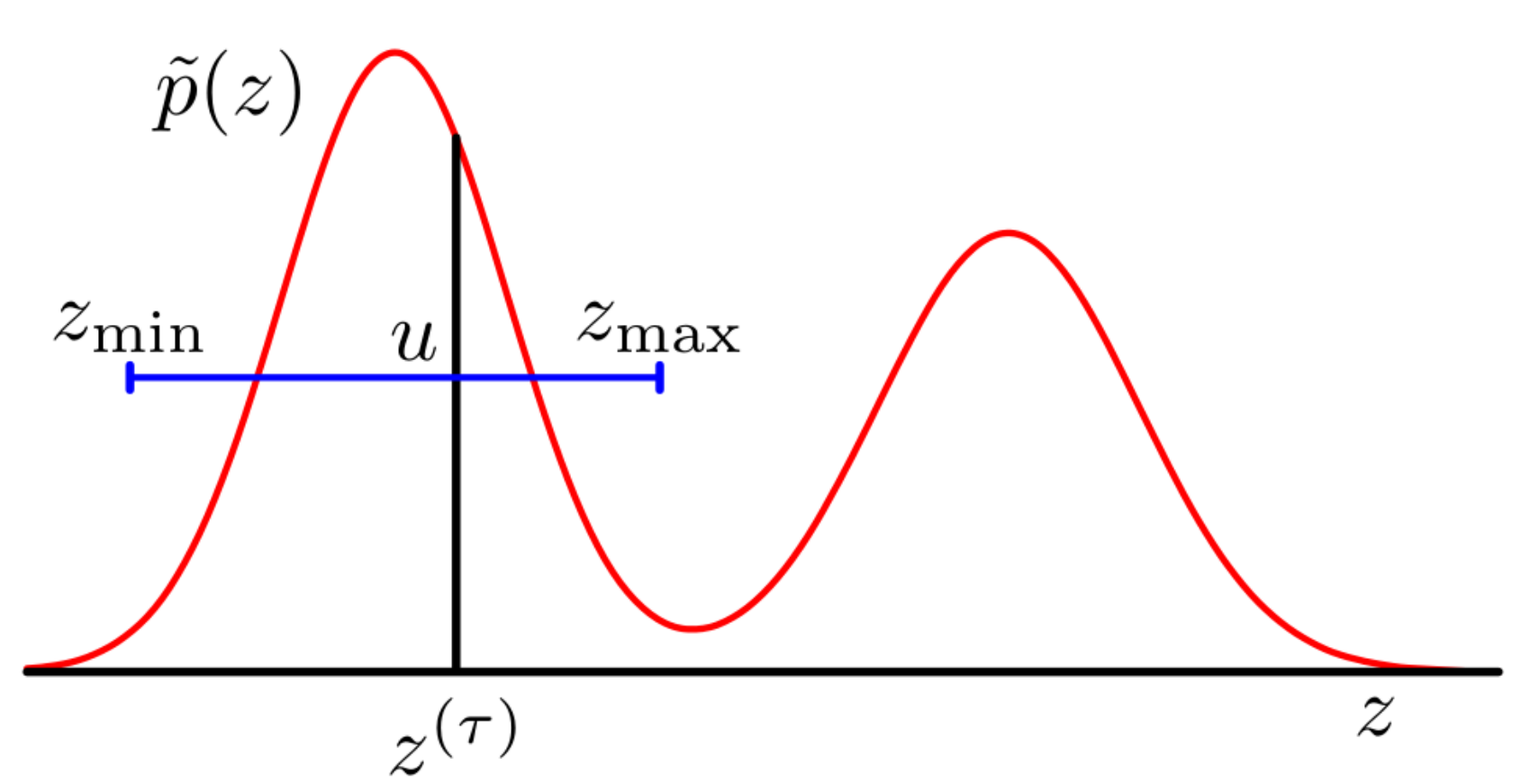
Суть

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

SIR-модели в эпидемиологии

SIR-модели

- И напоследок — конкретный (и весьма актуальный) пример
- Давайте попробуем применить то, о чём мы говорили, к эпидемиологии
- В модели SIR есть:
  - объекты (люди) $X = \{x_1, \dots, x_N\}$,
  - каждый эволюционирует между тремя состояниями $\mathcal{S} = \{S, I, R\}^N$;
  - S, I, R — ещё общее число объектов в соответствующих состояниях;
  - входные данные — число зарегистрированных случаев заболевания, изменяющееся во времени: $\mathbf{y} = \left(y^{(t)}\right)_{t=1}^{T}$.

SIR-модели

- Введём для каждого объекта *траекторию* (subject-path) $\mathbf{x}_j = \left(x_j^{(t)}\right)_{t=1}^{T}$, $j = 1, \dots, N$.
- Тогда и статистики изменяются во времени: $S^{(t)}$, $I^{(t)}$, $R^{(t)}$.
- Неизвестные параметры модели — это $\theta = \{\beta, \mu, \rho, \pi\}$:
  - π — начальное распределение заболевших, $x_j^{(1)} \sim \pi$;
  - ρ — вероятность обнаружить инфицированного в общей популяции, то есть вероятность того, что человек x_j в момент t, когда $x_j^{(t)} = I$, будет обнаружен тестированием и зачислен в данные $y^{(t)}$; тогда $y_t \mid I^{(t)}, \rho \sim \mathrm{Binom}(I^{(t)}, \rho)$;
  - μ — вероятность для заболевшего выздороветь, то есть вероятность перехода из состояния I в состояние R;
  - β — самый интересный параметр, вероятность заразиться за один отсчёт времени *от одного инфицированного человека*; будем предполагать самую простую модель, в которой вероятность заразиться от одного инфицированного равна β и все эти события независимы, а значит, вероятность остаться здоровым равна $(1 - \beta)^{I^{(t)}}$.

SIR-модели

- Обозначим вектор состояний всех людей, кроме x_j, через $\mathbf{x}_{-j}$ (и остальные величины так же).
- Вероятности перехода из $x_j^{(t-1)}$ в $x_j^{(t)}$:

$$
\begin{aligned}
p\left(x_j^{(t)} = S \middle| x_j^{(t-1)} = S, \mathbf{x}_{-j}^{(t-1)}\right) &= (1-\beta)^{I_{-j}^{(t-1)}}, \\
p\left(x_j^{(t)} = I \middle| x_j^{(t-1)} = S, \mathbf{x}_{-j}^{(t-1)}\right) &= 1-(1-\beta)^{I_{-j}^{(t-1)}}, \\
p\left(x_j^{(t)} = R \middle| x_j^{(t-1)} = I, \mathbf{x}_{-j}^{(t-1)}\right) &= \mu, \\
p\left(x_j^{(t)} = I \middle| x_j^{(t-1)} = I, \mathbf{x}_{-j}^{(t-1)}\right) &= 1-\mu, \\
p\left(x_j^{(t)} \middle| x_j^{(t-1)}, \mathbf{x}_{-j}^{(t-1)}\right) &= 0 \quad \text{во всех остальных случаях.}
\end{aligned}
$$

- Скрытые переменные — те же самые траектории $\mathbf{x}$ (не зря же мы их вводили).

SIR-модели

- Тогда полное правдоподобие $L(X, Y \mid \theta)$ получается как

$$\begin{aligned} L(X, Y \mid \theta) =& p(Y \mid X, \rho) p\left(X^{(1)} \middle| \pi\right) p\left(X \middle| X^{(1)}, \beta, \mu\right) \\ =& \left[\prod_{t=1}^{T} \binom{I^{(t)}}{y^{(t)}} \rho^{y^{(t)}} (1-\rho)^{I^{(t)} - y^{(t)}}\right] \times \\ & \times \left[\pi_S^{S^{(1)}} \pi_I^{I^{(1)}} \pi_R^{R^{(1)}}\right] \cdot \left[\prod_{t=2}^{T} \prod_{j=1}^{N} p\left(x_j^t \middle| \mathbf{x}_{-j}^{t-1}, \theta\right)\right], \end{aligned}$$

  где $p\left(x_j^t \middle| \mathbf{x}_{-j}^{t-1}, \theta\right)$ определено матрицей вероятностей переходов.
- Апостериорное распределение, которое нам нужно:

$$p\left(\theta | Y\right) \propto p(\theta) p\left(Y | \theta\right) = \int L(Y \mid X, \theta) p\left(X | \theta\right) p(\theta) dX,$$

  и этот интеграл, конечно, никак не подсчитать. Что же делать?

SIR-модели

- На помощь приходит алгоритм Метрополиса-Гастингса, точнее, сэмплирование по Гиббсу.
- Будем сэмплировать траектории $\mathbf{x}_j$ последовательно, зафиксировав все остальные $\mathbf{x}_{-j}$, данные $\mathbf{y}$ и параметры модели θ:

$$\mathbf{x}_j \sim p\left(\mathbf{x}_j \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right).$$

- Для этого нужно сначала понять, как выглядит распределение на траектории $\mathbf{x}_j$.
- Очевидно, её элементы $x_j^{(t)}$ нельзя считать независимыми, ведь человек проходит цепочку состояний $S \to I \to R$ только один раз и слева направо (если проходит вовсе). Всё это на первый взгляд опять выглядит сложно...

SIR-модели

- ...но здесь получается модель, которая нам уже хорошо знакома: последовательность случайных переменных $x_j^{(t)}$ образует марковскую цепь, а если добавить ещё известные нам данные, то получится скрытая марковская модель.
- Выбросим $\mathbf{x}_j$ из множества траекторий, получив статистики по всей остальной популяции $S_{-j}^{(t)}$, $I_{-j}^{(t)}$ и $R_{-j}^{(t)}$. Тогда параметры скрытой марковской модели таковы:
  - скрытые состояния $x_j^{(t)}$ с множеством возможных значений $\{S, I, R\}$;
  - матрица вероятностей перехода $p\left(x_j^t \middle| \mathbf{x}_{-j}^{t-1}, \theta\right)$, определённая выше;
  - наблюдаемые $\mathbf{y}$, вероятности получить которые зависят от того, заражён ли человек x_j в момент времени t:

$$p\left(y^{(t)} \middle| x_j^{(t)}\right) = \mathrm{Binom}\left(I_{-j}^{(t)} + \left[x_j^{(t)} = I\right], \rho\right).$$

SIR-модели

- Чтобы сэмплировать одну траекторию $\mathbf{x}_j$ при условии фиксированных остальных траекторий $\mathbf{x}_{-j}$, нужно сэмплировать траекторию вдоль скрытых состояний марковской модели.
- Здесь $\mathbf{x}_j$ будет эволюционировать от состояния S к состоянию R последовательно, с вероятностями перехода $\mathbf{x}_j$ на каждом шаге от S к R

$$p\left(x_j^{(t)} = I \middle| x_j^{(t-1)} = S, \mathbf{x}_{-j}\right) = 1 - (1-\beta)^{I_{-j}^{(t-1)}},$$

а вероятность перехода от I к R фиксирована и равна μ.

SIR-модели

- Стохастический алгоритм Витерби: два прохода по HMM слева направо и справа налево.
- На прямом проходе подсчитываем матрицы совместных вероятностей пар последовательных состояний

$$Q_j^{(t)} = \left(q^t_{j,s',s}\right)_{s',s\in\{S,I,R\}}, \quad \text{где}$$

$$q^t_{j,s',s} = p\left(x_j^{(t)} = s, x_j^{(t-1)} = s' \middle| Y, \mathbf{x}_{-j}, \theta\right).$$

- Фактически в нашей модели возможных пар таких состояний всего шесть (остальные переходы запрещены), и все матрицы Q выглядят как

$$Q_j^{(t)} = \begin{pmatrix} q^{(t)}_{j,S,S} & q^{(t)}_{j,S,I} & 0 \\ 0 & q^{(t)}_{j,I,I} & q^{(t)}_{j,I,R} \\ 0 & 0 & q^{(t)}_{j,R,R} \end{pmatrix}.$$

SIR-модели

- Чтобы вычислить $q^{(t)}_{j,s',s}$, нужно подсчитать

$$q^{(t)}_{j,s',s} = p\left(x_j^{(t)} = s, x_j^{(t-1)} = s' \middle| \mathbf{y}, \mathbf{x}_{-j}, \theta\right) \propto$$
$$\propto p\left(x_j^{(t-1)} = s' \middle| \mathbf{y}, \mathbf{x}_{-j}, \theta\right) p\left(x_j^{(t)} = s \middle| x_j^{(t-1)} = s', \mathbf{y}, \mathbf{x}_{-j}, \theta\right) \times$$
$$\times p\left(y_t \middle| x_j^{(t)} = s, \mathbf{y}, \mathbf{x}_{-j}, \theta\right) =$$
$$= \left[\sum_{s''} q^{(t-1)}_{j,s'',s'}\right] \cdot p\left(x_j^{(t)} = s \middle| x_j^{(t-1)} = s', \mathbf{x}_{-j}, \theta\right) \times$$
$$\times p_{\text{Binom}}\left(y^{(t)} \mid I^{(t)}_{-j} + \left[x_j^{(t)} = I\right], \rho\right),$$

  где $p\left(x_j^{(t)} = s \middle| x_j^{(t-1)} = s', \mathbf{x}_{-j}, \theta\right)$ — это те самые вероятности перехода в нашей модели, подсчитанные по статистикам $S^{(t-1)}_{-j}$, $I^{(t-1)}_{-j}$ и $R^{(t-1)}_{-j}$, а p_{Binom} — вероятность по биномиальному распределению.
- Потом нужно нормировать, учитывая, что $\sum_{s,s'} q^{(t)}_{j,s',s} = 1$.

SIR-модели

- Когда все матрицы $Q_j^{(t)}$ подсчитаны, их можно использовать для того, чтобы сэмплировать целые последовательности скрытых состояний. Для этого нужно разложить $p(\mathbf{x}_j \mid \mathbf{x}_{-j}, \mathbf{y}, \theta)$ не с начала времён, а с конца:

$$p\left(\mathbf{x}_j \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right) = p\left(x_j^{(T)} \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right) p\left(x_j^{(T-1)} \middle| x_j^{(T)}, \mathbf{x}_{-j}, \mathbf{y}, \theta\right) \times \ldots$$
$$\ldots \times p\left(x_j^{(2)} \middle| x_j^{(3)}, \ldots, x_j^{(T)}, \mathbf{x}_{-j}, \mathbf{y}, \theta\right) \times$$
$$\times p\left(x_j^{(1)} \middle| x_j^{(2)}, \ldots x_j^{(T)}, \mathbf{x}_{-j}, \mathbf{y}, \theta\right).$$

- И можно сэмплировать справа налево по матрицам Q.

SIR-модели

- Последнее состояние сэмплируется из сумм по строкам последней матрицы $Q_j^{(T)}$:

$$x_j^{(T)} \sim p\left(x_j^{(T)} = s \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right) = \\ = \sum_{s'} p\left(x_j^{(T)} = s, x_j^{(T-1)} = s' \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right) = \\ = \sum_{s'} q_{j,s',s}^{(T)}.$$

- А дальше достаточно, по марковскому свойству последовательности $\mathbf{x}_j$, сэмплировать при условии следующего состояния, то есть использовать распределение

$$x_j^{(t)} \sim p\left(x_j^{(t)} = s \middle| x_j^{(t+1)}, \mathbf{x}_{-j}, \mathbf{y}, \theta\right) \propto \\ \propto p\left(x_j^{(t)} = s, x_j^{(t+1)} = s' \middle| \mathbf{x}_{-j}, \mathbf{y}, \theta\right) = q_{j,s,s'}^{(t+1)}.$$

SIR-модели

- Так мы получим новую траекторию $\mathbf{x}_j$, и её можно подставить в X на место старой траектории и продолжать процесс сэмплирования: выбрать новый индекс j и повторить всё заново.
- В какой-то момент надо будет остановиться и обновить значения параметров.
- Теоретически можно даже сделать полноценный байесовский вывод, пересчитав параметры сопряжённых априорных распределений.
- Три основных параметра β, ρ и μ — это три монетки, а оставшийся параметр π — кубик с тремя гранями. Поэтому сопряжёнными априорными распределениями будут

$$\begin{aligned} p(\beta) &= \mathrm{Beta}\left(a_\beta, b_\beta\right), & p(\mu) &= \mathrm{Beta}\left(a_\mu, b_\mu\right), \\ p(\rho) &= \mathrm{Beta}\left(a_\rho, b_\rho\right), & p(\pi) &= \mathrm{Dir}\left(\mathbf{a}_\pi\right). \end{aligned}$$

SIR-модели

- Чтобы пересчитать их апостериорные значения, нужно аналогично обычным HMM подсчитать «статистику» того, сколько раз соответствующие монетки и кубики «бросали» и чем они «выпадали» в текущем наборе скрытых переменных (траекторий) X:
  - к параметрам $\mathbf{a}_\pi$ добавляются статистики того, в каких состояниях начинаются траектории:

$$a_{\pi,s} := a_{\pi,s} + \sum_{j=1}^{N} \left[x_j^{(1)} = s \right];$$

SIR-модели

- Чтобы пересчитать их апостериорные значения, нужно аналогично обычным HMM подсчитать «статистику» того, сколько раз соответствующие монетки и кубики «бросали» и чем они «выпадали» в текущем наборе скрытых переменных (траекторий) X:
  - параметры a_μ и b_μ обновляются в зависимости от того, каково было ожидаемое число переходов из состояния I в состояние R (выздоровлений) и сколько всего времени люди провели в состоянии I (проболели):

$$a_\mu := a_\mu + \sum_{t=1}^{T-1} \sum_{j=1}^{N} \left[x_j^{(t)} = I, x_j^{(t+1)} = R \right],$$

$$b_\mu := b_\mu + \sum_{t=1}^{T} I^{(t)} - \sum_{t=1}^{T-1} \sum_{j=1}^{N} \left[x_j^{(t)} = I, x_j^{(t+1)} = R \right].$$

SIR-модели

- Чтобы пересчитать их апостериорные значения, нужно аналогично обычным HMM подсчитать «статистику» того, сколько раз соответствующие монетки и кубики «бросали» и чем они «выпадали» в текущем наборе скрытых переменных (траекторий) X:
  - аналогично, параметры a_ρ и b_ρ получаются из статистики выявленных случаев, попавших в $\mathbf{y}$, по сравнению со случаями, которые оказались только в $I^{(t)}$:

$$a_\rho := a_\rho + \sum_{t=1}^{T} y^{(t)}, \quad b_\rho := b_\rho + \sum_{t=1}^{T} \left(I^{(t)} - y^{(t)} \right);$$

SIR-модели

- Параметры a_β и b_β самые интересные: нужно подсчитать ожидаемое число «возможностей заразиться», которые реализовались и не реализовались для всех людей в популяции:

$$p\left(x_j \text{ заразился при одном контакте} \middle| x_j \text{ заразился}\right) = \frac{\beta}{1-(1-\beta)^{I^{(t)}}},$$

а значит,

$$a_\beta := a_\beta + \sum_{t,j:\ x_j^{(t)}=S, x_j^{(t+1)}=I} \frac{\beta I^{(t)}}{1-(1-\beta)^{I^{(t)}}},$$

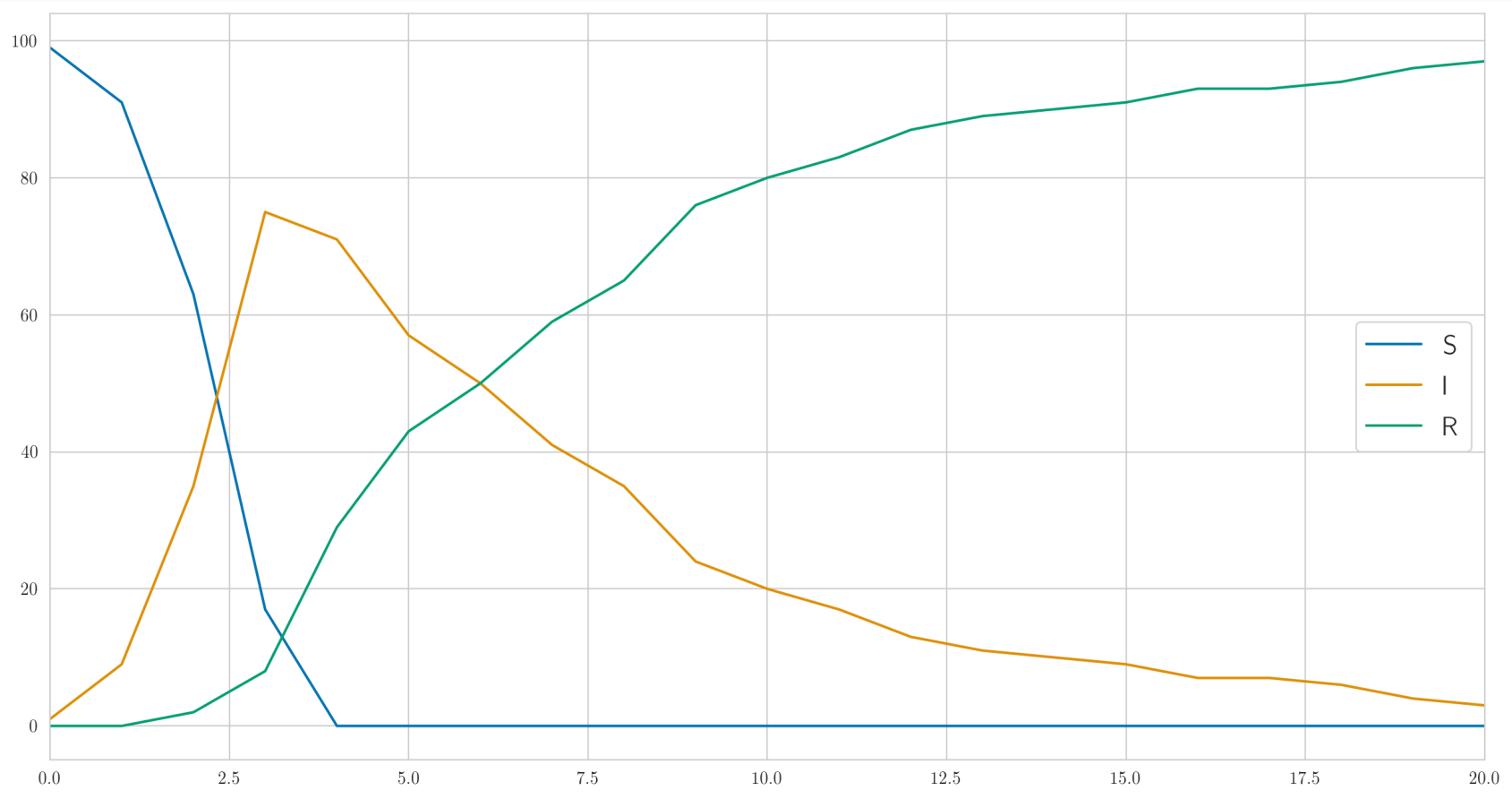
$$b_\beta := b_\beta + \sum_{t,j:\ x_j^{(t)}=S, x_j^{(t+1)}=S} I^{(t)} + \sum_{t,j:\ x_j^{(t)}=S, x_j^{(t+1)}=I} \left(I^{(t)} - \frac{\beta I^{(t)}}{1-(1-\beta)^{I^{(t)}}} \right)$$

SIR-модели

- Итого получили все компоненты нашей (сильно упрощённой!) SIR-модели: скрытые переменные в виде траекторий элементов популяции, алгоритм для сэмплирования по Гиббсу, который сэмплирует одну траекторию при условии всех остальных, и правила обновления параметров, которыми можно воспользоваться после того, как марковская цепь сэмплирования достаточно долго поработала.
- Давайте теперь посмотрим на практику...

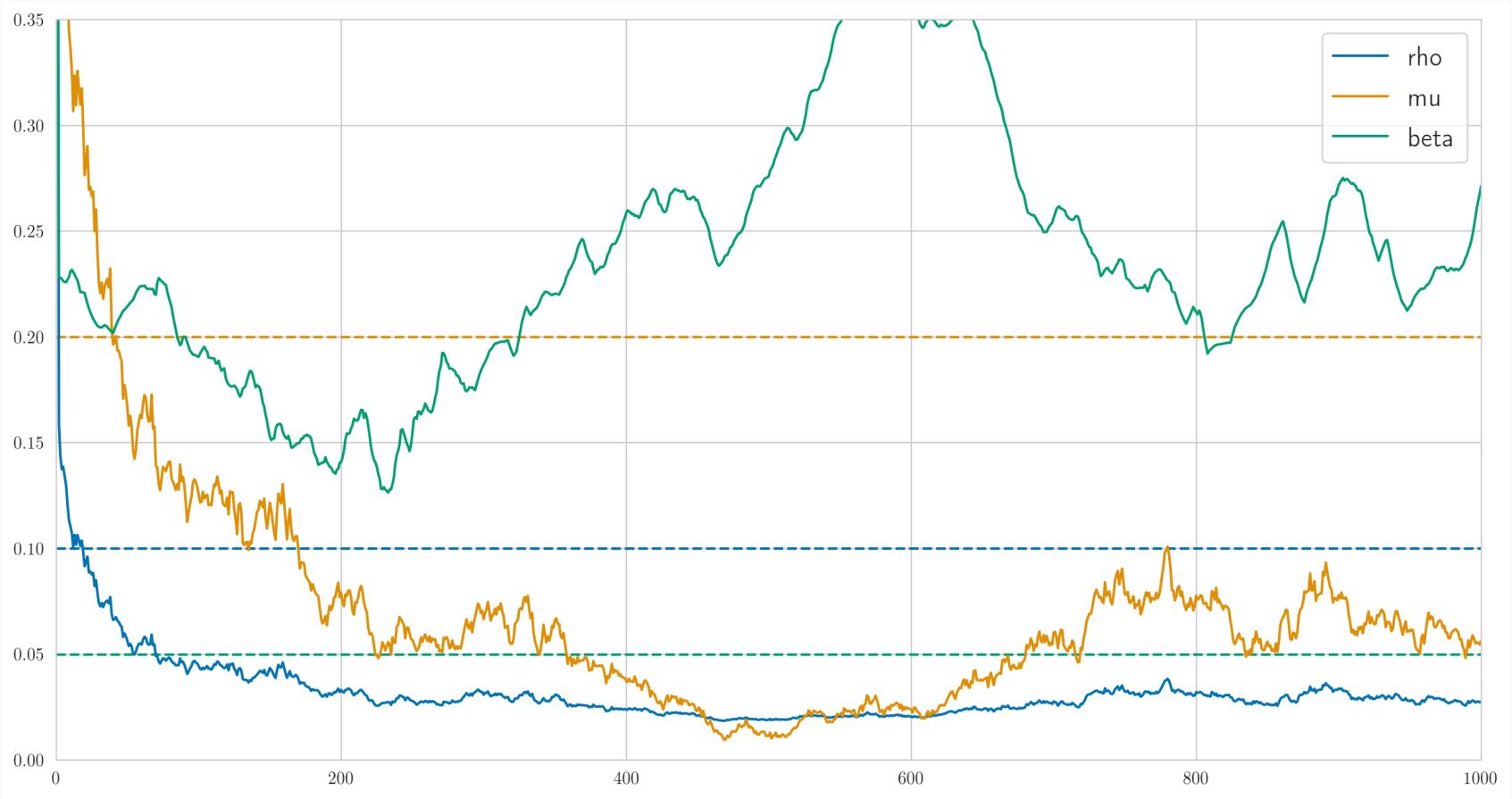
SIR-модели

- Пример визуализации статистик заражения при параметрах $N = 100$, $T = 20$, $\rho = 0.1$, $\beta = 0.05$, $\mu = 0.1$:



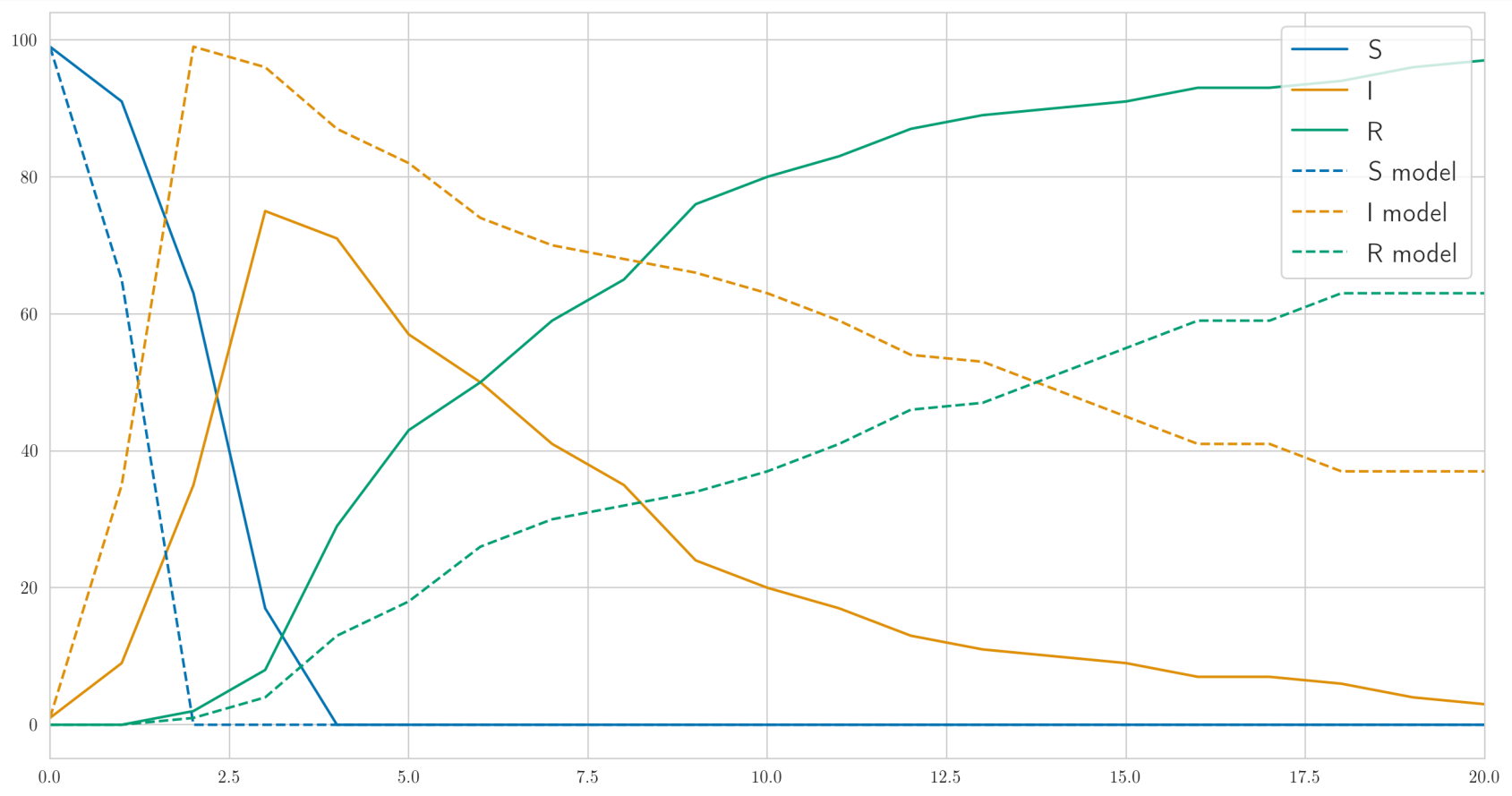
SIR-модели

- Пример обучения параметров модели SIR:



SIR-модели

- И если посэмплировать популяции из полученных параметров и из настоящих, получится совсем одно и то же:



- Какие выводы? Как это использовать на практике?

Спасибо!

Спасибо за внимание!