MH algo

$$p(x), q(x' \mid x)$$

$x' \sim q(x' \mid x_{t-1})$

$$x_t \begin{cases} x', & \mathbb{P} = \min\{1, \frac{p(x')q(x_{t-1}|x')}{p(x_{t-1})q(x'|x_{t-1})}\} \\ x_{t-1}, & 1 - \mathbb{P} \end{cases}$$

Всем нужно семплировать там, где нужны интегралы. Любой интеграл можно записать как мат. ожидание. А соответственно мо можно оценивать через семплирование.

Сила этого алгоритма — при достаточно мягких условиях на q мы получаем точные семплы из бяки-бяки $p(x)$.

В этом же основная слабость — очевидно, не для всех q мы получим одинаково эффективную схему.

Картинка

Random walk: $q(x_t \mid x_{t-1}) = \mathcal{N}(x_t \mid x_{t-1}, \sigma^2)$

Да, в бесконечности сойдется, но нам хотелось осмысленное количество итераций.

Этот вопрос волновал всех. Робертс, Розенталь показывают, как надо подбирать дисперсию random walk, даже при довольно больших размерностях. Но мы не хотим нейросети обучать random walk. Это работает, но фиг знает, насколько плохо. У нас есть градиент, который хотя бы помогает понять, куда ходить не надо. Вот, например, есть Ланжевен, который есть хитрый пропозал для MX, который это учитывает.

Мы можем маяться фигней, тк у нас есть коррекция MX.

Чтобы понять, какой пропозал хороший, какой плохой, надо придумать метрики.

Обычно мерят ESS (тоже зависит от задачи; если нужен интеграл, то мера точности — точность этого интеграла)

У нас есть акц и реджекты. Если мы отвергаем точку, то плотность в этой точке намного больше плотности в q.

Демо с хождением по нормальному распределению

Если мы будем реджектить в 99% случаев, то это плохо. Поэтому еще показатель — acceptance rate — мат. ожидание числа точек, которые мы приняли:

$$\text{AR} = \int p(x)q(x' \mid x) \min\left\{1, \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}\right\} dxdx'$$

Пусть мы всегда принимаем семплы. Максимум — единичка; логично. Почему?

$$p(x')q(x \mid x') = p(x)q(x' \mid x)$$
$$p(x') = \int dx p(x)q(x' \mid x)$$

Здесь есть некоторая проблема. Если мы всегда принимаем семплы, то пропозал может быть все равно очень плохим. Например, если подставить вместо q в дельта-функцию, то все плохо. AR $= 1$, но мы стоим на месте. Это нам никакой новой информации нам не дает. У него просто нет стационарного распределения, неэргодчный.

Будем рассматривать модификацию. Пусть $x' \sim q(x')$. Т.е. каждую точку мы будем выбирать независимо.

Для этого тоже можно записать AR:

$$\text{AR} = \int p(x)q(x') \min\{1, \frac{p(x)q(x')}{p(x')q(x)}\} dx dx'$$

Теперь это хорошая мера. Понятно, что если $p = q$, то все окей

Это такая модификация, которая позволяет сделать AR хорошей мерой.

Давайте охарактеризуем связь:

$$\text{AR} = 1 - \frac{1}{2} \int dx dx' |p(x)q(x' \mid x) - p(x')q(x \mid x')|$$

$$[\min\{a, b\} = \frac{a+b}{2} - \frac{|a-b|}{2}]$$

$$\text{AR} = 1 - \text{TV}(p(x)q(x' \mid x) \| p(x')q(x \mid x'))$$

И эта связь приводит к крутизне через связь KL и TV:

$$\geq 1 - \sqrt{\frac{1}{2}\text{KL}(p(x)q(x' \mid x) \| p(x')q(x \mid x'))}$$

То есть мы можем оптимизировать и KL.

Это можно записать и для independent пропозала:

$$\geq 1 - \sqrt{\frac{1}{2} KL(p(x)q(x') \| p(x')q(x)} =$$

$$1 - \sqrt{\frac{1}{2}(\text{KL}(p \| q) + \text{KL}(q \| p))}$$

(лучше TV оценивать не умеют. на практике — без разницы нам абсолютное отклонение. вся прелесть, что у них похожие градиенты. а это значит, что мы придем все равно в хорошую точку)

Давайте максимизировать AR. Мы можем по параметрам q оптимизировать AR.

Итак, мы хотим

$$-\mathbb{E}_{p(x), q(x')} l(\frac{p(x')q(x)}{p(x)q(x')}) \to \min_q$$

$l(.) \begin{cases} \min\{1, .\} \\ \log(.) \end{cases}$ если подставить лог, то получим симметризованный KL

Давайте с текущим q соберем немного семплов из МХ.

Buffer $\leftarrow \{x_i\} \quad x_i \sim p$

$\{x_i\} \sim$ Buffer

пропозал должен быть довольно экспрессивный. поэтому мы хотим взять нейросеть. кстати, потоки сюда идеально подходят.

Теперь сравним варвывод и потоки. Варвывод выучит 3-4 моды из всех, а потоки класс. Но немного покореженные, тк везде плотность не ноль. А если МХ, то он сделает красивые гауссианы (но пропозал тоже не оч хороший)

Также это можно применить к байесовскому выводу. Будем рассматривать симметризованный KL в качестве objective.

В байесовском выводе:

$$p(y_i \mid x_i, \theta) \quad p(\theta)$$

Хотелось бы уметь считать

$$p(y_i \mid x_i) = \mathbb{E}_{p(\theta|\mathcal{D})} p(y_i \mid x_i, \theta)$$

Но мы не умеем. Если мы умеем семплировать — все супер.

Пусть теперь целевое распределение будет целевым распределением МХ.

Будем приближать каким-то пропозалом:

$$\mathrm{KL}(q(\theta) \| p(\theta \mid \mathcal{D})) + \mathrm{KL}(p(\theta \mid \mathcal{D}) \| q(\theta))) \to \min$$
$$-\mathbb{E}_{q(\theta)} \log p(\mathcal{D} \mid \theta) + \mathrm{KL}(q(\theta) \| p(\theta)) - \mathbb{E}_{p(\theta|\mathcal{D}} \log q(\theta) \to \min$$
$$-\mathrm{ELBO} - \mathbb{E}_{p(\theta|\mathcal{D})} \log q(\theta) \to \min$$

Первая часть ELBO оценивается по минибатчам. Соответственно, и все остальное тоже, причем несмещенно. Мы можем оценивать минимум.

Чтобы посчитать качество, надо посчитать ESS. Это сколько из честного распределения нужно семплов, чтобы оценить с той же точностью, что и Ваш семплер. Это можно сравнить по ESS с другими семплерами, и это работает хорошо

Давайте рассмотрим другую задачу (не классическую задачу МЦМЦ), а классическую задачу, которую решают сейчас нейросети. То есть, распределения, заданные какими-то семплами. Да и из q тоже можем семлировать много, но не знаем плотность. А-ля GAN.

$$L = -\mathbb{E}_{p(x), q(x')} l(\frac{p(x')q(x)}{p(x)q(x')}) \to \min_q$$

Картинка с p, q, кроссэнтропией (как, когда объясняют про implicit и дискриминаторы)

$$L_D = -\mathbb{E}_{p(x)} \log D(x) - \mathbb{E}_{q(x)} \log(1 - D(x)) \to \min_D$$

$$D^*(x) = \frac{p(x)}{p(x) + q(x)}$$

Окей, придумали, как оценивать density ratio:

$$D^*(x) = \frac{p(x)}{p(x) + q(x)} \to \frac{D(x)}{1 - D(x)} = \frac{p(x)}{q(x)}$$

Тогда алгоритм следующий.
train D: $L_D \to \min$
train q: $L \to \min$

Мы получили GAN, но совершенно из других предположений. Подставим D :

$$-\mathbb{E}_{p(x),q(x')} \log \left(\frac{D(x')}{1 - D(x')} \frac{1 - D(x)}{D(x)} \right) \to \min_q$$
$$-\mathbb{E}_{p(x),q(x')} \log \left(\frac{D(x')}{1 - D(x')} \right) \to \min_q$$
$$-\left[\mathbb{E}_{q(x')} \log D(x') - \mathbb{E}_{q(x')} \log(1 - D(x')) \right] \to \min_q$$

Одно — лосс из генератора оригинального GAN'а. Другое — то, что на самом деле используют.

График $\log(1 - D)$.

Короче, просто получили GAN, но совершенно из других предположений. Типа сделаем MH, но в необычном сеттинге. А это GAN.

Понятно, что это работает так же, как GAN, но интересно другое. Давайте доведем идею до логического завершения: генерить картинки MX, а не генератором только. Типа фильтруем MX картинки из генератора. Тогда все будет много круче, чем было: IS поднимется, качество станет лучше и т.д.

(Inception Score: $\mathbb{E}_{x \sim q} \mathrm{KL}(p(y \mid x) \| p(y))$, $p(y) = \mathbb{E}_{x \sim q} p(y \mid x)$; картинки могут быть хреновые, но это оценивает не качество, а дайверсность, так что норм; как бы индикатор mode collapse)

Получается Implicit Metropolis-Hastings. Представьте, что p – реальные объекты: молекулы, траектории, а q — симулятор. Можно пофильтровать что правдоподобно, что нет. (куда сходится?).