Пару слов про сэмплирование

И. Куралёнок

СПб, 2017

Понятие сэмплирования

Сэмплирование — метод исследования множества путём анализа его подмножеств.

Применяется когда:

- множество слишком велико для перебора;
- каждое дополнительное измерение дорого;
- предварительный анализ.

Алгоритм сэмплирования

1. Понять какое множество мы изучаем
2. Осознать, что из этого множества мы можем измерить
3. Определить количество измерений
4. Разработать план сэмплирования
5. Провести сэмплирование

Типы сэмплирования

- Вероятностное сэмплирование:

$$p(x), \forall x : p(x) > 0$$

Например: попробуем посчитать соотношение мужчин/женщин

- Невероятностное сэмплирование:

$$p(x), \exists x : p(x) = 0$$

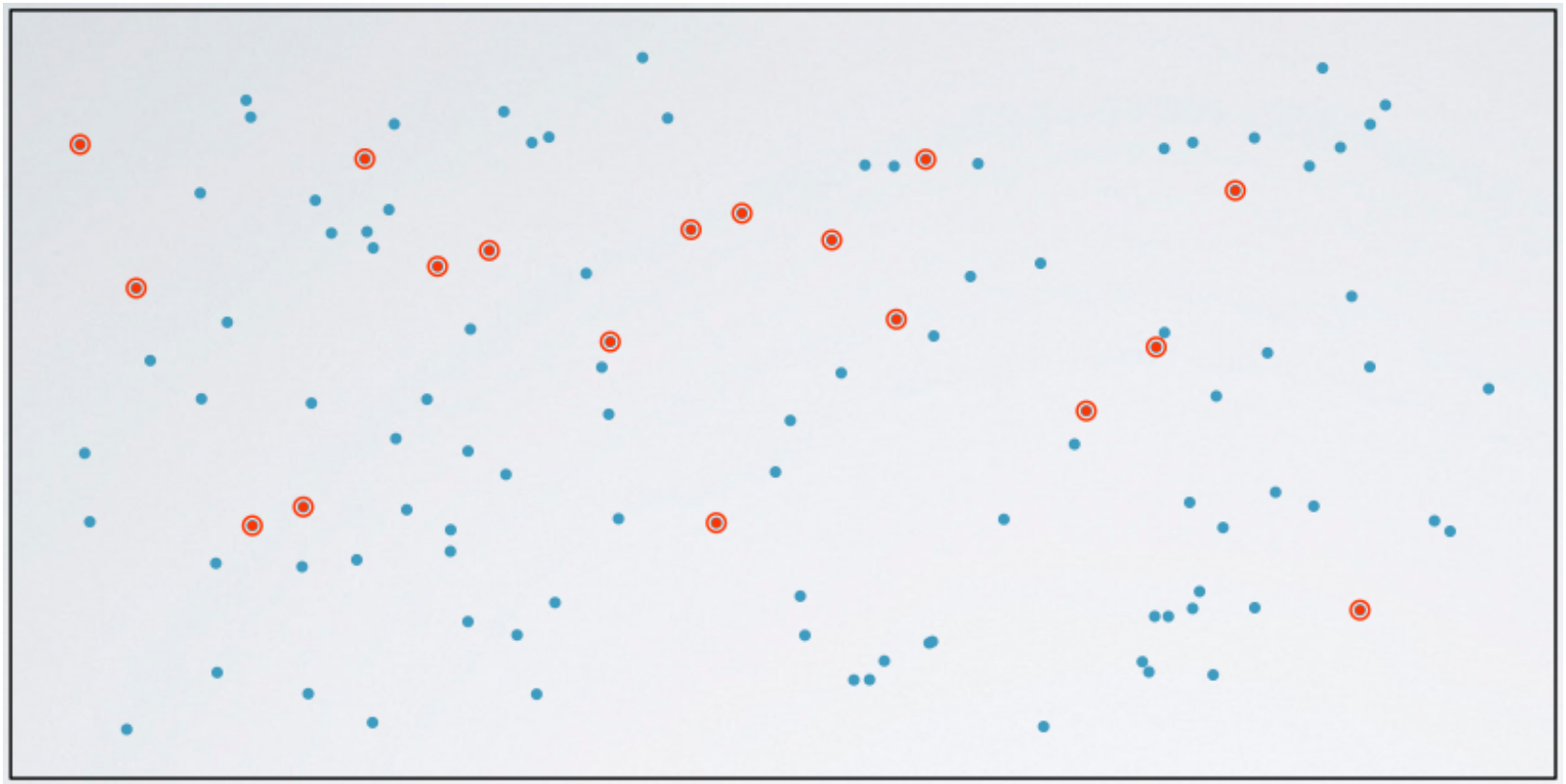
Например: "по результатам опроса superjob.ru, 100% россиян пользуются интернетом"

- Без возвращений
- С возвращениями

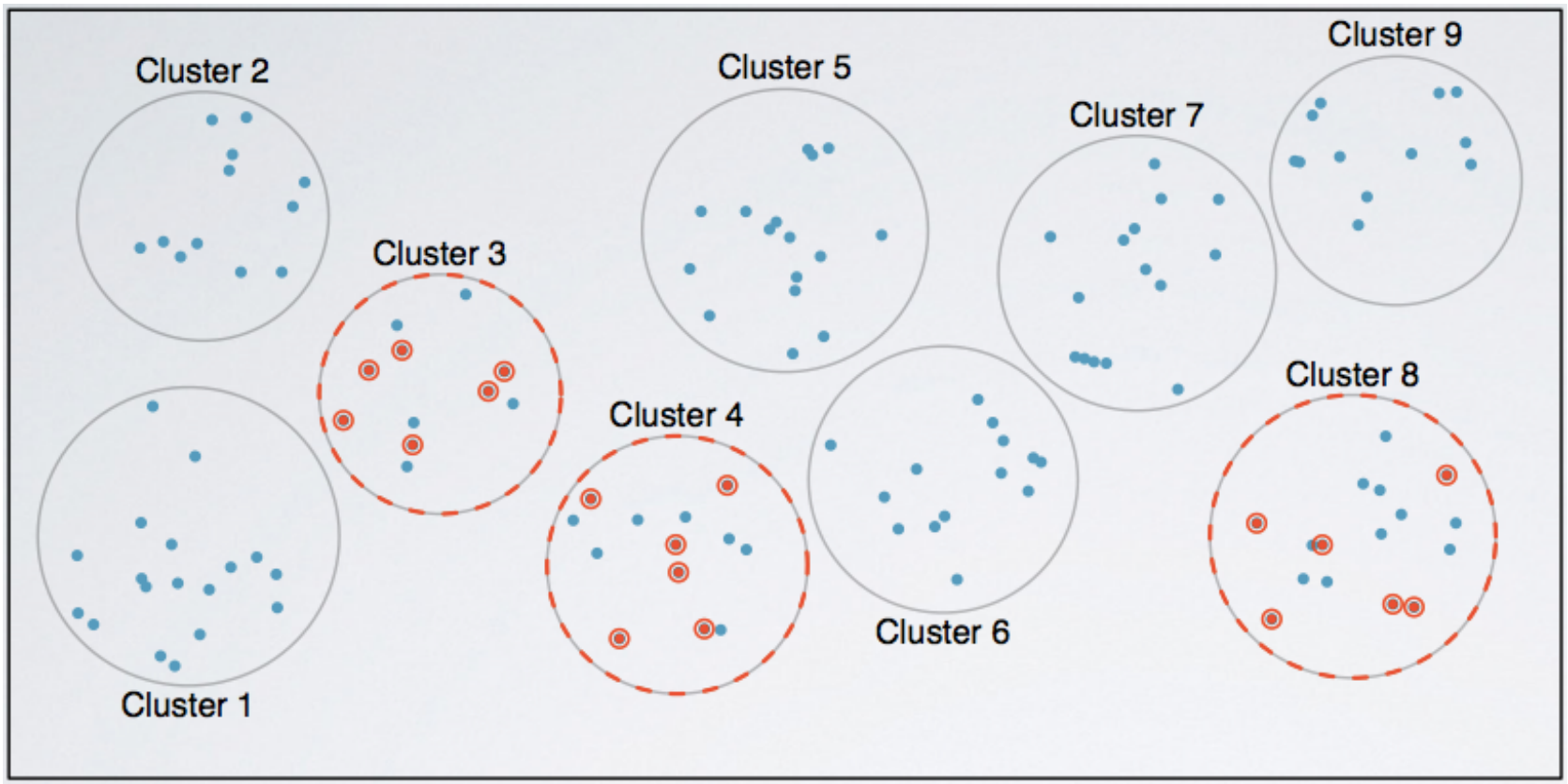
Виды сэмплирования

- Вероятностное сэмплирование
  - Простое вероятностное
  - Систематическое
  - Пропорциональное
  - Кластерное
  - Стратифицированное
- Невероятностное сэмплирование
  - Опрос ближайших
  - Панельное сэмплирование

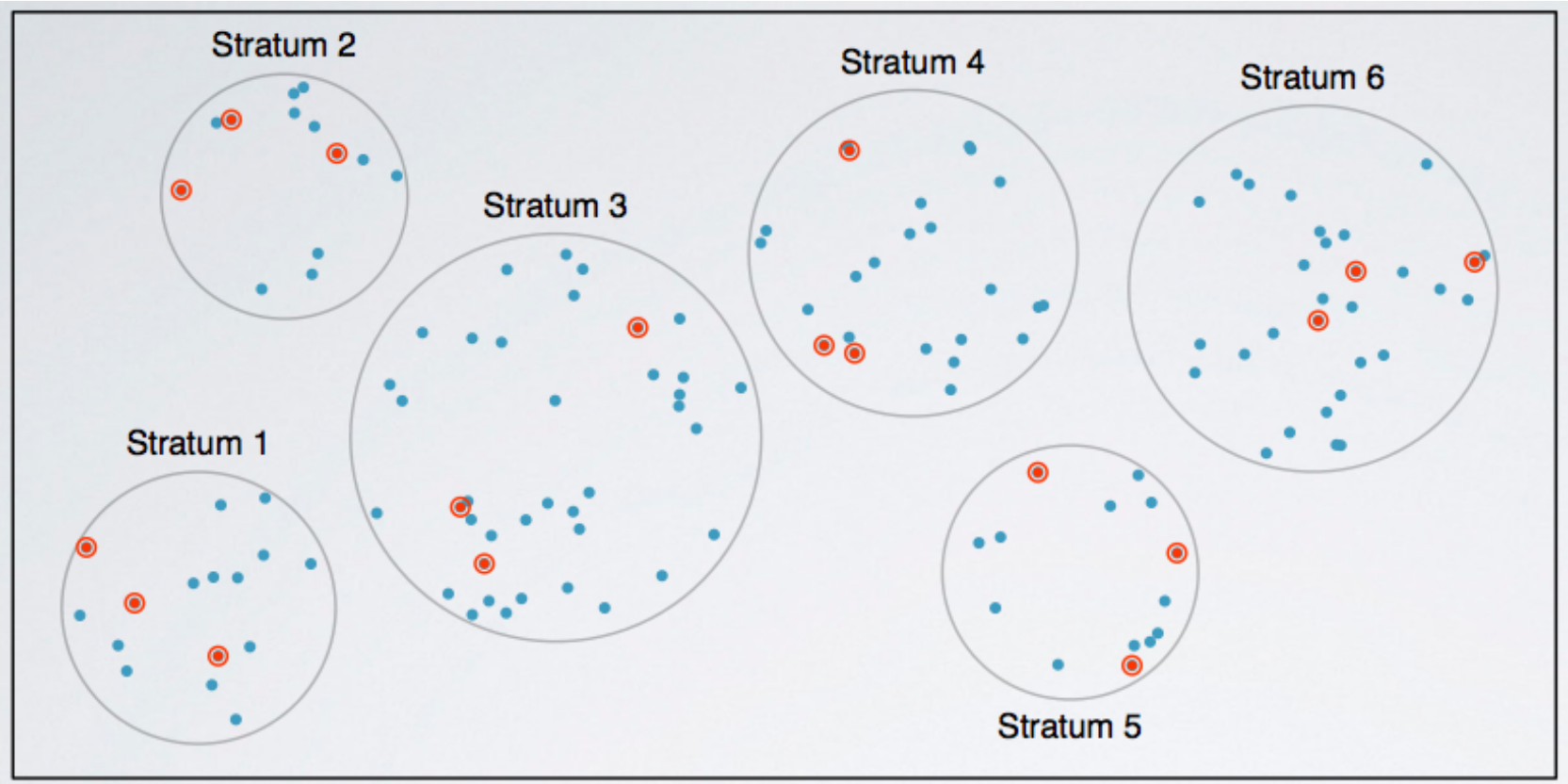
Простое вероятностное



Кластерное


Cluster 2
Cluster 5
Cluster 9
Cluster 7
Cluster 3
Cluster 8
Cluster 4
Cluster 6
Cluster 1

Стратифицированное



Как выбрать нужное?

Надо учитывать:

- природа и размер возможного сэмпла;
- наличие дополнительной информации об элементах;
- необходимая точность измерений;
- точность отдельных измерений в сэмплировании;
- стоимость измерений.

Как делать не нужно

Байки про ошибки в создании обучающей выборки

- Соотношение положительных и отрицательных примеров
- Все решения одинаково бесполезны
- Ошибка в данных больше разницы между методами
- Правда в глазах смотрящего (про ошибки в примерах)
- Нужно следить за распределением важных параметров, независимо от способа сбора данных
- Устаревшие данные
- Correlation vs. Causation
- Зависимость результата от одной точки или класса точек
- Закодировать ответ в DS

Создание DS — тоже оптимизация

Как же понять что построенное множество хорошее? Хотим такого:

$$\begin{aligned} F_0 = & \quad \arg\max_F \mu_{\xi\sim D} T(y_\xi, F(x_\xi)) \\ = & \quad \arg\max_F \mu_{\xi\sim \Gamma} T(y_\xi, F(x_\xi)) \end{aligned}$$

Но делать будем иначе:

$$D = \arg\max_D \mu_{\eta\sim\Gamma} T(y_\eta, \arg\max_F \mu_{\xi\sim D} T(y_\xi, F(x_\xi))(x_\eta))$$

Это такая оптимизация

Заключение о построении DS

Это все область выборочного контроля про которую много написано:
George E.P. Box, А. И. Орлов, В.П. Боровиков Создание хорошего обучающего множества — половина успеха. У меня нету универсальной схемы как такое делать, и есть ощущение, что это на грани искусства.

Выборка как генеральная совокупность

- LOO методы и jackknife оценки
- Bootstraping подход (Монте-Карло на выборке)
- Слабая аксиоматика Воронцова

Leave One Out

Выберем один из примеров и посмотрим насколько поменяется выборка, если его в ней не будет. Усредним по всем.

Эту логику можно применять например так:

$$Var_{LOO}(\{x_i\}) = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x}_{-i})^2 = \frac{1}{(m-1)^2}Var(\{x_i\})$$

Jackknife оценки

Пускай нас интересует статистика θ, $\hat{\theta}$—ее выборочная оценка и $\bar{\theta} = \frac{1}{m}\sum_i \hat{\theta}_{-i}$.

$$\begin{aligned} & \mathbb{E}(\hat{\theta}) = \theta + \frac{a}{m} + \frac{b}{m^2} + O(m^{-3}) \\ \Rightarrow \quad & \mathbb{E}(\hat{\theta} - \bar{\theta}) = \frac{a}{m(m-1)} + O(m^{-3}) \\ \Rightarrow \quad & b_{jack} = (m-1)(\bar{\theta} - \hat{\theta}) \end{aligned}$$

Bootstrapping

B. Efron “Bootstrap methods: another look at the jackknife”

Зачем ограничиваться одним примером? Устроим на нашей “генеральной совокупности” Монте-Карло:

1. рассматривая выборку как генеральную совокупность, породим повторную выборку X'
2. вычислим интересующую нас статистику $\hat{\theta}$ на X'
3. проделаем 1 и 2 достаточное количество раз и построим эмпирическое распределение θ
4. исследуем эмпирическое распределение и сделаем выводы про θ

Bootstrapping: простая реализация

1. случайно отсортируем исходную выборку
2. выбросим i равномерное от 1 до m
3. возьмем i-й пример в новую выборку
4. повторим п. 2 и 3 m раз
5. посчитаем на полученной выборке значение θ
6. исследуем поведение θ на порожденных выборках

Bootstrapping: практические реализации

К сожалению в исходной версии bootstraping часто не эффективен с точки зрения производительности

- Бросить много пуассонов с $\lambda = 1$
- Байесовский bootstrap (волшебный log)
- Гладкий bootstrap (шумный сигнал)
- etc.

Bootstrapping: практические реализации

К сожалению в исходной версии bootstraping часто не эффективен с точки зрения производительности (Почему?)

- Бросить много пуассонов с $\lambda = 1$
- Байесовский bootstrap (волшебный log)
- Гладкий bootstrap (шумный сигнал)
- etc.

Несколько рекомендаций

Когда нельзя пользоваться bootstrapping'ом:

- Когда бесконечная дисперсия
- Когда совсем мало примеров и возможны повторы X'

Нужно всегда помнить что:

- Повторные выборки зависимы и степень зависимости варьируется от m
- В повторных выборках значение, например, дисперсии часто ниже чем на генеральной совокупности
- Чтобы понять насколько результат смещен можно применить дополнительные деления выборки

Домашнее задание

- Сгенерировать последовательность $\{(x_i, y_i)|x_i, y_i \sim U(0,1)\}_{i=1}^{m}$, $m = 10000$
- Выбросить минимально возможное количество примеров так, чтобы y стало линейно зависимо от x с уровнем значимости $\alpha = 0.05$
- Сообщить процент выброшенного

Переборные методы как частный случай сэмплирования

Сэмплирование можно использовать не только для сбора данных! Представим себе, что процесс оптимизации устроен так:

$$F_0 = \arg\max_{\beta \in \mathbb{R}^m} \mu_{\xi \sim D} T(y_\xi, F(x_\xi, \beta))$$

И нету у нас никаких способов понять свойства зависимости T от β.

Формальная постановка

$$F_0 = \arg\max_F p(F|X)$$

+ если известны вероятности можно попробовать посэмплировать решения;
— не определено пространство F;
— неясно как устроить обход.

Иногда все просто

$$F_0 = \arg \max_{F \in \{f_i\}_{i=1}^n} p(F|X)$$

1. введём порядок обхода;
2. переберём все возможные решения;
3. составим взвешенное решение/выберем лучшее.

Но чаще всё непросто

$$F_0 = \arg \max_{F \in \{f_i\}_{i=1}^{\infty}} p(F|X)$$

1. введём порядок обхода;
2. применим систематическое сэмплирование;
3. составим взвешенное решение/выберем лучшее.

Случайное блуждание I

Чтобы построить порядок обхода можно воспользоваться такой схемой:

$$F = F(x, \beta), \beta \in \mathbb{R}^n$$
$$F_t = F(x, \beta_t)$$
$$\beta_{t+1} = \beta_t + \xi \quad C(\beta_{t+1} | \{\beta_i\}_0^t)$$

- будем блуждать по пространству параметров;
- необходимо определить:
  1. способ сделать шаг;
  2. условие принятия этого шага.

Случайное блуждание II

На что стоит обратить внимание при построении блуждания:

- размерность β может быть меньше чем кажется;
- ограничения на β существенно осложняют процедуру.

Некоторые виды случайного блуждания

- множество фиксированных шагов $\xi \sim U(\{\xi_i\}_1^m)$;
- гауссовское $\xi_t \sim N(\mu, \sigma^2)$;
- самозависимое;
- etc.

Simple hill climbing

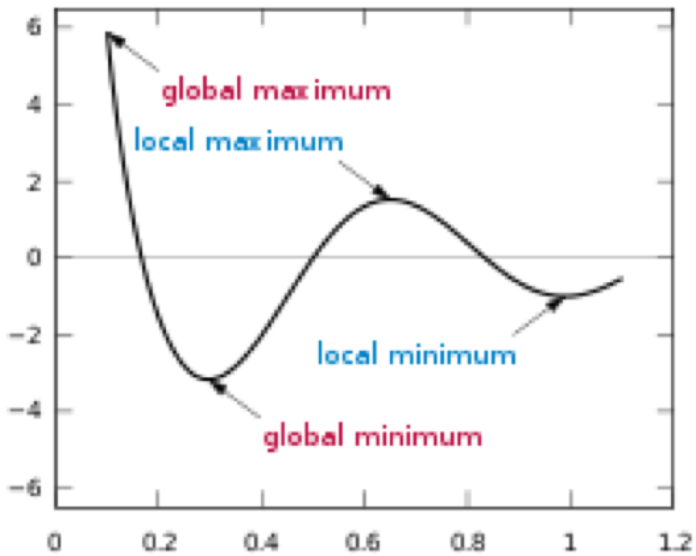
$$\xi \sim U(\{\xi_i\}_1^{2n}), \xi_{i\frac{i}{2}} = -1^{i \ mod \ 2}\omega, \xi_{ij} = 0, j \neq \frac{i}{2},$$

$$C(\beta_{t+1}|\beta_t) = \frac{p(F(\beta_{t+1})|X)}{p(F(\beta_t)|X)} > 1$$

Свойства:

- простой;
- быстро сходится;
- зависим от выбора начальной точки;
- etc.

Random-restart (shotgun) hill climbing



Проблемы:

- сходится в локальный максимум;
- может долго сходиться, если начало далеко от максимума;
- аллеи.

$\Rightarrow$ Можно рестартить hill climbing из разных начальных точек

Интуиция

Мы бы хотели получить сэмплирование, а для этого:

- хорошо бы обойти всё пространство;
- нельзя всегда ходить "по шерсти";
- скорость движения должна меняться в зависимости от плотности.

$\Rightarrow$ Markov Chain Monte-Carlo (MCMC)

Metropolis-Hastings алгоритм

Введем $p(\beta_1|\beta_2)$, отвечающую за локальность.

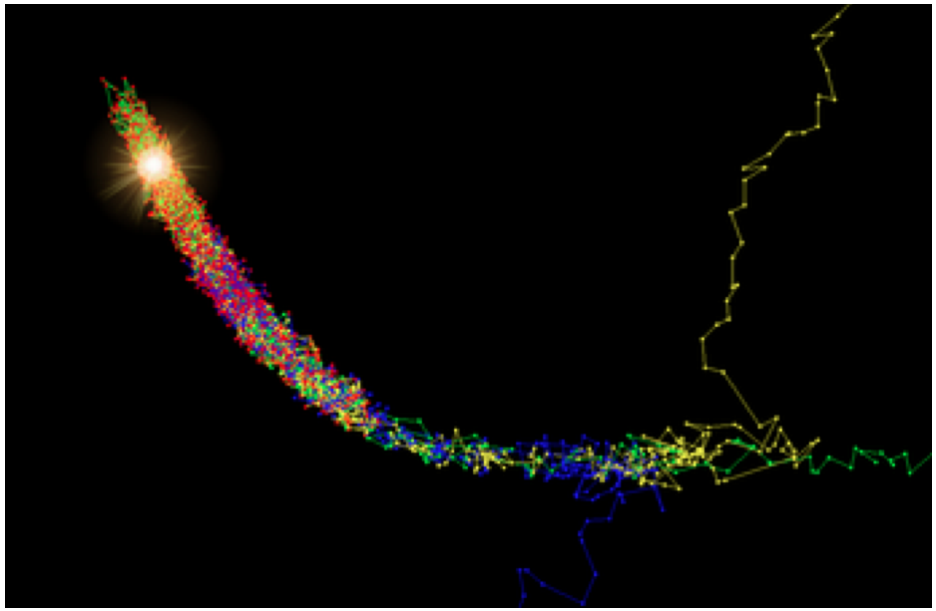
$$\alpha = \frac{p(F_{\beta_{t+1}}|X)p(\beta_{t+1}|\beta_t)}{p(F_{\beta_t}|X)p(\beta_t|\beta_{t+1})}$$

$$\psi \sim U(0,1)$$

$$C(\beta_{t+1}|\beta_t) = \begin{cases} 1, & \alpha \geq \psi \\ 0 \end{cases}$$

Например, $p(\beta_{t+1}|\beta_t) \sim N(\beta_t, \sigma^2 E)$
Если $p(\beta_1|\beta_2) = p(\beta_2|\beta_1)$ — это Metropolis

Свойства



\+ Обходит всё пространство

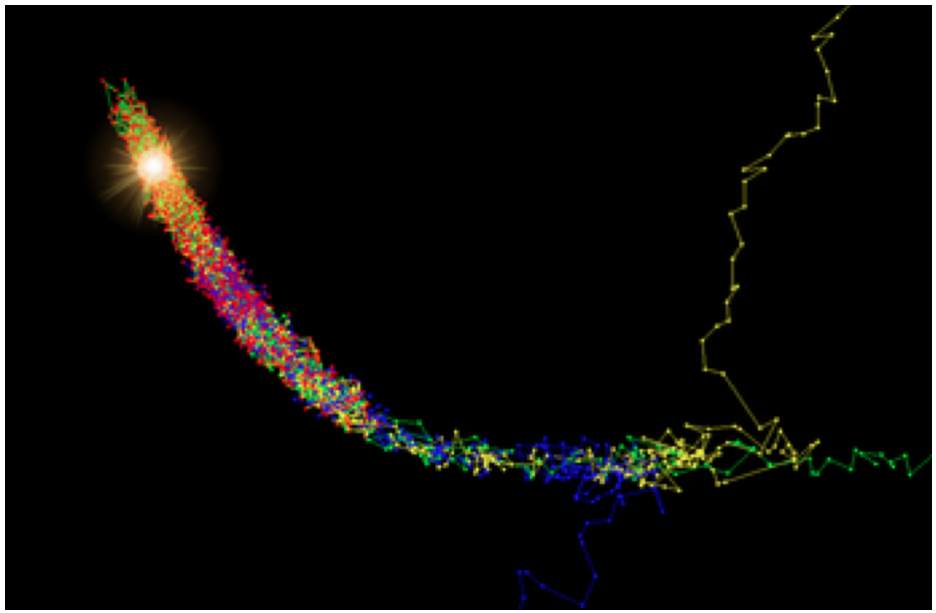
\+ Это действительно взвешенное сэмплирование

— Последовательные самплы похожи друг на друга

— На этапе разогрева показывает что-то странное

$\Rightarrow$ *Точно придём в максимум!*

Свойства



\+ Обходит всё пространство

\+ Это действительно взвешенное сэмплирование

— Последовательные самплы похожи друг на друга

— На этапе разогрева показывает что-то странное

$\Rightarrow$ *Точно придём в максимум!*

Проблема только с тем, что придём за бесконечное время

Сложности в использовании

- Сходимость зависит от выбора $p(\beta_{t+1}|\beta_t)$
- Если хотим использовать разумное распределение, оно многомерное $\Rightarrow$ его сложно реализовывать

Пример с дартс I

Вася и Петя повесели на стене мишень и поиграли в дартс. На следующий день пришел их бригадир Юра и задался вопросом кто из подчиненных будет заделывать дырки в стенах.

- Будет честнее, чтобы “косой” платил больше
- Дырки все одинаковые
- Каждый говорит: “да я токо разок кинул!”
- Один признался, что: “ну вот это — моя дырка”
- Играли без употребления, так что можно считать, что кидали $\sim N(m_i, \Sigma_i)$, $i \in \{\text{Вася, Петя}\}$ и параметры не менялись во времени

Надо помочь Юре!

Пример с дартс II

При фиксированных параметрах вероятность увидеть дырки $\{x_j\}, x_j \in \mathbb{R}^2$

$$LL = \sum_j \log(\pi N(m_A, \Sigma_A)(x_j) + (1 - \pi)N(m_B, \Sigma_B)(x_j))$$

где A — Вася, B — Петя, π — доля Васиных бросков, m_i — сбитость прицела, Σ_i — разброс стрелка.
Максимизировать такое добро трудновато из-за суммы под логарифмом. Было бы классно точно знать, что тот или иной бросок точно сделал Вася $A_t \in \{0, 1\}$:

$$LL = \sum_t A_t \log(N(m_A, \Sigma_A)(x_j)) + (1 - A_t) \log(N(m_B, \Sigma_B)(x_j))$$

В таком варианте все совсем просто считается (если мы еще не забыли формулу плотности нормального многомерного распределения :))

Пример с дартс III

Если чуть более формально, то мы ввели скрытую/ненаблюдаемую переменную $I(A)$ (такое называется data augmentation) и хотим:

1. Прикинуть начальные m_i, Σ_i, π
2. Посчитать ожидание скрытого параметра $A_j = \mu((m_i, \Sigma_i, \pi)$
3. В условиях найденных A_j максимизировать LL
4. Перейти к второму шагу до сходимости

Expectation maximization алгоритм

Фокус, который мы проделали называется *EM*-алгоритм. Пускай данные у нас состоят из видимой части L и скрытой Z, $X = (L, Z)$. Мы хотим найти оптимальные параметры β:

$$LL(p(L|\beta)) = LL(p(X|\beta))$$

1. Возьмем какой-то β_0
2. Посчитаем ожидание (expectation step):

$$E(\beta|\beta_t) = \mu\left(LL(\beta, X)|Z, \beta_t\right)$$

3. Проведем оптимизацию (maximization step):

$$\beta_{t+1} = \arg\max_{\beta} E(\beta|\beta_t)$$

4. Перейти к второму шагу до сходимости

Алгоритм Гиббса

В Метрополисе есть проблема: многомерное распределение $p(F|X) = p(\beta|X)$. Можно попробовать рассматривать его по частям.

1. Начнем с какого-то β
2. Сгенерируем β_{t+1} по правилам

$$p(\beta_{t+1,i}|X, \beta_{t+1,1}, \ldots, \beta_{t+1,i-1}, \beta_{t,i+1}, \ldots, \beta_{t,n})$$

3. Будем бегать по $i = 1 \ldots n$, пока не сойдется
4. Полученная β — следующая точка сэмплирования
5. Пока хочется идем в п.2

Пример с дартс IV

На языке товарищей Геман:

1. Прикинуть начальные m_i, Σ_i, π
2. Для каждого сампла сгенерировать A_j из текущего π
3. В условиях найденных A_j максимизировать LL
4. Перейти к второму шагу до сходимости того момента пока распределение параметров не перестанет меняться

Как можно построить $p(F|X)$ для RMSE

$$p(\beta|X) = \frac{e^{-c\|F(\beta|X)-Y\|_2}}{Z}$$

$$Z = \int_{\beta} e^{-c\|F(\beta|X)-Y\|} d\beta$$

Если максимизируем, то надеемся задрать Y так, чтообы Z был определён.

Заключение

Сэмплирование очень полезная вещь:

- собирать данные;
- брать интегралы;
- обучаться.

Посмотрите на bayesian inference и библиотеку Гиббса для него: BUGS.

Задание на дом

- Датасет, как обычно, в svn
- Суть - учимся предсказывать размер аудитории сервиса
- $N(t) = (\frac{1}{1+e^{at+b}})N_0$
- Ищем a, b, N_0
- Можно прямо hill climbing
- Можно подумать и воспользоваться алгоритмом Гиббса (это будет плюсом)
- Дедлайн 17 октября