

Стохастическая оптимизация и программа

Звонарев Никита

Слушатели: 4-й курс, специальность СМ-СМ

Санкт-Петербургский Государственный Университет
Кафедра Статистического моделирования



Санкт-Петербург – 2020 г.

Общая постановка

$\mathcal{D} \subset \mathbb{R}^k$ – область, $f(\mathbf{x})$ – целевая функция, $f : \mathcal{D} \rightarrow \mathbb{R}$.

Задача:

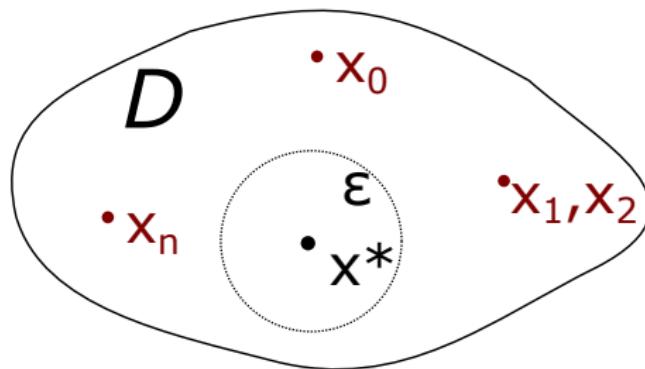
$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}).$$

Нас интересует поиск глобального минимума.

Идея: давайте на каждой итерации случайным образом кидать точку в область, и переходить в неё, если значение целевой функции меньше, чем в текущей точке.

Случайный поиск, версия 1

Пусть пока $\mathcal{D} \subset \mathbb{R}^k$ — ограниченная область.

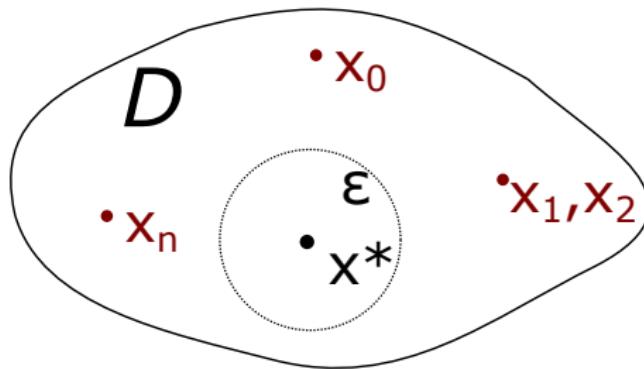


y_1, y_2, \dots — независимые равномерно распределенные в \mathcal{D} . x_0 — задана. Алгоритм:

$$x_{i+1} = \begin{cases} x_i, & f(y_i) \geq f(x_i), \\ y_i, & f(y_i) < f(x_i). \end{cases}$$

Какие есть хорошие свойства у такого простого алгоритма?

Случайный поиск: вероятность попадания

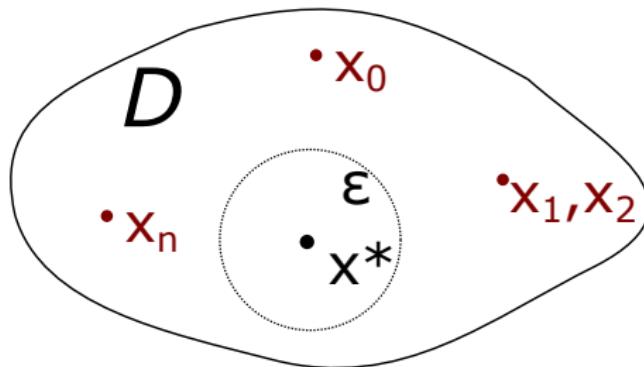


Давайте посчитаем вероятность, с которой попадём в ε -окрестность точки глобального минимума за n шагов.

$B(x^*, \varepsilon)$ — шарик с центром в точке x^* радиуса ε .

$|D|$, $|B(x^*, \varepsilon)|$ — меры области и шарика.

Случайный поиск: вероятность попадания



Давайте посчитаем вероятность, с которой попадём в ε -окрестность точки глобального минимума за n шагов.

$B(x^*, \varepsilon)$ — шарик с центром в точке x^* радиуса ε .

$|D|$, $|B(x^*, \varepsilon)|$ — меры области и шарика.

$$P(\text{за } n \text{ итераций не попали}) = \left(1 - \frac{|B(x^*, \varepsilon)|}{|D|}\right)^n \xrightarrow{n \rightarrow +\infty} 0.$$

Случайный поиск: локальный поиск

$$P(\text{за } n \text{ итераций не попали}) = \left(1 - \frac{|B(\mathbf{x}^*, \varepsilon)|}{|\mathcal{D}|}\right)^n \rightarrow_{n \rightarrow +\infty} 0.$$

Хорошее свойство, но низкая скорость (может требоваться много итераций для попадания в очень малую окрестность), плюс значение в текущей точке (почти) никак не используется.

Случайный поиск: локальный поиск

$$P(\text{за } n \text{ итераций не попали}) = \left(1 - \frac{|B(x^*, \varepsilon)|}{|\mathcal{D}|}\right)^n \rightarrow_{n \rightarrow +\infty} 0.$$

Хорошее свойство, но низкая скорость (может требоваться много итераций для попадания в очень малую окрестность), плюс значение в текущей точке (почти) никак не используется.

Решение: изменить распределение y_i .

$$y_i = \begin{cases} \text{Равномерно в } \mathcal{D} & \text{с вероятностью } 1 - p \\ \text{Равномерно в } B(x_i, \delta) \cap \mathcal{D} & \text{с вероятностью } p. \end{cases}$$

δ — малое число.

Вопрос аудитории: какая плотность у распределения y_i ?

Получили смесь из локального и глобального поиска.

Случайный поиск: дальнейшее развитие идеи

$$y_i = \begin{cases} \text{Равномерно в } \mathcal{D} & \text{с вероятностью } 1 - p \\ \text{Равномерно в } B(x_i, \delta) \cap \mathcal{D} & \text{с вероятностью } p. \end{cases}$$

δ — малое число.

$\delta = \delta_i$ — будем уменьшать δ . Зачем? Чтобы сделать локальный поиск более “острым” (ускорить сходимость).

Например, если было улучшение в точке x_{i+1} , при этом y_i разыгрывалось из малой окрестности, то можно уменьшить δ в α раз, $0 \leq \alpha < 1$: $\delta_{i+1} = \alpha \delta_i$; в противном случае $\delta_{i+1} = \delta_i$.

Случайный поиск: ещё идеи

Предлагается реализовать алгоритм при следующих предположениях:

- ① \mathcal{D} типа “коробка” (параллелепипед); равномерное распределение в нём
- ② $B(x_i, \delta)$ в метрике Чебышёва (т.е. многомерный куб)
- ③ Следствие: пересечение \mathcal{D} и $B(x_i, \delta)$ — параллелепипед
- ④ x_{i+1} берётся жадно, если $f(x_{i+1}) < f(x_i)$

В реальной жизни:

- ① $\mathcal{D} = \mathbb{R}^k$ или произвольной формы; распределение - ?
(нормальное?)
- ② Локальный поиск не обязательно в ограниченной области, не обязательно в шаре, с каким угодно распределением
- ③ Пересечение \mathcal{D} и $B(x_i, \delta)$ зависит от типа задачи (может быть очень сложным)
- ④ Иногда переходят в новую точку, даже если $f(x_{i+1}) \geq f(x_i)$.

Насчёт последнего — см. метод имитации отжига (simulated annealing).

Возможные критерии остановки

- $|f(\mathbf{x}_{n+j}) - f(\mathbf{x}_n)| < \varepsilon$, $j = \min\{m : f(\mathbf{x}_{n+m}) < f(\mathbf{x}_n)\}$.
- Число итераций больше n (должно быть всегда)
- Число итераций с последнего улучшения больше \hat{n}

О программе

Требования к консольной программе:

- ① ООП (сейчас расскажу подробнее)
- ② Несколько функций (3-5 шт.) $k = 2, \dots, 4$ (консольная версия),
 $k = 2$ (графическая)
- ③ Область определения типа “коробка”: “easy” — несколько вариантов границ, “hard” — возможность выбрать границу.
- ④ Несколько критериев остановки (по числу итераций, по значению функции, . . .)
- ⑤ Два метода: детерминированный (выданный на прошлой паре) и случайный
- ⑥ Задаём: функцию (из списка), область (из списка/параметры), критерий остановки (и его параметры), метод (и его параметры), начальную точку
- ⑦ Получаем результат работы программы в виде: найденная точка минимума, значение функции в ней, количество итераций
- ⑧ Графическая оптимизация: +графическое изображение функции и траектория

Методы на выбор для семестрового задания

- ① Покоординатный спуск
- ② Градиентный спуск (производные аналитически)
- ③ Градиентный спуск (производные численно)
- ④ Сопряженные градиенты Fletcher-Reeves
- ⑤ Сопряженные градиенты Polak-Ribiere
- ⑥ Ньютон (поиск по направлению, производные аналитически)
- ⑦ Ньютон (поиск по направлению, производные численно)
- ⑧ Ньютон (backtracking, производные аналитически)
- ⑨ Ньютон (backtracking, производные численно)
- ⑩ Nelder-Mead (метод деформируемого многогранника)

Ещё о программе

Мини-проблема: детерминированная оптимизация, оптимизируем из точки x_i по направлению p_i , вектор упирается в \mathcal{D} .

Решение: пересечь вектор с \mathcal{D} (все смогут это сделать?).

- ➊ Хороший стиль программирования: взятные названия классов/методов, переносы + правила хорошего тона C++ (см. документ от НЭ на Вики СтатМода)
- ➋ Документация. Заголовочные файлы продокументированы специальным образом, чтобы потом документация собиралась одной командой. Например, с помощью *doxygen*.
- ➌ Использование системы контроля версий. Ваши проекты (консольный и графический) должны быть расположены в git репозитарии. Лишнего хлама (бинарные файлы, временные файлы от IDE, прочий мусор) в репозитарии не должно быть. Соответственно, больше никаких zip-ов по почте — присылаете мне ссылку на репозитарий, я его смотрю.

Википедия и ссылки про Git

Git (произносится “гит”) — распределённая система управления версиями.

Система управления версиями (от англ. Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

git-scm.com — основной сайт, где вы можете загрузить git под Windows (linux, macOS — используйте менеджер пакетов).

<https://git-scm.com/book/ru/> — официальный самоучитель на русском.

<https://githowto.com/ru> — ещё один самоучитель.

<https://github.com/> — бесплатный хостинг opensource проектов с git.

<https://help.github.com/> — глобальная помощь с github, без которой поначалу будет трудно.

<https://bitbucket.org/> — ещё один хостинг.