Стохастическая оптимизация и программа

Звонарев Никита

Слушатели: 4-й курс, специальность СМ-СМ

Санкт-Петербургский Государственный Университет
Кафедра Статистического моделирования

Санкт-Петербург – 2018 г.

Общая постановка

$\mathcal{D} \subset \mathbb{R}^n$ — область, $f(\mathbf{x})$ — *целевая функция*, $f : \mathcal{D} \to \mathbb{R}$.
Задача:

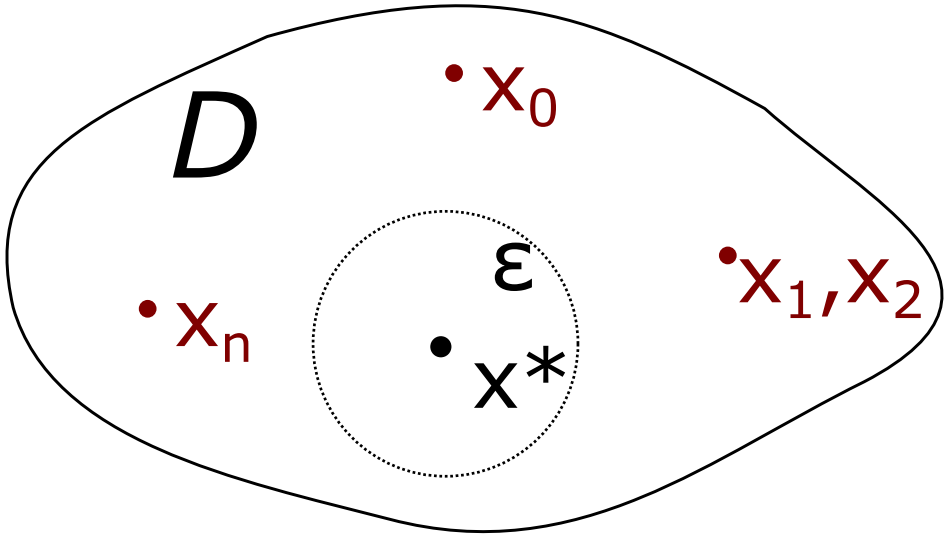
$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}).$$

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 $\mathcal{D} \subset \mathbb{R}^n$ — ограниченная область.

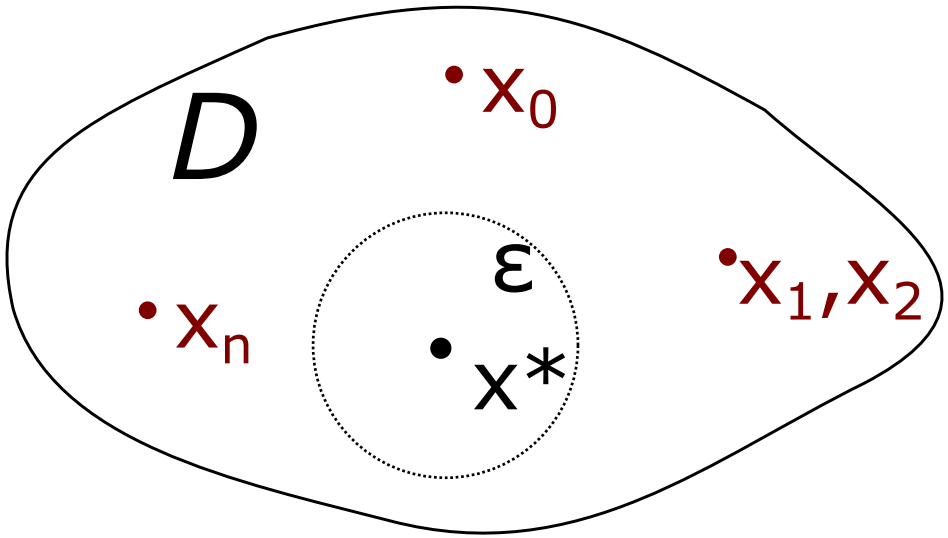


$\mathbf{y}_1, \mathbf{y}_2, \ldots$ — независимые равномерно распределенные в $\mathcal{D}$. $\mathbf{x}_0$ — задана. Алгоритм:

$$\mathbf{x}_{i+1} = \begin{cases} \mathbf{x}_i, & f(\mathbf{y}_i) >= f(\mathbf{x}_i), \\ \mathbf{y}_i, & f(\mathbf{y}_i) < f(\mathbf{x}_i). \end{cases}$$

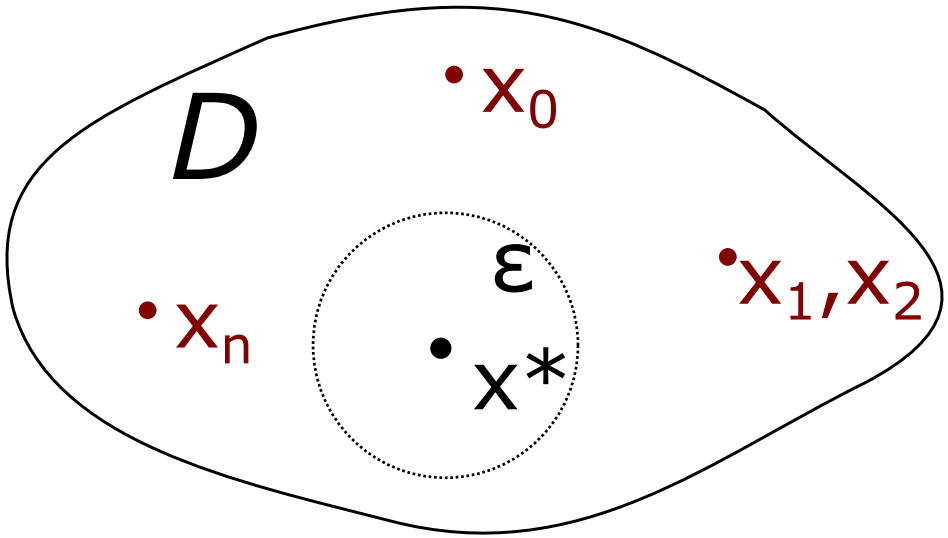
Какие есть хорошие свойства у такого простого алгоритма?

Случайный поиск: вероятность попадания



Давайте посчитаем вероятность, с которой попадём в ε-окрестность точки глобального минимума за n шагов.
$B(\mathbf{x}^*, \varepsilon)$ — шарик с центром в точке $\mathbf{x}^*$ радиуса ε.
$|\mathcal{D}|$, $|B(\mathbf{x}^*, \varepsilon)|$ — меры множества и шарика.

Случайный поиск: вероятность попадания



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

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

$|\mathcal{D}|$, $|B(\mathbf{x}^*, \varepsilon)|$ — меры множества и шарика.

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

Случайный поиск: локальный поиск

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

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

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

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 $\mathbf{y}_i$.

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

δ — малое число.

Вопрос аудитории: какая плотность у распределения $\mathbf{y}_i$?

Получили смесь из локального и глобального поиска.

Случайный поиск: дальнейшее развитие идеи

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

δ — малое число.

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

Например, если было улучшение в точке $\mathbf{x}_{i+1}$, при этом $\mathbf{y}_i$ 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. $B(\mathbf{x}_i, \delta)$ по метрике Чебышёва (т.е. многомерный куб)
3. Следствие: пересечение $\mathcal{D}$ и $B(\mathbf{x}_i, \delta)$ — параллелепипед
4. $\mathbf{x}_{i+1}$ берётся жадно, если $f(\mathbf{x}_{i+1}) < f(\mathbf{x}_i)$

В реальной жизни:

1. $\mathcal{D} = \mathbb{R}^n$ 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 $B(\mathbf{x}_i, \delta)$ зависит от типа задачи (может быть очень сложным)
4. $\mathbf{x}_{i+1}$ не всегда берётся, если значение функции меньше.

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

Возможные критерии остановки

- $|f(\mathbf{x}_{k+j}) - f(\mathbf{x}_k)| < \varepsilon$, $j = \min\{m : f(\mathbf{x}_{k+m}) < f(\mathbf{x}_k)\}$.
- 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.) $n = 2, \ldots, 4$ (консольная версия), $n = 2$ (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

Ещё о программе

Мини-проблема: детерминированная оптимизация, оптимизируем из точки $\mathbf{x}_i$ по направлению $\mathbf{p}_i$, вектор упирается в $\mathcal{D}$.
Решение: пересечь вектор с $\mathcal{D}_i$ (все смогут это сделать?).

1. Хороший стиль программирования: внятные названия классов/методов, переносы + правила хорошего тона C++ (см. документ от НЭ на Вики СтатМода)
2. Документация. Заголовочные файлы продокументированы специальным образом, чтобы потом документация собиралась одной командой. Например, с помощью doxygen.
3. 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://gitlab.com/ — ещё один хостинг.