Андреев Ю.А., Пронкина А.Д., Ермольев М.А.

Погружение в Лабиринт Решений: Введение в Искусство Backtracking

Backtracking — это класс алгоритмов для поиска решения некоторых вычислительных задач. Основная концепция - это создание кандидатов и проверка, можно ли потенциально довести его до решения; если нельзя - кандидат убивается.

Однако метод обратного отслеживания может быть применён только к задачам, в которых допускается понятие «частичного решения кандидата», и возможна относительно быстрая проверка того, можно ли завершить агента до допустимого решения. Однако, если применить backtracking возможно, он работает быстрее перебора всех вариантов, т.к. за одну проверку может отбросить множество решений.

Путеводитель по Древу Выбора: Алгоритм Раскрытия Потенциальных Решений

Backtracking перечисляет набор частичных кандидатов, которые, в принципе, могут быть как-то дополнены, чтобы дать все возможные решения заданной задачи. Дополнение выполняется поэтапно, с помощью последовательности шагов расширения кандидатов. Концептуально это выглядят как деревовидная структура. Где вершины - это кандидаты, рёбра - это расширения кандидатов, а листы - это тупиковые кандидаты, которых нельзя расширить (или которые признаны нерешаемыми).

Алгоритм же проходит по этому дереву рекурсивно и в глубину, в каждой вершине проверяя её валидность. Если вершина не ведёт к решению, то она и всё её поддерево удаляется. Если же вершина ведёт к решению, алгоритм рекурсивно проходит все поддеревья этой вершины.

Таким образом, фактическое дерево поиска, которое просматривается алгоритмом, является лишь частью потенциального дерева. Общая стоимость алгоритма равна количеству узлов фактического дерева, умноженному на стоимость получения и обработки каждого узла. Этот факт следует учитывать при выборе потенциального дерева поиска и реализации теста на удаление узлов.

Псевдокод

Чтобы это всё работало, нужно реализовать 6 методов для текущей задачи с данными P.

1. **root**$(\mathbf{P})$ Создаёт и возвращает изначального кандидата в корне дерева.
2. **reject**$(\mathbf{P}, \mathbf{c})$ Возвращает *true*, если кандидат не приведёт к решению.
3. **accept**$(\mathbf{P}, \mathbf{c})$ Возвращает *true*, если кандидат является решением.
4. **first**$(\mathbf{P}, \mathbf{c})$ Создаёт первое расширение для кандидата c.
5. **next**$(\mathbf{P}, \mathbf{s})$ Создаёт следующее расширение для кандидата после s.
6. **output**$(\mathbf{P}, \mathbf{c})$ Использует решение c для генерации вывода.

Листинг 1: Псевдокод

```
procedure backtrack(P, c) is
    if reject(P, c) then return
    if accept(P, c) then output(P, c)
    s <- first(P, c)
    while s != NULL do
        backtrack(P, s)
        s <- next(P, s)
```

Побег из Тупика: Тайны Локальной Оптимизации и Искусство Выбора Шага

К настоящему времени разработано множество методов локальной оптимизации для решения задач общего вида. Большинство из них используют принцип локального спуска, где на каждом шаге метод последовательно переходит к точкам с существенно меньшим (или большим, в случае максимизации) значением целевой функции.

О шагах: Итерационная форма методов локальной оптимизации для решения задач поиска экстремума целевой функции требует выбора шага расчета вдоль заданных направлений на каждом шаге итерации. Шаг расчета может быть постоянным или переменным, но оптимальное значение длины шага определяется в результате поиска экстремума целевой функции в выбранном направлении с использованием методов одномерной оптимизации[1]

В результате для определения оптимального шага расчета требуется выполнить большой объем вычислений целевой функции. Для снижения числа операций на практике используют другой подход: подбирают такие значения шага расчета h, чтобы они удовлетворяли любому из представленных ниже условию.

Правила Игры: условия от Армихо до Голдстейна-Армийо

Первое условие(правило Армихо)

Правило Армихо утверждает, что функция в точке $f(x_k \pm \lambda \cdot g(x_k))$ не должна превышать значения некоторой убывающей линейной функции (аппроксимации), равной $f(x_k)$ в нуле:

$$f(x_k \pm \lambda \cdot g(x_k)) \ll f(x_k) \pm \sigma \cdot \lambda \cdot \Delta f_k \cdot g(x_k) \qquad (\star)$$

σ — коэффициент, удовлетворяющий $0 < \sigma < 1$. Шаг расчёта λ определяется итеративно путём умножения первоначального шага λ_0 на коэффициент β $(0 < \beta < 1)$ до тех пор, пока не будет выполняться условие $(\star)$.

Это условие часто называют *адаптивным* или *динамическим*, так как длина шага не фиксирована заранее, а подбирается в процессе итераций.

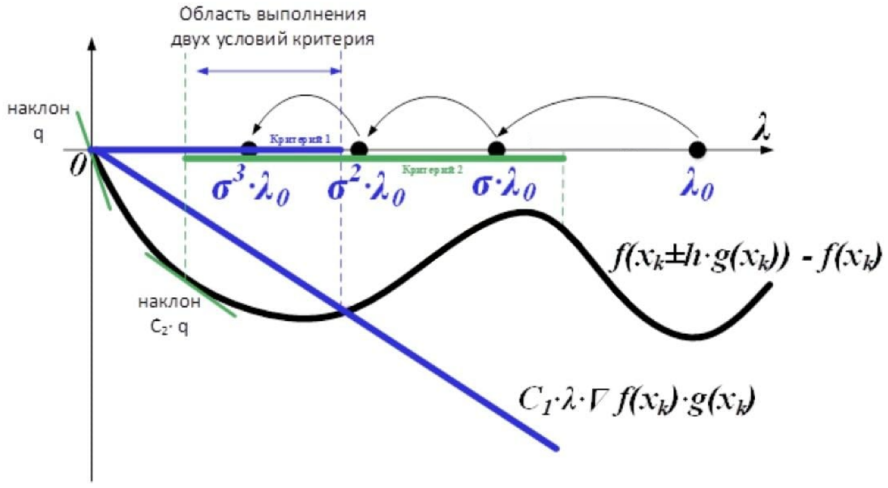

Рис.1. Критерий выбора шага расчёта по правилу Армихо

На рисунке схематично показано, как выбирается шаг: при слишком большом значении σ значение $f(x_k \pm \lambda \cdot g(x_k))$ может оказаться выше, чем позволяет условие Армихо. Поэтому σ уменьшают до тех пор, пока не будет выполнено неравенство $(\star)$.

Алгоритм вычисления шага

Методика определения шага расчёта оптимизационной задачи в соответствии с правилом Армихо заключается в следующем:

1. **Инициализация.** Задаётся начальное значение коэффициента σ, $\sigma \in (0; 1)$.
2. **Инициализация.** Также задаётся начальное значение λ_0.
   Далее: процедура поиска (проверка выполнения условия по правилу Армихо).
3. **Проверка условия (если условие $(\star)$ не выполняется).** В случае если условие по правилу Армихо не выполняется, тогда необходимо скорректировать шаг расчёта $\lambda_k = \lambda_0 \cdot \beta^k$, где $\beta \in (0; 1)$. По умолчанию примем, что $\beta = 0.5$, а $k = 0, 1, 2, \ldots$ — текущий шаг поиска.
4. **Проверка условия (если условие $(\star)$ выполняется).** В случае если условие по правилу Армихо выполняется, тогда в качестве шага расчёта можно принять $\lambda = \lambda_k$, а процедура поиска завершается.

[1] Поиск считается одномерным, в случае если аргументом целевой функции является один управляемый параметр.

- Метод дихотомического деления и метод золотого сечения – это методы одномерной оптимизации, основанные на делении отрезка, на котором ищется экстремум, пополам или в пропорциях золотого сечения (0,382 / 0,618), соответственно.
- Метод полиномиальной аппроксимации (метод квадратичной интерполяции) – это метод одномерной оптимизации, в соответствии с которым целевая функция аппроксимируется квадратичным полиномом.

Данное правило требует однократного вычисления градиента[2], после чего выполняется относительно небольшое число дополнительных итераций, необходимых для выбора подходящего шага. На каждой такой вложенной итерации требуется только вычислить значение целевой функции (без вычисления градиента)[3]. Поскольку эти проверки являются сравнительно «лёгкими», метод в целом остаётся эффективным.

Следует отметить, что данное условие удовлетворяется для всех достаточно малых λ.

Кроме того, **правило Армихо** может быть обобщено и на многокритериальные задачи.

Второе условие (Правило Вольфе-Пауэлла - Wolfe. P)

Второе условие, называемое *Правилом Вольфа–Пауэлла* (Wolfe–P), помогает корректно выбрать длину шага при поиске минимума функции $f(\mathbf{x})$ в направлении $\mathbf{d}_k$ из точки $\mathbf{x}_k$. Это правило формулируется так, что мы рассматриваем функцию $f(x_k + \pm\lambda \cdot g(x_k))$ и далее два условия:

1. **Case (Первое условие Вольфе–Пауэлла):**
   Функция $f(x_k \pm \lambda \cdot g(x_k))$ не должна превышать значения некоторой убывающей линейной функции, равной $f(x_k)$ в нуле:
   $$f\big(x_k \pm \lambda \cdot g(x_k)\big) \ll f(x_k) \ \pm \ C_1 \cdot \lambda \cdot \nabla f_k \cdot g(x_k) \qquad \text{(Kriterium1)}$$

2. **Case (Второе условие Вольфе–Пауэлла):**
   Величина скорости изменения функции в заданном направлении $f(x_k \pm \lambda \cdot g(x_k))$ должна быть в C_2 раз больше, чем скорость изменения функции в первоначальной точке x_k:
   $$\nabla f\big(X_k \pm \lambda \cdot g(X_k)\big) \cdot g(X_k) \ \ \geq \ \ C_2 \ \cdot \ \nabla f\big(X_k\big) \cdot g(X_k). \qquad \text{(Kriterium2)}$$

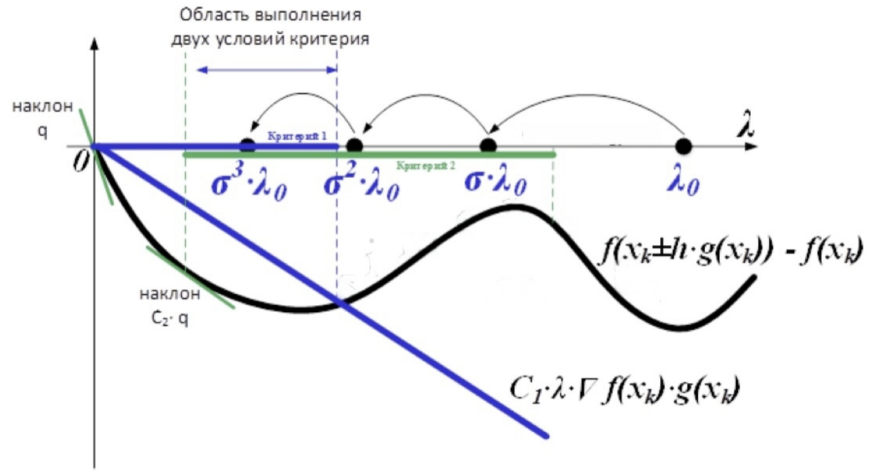

Рис.2. Критерий выбора шага расчета по правилу Вольфе-Пауэлла

Где:

- x_k — текущая точка, из которой осуществляется поиск минимума.
- λ — скаляр, задающий длину шага в направлении $g(x_k)$.
- $g(x_k)$ — в общем случае это направление спуска (или градиент, или некоторая функция, задающая направление поиска).
- $\nabla f_k \equiv \nabla f(x_k)$ — градиент функции f в точке x_k.
- C_1, C_2 — коэффициенты (параметры) условия Вольфа–Пауэлла, которые удовлетворяют соотношениям $0 < C_1 < C_2 < 1$. Как правило, C_1 отвечает за *достаточное убывание* (насколько сильно должна уменьшиться функция), а C_2 — за *ограничение на производную* (насколько сильно должна измениться скорость убывания).

[2] *Градиент* $\nabla f(x)$ есть вектор частных производных функции f, который указывает направление наибольшего возрастания f. В методах спуска чаще используют антиградиент $-\nabla f(x)$, так как он указывает направление убывания функции.

[3] Вычисление значения $f(x)$ (скалярной функции) зачастую заметно проще и быстрее, чем вычисление её градиента, особенно в задачах высокой размерности.

Алгоритм вычисления шага

Методика определения шага расчета оптимизационной задачи в соответствии с правилом Вольфе-Пауэлла заключается в следующем:

1. **Инициализация.** Задаются коэффициенты C_1 и C_2, $(0 < C_1 < C_2 < 1)$.
2. **Инициализация.** Задаётся начальное значение шага $\lambda = \lambda_0$, принимая коэффициенты $\alpha_1 = \alpha_2 = 0$.
   Далее: процедура поиска (проверка выполнения условий по правилу Вольфе–Пауэлла).
3. **Проверка условия (если первое условие** (Kriterium1) **не выполняется).** Тогда принять коэффициент $\alpha_1 = \lambda$. Перейти к пункту 5.
4. **Проверка условия (если второе условие** (Kriterium2) **не выполняется).** Тогда принять коэффициент $\alpha_2 = \lambda$.
   Case: $\alpha \neq 0$, перейти к пункту 5.
   Case: $\alpha_1 = 0$ — выполнить экстраполяцию[4], положив $\lambda = \alpha_2 \cdot r$, где $r > 1$ — коэффициент экстраполяции.
   Перейти к пункту 3.
5. **Выполнение текущего расчёта шага по формуле:**

   $$\lambda = \alpha_1 \cdot \gamma \ + \ \alpha_2 \cdot (1 - \gamma),$$

   где γ — коэффициент интерполяции[5], $0 < \gamma < 1$.
   Перейти к пункту 3.
6. **Проверка условия (если выполняются оба условия** (Kriterium1) **и** (Kriterium2)**):** в качестве шага расчёта можно принять $\lambda = \lambda_k$, а процедура поиска завершается.

Третье условие(правило Голдстейна-Армийо)

Правило Голдстейна-Армийо позволяет выбрать шаг расчета, рассматривая следующее неравенство:

$$\xi_1 \ < \ \frac{f\big(x_k \pm \lambda \cdot g(x_k)\big) \ - \ f(x_k)}{\lambda \cdot \nabla f(x_k) \cdot g(x_k)} \ < \ \xi_2.$$

Алгоритм вычисления шага

Методика определения шага расчета оптимизационной задачи в соответствии с правилом Голдстейна-Армийо заключается в следующем:

1. **Инициализация.** Задаются коэффициенты ξ_1 и ξ_2, $0 < \xi_1 < \xi_2 < 1$
2. **Инициализация.** Задаёется начальное значение шага $\lambda = \lambda_0$
3. **Проверка условия (если условие по правилу Голдстейна-Армийо не выполняется)**, тогда необходимо скорректировать шаг расчета $\lambda_k = \lambda_0 \cdot \beta^k$, $\beta \in (0, 1)$. По умолчанию примем $\beta = 0.5$, $k = 0, 1, 2 \ldots$ - текущий шаг.
4. **Проверка условий (если условие по правилу Голдстейна-Армийо выполняется**, тогда в качестве шага расчета можно принять $\lambda = \lambda_k$, а процедура поиска завершается.

[4]Экстраполяция — разновидность аппроксимации, при которой оценивание значения переменной производится за пределами интервала её изменения.

[5]Интерполяция — способ нахождения промежуточных значений величины по известному набору значений.

Заключение и вывод: in progress

Источники:

1. simenergy
2. wikipedia
3. chatgpt
4. конспекты by imkochelorov