Введение в оптимизацию

Звонарев Н.К., Голяндина Н.Э.

Слушатели: 1-й курс магистратуры

Санкт-Петербургский Государственный Университет
Кафедра Статистического моделирования

Санкт-Петербург – 2020 г.

Мотивация

1. ОМП (оценки максимального правдоподобия)
   $X_1, \ldots, X_N$ — выборка, θ — неизвестный параметр, $p(x|\theta)$ — семейство плотностей.

   $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p(X_i|\theta)$ — ОМП.

2. МНК (нелинейный метод наименьших квадратов)
   $\mathbf{y} = (Y_1, \ldots, Y_N)^{\mathrm{T}}$ — выборка, $g : \mathbb{R}^k \to \mathbb{R}^N$.

   $\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{y} - g(\mathbf{x})\|^2$.

   Линейный случай: $g(\mathbf{x}) = \mathbf{F}\mathbf{x}$, $\mathbf{F} \in \mathbb{R}^{N \times k}$.

3. Методы машинного обучения.

4. Впрочем, всегда хочется чего-то лучшего, так что мотивация оптимизировать есть у каждого.

Общая постановка задачи

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

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

Она же, в виде поиска точки минимума:

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

Если можно решить задачу аналитически, то нужно использовать аналитическое решение.

Если нельзя, то приходится использовать численные методы.

(Примерная) классификация методов оптимизации

Определение: последовательный метод численной оптимизации — алгоритм, который начиная с точки $\mathbf{x}_0$ строит последовательность $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots$ такую, что $\mathbf{x}_n \to \mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$.

Что нужно искать?

1. Локальная оптимизация
   (Строгий) локальный минимум $\mathbf{x}^*$ — существует $\varepsilon > 0$: $f(\mathbf{x}^*) < f(\mathbf{x})$ для любого $\mathbf{x}$: $\|\mathbf{x} - \mathbf{x}^*\| < \varepsilon$, $\mathbf{x} \neq \mathbf{x}^*$.
2. Глобальная оптимизация
   (Строгий) глобальный минимум $\mathbf{x}^*$: $f(\mathbf{x}^*) < f(\mathbf{x})$ для любого $\mathbf{x} \in \mathcal{D}$, $\mathbf{x} \neq \mathbf{x}^*$.

Как нужно искать?

1. Детерминированная оптимизация: для поиска локальных минимумов гладких функций.
2. Стохастическая оптимизация (случайный поиск): для поиска глобального минимума, если f "плохая/сложная" функция.

Задание: нарисуйте непрерывную функцию, которая очень плохо подходит для поиска минимума.

Некоторые характеристики и утверждения

Градиент:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_k}(p) \end{bmatrix}$$

Гессиан (Hessian):

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_k} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_k \, \partial x_1} & \frac{\partial^2 f}{\partial x_k \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_k^2} \end{bmatrix}$$

Некоторые характеристики и утверждения

- Достаточное условие локального экстремума:
  (Заметим, что достаточного условия глобального экстремума не существует, кроме его определения)
  Пусть в точке $\mathbf{x}^*$ выполнено:
  $\nabla f(p) \equiv \mathbf{0}$ и матрица Гессе $\mathbf{H}f$ положительно определена. Тогда точка $\mathbf{x}^*$ является точкой локального минимума.

- Достаточное условие строгой выпуклости функции:
  Если в каждой точке матрица Гессе $\mathbf{H}f$ положительно определена, то функция является строго выпуклой.

- Достаточное условие существования минимума:
  Если функция строго выпуклая, а область поиска выпуклая, то существует единственная точка минимума.

Задание. Является ли точка (0,0) точкой минимума
(a) $f(x, y) = x^2 + y^2$ и (b) $f(x, y) = x^2 - y^2$?

(Примерная) классификация методов оптимизации

По виду целевой функции:

1. Нет предположения о гладкости f
2. Есть предположение о гладкости f

По использованию производной:

1. Не используется производная
2. Используется производная (первая, вторая, . . .)
   Первые производные образуют *градиент* (gradient), вторые — *матрицу Гессе* (Hessian).
   Как считать производную: аналитически, численно, использовать аппроксимацию . . .

Численная оценка производной: $f'(x) = (f(x+\delta) - f(x-\delta))/(2\delta)$ или $f'(x) = (f(x+\delta) - f(x))/\delta$.

Задание: как численно оценить вторую производную?

(Примерная) классификация методов оптимизации

По виду ограничений:

1. Безусловная: $\mathcal{D} = \mathbb{R}^k$
2. Условная
   Линейные/нелинейные ограничения, (пример: сумма параметров равна 1)
   f — линейная, $\mathcal{D}$ ограничен линейными функциями — линейное программирование.
   f — квадратичная, $\mathcal{D}$ ограничен линейными функциями — квадратичное программирование.

Предположения о выпуклости области/целевой функции:

1. f — не выпуклая или $\mathcal{D}$ — не выпуклое
2. f — выпуклая (при этом $\mathcal{D}$ — выпуклое, например $\mathcal{D} = \mathbb{R}^k$)
   Если f — строго выпукла, то у неё один строгий локальный минимум (он же строгий глобальный)

Сведение условной оптимизации к безусловной

Задача: $\min_{\mathbf{x}\in D} f(\mathbf{x})$
Условия: $g_i(\mathbf{x}) \le 0$, $i = 1, \ldots, m$ и $h_j(\mathbf{x}) = 0$, $j = 1, \ldots, \ell$.

- Метод штрафных или барьерных функций. Значение оптимизируемой функции увеличивается, если условие не выполнено. Это больше подходит к ограничениям типа неравенств.
- Включение условий в оптимизируемую функцию. Множители Лагранжа $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{j=1}^{\ell} \lambda_j h_j(\mathbf{x})$ (ограничения типа равенств). В общем случае, теорема Куна-Такера (Karush–Kuhn–Tucker). Это не совсем безусловная оптимизация, но задача может стать проще.

Сведение к одномерной оптимизации

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

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

Теперь $k > 1$.

Идея: свести задачу к одномерной оптимизации. (Как?)
Подход: итеративный, каждая итерация — в два этапа:

1. Поиск направления оптимизации ($\mathbf{x}_n$ — текущая точка, направление — $\mathbf{p}_n$ (как найти?)).
2. Шаг по направлению $\mathbf{p}_n$ либо одномерная оптимизация: $\alpha_n^* = \arg\min_{0 \le \alpha \le 1} f(\mathbf{x}_n + \alpha \mathbf{p}_n)$, $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n^* \mathbf{p}_n$. Здесь α_n^* определяет размер шага, а в ML называется скоростью обучения.

Трудоемкость: (число шагов до STOP) $\times$ (трудоемкость шага).

Одномерная оптимизация: перебор по решетке

$k = 1$, $\mathcal{D} = [a, b] \subset R$.

Поиск минимума на решётке с шагом ε (линейный поиск, последовательный поиск, linear search).
"Число узлов" $n = \lfloor \frac{b-a}{\varepsilon} \rfloor$.
"Узлы решётки" $y_j^{(n)} = a + j\frac{b-a}{n}$, $j = 0, \ldots, n$.

$$x_n = \arg\min_{j=0,\ldots,n} f(y_j^{(n)}).$$

Если f — выпуклая, то $|x_n - x^*| \leq \varepsilon$.
Если f — непрерывная, то $|x_n - x^*| = O(\varepsilon)$.
Сколько вычислений функции нужно сделать, чтобы найти минимум с точностью ε?
(А если делать перебор по решетке в k-мерном кубе?)

Одномерная оптимизация: перебор по решетке

$k = 1$, $\mathcal{D} = [a, b] \subset R$.

Поиск минимума на решётке с шагом ε (линейный поиск, последовательный поиск, linear search).
"Число узлов" $n = \lfloor \frac{b-a}{\varepsilon} \rfloor$.
"Узлы решётки" $y_j^{(n)} = a + j\frac{b-a}{n}$, $j = 0, \ldots, n$.

$$x_n = \arg\min_{j=0,\ldots,n} f(y_j^{(n)}).$$

Если f — выпуклая, то $|x_n - x^*| \leq \varepsilon$.
Если f — непрерывная, то $|x_n - x^*| = O(\varepsilon)$.
Сколько вычислений функции нужно сделать, чтобы найти минимум с точностью ε? $O(1/\varepsilon)$
(А если делать перебор по решетке в k-мерном кубе?) $O(1/\varepsilon^k)$

Одномерная оптимизация: метод дихотомии

$k = 1$, $\mathcal{D} = [a, b] \subset R$, f — выпуклая, существует производная.

Суживать отрезок поиска. Посчитаем знак производной в середине отрезка, и в зависимости от этого оставим нужную половину отрезка.

Алгоритм (binary search)

$l_0 = a$, $r_0 = b$, $i = 0$.
Пока $r_i - l_i > \varepsilon$:

1. $m_i = (l_i + r_i)/2$.
2. *Если* $f'(m_i) < 0$, *то* $l_{i+1} = m$, *иначе* $r_{i+1} = m$.
3. *Положить* $i = i + 1$.

Результат: $(l_i + r_i)/2$

Выполнено $|x_n - x^*| \leq \varepsilon$.

Сколько вычислений (функции+производной)?

Одномерная оптимизация: метод дихотомии

$k = 1$, $\mathcal{D} = [a, b] \subset R$, f — выпуклая, существует производная.

Суживать отрезок поиска. Посчитаем знак производной в середине отрезка, и в зависимости от этого оставим нужную половину отрезка.

Алгоритм (binary search)

$l_0 = a$, $r_0 = b$, $i = 0$.
Пока $r_i - l_i > \varepsilon$:

1. $m_i = (l_i + r_i)/2$.
2. *Если* $f'(m_i) < 0$, *то* $l_{i+1} = m$, *иначе* $r_{i+1} = m$.
3. *Положить* $i = i + 1$.

Результат: $(l_i + r_i)/2$

Выполнено $|x_n - x^*| \leq \varepsilon$.

Сколько вычислений (функции+производной)? $O(\log(1/\varepsilon))$

Одномерная оптимизация: тернарный (троичный) поиск

$k = 1$, $\mathcal{D} = [a, b] \subset R$, f — выпуклая.

Идея: будем суживать отрезок, на котором ищем минимум. На каждой итерации разобьём его на три подотрезка. Выкинем тот, который точно не содержит минимум.

Алгоритм (ternary search)

$l_0 = a$, $r_0 = b$, $i = 0$.
Пока $r_i - l_i > \varepsilon$:

1. $\hat{l}_i = (2l_i + r_i)/3$, $\hat{r}_i = (l_i + 2r_i)/3$.
2. *Если* $f(\hat{l}_i) < f(\hat{r}_i)$, *то* $r_{i+1} = \hat{r}_i$, $l_{i+1} = l_i$, *иначе* $l_{i+1} = \hat{l}_i$, $r_{i+1} = r_i$.
3. *Положить* $i = i + 1$.

Результат: $(l_i + r_i)/2$

Выполнено $|x_n - x^*| \leq \varepsilon$.
Время работы: $O(\log(1/\varepsilon))$.

Одномерная оптимизация: метод золотого сечения

$k = 1$, $\mathcal{D} = [a, b] \subset R$, f — выпуклая.

Проблема тернарного поиска: целевая функция f вычисляется на каждой итерации дважды, в точках $\hat{l}_i$ и $\hat{r}_i$. Если f долго вычисляется, то это может быть неприемлемо.
Идея: как-то иначе разбить отрезок, чтобы f в одной из точек уже была подсчитана на прошлой итерации?

Да, это разбиение в пропорции золотого сечения (метод золотого сечения).
В золотом сечении точки разбиения отрезка $[0, 1]$ имеют вид $\phi = \frac{\sqrt{5}-1}{2}$ и $1 - \phi$ — золотое сечение. Это примерно 0.382 и 0.618.

Время работы: по-прежнему, $O(\log(1/\varepsilon))$ (выигрываем в константе).

Одномерная оптимизация: метод Ньютона

А можно быстрее? Да, если у f существует вторая производная.

Идея: будем последовательно строить касательные параболы к f в точке x_n, в качестве x_{n+1} брать минимум у этой параболы.

Дано: x_0.
Итерация:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

(Это просто нахождение нуля производной методом касательных.)

Свойство: $|x^* - x_n| = \alpha|x^* - x_{n-1}|^2$ (квадратичная скорость сходимости).
В методах сужения отрезка было $|x^* - x_n| = \alpha|x^* - x_{n-1}|$, $\alpha < 1$ (линейная скорость сходимости).

Вопрос: какой метод быстрее сходится?

Многомерная оптимизация

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

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

Теперь $k > 1$.

Идея: свести задачу к одномерной оптимизации. (Как?)
Подход: итеративный, каждая итерация — в два этапа:

1. Поиск направления оптимизации ($\mathbf{x}_n$ — текущая точка, направление — $\mathbf{p}_n$ (как найти?)).
2. Шаг по направлению $\mathbf{p}_n$ либо одномерная оптимизация: $\alpha_n^* = \arg\min_{0 \le \alpha \le 1} f(\mathbf{x}_n + \alpha \mathbf{p}_n)$, $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n^* \mathbf{p}_n$.

Покоординатный спуск

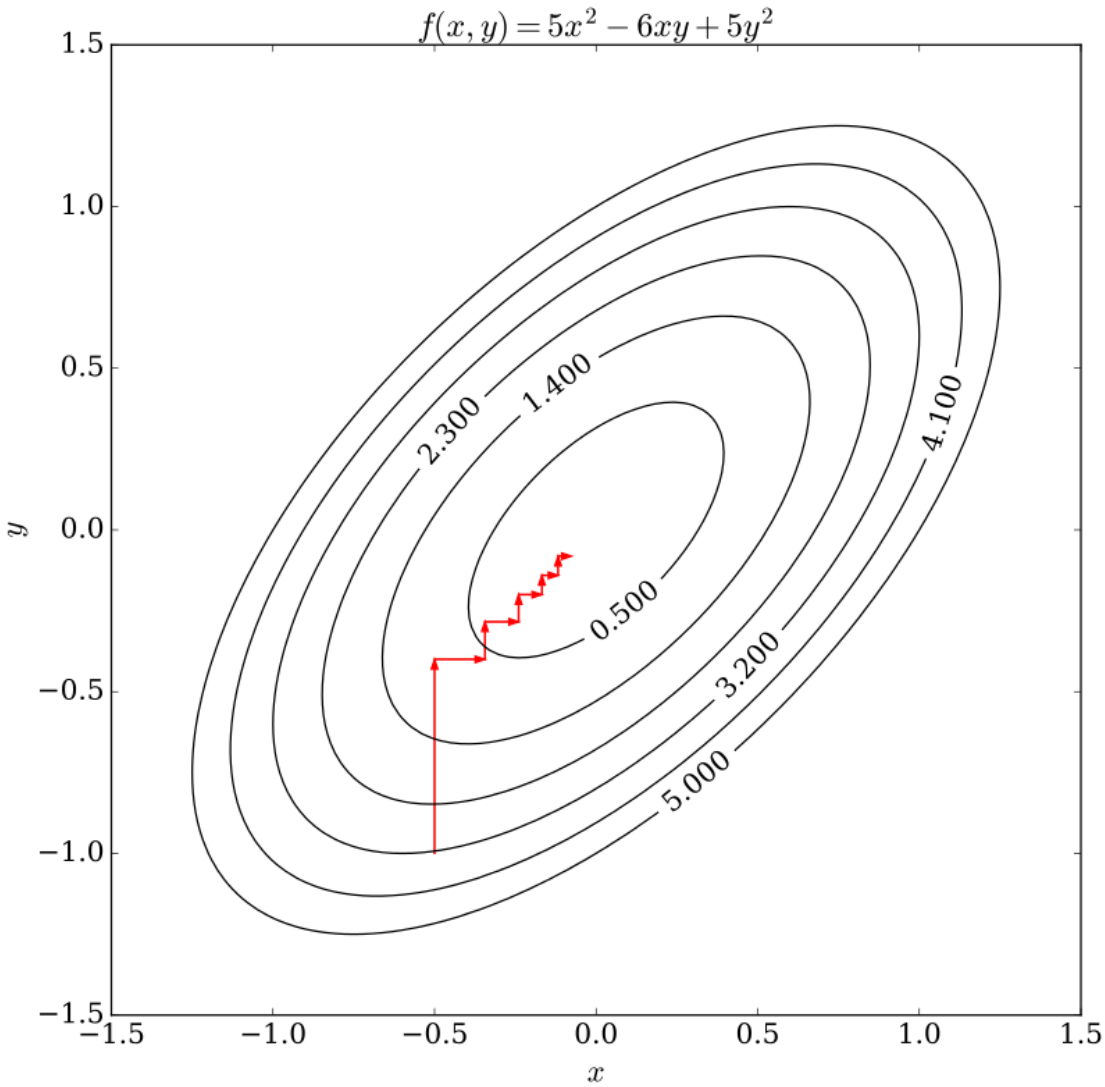
Он же *coordinate descent*. Самое простое, что можно придумать.

В терминах $\mathbf{p}_n$: $\mathbf{p}_n = \mathbf{e}_{(n \mod k)+1}$, $\mathbf{e}_i \in \mathbb{R}^k$ — i-й орт.
Схема итерации та же:
$\alpha_n^* = \arg\min_{\alpha \in \mathbb{R}} f(\mathbf{x}_n + \alpha \mathbf{p}_n)$, $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n^* \mathbf{p}_n$.

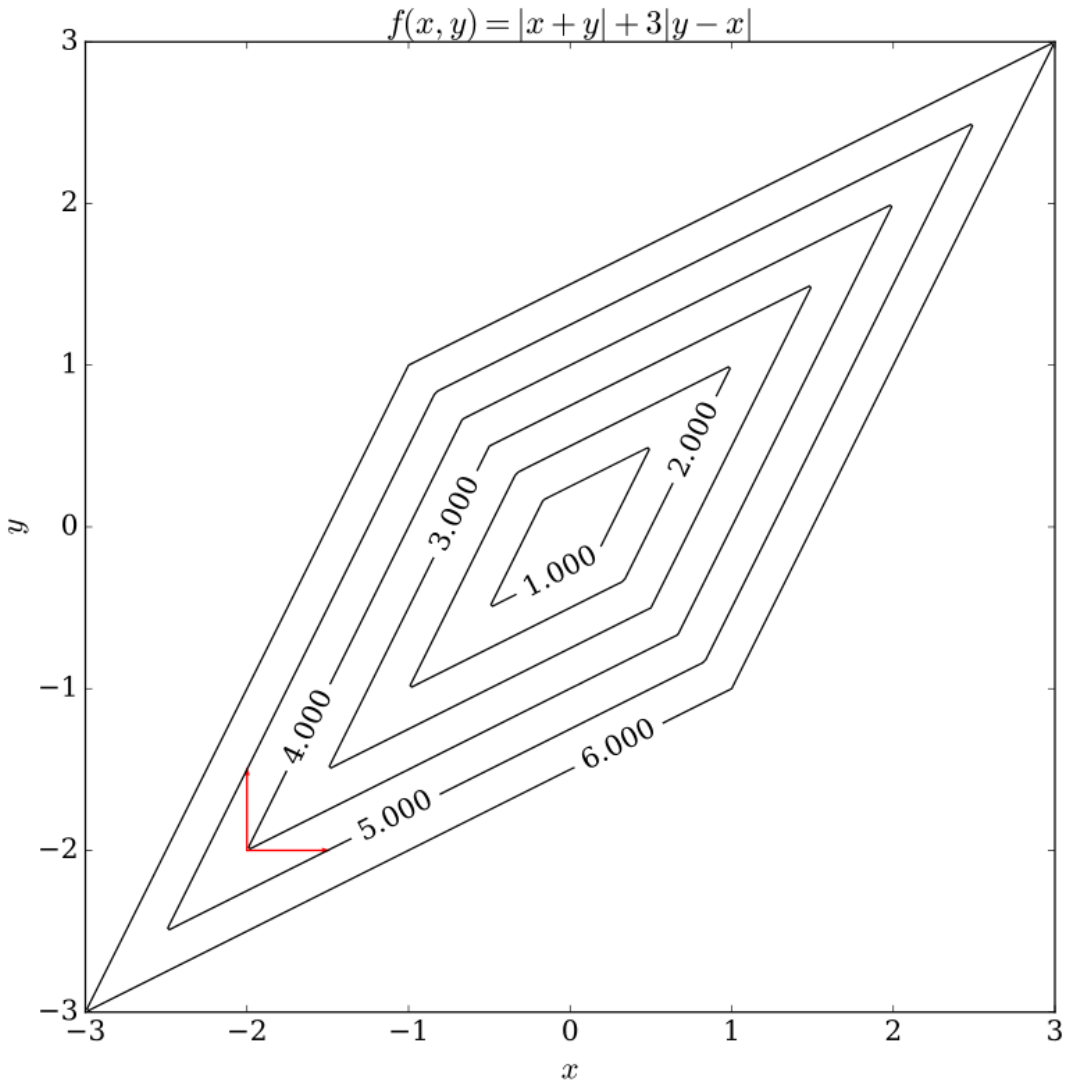
Что выполнено: $f(\mathbf{x}_{n+1}) \leq f(\mathbf{x}_n)$.

Со сходимость у этого метода могут быть проблемы, даже в гладком случае, так как метод останавливается, если имеется условный минимум всего в k направлениях.

Покоординатный спуск: пример, когда есть сходимость



Покоординатный спуск: пример, когда нет сходимости



Покоординатный спуск

Ой! Метод может “застрять” там, где нет локального минимума.
На гладких функциях может сходиться медленно.

Получается, в нём нет смысла?

Идею используют. Например, вместо спуска по одной координате разбивают координаты на два набора, $k_1 + k_2 = k$, и чередуют их. На итерации — многомерная оптимизация по подмножеству координат.

Зачем так делают? Когда напрямую $f(\mathbf{x})$ оптимизировать слишком сложно.

Применения: статистика (важный частный случай — *EM-алгоритм*), ...

Градиентный спуск

Он же *gradient descent*. f — дифференцируемая
$\mathbf{p}_n$ — направление *антиградиента*: $\mathbf{p}_n = -\nabla f(\mathbf{x}_n)$.
Схема итерации та же:
Наискорейший спуск:
$\alpha_n^* = \arg\min_{0 \le \alpha < \infty} f(\mathbf{x}_n + \alpha \mathbf{p}_n), \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n^* \mathbf{p}_n.$

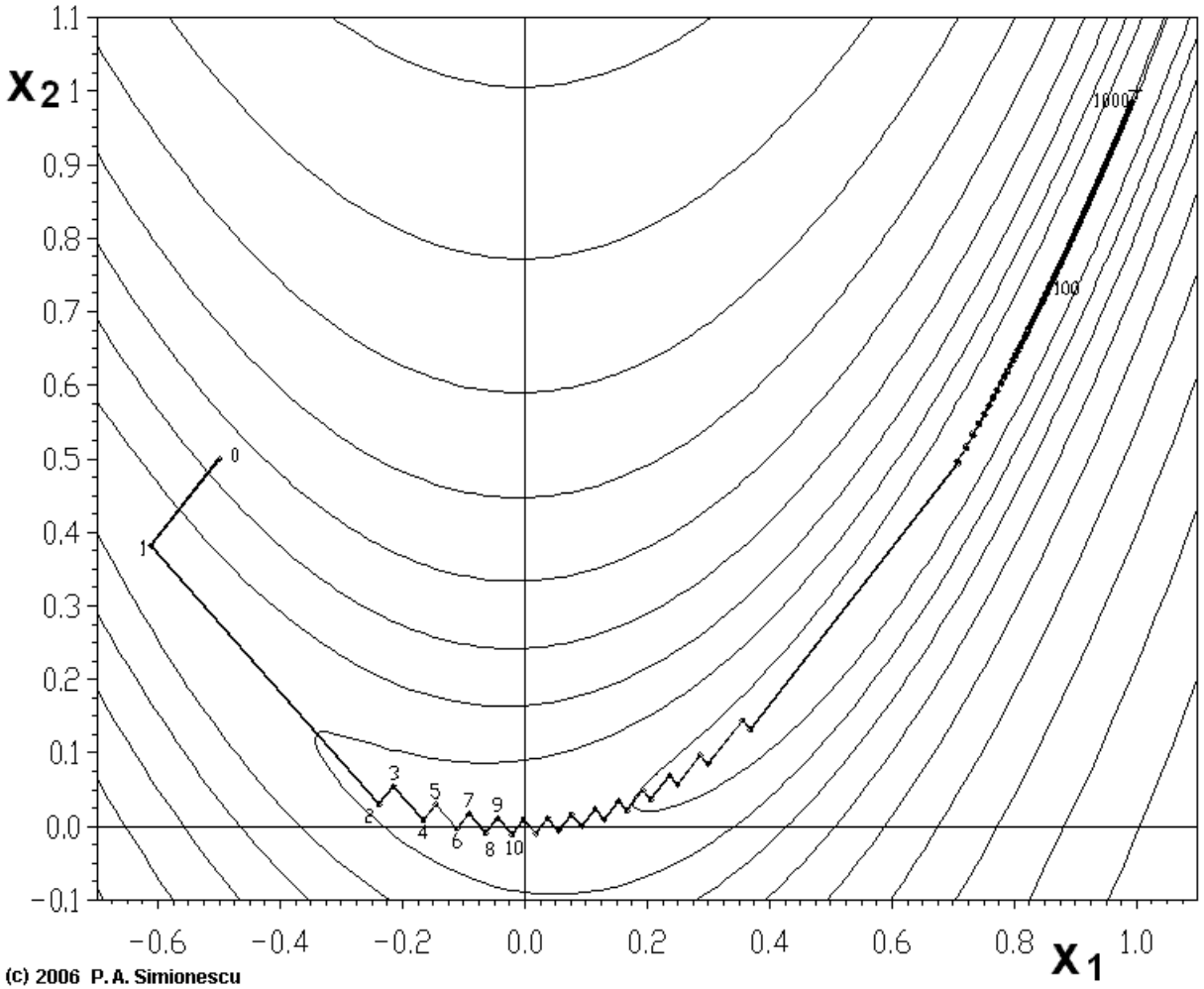
Или шаг по направлению:
$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n$, где шаг α_n уменьшается адаптивно, чтобы было выполнено $f(\mathbf{x}_{n+1}) \le f(\mathbf{x}_n)$.

Линейная сходимость есть при определенных условия (выпуклость f, липшицевость ∇f, ...).

Вопрос. $f(x, y) = x^2 + y^2$: чему равен антиградиент в точке $(1, 1)$?
Какой вариант шага лучше для квадратичной функции $f(x, y) = x^2 + y^2$?

Градиентный спуск: пример медленной сходимости

Rosenbrock function: $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$.



Градиентный спуск: применения

Применяют, когда более сложные методы слишком дороги по времени/памяти. В этом случае применяют не поиск по направлению, а шаг на определенное расстояние.

В статистике (точнее, машинном обучении) применяют вариант градиентного спуска к целевым функциям вида:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{1}^{N} f_i(\mathbf{x}).$$

f_i — это ....
Вместо того, чтобы на каждой итерации оптимизировать f, случайным образом разыгрывается индекс i, и вычисляется шаг по антиградиенту $-\nabla f_i$ с угасающим весом. Получили метод стохастического градиента *(Stochastic gradient descent, SGD)*.
Модификации: AdaGrad, RMSProp, Adam, . . . (2010-е)

Многомерный метод Ньютона

$\mathbf{p}_n = -\nabla^2 f_n^{-1} \nabla f_n$, ∇f_n — градиент (вектор первых производных), $\nabla^2 f_n$ — матрица Гессе (матрица вторых производных) в точке $\mathbf{x}_n$. (Аппроксимация многомерной параболой и вычисление направления в ее точку минимума.)

Квадратичная сходимость есть при определенных условия (дважды дифференцируемость f, липшицевость гессиана).

Вопрос: Сколько шагов требуется методу для сходимости на квадратичной функции $f(x, y) = x^2 + 2y^2$?

Когда можно обойтись без вычисления матрицы Гессе?
Пример: пусть $f = \|\mathbf{y} - g(\mathbf{x})\|^2$, $g : \mathbb{R}^k \to \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^N$.
Метод Гаусса-Ньютона: $\mathbf{p}_n = -(\nabla g(\mathbf{x}_n))^+ (g(\mathbf{x}) - \mathbf{y})$, $\nabla g(\mathbf{x}_n) \in \mathbb{R}^{N \times k}$ — якобиан, $\mathbf{F}^+ = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}$ — псевдообращение Мура-Пенроуза.

На практике: методы *BFGS*, *L-BFGS-B* — используют приближение гессиана (квазиньютоновские методы).

Возможные проблемы

- Выбор начальной точки. Решение: (1) мультистарт, (2) использовать сначала “грубый” метод, даже можно по крупной решетке.
- Выбор критерия остановки
  - $\|\nabla f(\mathbf{x}_k)\| < \varepsilon$
  - $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \varepsilon$
  - $\|\frac{f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})}{f(\mathbf{x}_k)}\| < \varepsilon$

  \+ ограничение на число итераций! (чтобы метод не работал бесконечно долго, если он “застрял”).

  Если мы используем численное вычисление градиента (с какой-то точностью), то надо обратить внимание на ε.
- Область поиска. В бесконечной области искать сложно. Выход: искать в ограниченной области, но если решение на границе, то область расширять.

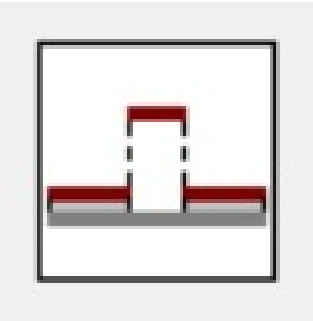
Простейший случайный поиск

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

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

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

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



Параметры: δ — размер окрестности и p — мера локальности поиска.

Развитие случайного поиска

- Адаптивный выбор параметров
- Мультистарт
- Непрерывные генетические алгоритмы

  Шаг в окрестности с улучшением — саморазвитие,
  случайный выбор в области — мутация,
  выбор новых начальных точек — генерация новых особей,
  после мультистарта выбор лучших — селекция,
  выбор новой точки как (например) линейной комбинации двух ранее выбранных — скрещивание,
  оценка приспособленности — вычисление целевой функции.