Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий

Неделя науки 2019

Многомерная оптимизация в машинном обучении

Выполнили:
Данилова Дарья, 3530203/80001
Данилова Александра, 3530203/80001

Научный руководитель:
доцент каф. «Высшая математика»
Филимоненкова Надежда Викторовна

Предмет и задачи

Предмет исследования: методы многомерной оптимизации в машинном обучении.

Задачи:

1) Изучить четыре метода приближенной оптимизации: градиентный спуск, метод Ньютона, BFGS, метод Нелдера-Мида.
2) Провести сравнительный анализ методов: теоретический анализ и численное тестирование.

Постановка задачи оптимизации

Для того, чтобы корректно поставить задачу оптимизации, необходимо задать:

1. *Допустимое множество* — множество $\mathbb{X} = \{\vec{x} |\ g_i(\vec{x}) \leq 0,\ i = 1, \ldots, m\} \subset \mathbb{R}^n$;
2. *Целевую функцию* — отображение $f:\ \mathbb{X} \to \mathbb{R}$;
3. *Критерий поиска* (max или min).

Найти $\vec{x}^* \in \mathbb{X}:\ f(\vec{x}^*) = \min\limits_{\vec{x} \in \mathbb{X}} f(\vec{x})$.

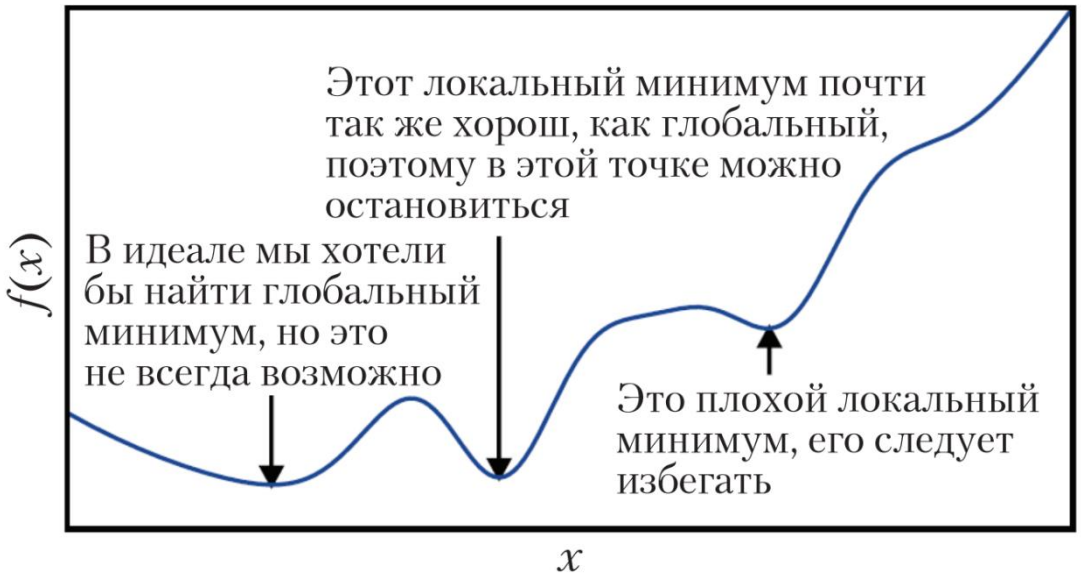
Алгоритмы приближенной оптимизации

- градиентный спуск
- метод Ньютона
- BFGS
- метод Нелдера-Мида

Позволяют приближенно найти локальный минимум.

Итерационные: $x_{k+1} = \varphi(x_k)$.

x_0 - начальное приближение, выбираем самостоятельно.

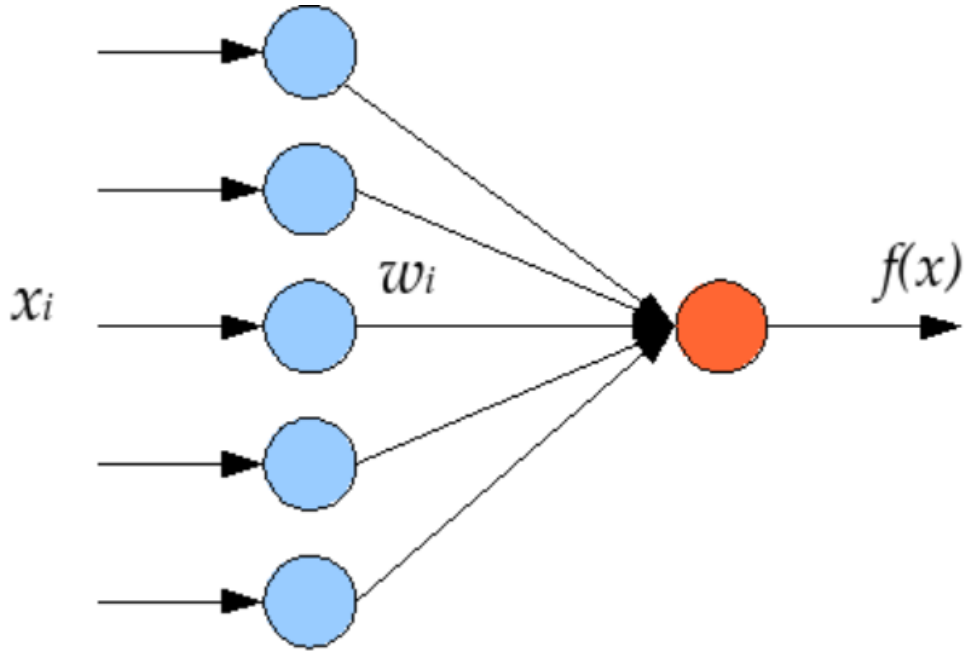


Классификация методов по требованию гладкости функции

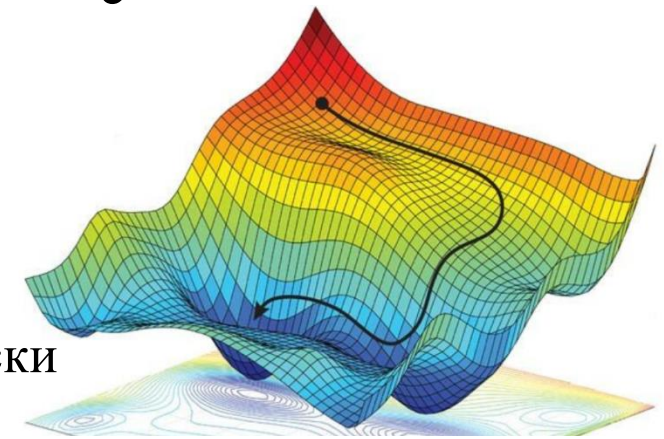
прямые методы: метод Нелдера-Мида	• вычисления целевой функции в точках приближений
методы первого порядка: градиентный спуск	• вычисления первых частных производных функции
методы второго порядка: метод Ньютона, BFGS	• вычисления вторых частных производных

Градиентный спуск

- Применяется для обучения нейросети типа «перцептрон».
- Изменяем весовые коэффициенты сети w_i так, чтобы минимизировать среднюю ошибку на выходе $f(x)$ нейронной сети при подаче на вход последовательности обучающих входных данных x_i.



Градиентный спуск



Главные особенности:

- прост в реализации локальной оптимизации
- имеет слабые условия сходимости
- на практике размер шага подбирают эмпирически

Идея: идти в направлении наискорейшего спуска, которое задаётся антиградиентом $-\nabla f$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

до тех пор, пока не выполнится условие

$$x_{k+1} - x_k \leq \varepsilon \ (\Rightarrow (x_{k+1} \approx x_k \Leftrightarrow \nabla f(x) \approx 0)),$$

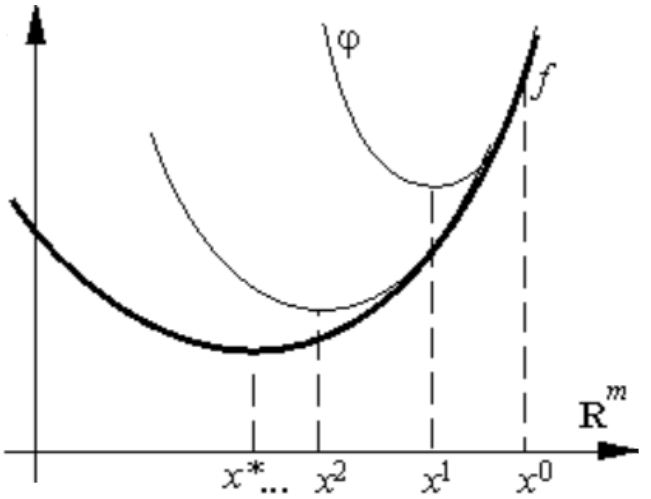
где ε – точность расчёта, α_k – размер шага.

Метод Ньютона

Главные особенности:

- хорошо работает, когда близок к решению
- не справляется с седловыми точками
- можно обучить сети с небольшим числом параметров

(обращение к матрице размера $k \times k$ требует $O(k^3)$ вычислительных ресурсов)



Идея: заменить функцию f в окрестности текущего приближения x_k разложением в ряд Тейлора с точностью до членов второго порядка.

$$f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k) = q(x).$$

Матрица $H(x_k)$ положительно определена $\Longrightarrow q(x)$ выпукла $\Longrightarrow q(x)$ имеет единственный минимум $\Longrightarrow$ минимум как решение системы уравнений:

$$\nabla q(x_{k+1}) = 0 \Leftrightarrow 0 - \nabla f(x_k) - H(x_k)(x_{k+1} - x_k) = 0 \Leftrightarrow -\nabla f(x_k) = H(x_k)(x_{k+1} - x_k).$$

$$\boxed{x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)},$$

где $-H^{-1}(x_k)\nabla f(x_k)$ – направление спуска.

Алгоритм Бройдена - Флетчера - Гольдфарба - Шанно (BFGS)

Главные особенности алгоритма:

- гессиан функции вычисляется приближенно, исходя из сделанных до этого шагов о накопленной информации о кривизне функции (квазиньютоновский метод);
- не используются сложные математические операции;
- не пригоден для современных моделей глубокого обучения, насчитывающих миллионы параметров.

Идея: приближенная оценка матрицы Гессе строится в процессе накопления информации о кривизне функции.

Алгоритм Бройдена - Флетчера - Гольдфарба - Шанно (BFGS)

В качестве начального приближения можно взять матрицу Гессе функции, вычисленную в начальной точке x_0.

$$C_{k+1} = (I - \rho_k s_k y_k^T) C_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

где $\rho_k = \dfrac{1}{y_k^T s_k}$,

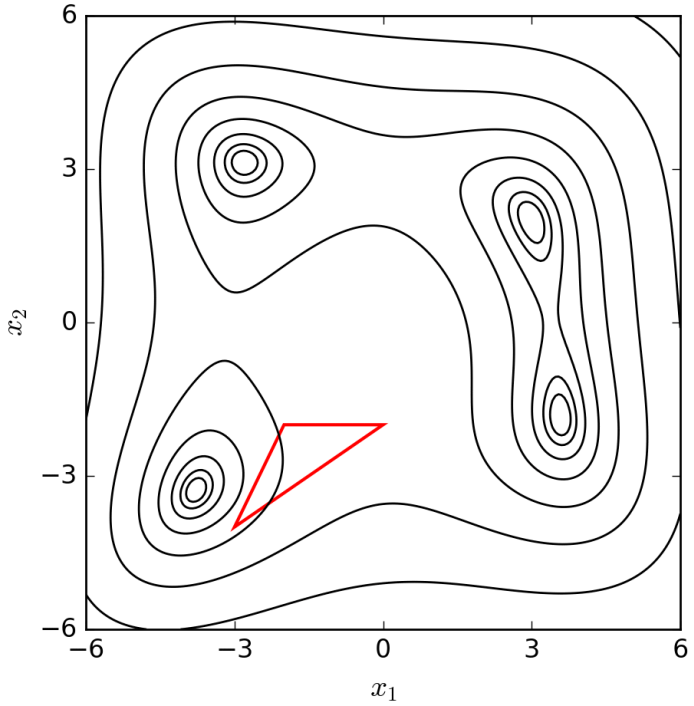
$s_k = x_{k+1} - x_k$ – шаг алгоритма на итерации,

$y_k = \nabla f_{k+1} - \nabla f_k$ – изменение градиента на итерации.

Метод Нелдера-Мида

Главные особенности:

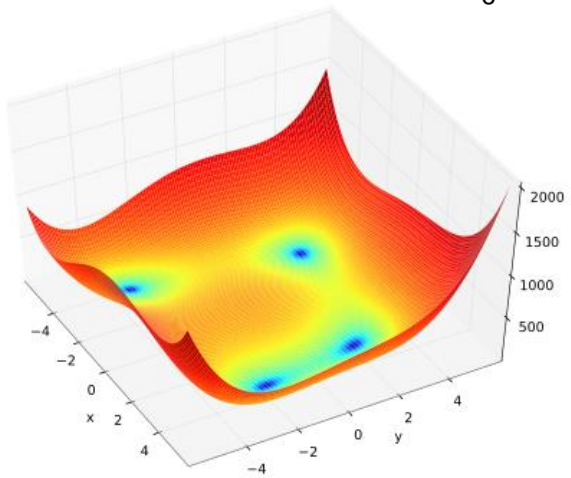
- нет ограничений на гладкость функции
- эффективен при низкой скорости вычисления минимизируемой функции
- отсутствие теории сходимости



Идея:

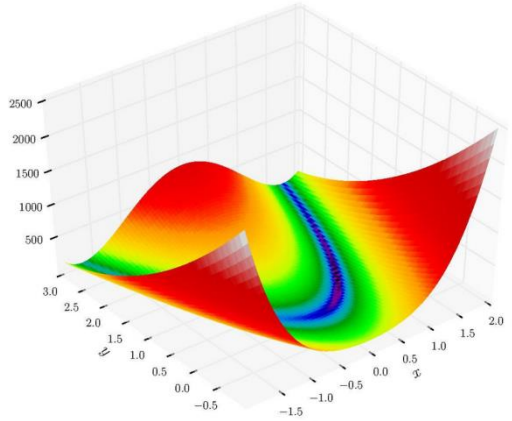
Формирование симплекса из (n + 1) точек вблизи локального экстремума. Его деформирование в направлении экстремума, посредством четырех операций:

- отражение;
- сжатие;
- растяжение;
- редукция.



Примеры

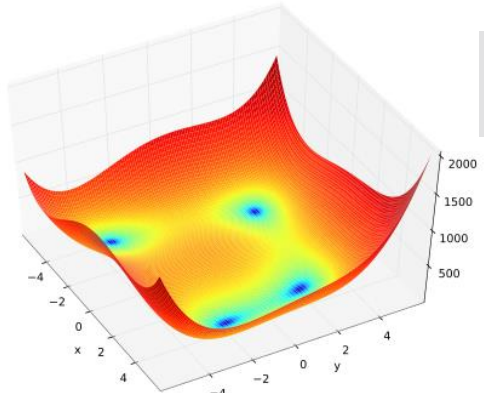
Функция Розенброка



$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

$$f(1,1) = 0 \qquad -\infty < x < +\infty$$

Функция Химмельблау

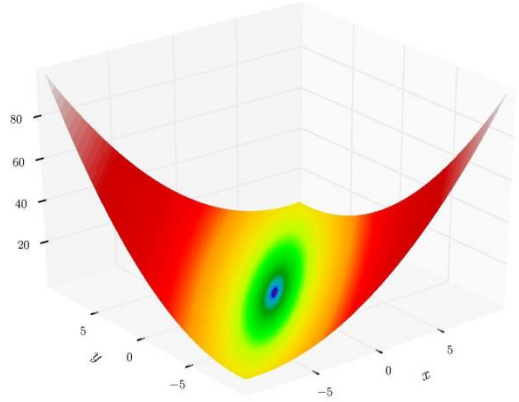


$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

$$\text{Min} = \begin{cases} f(3.0, 2.0) & = 0.0 \\ f(-2.805118, 3.131312) & = 0.0 \\ f(-3.779310, -3.283186) & = 0.0 \\ f(3.584428, -1.848126) & = 0.0 \end{cases}$$

$$-5 \le x, y \le 5$$

Функция Матьяса



$$f(x,y) = 0.26\left(x^2 + y^2\right) - 0.48xy$$

$$f(0,0) = 0 \qquad -10 \le x, y \le 10$$

Результаты

- среда тестирования – **Jupyter Notebook**
- модуль scipy.optimize
- точность $\varepsilon = 10^{-8}$

Метод	Функция Розенброка		Функция Химмельблау		Функция Матьяса	
	количество итераций	время	количество итераций	время	количество итераций	время
Градиентный спуск	8	1.62 мс	7	1.61 мс	5	947 мкс
Ньютон	9	1.7 мс				
BFGS	8	927 мкс	8	1.35 мс	5	479 мкс
Нелдер-Мид	46	3.27 мс	57	3.33 мс	39	2.24 мс

Название метода	Достоинства	Недостатки	Сходимость	Объем памяти
Градиентный спуск	• прост и интуитивен; • обычно обходят седловые точки.	• медленный при движении по оврагу	слабая сходимость	$O(n^2)$
Метод Ньютона	• высокая скорость сходимости вблизи экстремума; • использует информацию о кривизне.	• высокий шанс уйти не туда, если находится далеко от экстремума; • не справляется с седловыми точками.	квадратичная сходимость	$O(n^3)$
BFGS	• эффективен и устойчив; • по опыту хорошо справляется с невыпуклыми функциями.	• сложен в реализации; • непригоден для моделей, насчитывающих миллионы параметров.	значительно более точная и быстрая сходимость	$O(n^2)$
Метод Нелдера-Мида	• применяется для негладких и зашумленных функций; • просто устроен, легко реализовать.	• отсутствие полной теории сходимости; • метод может расходиться даже на гладких функциях.	полная теория сходимости отсутствует	$O(n)$

Какой алгоритм выбрать?

- статья «Natural Evolution Strategies» в *Journal of Machine Learning Research*
- единого мнения нет, победитель не был выявлен

Tom Schaul, Ph.D. - старший научный сотрудник **Google DeepMind**.

Список источников

1. Бенджио И., Гудфеллоу Я., Курвилль А. Глубокое обучение.
2. Test functions for optimization URL: https://en.wikipedia.org/wiki/Test_functions_for_optimization
3. Метод оптимизации Нелдера — Мида. Пример реализации на Python URL: https://habr.com/ru/post/332092/
4. Метод BFGS или один из самых эффективных методов оптимизации. Пример реализации на Python URL: https://habr.com/ru/post/333356/
5. Nelder–Mead method URL: https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method
6. Градиентный спуск URL : https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA
7. §4. Метод Ньютона URL: http://w.ict.nsc.ru/books/textbooks/akhmerov/mo/4.html

Спасибо за внимание!