

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК XXXXX

**Отчет об исследовательском проекте на тему:
Глобальные градиентные методы оптимизации**

Выполнил:

студент группы БПМИ214
Судаков Илья Александрович

(подпись)

(дата)

Принял руководитель проекта:

Посыпкин Михаил Анатольевич
Научный сотрудник
Факультета компьютерных наук НИУ ВШЭ

(подпись)

(дата)

Москва 2023

Содержание

| | |
|--|-----------|
| Аннотация | 3 |
| 1 Координатный спуск | 4 |
| 2 Градиентный спуск | 6 |
| 3 Метод золотого сечения | 7 |
| 4 Глобальная одномерная оптимизация | 10 |
| 5 Тестирование | 14 |
| 6 Примеры | 23 |
| 6.1 Ссылки на статьи | 23 |
| 6.2 Рисунки | 23 |
| 6.3 Таблицы | 23 |
| 6.4 Формулы | 23 |
| Список литературы | 25 |

Аннотация

Ваша аннотация на русском языке.

Ключевые слова

Глубинное обучение, разреживание моделей, рекуррентные нейронные сети

1 Координатный спуск

Рассмотрим один из наиболее простых методов многометрной оптимизации - **координатный спуск**. На каждой итерации происходит оптимизация по одной переменной, то есть происходит поиск минимума вдоль заданного направления. Существуют различные порядки перебиравания направлений поиска, в простейшем варианте они перебираются по порядку.

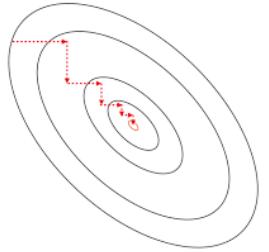


Рис. 1.1: Наглядное представление метода

Замечание. Алгоритм не требует вычисления производной функции в точке.

Алгоритм. Пусть целевая функция имеет вид: $f(x) : \mathbb{X} \rightarrow \mathbb{R}$, где $\mathbb{X} \subset \mathbb{R}^n$

Задача оптимизации задана в следующем виде: найти $x_{opt} : f(x_{opt}) = \min_{x \in \mathbb{X}} f(x) = \mu$

- Задать начальное приближение и точность расчёта: X_0, ε
- Рассчитать $x_i^{k+1} = \arg \min_{y \in \mathbb{X}_i} f(x_1^{k+1}, \dots, x_{i-1}^{k+1}, y, x_{i+1}^k, \dots, x_n^k)$
- Проверить условие остановки (на усмотрение):
 - $|x^j - x^{j+1}| < \varepsilon$
 - $|F(x^j) - F(x^{j+1})| < \varepsilon$
 - иначе перейти к пункту 2

Таким образом, на каждой итерации будет выполнено $F(x^0) \geq F(x^1) \geq F(x^2) \geq \dots$.

Реализация

```
1 def FastSearch(func_index, D, p, e_d, e_f, method): # F(function), D(set), p(start
2     point), e(error)
3     while True:
4         p_0 = p
5         for index in range(len(p)):
6             p = method(func_index, index, D[index][0], D[index][1], p, e_d, e_f)
7             if (norm(p_0, p) < e_d) & (Functions[func_index * 2](p_0) - Functions[
8                 func_index * 2](p) < e_f):
9                 break
10    best_point = p
11    best_value = Functions[func_index * 2](p)
```

```
11     return best_point, best_value
```

Замечание. Чтобы найти $x_i^{k+1} = \arg \min_{y \in \mathbb{X}_i} f(x_1^{k+1}, \dots, x_{i-1}^{k+1}, y, x_{i+1}^k, \dots, x_n^k)$ будем использовать метод золотого сечения и алгоритм глобальной одномерной оптимизации.

2 Градиентный спуск

Другой, не менее известный метод многомерной оптимизации - **градиентный спуск**. На каждой итерации вычисляется градиент функции в текущей точке, вдоль которого происходит поиск минимума вдоль направления.

Замечание. Алгоритм требует вычисления градиента функции в точке, поэтому необходимо, чтобы функция была дифференцируемой.

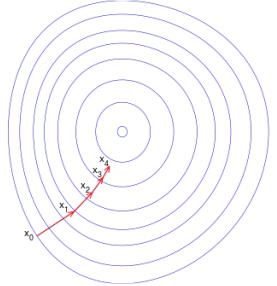


Рис. 2.1: Наглядное представление метода

Алгоритм. Пусть целевая функция имеет вид: $f(x) : \mathbb{X} \rightarrow \mathbb{R}$, где $\mathbb{X} \subset \mathbb{R}^n$

Задача оптимизации задана в следующем виде: найти $x_{opt} : f(x_{opt}) = \min_{x \in \mathbb{X}} f(x) = \mu$

- Задать начальное приближение и точность расчёта: X_0, ε
- Рассчитать $x^{j+1} = x^j - \lambda^j \nabla F(x^j)$
- Проверить условие остановки (на усмотрение):
 - $|x^j - x^{j+1}| < \varepsilon$
 - $|F(x^j) - F(x^{j+1})| < \varepsilon$
 - $\|\nabla F(x^i)\| < \varepsilon$
 - иначе перейти к пункту 2

Замечание. $x^{j+1} = x^j - \lambda^j \nabla F(x^j)$, где λ^j задает скорость градиентного спуска и может быть выбрана как:

- постоянная, тогда метод может не сходиться
- убывать по какому-то закону
- гарантировать наискорейший спуск

Модификация: метод наискорейшего спуска

В случае наискорейшего спуска λ^j определяется как:

$$\lambda^j = \operatorname{argmin}_{\lambda} F(x^{j+1}) = \operatorname{argmin}_{\lambda} F(x^j - \lambda \nabla F(x^j))$$

Замечание. Чтобы вычислить λ^j , будем использовать метод золотого сечения и алгоритм Moore-Skelboe.

3 Метод золотого сечения

Метод золотого сечения — это эффективный и простой способ нахождения локального минимума функции на заданном отрезке, за счет того, что на очередной итерации требует вычисления значения функции только в одной точке, что может быть очень важно, для сложновычислимых функций.

Замечание. Метод золотого сечения находит глобальный минимум только в случае унимодальных функций.

Определение. Функция **унимодальная** на отрезке $[a, b]$, если $\exists \alpha, \beta : a \leq \alpha \leq \beta \leq b$ такие, что:

- на отрезке $[a, \alpha]$ функция монотонно убывает
- на отрезке $[\beta, b]$ функция монотонно возрастает
- $\forall x \in [\alpha, \beta] : f(x) = \min_{x \in [a, b]} f(x)$

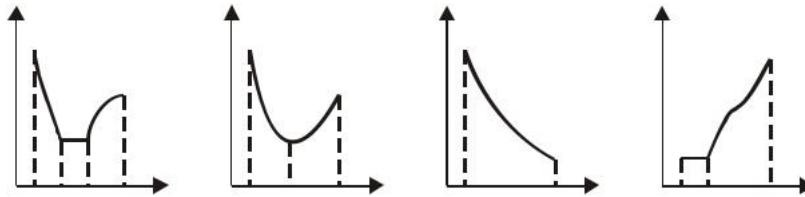


Рис. 3.1: Примеры унимодальных функций

Свойство. Любой локальный минимум будет являться одновременно и глобальным минимумом.

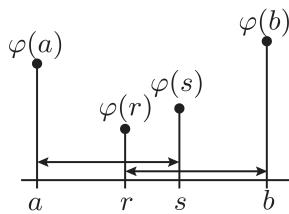


Рис. 3.2: Итерация

Алгоритм. Пусть хотим найти минимум функции φ на отрезке $[a, b]$, разобьем его на три части двумя точками r, s так, чтобы при отсекании одного из крайних подотрезков, одна из точек стала границей подотрезка для следующей итерации, то есть соотношение длин подотрезков оставалось прежним:

$$|[a, r]| = |[s, b]| = l_1$$

$$|[r, s]| = l_2$$

тогда, если $\varphi(r) \leq \varphi(s)$, то s станет правой границей рассматриваемого отрезка, а r станет правой внутренней точкой:

$$\frac{|[a, s]|}{|[a, b]|} = \frac{|[a, r]|}{|[a, s]|} \Rightarrow \frac{l_1 + l_2}{2l_1 + l_2} = \frac{l_1}{l_1 + l_2} \Rightarrow l_1^2 - l_1 \cdot l_2 - l_2^2 = 0 \Rightarrow \frac{l_1}{l_2} = \frac{1 + \sqrt{5}}{2} = \Phi \approx 1.618$$

Замечание. Число Φ называют золотым сечением

Несмотря на то, что данный метод гарантированно находит глобальный минимум только для унимодальных функций, мы будем его использовать для сравнения с более надежными методами.

Реализация

```

1 def GoldenRatio(func_index, index, a, b, p, e_d, e_f): # f(function), i(index of
   direction),
2   # a(left border), b(right border), p(current point), e_d(error of d), e_f(error
   of f)
3   F = Functions[func_index * 2]
4   phi = (1 + np.sqrt(5)) / 2 # constant of golden ratio
5   x_1 = b - (b - a) / phi
6   x_2 = a + (b - a) / phi
7   p_1 = p.copy() # current point
8   p_2 = p.copy() # current point
9   p_1[index] = x_1
10  p_2[index] = x_2
11  f_1 = F(p_1) # value in 1-st point
12  f_2 = F(p_2) # value in 2-nd point
13  while (b - a > e_d) | (abs(f_1 - f_2) > e_f): # termination criteria
14    if f_1 <= f_2:
15      b = x_2
16      x_2 = x_1
17      x_1 = b - (b - a) / phi
18
19      p_1[index] = x_1
20      p_2[index] = x_2
21
22      f_2 = f_1
23      f_1 = F(p_1)
24    else:
25      a = x_1
26      x_1 = x_2

```

```

27     x_2 = a + (b - a) / phi
28
29     p_1[index] = x_1
30     p_2[index] = x_2
31
32     f_1 = f_2
33     f_2 = F(p_2)
34
35 best_point = []
36 for i in range(len(p)):
37     best_point.append((p_1[i] + p_2[i]) / 2)
38
39 return best_point # point of extremum with error e_d

```

Сложность

На каждой итерации длина рассматриваемого отрезка умножается на

$$\frac{l_1 + l_2}{l_1 + l_2 + l_2} = \frac{3 + \sqrt{5}}{4 + 2\sqrt{5}} = \Phi^{-1} = \phi$$

Для достижения точности δ потребуется

$$\frac{\ln(\frac{|[a,b]|}{\delta})}{\ln(\Phi)} \approx 2 \ln\left(\frac{|[a,b]|}{\delta}\right)$$

итераций.

4 Глобальная одномерная оптимизация

Алгоритм поиска глобального минимума функции вдоль одного направления будет основываться на интервальном анализе. Приведем основные определения и теоремы, которые нам пригодятся для построения алгоритма.

Определение. Интервалом $[a, b]$ называется следующее множество:

$$[a, b] := \{x \in \mathbb{R} | a \leq x \leq b\}$$

в других обозначениях:

$$\mathbf{x} := [\underline{\mathbf{x}}, \bar{\mathbf{x}}]$$

где $\underline{\mathbf{x}}, \bar{\mathbf{x}}$ - левая и правая граница интервала соответственно.

Арифметические свойства интервалов.

- $\mathbf{x} + \mathbf{y} = [\underline{\mathbf{x}} + \underline{\mathbf{y}}, \bar{\mathbf{x}} + \bar{\mathbf{y}}]$
- $\mathbf{x} - \mathbf{y} = [\underline{\mathbf{x}} - \bar{\mathbf{y}}, \bar{\mathbf{x}} - \underline{\mathbf{y}}]$
- $\mathbf{x} \cdot \mathbf{y} = [\min(\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\bar{\mathbf{y}}, \bar{\mathbf{x}}\underline{\mathbf{y}}, \bar{\mathbf{x}}\bar{\mathbf{y}}), \max(\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\bar{\mathbf{y}}, \bar{\mathbf{x}}\underline{\mathbf{y}}, \bar{\mathbf{x}}\bar{\mathbf{y}})]$
- $\mathbf{x}/\mathbf{y} = \mathbf{x} \cdot [1/\bar{\mathbf{y}}, 1/\underline{\mathbf{y}}]$, если $\underline{\mathbf{y}} > 0$

Пример. Пусть известно, что первый гонщик проезжает длину гонки за время от 80 до 100 минут, а второй от 85 до 90 минут, какая может быть разница во времени приезда к финишу? Воспользуемся вторым арифметическим свойством:

$$\mathbf{T}_1 = [80, 100]$$

$$\mathbf{T}_2 = [85, 90], \text{ тогда результатом будет интервал } \mathbf{T}_1 - \mathbf{T}_2 = [80 - 90, 100 - 85] = [-10, 15]$$

То есть первый гонщик может как приехать на 10 минут раньше, так и задержаться на 15 минут.

Замечание. Так как интервалы являются множествами, то для них можно определить частичный порядок относительно включения:

$$\mathbf{a} \subseteq \mathbf{b} \iff \underline{\mathbf{b}} \leq \underline{\mathbf{a}} \cap \bar{\mathbf{a}} \leq \bar{\mathbf{b}}$$

Свойство. Интервалы обладают свойством монотонности относительно арифметических операций:

Пусть $\mathbf{a} \subseteq \mathbf{a}', \mathbf{b} \subseteq \mathbf{b}', \star \subseteq \{+, -, \cdot, /\}$, тогда $\mathbf{a} \star \mathbf{b} \subseteq \mathbf{a}' \star \mathbf{b}'$

Теорема. Основная теорема интервальной арифметики

Пусть $f(x_1, \dots, x_n)$ - функция вещественных аргументов x_1, \dots, x_n , и для нее определен ре-

зультат $\mathbf{F}(\mathbf{X}_1, \dots, \mathbf{X}_n)$ подстановки вместо аргументов интервалов которые они пробегают $(\mathbf{X}_1, \dots, \mathbf{X}_n) \subset \mathbb{R}^n$ и для $(\mathbf{X}_1, \dots, \mathbf{X}_n)$ операции выполняются по правилам интервальной арифметики. Тогда выполнено следующее:

$$\{f(x_1, \dots, x_n) | x_1 \in \mathbf{X}_1, \dots, x_n \in \mathbf{X}_n\} \subseteq \mathbf{F}(\mathbf{X}_1, \dots, \mathbf{X}_n)$$

Предложение. Пусть $f(x_1, \dots, x_n)$ - функция вещественных аргументов x_1, \dots, x_n , и $\mathbf{F}(\mathbf{X}_1, \dots, \mathbf{X}_n)$ соответствующая ей интервальная функция, тогда выполняется монотонность по включению:

Пусть $\mathbf{X}_1, \dots, \mathbf{X}_n$ и $\mathbf{Y}_1, \dots, \mathbf{Y}_n$, такие что $\mathbf{X}_1 \subseteq \mathbf{Y}_1, \dots, \mathbf{X}_n \subseteq \mathbf{Y}_n$, тогда:

$$\mathbf{F}(\mathbf{X}_1, \dots, \mathbf{X}_n) \subseteq \mathbf{F}(\mathbf{Y}_1, \dots, \mathbf{Y}_n)$$

Определение. Пусть $N > 0$ натуральное число, тогда если $\langle S_0, \dots, S_{n-1} \rangle$ семейство непустых подмножеств множества S , тогда будем называть его покрытием внутри S . В частности, если объединение S_0, \dots, S_{n-1} равно S , тогда последовательность $\langle S_0, \dots, S_{n-1} \rangle$ является покрытием S .

Теорема. Рассмотрим задачу глобальной оптимизации. Пусть $\langle B_0, \dots, B_{n-1} \rangle$ семейство множеств, содержащее глобальный минимум, такое что упорядочено по возрастанию нижней границы $\mathbf{F}(B_i)$ для $i = 0, 1, 2, \dots, N - 1$. Пусть U наименьшее из верхних значений функции для подмножеств $\langle \mathbf{F}(B_0), \dots, \mathbf{F}(B_{n-1}) \rangle$. Тогда интервал $[lb(\mathbf{F}(B_0)), U]$ содержит глобальным минимум μ .

Используя данную теорию, напишем алгоритм поиска глобального минимума вдоль направления.

Алгоритм Moore-Skelboe

Реализация

```

1 def MooreSkelboe(func_index, index, a, b, p, e_d,
2                     e_f): # func_index(number of function in list of functions),
3                     index(index of direction),
4                     # a(left border), b(right border), p(current point), e_d(error of d), e_f(error
5                     of f)
5     F = Functions[func_index * 2 + 1]
6     interval_d = []

```

```

6     for i in range(len(p)):
7         interval_d.append(interval[p[i], p[i]])
8     interval_d[index] = interval[a, b]
9
10    interval_f = F(interval_d)
11    set_of_intervals = [[interval_d, interval_f]]
12    U = right(interval_f)
13    w_f = right(interval_f) - left(interval_f)
14    w_d = right(interval_d[index]) - left(interval_d[index])
15    best_interval = set_of_intervals[0]
16    while (w_d > e_d) | (w_f > e_f):
17        set_of_intervals.pop(0)
18        mid_p = mid(best_interval[0][index])
19        interval_1 = best_interval[0].copy()
20        interval_2 = best_interval[0].copy()
21        interval_1[index] = interval[left(best_interval[0][index]), mid_p]
22        interval_1_f = F(interval_1)
23        interval_2[index] = interval[mid_p, right(best_interval[0][index])]
24        interval_2_f = F(interval_2)
25        U = min(U, right(interval_1_f))
26        U = min(U, right(interval_2_f))
27
28        for i in range(len(set_of_intervals)):
29            if U < left(set_of_intervals[i][1]):
30                set_of_intervals = set_of_intervals[:i]
31                break
32
33        val_1 = left(interval_1_f)
34        val_2 = left(interval_2_f)
35
36        if (len(set_of_intervals) == 0) or (val_1 > left(
37            set_of_intervals[-1][1])):
38            set_of_intervals.append([interval_1, interval_1_f])
39        else:
40            l = 0
41            r = len(set_of_intervals) - 1
42            while l < r:
43                m = int((l + r) / 2)
44                if left(set_of_intervals[m][1]) > val_1:
45                    r = m

```

```

46             else:
47                 l = m + 1
48                 set_of_intervals.insert(l, [interval_1, interval_1_f])
49
50             if val_2 > left(set_of_intervals[-1][1]):
51                 set_of_intervals.append([interval_2, interval_2_f])
52             else:
53                 l = 0
54                 r = len(set_of_intervals) - 1
55                 while l < r:
56                     m = int((l + r) / 2)
57                     if left(set_of_intervals[m][1]) > val_2:
58                         r = m
59                     else:
60                         l = m + 1
61                 set_of_intervals.insert(l, [interval_2, interval_2_f])
62
63             best_interval = set_of_intervals[0]
64             w_f = right(best_interval[1]) - left(best_interval[1])
65             w_d = right(best_interval[0][index]) - left(best_interval[0][index])
66
67             best_point = []
68             for i in range(len(p)):
69                 best_point.append(mid(best_interval[0][i]))
70
71             return best_point

```

Сложность

На каждой итерации длина одного из интервалов в семействе интервалов делится на 2.

Каждая итерация совершается за $O(\log(\text{size}))$. Для достижения точности δ потребуется

$$\frac{\ln(\frac{|[a,b]|}{\delta})}{\ln(\Phi)} \approx 2 \ln\left(\frac{|[a,b]|}{\delta}\right)$$

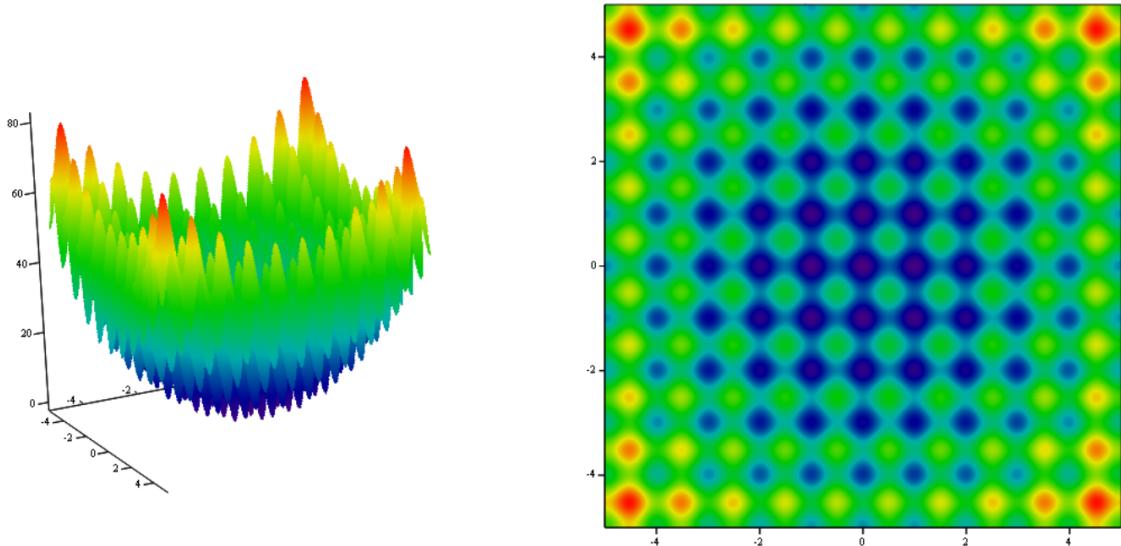
итераций.

5 Тестирование

Ниже приведены результаты тестирования двух методов оптимизации на тестовых функциях, первый использует метод золотого сечения, для поиска минимума вдоль направления, а другой алгоритм Moore-Skelboe.

Функция Растигина

График функции при $n = 2$



Формула

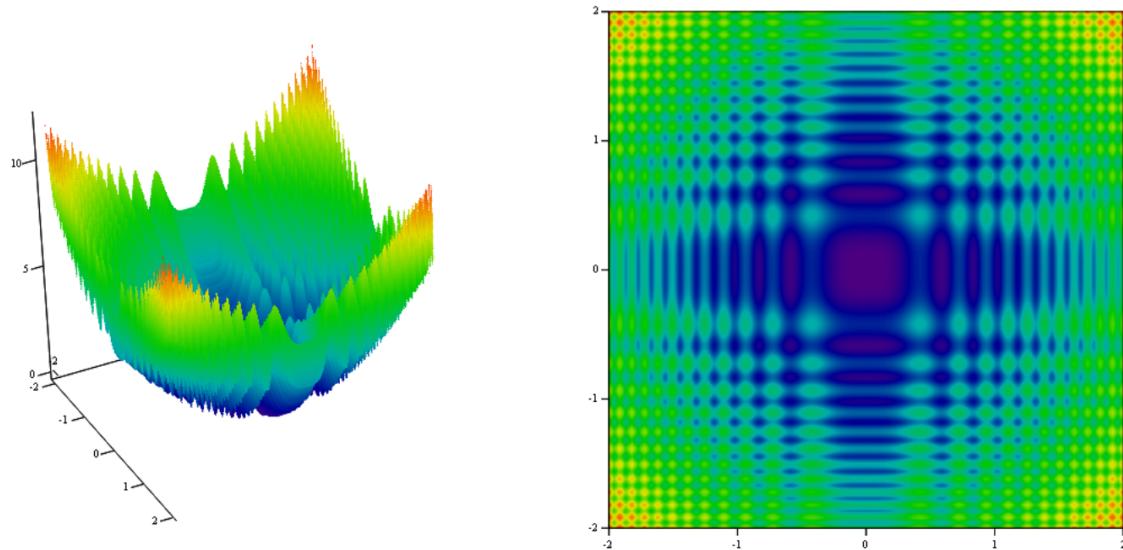
$$f(x) = 10n + \sum_{i=1}^n \left(x_i^2 - 10 \cdot \cos(2\pi \cdot x_i) \right)$$

Результаты тестирования

| n | точность | метод | время | минимум |
|----|-----------|---------------|---------|------------------------|
| 5 | 10^{-3} | Golden ratio | 2.22 ms | 4.974821275706137 |
| | 10^{-3} | Moore-Skelboe | 178 ms | 9.238348694395881e-05 |
| 10 | 10^{-3} | Golden ratio | 4.78 ms | 9.94964255141227 |
| | 10^{-3} | Moore-Skelboe | 736 ms | 0.00018476697390212848 |
| 20 | 10^{-3} | Golden ratio | 13.3 ms | 19.899285102824635 |
| | 10^{-3} | Moore-Skelboe | 2.52 s | 0.0003695339478184678 |
| 30 | 10^{-3} | Golden ratio | 24.1 ms | 29.84892765423696 |
| | 10^{-3} | Moore-Skelboe | 5.38 s | 0.0005543009217348072 |
| 40 | 10^{-3} | Golden ratio | 43.2 ms | 39.79857020564907 |
| | 10^{-3} | Moore-Skelboe | 9.58 s | 0.0007390678956511465 |
| 50 | 10^{-3} | Golden ratio | 58.8 ms | 49.748212757061154 |
| | 10^{-3} | Moore-Skelboe | 14.9 s | 0.0009238348695674858 |

Функция Растрогина новгородская

График функции при $n = 2$



Формула

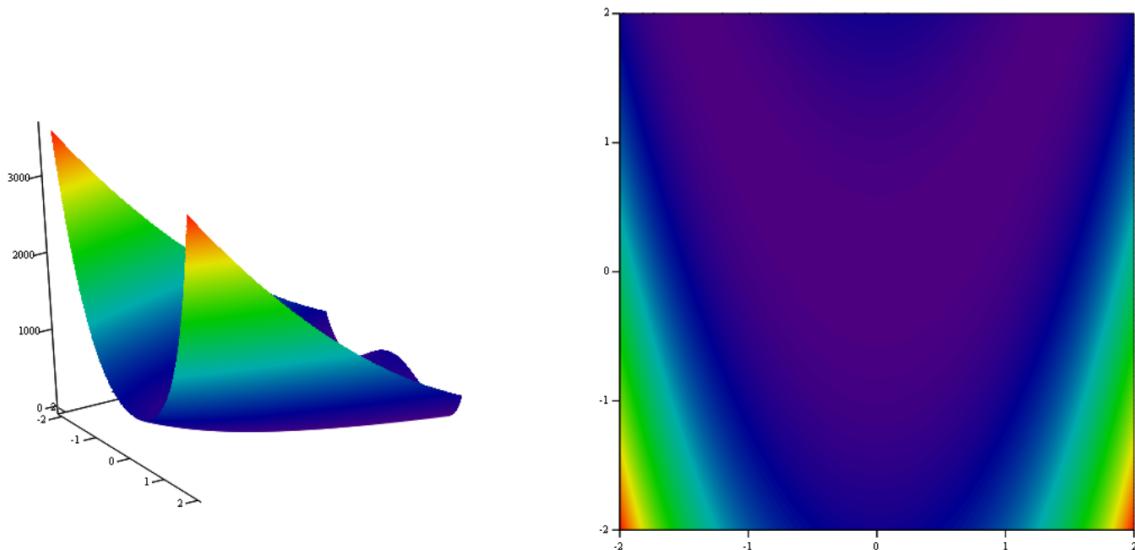
$$f(x) = n + \sum_{i=1}^n \left(x_i^2 - \cos(18 \cdot x_i^2) \right)$$

Результаты тестирования

| n | точность | метод | время | значение |
|----|-----------|---------------|---------|-----------------------|
| 5 | 10^{-3} | Golden ratio | 1.36 ms | 1.542458003137213 |
| | 10^{-3} | Moore-Skelboe | 151 ms | 0.0007449817996270092 |
| 10 | 10^{-3} | Golden ratio | 5.18 ms | 3.0849160062744287 |
| | 10^{-3} | Moore-Skelboe | 636 ms | 0.0014899635992544624 |
| 20 | 10^{-3} | Golden ratio | 20.8 ms | 6.16983201254885 |
| | 10^{-3} | Moore-Skelboe | 1.94 s | 0.002979927198509369 |
| 30 | 10^{-3} | Golden ratio | 37 ms | 9.254748018823253 |
| | 10^{-3} | Moore-Skelboe | 4.54 s | 0.0044698907977642754 |
| 40 | 10^{-3} | Golden ratio | 54.2 ms | 12.339664025097658 |
| | 10^{-3} | Moore-Skelboe | 7.79 s | 0.005959854397019182 |
| 50 | 10^{-3} | Golden ratio | 79.7 ms | 15.424580031372061 |
| | 10^{-3} | Moore-Skelboe | 11.6 s | 0.0074498179962740885 |

Функция Розенброка

График функции при $n = 2$



Формула

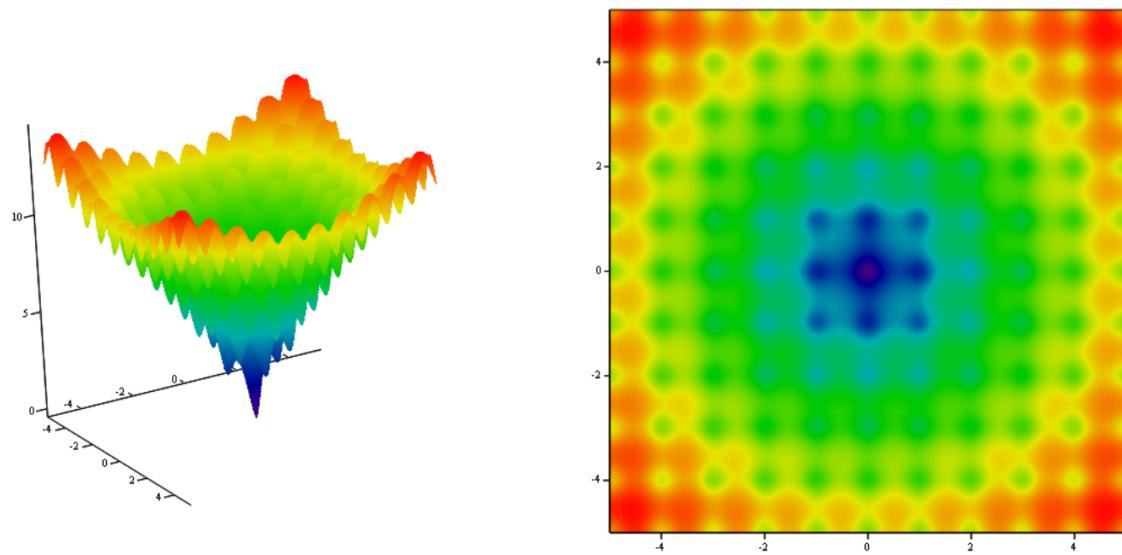
$$f(x) = \sum_{i=1}^{n-1} \left(100 (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right)$$

Результаты тестирования

| n | точность | метод | время | значение |
|---|-----------|---------------|---------|---------------------|
| 2 | 10^{-3} | Golden ratio | 1.12 ms | 0.16956582889643249 |
| | 10^{-3} | Moore-Skelboe | 155 ms | 0.16036252999621464 |
| 3 | 10^{-3} | Golden ratio | 5.95 ms | 0.259397310396006 |
| | 10^{-3} | Moore-Skelboe | 3.99 s | 0.11254106932659813 |
| 4 | 10^{-3} | Golden ratio | 17 ms | 0.28064340055794645 |
| | 10^{-3} | Moore-Skelboe | 8.75 s | 0.11734786681147619 |
| 5 | 10^{-3} | Golden ratio | 27.3 ms | 0.2830939134564683 |
| | 10^{-3} | Moore-Skelboe | 15.6 s | 0.11851270446807871 |
| 6 | 10^{-3} | Golden ratio | 40.4 ms | 0.29047947563212634 |
| | 10^{-3} | Moore-Skelboe | 24.5 s | 0.11879389607067276 |
| 7 | 10^{-3} | Golden ratio | 49.2 ms | 0.29064024190256904 |
| | 10^{-3} | Moore-Skelboe | 35.6 s | 0.1136410373197746 |
| 8 | 10^{-3} | Golden ratio | 18.5 ms | 0.33244082803407593 |
| | 10^{-3} | Moore-Skelboe | 41.7 s | 0.14015453688727936 |

Функция Экли

График функции при $n = 2$



Формула

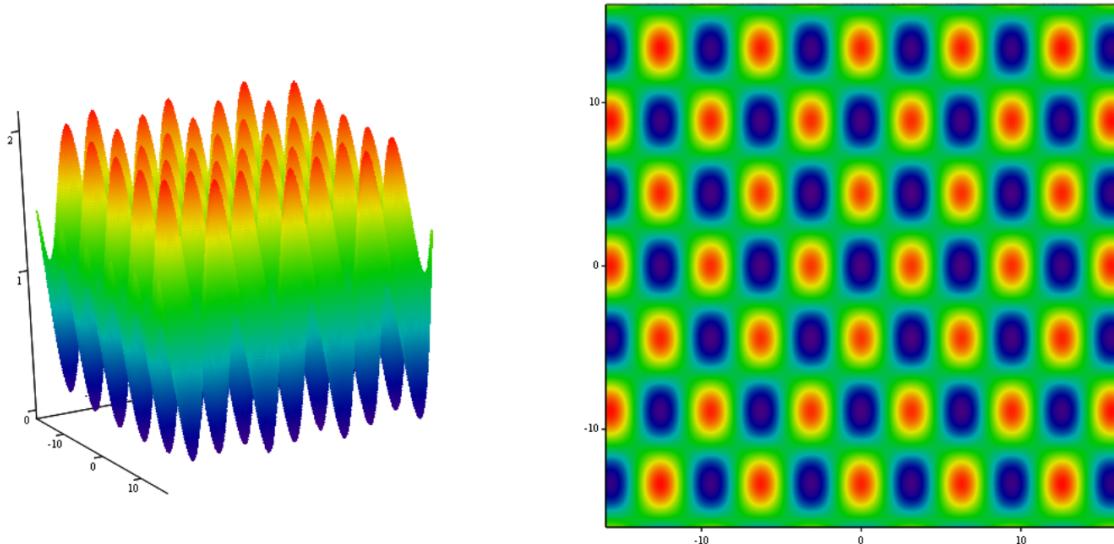
$$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi \cdot x_i)}$$

Результаты тестирования

| n | точность | метод | время | значение |
|----|-----------|---------------|---------|------------------------|
| 5 | 10^{-3} | Golden ratio | 2.41 ms | -4.440892098500626e-16 |
| | 10^{-3} | Moore-Skelboe | 553 ms | 0.0012256630393632229 |
| 20 | 10^{-3} | Golden ratio | 9.06 ms | 3.5744545080266543 |
| | 10^{-3} | Moore-Skelboe | 1.35 s | 0.0012256630393632229 |
| 20 | 10^{-3} | Golden ratio | 30.6 ms | 3.5744545080266543 |
| | 10^{-3} | Moore-Skelboe | 3.41 s | 0.0012256630393632229 |
| 30 | 10^{-3} | Golden ratio | 58.5 ms | 3.574454508026655 |
| | 10^{-3} | Moore-Skelboe | 6.23 s | 0.0012256630393645551 |
| 40 | 10^{-3} | Golden ratio | 84.1 ms | 3.574454508026654 |
| | 10^{-3} | Moore-Skelboe | 10.6 s | 0.0012256630393632229 |
| 50 | 10^{-3} | Golden ratio | 124 ms | 3.5744545080266525 |
| | 10^{-3} | Moore-Skelboe | 14.9 s | 0.0012256630393618906 |

Функция Грибанка

График функции при $n = 2$



Формула

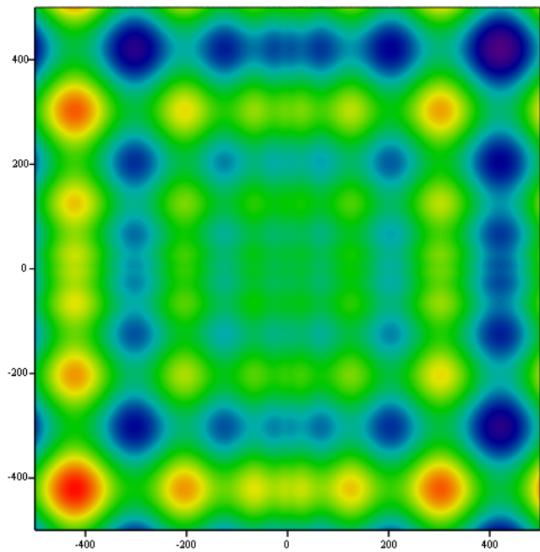
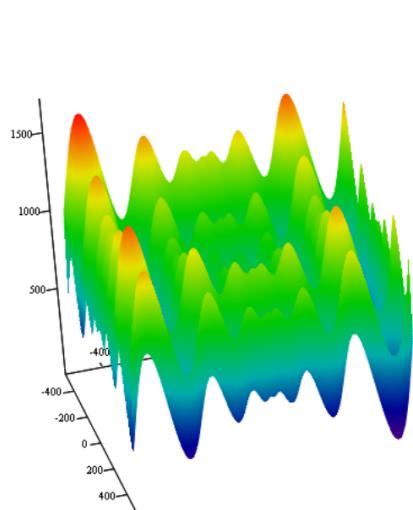
$$f(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Результаты тестирования

| n | точность | метод | время | значение |
|----|-----------|---------------|---------|------------------------|
| 5 | 10^{-3} | Golden ratio | 2.41 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 183 ms | 1.0644240466817223e-07 |
| 10 | 10^{-3} | Golden ratio | 6.05 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 739 ms | 1.3662353537391425e-07 |
| 20 | 10^{-3} | Golden ratio | 21.4 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 2.44 s | 1.6799845647952338e-07 |
| 30 | 10^{-3} | Golden ratio | 40.2 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 5.16 s | 1.867295605917363e-07 |
| 40 | 10^{-3} | Golden ratio | 59.3 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 9.09 s | 2.0016648982768004e-07 |
| 50 | 10^{-3} | Golden ratio | 84.4 ms | 0.009864698515380743 |
| | 10^{-3} | Moore-Skelboe | 14 s | 2.1067470723501458e-07 |

Функция Швефеля

График функции при $n = 2$



Формула

$$f(x) = 418.9829n - \sum_{i=1}^n \left(x_i \sin \left(\sqrt{|x_i|} \right) \right)$$

Результаты тестирования

| n | точность | метод | время | значение |
|---|-----------|---------------|-------------|------------------------|
| 2 | 10^{-3} | Golden ratio | 721 μ s | 236.8766946858181 |
| | 10^{-3} | Moore-Skelboe | 4.98 s | 2.5455285594944144e-05 |
| 3 | 10^{-3} | Golden ratio | 800 μ s | 355.3150420287272 |
| | 10^{-3} | Moore-Skelboe | 9.41 s | 3.8182928392416216e-05 |
| 4 | 10^{-3} | Golden ratio | 1.44 ms | 473.7533893716363 |
| | 10^{-3} | Moore-Skelboe | 15 s | 5.091057118988829e-05 |
| 5 | 10^{-3} | Golden ratio | 1.4 ms | 592.1917367145454 |
| | 10^{-3} | Moore-Skelboe | 22 s | 6.363821398736036e-05 |
| 6 | 10^{-3} | Golden ratio | 2.19 ms | 710.6300840574543 |
| | 10^{-3} | Moore-Skelboe | 31.1 s | 7.636585655745876e-05 |
| 7 | 10^{-3} | Golden ratio | 1.88 ms | 829.0684314003631 |
| | 10^{-3} | Moore-Skelboe | 40 s | 8.909349912755715e-05 |
| 8 | 10^{-3} | Golden ratio | 4.92 ms | 947.5067787432718 |
| | 10^{-3} | Moore-Skelboe | 52 s | 0.00010182114169765555 |

6 Примеры

6.1 Ссылки на статьи

Ссылки на статьи оформляются с помощью пакета `biblatex`, например [1]. В описании статье в `bib` файле нужно обязательно указывать место публикации работы (журнал или конференцию) и год. Обратите внимание, что для описания статей из разных источников в списке литературы используются разные команды в `bib` файле: статья из журнала [3], статья с конференции [1], книга [6], глава книги [5]. Если статья еще не опубликована нигде, а только выложена на arXiv, то на нее тоже можно сослаться [2], но предпочтительно ссылаться на опубликованную версию, если она уже существует. Если вы хотите сослаться на сайт, то можно либо так же внести его в список литературы [4] (рекомендуется, если таких ссылок у вас много из-за особенностей темы вашего проекта), либо использовать ссылку внизу страницы¹. При работе с онлайн ресурсами не забывайте указывать дату обращения к этому ресурсу, так как в отличие от опубликованных статей, эти ресурсы могут измениться в любой момент.

6.2 Рисунки

Все рисунки в тексте должны иметь подписи и вы на них должны ссылаться в тексте. Например, на Рисунке 6.1 изображен пример графика. Не забывайте подписывать все оси на графиках, добавлять легенду и пояснить все обозначения, а также используйте адекватного размера шрифты и толщину линий на графиках (все должно быть видно и понятно без многократного увеличения). На рисунке из примера явно не хватает обозначения синей линии в легенде.

6.3 Таблицы

Все таблицы в тексте тоже должны иметь подписи и вы на них должны ссылаться в тексте. Например, в Таблице 6.1 показаны результаты примерного эксперимента.

6.4 Формулы

Формулы стоит центрировать, а также нумеровать, если вы ссылаете на них в тексте. Также не забывайте пояснить все обозначения в формулах. Например, запишем следующую задачу

¹Книги доступны по ссылке: <http://www-cs-faculty.stanford.edu/~uno/abcde.html>, дата обр. 16.05.2013



Рис. 6.1: Пример графика. Тут должна быть подпись, поясняющая что происходит на рисунке (краткая, но достаточная для понимания основной идеи графика).

оптимизации:

$$\theta^* = \min_{\theta} F(\theta), \quad (1)$$

где F – квадратичная функция от параметра θ . При необходимости, далее в тексте можно сослаться на формулу (1). При этом, в зависимости от конкретных формул, можно использовать разные слова: формула, уравнение, задача оптимизации и т.п.

Таблица 6.1: Пример таблички. Тут должна быть подпись, поясняющая что происходит в таблице (краткая, но по делу).

| | Val | | | Test | | | nodes | subtokens |
|------------------|---------------|---------------|---------------|---------------|---------------|---------------|-------|-----------|
| | Prec | Rec | F1 | Prec | Rec | F1 | | |
| запуск 1 | 0.4894 | 0.3775 | 0.4263 | 0.4824 | 0.3683 | 0.4177 | 10029 | 179 |
| запуск 2 | 0.4887 | 0.3739 | 0.4237 | 0.4891 | 0.3724 | 0.4228 | 10039 | 177 |
| запуск 3 | 0.4820 | 0.3751 | 0.4219 | 0.4838 | 0.3677 | 0.4178 | 10037 | 180 |
| среднее | 0.4867 | 0.3755 | 0.4239 | 0.4851 | 0.3695 | 0.4195 | | |
| дисперсия | 0.0041 | 0.0019 | 0.0022 | 0.0036 | 0.0025 | 0.0029 | | |

Список литературы

- [1] Nadezhda Chirkova, Ekaterina Lobacheva и Dmitry Vetrov. “Bayesian Compression for Natural Language Processing”. B: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [2] Nadezhda Chirkova, Ekaterina Lobacheva и Dmitry Vetrov. “Bayesian Compression for Natural Language Processing”. B: *arXiv preprint, arXiv:1810.10927, version 2* (2018).
- [3] George D. Greenwade. “The Comprehensive Tex Archive Network (CTAN)”. B: *TUGBoat* 14.3 (1993), c. 342–351.
- [4] Donald Knuth. *Knuth: Computers and Typesetting*. URL: <http://www-cs-faculty.stanford.edu/~uno/abcde.html> (дата обр. 16.05.2013).
- [5] Donald E. Knuth. “Fundamental Algorithms”. B: Addison-Wesley, 1973. Гл. 1.2.
- [6] Donald E. Knuth. *The Art of Computer Programming*. Four volumes. Seven volumes planned. Addison-Wesley, 1968.