Continious optimization

Pavel Fakanov

February 2020

1 Задачи

1.1 Задача 3

Функция потерь выглядит следующим образом (в предположении, что класс объекта 0 или 1):

$$L(x) = -\frac{1}{m}\sum_{i=1}^{m} b_i log\left(\frac{1}{1+e^{-x^T a_i}}\right) + (1-b_i) log\left(1 - \frac{1}{1+e^{-x^T a_i}}\right) + \frac{\lambda}{2}\|x\|_2^2$$

Введем функцию

$$g(z) = \frac{1}{1+e^{-z}}$$

Тогда вероятность принадлежности i-го объекта к положительному классу можно расписать следующим образом:

$$h_x(a) = g(x^T a) = \frac{1}{1+e^{-x^T a}}$$

Функция $g(z)$ обладает следующими полезными свойствами, которые доказываются непосредственно:

$$g'(z) = g(z)(1-g(z)) \tag{1}$$
$$g(-z) = 1 - g(z) \tag{2}$$

Рассмотрм случай, когда у нас присутствует только один элемент выборки - со значениями признаков a и значением целевой перемнной b. Тогда вычислим $\frac{\partial}{\partial x_j} L(x)$ (пока без регуляризации):

$$\frac{\partial}{\partial x_j} L(x) = -\left(b\frac{1}{g(x^T a)} - (1-b)\left(\frac{1}{1-g(x^T a)}\right)\right)\frac{\partial}{\partial x_j} g(x^T a) =$$
$$= -\left(b\frac{1}{g(x^T a)} - (1-b)\left(\frac{1}{1-g(x^T a)}\right)\right) g(x^T a)(1-g(x^T a))\frac{\partial}{\partial x_j} x^T a =$$
$$= -b(1-g(x^T a) + (1-b)g(x^T a))a_j =$$
$$= (h_x(a) - b)a_j$$

В случае m объектов, производная будет выглядеть следующим образом (без регуляризации):

$$\frac{\partial}{\partial x_j} L(x) = \frac{1}{m}\sum_{i=1}^{m}(h_x(a_i) - b_i)a_{ij}$$

В матричном виде это переписыается следующим образом:

$$\nabla L(x) = \frac{1}{m} A^T (g(Ax) - b)$$

Если добавить регуляризацию:

$$\nabla L(x) = \frac{1}{m} A^T (g(Ax) - b) + \lambda x$$

Сама же функция логистической регрессии в матричном виде может быть записана в следующей форме:

$$L(x) = -\frac{1}{m}(1, 1, ..., 1) * log(-(2b - 1) \odot g(Ax)) + \frac{\lambda}{2}\|x\|_2^2$$

Здесь $(1, 1, ..., 1)$ - единичный вектор длины m

Вычислим гессиан для логистической регрессии:
Также, как и для градиента сначала рассмотрим случай с одним объектом без регуляризации, Пусть $a_i \in \mathbf{R}^d$ - признаковое описание $i-$го объекта, $b_i \in \mathbf{R}$ - класс $i-$го объекта. Также введем $z_i = x^T a_i$.

Если обозначить $L_i(x) = -b_i log(g(z_i) - (1 - b_i)(log(1 - g(z_i))$, то, как мы уже знаем по предыдущим выкладкам:

$$\nabla L_i(x) = \frac{\partial}{\partial x^T} L_i(x) = a_i(g(z_i) - b_i)$$

Вычислим $\frac{\partial g(z_i)}{x}$:

$$\frac{\partial g(z_i)}{x} = \frac{\partial g(z_i)}{z_i} \frac{\partial z_i}{x} = g(z_i)(1 - g(z_i))a_i$$

Тогда:

$$\nabla^2 L_i(x) = \frac{\partial L_i(x)}{\partial x \partial x^T} = \frac{\partial}{\partial x} a_i(g(z_i) - b_i) = a_i a_i^T g(z_i)(1 - g(z_i))$$

В случае m объектов, гессиан будет выглядеть следующим образом (без регуляризации):

$$\nabla^2 L(x) = \sum_{i=1}^{m} a_i a_i^T g(z_i)(1 - g(z_i))$$

Выпишем теперь данную формулу в матричном виде: заметим, что чтобы из матрицы A^T, которая состоит из векторов a_i, которые стоят по столбцам, получить матрицу, i-й столбец который домножен на $g(z_i)(1 - g(z_i))$, достаточно рассмотреть матрицу $A^T Z$, где Z - диагональная матрица со значением $g(z_i)(1 - g(z_i))$ в клетке (i, i).

Дополнительно заметим, что $\sum_{i=1}^{m} a_i a_i^T = A^T A$. Таким образом:

$$\nabla^2 L(x) = A^T Z A$$

С учетом регуляризации формула примет вид:

$$\nabla^2 L(x) = A^T Z A + \lambda E_d$$

2 Эксперименты

2.1 Траектория градиентного спуска на квадратичной функции

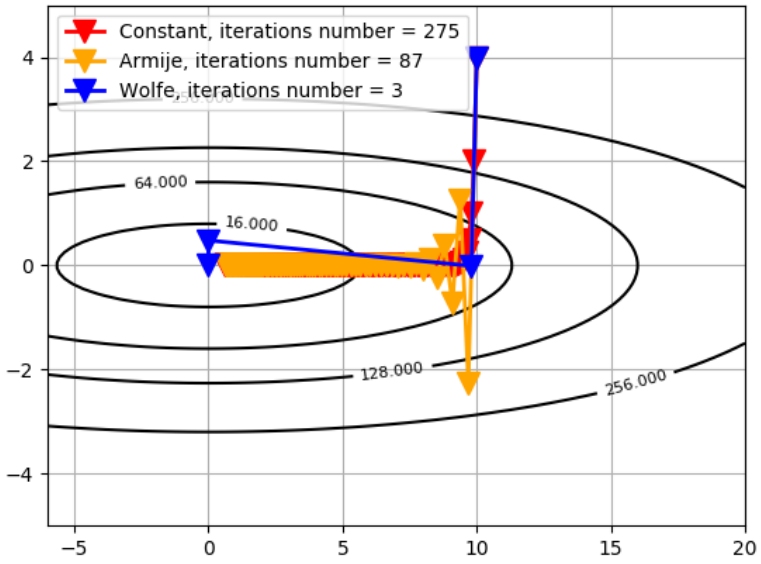


Рис. 1: Количество итераций градиентного спуска для различных стратегий выбора шага $\left(A = \begin{pmatrix} 1 & 0 \\ 0 & 50 \end{pmatrix}\right)$, $x_0 = (10, 4)$

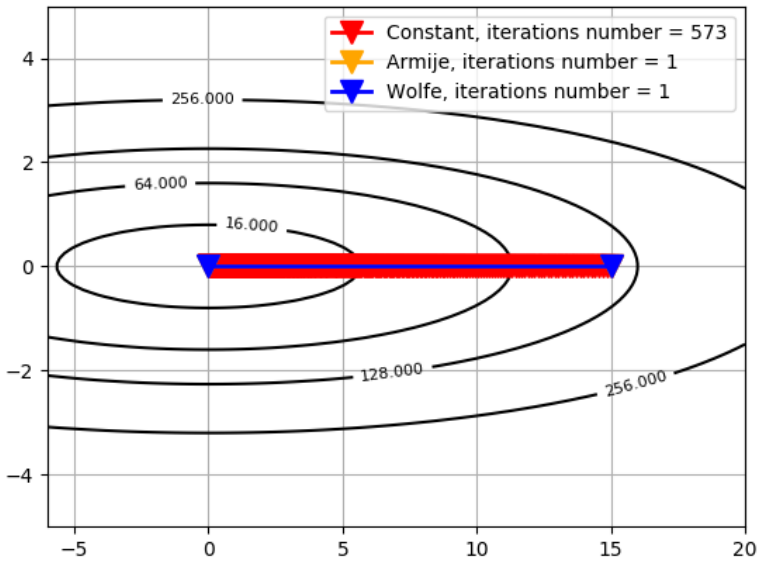


Рис. 2: Количество итераций градиентного спуска для различных стратегий выбора шага $\left(A = \begin{pmatrix} 1 & 0 \\ 0 & 50 \end{pmatrix}\right)$, $x_0 = (15, 0)$

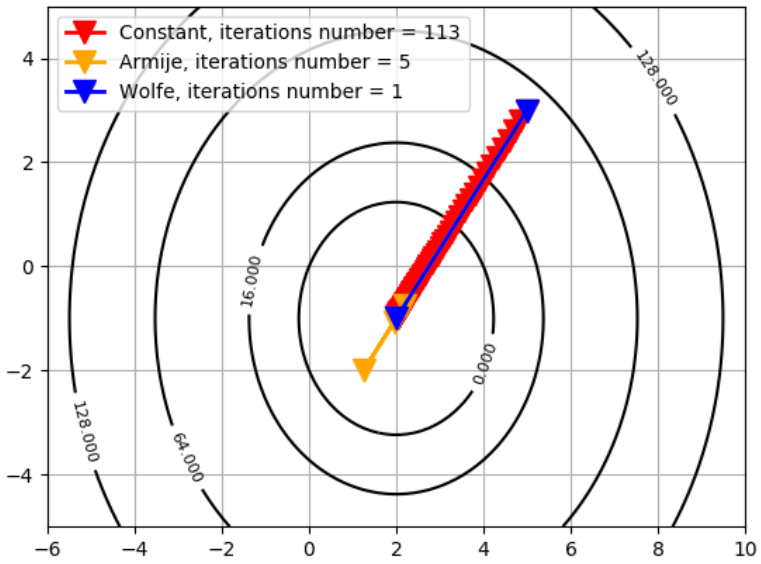


Рис. 3: Количество итераций градиентного спуска для различных стратегий выбора шага $\left(A = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\right)$, $x_0 = (5, 3)$

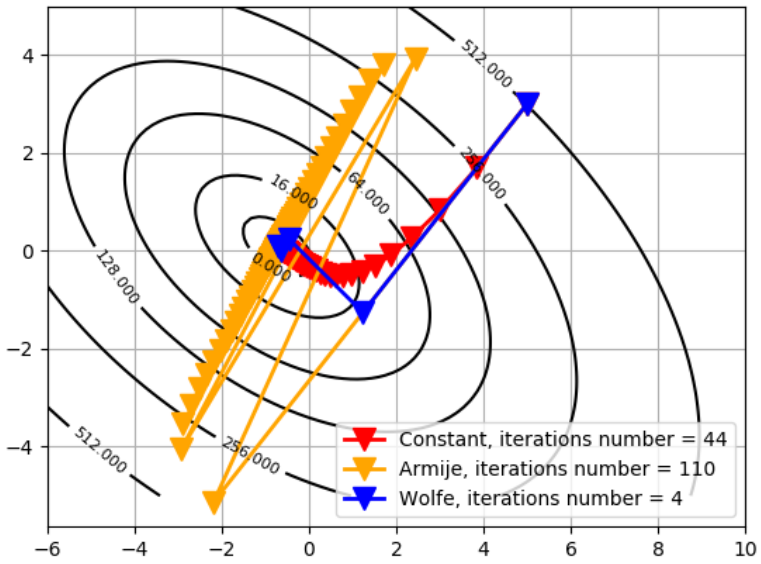


Рис. 4: Количество итераций градиентного спуска для различных стратегий выбора шага $\left(A = \begin{pmatrix} 15 & 10 \\ 10 & 25 \end{pmatrix}\right)$, $x_0 = (5, 3)$

Согласно проведенным экспериментам можно сделать следующие выводы:

1. В большинстве случаев гораздо предпочтительнее использовать метод Вульфа, так как он требует куда меньше итераций для сходимости

2. Выбор начальной точки сильно влияет на скорость сходимости, особенно для метода Армихо (рис.1 и рис.2)

3. При большем числе обусловленности матрицы, данным методам потребуется большее число итераций, (рис.2 и рис.3 для константного метода, рис.1 и рис.3 для других методов)

2.2 Зависимость числа итераций градиентного спуска от чис- ла обусловленности и размерности пространства

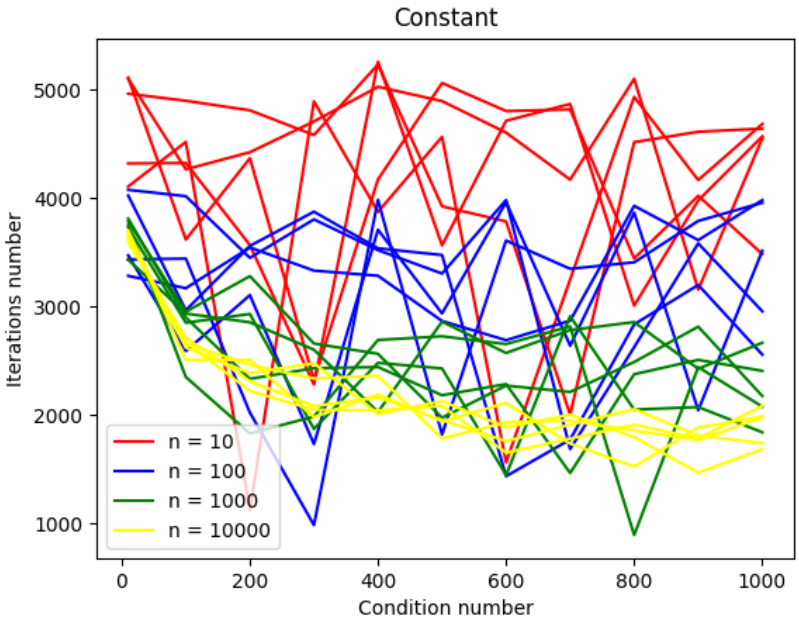


Рис. 5: Количество итераций градиентного спуска в зависимости от числа обусловленности и размерности пространства переменных для константного метода

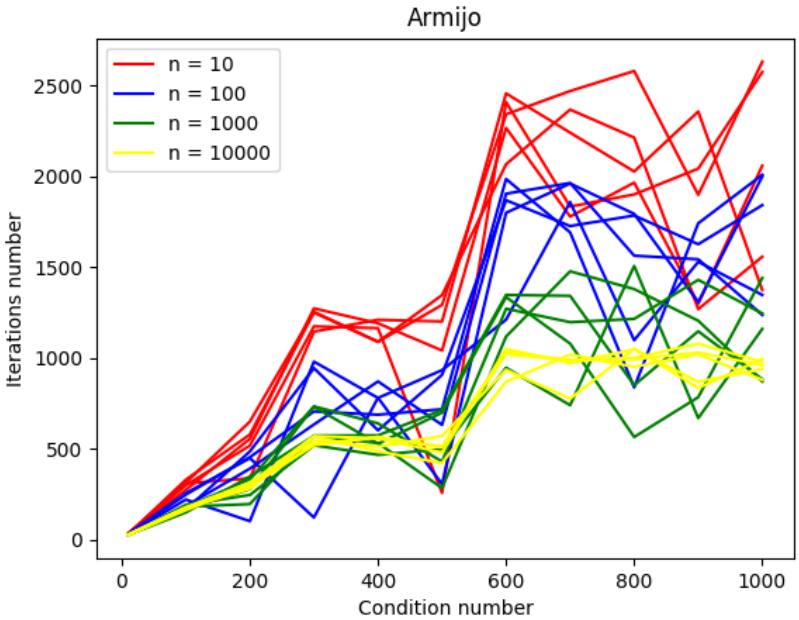


Рис. 6: Количество итераций градиентного спуска в зависимости от числа обусловленности и размерности пространства переменных для метода Армихо

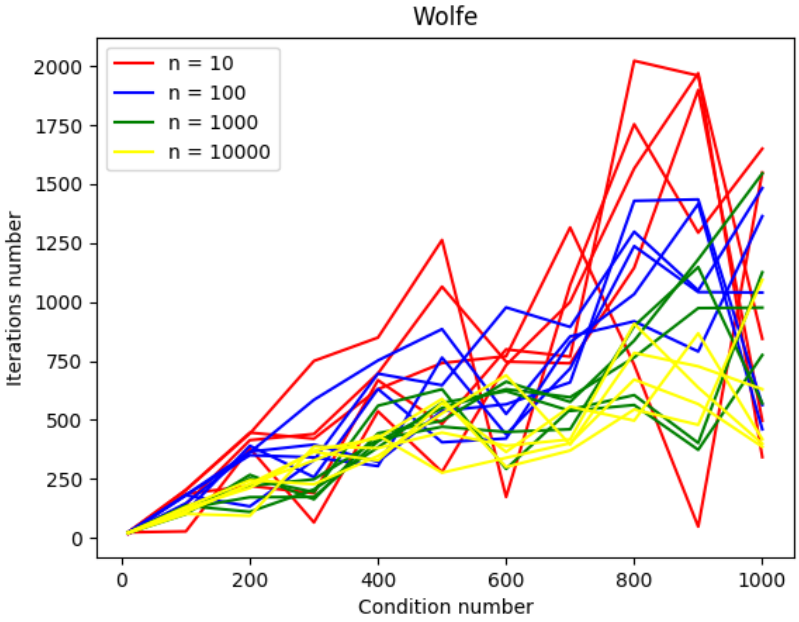
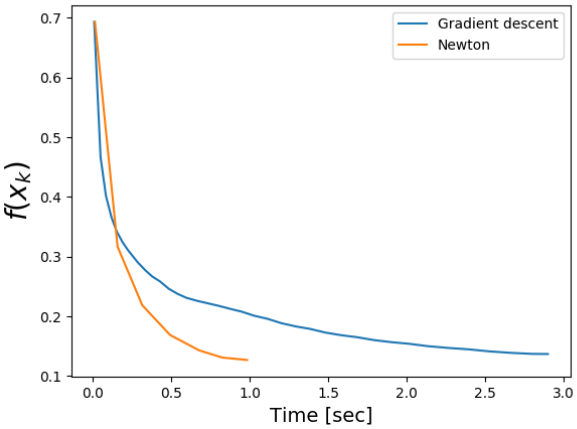


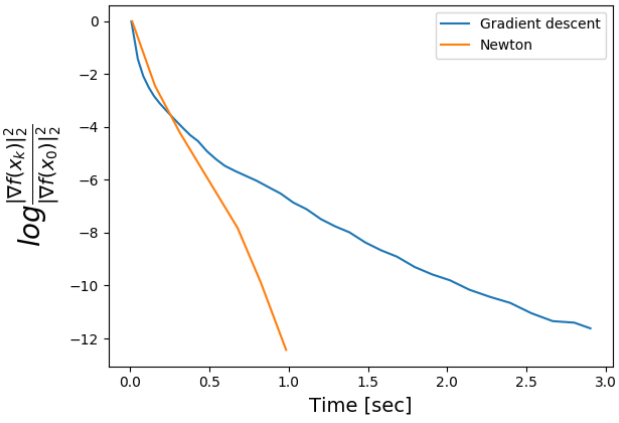
Рис. 7: Количество итераций градиентного спуска в зависимости от числа обусловленности и размерности пространства переменных для метода Вульфа

Согласно данным графикам мы можем заключить, что для методов Вульфа и Армихо с ростом числа обусловленности также растет число итераций, необходимых методу для сходимости, напротив, с ростом размерности пространства переменных число итераций уменьшается. Для константного же метода явной зависимости от числа обусловленности не наблюдается, в то время, как зависимость от размерности пространства переменных такая же, как для двух других методов

2.3 Сравнение методов градиентного спуска и Ньютона на реальной задаче логистической регрессии

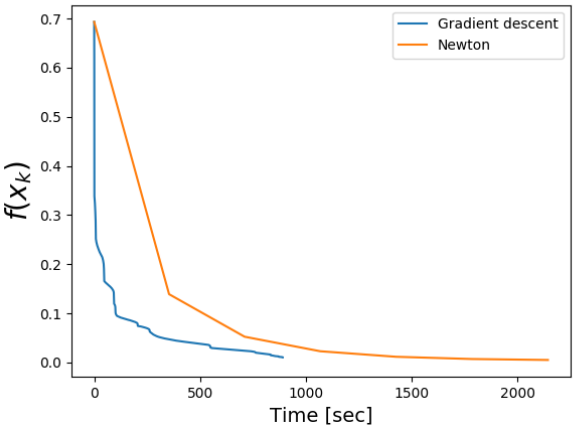


(a) Зависимость значения функции от реального времени работы метода

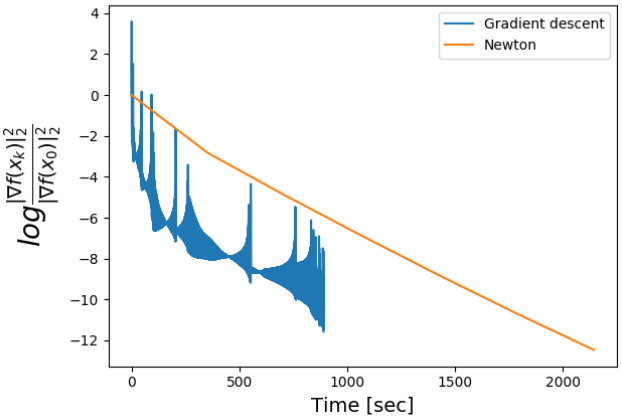


(b) Зависимость относительного квадрата нормы градиента против реального времени работы

Рис. 8: Датасет w8a

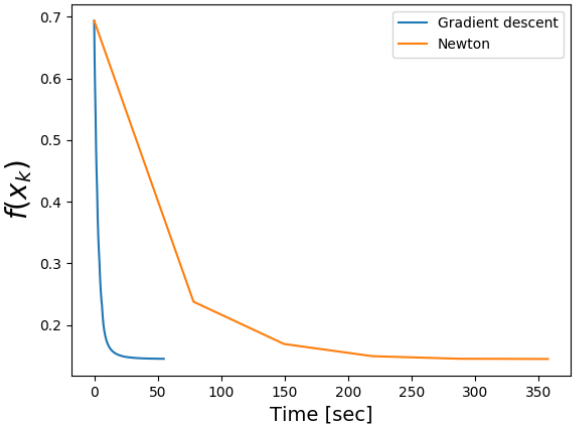


(a) Зависимость значения функции от реального времени работы метода

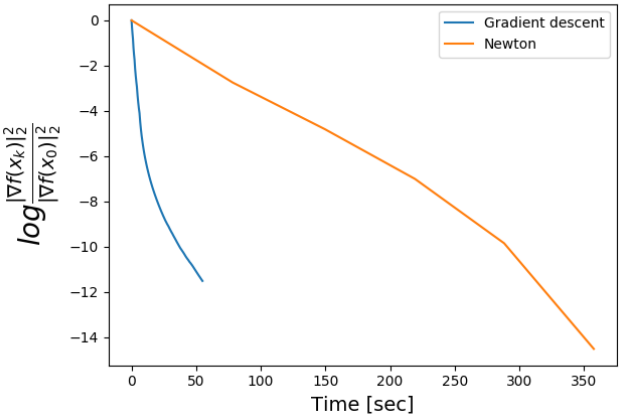


(b) Зависимость относительного квадрата нормы градиента против реального времени работы

Рис. 9: Датасет gissete



(a) Зависимость значения функции от реального времени работы метода



(b) Зависимость относительного квадрата нормы градиента против реального времени работы

Рис. 10: Датасет real-sim

Как выводилось на лекции данные методы оптимизации имеют следующие тсоимости итераций по времени и по памяти?

	Memory	Iteration time
Newton's method	$O(n^2)$	O(nq) + O(qk)
Gradient descent	O(n)	$O(n^2q) + O(qk) + O(n^3)$

Рис. 11: Стоимость итерации и использование памяти

2.4 Оптимизация вычислений в градиентном спуске

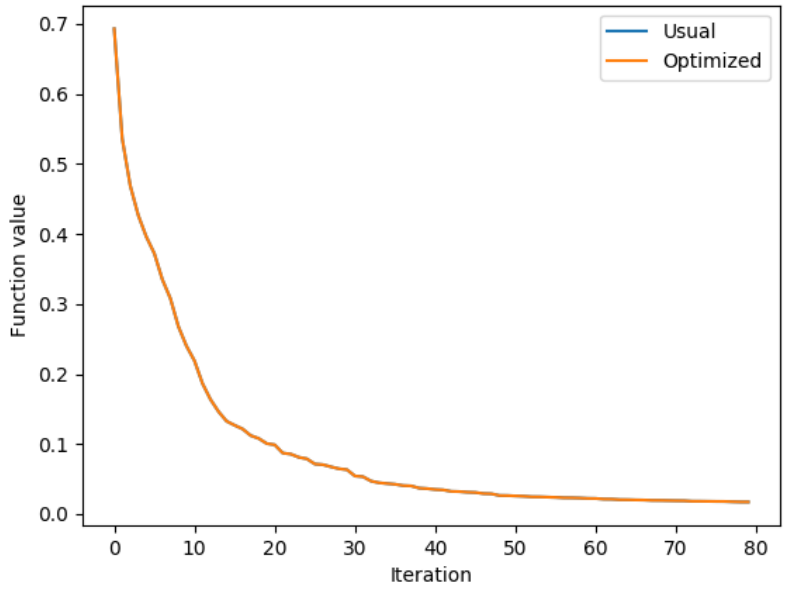


Рис. 12: Зависимость значения функции от номера итерации

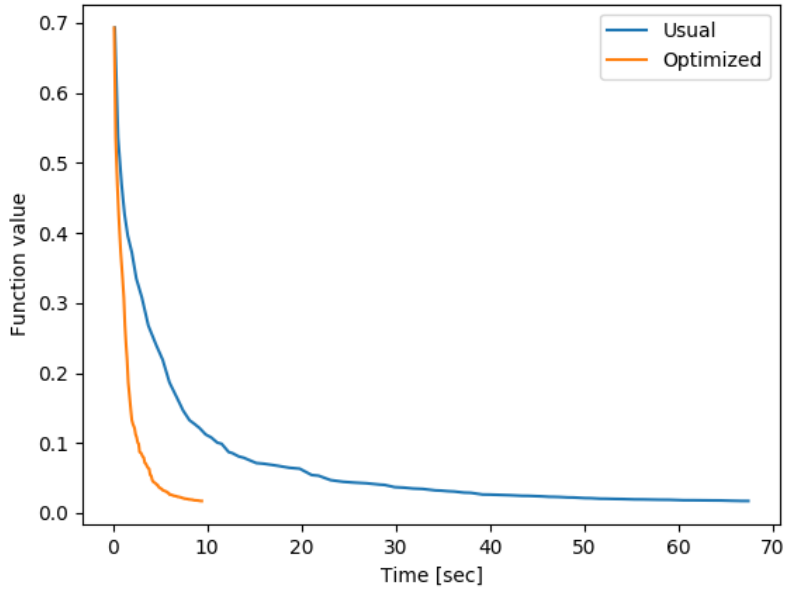


Рис. 13: Зависимость значения функции от реального времени работы метода

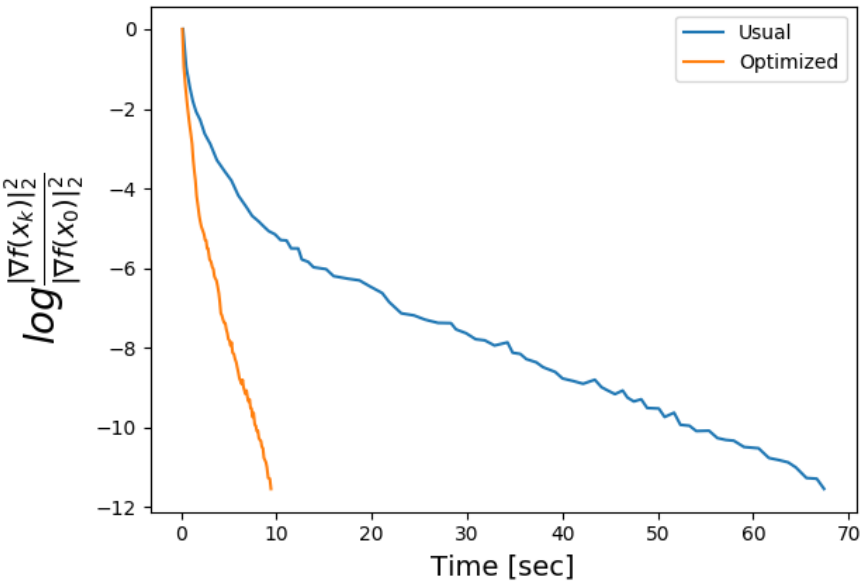


Рис. 14: Зависимость относительного квадрата нормы градиента против реального времени работы

Как мы видим, использование оптимизированного метода дает значительный прирост по времени работы алгоритма. Также можно заметить, что на первом графике кривые для обоих методов совпадают. Это объясняется тем, что итерации оптимизированного оракула ничем в вычислениях не отличаются от итераций обычного оракула, оптимизированный оракул лишь переиспользует ранее подсчитанные элементы.

2.4.1 Стратегия выбора длины шага в градиентном спуске

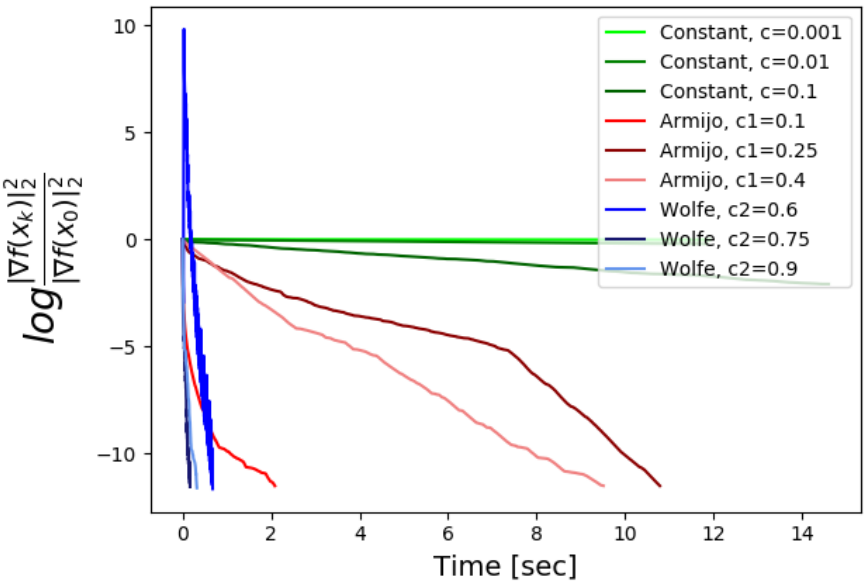


Рис. 15: Кривые сходимости для различных стратегий

Как мы уже видели, в предыдущих экспериментах, лучше всего работает метод Вульфа, в то время как константный метод показывает самые худшие результаты. Оптимальная константа для метода Вульфа - 0.75 согласно результатам эксперимента