Второе задание по методам оптимизации

Иван Ермаков

3

Логистическая регрессия

$$\min_{x\in\mathbb{R}^n}\left\{f(x)=\frac{1}{m}\left\langle\ln\left[1+\exp\left(-b\odot(Ax)\right)\right],1_m\right\rangle+\frac{\lambda}{2}\|x\|^2\right\} \tag{1}$$

Здесь b — вектор из b_i, A^T – матрица $m\times n$, в которой строки являются векторами a_i.

$$\begin{aligned}\mathrm{D}\,f(x)&=\lambda x-\frac{A^T}{m}\left(b\odot\frac{\exp\left(-b\odot(Ax)\right)}{1+\exp\left(-b\odot(Ax)\right)}\right)=\lambda x-\frac{A^T}{m}\left(b\odot u\left[-b\odot(Ax)\right]\right)\\ u(y)&=\frac{e^y}{1+e^y}=\frac{1}{1+e^{-y}}\end{aligned} \tag{2}$$

Здесь использовано много свойств произведения Адамара.

$$\frac{\mathrm{d}u}{\mathrm{d}y}=\frac{e^y(1+e^y)-e^{2y}}{(1+e^y)^2}=\frac{e^y}{(1+e^y)^2} \tag{3}$$

$$\mathrm{D}^2\,f[\mathrm{d}x]=\lambda\mathrm{d}x+\frac{A^T}{m}\left(b\odot\frac{e^y}{(1+e^y)^2}\odot b\odot(A\mathrm{d}x)\right) \tag{4}$$

$$\mathrm{D}^2\,f=\lambda+\frac{A^T}{m}\left\{\left[\left(b\odot b\odot\frac{e^y}{(1+e^y)^2}[-b\odot(Ax)]\right)1_n^T\right]\odot A\right\} \tag{5}$$

Потратив еще *немного* времени, можно понять, что это выражение можно переписать как

$$\mathrm{D}^2\,f=\lambda+\frac{1}{m}\cdot A^T\,\mathrm{diag}\left\{b\odot b\odot v(-b\odot(Ax))\right\}A \tag{6}$$

где

$$v(y)=\frac{e^y}{(1+e^y)^2} \tag{7}$$

1 Эксперимент: траектория градиентного спуска

Построение графиков выполнено при помощи скрипта “grad_trajectory.py”. Для хорошо обусловленной матрицы поиск останавливается за 109000 шагов, а для плохо обусловленной за 69000. Скорее всего так происходит из-за того, что в хорошо обсусловленной матрице метод постоянно проскакивает оптимальную точку из-за высокой требуемой точности.

Теперь все встало на свои места: для хорошо обусловленной матрицы метод сходится всего за 18 шагов. Для плохо обусловленной требуется 327 шагов. Видно, что в первом случае довольно быстро тракетория становится прямой, а во втором случае она все время колеблется вокруг оптимального пути и совершает больше шагов, чем нужно.

Для условия Вольфа ситуация аналогичная первому случаю: для хорошо обусловленной матрицы метод работает “слишком хорошо” и проскакивает точку экстремума. Метод сходится за 18 шагов. Для плохо обусловленной матрицы требуется всего 4 шага. Понятно, что в общем случае для хорошо обусловленой матрицы метод Вольфа тоже работает хорошо, и некоторое $O(1)$ количество операций в конце, связанное с проскакиванием, роли не играет.

При этом, конечно, чем ближе прямая градиента к оптимальной точке, тем быстрее будут работать все методы, потому что траектория будет ближе к прямой.

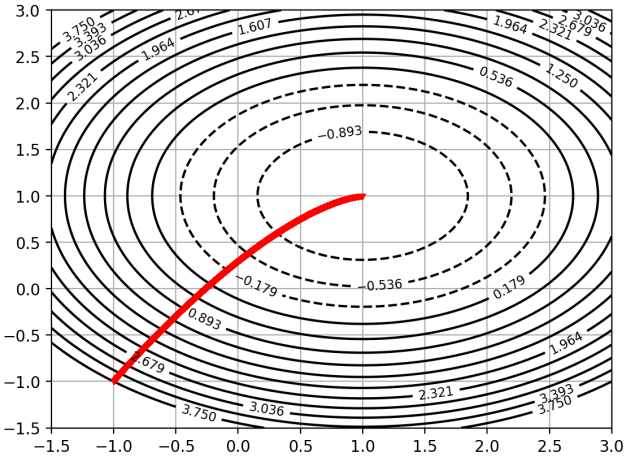

Рис. 1: Постоянный шаг, хорошо обусловленная матрица.

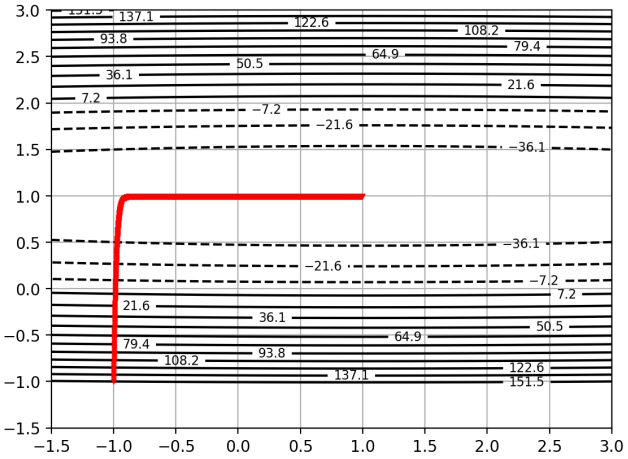

Рис. 2: Постоянный шаг, плохо обусловленная матрица.

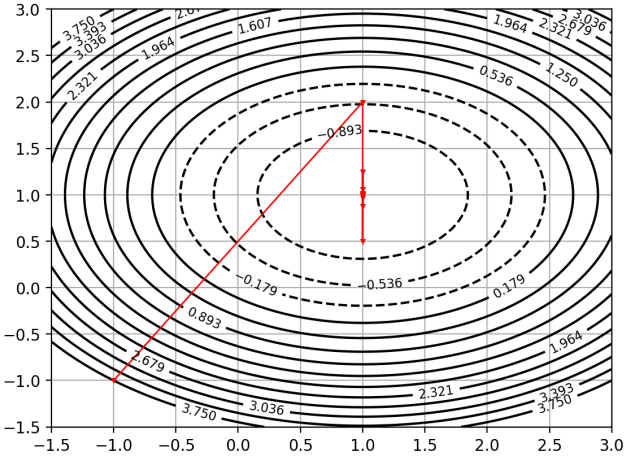

Рис. 3: Условие Армихо, хорошо обусловленная матрица.

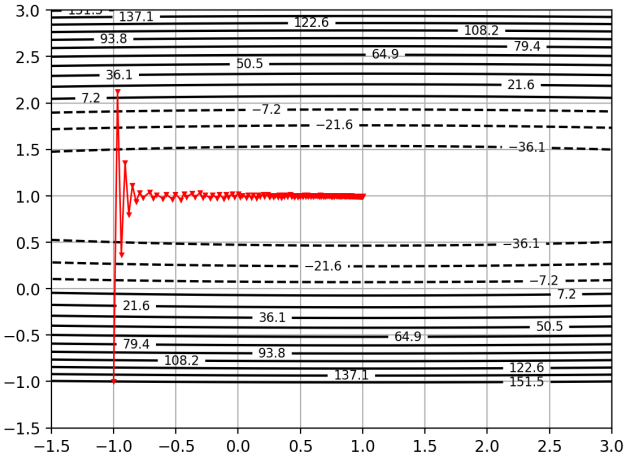

Рис. 4: Условие Армихо, плохо обусловленная матрица.

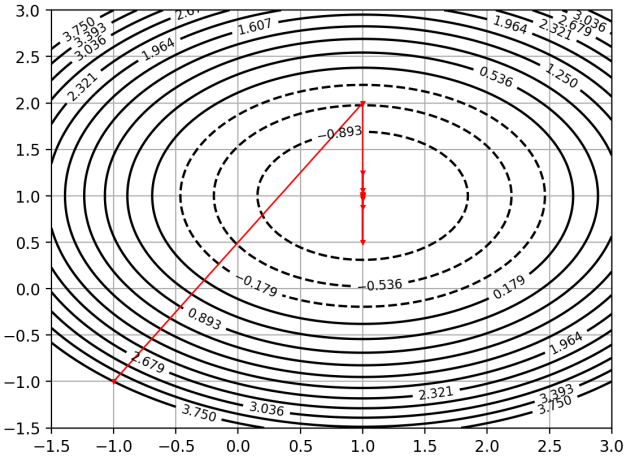

Рис. 5: Условие Вольфа, хорошо обусловленная матрица.

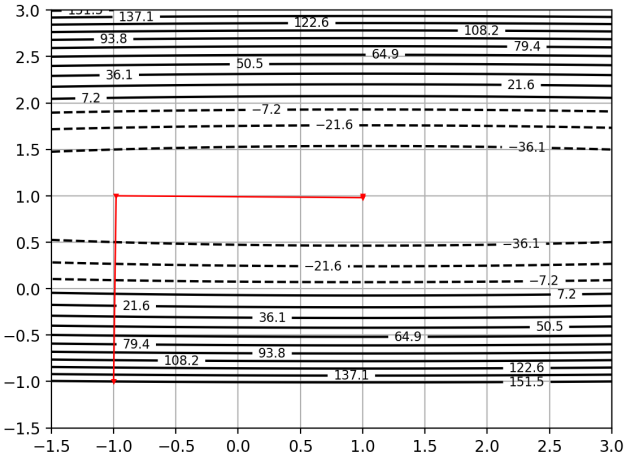

Рис. 6: Условие Вольфа, плохо обусловленная матрица.

Эксперимент: зависимость числа итераций итераций градиентного спуска от числа обусловленности и размерности пространства

Выборка генерируется при помощи скрипта “perf_plots.py” так, как ьыло рекомендовано в задании: генерируется диагональная матрица, далее у нее фиксируется число обусловленности. Потом случайным образом выбираются x_0 и b. Получившиеся семейства кривых изображены на рисунке 7

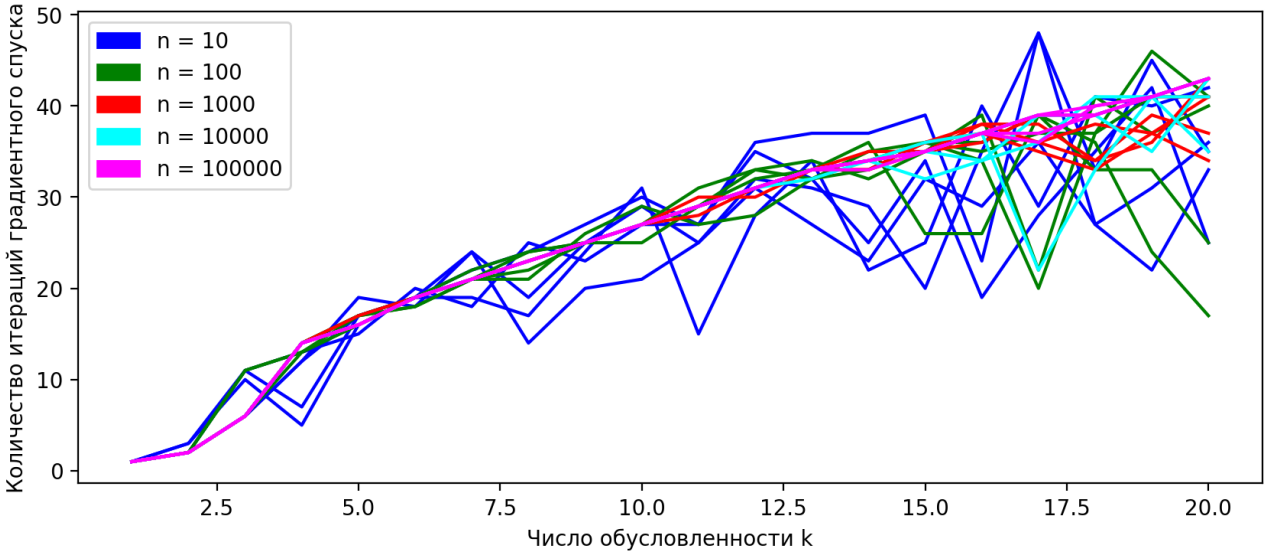

Рис. 7: Зависимость числа итераций от числа обусловленности и размерности пространства

Видно, что от размерности пространства количество итераций не зависит. А вот с ростом числа обусловленности количество итераций растет. Причем рост, хотя это и сложно определить, похож на линейный. Линейный рост можно объяснить так:

$$f(x_k) - f^* \leq \frac{L}{2}\left(\frac{\kappa - 1}{\kappa + 1}\right)^{2k} \|x_0 - x^*\|^2 \tag{8}$$

Отсюда

$$k = \frac{\ln \alpha}{2 \ln\left(\frac{\kappa - 1}{\kappa + 1}\right)} \approx -\kappa \ln \alpha \tag{9}$$

из Тейлоровского разложение при $\kappa \gg 1$.

Эксперимент: Сравнение методов градиентного спуска и Ньютона на реальной задаче логистической регрессии

На одну итерацию градиентному спуску требуется $O(n)$ памяти, поскольку хранятся только векторы x и ∇f

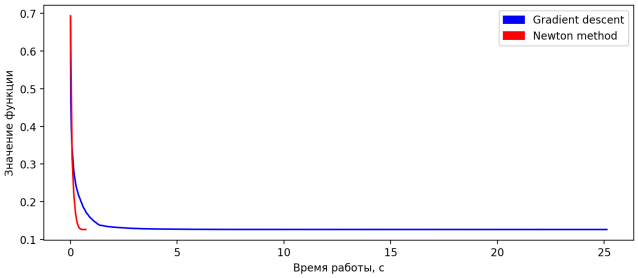

Рис. 8: Датасет w8a, значения функции

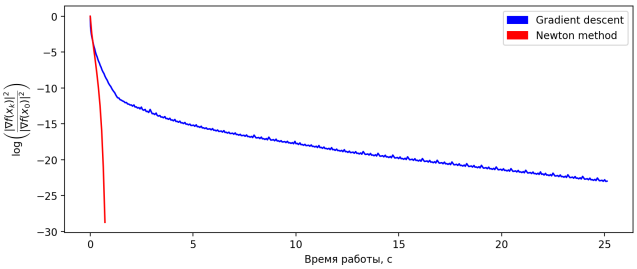

Рис. 9: Датасет w8a, значения градиента

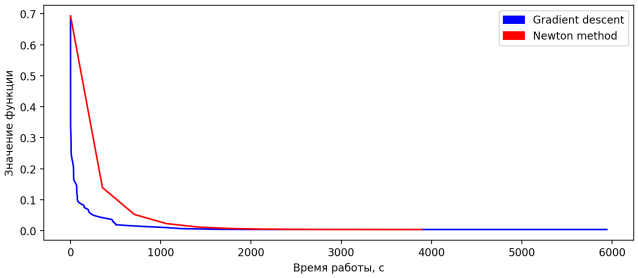

Рис. 10: Датасет gisette, значения функции

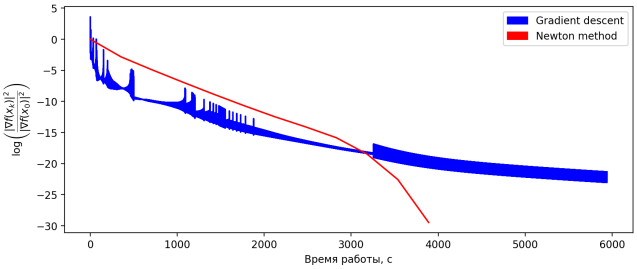

Рис. 11: Датасет gisette, значения градиента

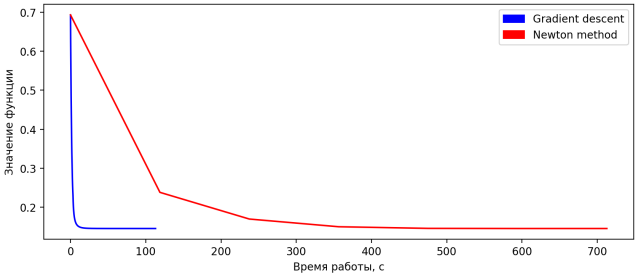

Рис. 12: Датасет real-sim, значения функции

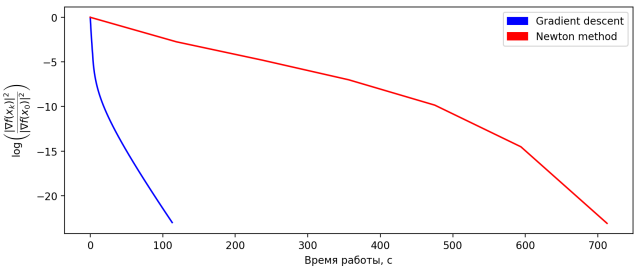

Рис. 13: Датасет real-sim, значения градиента