

Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации

Отчёт по лабораторной работе №2

Выполнили:
Ночевкина Наталья, М3233
Чуракова Александра, М3232
Ильченко Артём, М3233

Санкт-Петербург
2023

Цели работы

1. Исследовать стохастический градиентный спуск и его сходимость с разным размером батча.
2. Исследовать функцию изменения шага для улучшения сходимости.
3. Оценить влияние различных подходов lr-scheduling, а также модификаций:
 - Nesterov
 - Momentum
 - AdaGrad
 - RMSProp
 - Adamна эффективность реализованных методов.
4. Исследовать сходимость алгоритмов и сравнить методы по:
 - скорости сходимости
 - надежности
 - требуемым машинным ресурсам
5. Исследовать полиномиальную регрессию:
 - стандартную
 - с регуляризациями
6. Исследовать влияние регуляризации на восстановление регрессии.
7. Исследовать методы пакетного и стохастического градиентного спусков для решения задачи линейной регрессии
8. Реализовать полиномиальную регрессию и исследовать влияние регуляризации на эффективность восстановления.

Задачи

1. Реализовать стохастический градиентный спуск для решения линейной регрессии.
2. Подобрать функцию изменения шага для улучшения сходимости.
3. Реализовать модификации градиентного спуска (перечислены в целях).
4. Исследовать сходимость и сравнить методы по критериям (перечислены в целях).
5. Построить траекторию спуска различных алгоритмов из одной и той же исходной точки с одинаковой точностью.

Задание 1. Стохастический градиентный спуск.

Постановка задачи: необходимо решить задачу линейной регрессии с использованием стохастического градиентного спуска. Требуется исследовать сходимость алгоритма с разными размерами батча: 1 - SGD, 2, .., n – 1 - Minibatch GD, n - градиентный спуск с постоянным шагом.

Описание методов:

Задача линейной регрессии заключается в восстановлении линейной зависимости вида $y = ax + b + \varepsilon$ между переменными по заданному набору из n точек на плоскости с координатами $(x_i; y_i)$, $i \in [0; n - 1]$, где ε - ошибка модели.

Обозначим y за $f(x)$, а коэффициенты полинома, задающего f за вектор $\{\theta_i\}_{i=0}^d$, где d – степень $f(x)$, в данном случае, т.к. решается задача линейной регрессии $d=1$, а $\theta.length = 2$. Итак, задача регрессии сводится к поиску такого вектора

коэффициентов θ : $\sum_{i=0}^n (y_i - f(x_i, \theta))^2 \rightarrow \min$, т.е. поиска вектора коэффициентов f

такого, что $f(x, \theta)$ наиболее точно описывает зависимость заданного набора точек. Данная задача решается с помощью метода наименьших квадратов.

Стохастический градиентный спуск (SGD) - это оптимизационный алгоритм, который используется для минимизации функционала ошибки. Он заключается в последовательном обновлении вектора весов w на каждой итерации, используя градиент функционала ошибки на текущем шаге и случайно выбранную точку данных.

Пусть имеется набор признаков $\{f_i\}_{i=0}^{n-1}$: $f_i: X \rightarrow Y$, в нашем случае линейной задачи f_i можно представить в качестве точки на плоскости, задаваемой парой координат (x_i, y_i) . Мы хотим решить задачу линейной регрессии и найти вектор коэффициентов θ : $\sum_{i=0}^n (y_i - f(x_i, \theta))^2 \rightarrow \min$. Для решения данной задачи можно применить метод стандартного градиентного спуска, поиск k -го коэффициента в котором можно описать следующим образом:

$$\theta_k = \theta_k + \alpha \sum_{i=0}^{n-1} \nabla(y_i - f(x_i, \theta_k)),$$

где $f(x, \theta) = \theta_1 x + \theta_0$; α - шаг градиентного спуска. Отметим, что для корректности определения коэффициентов тут следует обновлять значения каждого параметра одновременно, то есть:

```
while(Derror > eps) {
    for(i = 0; i < n; i++) {
        for(k = 0; k < deg(f); k++) { // Batch.length = batch_size
            theta_k = theta_k + alpha * grad(y_i - f(x_i, theta_k));
        }
    }
}
```

Однако, у данного метода есть недостаток: для определения значения коэффициентов необходимо использовать все значения элементов из набора. Stochastic Gradient Descent может быть более оптимальен при решении данной задачи, т.к., в отличие от обычного градиентного спуска, последовательно проходящего по всем элементам данных, в нем случайным образом выбирается индекс $j \in [0; n - 1]$ и обновление значения θ_k происходит от координат точки (x_j, y_j) .

Условием останова в данном случае будет достижение определенного значения изменения функции ошибки, обозначаемого eps . Тогда алгоритм вычисления вектора θ можно представить следующим образом:

```
while(Δerror > eps) {
    j = rand(0; n - 1);
    for(k = 0; k ≤ deg(f); k++) {
        θ_k = θ_k + α∇(y_j - f(x_j, θ_k));
    }
}
```

Подробнее с реализацией можно ознакомиться по [ссылке](#).

Также существует так называемый пакетный градиентный спуск, вычисляющий значение коэффициентов θ , не выбирая случайный элемент набора, а вычисляя его значение, пользуясь сразу всеми элементами выборки. Его разновидностью является MiniBatch GD, вычисляющий значение θ , основываясь на фиксированном числе элементов выборки, называемом $batch_size \in [1; n]$; собственно, при $batch_size = n$ получим обычный метод градиентного спуска. Его принцип работы следующий:

```
while(Δerror > eps) {
    for((x_i, y_i) ∈ Batch) { // Batch.length = batch_size
        for(k = 0; k ≤ deg(f); k++) {
            θ_k = θ_k + α * ∇(y_i - f(x_i, θ_k));
        }
    }
}
```

Реализация приведена по [ссылке](#).

Minibatch GD - это модификация Stochastic GD, которая позволяет использовать несколько точек данных для вычисления градиента функционала ошибки. То есть, если стохастический градиентный спуск использует только один элемент из набора данных для вычисления параметра на следующем шаге, то Minibatch GD вычисляет его, основываясь на нескольких значениях.

Minibatch стремится сбалансировать устойчивость стохастического градиентного спуска и эффективность пакетного градиентного спуска. Таким образом, при достаточно сбалансированном размере пакета Minibatch GD должен в теории показывать себя лучше, чем вышеперечисленные методы градиентного спуска.

Промежуточные выводы:

Исследование показало, что стохастический градиентный спуск сходится быстрее, чем градиентный спуск с постоянным шагом. Это было ожидаемо, поскольку

стохастический градиентный метод, Minibatch GD и пакетный градиентный спуск вычисляют одни и те же градиенты, но для разного количества параметров. На графиках ниже показана зависимость времени работы от размера батча (слева) и зависимость количества эпох (iterations) от размера батча (справа).

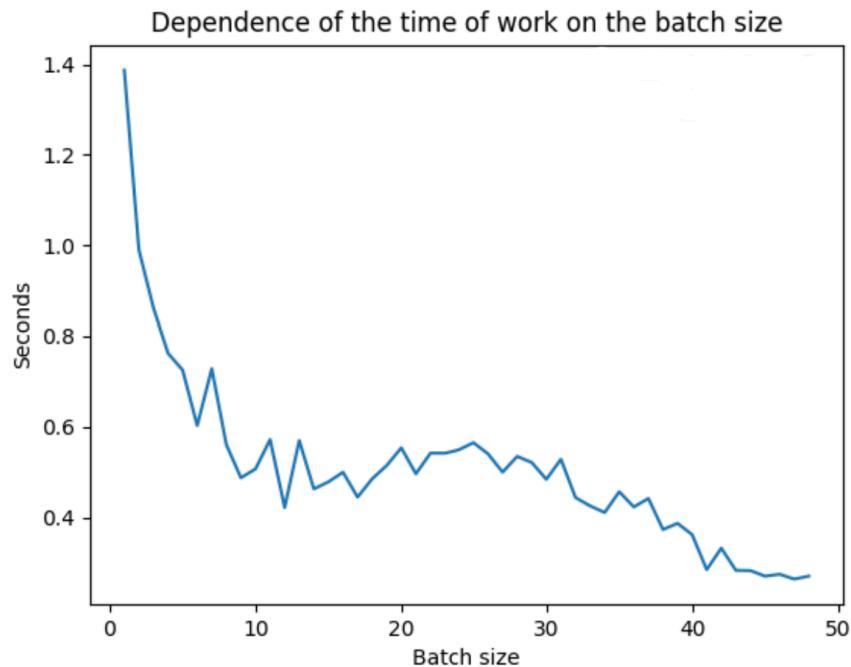


Рис. №1: Результаты исследования времени работы метода в зависимости от размера батча

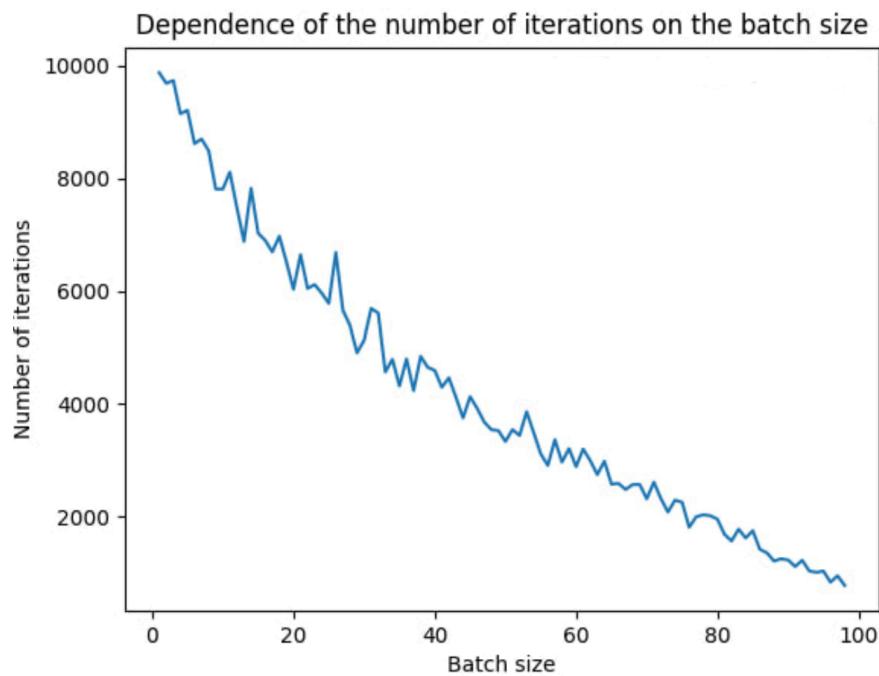


Рис. №2: Результаты исследования числа итераций, затрачиваемых методом в зависимости от размера батча

Задание 2. Функция изменения шага (learning rate scheduling).

Постановка задачи:

В ходе лабораторной работы нам необходимо было подобрать функцию изменения шага (learning rate scheduling) для улучшения сходимости модели.

Описание методов:

Для решения данной задачи мы использовали четыре метода изменения шага: const (постоянный), time (временной), step (ступенчатый) и exp (экспоненциальный).

Постоянный метод изменения шага (constant learning rate) является наиболее простым в реализации. Шаг остаётся неизменным для всех итераций градиентного спуска, но для более точных результатов необходимо менять и подбирать его вручную. Также существует опасность расхождения метода.

Основанный на времени график обучения изменяет темп обучения в зависимости от темпа обучения на предыдущей временной итерации. С учетом затухания математическая формула для темпа обучения выглядит следующим образом:

$$\eta_{n+1} = \frac{\eta_n}{1 + dn}$$

где η – это темп обучения, d – параметр затухания и n – шаг итерации.

Основанный на шаге график обучения изменяет темп обучения в соответствии с предопределенными шагами. Формула с применением затухания определяется следующим образом:

$$\eta_n = \eta_0 d^{\left[\frac{1+n}{r} \right]}$$

где η_n – это темп обучения в итерации n , η_0 – изначальный темп обучения, d соответствует величине изменения темпа обучения на каждом шаге (0.5 соответствует делению пополам) и r соответствует частоте сброса или тому, как часто следует сбрасывать темп (10 соответствует сбросу каждые 10 шагов).

Экспоненциальный график обучения похож на основанный на шаге, но вместо ступенчатых шагов, используется убывающая экспоненциальная функция. Математическая формула для использования в затухании следующая:

$$\eta_n = \eta_0 e^{-dn}$$

где d – параметр затухания.

Промежуточные выводы:

В результате экспериментов мы смогли выявить, что почти всегда лучшим с точки зрения времени работы является ступенчатый метод обучения. С другой стороны, при достаточно малых размерах батча лидером является экспоненциальный, а в случае обычного градиентного спуска (batch_size = 100%) лучше всего работают как экспоненциальный метод, так и метод, основанный на времени.



Рис. №3: Результаты исследования зависимости времени работы от размера батча для разных темпов обучения.

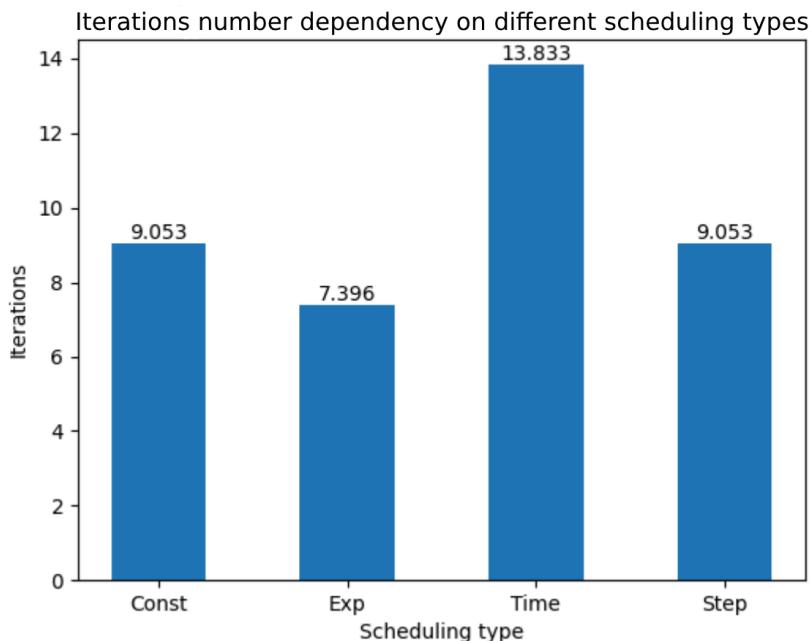


Рис. №4: Результаты исследования числа итераций, затрачиваемых методом в зависимости от функции изменения шага

Задания 3-4. Исследуйте сходимость алгоритмов. Сравнить различные методы по скорости сходимости, надежности, требуемым машинным ресурсам (объем оперативной памяти, количеству арифметических операций, времени выполнения)

1. AdaGrad

$$G_t = G_t + g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + e}} g_t$$

В данном методе на каждом шаге накапливается сумма квадратов обновлений параметров, за счет чего постепенно замедляется скорость спуска и метод хорошо показывает себя как на плохо-, так и на хорошо обусловленных функциях при высокой скорости обучения. Однако недостатком данного метода является то, что при выборе стартовой точки на достаточно большом расстоянии от точки минимума метод перестает сходиться в силу того, что накопленный квадрат обновления в корне знаменателя становится слишком большим, и на каждом следующем шаге изменение параметра θ становится все более незначительным. Также метод плохо сходится или же не сходится совсем при выборе низкой скорости обучения

// не знаю, стоит ли прямо сюда пихать большую табличку с данными или оставить ссылку на гит (вот резы

https://docs.google.com/spreadsheets/d/14_-1hctI4ZylfbE7LUhGdAM107oDsLcLV_yoWQeOo/edit?usp=sharing)

2. Adam

В этом методе сливаются идея накопления импульса, используемая в Nesterov и Momentum, и идея замедления обновлений часто изменяющихся значений, использующая, как RMSProp, среднее квадратов градиентов на предыдущих итерациях.

$$m_t = m_{t-1} * \beta_1 + (1 - \beta_1) \nabla f(\theta)$$

$$v_t = v_{t-1} * \beta_2 + (1 - \beta_2) \nabla^2 f(\theta)$$

$$\theta_t = \theta_{t-1} - \frac{lr}{\sqrt{\frac{v_t}{1-\beta_2^t} + e}} * \frac{m_t}{1-\beta_1^t}$$

Параметр v_t является вычисленным значением средней нецентрированной дисперсии, хранящей информацию о частоте изменения градиента, а m_t - накопленным импульсом. Недостатком данного алгоритма можно назвать его чувствительность к задаваемым в начале параметрам: β_1 , β_2 , lr . При их неправильном выборе метод начинает медленно работать, затрачивая в несколько раз больше итераций на нахождение минимума, или же перестает сходиться вовсе.

3. Momentum

В данном методе реализована идея накопления импульса, т.е. выбора направления спуска на текущей итерации на основании информации о значениях градиента на предыдущих шагах.

$$v_t = m * v_{t-1} + lr * \nabla f(\theta)$$

$$\theta_t = \theta_{t-1} + v_t$$

Так, параметр изменяется, учитывая не только вычисленное значение в текущей точке, но и экспоненциальное скользящее среднее градиентов в точках на предыдущих итерациях; стоит отметить, что вычисление скользящего среднего позволяет не вычислять среднее значение напрямую, тратя $O(n)$ операций на каждом шаге и столько же памяти на хранение информации о значениях градиентов, а сохраняет нативно информацию о них в переменной v_t , на обновление которой тратится лишь $O(1)$, операций и памяти.

Стоит отметить, согласно результатам проведенных экспериментов, что на плохообусловленных функциях метод чувствителен к выбору параметров, как например, скорость обучения(даже с применением learning rate scheduling), eps, величина параметра m .

4. Nesterov

Метод, как и Momentum, использует идею накопления импульса, что позволяет, пользуясь экспоненциальным скользящим средним, оценивать направление движения на предыдущих итерациях алгоритма.

$$v_t = \gamma v_{t-1} + lr * \nabla f(\theta - \gamma v_{t-1})$$

$$\theta_t = \theta_{t-1} - v_t$$

В отличие от Momentum, Nesterov не только накапливает импульс, но и сразу “заглядывает” вперед по вектору обновления, вычисляя значение функции не в точке θ , а в $\theta - \gamma v_{t-1}$, что позволяет быстро скорректировать траекторию спуска в случае, если градиент функции возрастает в текущем направлении движения.

В отличие от AdaGrad, не позволяет использовать высокую скорость обучения.

5. RMSProp

Как было отмечено выше, AdaGrad перестает работать при выборе точки, далеко отстоящей от минимума функции, потому как накопленный квадрат обновлений становится чрезмерно велик, и последующие обновления параметра стремятся к нулю. Adam решает эту проблему, сохраняя вместо полной суммы квадратов обновлений усредненный квадрат градиента.

$$B_t = B_{t-1} * \gamma + (1 - \gamma) \nabla^2 f(\theta)$$

$$\theta_t = \theta_{t-1} - \frac{lr}{\sqrt{B_t + e}} \nabla f(\theta)$$

Ниже приведены графики, сравнивающие вышеописанные методы по различным параметрам:

Зависимость количества арифметических функций от выбора метода

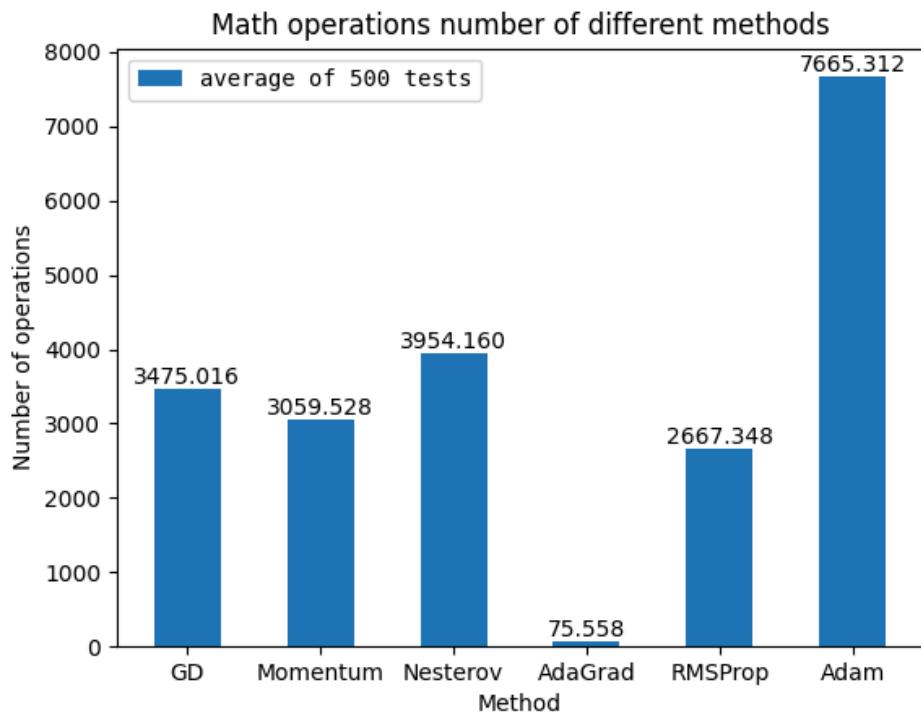


рис №5: Результаты исследования числа вычисляемых функций в зависимости от модификации GD

Задание 5

В качестве примера взята функция $x^2 + 9y^2 + 5$ с начальной точкой (-40, 45).

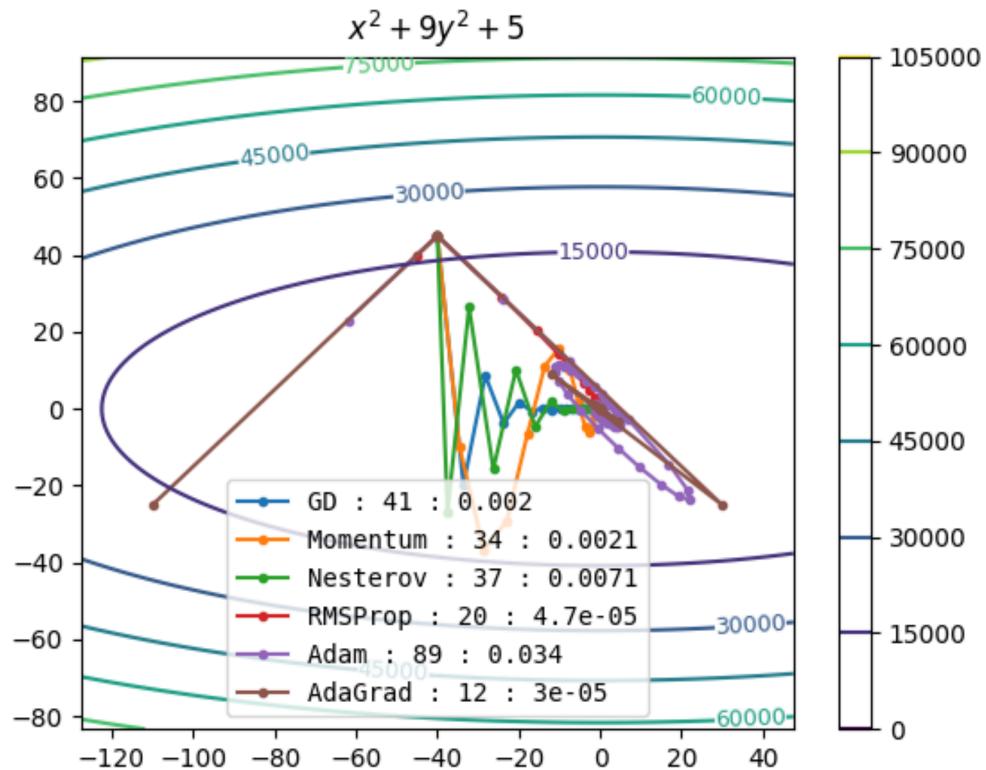


рис №6: Траектории спусков для различных модификаций GD

Доп. задание

1. Polynomial Regression

Суть задачи полиномиальной регрессии заключается в восстановлении по

заданному набору точек $\{(x_i, y_i)\}_{i=1}^n$ на плоскости полинома степени

$k f(x) = \sum c_i x^i$, наилучшим образом приближающим паттерн распределения данных точек, т.е. нахождение набора коэффициентов $\{c_i\}_{i=1}^k$,

минимизирующих сумму квадратов отклонений $\sum_{i=1}^n (\sum_{j=0}^k c_j x_i^j - y_i)^2$.

Ниже приведены примеры работы реализованной регрессии на полиномах различной степени.

2. Regularization

Регуляризация - метод наложения некоторых ограничений на работу метода с целью предотвращения переобучения модели.

a. L1 - regularization

L1 или Lasso регуляризация вводит штраф на абсолютные значения коэффициентов рассматриваемых признаков, тем самым отбирая наиболее значимые признаки модели и обнуляет несущественные.

$$E(\theta) = \sum_{i=0}^n L_i(\theta) + \alpha \sum_{i=0}^k |\theta|$$

Далее приводятся примеры работы полиномиальной регрессии с использованием L1 регуляризации на рассмотренных выше функциях

b. L2 - regularization

L2 регуляризация или регуляризация Тихонова, в отличие от L1 регуляризации, вводит штраф за квадрат коэффициентов признаков, что позволяет избежать появления весов слишком большой величины, а также значительной разницы между разными весами, за счет чего получается учитывать большее количество признаков и избегать переобучения модели.

$$E(\theta) = \sum_{i=0}^n L_i(\theta) + \alpha \sum_{i=0}^k \theta^2$$

c. Elastic regularization

Elastic регуляризация является линейной комбинацией рассмотренных ранее L1 и L2 регуляризаций, сочетая в себе свойства обеих

$$E(\theta) = \sum_{i=0}^n L_i(\theta) + \alpha \sum_{i=0}^k |\theta| + \beta \sum_{i=0}^k \theta^2$$

3. Сравнение работы разных типов регуляризаций при решении задачи о полиномиальной регрессии

$$f_1 = -13x^2 - 23x + 8$$

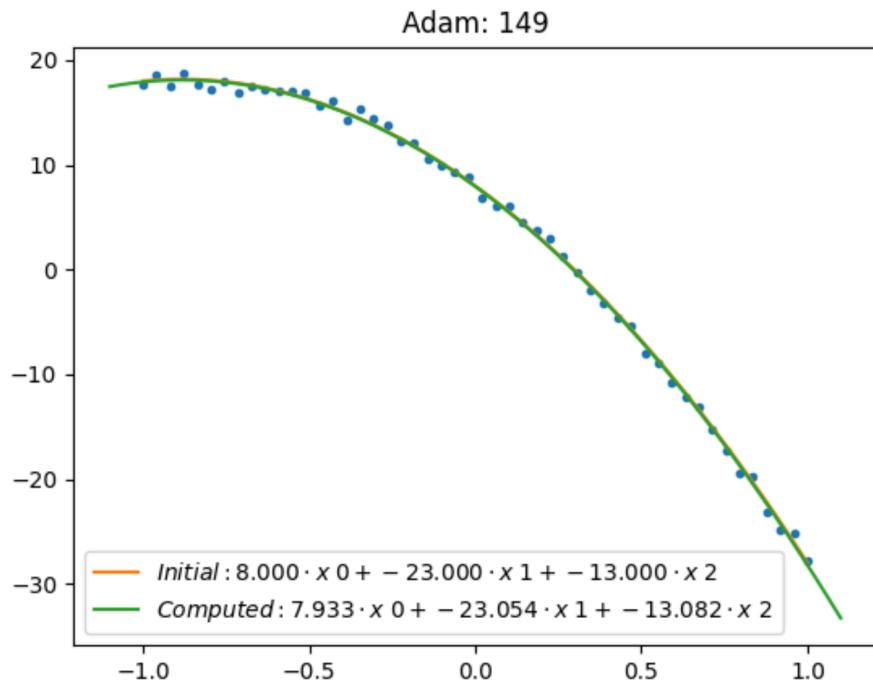


рис №7. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam без использования регуляризации

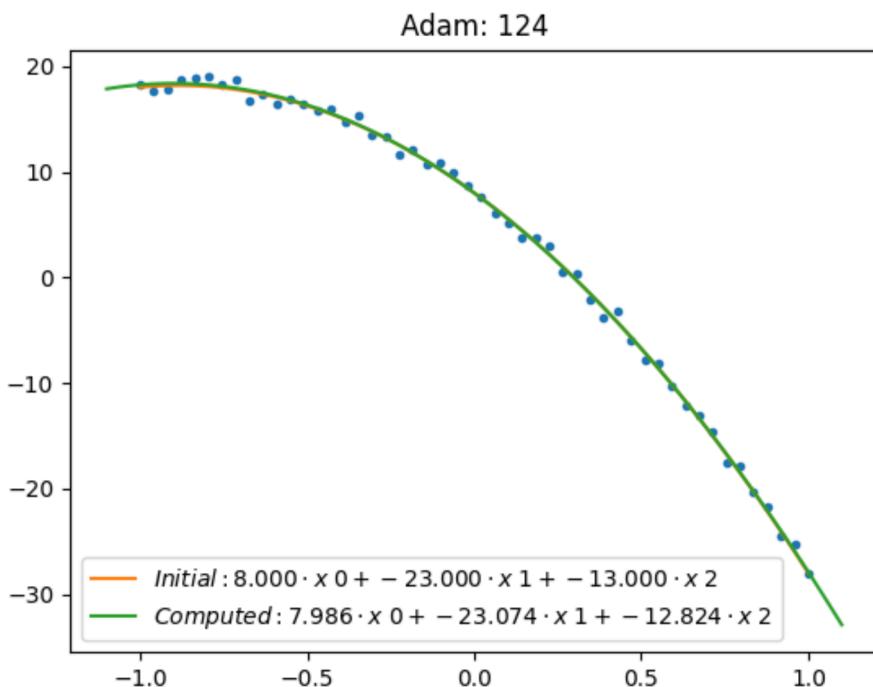


рис №8. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-1; 1]$ с использованием модификации градиентного

спуска Adam и регуляризации L1 (Lasso)

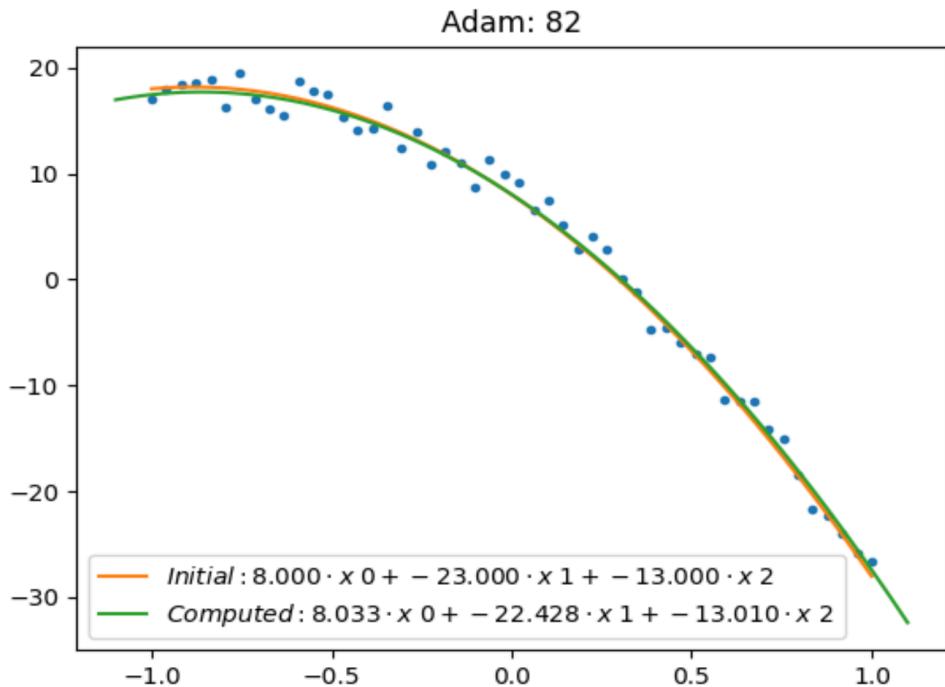


рис №9. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации L2

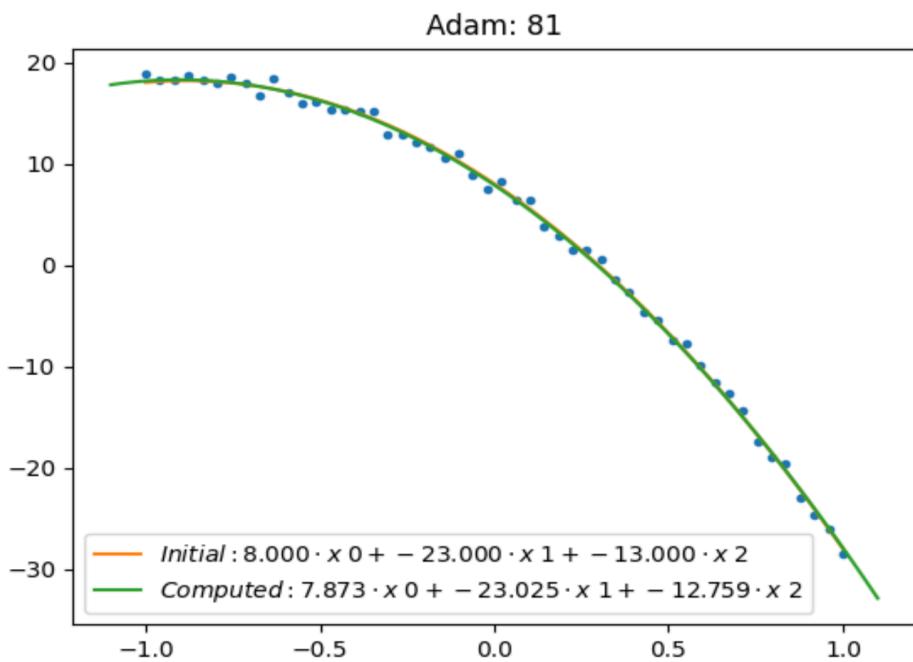


рис №10. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

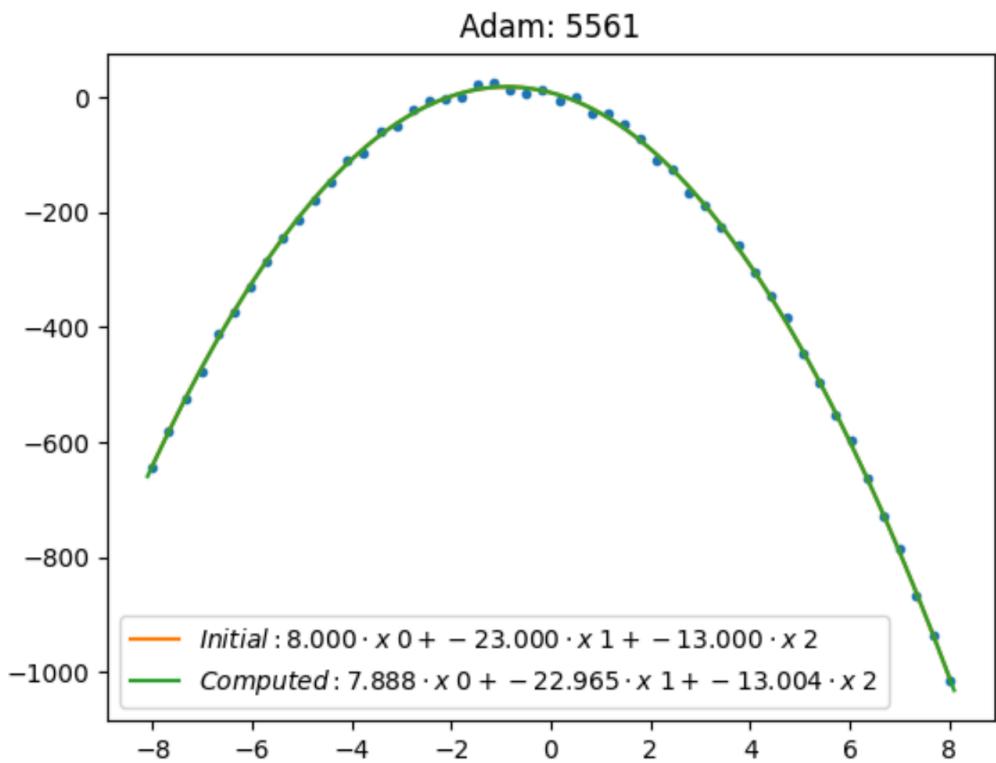


рис №11. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam без использования регуляризации

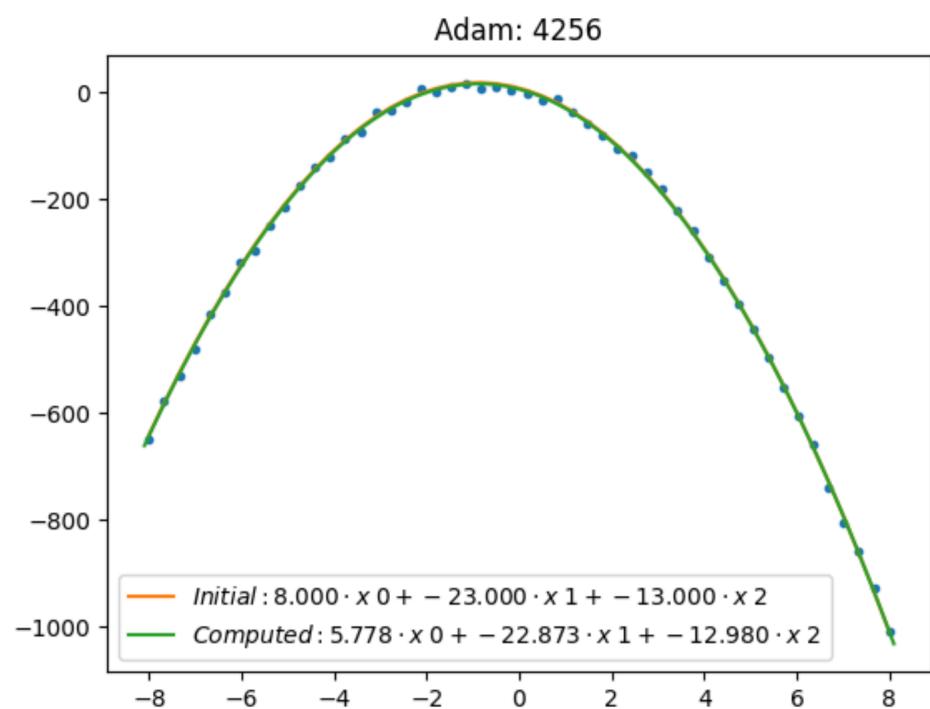


рис №12. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L1 (Lasso)

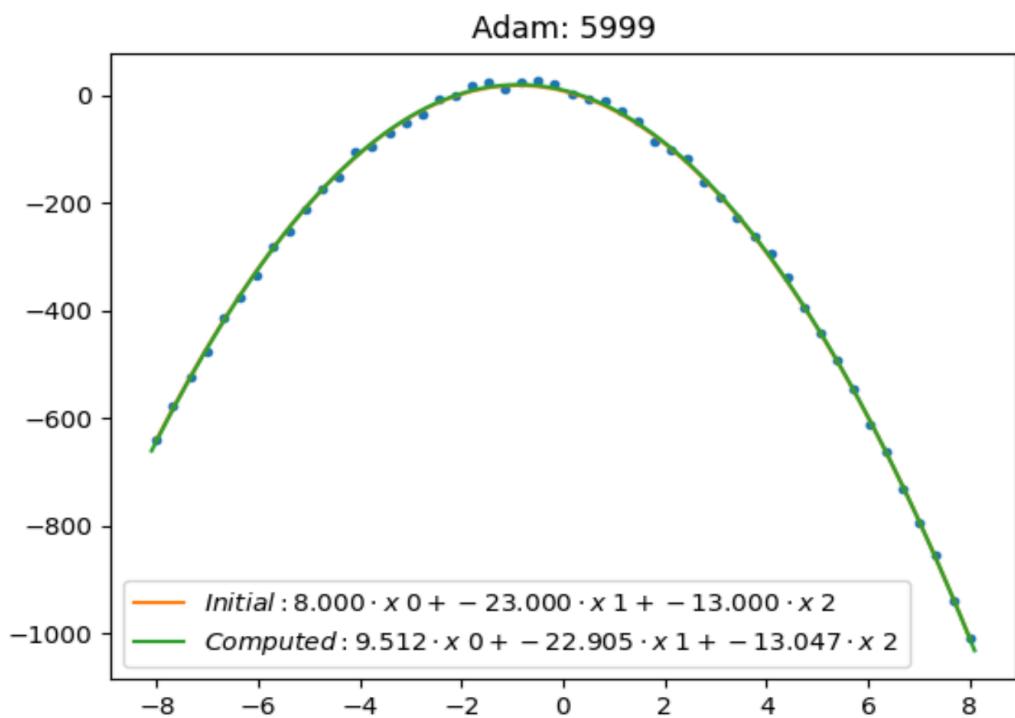


рис №13. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L2

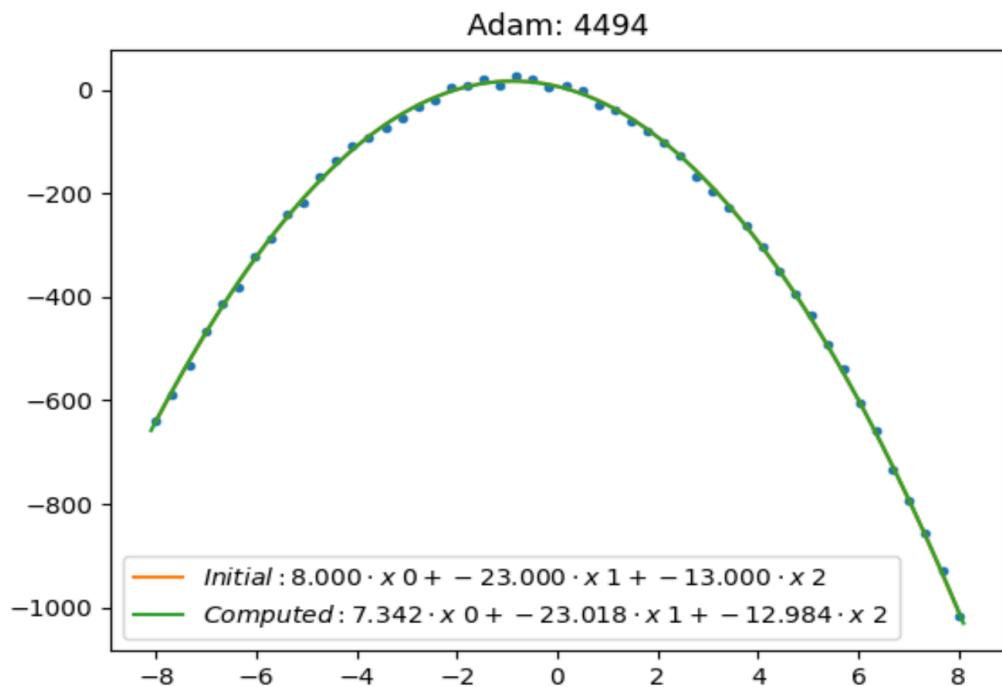


рис №14. Поиск полиномиальных коэффициентов функции f_1 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

$$f_2 = 9x^4 - 3x^3 - 7x^2 + 5x + 10$$

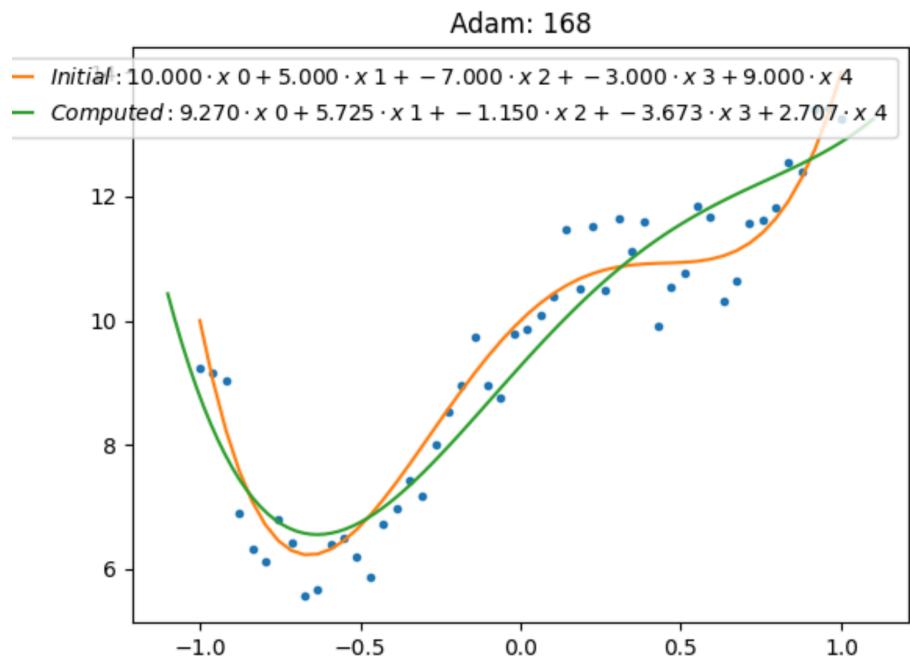


рис №15. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam без использования регуляризации

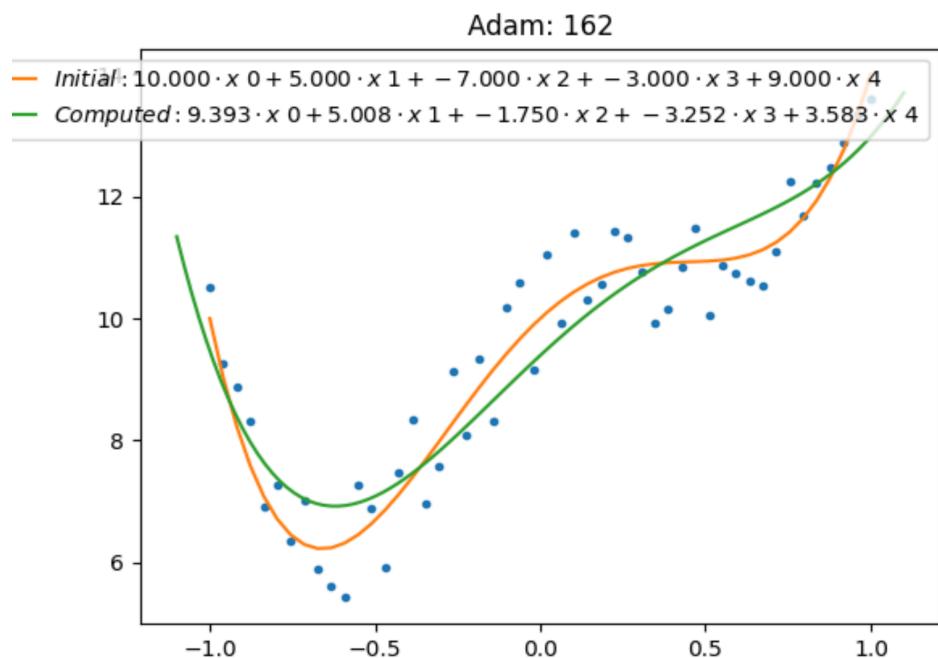


рис №16. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации L1 (Lasso)

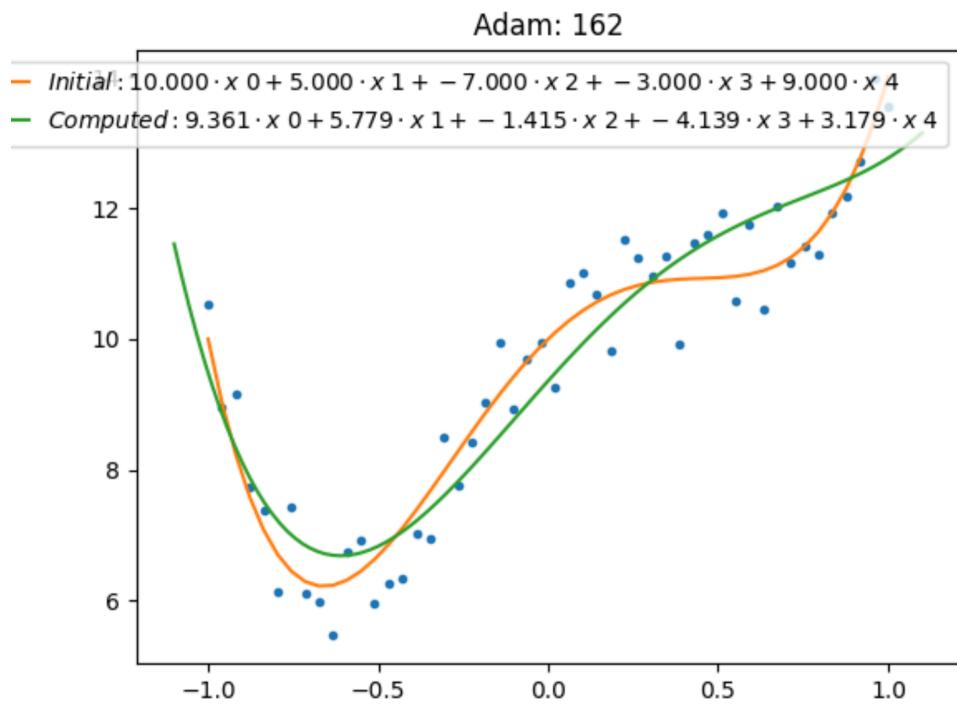


рис №17. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации L2

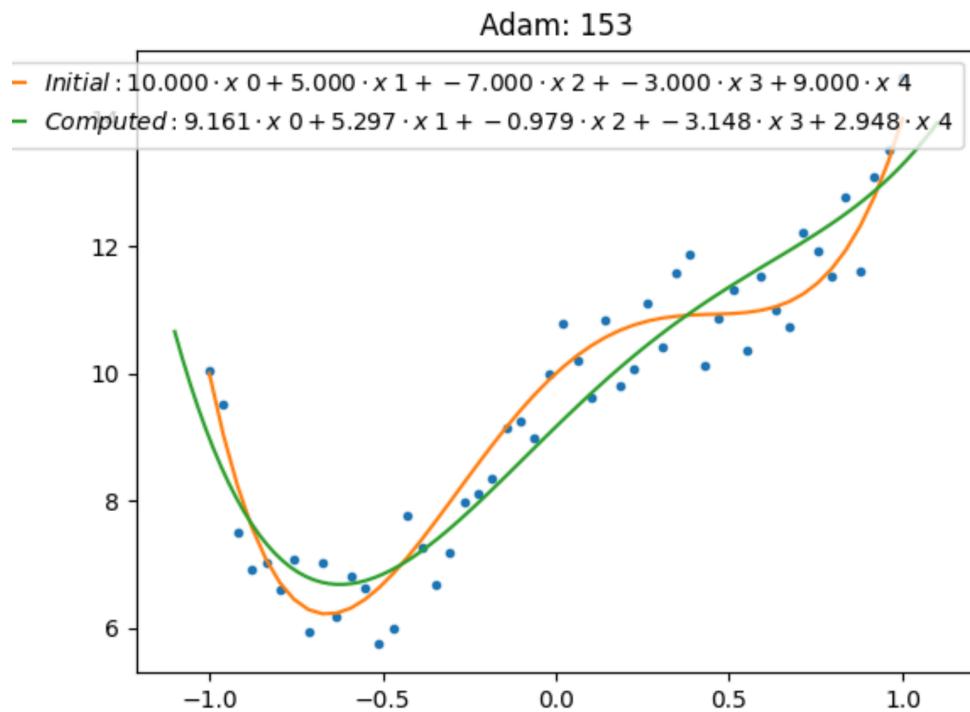


рис №18. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

Adam: 19981

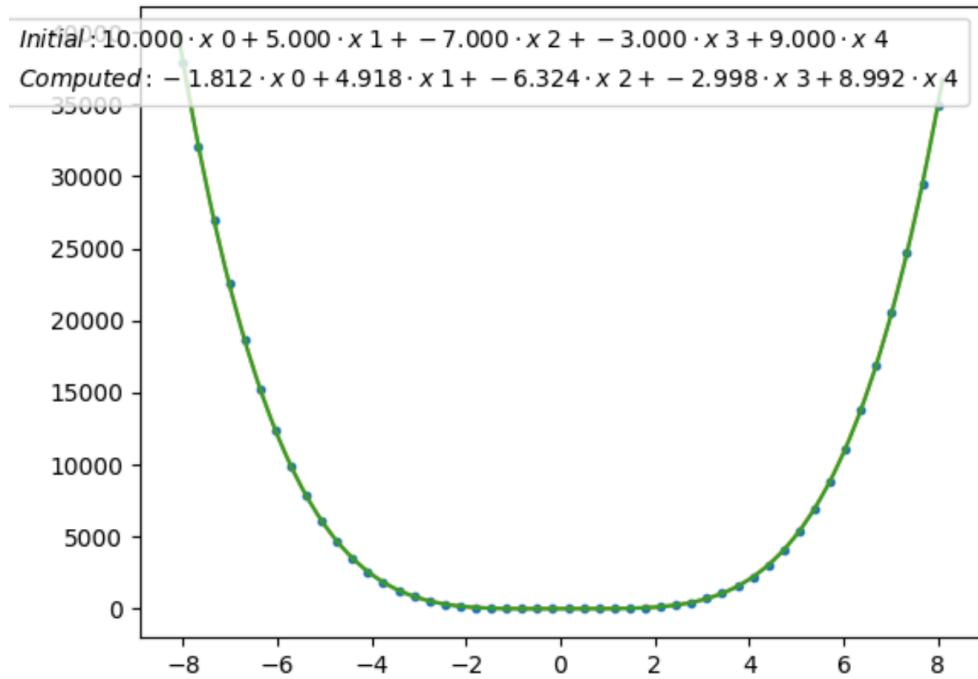


рис №19. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam без использования регуляризации

Adam: 18212

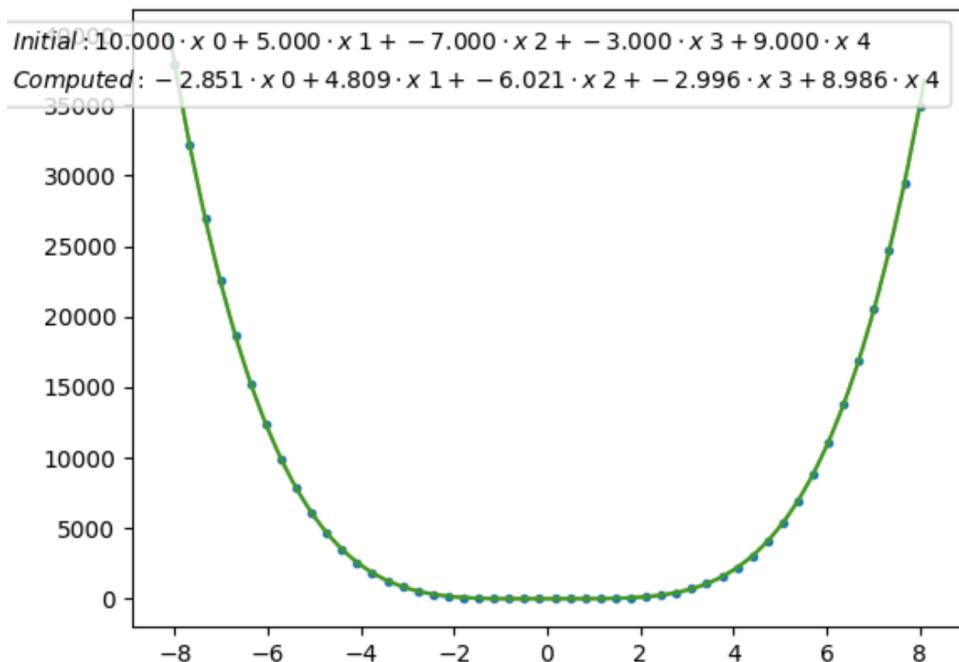


рис №20. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L1 (Lasso)

Adam: 17984

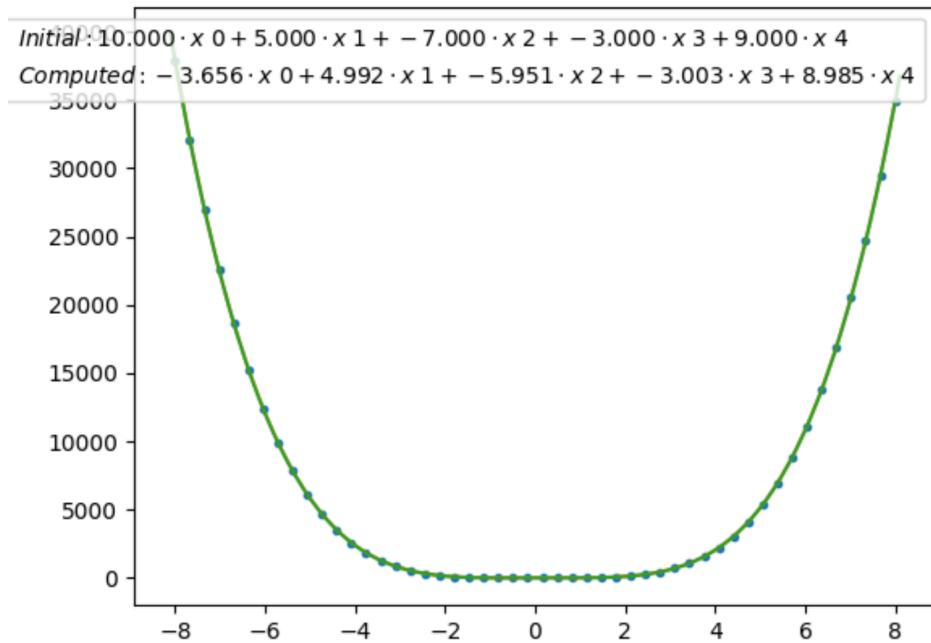


рис №21. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L2

Adam: 15570

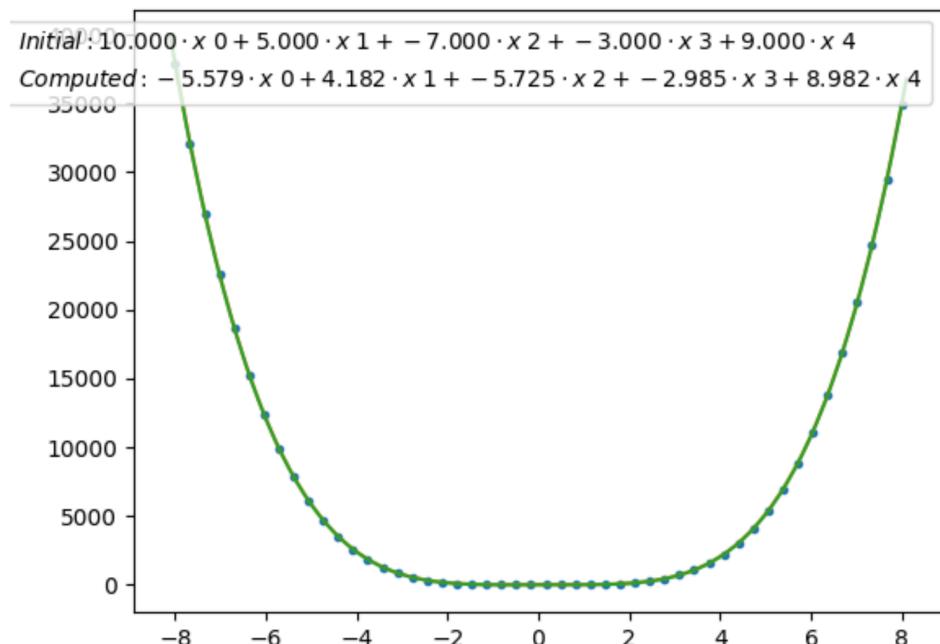


рис №22. Поиск полиномиальных коэффициентов функции f_2 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

$$f_3 = -7x^9 - 11x^7 - 2x^6 + 32x^5 - 10x^4 + 2x^3 - 19x + 1$$

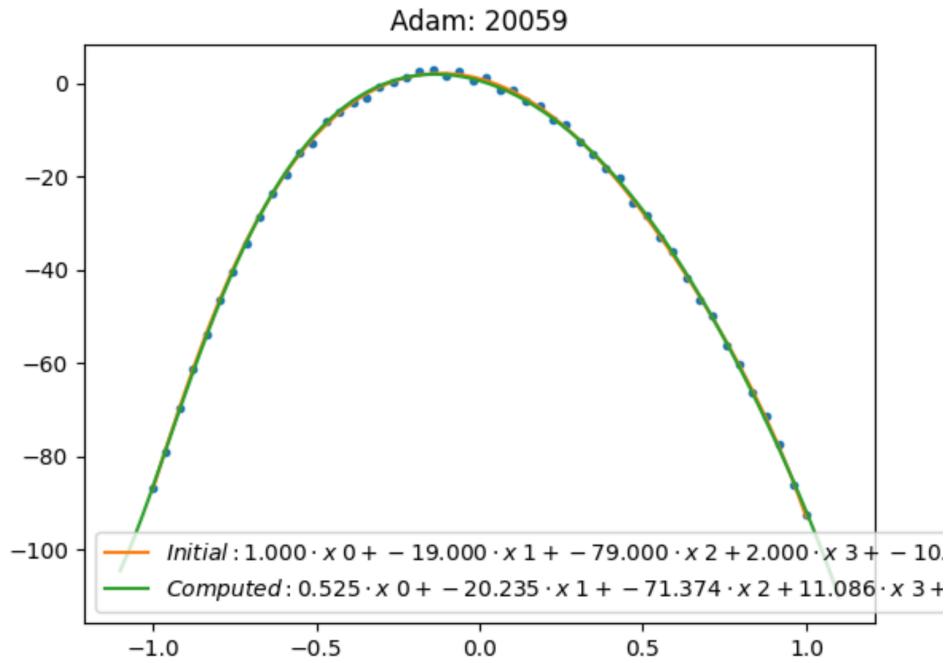


рис №23. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam без использования регуляризации.

Computed:

$$0.52 - 20.23x - 71.37x^2 + 11.09x^3 - 27.67x^4 + 14.52x^5 + 2.52x^6 - 0.4x^7 + 6.79x^8 - 7.59x^9$$

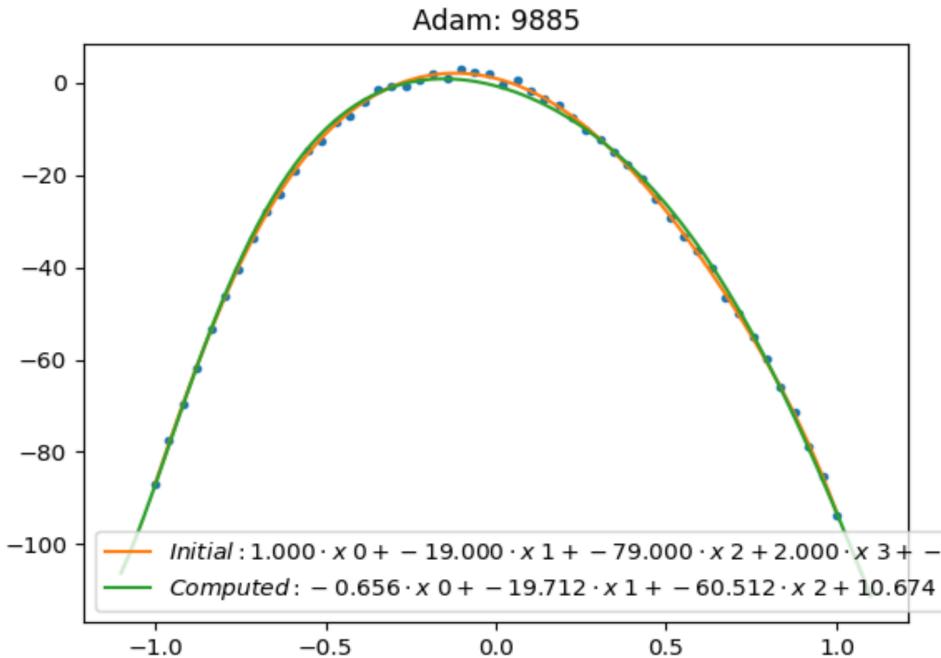


рис №24. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации L1 (Lasso)

Computed:

$$-0.66 - 19.71x - 60.51x^2 + 10.67x^3 - 38.56x^4 + 12.92x^5 - 3.02x^6 - 0.84x^7 + 12.57x^8 - 6.02x^9$$

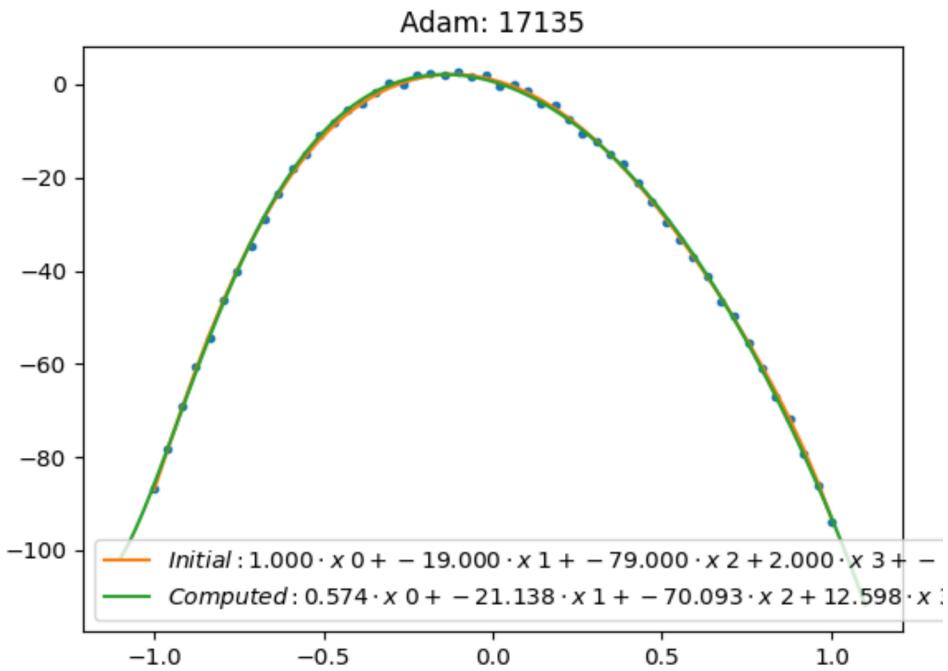


рис №25. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации L2

Computed:

$$0.57 - 21.14x - 70.09x^2 + 12.59x^3 - 30.23x^4 + 14.63x^5 + 1.13x^6 - 1.84x^7 + 9.19x^8 - 7.82x^9$$

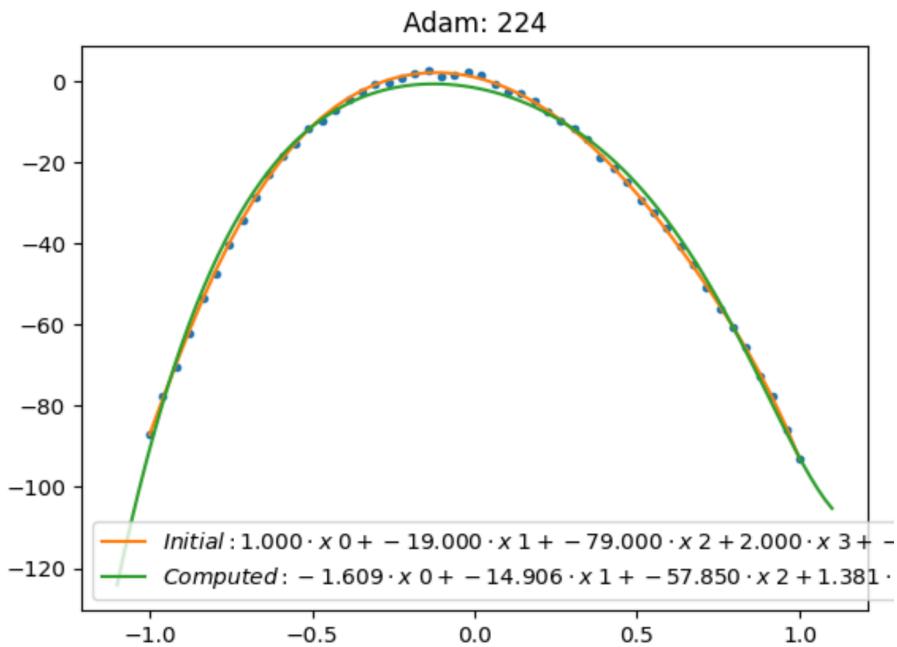


рис №26. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-1; 1]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

Computed:

$$-1.61 - 14.91x - 57.85x^2 + 1.38x^3 - 33.34x^4 + 4.78x^5 - 9.09x^6 + 4.54x^7 + 10.20x^8 + 3.15x^9$$

Adam: 33757

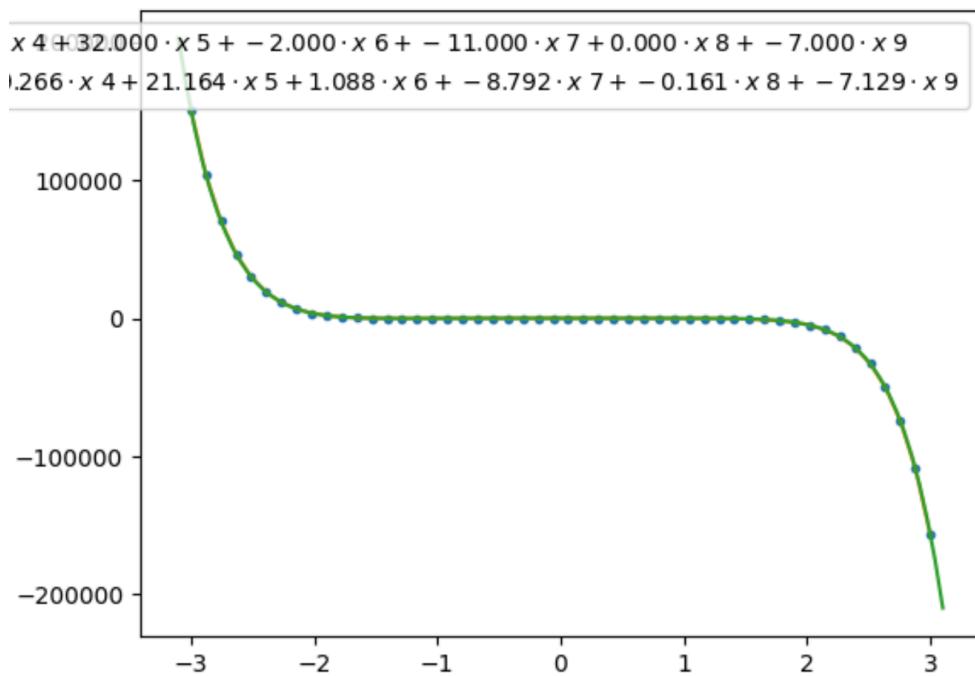


рис №27. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam без использования регуляризации

Computed:

$$- 21.09 + 2.14x - 36.7x^2 + 11.3x^3 - 29.27x^4 + 21.16x^5 + 1.09x^6 - 8.79x^7 - 0.16x^8 - 7.13x^9$$

Adam: 31577

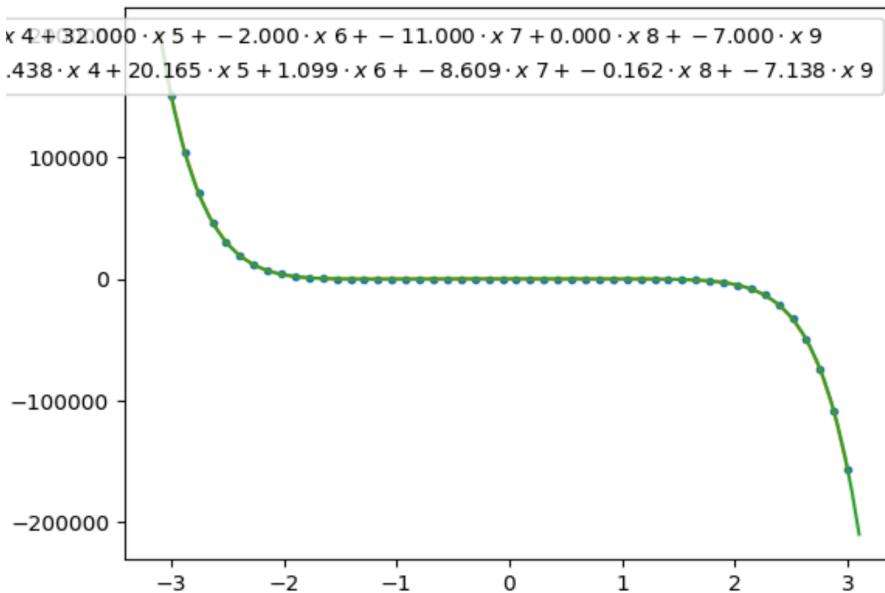


рис №28. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L1 (Lasso)

Computed:

$$- 23.67 + 4.42x - 35.03x^2 + 12.4x^3 - 29.44x^4 + 20.17x^5 + 1.09x^6 - 8.61x^7 - 0.16x^8 - 7.14x^9$$

Adam: 32639

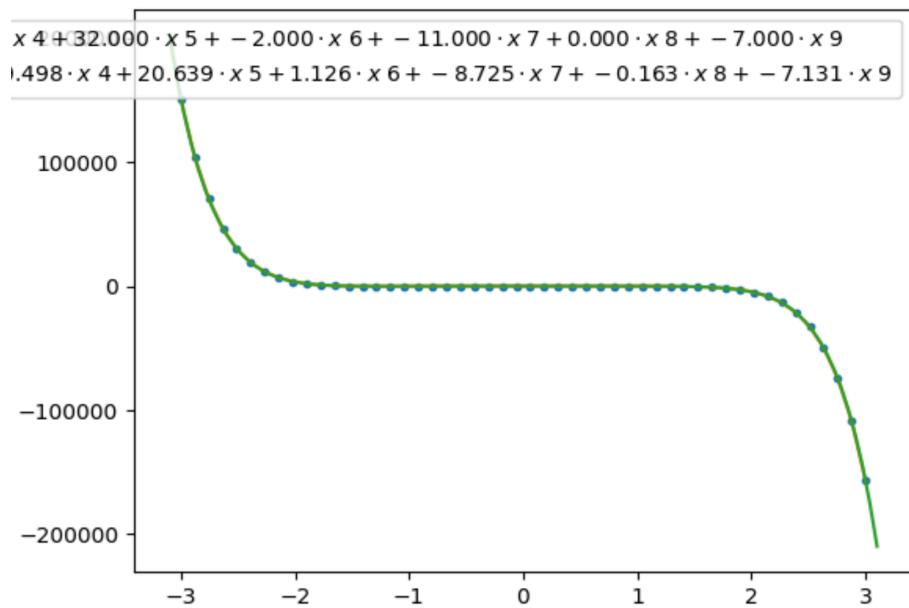


рис №29. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации L2

Computed:

$$- 22.13 + 3.63x - 35.83x^2 + 12.22x^3 - 29.5x^4 + 20.64x^5 + 1.13x^6 - 8.73x^7 - 0.16x^8 - 7.13x^9$$

Adam: 29740

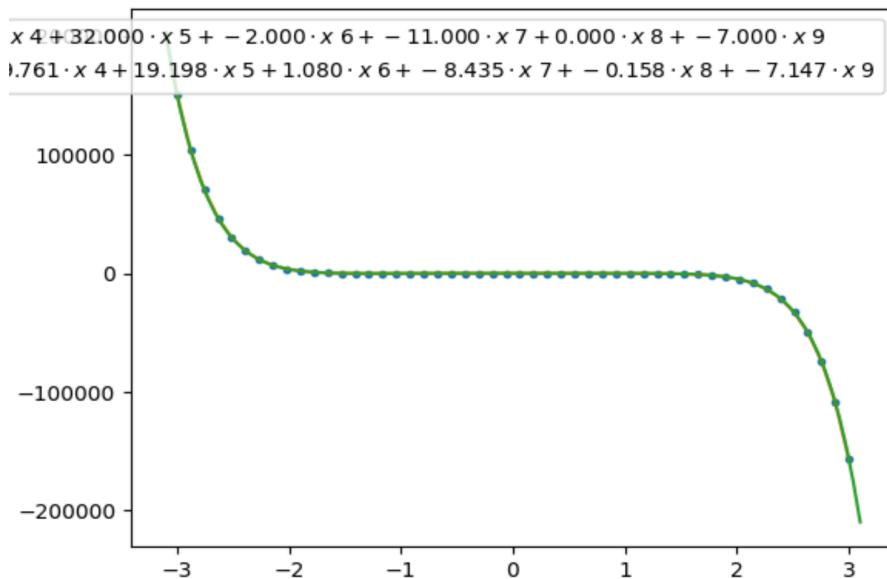


рис №30. Поиск полиномиальных коэффициентов функции f_3 в промежутке $[-8; 8]$ с использованием модификации градиентного спуска Adam и регуляризации Elastic

Computed:

$$- 24.92 + 6.38x - 33.37x^2 + 13.47x^3 - 29.76x^4 + 19.2x^5 + 1.08x^6 - 8.44x^7 - 0.16x^8 - 7.15x^9$$

4. Массовые тесты

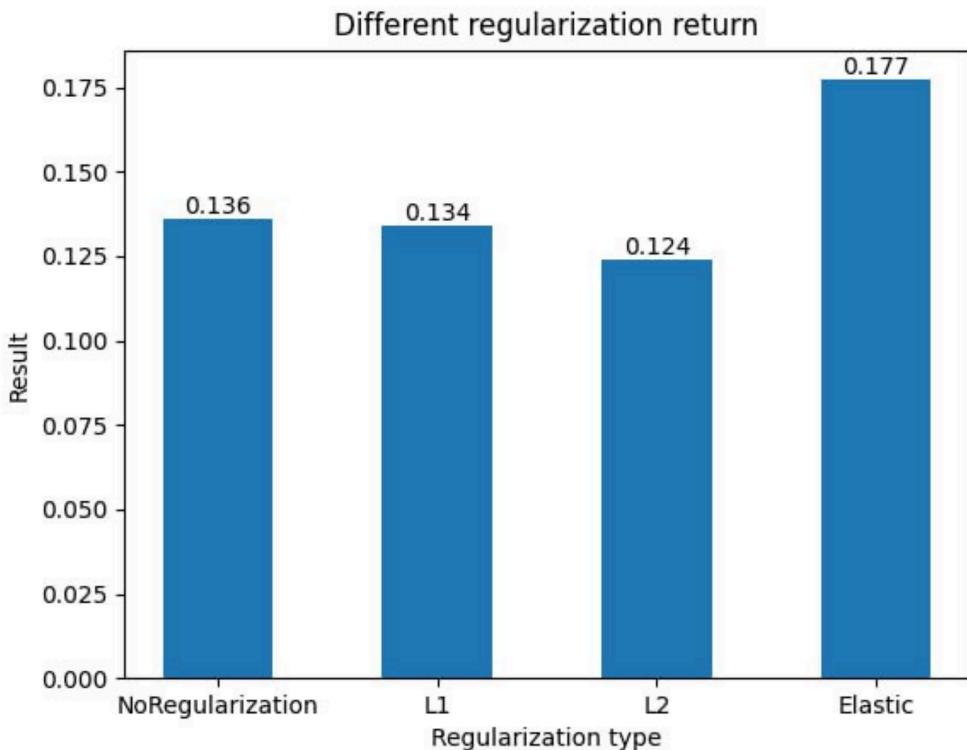


рис. №31. Показатели среднего отклонения значений коэффициентов восстановленного многочлена от коэф-в исходного для разный типов регуляризации по результатам 500 тестов.

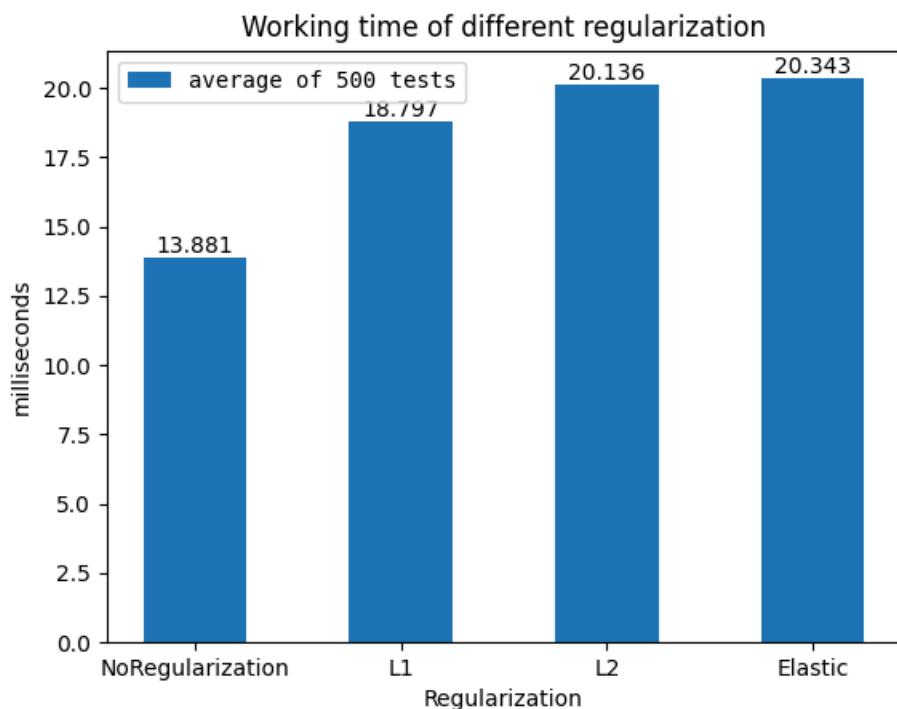


рис. №32. Показатели среднего времени работы метода полиномиальной регрессии с использованием разных типов регуляризации по результатам 500 тестов.

5. Выводы

В пт. 3 дополнительного задания рассматриваются два типа набора данных: со значительной ошибкой (наборами данных в промежутках [-1; 1]) и незначительной (в промежутках [-8; 8]). Здесь и далее мерой незначительной ошибкой будет считаться среднее отклонение, много меньшее масштабов по оси ОY.

В проведенных тестах для f_1 и f_3 можно увидеть, что найденные функции (для поиска с помощью L2 в первом случае, и L1 и Elastic во втором) не совпадают с реальными графиками в точности, а значит,

Как показали одиночные, а также массовые тесты, применение различных видов регуляризаций не оказывает существенного положительного эффекта как на время работы метода и число итераций, необходимых для восстановления полинома, так и на среднее отклонение значений коэффициентов восстановленного многочлена от коэффициентов исходного.

По результатам проведенных массовых испытаний, наилучшие результаты показала L2-регуляризация. Результаты, продемонстрированные L1-регуляризацией, лишь незначительно отличаются в лучшую сторону от результатов работы метода без применения регуляризации. Показатели Elastic регуляризации, по итогам проведенных тестов, только в худшую сторону отличаются от показателей метода без регуляризации.