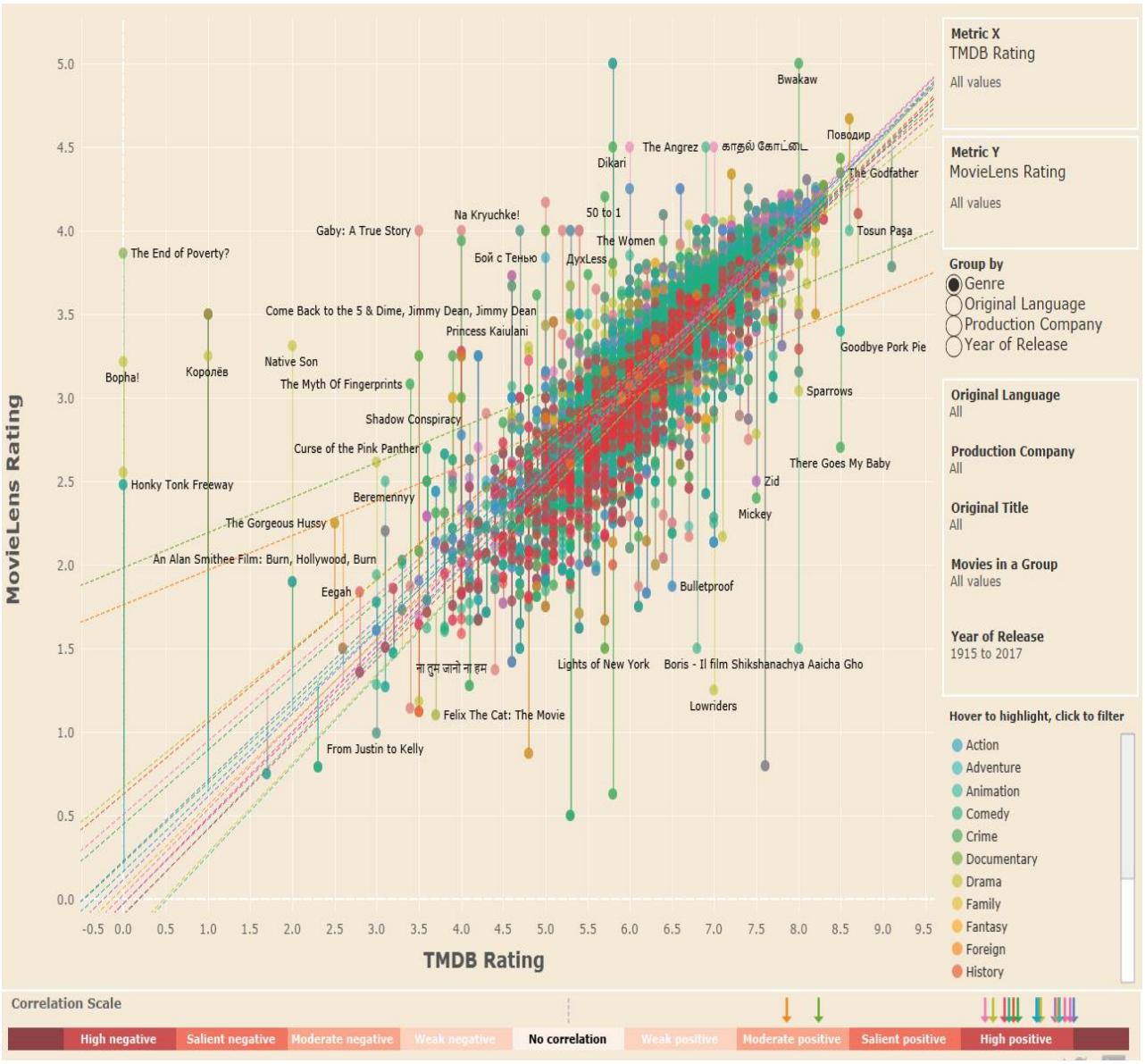


Методы оптимизации

Лабораторная работа 3

ЛИНЕЙНАЯ РЕГРЕССИЯ

Авторы: Куприянов М3233, Долматова М3232, Шайдулин М3233

Ссылка на реализацию

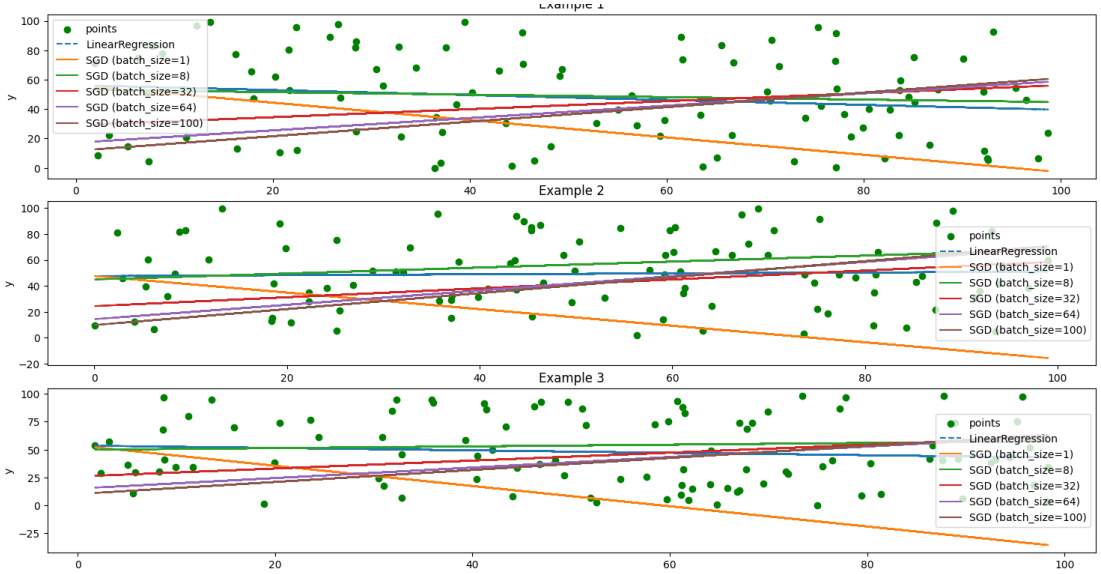
Цели исследования

В рамках лабораторной работы 3, наша команда хотела разобраться с различными вариациями линейной регрессии, полиномиальной регрессии и различные модификации данных алгоритмов, а также изучить возможности готовых версий данных методов из различных библиотек в Python 3 и сравнить полученную скорость работы, эффективность и точность с тем, что получилось у нас.

РАБОТА

Начнем с первого пункта: в нем нужно было реализовать и исследовать на эффективность SGD для решения линейной регрессии с разным размером батча – от одного до размера полной коллекции или так называемый обычный метод GD);Идея метода заключается в том, что нам нужно по набору точек найти прямую, которая будет “как можно ближе ко всем точкам”, для этого мы будем минимизировать функцию подсчета ошибок – квадрат разности реальных координат точек и предполагаемых точек делить на их количество. Будем делать это при помощи градиентного спуска, но в качестве вычисления градиента будем брать случайную перестановку и считать градиент внутри батча. Так мы хотели исследовать, какой из размеров будет оптимальным.

Перейдем к результатам и выводам, которые мы можем из них сделать. Исходя из комментария по предыдущей работе мы решили сделать результаты менее объемными и более простыми для восприятия,Начнем с примера работы программы для 100 случайно сгенерированных точек и разных размеров батча, мы можем получить такие визуализации прямых:

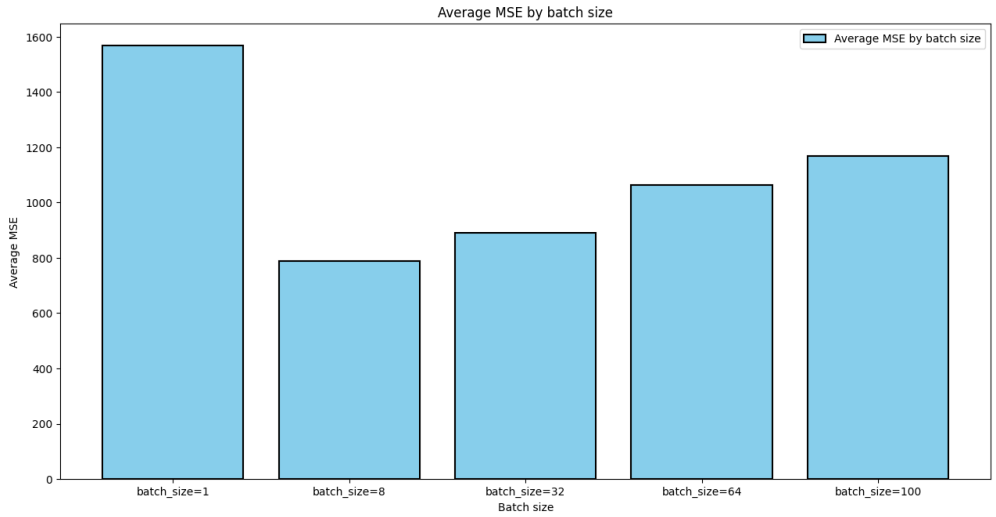


Вот так примерно будут выглядеть результаты в виде небольшой таблицы, с учетом числа вычислений и полученной суммой ошибок:

Experiment	Model	Batch Size	Slope	Intercept	MSE	Calculations
1	LinearRegression		0.084878	45.149380	716.189244	0
1	SGD	1	0.208924	45.034996	764.332392	100000
1	SGD	8	0.474428	21.619548	863.414839	13000
1	SGD	32	0.690706	6.789896	1100.203768	4000
1	SGD	64	0.717418	3.533132	1166.000190	2000
1	SGD	100	0.734908	2.295964	1193.135675	1000
2	LinearRegression		0.116461	41.582313	779.619793	0
2	SGD	1	0.726511	41.250252	2140.052648	100000
2	SGD	8	0.521806	19.302697	932.281902	13000
2	SGD	32	0.631873	6.011992	1095.006798	4000
2	SGD	64	0.669071	3.124459	1148.015686	2000
2	SGD	100	0.679040	2.029337	1169.170125	1000
3	LinearRegression		-0.150984	59.401505	884.388068	0
3	SGD	1	0.655341	58.844911	3233.743916	100000
3	SGD	8	0.393654	25.725958	1152.483089	13000
3	SGD	32	0.646240	7.849772	1494.202284	4000
3	SGD	64	0.627877	4.062972	1578.298288	2000
3	SGD	100	0.671608	2.634046	1612.548516	1000

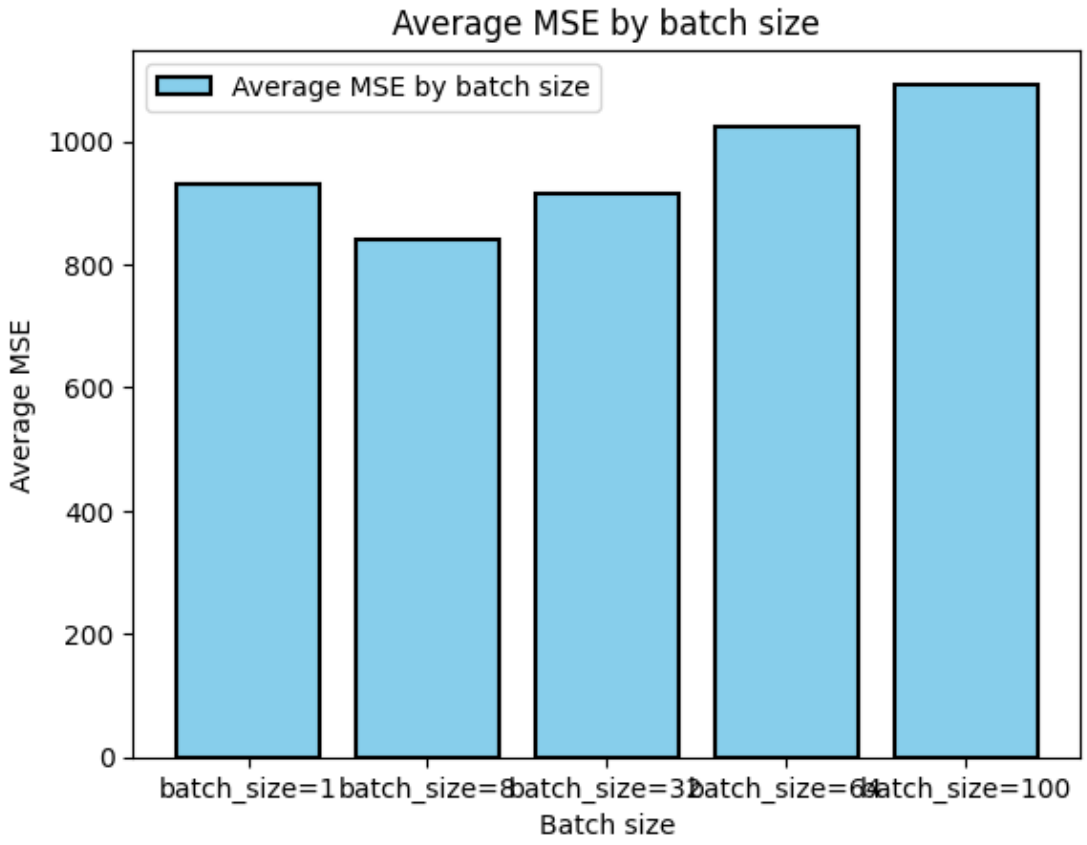
Table 1: Experiment Results

Чтобы сделать результаты более простыми для восприятия, выведем диаграмму среднего размера ошибок в зависимости от размера батча, при шаге = 0.001 и ограничением с 1000 итераций при большом числе экспериментов:



Для сравнения вот таблица и график при увеличении числа итераций до 5000,

Experiment	Model	Batch Size	Slope	Intercept	MSE	Calculations
1	LinearRegression		0.080661	43.976329	825.327189	0
1	SGD	1	0.021755	43.851521	836.078499	500000
1	SGD	8	-0.039264	42.982066	878.784423	65000
1	SGD	32	0.339576	27.068177	912.714937	20000
1	SGD	64	0.521191	16.708398	1053.590077	10000
1	SGD	100	0.581599	11.586633	1145.937555	5000
2	LinearRegression		0.084284	43.727196	845.787294	0
2	SGD	1	-0.207758	43.937105	1111.071945	500000
2	SGD	8	0.140305	41.383926	848.299161	65000
2	SGD	32	0.341724	22.691493	964.382570	20000
2	SGD	64	0.553716	13.394547	1060.846851	10000
2	SGD	100	0.621429	9.121652	1125.696796	5000
3	LinearRegression		0.139530	39.666530	772.767377	0
3	SGD	1	-0.002284	39.707415	845.251914	500000
3	SGD	8	0.096215	37.763062	791.861018	65000
3	SGD	32	0.359863	21.097259	864.686567	20000
3	SGD	64	0.543891	12.524612	952.020415	10000
3	SGD	100	0.588774	8.549876	1007.615524	5000



Как мы можем видеть самое точно значение получается у готовой реализации, но при batch size = 8 получаем очень близкий результат, притом, что у нас не так много вычислений.

Из данных результатов можем сделать вывод:

1. SGD с маленьким размером батча:

плюсы:

- Быстрая итерация.
- Высокая вероятность избегания локальных минимумов (из-за шума при вычислении градиента).

минусы:

- Высокая вариативность градиента, что может затруднять сходимость.
- Большой шум при обновлении параметров.

Средний батча, в нашем случае 8, это наиболее эффективный вариант.

плюсы:

- Умеренный шум в оценке градиента.
- Баланс между скоростью обучения и стабильностью.
- Точный результат

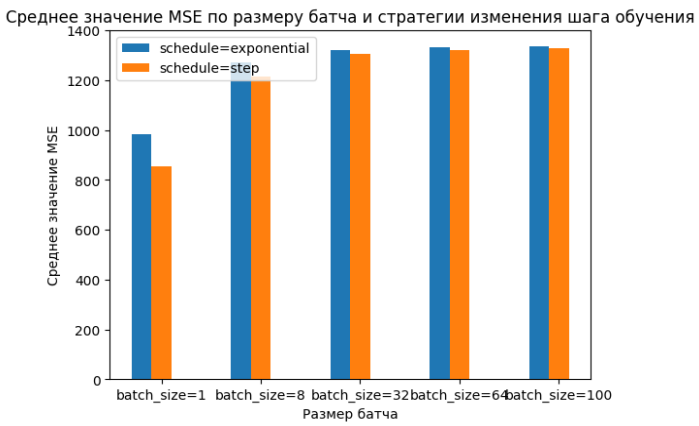
минусы:

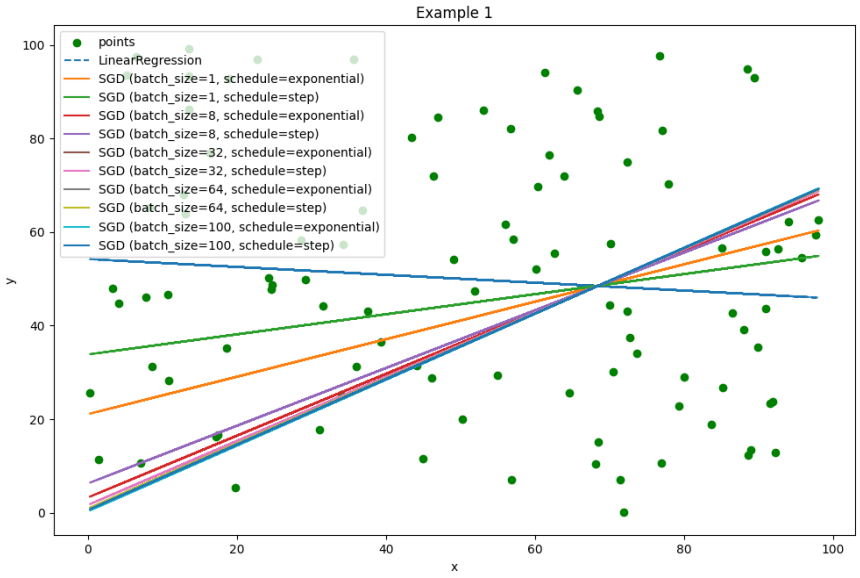
- выбора оптимального размера батча может быть неочевидным.

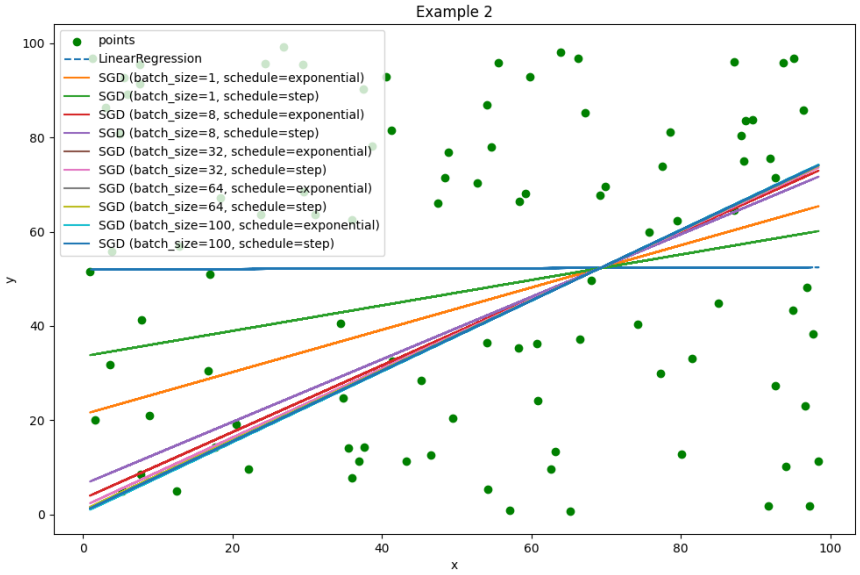
В общем, маленькие батчи могут улучшить точность за счет добавления шума, который помогает не застрять в локальных минимумах. Но они могут замедлить сходимость или сделать её менее стабильной + требуют много итераций. Большие батчи делают процесс более гладким и предсказуемым, но могут застрять в локальных минимумах, зато при этом требуют меньше итераций и в теории их можно оптимизировать через использование векторов или параллелизма.

SGD для решения линейной регрессии с экспоненциальной или ступенчатой функцией изменения шага

Результаты в виде диаграммы:







Рассмотрим плюсы и минусы каждого подхода:

Экспоненциальное уменьшение шага (Exponential Decay)

Плюсы:

- У такого подхода плавное уменьшение шага на каждой итерации, что дает стабильную постепенную сходимость.
  А это помогает избежать сильных изменений при обновлениях веса, а значит будет меньше вероятность колебаний и застреваний около минимума.
- Такое уменьшение шага может быть выгодно, при функции с большим количеством локальных минимумов, поскольку позволяет более эффективно исследовать параметры на начальных этапах обучения и более точно сходиться к минимуму в конце.

Минусы:

- Есть вероятность слишком быстрого уменьшения шага, если коэффициент уменьшения изначально выбран неправильно, то шаг может уменьшаться слишком быстро, что приведет к замедлению сходимости или к застреванию в локальных минимумах.
- Нужен подбор оптимального начального шага и коэффициента уменьшения
- Меньшая точность.

Ступенчатое уменьшение шага (Step Decay)

Плюсы:

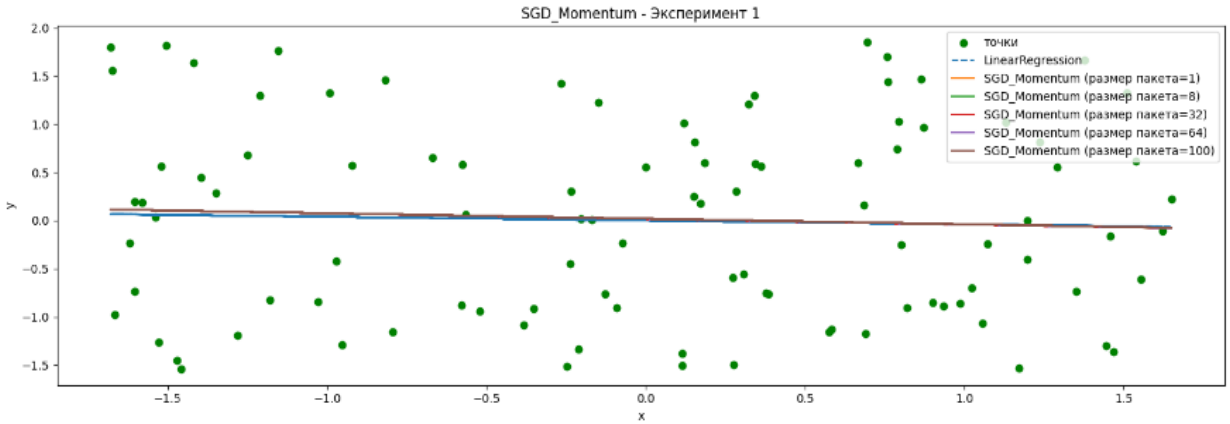
- Лучшая точность (что ярко видно из результатов).

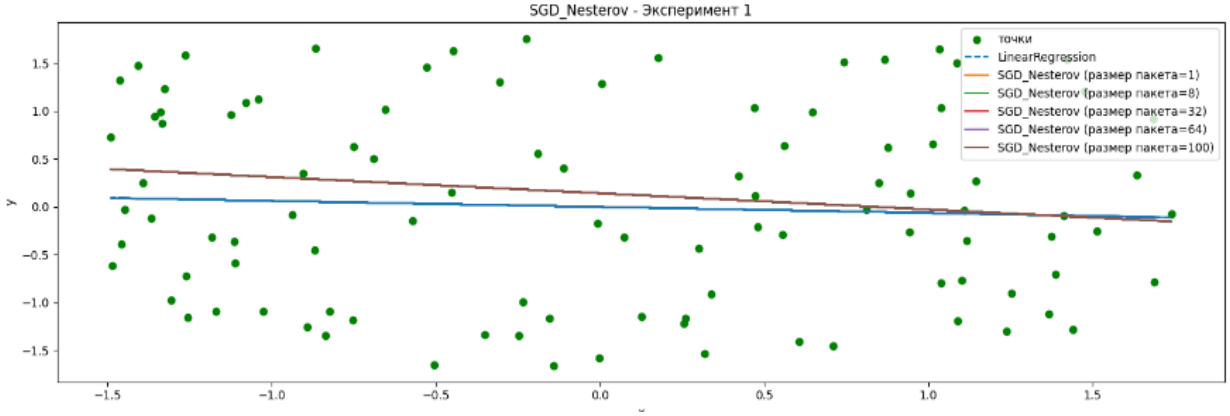
- Ступенчатое уменьшение шага проще, так как изменение шага происходит через фиксированные интервалы времени, + более легкая настройка
- Есть возможность изменения шага на заранее определенных итерациях, дает возможность контролировать обучение и менять что-то в зависимости от промежуточных результатов.
- Так как шаг уменьшается через фиксированные промежутки, можно избежать риска слишком быстрого уменьшения шага, который будет при экспоненциальном уменьшении.

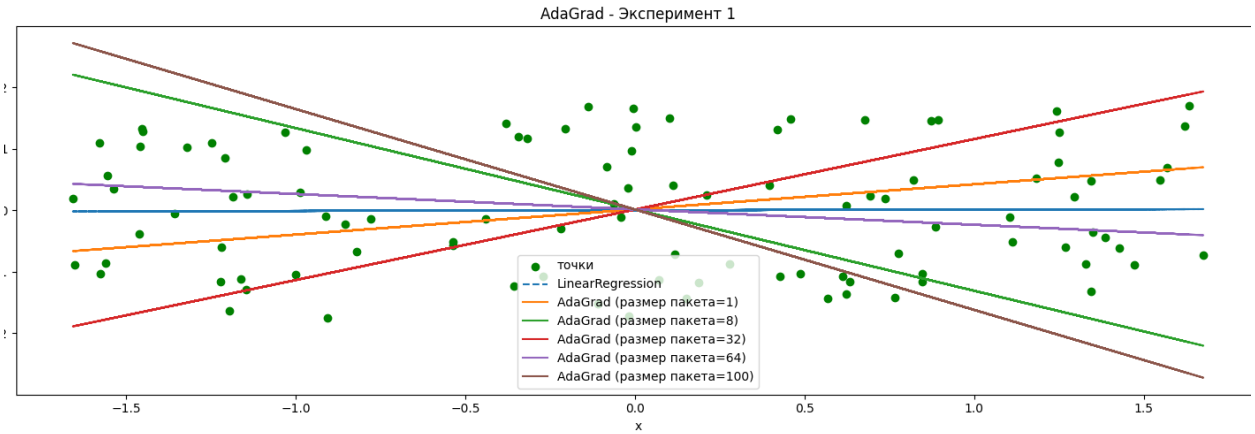
Минусы:

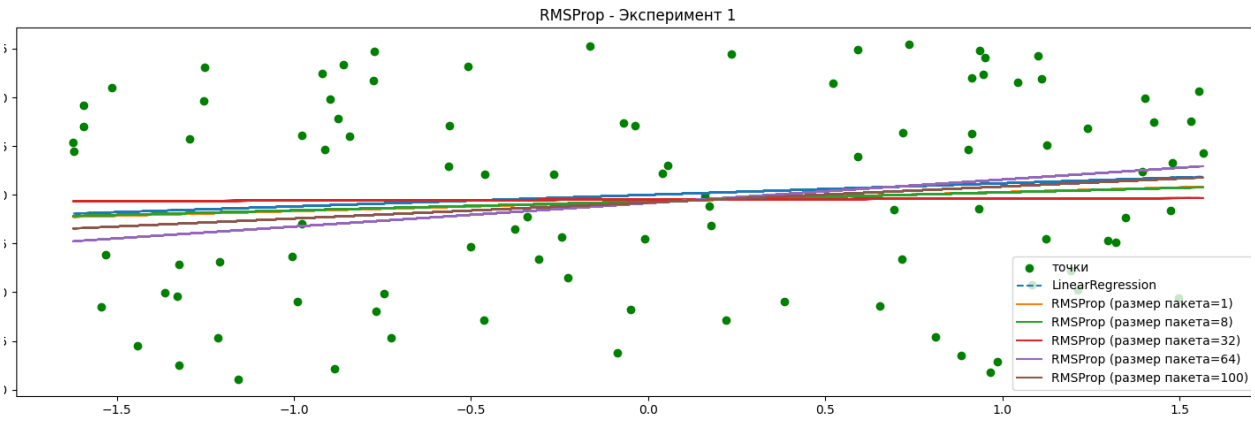
- Возможно слишком резкое изменения шага, что приводит к замедлению обучения и небольшим колебания при подходе к минимуму.
- Фиксированные интервалы уменьшения шага могут быть менее выгодны для меняющегося обучения (сравнения с “плавным” экспоненциальным уменьшением)
- Чуть меньшая скорость чем у прошлого метода.

МОДИФИКАЦИИ









1 Momentum

Идея модификации:

Добавление инерции, чтобы сгладить и ускорить поиск (притом если β близок к 0, метод Momentum приближается к стандартному SGD, где влияние прошлых градиентов минимально, иначе сильно увеличиваем предыдущие градиенты)

Плюсы:

- Ускоряет движение по направлениям с постоянным градиентом и помогает преодолевать плоскости.
- Уменьшает застревания в направлениях с переменными градиентами.

Минусы:

- Надо подбирать коэффициент инерции β.

2 Nesterov

Идея модификации:

Предвосхищение следующего шага для корректировки направления градиента.

Плюсы:

- точное вычисление направления, что ускоряет сходимость (по сравнению с обычным SGD).
- Улучшает стабильность при сложной функции.

Минусы:

- Занимает большое время при оценке градиента на предвосхищенном шаге.

3 RMSProp
Идея модификации:
Использование среднего квадратов градиентов для управления шагом обучения: $E[g^2]_t=\beta E[g_2]_{t-1}+(1-\beta)gt^2$

а потом делим шаг обучения на корень полученной формулы + eps и умножаем градиентю.

Плюсы:

- Предотвращает чрезмерное уменьшение шага обучения.

Минусы:

- подбор коэффициента затухания.

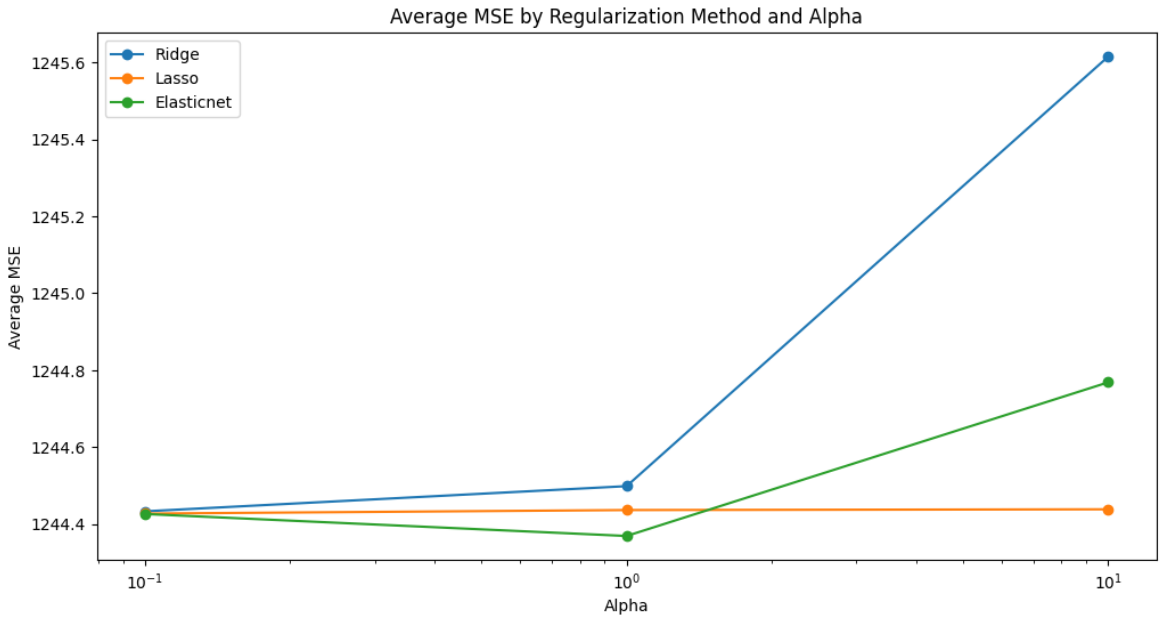
4 Adam это просто AdaGrad + RMSProp

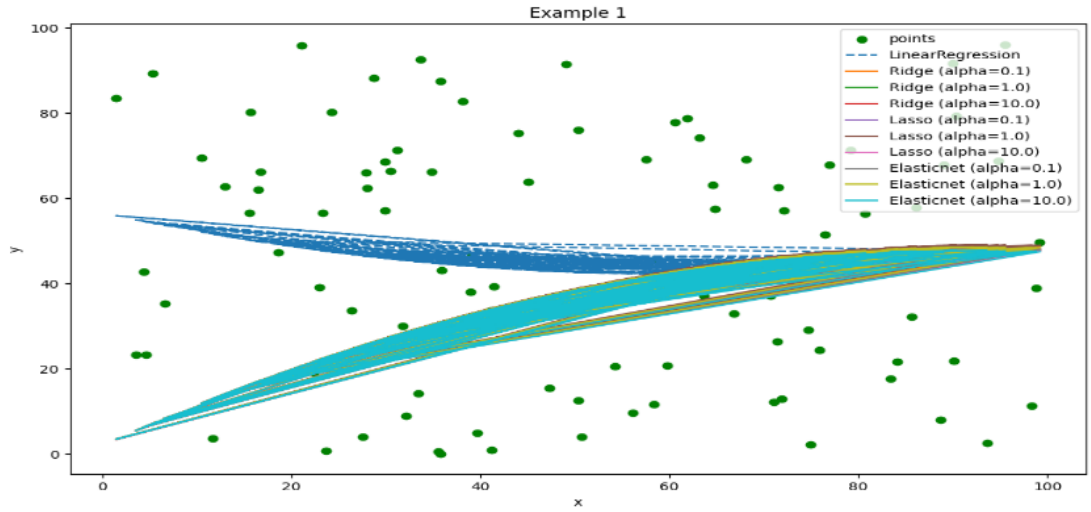
ДОП. ПУНКТ 1

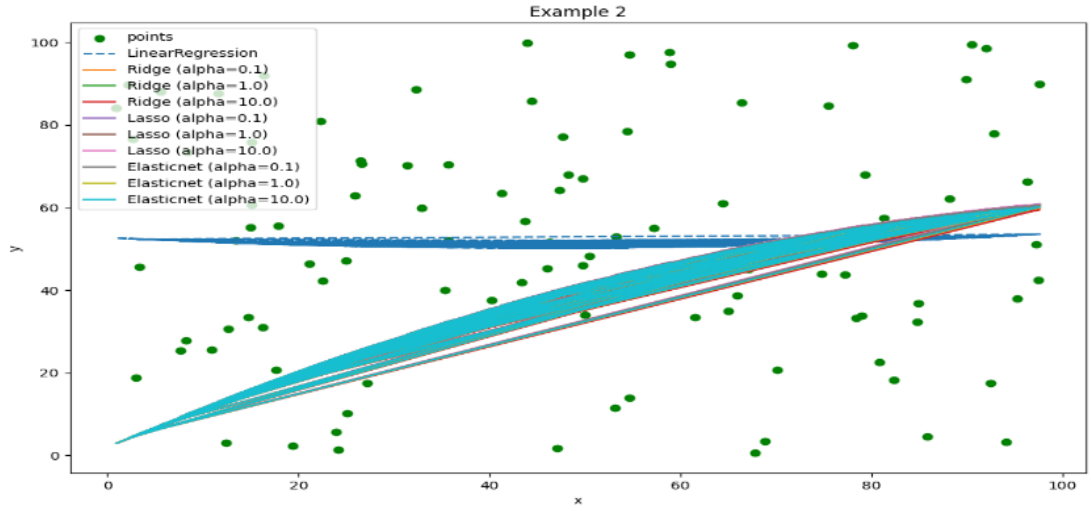
Полиномиальная регрессия:

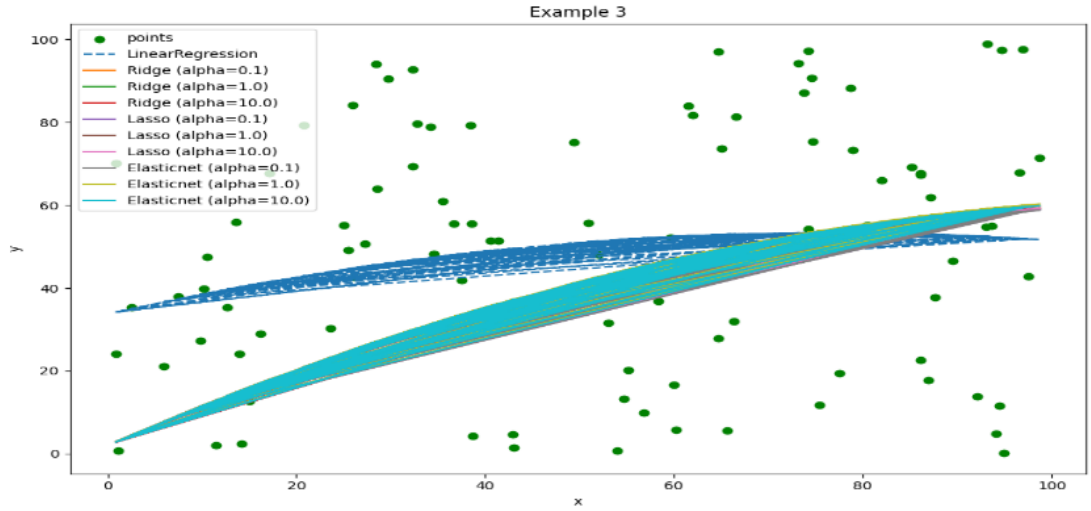
В данном доп. Пункте надо реализовать полиномиальную регрессию, отличие от обычной – что теперь у нас будет не прямая, которую мы ищем, а многочлен, то есть прибавился еще один вес, который мы ищем.

Для больших степеней получается достаточно сложно, поэтому рассматривали степень = 2, а также сравнили разные подходы для регуляризации. А вот среднее значение ошибки в зависимости от alpha:









(Таблицы с подробными результатами можно посмотреть в реализации, не будем добавлять, чтобы не загромождать отчет)

Кратко про эти методы:

1. **L1 Регуляризация (Lasso)**

   L1 регуляризация добавляет к функции потерь сумму значений коэффициентов. Это приводит к обнулению некоторых коэффициентов, что делает модель разреженной и способствует отбору признаков.

   Плюсы:

   - многие коэффициенты равны нулю.
   - Высокая точность (если не поменять важный признак)
   - простота

   Минусы:

   - Нестабильность градиентов.
   - Возможное удаление значимых признаков при корреляции

2. **L2 Регуляризация (Ridge)**

   L2 регуляризация добавляет к функции потерь сумму квадратов коэффициентов, что предотвращает сильное увеличение весов, что уменьшает колебание градиентов, но не обнуляет коэффициенты.
   Плюсы:

   - Стабильность и гладкость модели.
   - Хорошо определенные градиенты.

   Минусы:

   - Менее эффективна при корреляции признаков, как видно из графика

3. **Elastic Регуляризация**

   Elastic регуляризация что-то типа L1 и L2 вместе, она добавляет к функции потерь линейную комбинацию суммы значений и суммы квадратов коэффициентов. Берем идеи предыдущих методов и их плюсы.

   Минусы:

   - Сложность.
   - Более высокая вычислительная стоимость.

ДОП. ПУНКТ 2

Тут надо было просто рассмотреть конкретный пример и объяснить, что происходит, поэтому не будем писать еще раз, смотрите Отчет

ЗАКЛЮЧЕНИЕ

В заключение хотелось сказать, что наша команда подробно рассмотрела алгоритм линейной регрессии, а также его модификации, вникла в детали реализации и необходимой теории, что позволило осознать преимущества и недостатки методов и сравнить это с тем, что вышло на практике. Также учли просьбу представления данных в более наглядном и визуализированном виде без добавления громоздких таблиц с данными и старались добавлять больше графиков и диаграмм, но если что, все таблицы можно найти в отчете и подробнее ознакомится с результатами.