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

Методы оптимизации
Отчёт по лабораторной работе №3
Метод стохастического градиентного спуска (SGD) и его модификации

Работу выполнили:
Зызлаев Артем M3233
Жунусов Данияр M3233
Преподаватель:
Андреев Юрий Александрович

Санкт-Петербург
2025

Постановка задач и целей работы

1. Реализовать и исследовать на эффективность SGD для решения полиномиальной или многомерной линейной регрессии:
   1.1. С разными размерами батча
   1.2. С разной функцией изменения шага и регуляризацией
   1.3. С библиотечными средствами (keras.optimize, torch.optim)
2. Сделать свою реализацию модификации SGD
3. Решить задачу из области ML с помощью SGD

Программная реализация

В рамках пунктов 1.1 и 1.2 был создан класс, реализующий метод стохастического градиентного спуска для задачи полиномиальной регрессии. При инициализации требуется/разрешается указать:

1. Размерность входных данных
2. Степень полиномиальной регрессии
3. Размер батча
4. Регуляризацию
5. Функцию изменения шага
6. Количество итераций
7. Ограничение на модуль градиента
8. ε

Для некоторых из этих параметров реализованы часто применяемые их вариации:

1. Для функций изменения шага: fixed, decay, sqrt_decay, exponential, reset_exp
2. Для функций потери (используются в регуляризации): L1, L2
3. Для регуляризаций: mean (mean + L1 = MAE, mean + L2 = MSE) и elastic net (MSE + L1 penalty + L2 penalty)

В рамках пункта 1.3 используются аналоги следующих методов в библиотеках keras и torch:

1. SGD (классический)
2. SGD с импульсом
3. SGD с импульсом Нестерова
4. AdaGrad
5. RMSProp
6. Adam

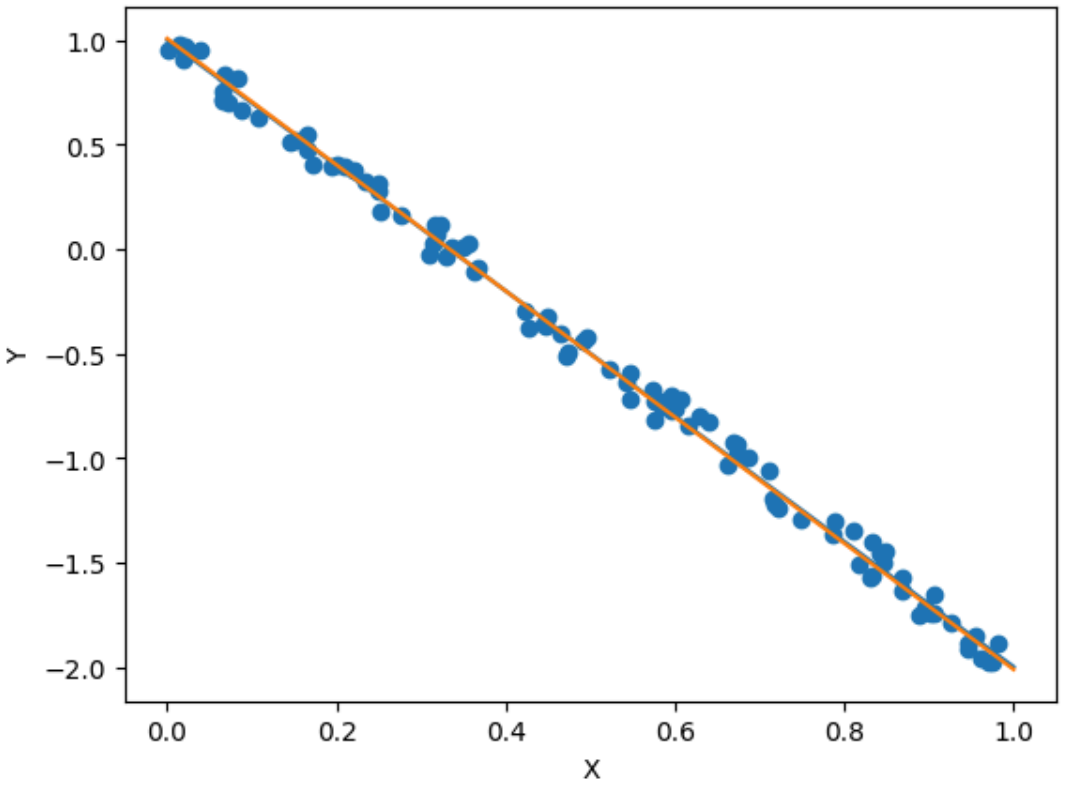
В рамках дополнительного задания 1 реализованы две модификации классического градиентного спуска: с импульсом и с импульсом Нестерова.

Проверка работоспособности

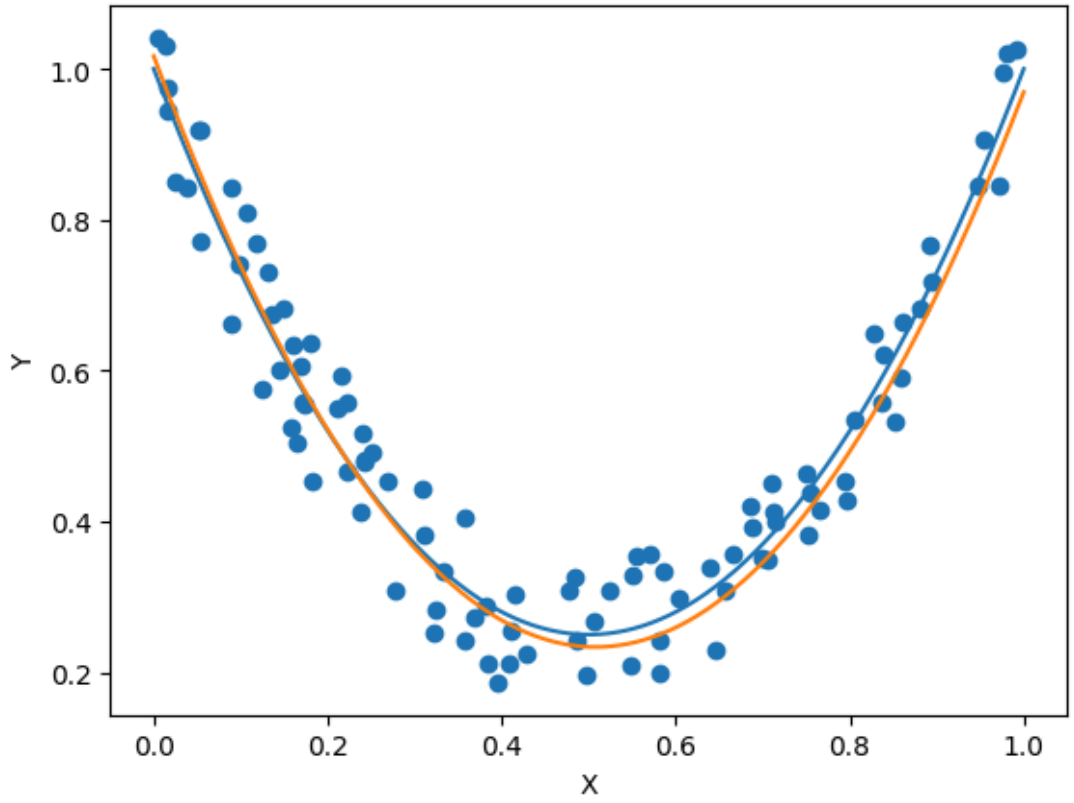
Рассмотрим одномерный случай - имеется набор точек разбросанных неподалеку от заданной кривой. Хотим получить полиномиальное приближение, минимизирующее MSE.

```
#f = lambda x: -3 * x + 1
#f = lambda x: 3 * x**2 - 3 * x + 1
f = lambda x: 16 * x**3 - 24 * x**2 + 9 * x + 1
dots, input, output = init_plane_data(f)
sgd = StochasticGradientDescent(1, 3, 3,
Regularizers.mean(LossFunctions.L2), Schedulers.fixed(0.5), 100000)
# Количество требуемых эпох сильно зависит от степени полинома
weights = sgd.run(input, output)
print(poly1d_to_string(weights))
```

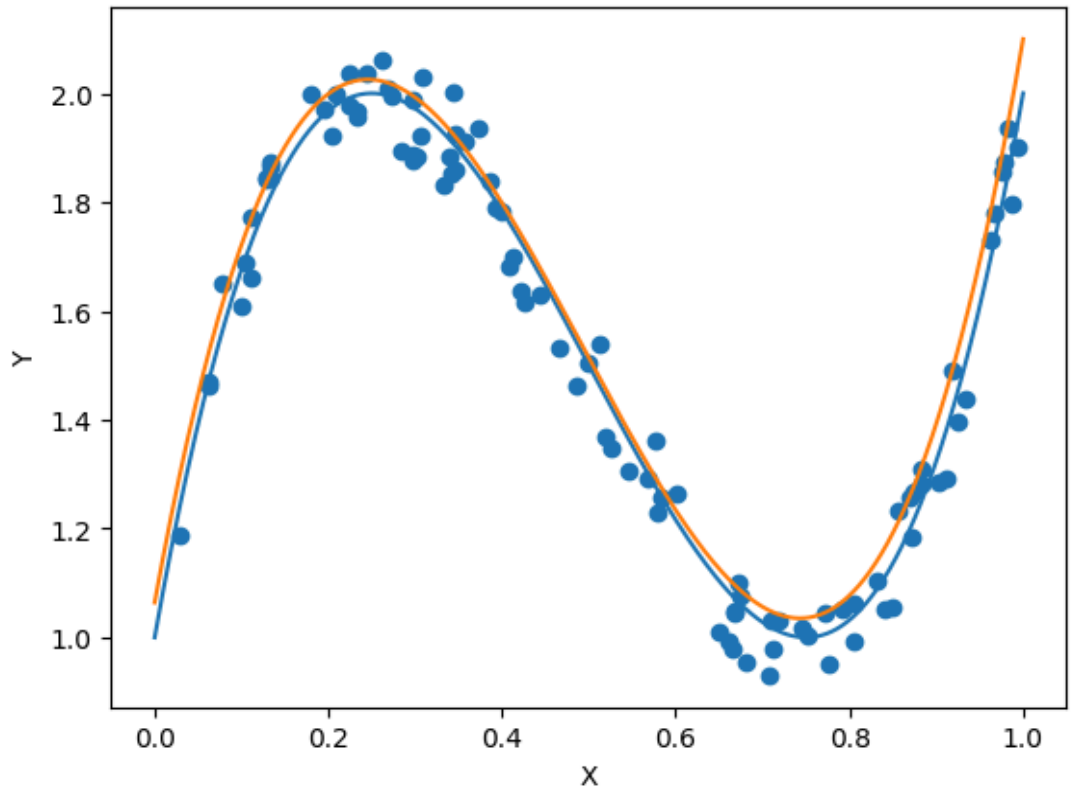
$$f = -3x + 1 \Leftrightarrow f_{calc} = -3.023x + 1.017$$



$$f = 3x^2 - 3x + 1 \Leftrightarrow f_{calc} = 3.037x^2 - 3.085x + 1.017$$



$$f = 16x^3 - 24x^2 + 9x + 1 \Leftrightarrow f_{calc} = 16.075x^3 - 23.838x^2 + 8.798x + 1.064$$



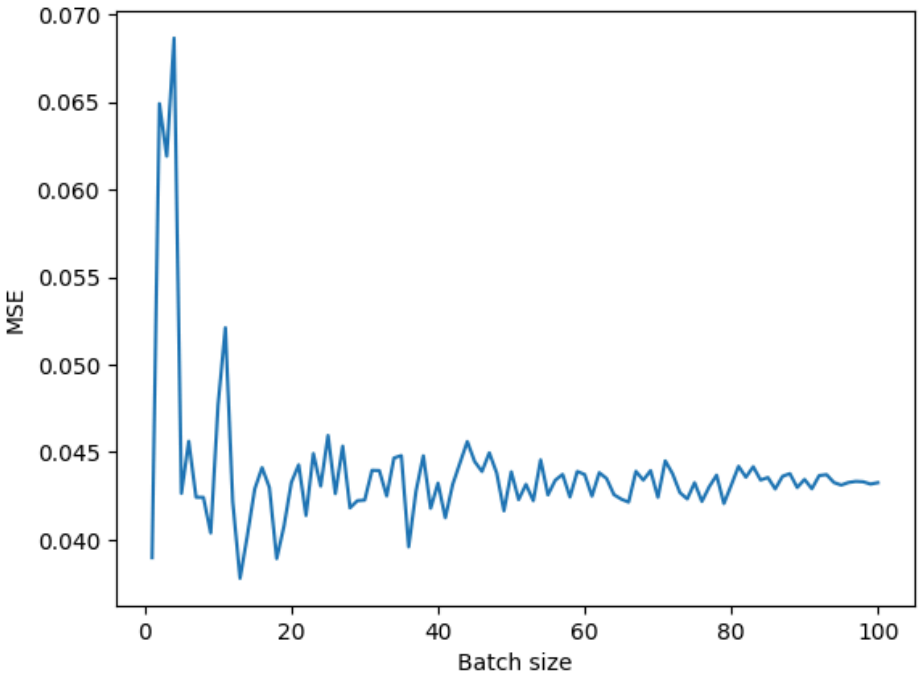
Многомерное применение

Рассмотрим использование средств лабораторной работы на данных реального датасета: Wine Quality. Датасет имеет 11 входных параметров состава вина и один исходящий – качество вина по десятибалльной шкале (на самом деле все оценки находятся в диапазоне от 3 до 8). Для решения этой задачи воспользуемся лучше оптимизированными библиотечными методами.
Так, с помощью достаточного количества итераций метода Adam на модели библиотеки PyTorch, аналогичной линейной регрессии, получаем следующее распределение угаданных образцов:

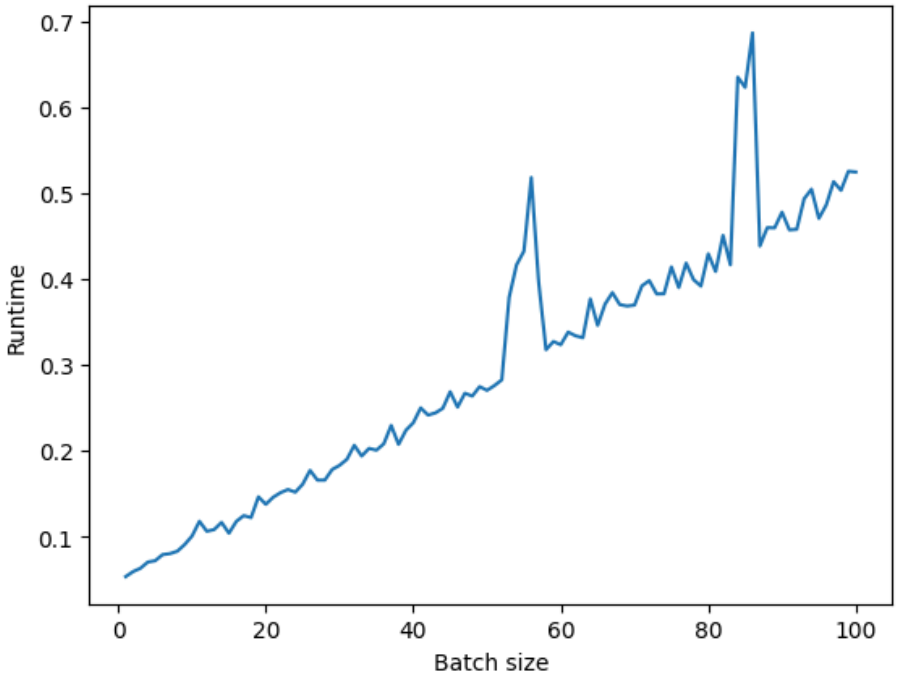
То есть более чем трети винам присвоена верная оценка, а более 87% присвоена оценка, отличающаяся не более, чем на единицу.

Исследование реализации

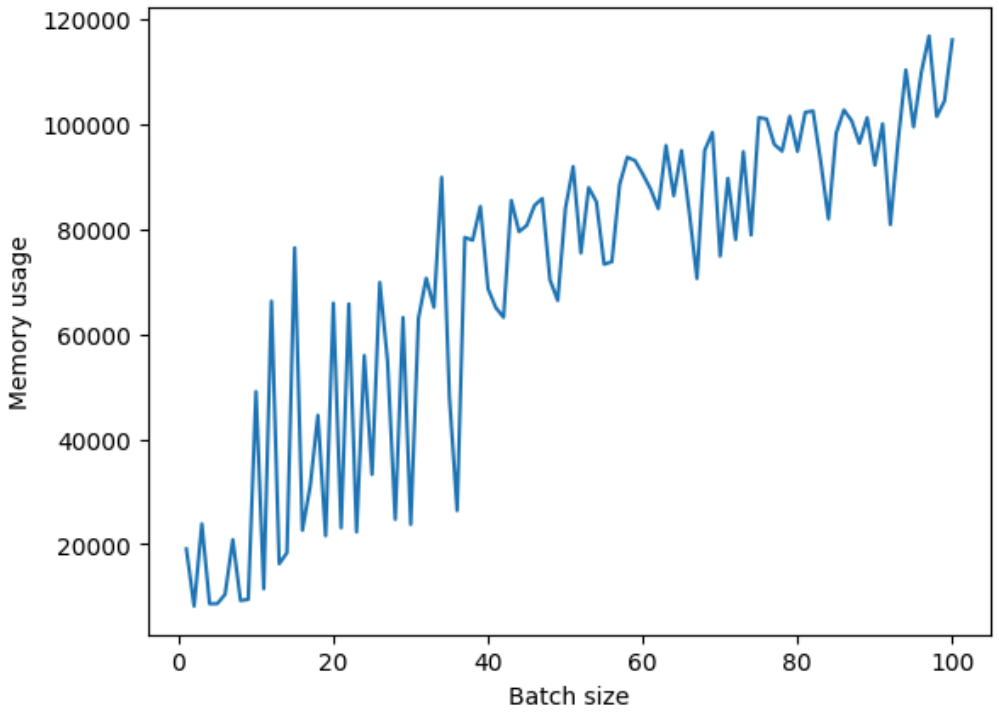
Будем сравнивать эффективность нашей реализации на одномерном случае.



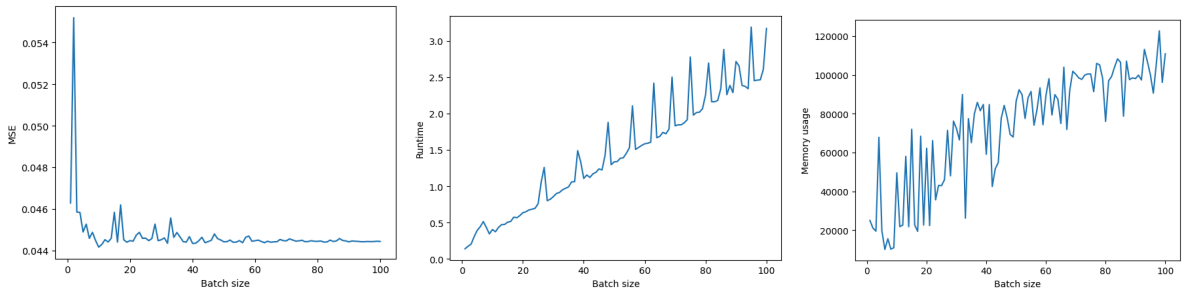
При малых размерах батча не всегда удается достичь хорошего приближения. При увеличении размера батча отклонение стабилизируется вокруг значения, зависящего от количества эпох.



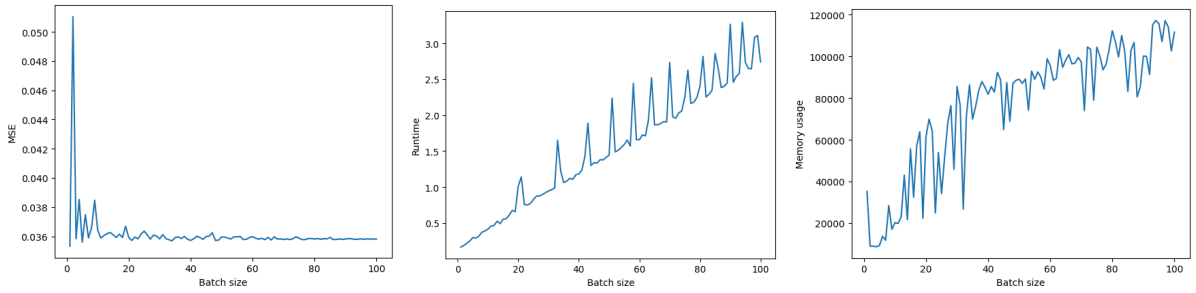
При увеличении размера батча время обработки увеличивается линейно. Возможны артефакты, связанные с неудачно выбранным направлением движения.



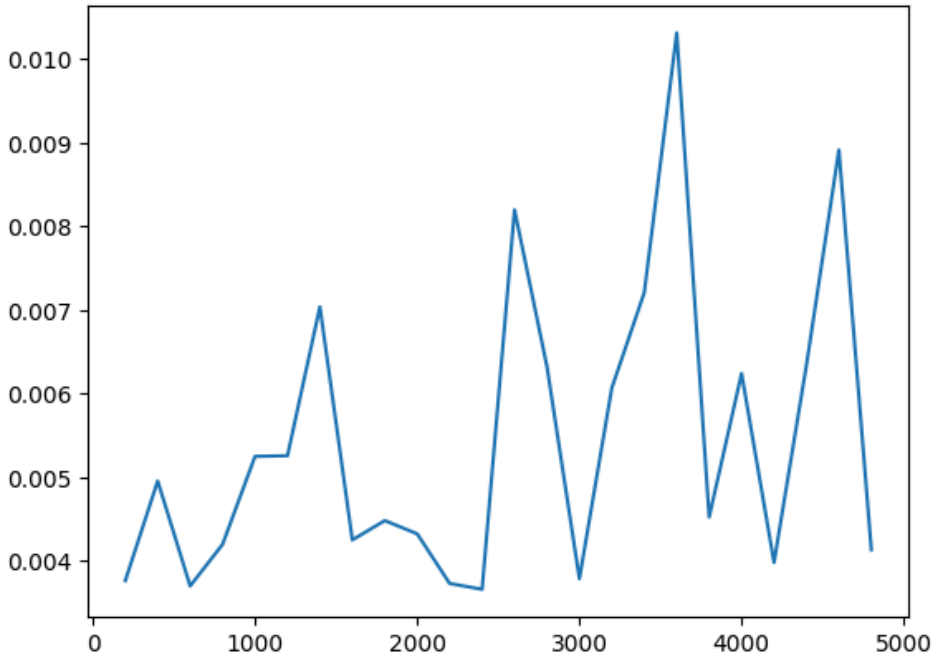
Количество требуемой оперативной памяти для вычислений также возрастает линейно.



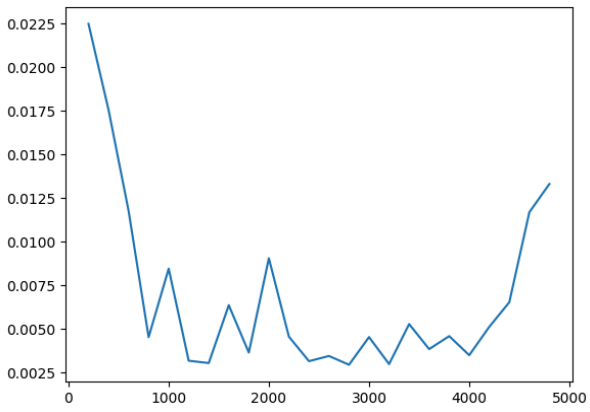
Эти выводы ещё более явно выражены на примере квадратичной и кубической регрессии.

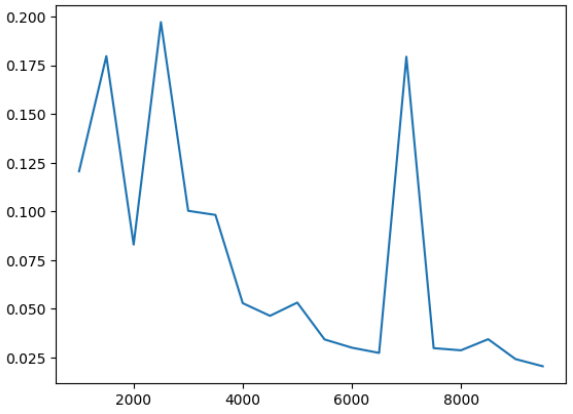


Зависимость эффективности от количества эпох



В линейном примере не прослеживается зависимости между итоговым среднеквадратическим отклонением и количеством эпох.

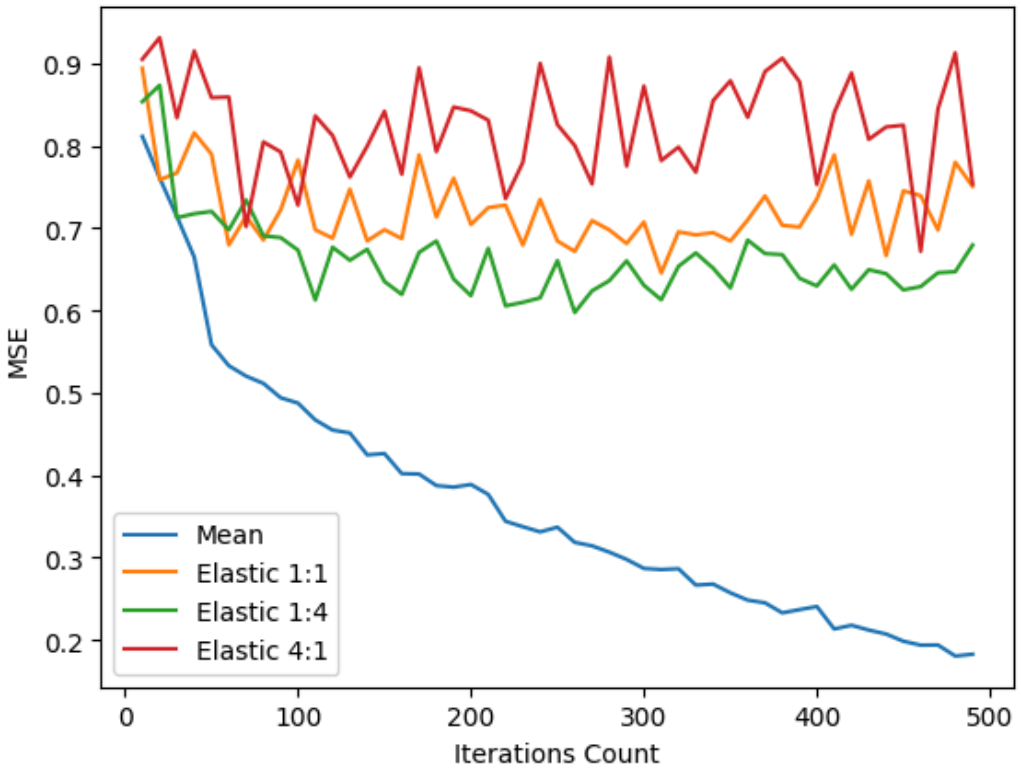




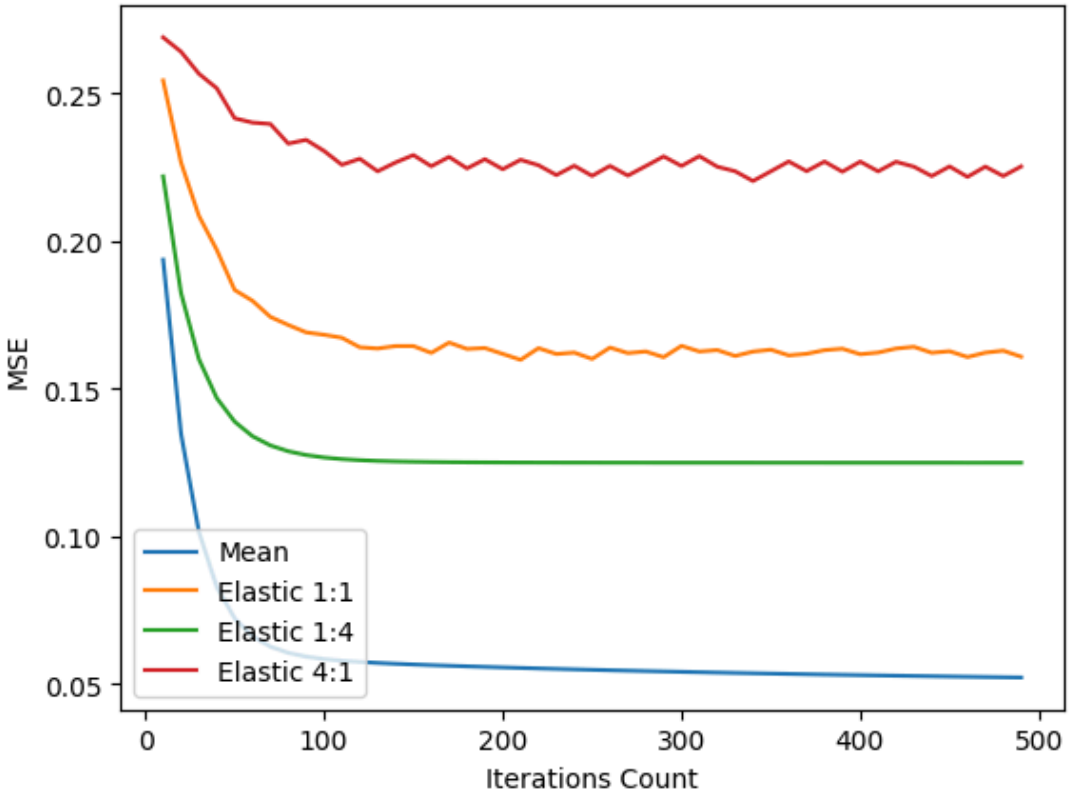
Однако для полиномов высших степеней для хорошей сходимости требуется увеличивать количество эпох.

Сравнение гиперпараметров

До этого момента мы везде использовали регуляризацию MSE в качестве минимизируемой функции. Рассмотрим так называемую Elastic Net регуляризацию. В ней присутствуют две дополнительные суммы, зависящие от L1 и L2 норм весов.

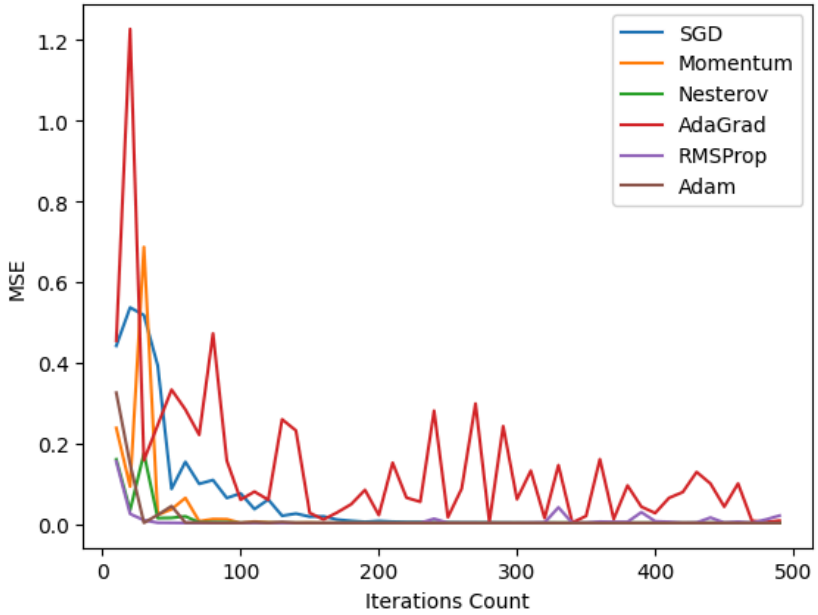


В рамках этой лабораторной на простых функциях Elastic Net регуляризация показывает менее убедительные результаты, чем прямолинейное MSE. Как с маленьким размером батча (сверху), так и с большим размером батча (снизу), не удаётся улучшить результат.

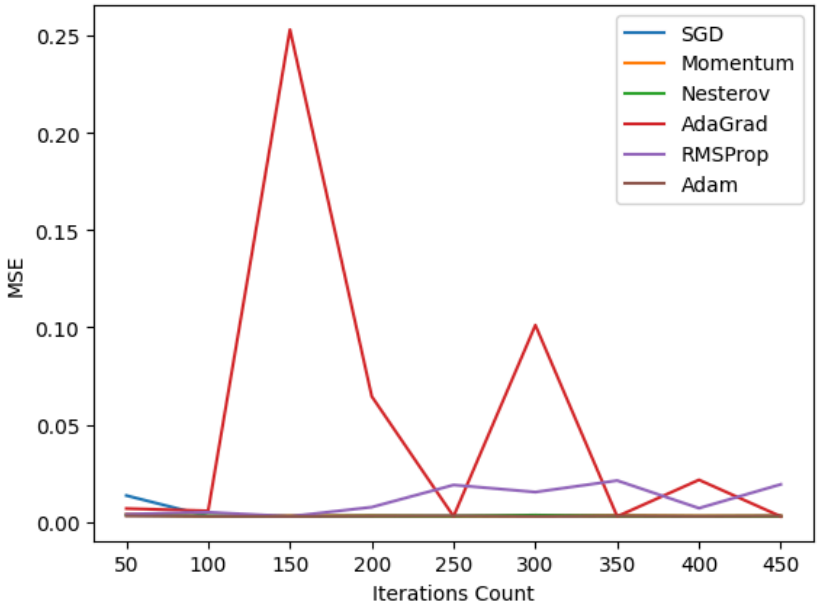


Библиотечные модификации SGD

Оба пакета (keras и torch), предоставляют схожий набор вариаций методов стохастической оптимизации.



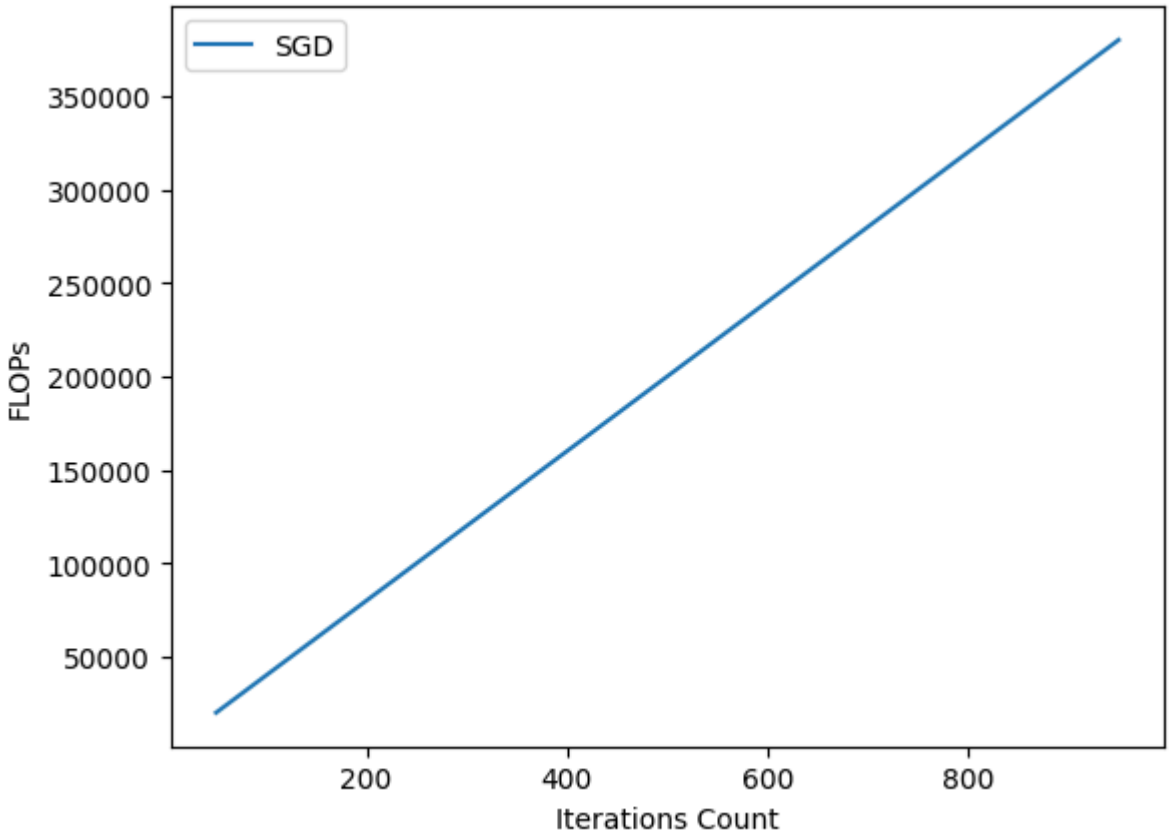
Методы пакета torch, помимо метода AdaGrad, показывают хорошую скорость сходимости и предсказуемо стремятся к минимуму.



Аналогичные методы пакета keras выполняются значительно дольше, но в среднем дают более близкий результат.

Подсчёт операций с плавающей точкой

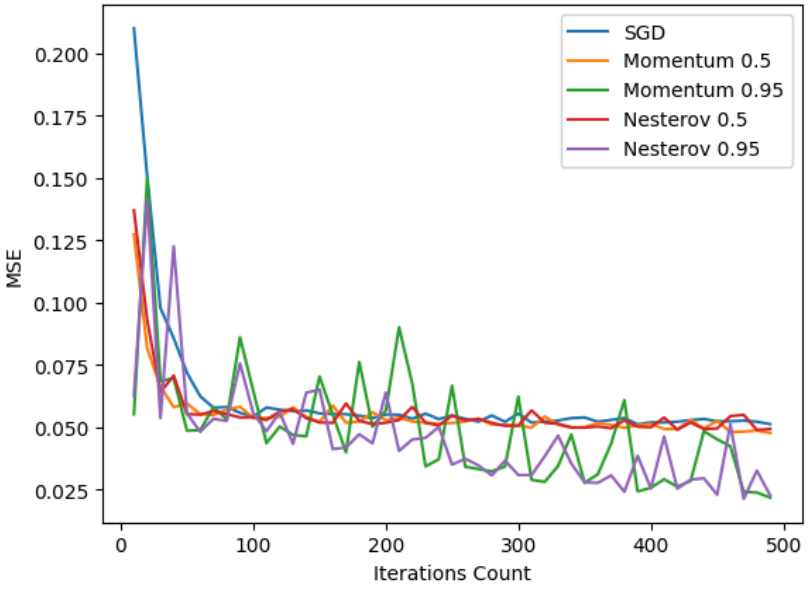
Библиотека PyTorch обладает профайлером для анализа работы моделей. В частности, профайлер может оценивать количество операций с плавающей точкой, совершённых моделью.



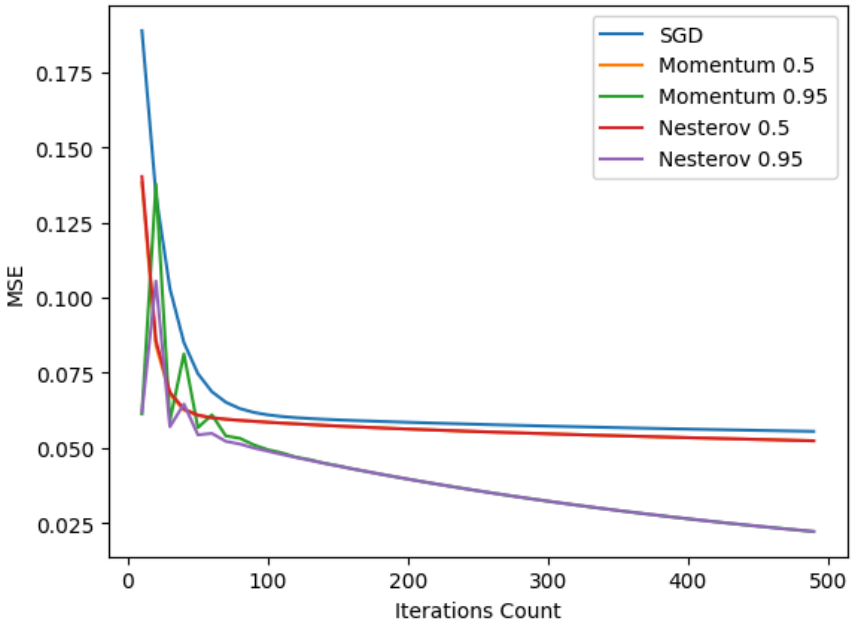
Оценённое количество операций в точности линейно зависит от количества итераций. Так как на каждом шаге мы обрабатываем batch определённого размера, то количество FLOPs также будет зависеть и от него.

Модификации SGD (доп. задание 1)

От класса StochasticGradientDescent наследуются две вариации: MomentumSGD и NesterovSGD. Эти вариации сохраняют часть инерции, чтобы, возможно, найти лучший путь к минимуму.



Видно, что вариации с высоким коэффициентом сохранения моментума ведут себя намного более хаотично, чем прочие варианты. Однако также можно заметить, что благодаря этому часто удается добиться сходимости к лучшему минимуму.



Заметно, что при больших размерах батча (на нижней картинке - весь датасет), методы с высокой инерцией становятся не такими хаотичными, однако всё так же стремятся к минимуму значительно быстрее.