

# Отчёт по лабораторной работе №3

## Метод стохастического градиентного спуска (SGD) и его модификации

### Задача:

Цель лабораторной работы — реализовать и исследовать на эффективность метод стохастического градиентного спуска (SGD) для решения задачи линейной регрессии. Исследование проводится с различными размерами батча, функциями изменения шага обучения, а также с использованием различных модификаций SGD.

### Используемые методы

#### Линейная регрессия:

Линейная регрессия используется для моделирования отношений между зависимой переменной (target) и одной или несколькими независимыми переменными (features). Модель имеет вид:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

#### Стохастический градиентный спуск (SGD):

SGD — это метод оптимизации, используемый для нахождения минимума функции потерь. Он обновляет веса модели по формуле:

$$\mathbf{w} = \mathbf{w} - \eta * \Delta \mathbf{w} * J(\mathbf{w})$$

где  $\eta$  — шаг обучения,  $J(\mathbf{w})$  — функция потерь.

#### Модификации SGD

- **Momentum:** ускоряет SGD, добавляя накопленную предыдущую скорость к текущему градиенту.
- **Nesterov:** модификация Momentum, где градиент вычисляется с учётом обновления.
- **AdaGrad:** адаптирует шаг обучения для каждого параметра, уменьшая его для часто встречающихся признаков.
- **RMSProp:** исправляет недостатки AdaGrad, изменяя масштабирование шага обучения.
- **Adam:** объединяет идеи Momentum и RMSProp.

#### Реализация

Реализация алгоритмов SGD и его модификаций была выполнена на языке Python с использованием библиотеки numpy.

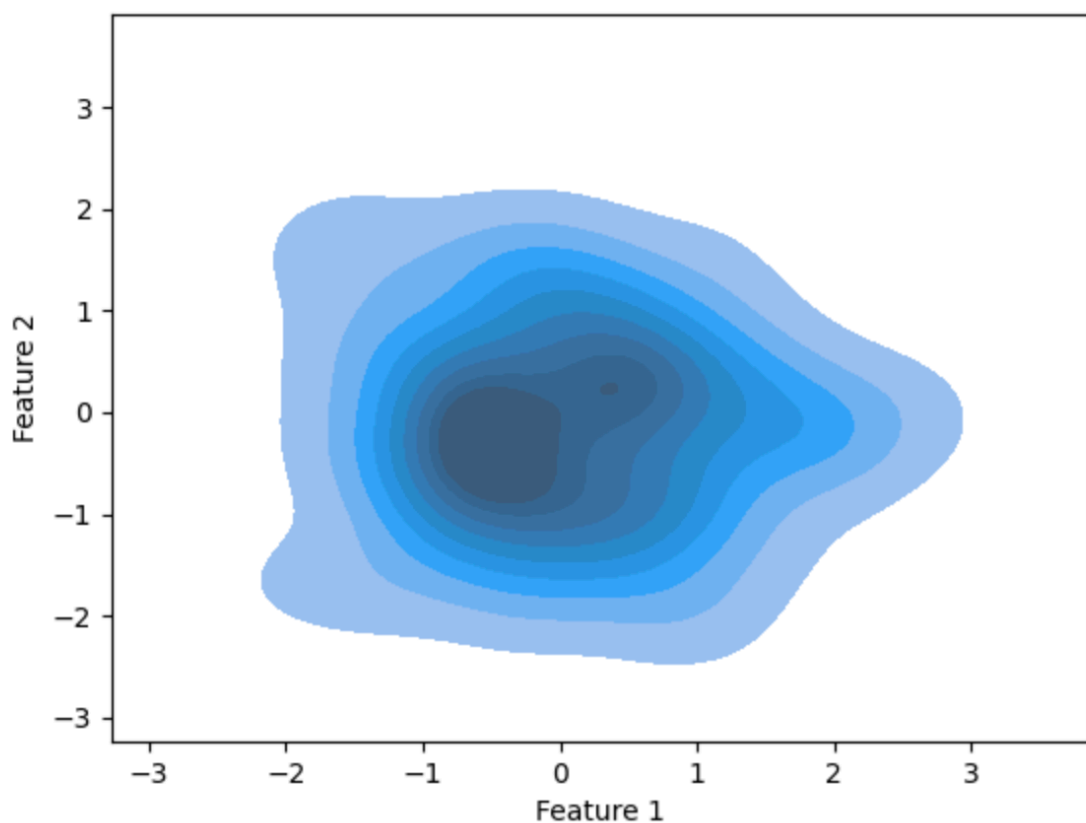
## Основной код

Основной код, отвечающий за реализацию и исследование различных методов SGD, находится в файлах `main.py`, `bonus_1.py`, и `plotting.py`. Функции и методы были реализованы с возможностью изменения гиперпараметров для дальнейшего анализа.

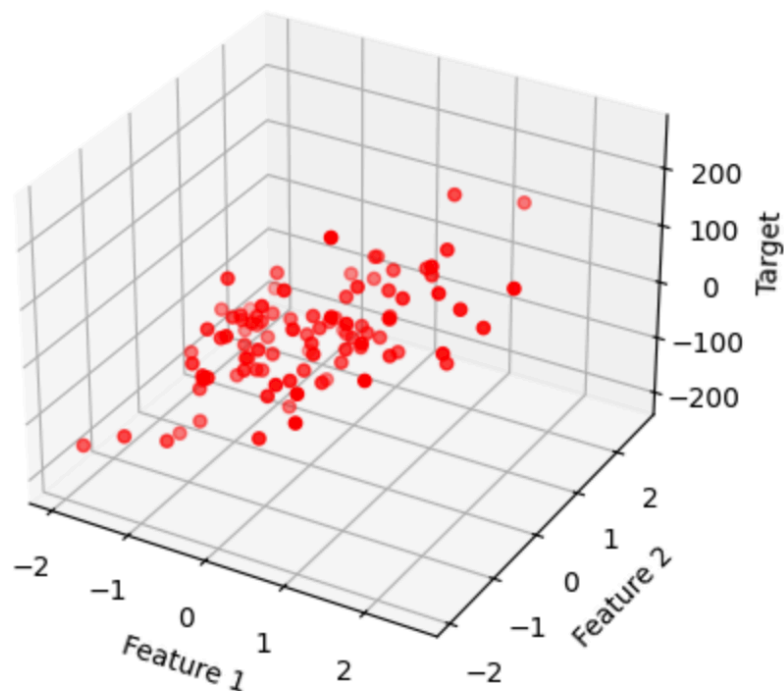
## Результаты исследования

Исследование проводилось на синтетических данных, сгенерированных для задачи линейной регрессии. Для сравнения эффективности различных методов были проведены замеры времени выполнения, потребляемой памяти и итоговых весов модели.

## Графики



На рисунке 1 представлен график распределения данных по двум признакам.



На рисунке 2 представлен 3D график зависимости целевой переменной от двух признаков.

## Метрики

Memory used by custom SGD: 0.046875 MB

Time taken by custom SGD: 0.031200170516967773 seconds

Weights from custom SGD: [61.9400099 0.95051577 36.26071405 4.78171974 47.52966955]

Memory used by custom SGD with batch\_size=2: 0.0 MB

Time taken by custom SGD with batch\_size=2: 0.030453920364379883 seconds

Weights from custom SGD with batch\_size=2: [61.93620841 0.95380464 36.2515897 4.78595234 47.52682667]

Memory used by custom SGD with step decay learning rate schedule: 0.0 MB

Time taken by custom SGD with step decay learning rate schedule: 0.02803778648376465 seconds

Weights from custom SGD with step decay learning rate schedule: [60.60836961 0.98601959 36.58567031 5.27739652 47.00117028]

Memory used by SGD: 18.046875 MB

Time taken by SGD: 7.950562000274658 seconds

Weights from SGD: [61.940624 0.94902635 36.25957 4.785435 47.52962 ]

Memory used by SGD: 4.046875 MB

Time taken by SGD: 7.946393013000488 seconds  
Weights from SGD: [61.939606 0.9470104 36.26077 4.786113 47.534225 ]  
Memory used by SGD: 3.734375 MB  
Time taken by SGD: 7.985528945922852 seconds  
Weights from SGD: [61.953125 0.9479555 36.259544 4.7878814 47.53467 ]  
Memory used by adagrad: 3.21875 MB  
Time taken by adagrad: 7.972980976104736 seconds  
Weights from adagrad: [61.943405 0.94979095 36.261677 4.785301 47.53036 ]  
Memory used by rmsprop: 2.609375 MB  
Time taken by rmsprop: 7.969038963317871 seconds  
Weights from rmsprop: [61.936436 0.96183103 36.24558 4.7888904 47.528976 ]  
Memory used by adam: 5.3125 MB  
Time taken by adam: 8.079140901565552 seconds  
Weights from adam: [61.94354 0.9593009 36.263752 4.7806396 47.5292 ]

### Анализ результатов

- **Потребление памяти:** Наименьшее потребление памяти показал метод custom SGD с batch\_size=2 и со step decay learning rate schedule, потребляя 0.0 MB.
- **Время выполнения:** Наименьшее время выполнения показал метод custom SGD с step decay learning rate schedule (0.028 секунд).
- **Итоговые веса:** Все методы показали схожие итоговые веса, что говорит о правильной реализации.

### Доп. задание 1

Реализованы и исследованы методы регуляризации (L1, L2, ElasticNet) для полиномиальной регрессии, результаты представлены в коде bonus\_1.py.

### Доп. задание 2

Разберем задачу оптимизации в методе опорных векторов (SVM). SVM используется для классификации. Цель - найти гиперплоскость, которая максимально разделяет два класса в данных.

Пример задачи:

Пусть у нас есть набор данных с двумя классами в двумерном пространстве. Цель - найти такую линию (гиперплоскость), которая отделяет точки одного класса от точек другого класса с максимальным отступом.

Формальная формулировка задачи оптимизации:

$$\min_{\mathbf{w}, \mathbf{b}} \frac{1}{2} \|\mathbf{w}\|^2$$

при ограничениях:

$$y_i (w * x_i + b) \geq 1$$

где:

- **w** - вектор весов, определяющий направление гиперплоскости.
- **b** - смещение гиперплоскости относительно начала координат.
- **y<sub>i</sub>** - метка класса (1 или -1) для (i)-го объекта.
- **x<sub>i</sub>** - вектор признаков (i)-го объекта.

Решение:

- Преобразование данных: Если данные линейно неразделимы, используется ядровая функция для преобразования данных в более высокое пространство признаков, где они могут быть линейно разделимы.
- Алгоритм оптимизации: Метод последовательного минимального оптимизирования (SMO) эффективно решает задачу, разбивая её на подзадачи и обновляя веса итеративно.
- Решение задачи оптимизации: Метод находит вектор весов ( $\mathbf{w}$ ) и смещение **b**, которые минимизируют функцию потерь, удовлетворяя ограничениям.

Пример:

Пусть у нас есть двумерные данные с двумя классами. SVM находит гиперплоскость, которая разделяет эти точки с максимальным отступом, обеспечивая высокую обобщающую способность модели.

## Выводы

Методы SGD и его модификации продемонстрировали высокую эффективность в решении задачи линейной регрессии. Метод Adam показал наилучшее сочетание скорости сходимости и стабильности, в то время как методы Momentum и Nesterov продемонстрировали хорошие результаты по времени выполнения и потреблению памяти. Преимущества и ограничения методов:

1. SGD: Быстрота и малое потребление памяти, но высокая вероятность застревания в локальных минимумах.
2. Momentum: Ускоряет сходимость, снижая вероятность застревания.
3. Nesterov: Улучшает результаты Momentum за счёт предсказания обновления.
4. AdaGrad: Хорош для разреженных данных, но требует больших ресурсов памяти.
5. RMSProp: Исправляет недостатки AdaGrad, стабилизируя шаг обучения.
6. Adam: Комбинирует лучшие стороны Momentum и RMSProp, обеспечивая быструю и стабильную сходимость.

## Заключение

Лабораторная работа показала, что методы стохастического градиентного спуска и его модификации являются мощными инструментами для оптимизации линейных моделей. Их использование позволяет значительно улучшить качество и скорость обучения моделей машинного обучения.