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

Методы оптимизации

Отчет по лабораторной работе №2

Работу выполнил:

Лаптев Александр М34371

Санкт-Петербург
2023

1 Постановка задачи

1. Реализуйте стохастический градиентный спуск для решения линейной регрессии. И

   Исследуйте сходимость с разным размером батча

   - 1 - SGD
   - $2 \dots n-1$ Minibatch GD
   - n - GD из предыдущей работы

2. Подберите функцию изменения шага (learning rate scheduling), чтобы улучшить сходимость, например экспоненциальную или ступенчатую.

3. Исследуйте модификации градиентного спуска

   - Nesterov
   - Momentum
   - AdaGrad
   - RMSProp
   - Adam

4. Исследуйте сходимость алгоритмов.

   Сравнить различные методы по скорости сходимости, надежности, требуемым машинным ресурсам (объем оперативной памяти, количеству арифметических операций, времени выполнения)

5. Постройте траекторию спуска различных алгоритмов из одной и той же исходной точки с одинаковой точностью.

   В отчете наложить эту траекторию нарисунок с линиями равного уровня заданной функции.

6. Реализуйте полиномиальную регрессию.

   Постройте графики восстановленной регрессии для полиномов разной степени.

7. Модифицируйте полиномиальную регрессию добавлением регуляризации в модель (L1, L2, Elastic регуляризации).

8. Исследуйте влияние регуляризации на восстановление регрессии.

2 Стохастический и батчевый градиентный спуск

Рассмотрим функцию вида $f(x) = \sum_{i=1}^{m} f_i(x)$ и параметр BS - размер батча. На каждой итерации градиентного спуска вместо вычисления градиента функции $f(x)$ вычислим градиент функции

$f'(x) = \sum_{i \in I} f_i(x)$, где I - случайное подмножество $\{1, 2 \dots m\}$ на текущей итерации, размер которого равен BS.

После воспользуемся градиентом функции $f'(x)$ для вычисления точки x_k, вместо градиента функции $f(x)$.

Такая разновидность градиентного спуска называется батчевым градиентным спуском с размером батча BS. Если $BS = 1$, метод называют стохастическим градиентным спуском.

2.1 Линейная регрессия

Пусть

$X = (x_1, x_2 \dots x_m)$ - набор векторов размерности n.

$Y = (y_1, y_2 \dots y_m)$ - набор чисел.

Задача линейной регрессии заключается в нахождении такого вектора w размерности n и числа b, минимизирующей следующую функцию:

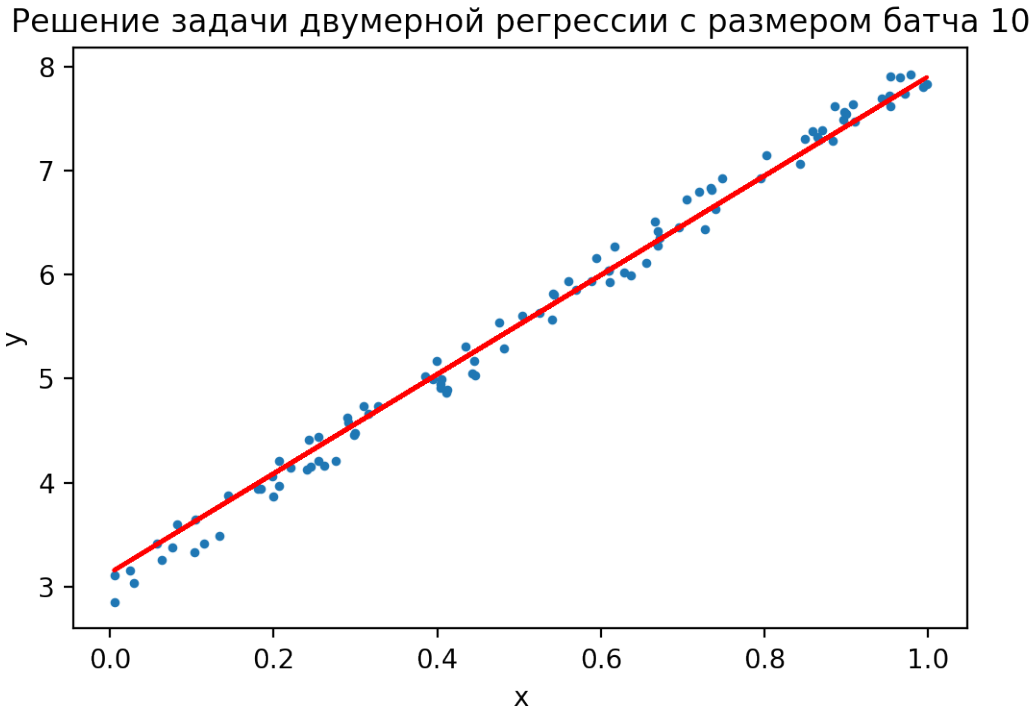
$L(w, b) = \sum_{i=1}^{m} (x_i w + b - y_i)^2$. Функцию L называют функцией ошибки.

Иными словами, нужно найти n-мерную плоскость наилучшим образом аппроксимирующую входные данные.

2.2 Применение градиентного спуска к решению задачи линейной регрессии.

Для решения задачи линейной регрессии, может быть применен батчевый градиентный спуск. В случае линейной регрессии, функция ошибки является суммой слагаемых вида $(x_i w + b - y_i)^2$, а значит можно применить батчевый градиентный спуск, где $f_i = (x_i w + b - y_i)^2$.

Пример решения одномерной линейной регрессии представлен ниже.



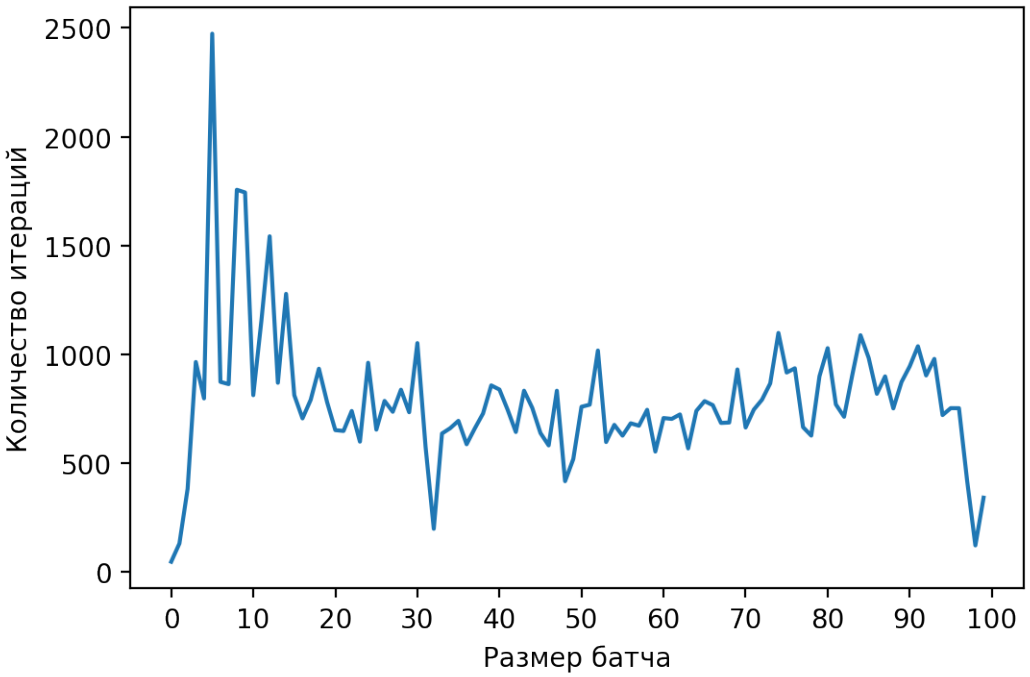
Исследуем сходимость градиентного спуска с шагом $\alpha = 0.01$ в зависимости от размера батча на примере задачи линейной регрессии, где $n = 20$ и $m = 100$.

Для этого рассмотрим функцию $g(x_1 \dots x_n) = 2.5 + \sum_{i=0}^{i=19} \frac{i}{10} x_i$ - зависимость которую, необходимо восстановить.

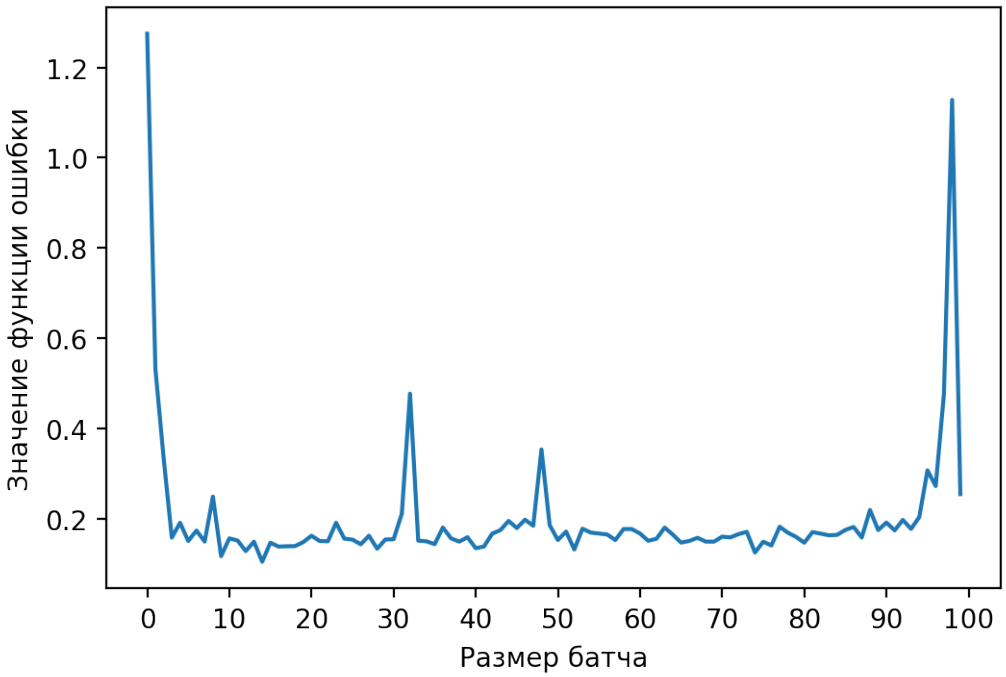
Сгенерируем случайный набор векторов $X = (x_1 \dots x_m)$, а набор чисел $Y = (y_1 \dots y_m)$, получим как $y_i = g(x_i) + \epsilon_i$, где ϵ_i - случайное числи из интервала $[-0.5, 0.5]$.

Далее применим метод градиентного спуска с размерами батчей от 1 до m для решения поставленной задачи, и исследуем зависимость значения функции ошибки и количество итераций необходимых для сходимости в зависимости от размера батча.

Зависимость количества итераций необходимых для сходимости в зависимости от размера батча:



Зависимость итоговой функции ошибки в зависимости от размера батча:



На основе полученных данных, можно сделать вывод, что наибольшее значение значение функции ошибки достигается при размере батча близком к минимально или максимально возможному (для размеров батчей близких к максимальному значению данный эффект может быть связан наличием шума ϵ в наборе чисел Y). Для промежуточных значений, значение ошибки практически одинаково, для всех размеров батчей (за исключением нескольких пиков в которых значение боль-

ше). При этом, начиная с размера батча равному 50 значение функции ошибки не возрастает, а количество итераций лежит в интервале от 500 до 1000 и не уменьшается с увеличением размера батча. То есть, выбрав размер батча равный 50, можно получить эквивалентное значение функции ошибки, что и для больших размеров батчей, при этом количество итераций останется тем же, а значит уменьшится время сходимости, так как обработка батча меньшего размера занимает меньше времени.

Таким образом, можно сформулировать вывод:

Батчевый градиентный спуск позволяет ускорить сходимость градиентного спуска, не потеряв при этом точности решения.

2.3 Градиентный спуск с функцией изменения шага

Рассмотрим стохастический градиентный спуск со ступенчатой функцией шага, а именно:

Значение шага на итерации it равняется $\alpha = max(\alpha_{min}, \alpha_{start} - it \cdot decay)$, α_{min} - минимальное значение шага, α_{start} - начальное значение шага, $decay$ - величина уменьшения шага на каждой итерации.

Данная модификация позволяет выбрать стартовое значение шага, большее чем в градиентном спуске с постоянным шагом, за счёт чего улучшить сходимость на первых итерациях, а также уменьшить значение шага на более поздних итерациях.

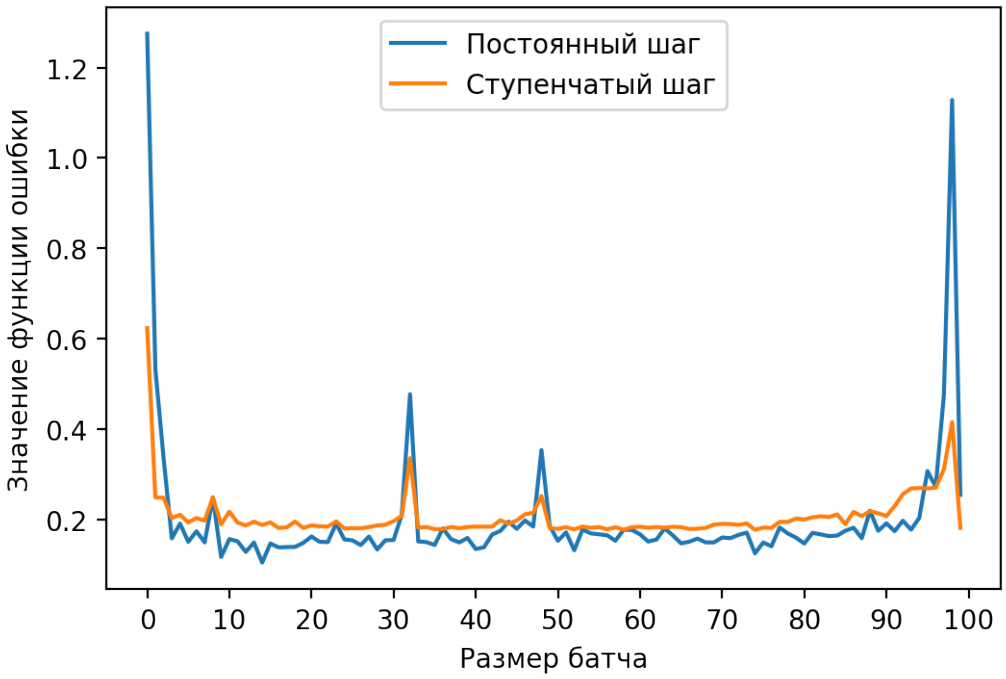
Исследуем итераций необходимых для сходимости стохастическому градиентному спуску со ступенчатой функций с параметрами:

- $\alpha_{min} = 0.001$
- $\alpha_{start} = 0.1$
- $decay = 0.001$

по сравнению с стохастическим градиентным спуском с постоянным шагом равным 0.01.

Сравним значение функции ошибки в зависимости от размера батча для градиентного спуска с постоянным шагом и для градиентного шага со ступенчатой функцией шага.

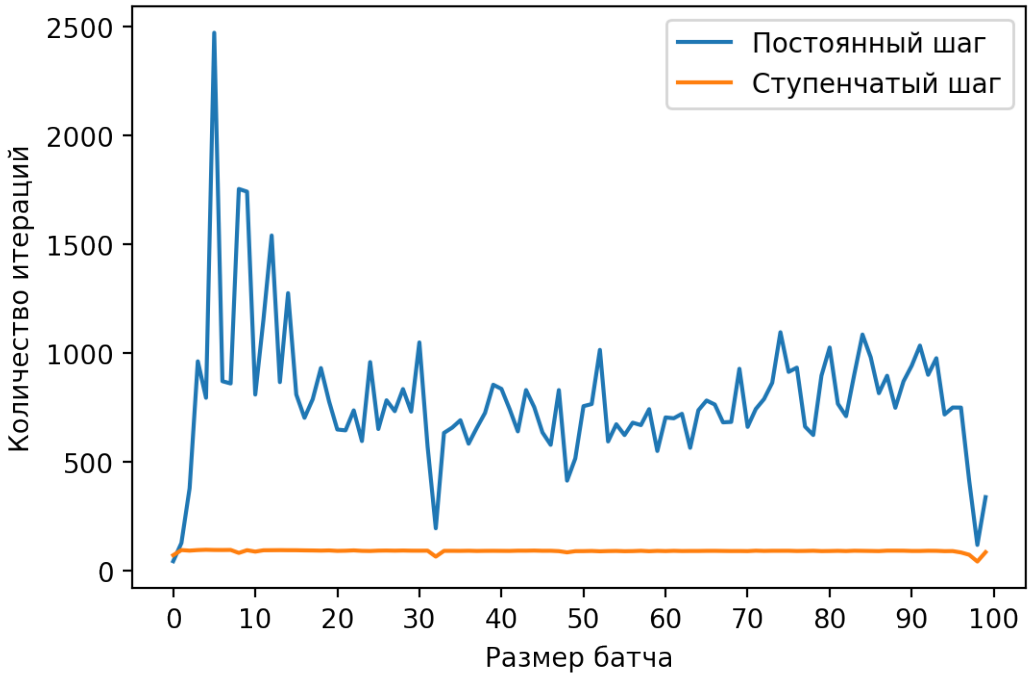
График зависимости функции ошибки от размера батча



Из полученных данных, можно сделать вывод, что значение функции ошибки для градиентного спуска со ступенчатым шагом близко к значению функции ошибки для градиентного спуска с постоянным шагом, для всех размеров батчей.

Рассмотрим количество итераций, необходимых для сходимости.

График зависимости количества итераций от размера батча:



Из полученных данных, можно сделать вывод, что количество итераций необходимых для сходимости градиентному спуску со ступенчатым шагом значительно меньше, чем количество итераций необходимых градиентному спуску с постоянным шагом.

Таким образом можно сделать вывод:

Применение функции изменения шага в градиентном спуске, позволяет уменьшить количество итераций необходимых для сходимости, не потеряв при этом точности решения.

3 Модификации градиентного спуска

Рассмотрим модификации градиентного спуска, направленные на улучшение его сходимости.

1. Momentum

   Данный метод использует экспоненциальное сглаживание для значения градиента, после сглаженный градиент используется для обновления точки минимума.

   $v_{i+1} = \gamma v_i + (1 - \gamma)\nabla f(w_i)$ - сглаженное значение градиента

   $w_{i+1} = w_i - \alpha v_{i+1}$ - обновление точки минимума

   Будем рассматривать данный метод с $\alpha = 0.065$ и $\gamma = 0.9$.

2. Nesterov

   Данный метод использует экспоненциальное сглаживание для значения градиента, но в отличие от Momentum значение градиента берется в точке, в которую градиентный спуск бы попал используя предыдущее значение градиента.

   $v_{i+1} = \gamma v_i + (1 - \gamma)\nabla f(w_i - \alpha\gamma v_i)$ - сглаженное значение градиента

$w_{i+1} = w_i - \alpha v_{i+1}$ - обновление точки

Будем рассматривать данный метод с $\alpha = 0.065$ и $\gamma = 0.9$.

3. AdaGrad

   Данный метод накапливает значение квадрата градиента, и делит шаг α на величину обратную корню накопленного значения, что позволяет обновлять с большим коэффицентом, значение тех переменные, по которым значение градиента мало.

   $G_{i+1} = G_i + \nabla f(w_i) \nabla f(w_i)^T$ - накопление квадрата градиента

   $w_{i+1} = w_i - \alpha \frac{\nabla f(w_i)}{\sqrt{diag(G_{i+1})}}$ - обновление точки минимума

   Будем рассматривать данный метод с $\alpha = 2$.

4. RMSProp

   Данный метод использует экспоненциальное сглаживание для квадрата градиента.

   Далее, шаг делится на корень из сглаженного квадрата градиента, аналогично AdaGrad.

   $s_{i+1} = \gamma s_i + (1 - \gamma) \nabla f(w_i)^2$ - накопление квадрата градиента

   $w_{i+1} = w_i - \alpha \frac{\nabla f(w_i)}{\sqrt{s_{i+1}}}$ - обновление точки минимума

   Будем рассматривать данный метод с $\alpha = 0.065$ и $\gamma = 0.99$.

5. Adam

   Данный метод использует экспоненциальное сглаживание для градиента и квадрата градиента и использует их для обновления объединяя идеи предыдущих алгоритмов.

   $v_{i+1} = \beta_1 v_i + (1 - \beta_1) \nabla f(w_i)$ - сглаженное значение градиента

   $s_{i+1} = \beta_2 s_i + (1 - \beta_2) \nabla f(w_i)^2$ - накопление квадрата градиента

   $v'_{i+1} = \frac{v_{i+1}}{1 - \beta_1^{i+1}}$

   $s'_{i+1} = \frac{s_{i+1}}{1 - \beta_2^{i+1}}$

   $w_{i+1} = w_i - \alpha \frac{v'_{i+1}}{\sqrt{s'_{i+1}}}$ - обновление точки минимума

   Будем рассматривать данный метод с $\alpha = 0.065$, $\beta_1 = 0.9$, $\beta_2 = 0.99$.

Сравним данные методы на примере функции

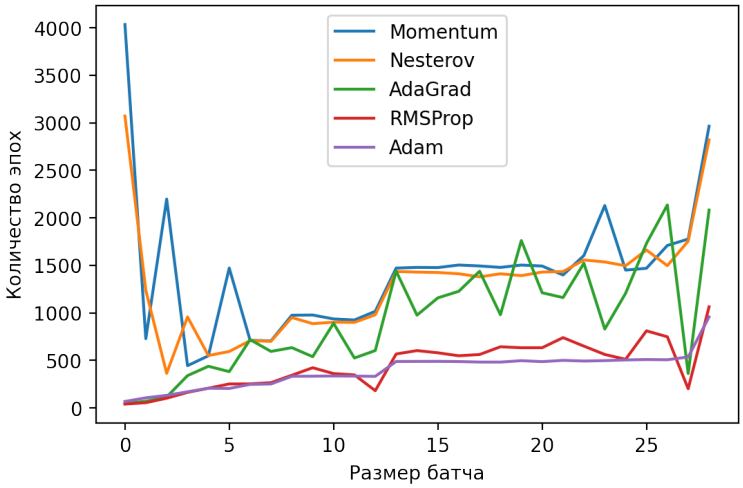
$f(x, y) = \sum_{i=1}^{i=19} \frac{1}{19}(1.5 - x + xy)^2 + \sum_{i=1}^{i=8} \frac{1}{8}(1.5 - x + xy^2)^2 + \sum_{i=1}^{i=3} \frac{1}{3}(1.5 - x + xy^3)^2$

с точкой минимума равной $(3, 0.5)$. Стартовую точку установим равной $(1.5, 2.3)$

3.1 Скорость сходимости

Исследуем скорость сходимости методов с точки зрения количества эпох, необходимых для сходимости в зависимости от размера батча. Для этого, запустим каждый из описанных методов на размере батча от 2 до 30 по 5 раз, полученные данные усредним.

График зависимости количества эпох до сходимости в зависимости от размера батча



На основе полученных данных, можно сделать вывод, что методы Momentum, Nesterov и AdaGrad требуют наибольшее количество эпох для сходимости, а RMSProp и Adam наименьшее количество эпох.

3.2 Надежность методов

Исследуем надежность методов с точки зрения достижения точки минимума в зависимости от выбора начальной точки. Для этого запустим каждый из описанных методов из начальных точек на квадратной сетке, с границами по x и y равными -3.5 и 3.5, размером 10 на 10. Для каждого метода посчитаем в скольких случаях итоговая точка будет лежать в окрестности точки минимума равной $(3, 0.5)$ с радиусом 0.1.

Результаты эксперимента приведены в таблице:

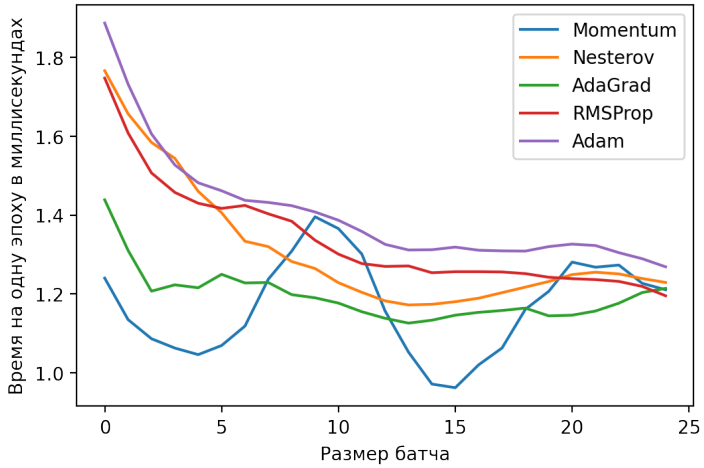
Метод	Momentum	Nesterov	AdaGrad	RMSProp	Adam
Количество итераций	55	57	63	68	67

На основе полученных данных, можно сделать вывод что самыми надежными методами являются Adam и RMSProp, а самыми ненадежными Momentum и Nesterov.

3.3 Время исполнения

Для оценки времени исполнения рассмотрим время необходимое на одну эпоху в зависимости от размера батча. Для этого, запустим каждый из описанных методов на размере батча от 1 до 30 по 5 раз, усредним данные по количеству эпох и общему количеству миллисекунд, а затем посчитаем отношение среднего количества миллисекунд к среднему количеству эпох.

График зависимости времени затраченного на одну эпоху зависимости от размера батча

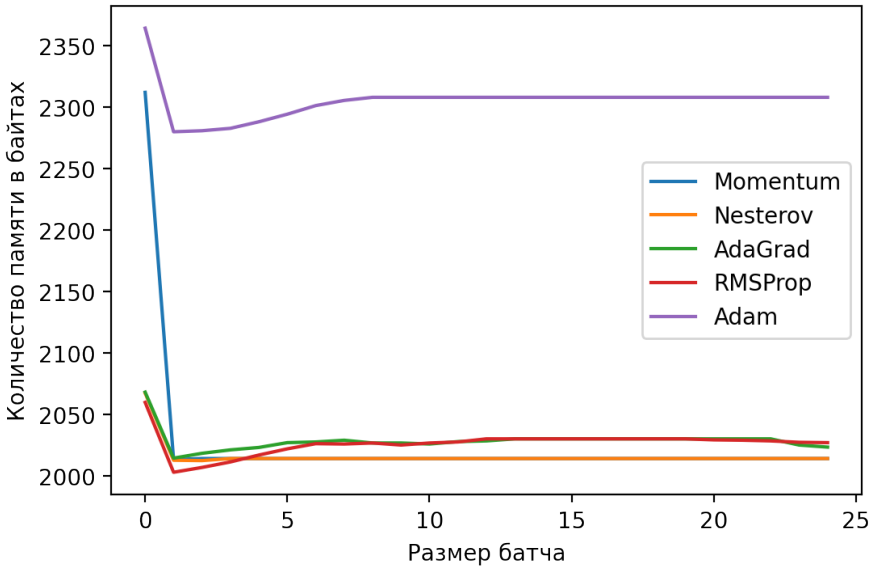


Из полученных данных, можно сделать вывод, что методы Adam, RMSProp и Nesterov требуют наибольшего количества миллисекунд на одну эпоху, а наименьшего методы Momentum и AdaGrad.

3.4 Количество памяти

Исследуем максимальное количество памяти необходимое для работы каждого из методов в зависимости от размера батча. Для этого, запустим каждый из описанных методов на размере батча от 2 до 30 по 5 раз, полученные данные усредним.

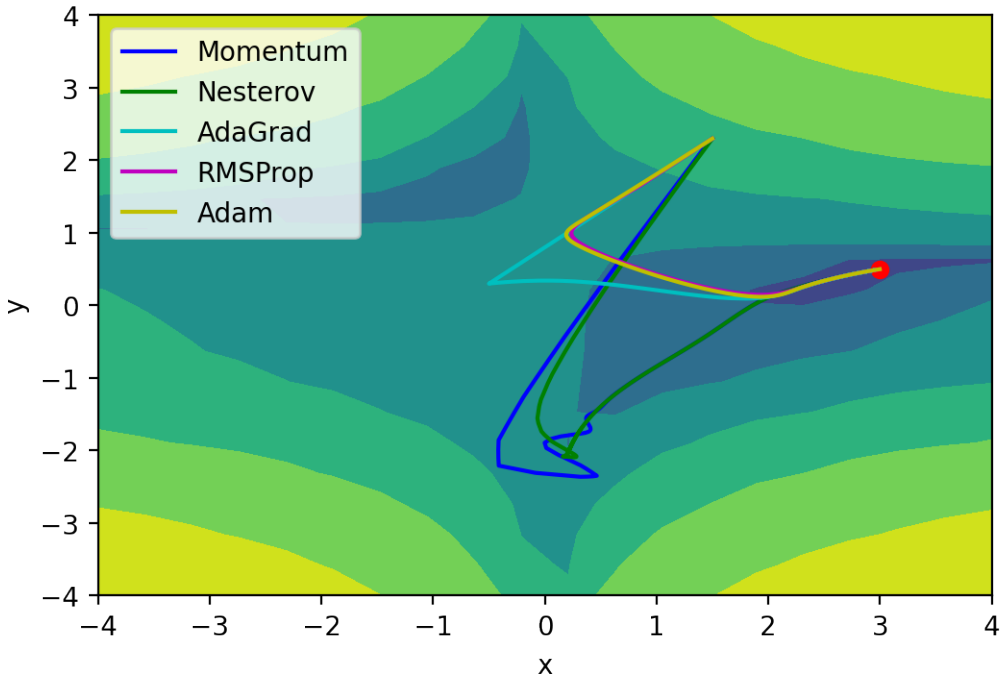
График зависимости количества памяти в зависимости от размера батча



Из полученных данных, можно сделать вывод, что количество выделенной памяти не зависит от размера батча. Наибольшее количество памяти понадобилось методу Adam, а наименьшее количество методам Nesterov и Momentum. Методы AdaGrad и RMSProp потребляют одинаковое количество памяти, незначительно большее по сравнению с методами Nesterov и Momentum.

3.5 Анализ траекторий

Рассмотрим траектории каждого из описанных методов с размером батча равным десяти.



На приведенном графике, траектории для методов Adam и RMSProp практически совпадают. Также, Nesterov и Momentum имеют схожую траекторию на начальных итерациях, но Nesterov быстрее изменяет траекторию по направлению к точке минимума. Adagrad, на первых итерациях делает шаги размера большего чем Adam и RMSProp, после сходится к той же траектории что и эти два метода.

3.6 Выводы

Сформилируем общие выводы по модификация градиентного спуска

- Методы Adam и RMSProp являются самыми надежными и быстросходящиеся методы, при это требующие большее количество машинных ресурсов.
- Метод Momentum является наиболее оптимальным по машинным ресурсам, при этом менее надеждным и быстросходящимся.
- Метод AdaGrad является методом имеющим среднии характеристики по всем оцениваемым параметрам.
- Метод Nesterov имеет надежность и скорость сходимости на уровне метода Momentum, при этом требует большего количества машинных ресурсов.

4 Полиноминальная регрессия

Пусть $X = (x_1, x_2 \dots x_m)$ - набор чисел, $Y = (y_1, y_2 \dots y_m)$ - набор чисел, d - максимально возможная степень полинома.

Задача полиноминальной регрессии заключается в нахождении полинома $f(x)$ степени не больше d, минизирующего следующую функцию:

$L(w) = \sum_{i=1}^{m}(f(x_i) - y_i)^2$. Функцию L называют функцией ошибки.

Иными словами, нужно найти полином, наилучшим образом аппроксимирующий входные данные.

4.1 Восстановление регрессии

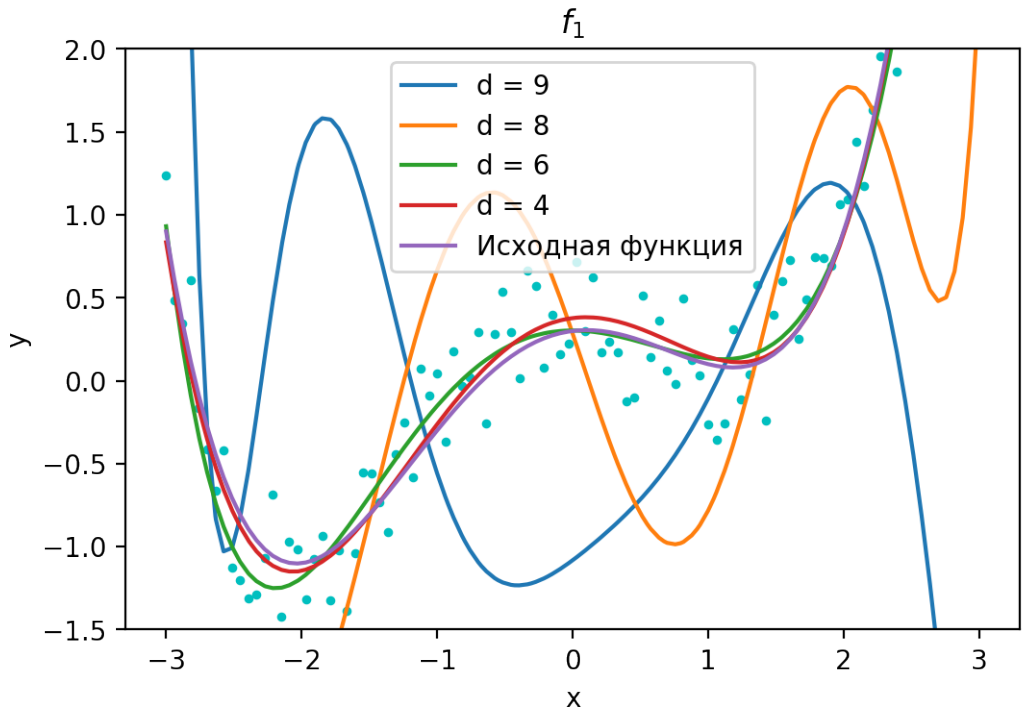
Исследуем восстановление регрессии на примере функции

1. $f(x) = 0.3 + 0.1x - 0.5x^2 + x^3 + x^4$

Для каждой функции сгенерируем случайный набор чисел $X = (x_1 \dots x_m)$ где $m = 50$, а набор чисел $Y = (y_1 \dots y_m)$, получим как $y_i = g(x_i) + \epsilon_i$, где ϵ_i - случайное число из интервала $[-0.25, 025]$.

Далее применим метод градиентного спуска с размером батча 10 и оптимизатором Adam с параметрами $\alpha = 0.03, \beta_1 = 0.9, \beta_2 = 0.9$ для решения поставленной задачи, и исследуем зависимость между восстановлением регрессии и максимальной степенью полинома d.

Рассмотрим восстановление функции f для максимальной степени полинома равной 9, 8, 6, 4.



На основе полученных данных, можно сделать вывод, что максимальных степенях полинома равных 4 и 6 исходная функция полностью восстановилась. При увеличении максимальной степени полинома, исходная зависимость перестала восстанавливаться.

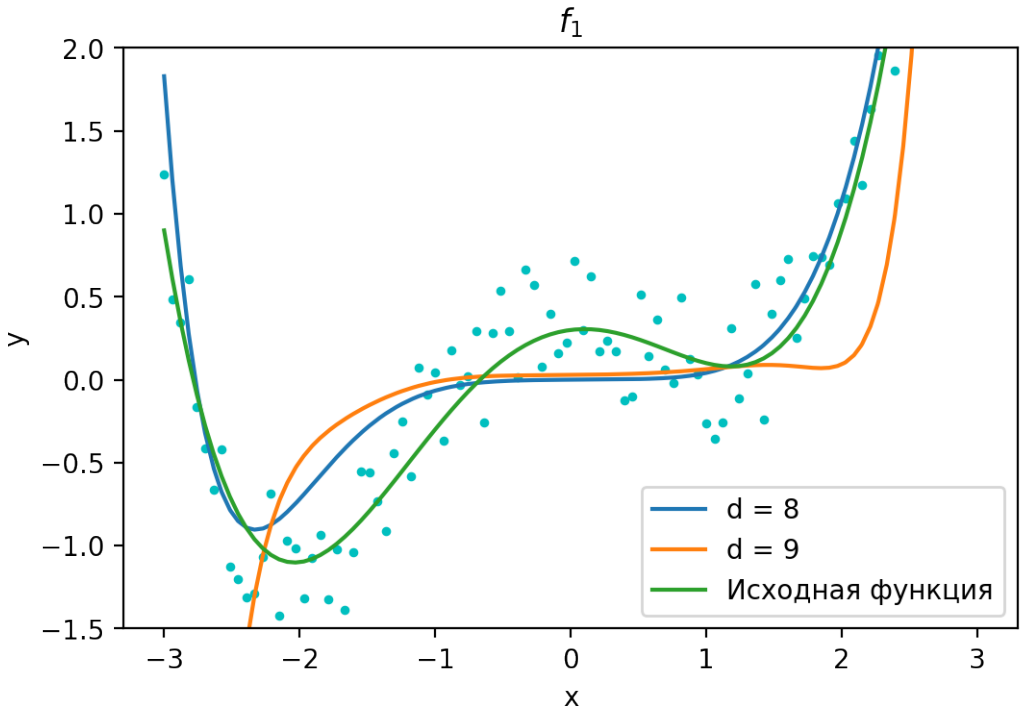
5 L_2 регуляризация

В условиях задачи полиномальной регрессии, добавим к функции ошибки регуляризацию, а именно:

$L_{reg}(w) = L(w) + \lambda||w||_2^2$ - новая функция для минимазации.

Добавление члена $\lambda||w||_2^2$ позволяет среди всех подходящих параметров искать параметры значение которых минимально.

Рассмотрим восстановление функции f с использованием регуляризации для максимальных степеней полинома 8 и 9, с которыми восстановить полиноминальную регрессию не удалось.



Сравнивая текущий график, с графиком из прошлого пункта, можно заметить, что востановленная зависимость стала более похожа на исходную функцию. Таким образом можно сделать вывод, что регуляризация может быть полезной с точки зрения восстановления исходной функцию для максимальной степени полинома большей, чем в исходной зависимости.

6 Выводы

В ходе работы был рассмотрен батчевый градиентный спуск для решений задачи линейной регрессии с постоянным шагом и ступенчатым шагом, различные модификации градиентного спуска направленные на улучшение сходимости и влияние регуляризации для решения задачи полиномиальной регрессии.

Сформулируем общие выводы:

1. Батчевый градиентный спуск с позволяет ускорить сходимость градиентного спуска, не потеряв при это точности решения.
2. Изменение шага градиетного спуска ступенчатой функций позволяет ускорить сходимость метода.
3. Среди модификаций градиетного спуска наиболее надежными являются RMSProp и Adam, наименее требовательной по машинным ресурсам - Momentum.
4. Регуляризация помогает восстанавливать полиномальную зависимость, если максимальное значение степени полинома выбрано больше чем в исходной зависимости.