Южный федеральный университет

@SINECOR

Алгоритмы стохастического градиентного спуска



Сергей Николенко
9 ноября 2026 г.

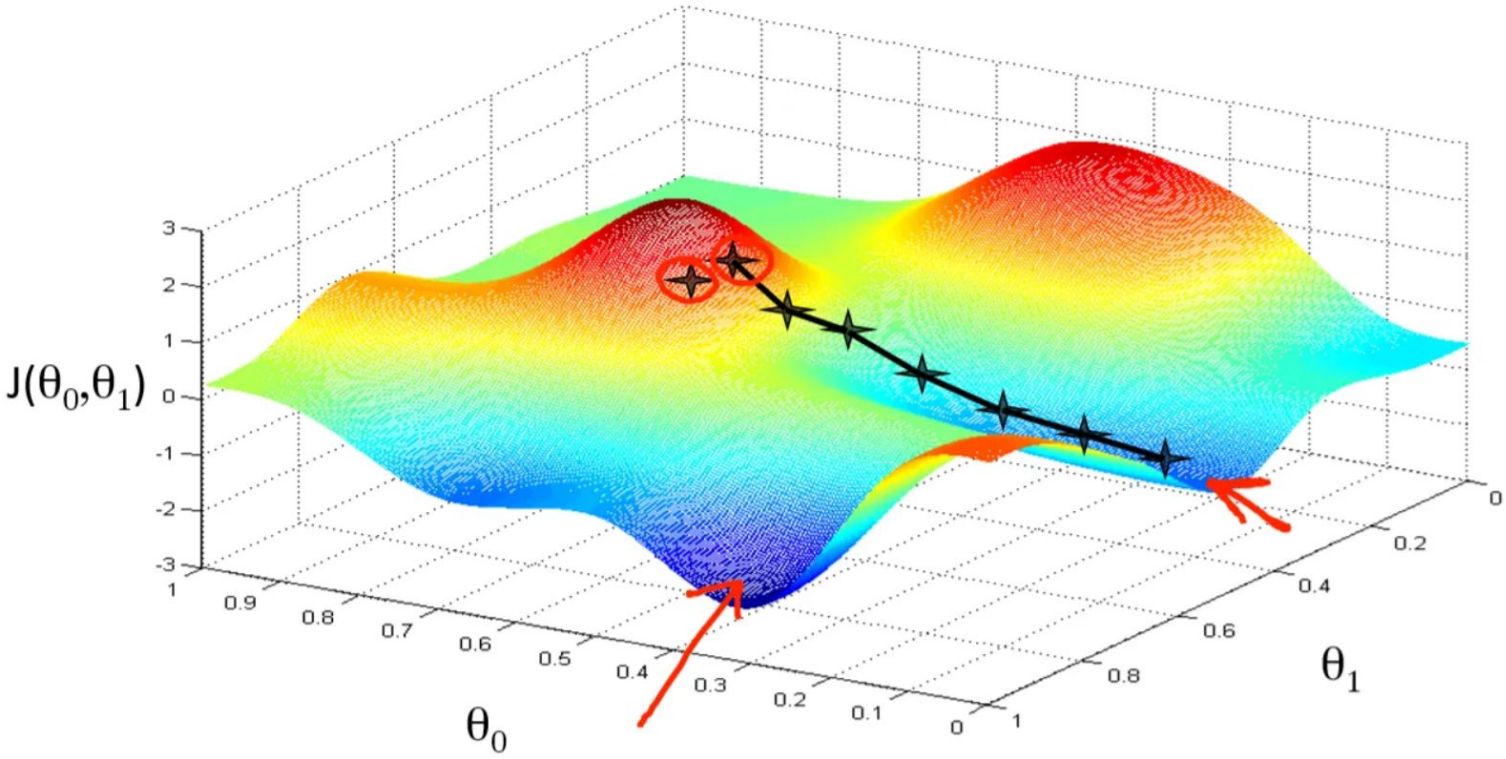
Технологии и фронтиры науки о данных

Стохастический градиентный спуск



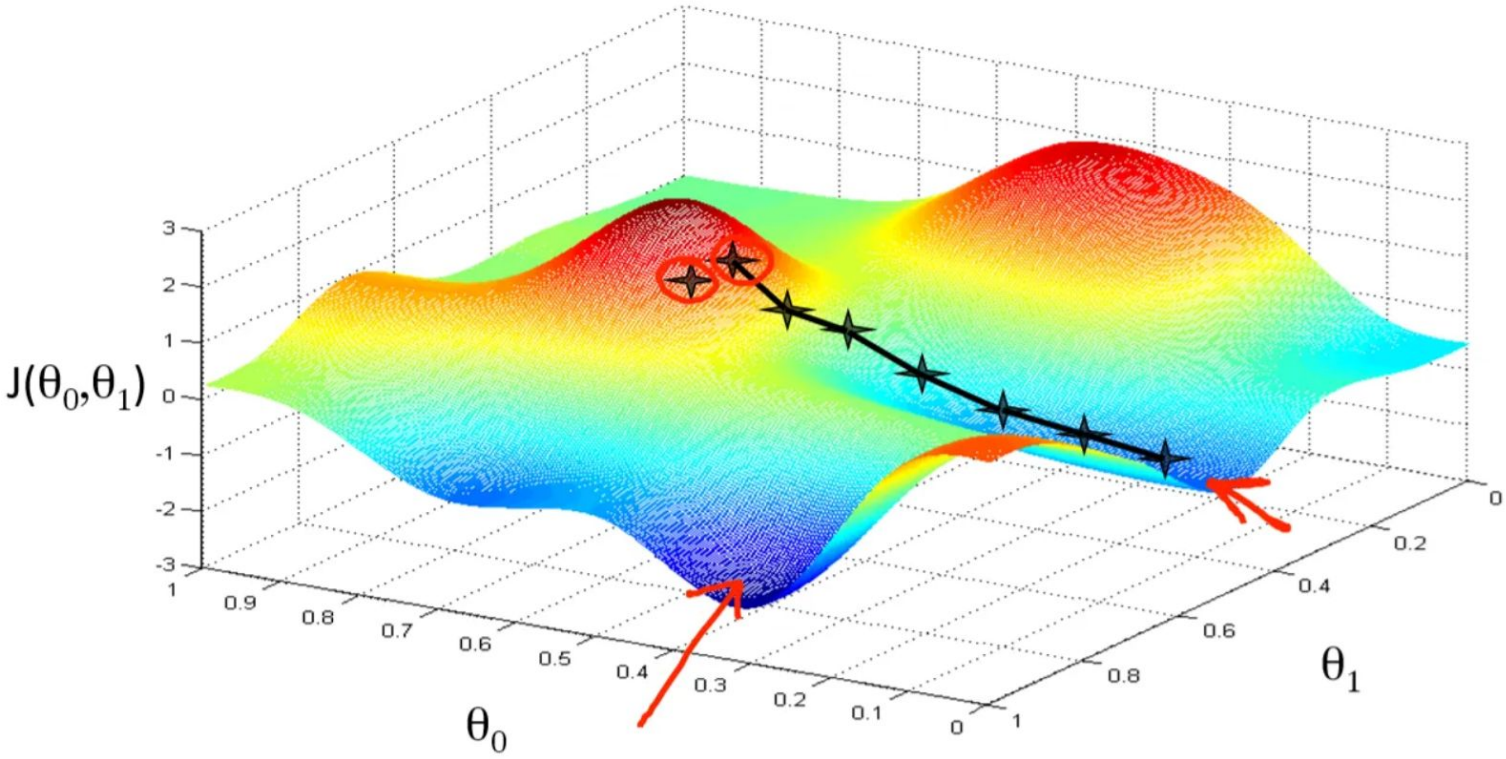
Градиентный спуск

- Мы уже обсудили, как устроены нейросети, как происходит обратное распространение ошибки, то есть как можно считать градиенты
- А потом происходит градиентный спуск, о котором мы уже говорили. Точнее...



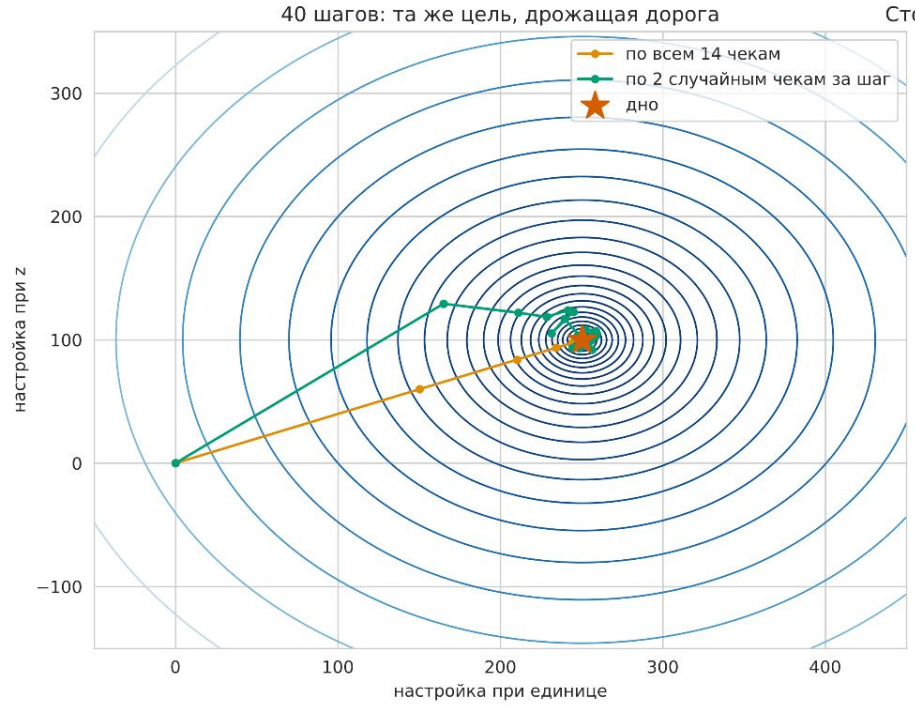
Градиентный спуск

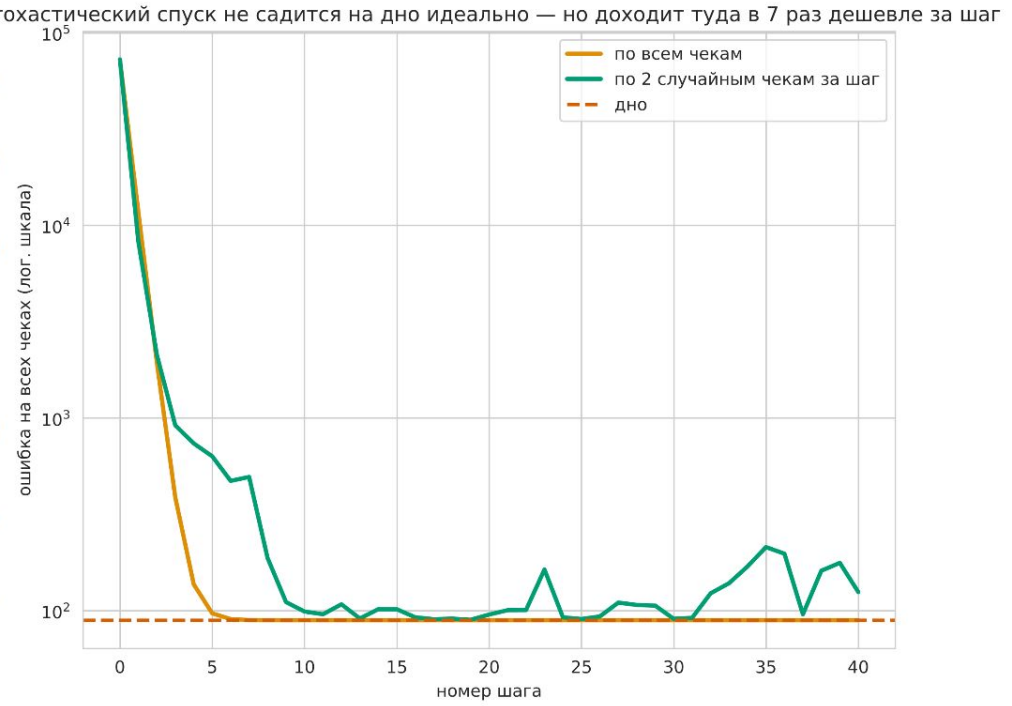
- Мы уже обсудили, как устроены нейросети, как происходит обратное распространение ошибки, то есть как можно считать градиенты
- А потом происходит градиентный спуск, о котором мы уже говорили. Точнее, *стохастический градиентный спуск* (stochastic gradient descent, SGD), потому что вычислить всю функцию ошибки было бы слишком дорого



Градиентный спуск

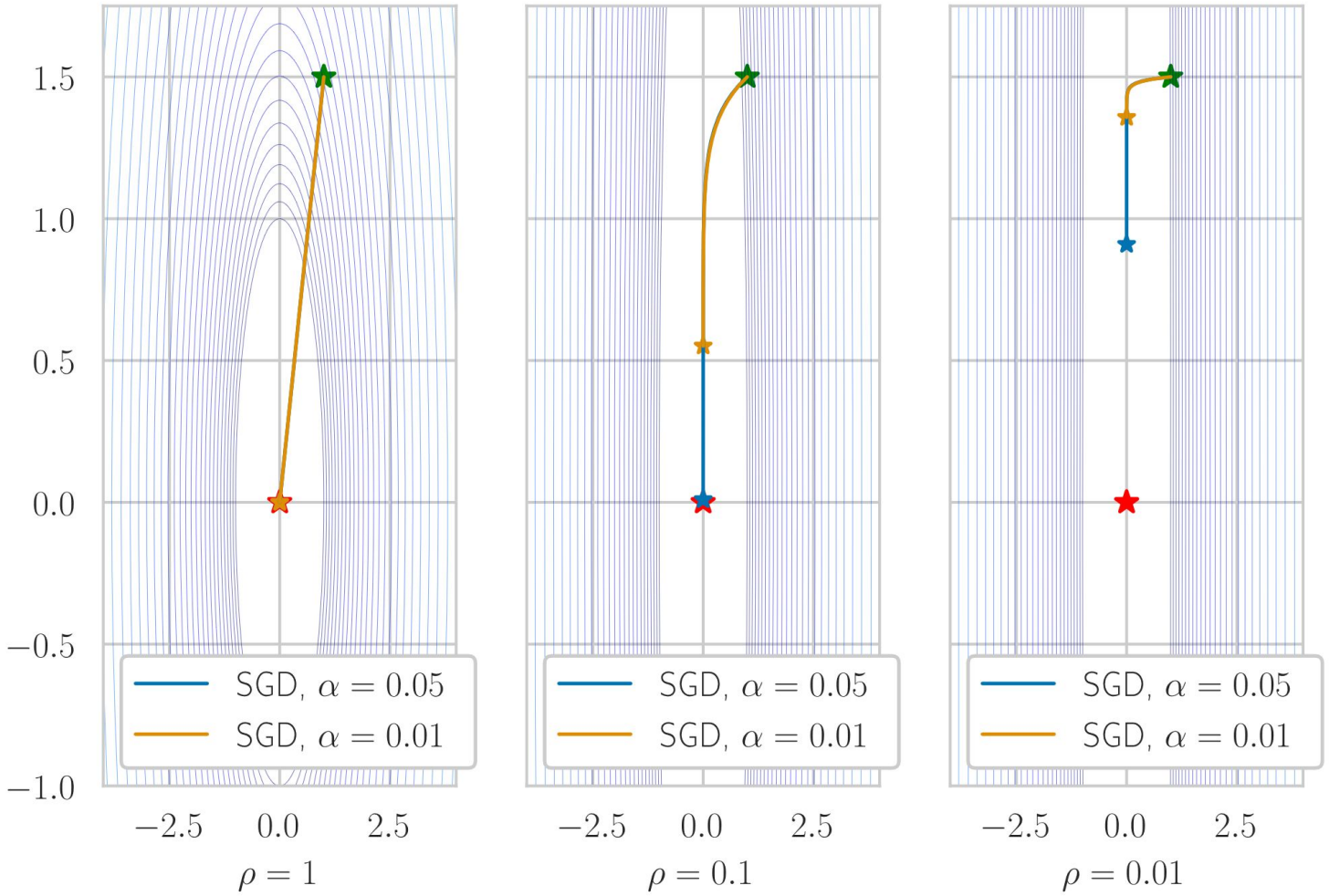
- Даже в наших примерах стохастически было бы быстрее
- Но не сошлось бы окончательно, что логично:





Градиентный спуск

- Главные проблемы начинаются в высоких размерностях, при разном масштабе переменных:

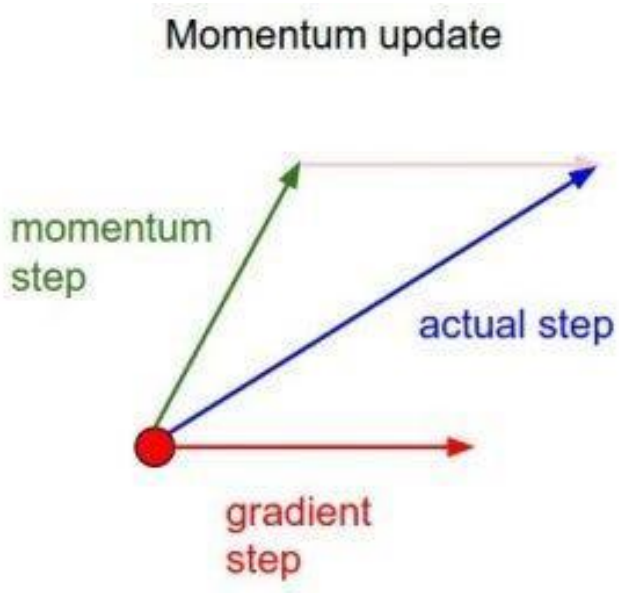


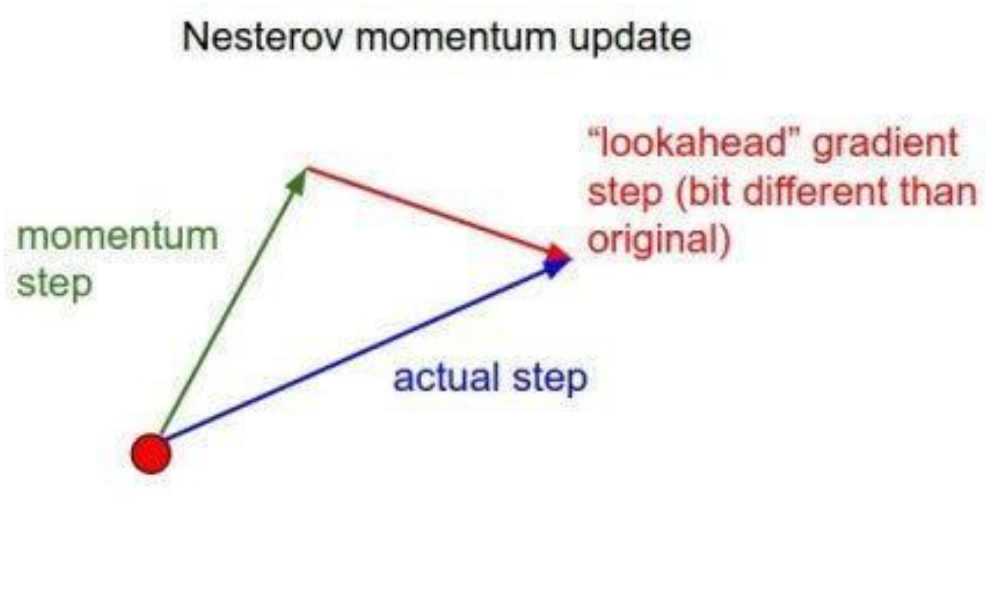
Алгоритмы стохастического градиентного спуска



Варианты градиентного спуска

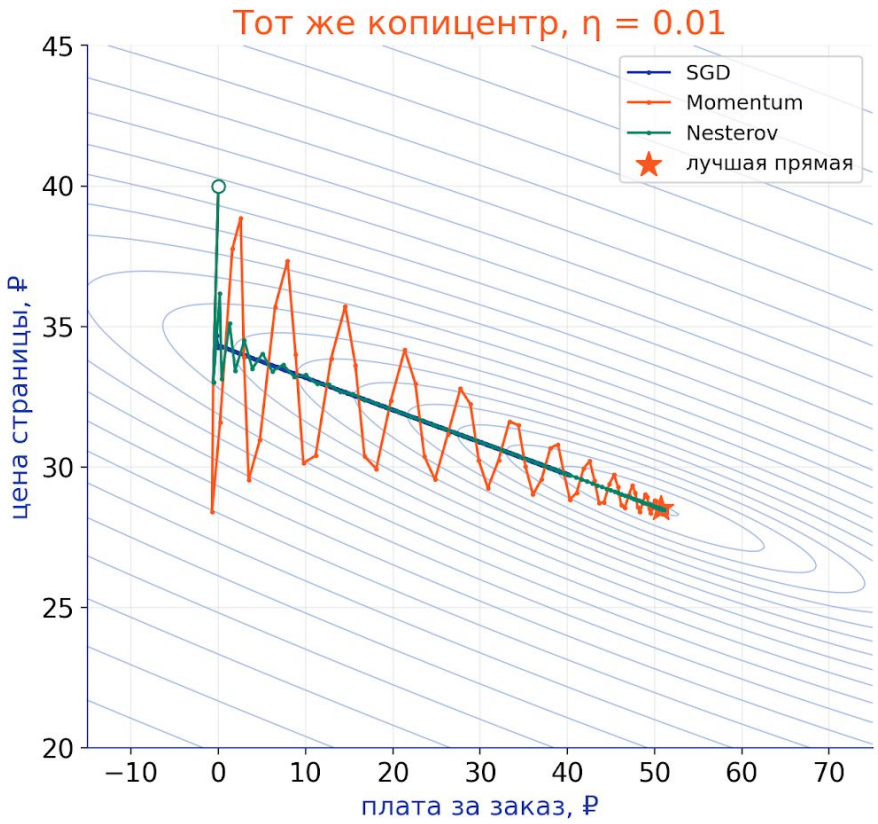
- Варианты градиентного спуска — *метод моментов*
- *И моменты Нестерова*

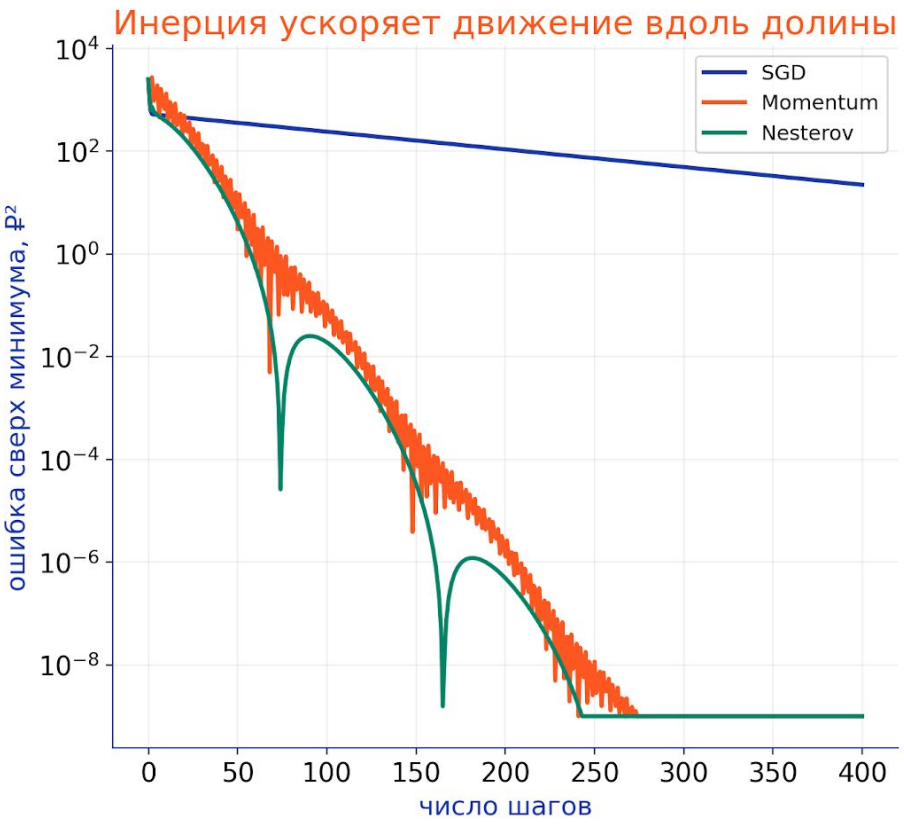




Инерция в узкой долине

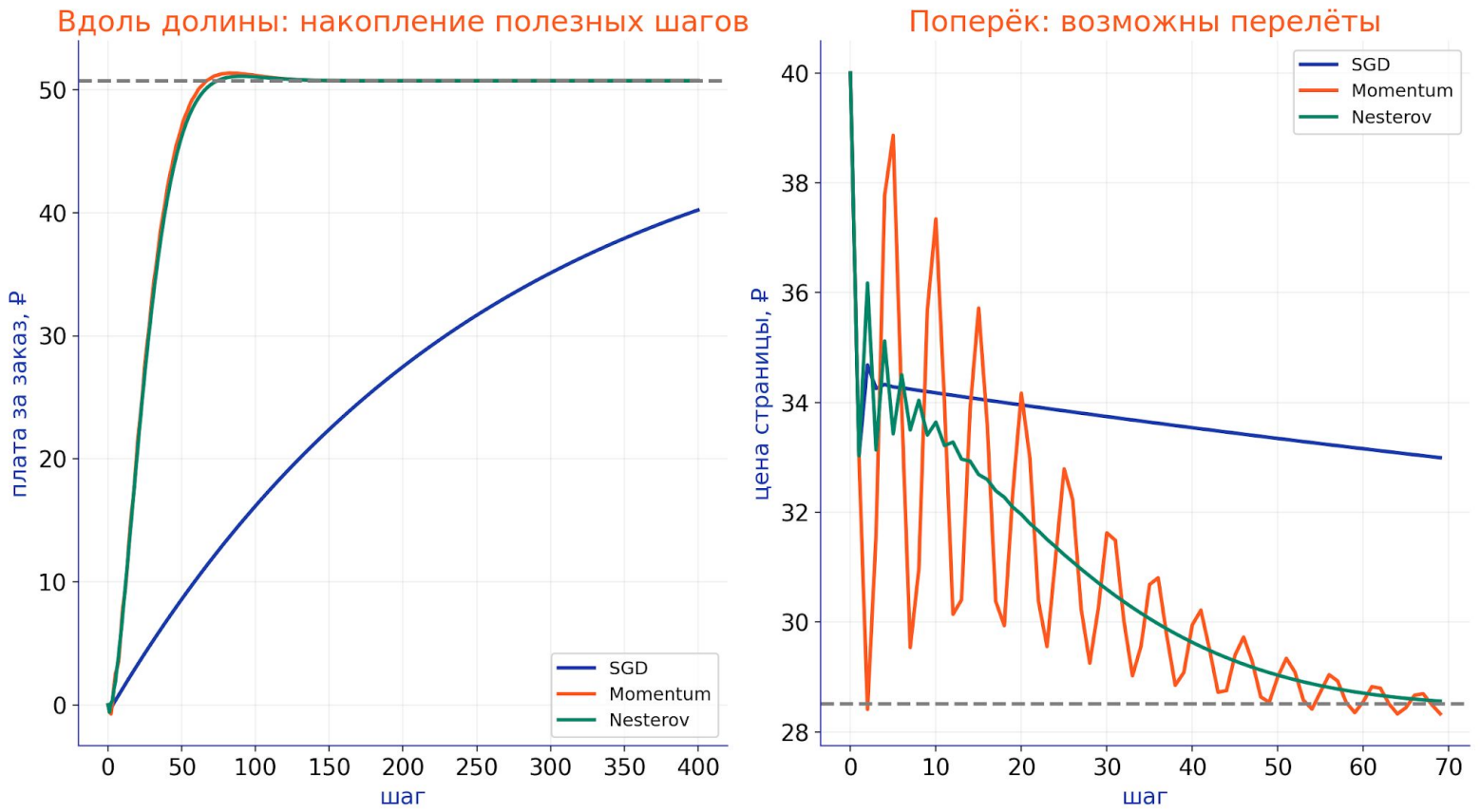
- Момент накапливает шаги вдоль долины.
- При одном старте и η = 0.01 он обгоняет обычный спуск.





Разгон и перелёты

- Постоянные толчки разгоняют движение к решению.
- Поперёк долины инерция может усилить перелёты.



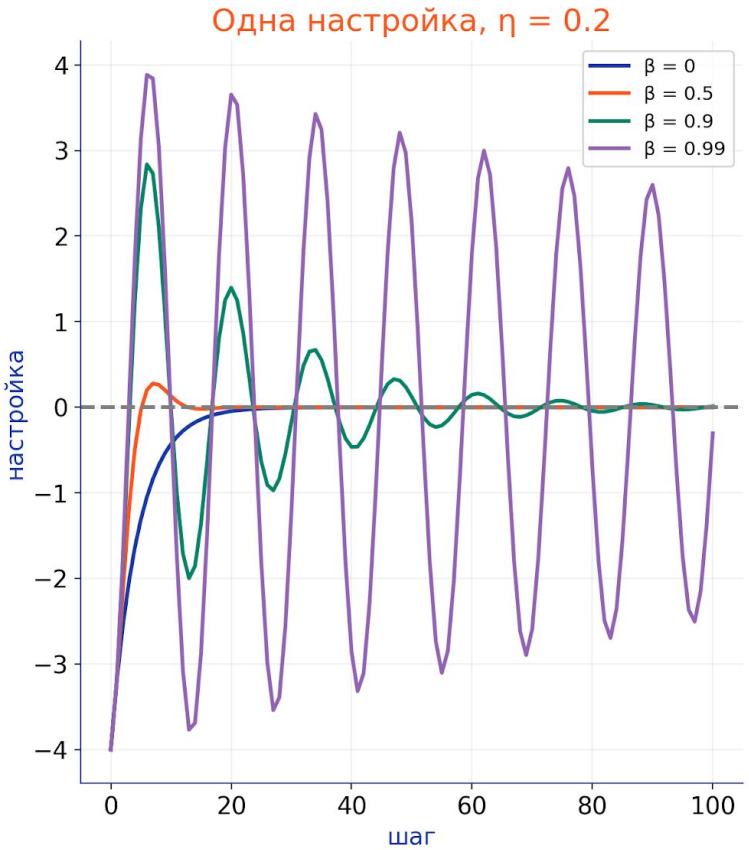
Память сглаживает шум

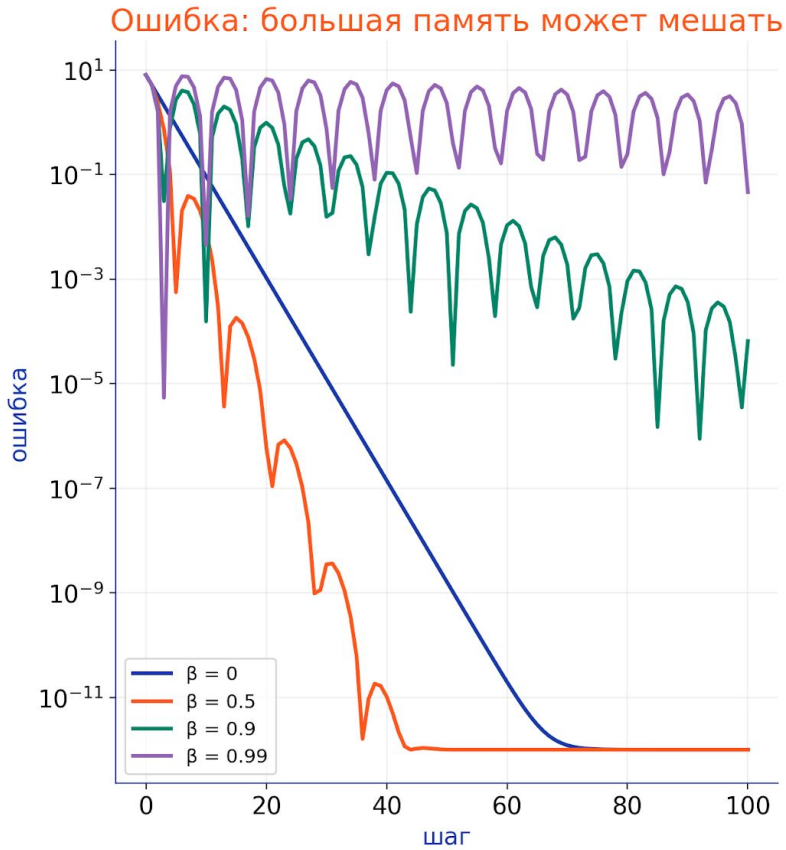
- Среднее нескольких стрелок меняется спокойнее.
- Длинная память медленнее замечает смену направления.



Слишком много инерции

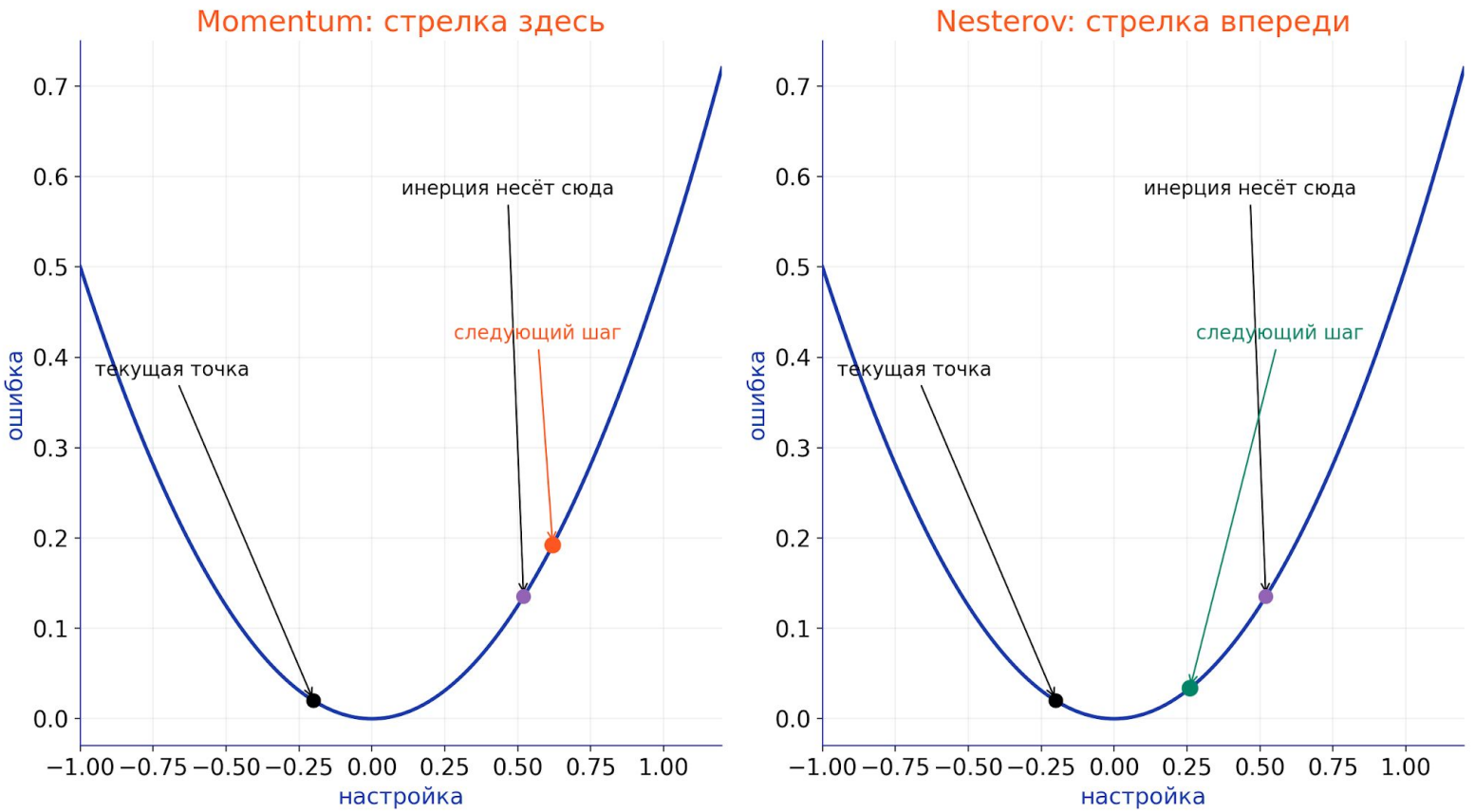
- Меняем только β: старт и скорость одинаковы.
- При β = 0.99 настройка долго качается вокруг минимума.





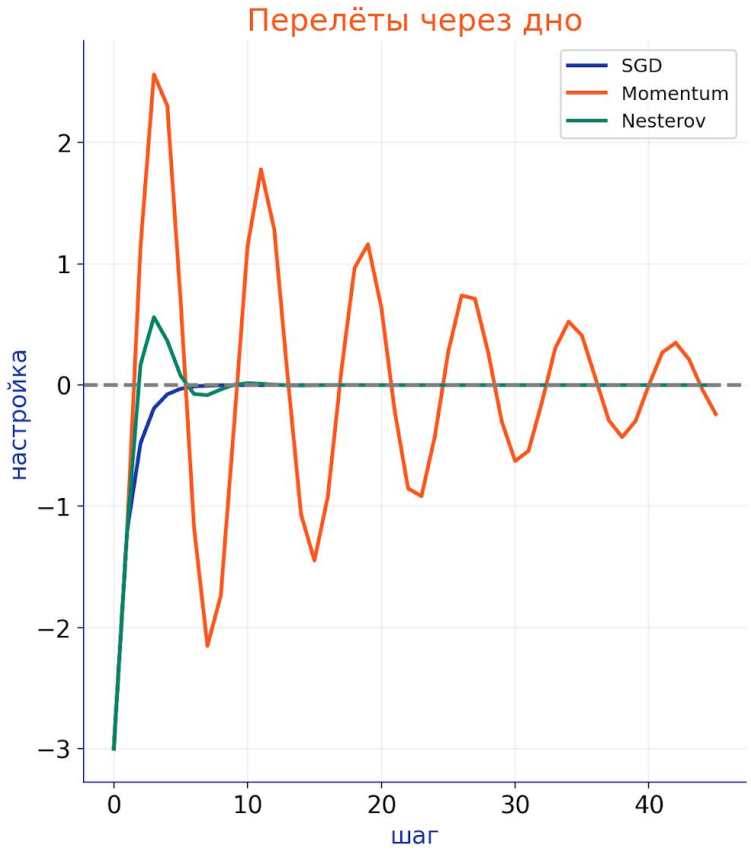
Нестеров смотрит вперёд

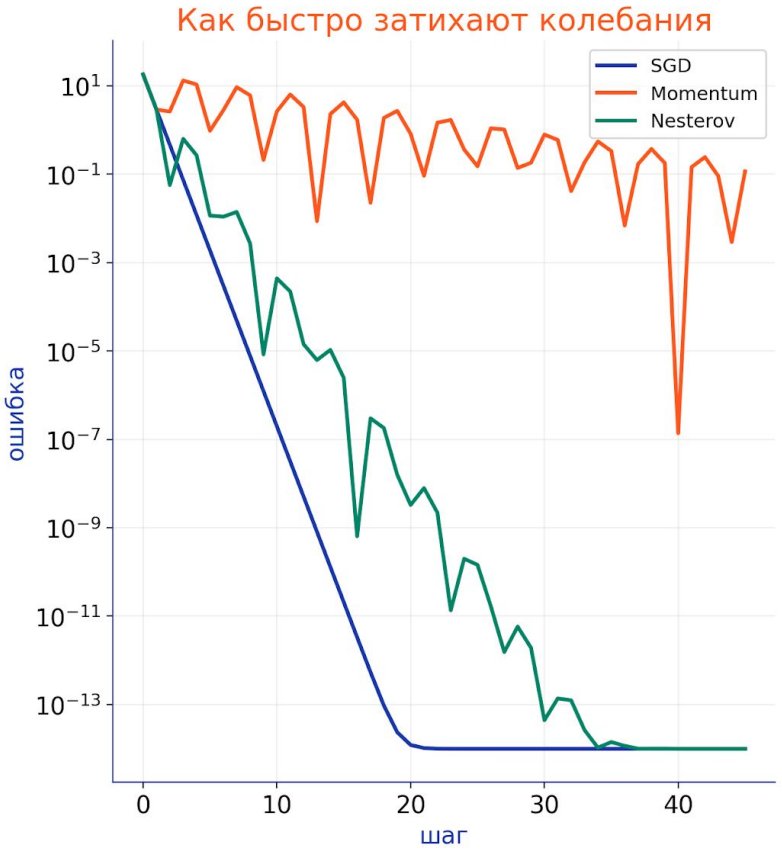
- Момент берёт градиент в текущей точке.
- Нестеров — там, куда нас уже несёт инерция.



Торможение перед минимумом

- Здесь взгляд вперёд помогает погасить перелёты.
- У обычного момента колебания затухают гораздо дольше.



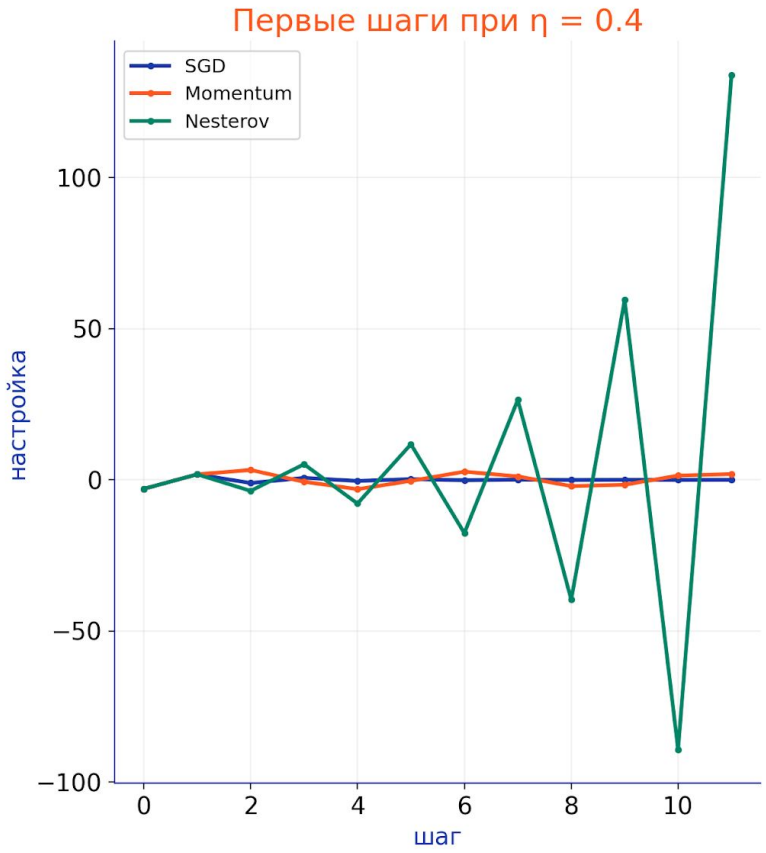


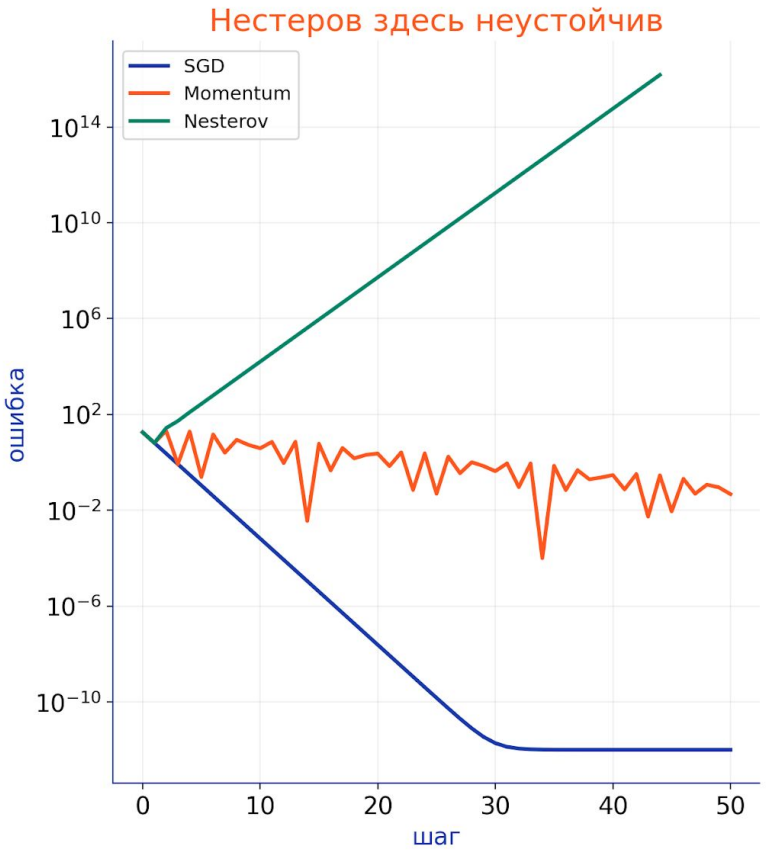
Адаптивные алгоритмы SGD



Но шаг может быть слишком велик

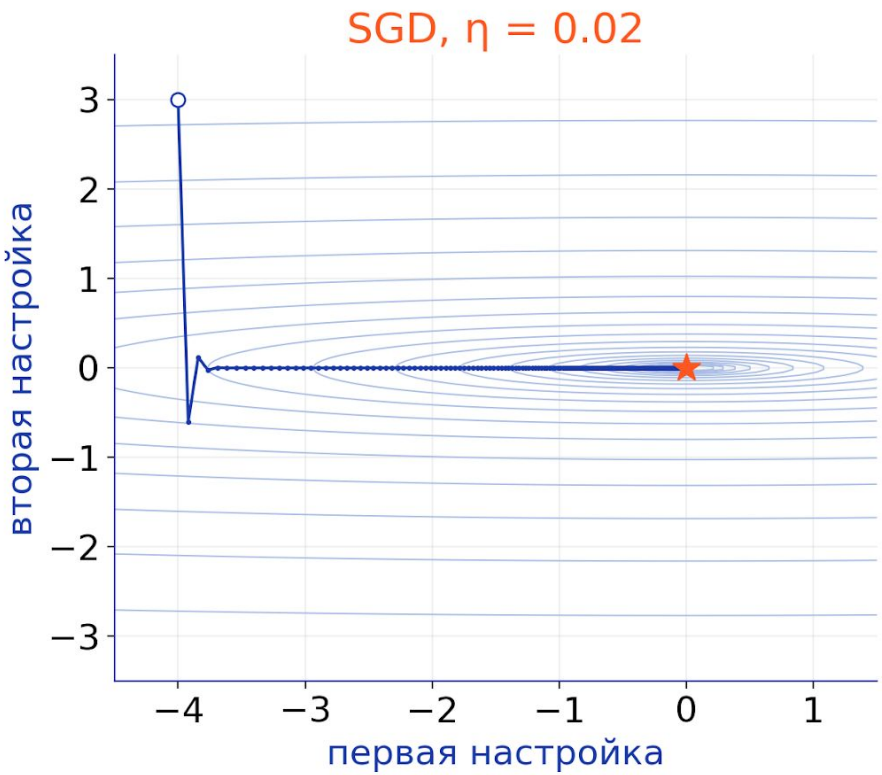
- Берём ту же чашу, но увеличиваем η до 0.4.
- Обычный момент ещё устойчив, а Нестеров уже расходится.

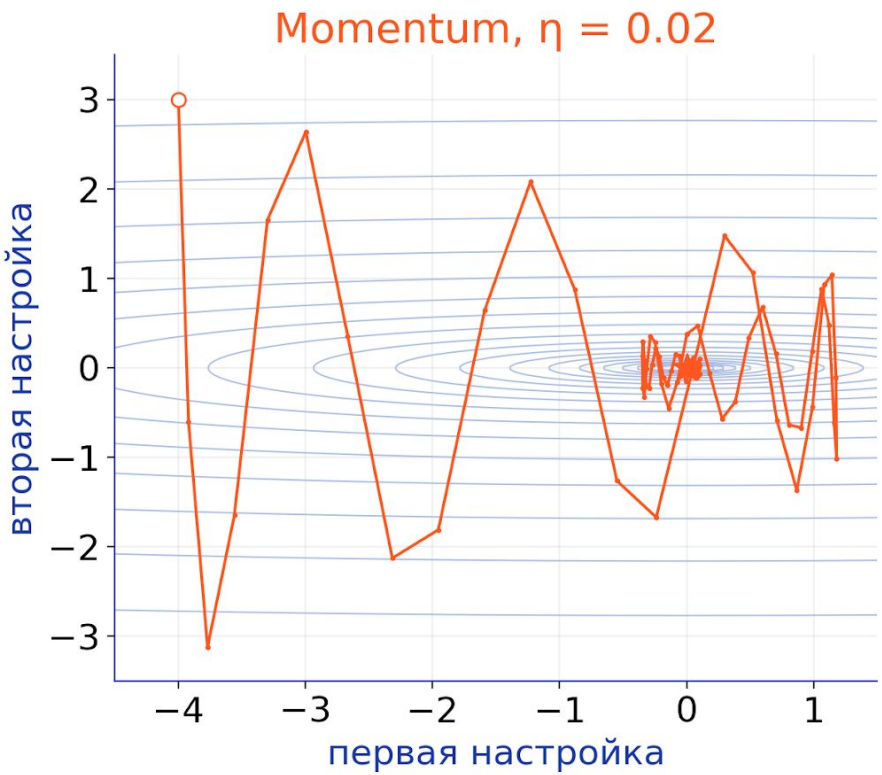




Одна чаша: SGD и момент

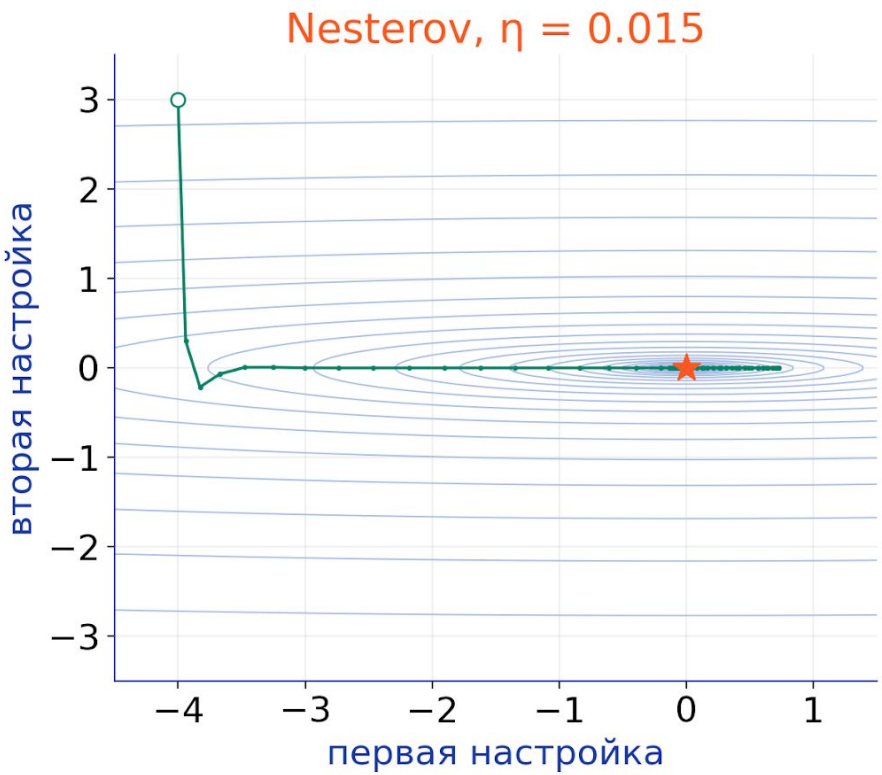
- Функция, старт и число шагов одинаковы.
- У момента появляется движение вдоль пологого направления.

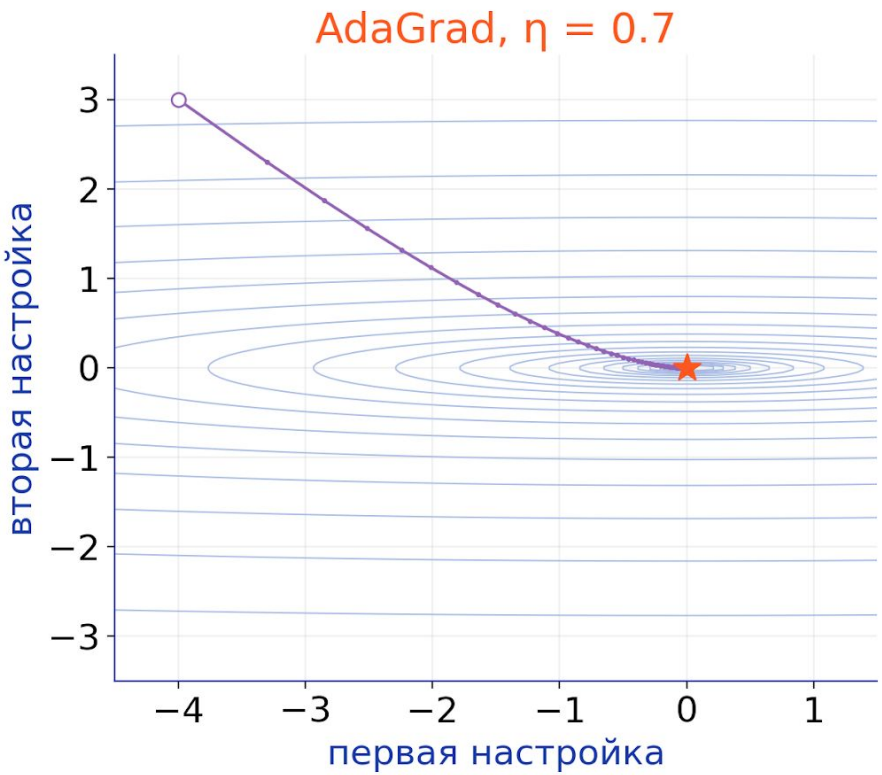




Одна чаша: Нестеров и AdaGrad

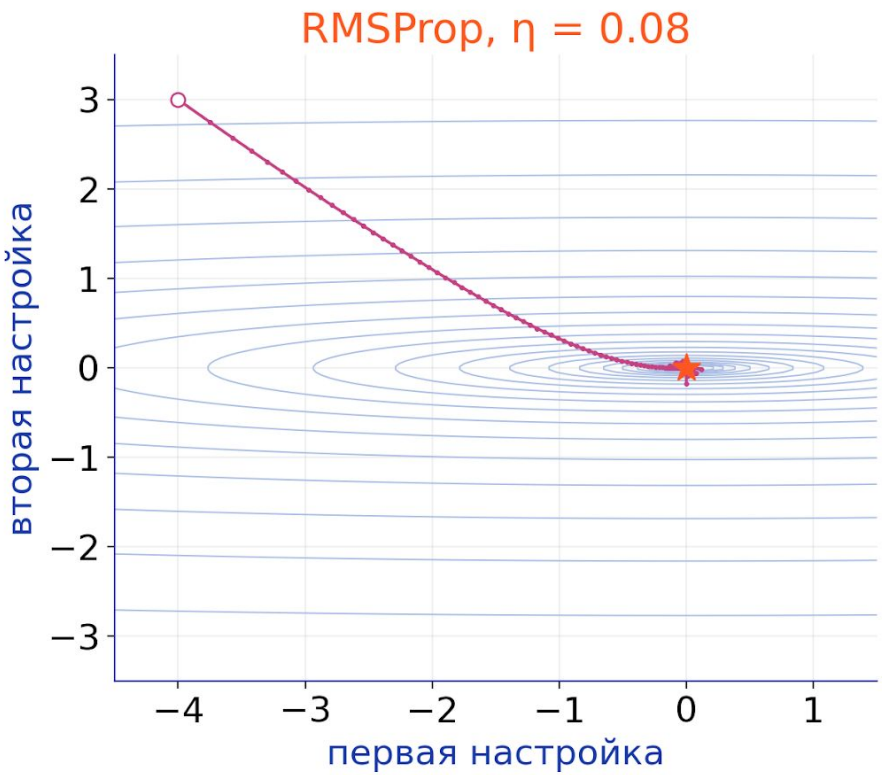
- Нестеров использует взгляд вперёд.
- AdaGrad выбирает свой коэффициент шага для каждой оси.

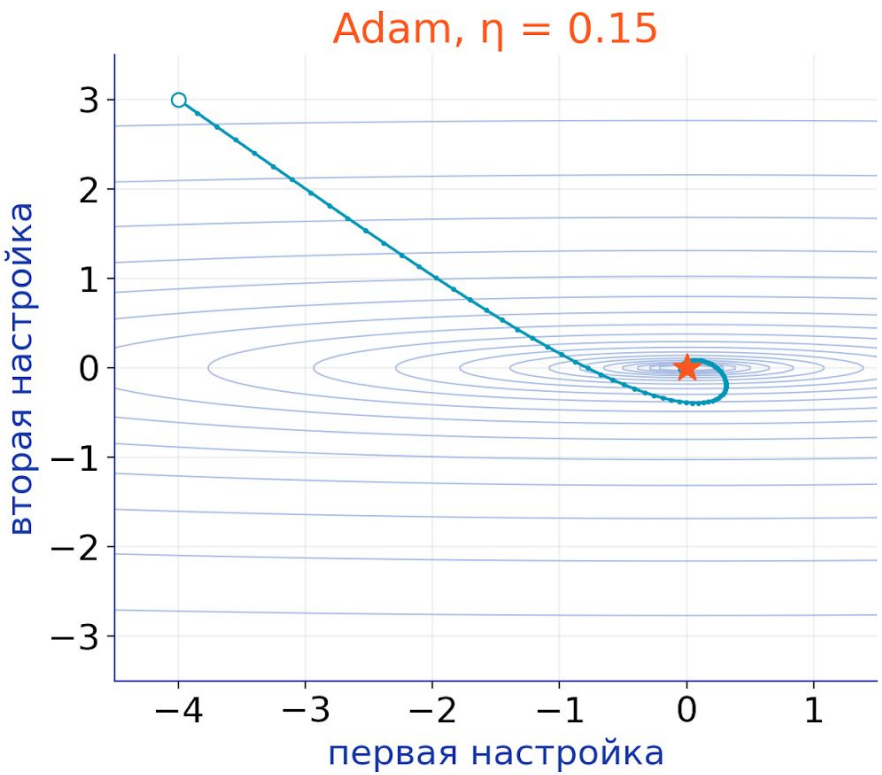




Одна чаша: RMSProp и Adam

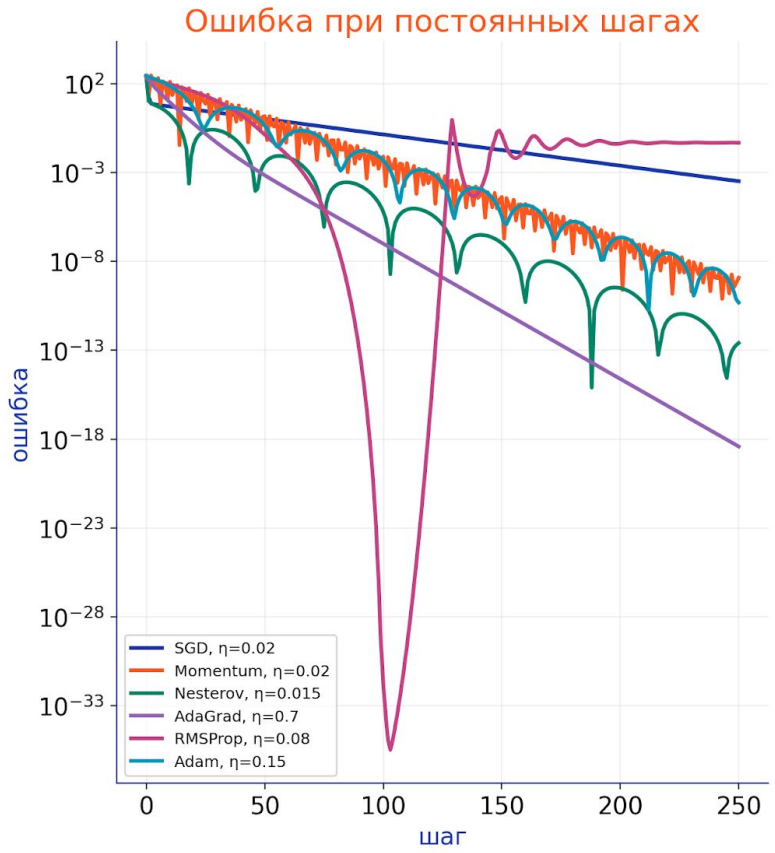
- RMSProp учитывает недавний масштаб градиента.
- Adam добавляет к этому память о направлении.

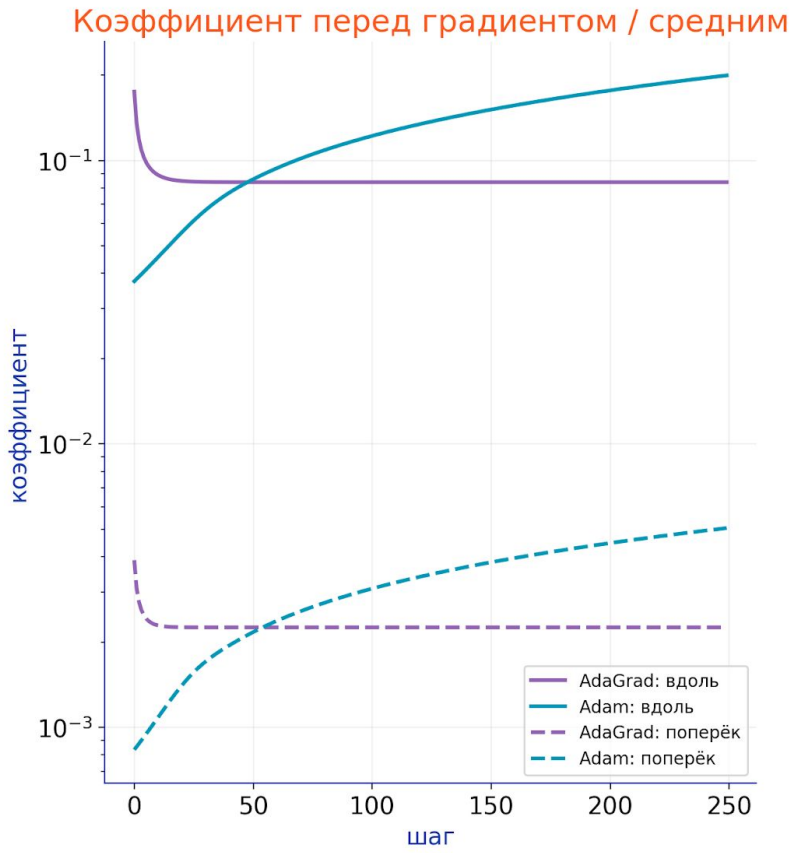




Что происходит с размером шага

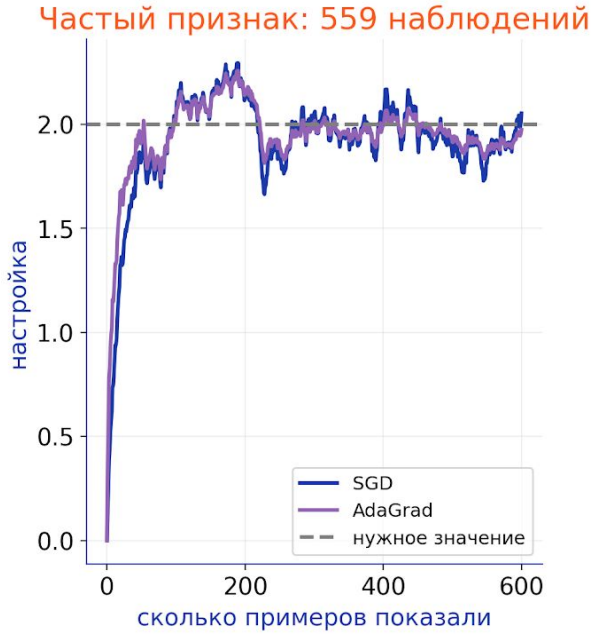
- У адаптивных методов коэффициенты по осям различаются.
- Постоянная скорость может оставлять колебания около дна.

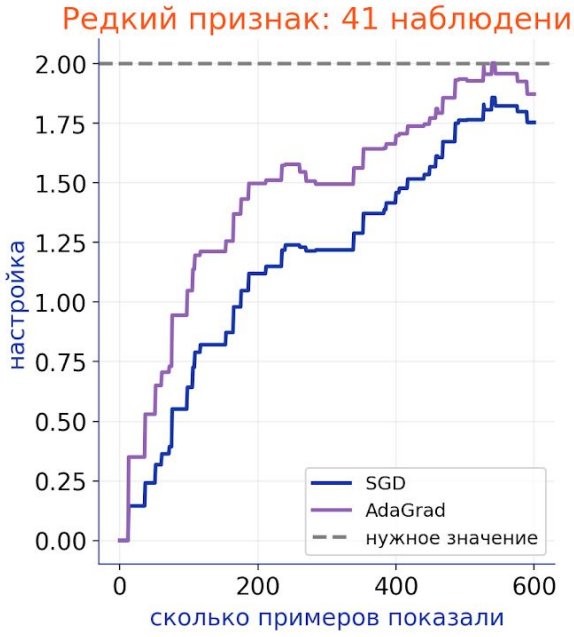


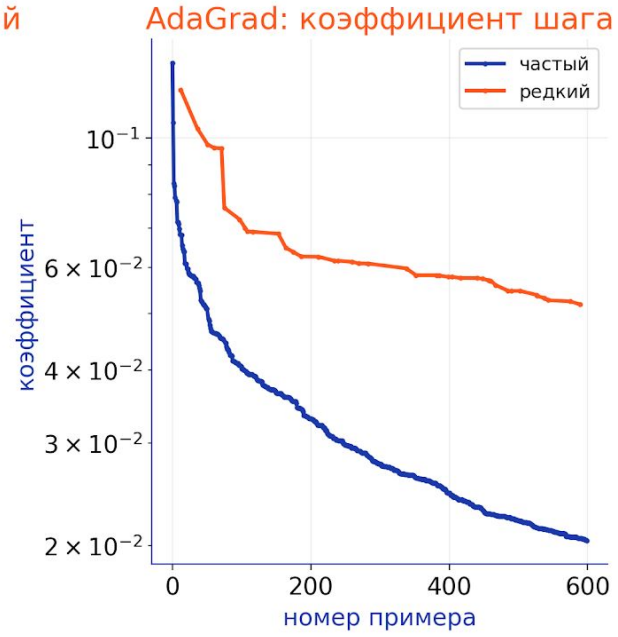


Редкому признаку тоже нужен шаг

- Обычный SGD редко обновляет редкий признак.
- AdaGrad оставляет для него больший коэффициент шага.

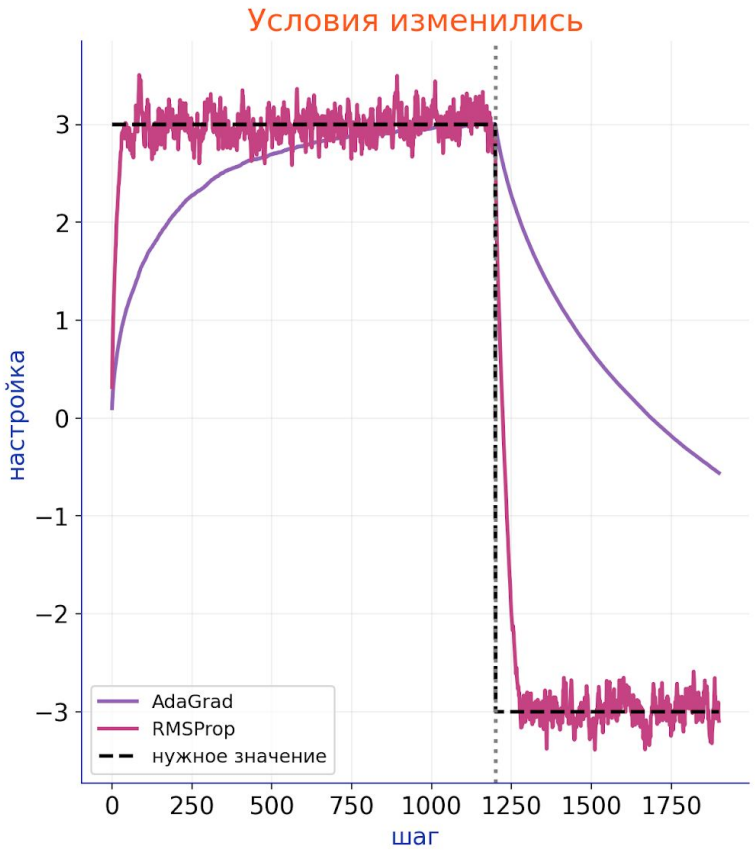


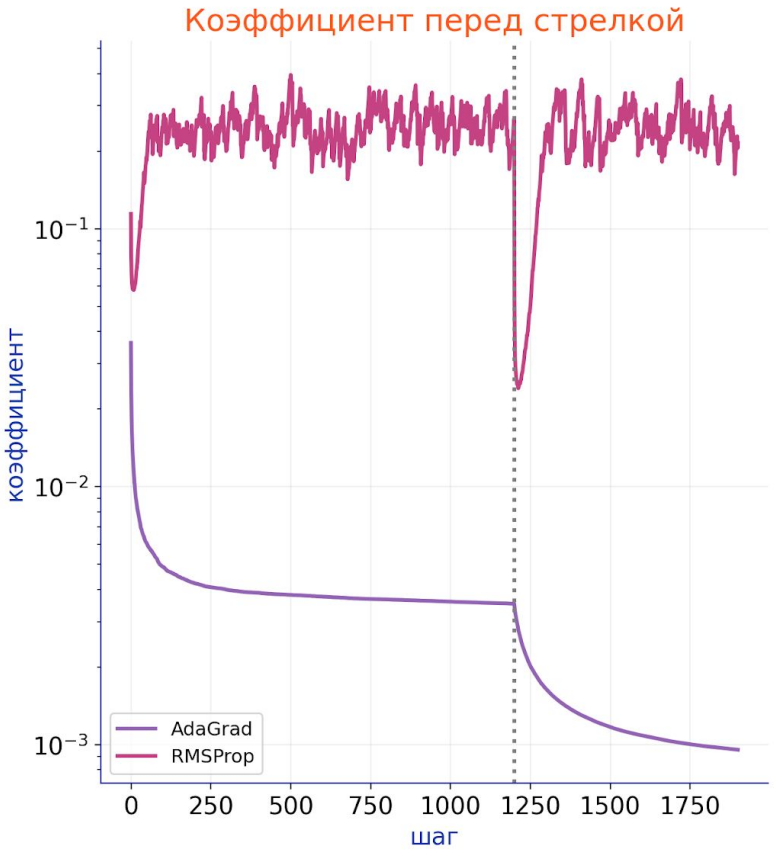




А если задача изменилась?

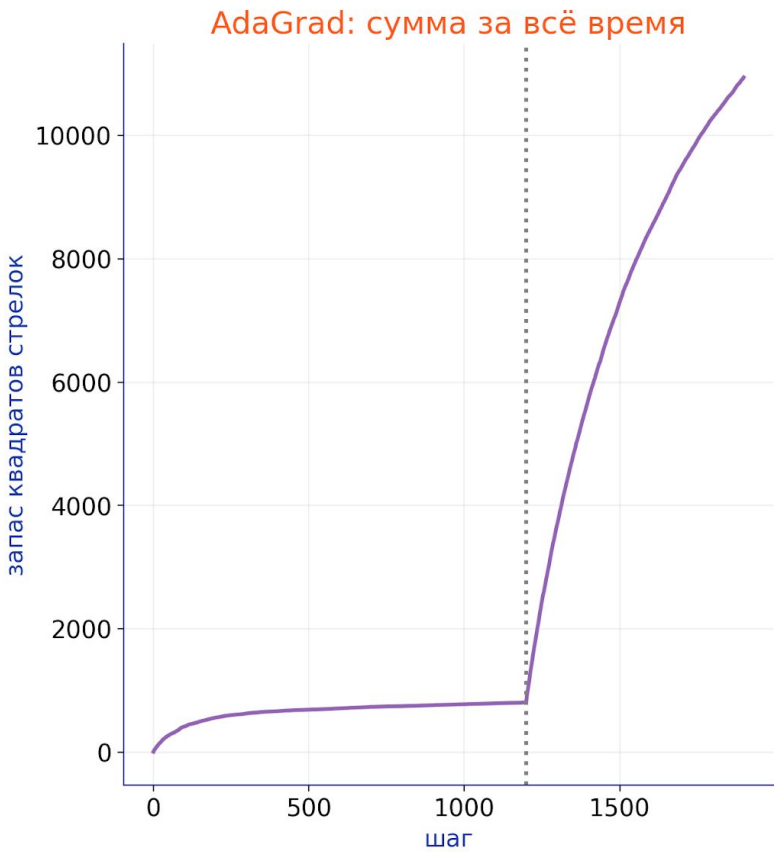
- AdaGrad копит квадраты градиентов за всё время.
- RMSProp забывает старое и быстрее идёт к новой цели.

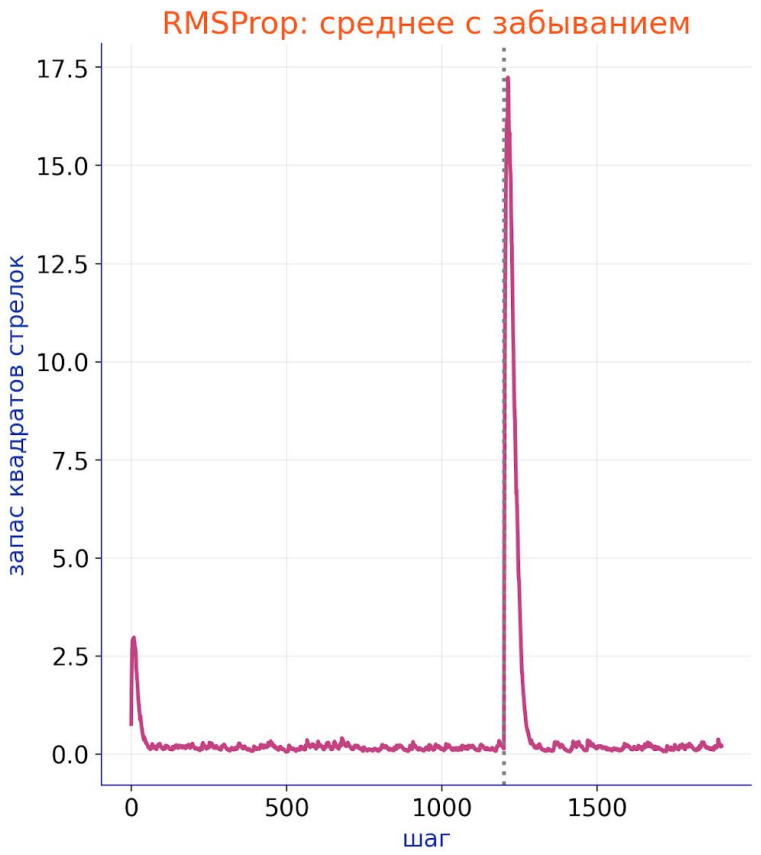




Две разные памяти

- У AdaGrad сумма квадратов только растёт.
- У RMSProp скользящее среднее может и уменьшаться.





Adam: направление и масштаб

- Средний градиент задаёт направление, средний квадрат — масштаб.
- После смены знака направление меняется с задержкой.

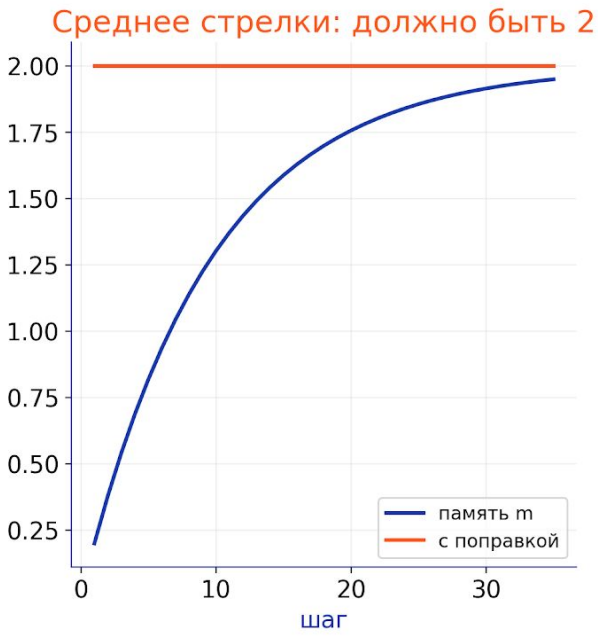


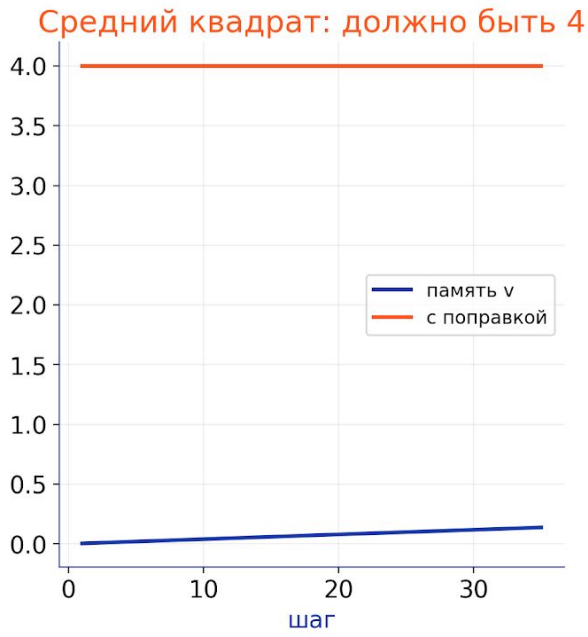


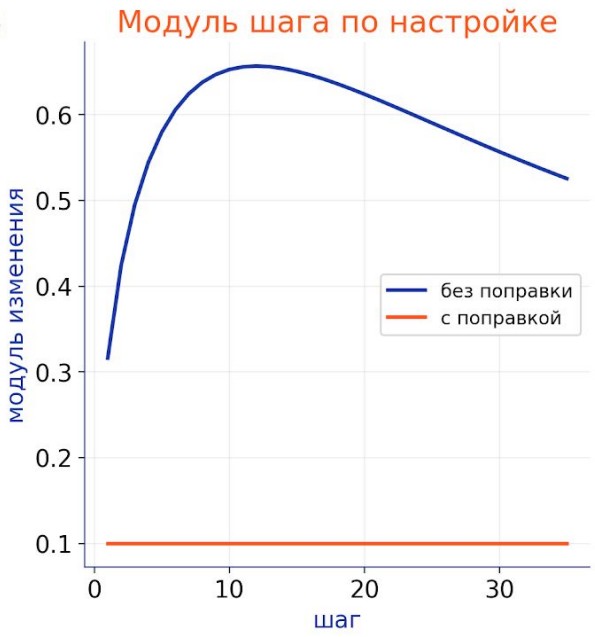


Поправка на нулевой старт

- В начале обе памяти занижены из-за нулевого старта.
- Поправка исправляет оценки; без неё шаг здесь получается больше.

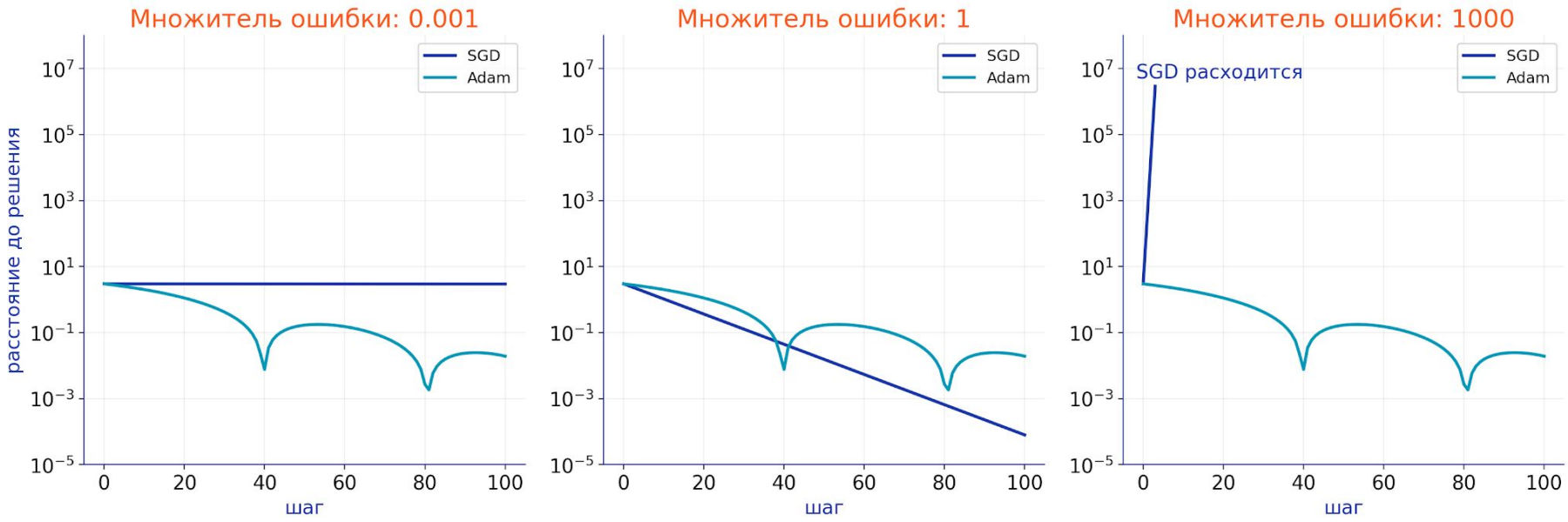






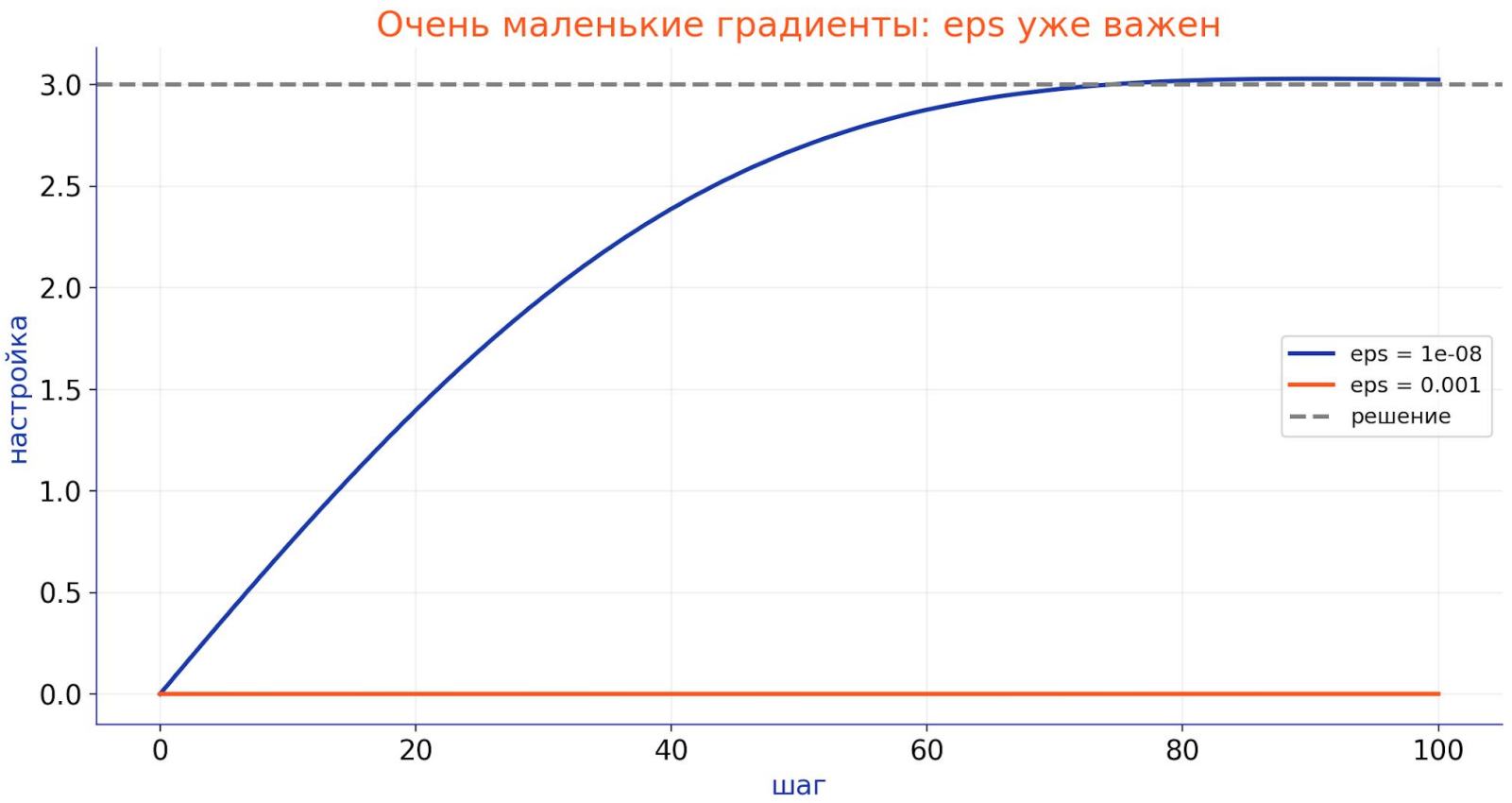
Меняем масштаб ошибки

- У всех трёх функций один минимум.
- SGD требует перенастройки η, а пути Adam почти совпадают.



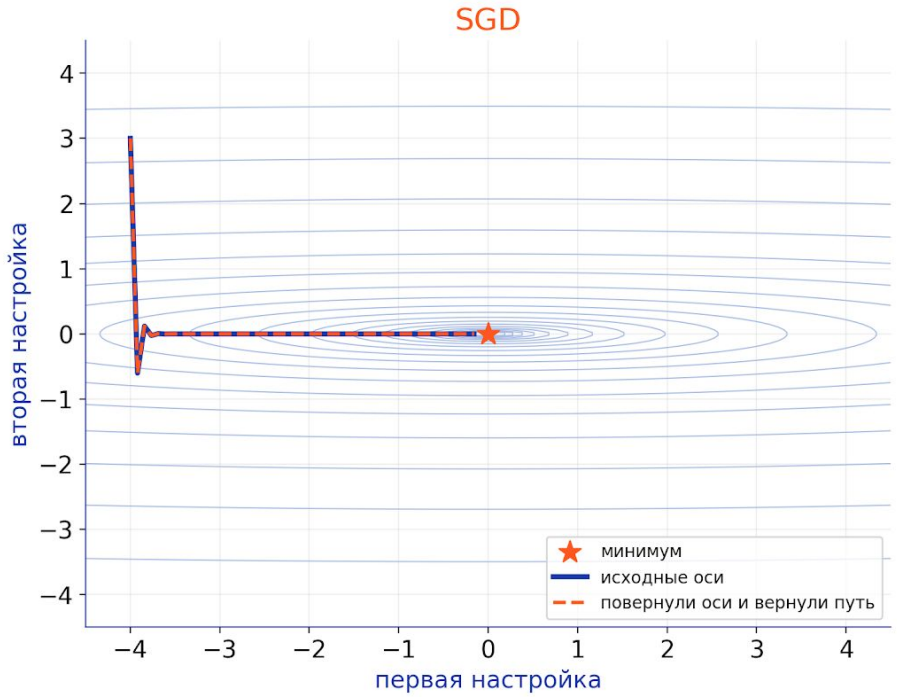
Маленькое eps тоже влияет

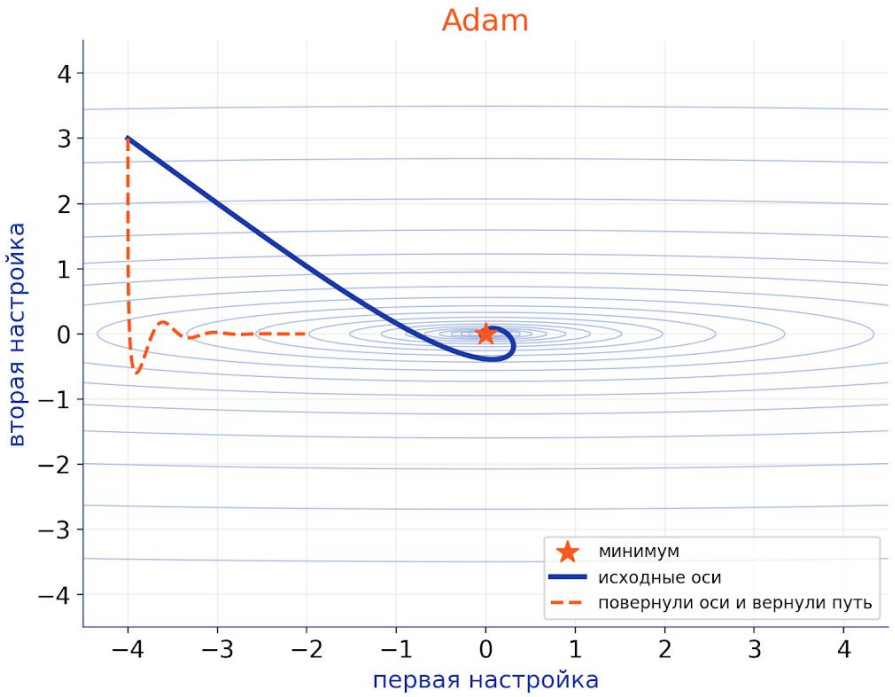
- При очень малых градиентах eps становится заметным.
- Слишком большое eps почти останавливает движение.



Поворачиваем ту же задачу

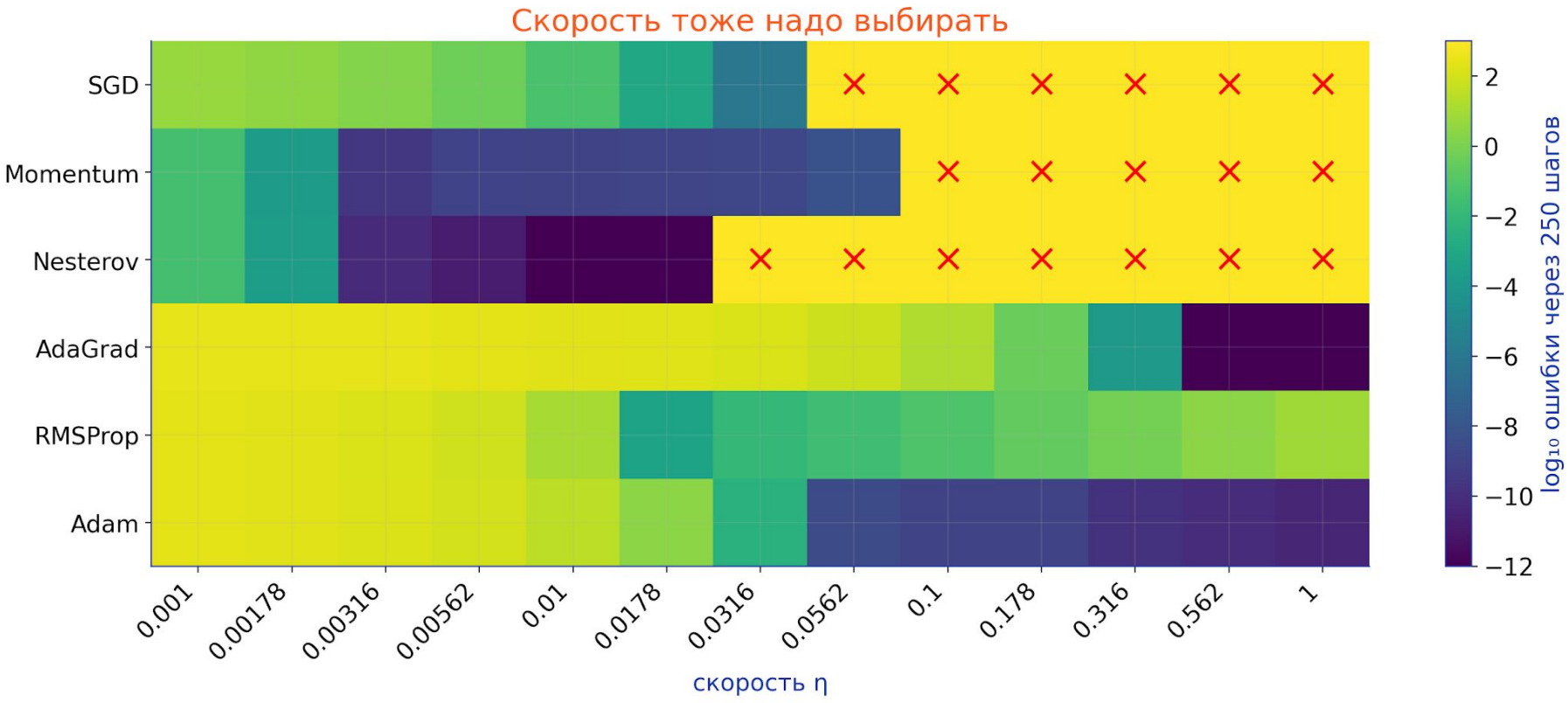
- SGD поворачивается вместе с чашей: пути совпадают.
- Покоординатная адаптация Adam зависит от направления осей.





Алгоритм и скорость выбираем вместе

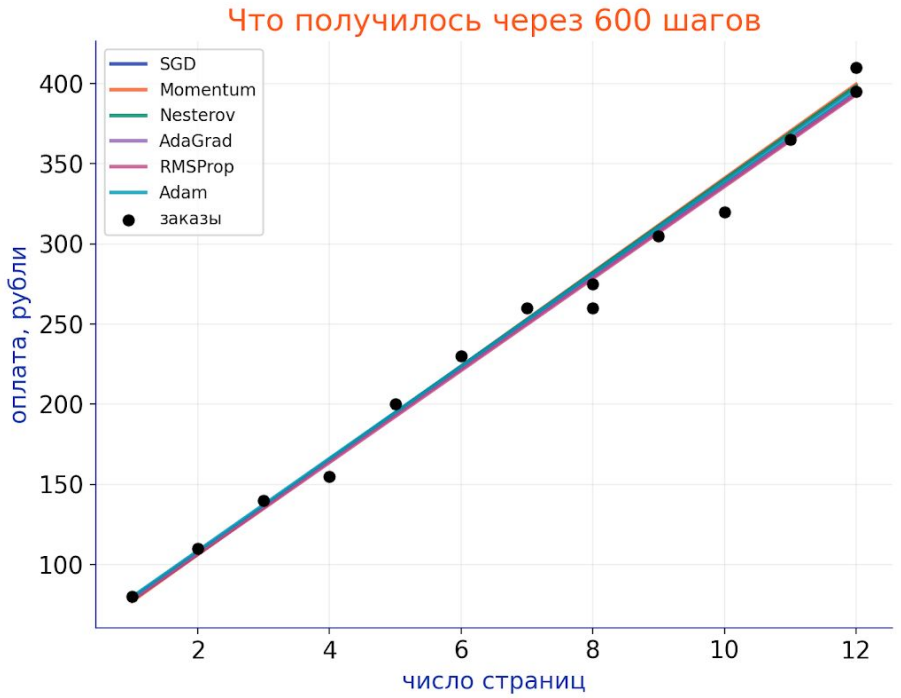
- Темнее — меньше ошибка через 250 шагов; крестик — неудачный запуск.
- У каждого метода свой диапазон хороших скоростей.



Случайные заказы копицентра

- Всем методам даём одинаковые мини-батчи по три заказа.
- Пунктир — лучшая ошибка этой линейной модели на всех данных.





Варианты градиентного спуска

- От Adagrad...
- ...через RMSprop и Adadelta до Adam...
- ...а сейчас ещё Muon, Lion, и на этом наверняка прогресс ещё не закончился

$$\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$
$$u_t = \frac{\eta}{\sqrt{v + \epsilon}} m_t.$$

```
def train(w, g, m, v, lr):
  g = clip(g, lr)
  g = arcsin(g)
  m = interp(g, v, 0.899)
  m2 = m * m
  v = interp(g, m, 1.109)
  abs_m = sqrt(m2)
  update = m / abs_m
  wd = w * 0.4602
  update = update + wd
  lr = lr * 0.0002
  m = cosh(update)
  update = update * lr
  return update, m, v
```

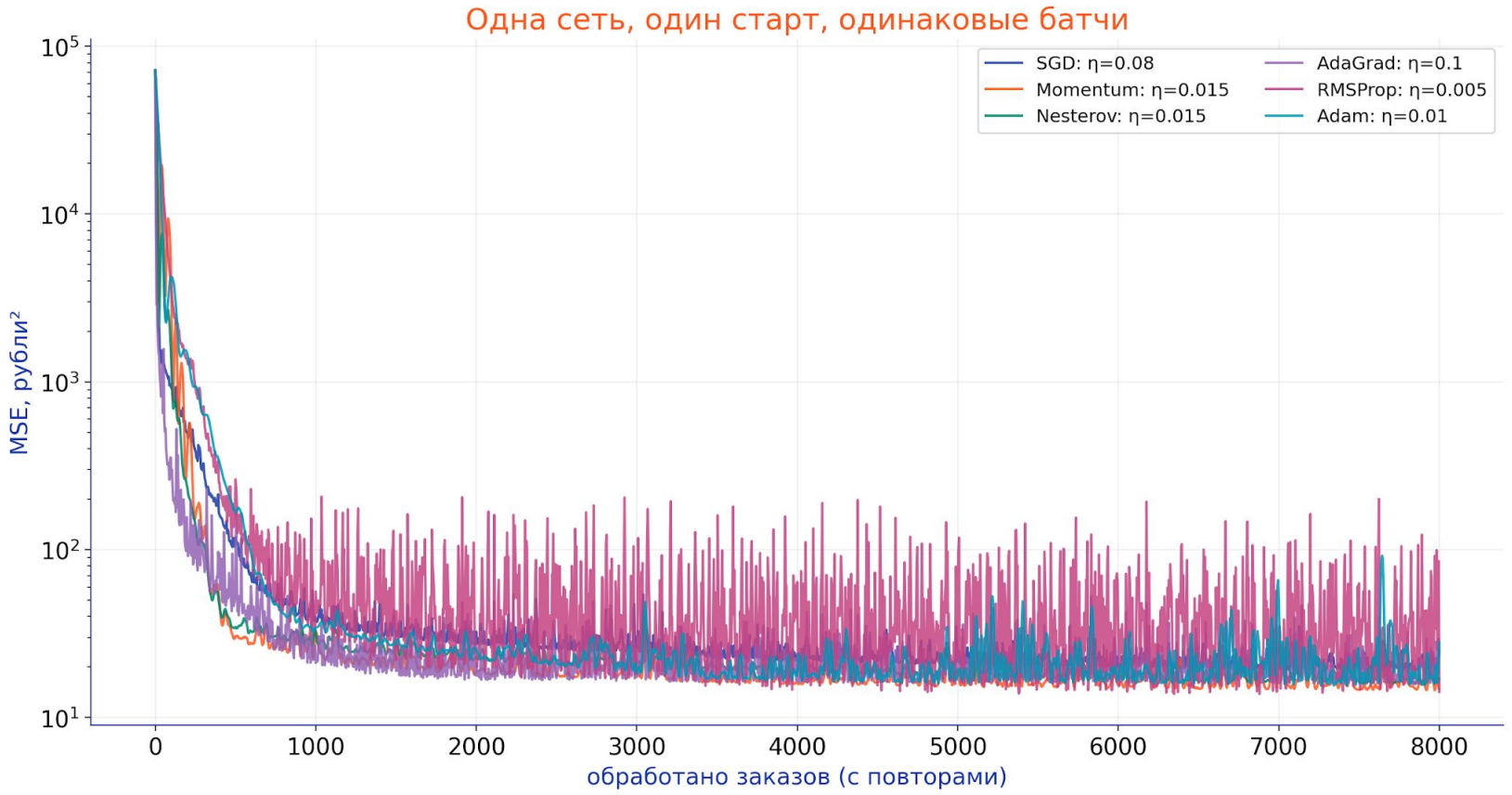
Algorithm 2 Muon

Require: Learning rate η, momentum μ

1: Initialize $B_0 \leftarrow 0$
2: **for** $t = 1, \ldots$ **do**
3: Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
4: $B_t \leftarrow \mu B_{t-1} + G_t$
5: $O_t \leftarrow \text{NewtonSchulz5}(B_t)$
6: Update parameters $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
7: **end for**
8: **return** θ_t

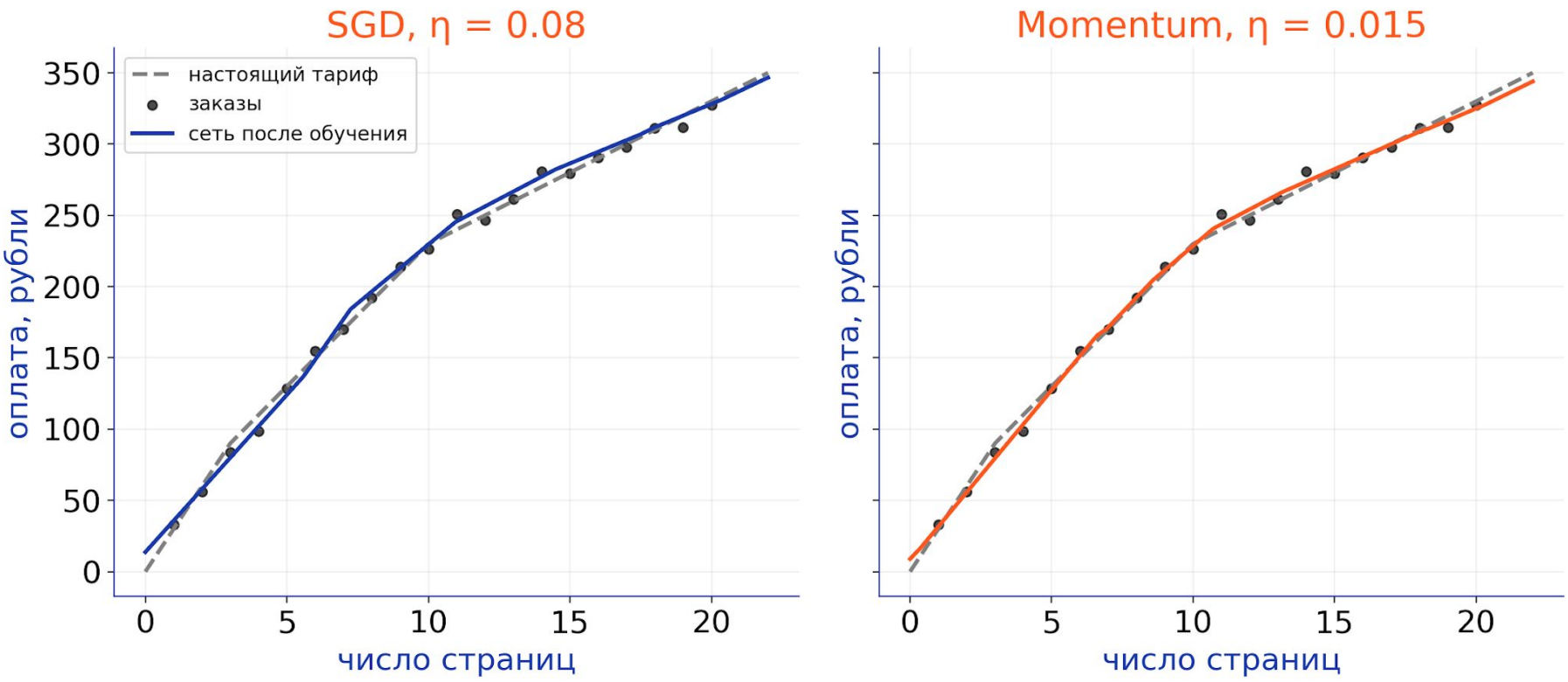
Теперь обучим маленькую сеть

- Одна сеть с 14 ReLU, один старт и одинаковые батчи.
- По горизонтали — число обработанных заказов с повторами.



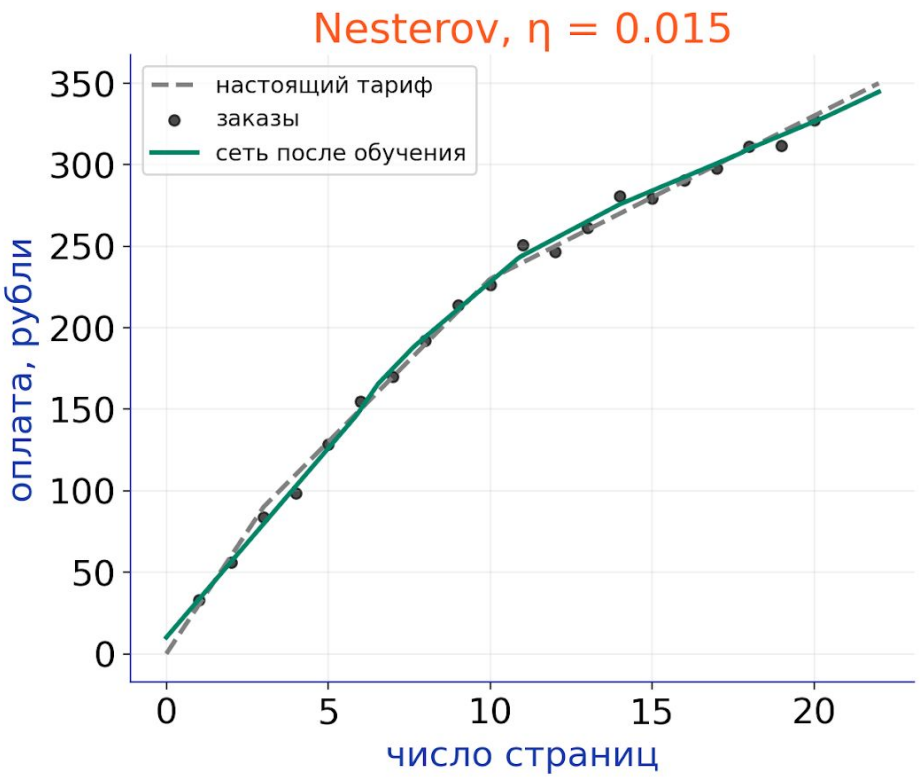
Что выучили SGD и момент

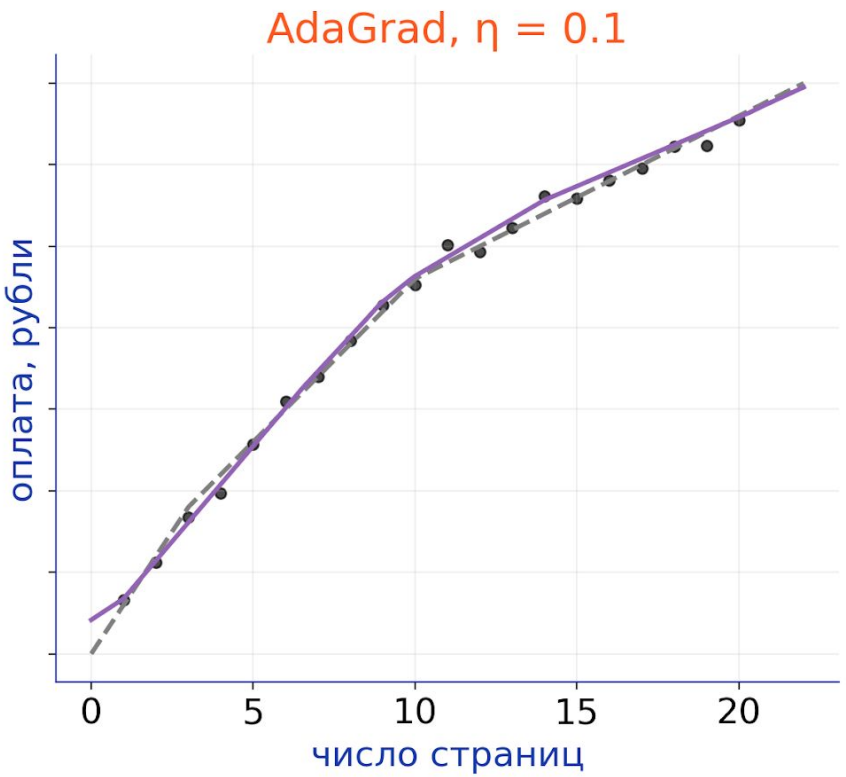
- Чёрные точки — заказы, пунктир — настоящий тариф.
- Цветная линия — ответ одной и той же сети после обучения.



Что выучили Нестеров и AdaGrad

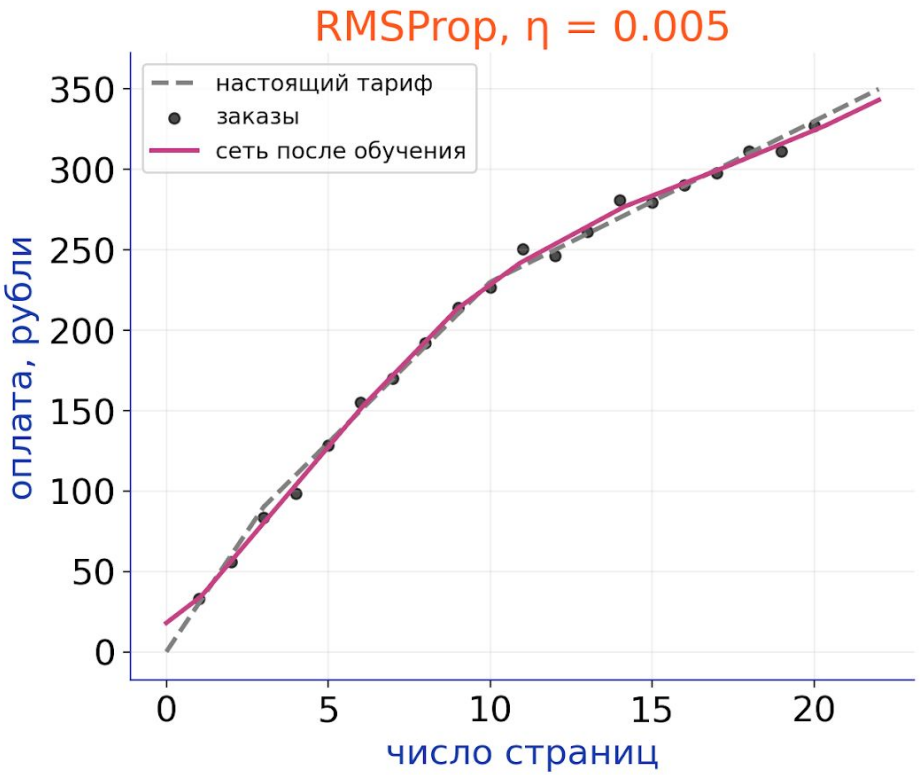
- Сеть, данные и начальные настройки одинаковы.
- Сравниваем получившуюся функцию, а не только число ошибки.

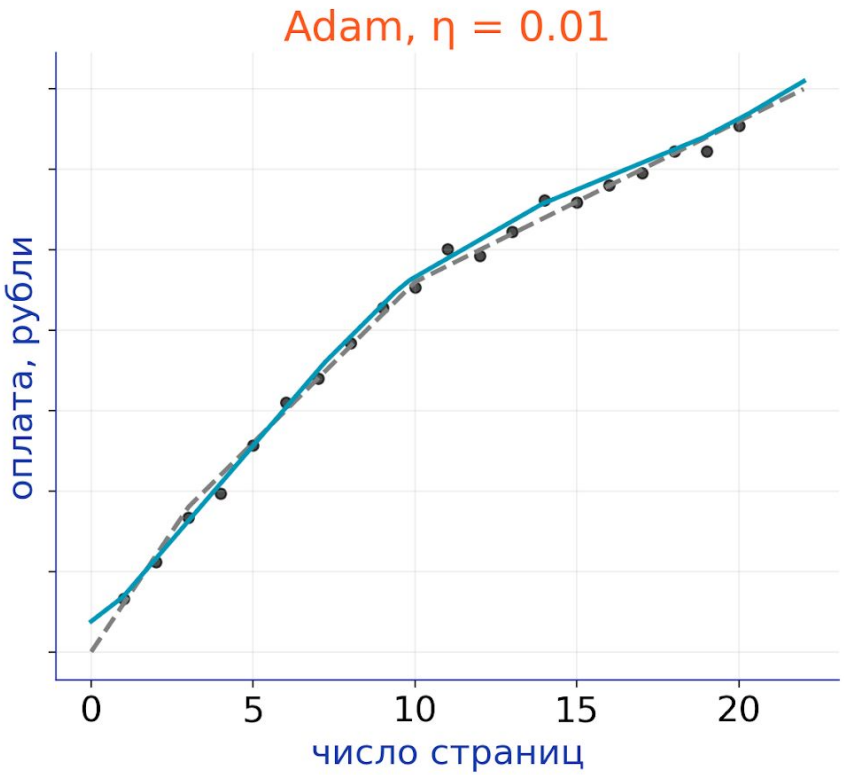




Что выучили RMSProp и Adam

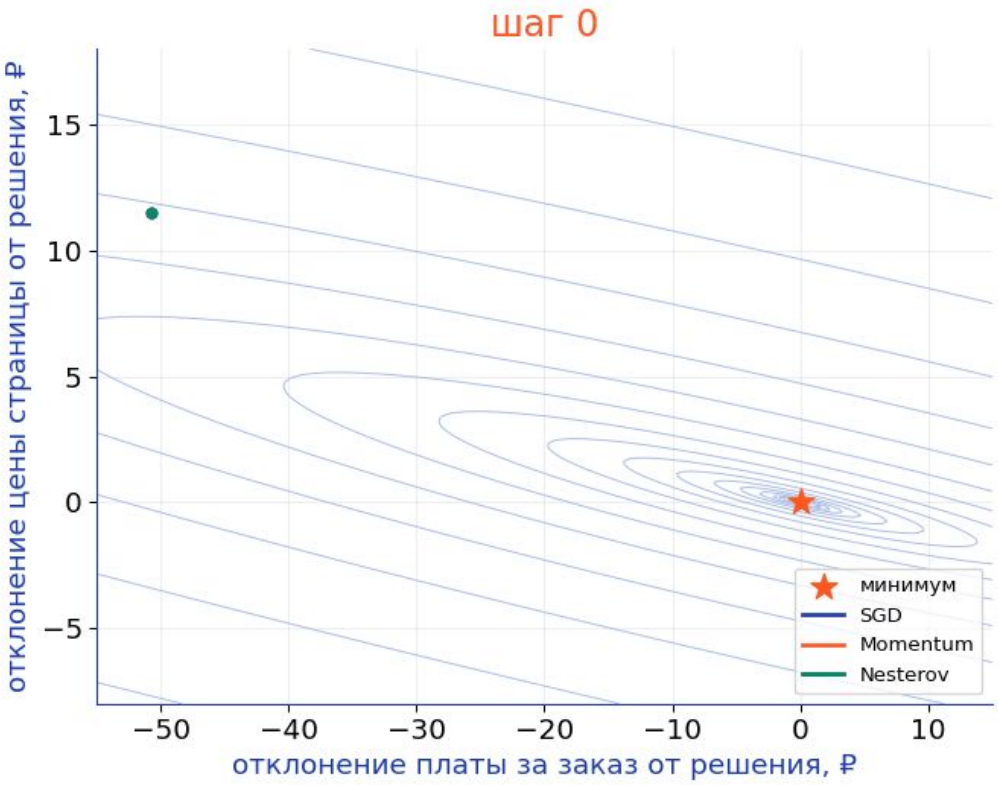
- Даже Adam не обязан выиграть в каждом маленьком примере.
- Скорость, начальная сеть и батчи влияют на результат.





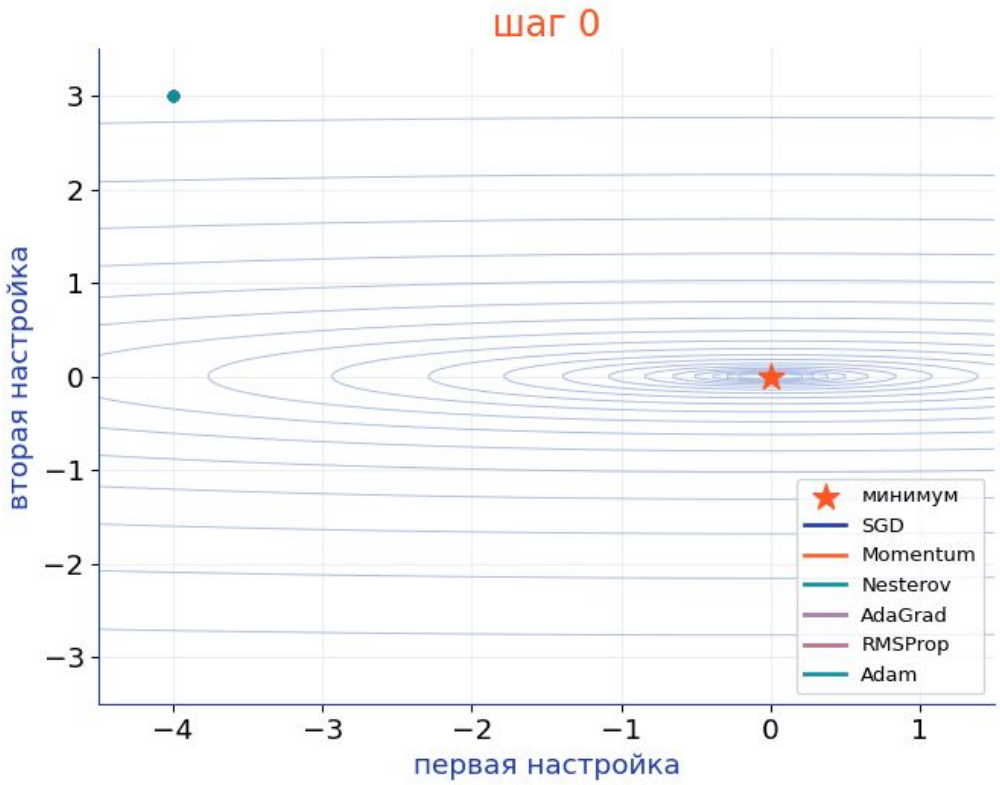
Анимация: движение по долине

- Тот же копицентр, старт и скорость, что в первом примере.
- Наблюдаем, как инерция разгоняет движение вдоль долины.



Шесть методов: анимация

- Чаша и старт одинаковы; скорости указаны в сравнении выше.
- Около минимума RMSProp продолжает колебаться.



Что изменилось по сравнению с SGD

- Момент хранит скорость; Нестеров берёт градиент впереди.
- AdaGrad и RMSProp выбирают покоординатный масштаб шага.
- Adam сочетает память о направлении и масштабе градиента.
- Алгоритм, скорость и батчи надо рассматривать вместе.

Южный федеральный университет

Спасибо за внимание!

@SINECOR



