Стохастическая оптимизация
Алгоритмы для улучшения

Симкин Алексей
НИС МОП 161
28 сентября 2018

Stochastic gradient descent (SGD)

- Обычно, функционал представим в виде:

$$Q(w) = \frac{1}{l}\sum_{i=1}^{l} Q_i(w)$$

- Тогда градиент равен:

$$\nabla_w Q(w) = \sum \nabla_w Q_i(w)$$

- На большой выборке на вычисления тратится слишком много ресурсов.

- При этом шаги градиентного спуска маленькие

SGD

- Значит точность не очень важна
- Вместо градиента будем считать его оценку по случайному слагаемому

$$\nabla_w Q \approx \nabla_w Q_{i_k}(w)$$

- Веса обновляются следующим образом:

$$w^{(k)} = w^{(k-1)} - \eta_\kappa \nabla Q_{i_k}\left(w^{(k-1)}\right)$$

- Для выпуклого гладкого функционала Q с минимумом w^*:

$$Q\left(w^{(k)}\right) - Q(w^*) = O\left(\frac{1}{\sqrt{k}}\right)$$

SGD

- Можно совместить стохастический и обычный градиентные спуски.
- Будем использовать все слагаемые, но на одном шаге пересчитывать только одно.
- Метод называется stochastic average gradient (SAE).
- Считать нужно все еще меньше, чем в обычном градиентном спуске.
- Оценка сходимости такая же: $O\left(\frac{1}{k}\right)$

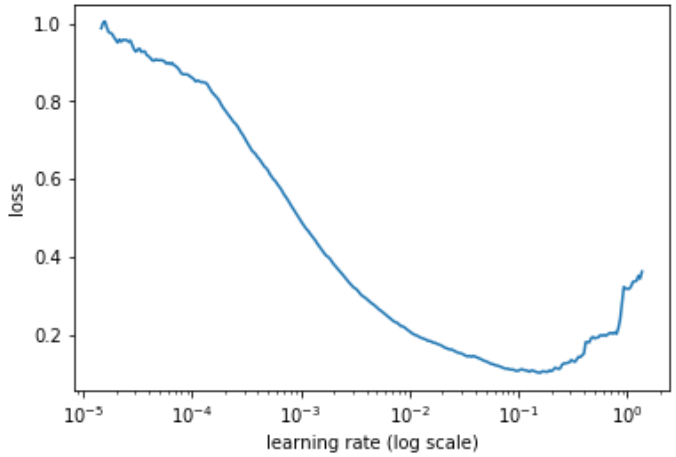
Adaptive learning rate

- Многое зависит от скорости обучения.
- При низком значении lr модель будет обучаться слишком долго.
- При высоком – есть риск пропустить минимум.
- Одним из самых простых способов является выбор learning rate в зависимости от ошибки: чем она ниже – тем ниже ставить lr.

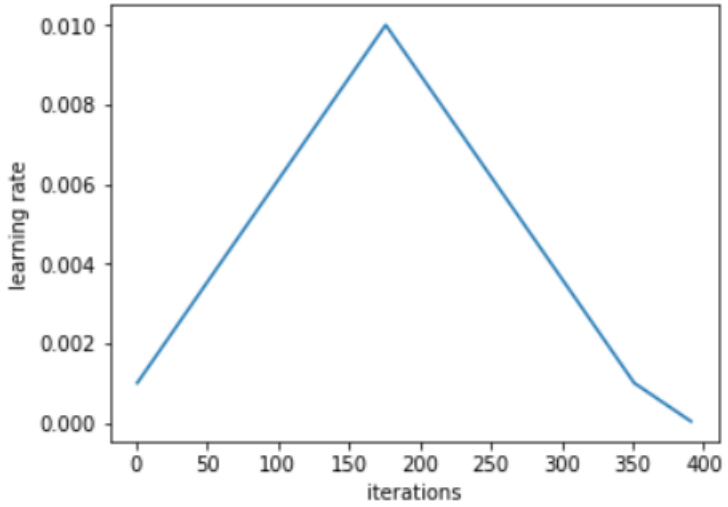
Adaptive learning rate

- Также есть и более сложные вариации на эту тему.
- Например: сначала увеличивать lr, а затем его уменьшать.
- При таком подходе высокий lr может выступать в качестве регуляризатора, помогая избежать переобучения.
- Также с его помощью можно выбраться из локального минимума.

Adaptive learning rate

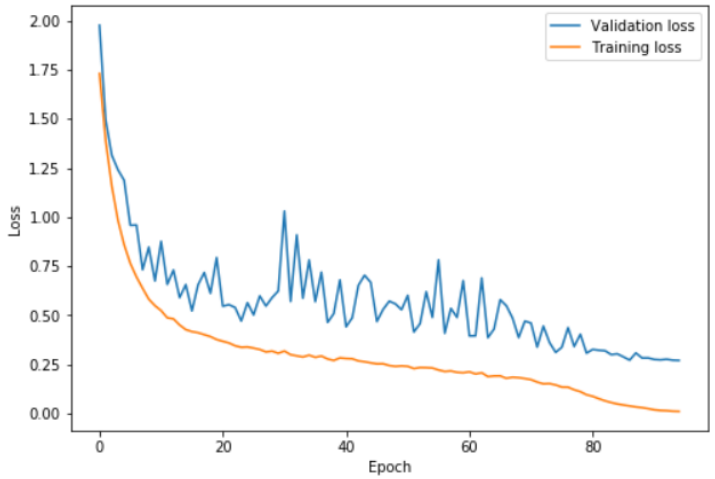


- Запускаем модель с увеличивающимся lr.
- Выбираем lr из того диапазона, где все еще есть улучшение.

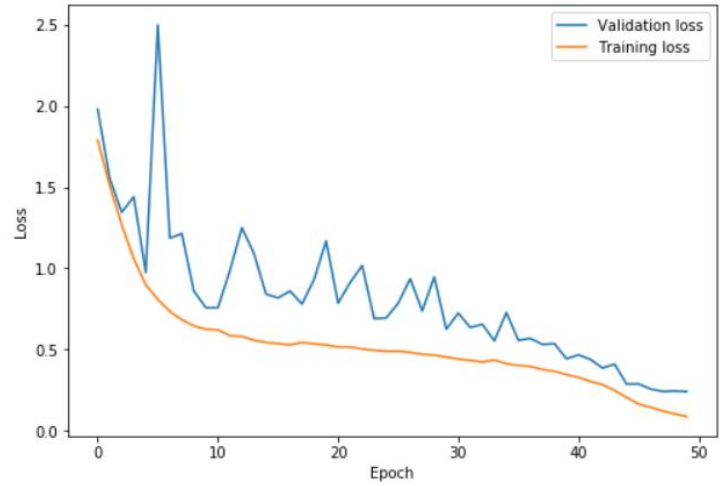


- Ставим lr от минимума к максимуму и обратно.
- Далее еще больше снизим lr

Adaptive learning rate



- lr меняется от 0,08 и 0,8
- Далее еще снижается до 0,0008



- lr меняется от 0,15 до 3 и обратно
- В конце снижается до 0,0015

Momentum

- Иногда, направление градиента может сильно меняться при каждом шаге.
- Чтобы этого избежать, усредним градиенты предыдущих шагов

$$h_0 = 0$$

$$h_k = \alpha h_{k-1} + \eta_k \nabla_w Q\left(w^{(k-1)}\right)$$

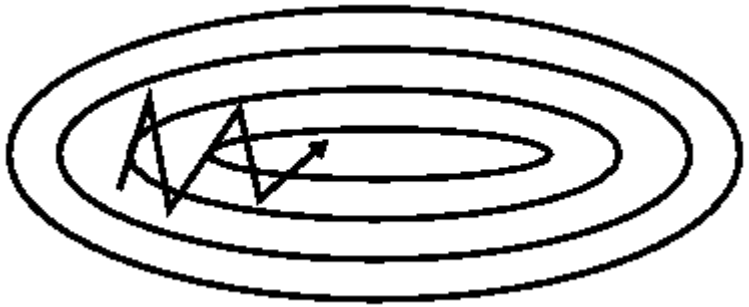
- Для шага просто сдвинемся на вектор инерции:

$$w^{(k)} = w^{(k-1)} - h_\kappa$$

Momentum

SGD without momentum

SGD with momentum



Nesterov accelerated gradient

- Посчитаем оценку среднего градиента.

$$v_k = \gamma v_{k-1} + (1-\gamma)\eta \nabla_w Q\left(w^{(k-1)}\right)$$

- При этом, сначала сдвинемся по вектору:

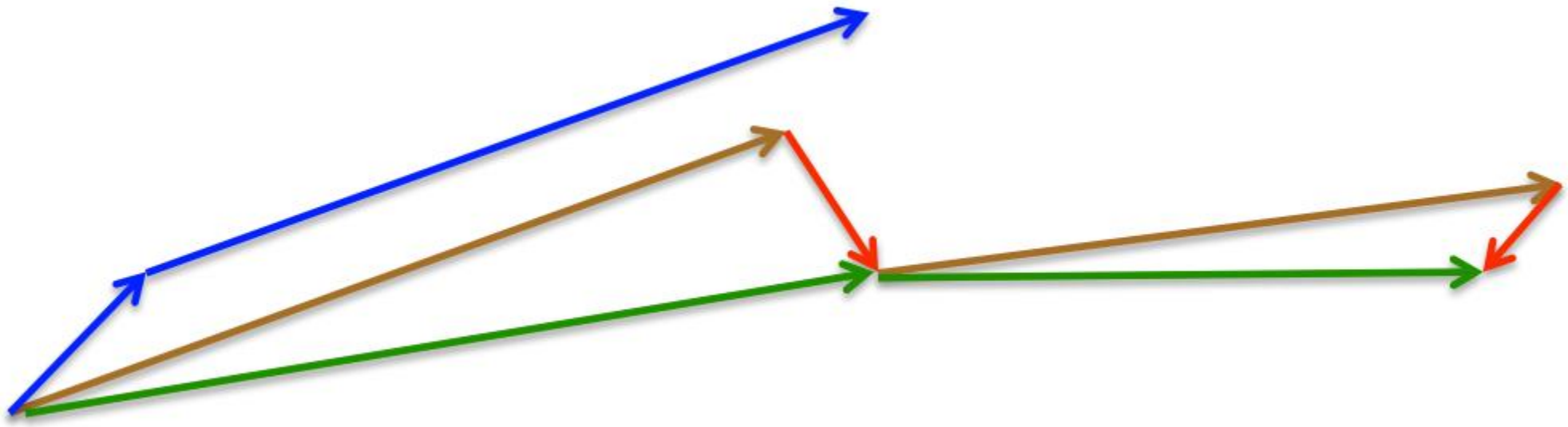
$$v_k = \gamma v_{t-1} + (1-\gamma)\eta \nabla_w Q\left(w^{(k-1)} - \gamma v_{k-1}\right)$$

- А затем посчитаем веса:

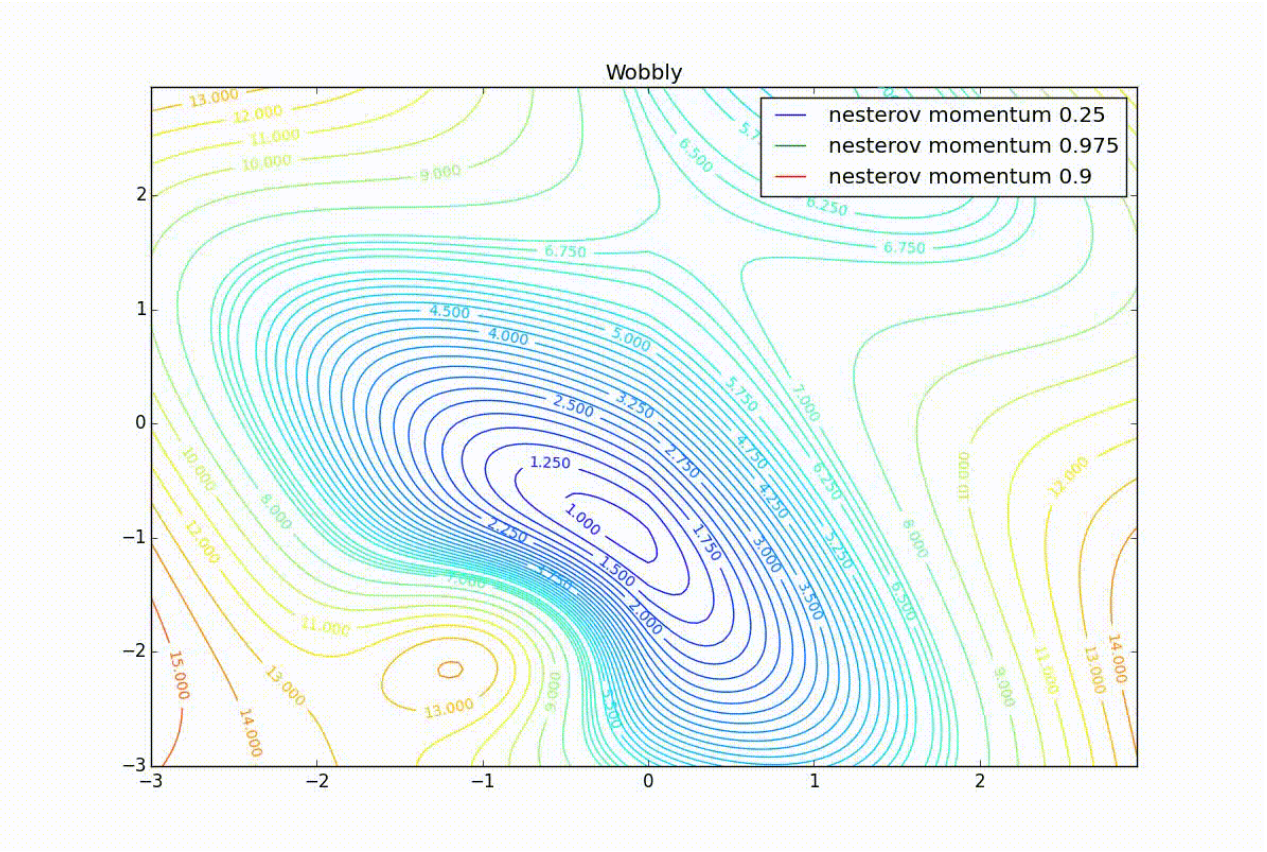
$$w^{(k)} = w^{(k-1)} - v_k$$

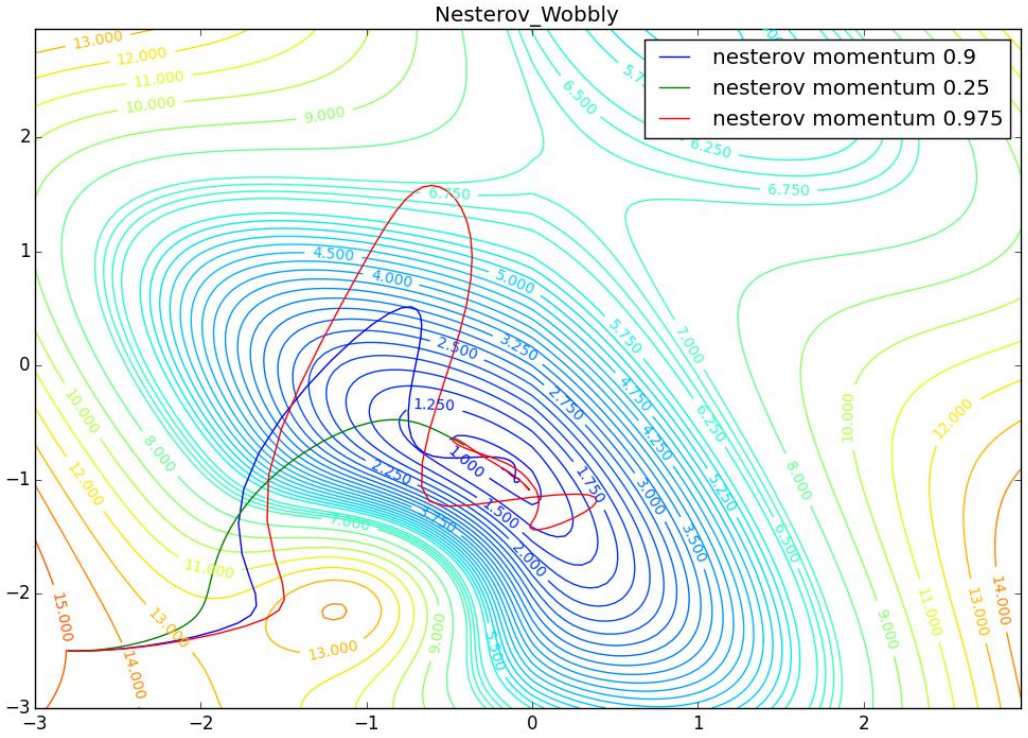
Nesterov accelerated gradient

- Такой подсчет позволяет двигаться быстрее, если там, куда мы движемся, производная больше.
- И медленнее, если это не так.
- Однако, при больших γ и η можно оказаться в слишком далеко, и уйти от точки минимума.

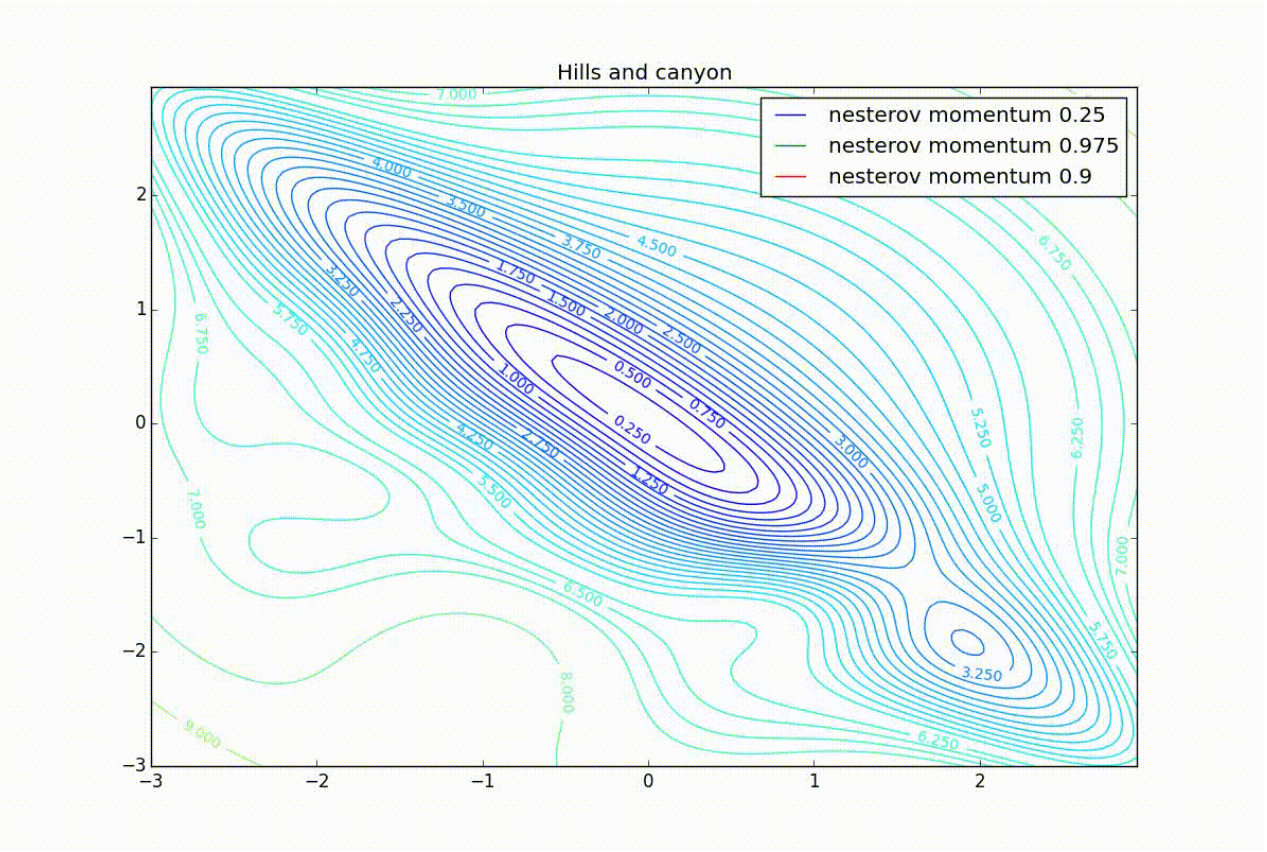


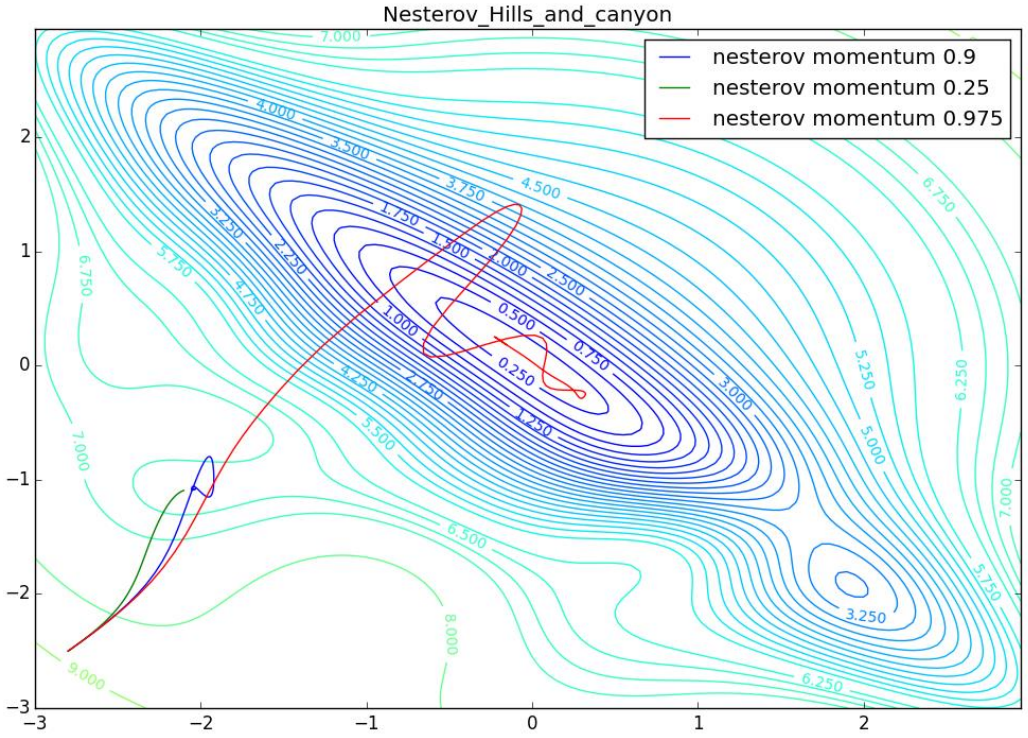
Nesterov accelerated gradient



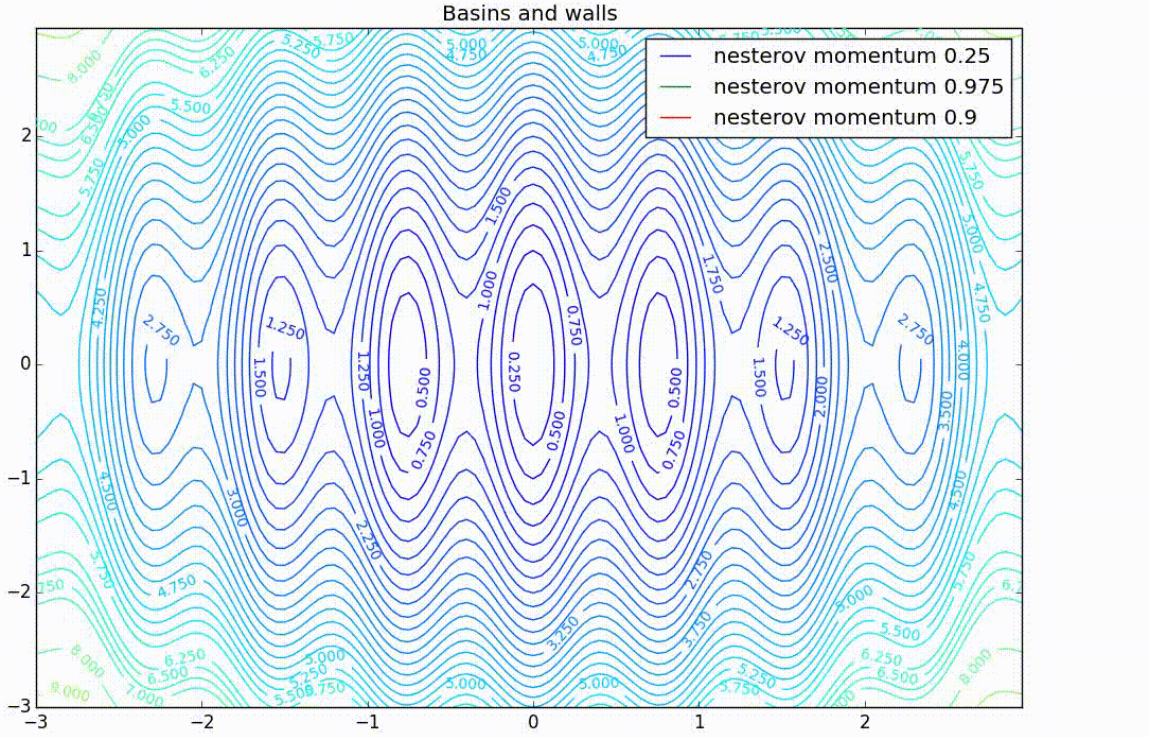


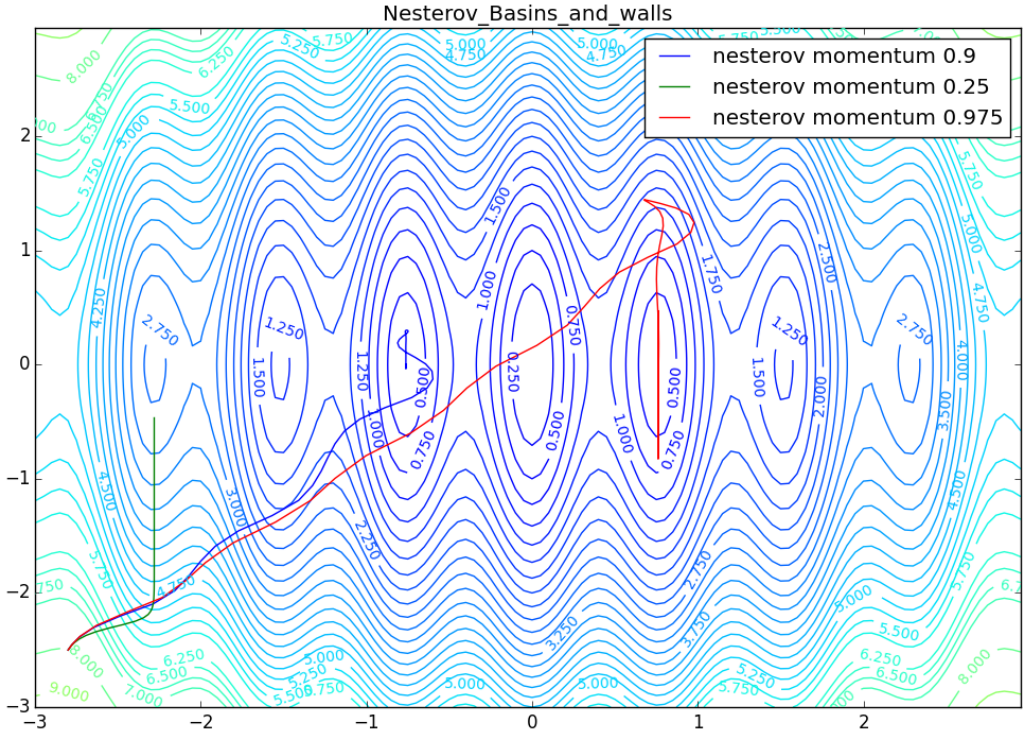
Nesterov accelerated gradient





Nesterov accelerated gradient





AdaGrad

- Подобрать длину шага для градиентного спуска сложно.
- Это будет либо слишком долго
- Либо можно пропустить точки минимума
- Также, некоторые важные признаки могут редко встречаться.

AdaGrad

- Для каждой компоненты можно хранить длину шага и выбирать свою длину шага, исходя из этого.
- При этом, чем быстрее мы двигались раньше, тем медленнее будут новые шаги, и наоборот

$$G_{k_j} = G_{k-1_j} + \left(\nabla_w Q\left(w^{(k-1)}\right)\right)_j^2$$

$$w_j^{(k)} = w_j^{(k-1)} - \frac{\eta_t}{\sqrt{G_k + \varepsilon}} \left(\nabla_w Q\left(w^{(k-1)}\right)\right)_j$$

RmsProp

- Но есть проблема.
- G_{k_j} монотонно растет.
- Поэтому можно остановиться до того, как попадем в минимум функционала.

RmsProp

- Предлагается использовать экспоненциальное затухание градиентов:

$$G_{k_j} = \alpha G_{k-1_j} + (1-\alpha)\left(\nabla_w Q\left(w^{(k-1)}\right)\right)_j^2$$

$$w_j^{(k)} = w_j^{(k-1)} - \frac{\eta_t}{\sqrt{G_k + \varepsilon}}\left(\nabla_w Q\left(w^{(k-1)}\right)\right)_j$$

Adam (Adaptive moment estimation)

- Несколько идей можно использовать сразу.
- Например, идею накопления градиента и идею учета частоты признака
- Формула для момента:

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k$$

- Формула для длины шага:

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2) g_k^2$$

Adam

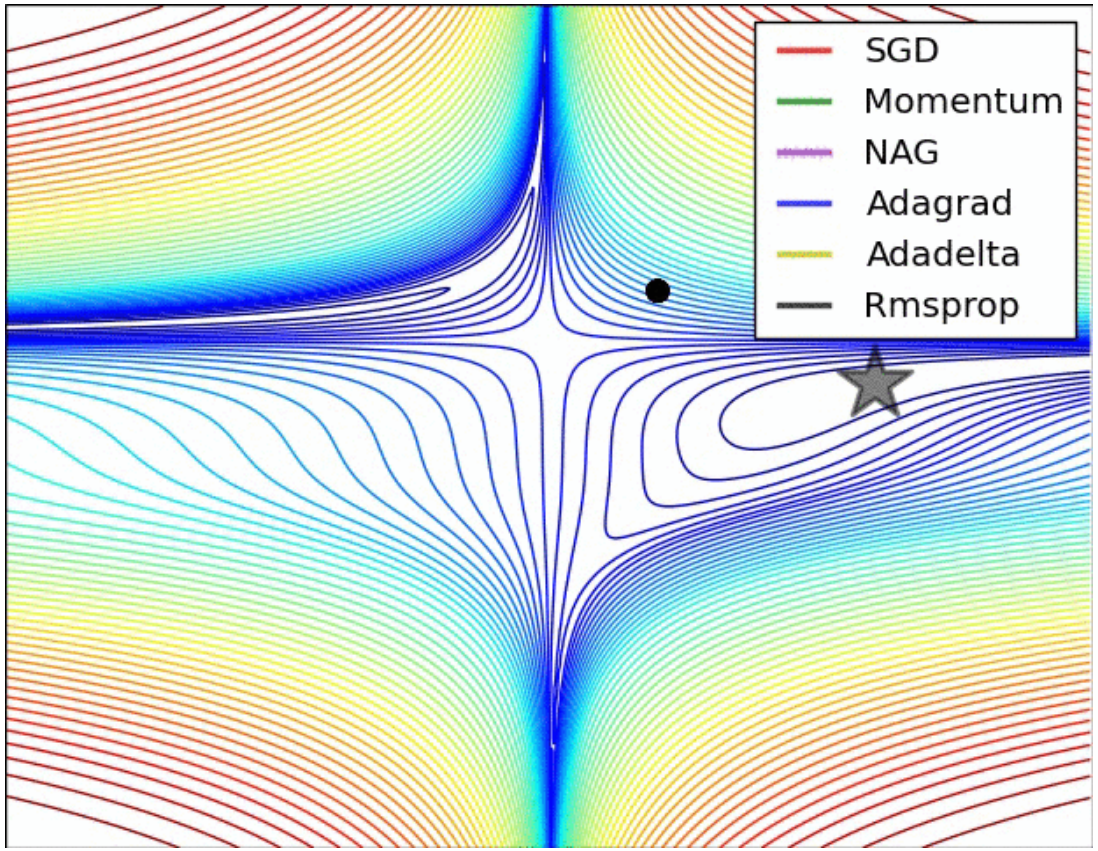
- При слишком маленьких начальных m и v алгоритм будет работать неэффективно какое-то время.
- Чтобы не подбирать эти параметры, предлагается увеличить их так:

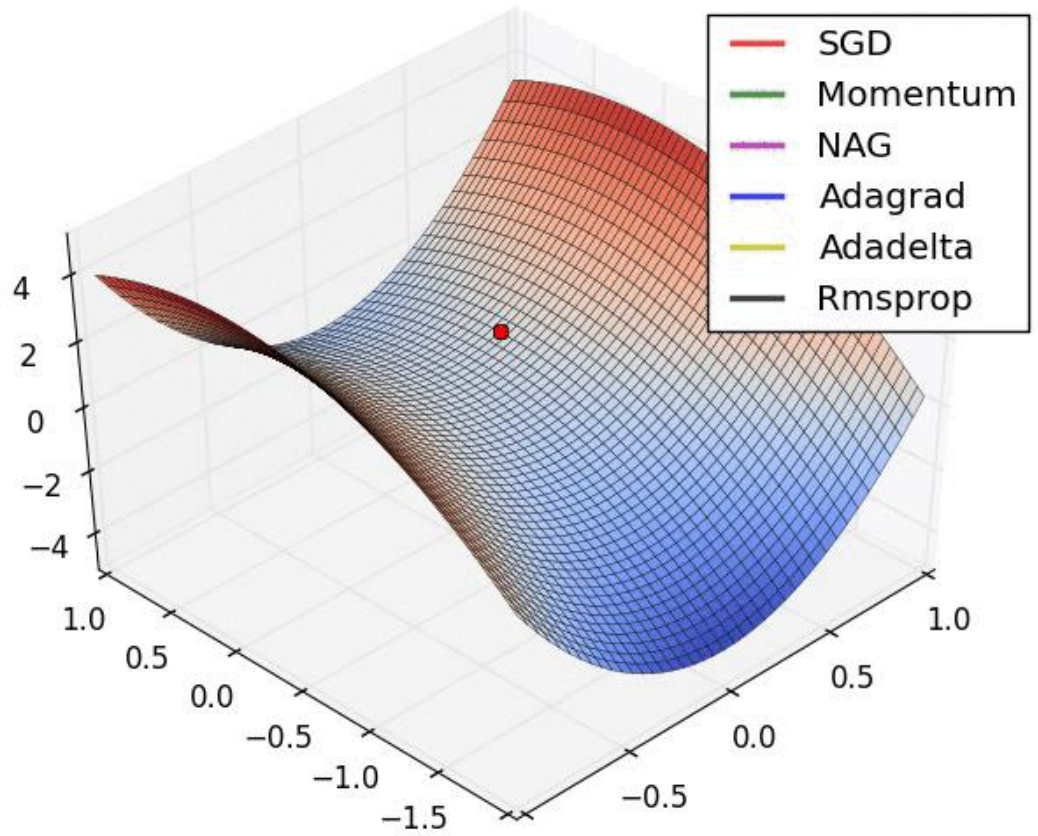
$$\hat{m}_k = \frac{m_k}{1-\beta_1^k}, \hat{v}_k = \frac{v^k}{1-\beta_2^k}$$

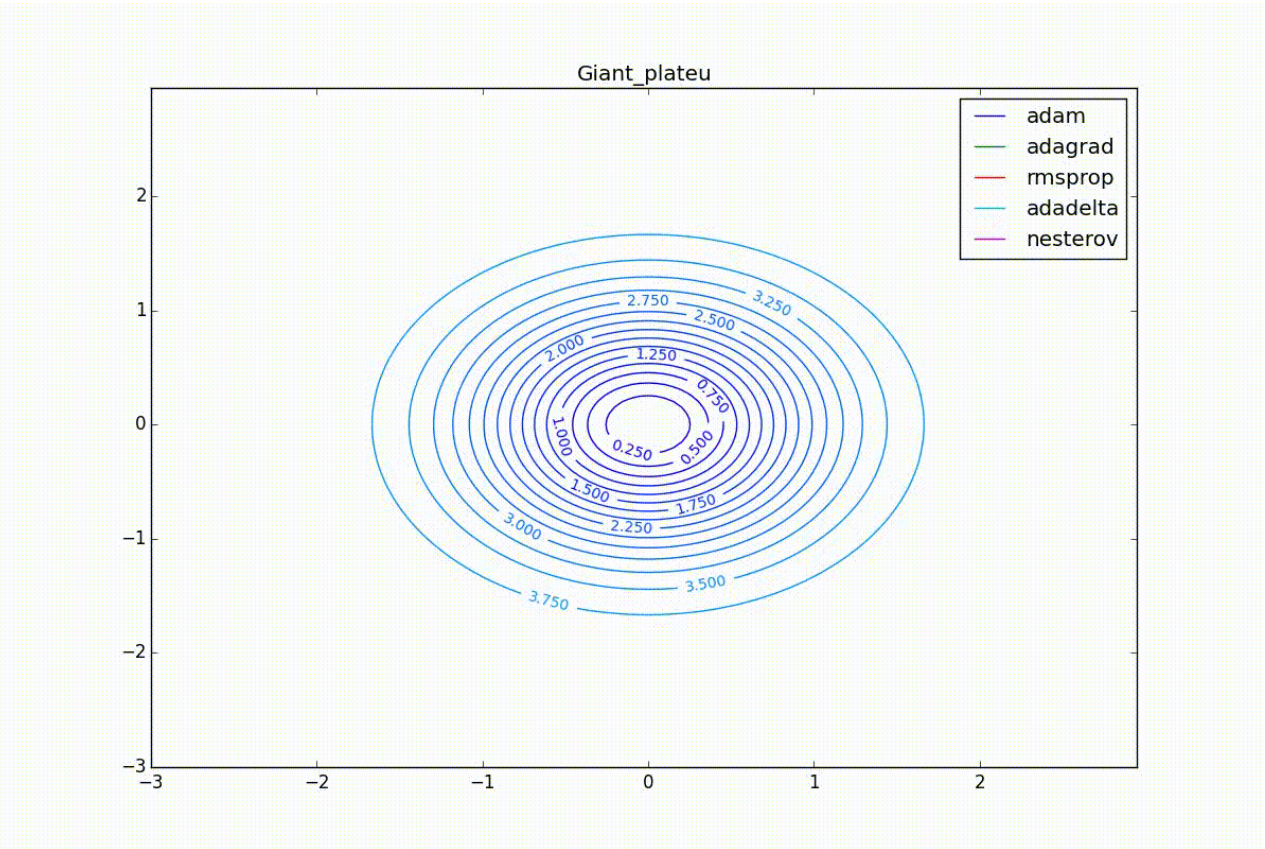
- Тогда новые веса будут равны:

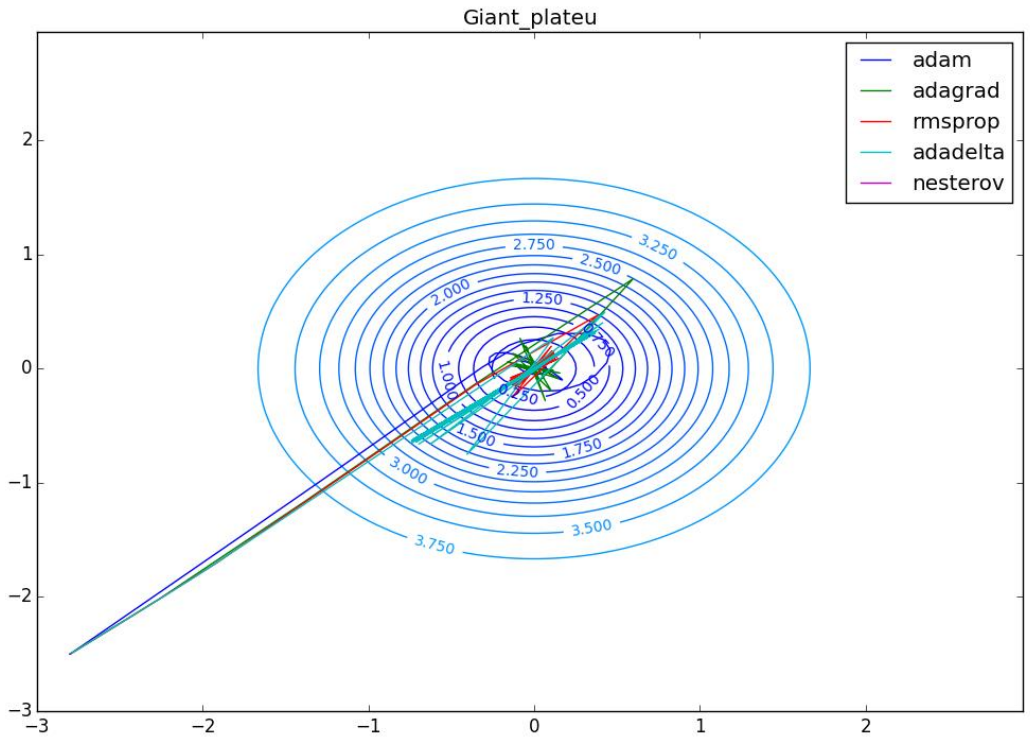
$$w^{(k)} = w^{(k-1)} - \frac{\eta}{\sqrt{\hat{v}_k + \varepsilon}} \hat{m}_k$$

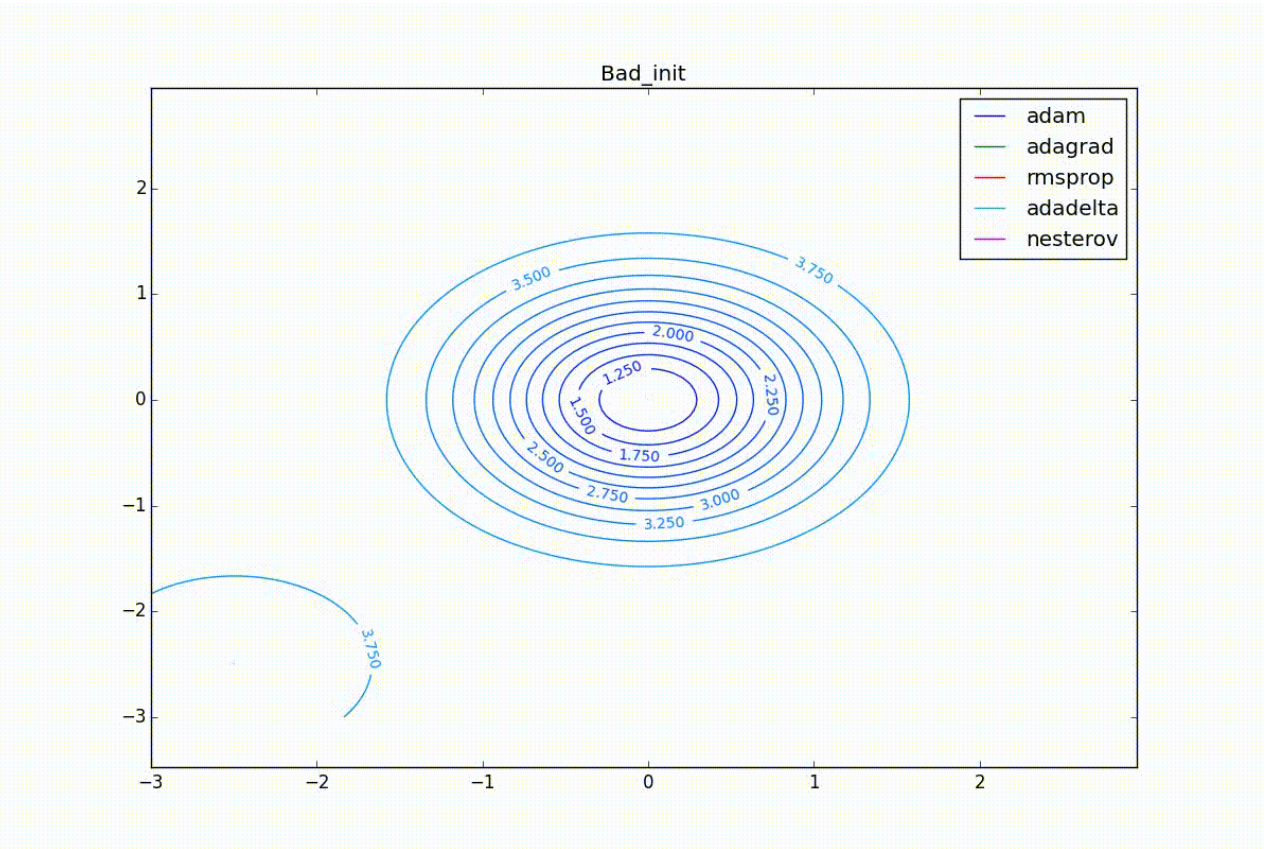
Примеры работы

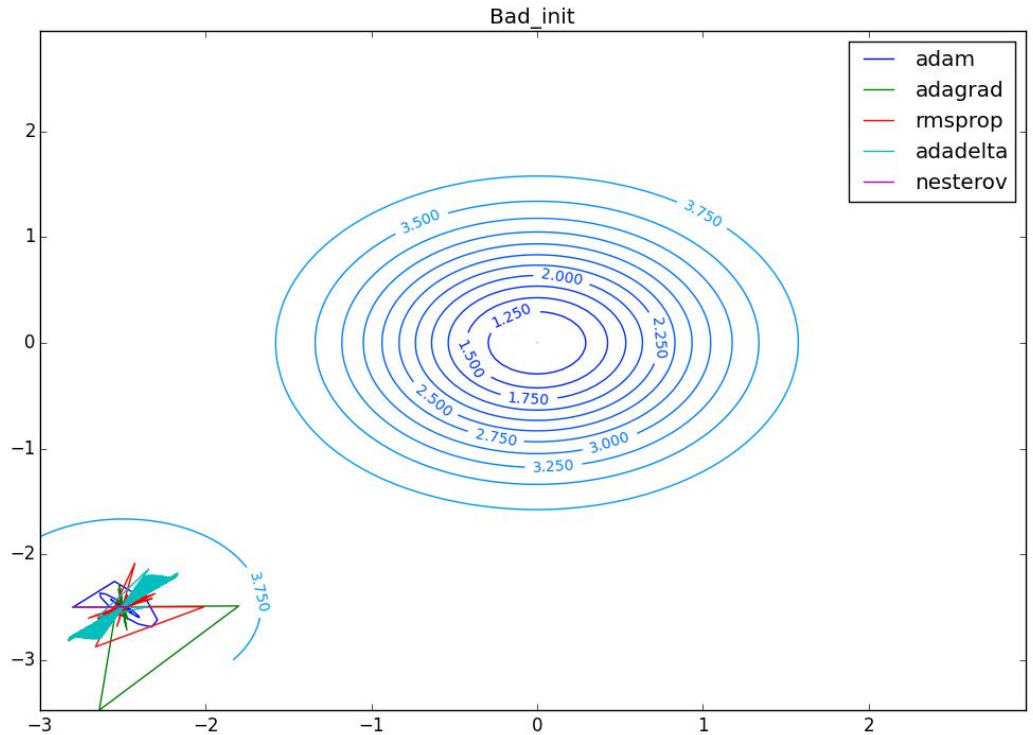

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

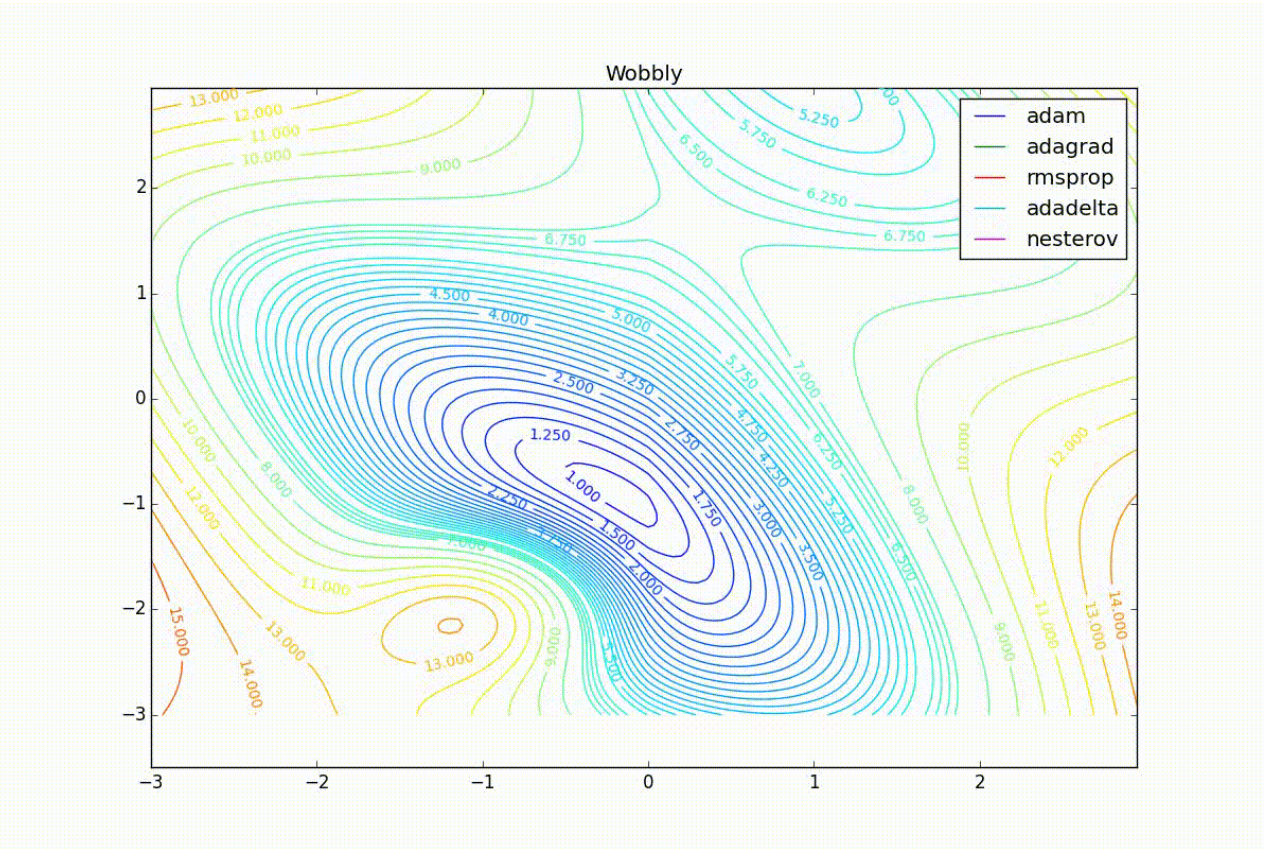

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

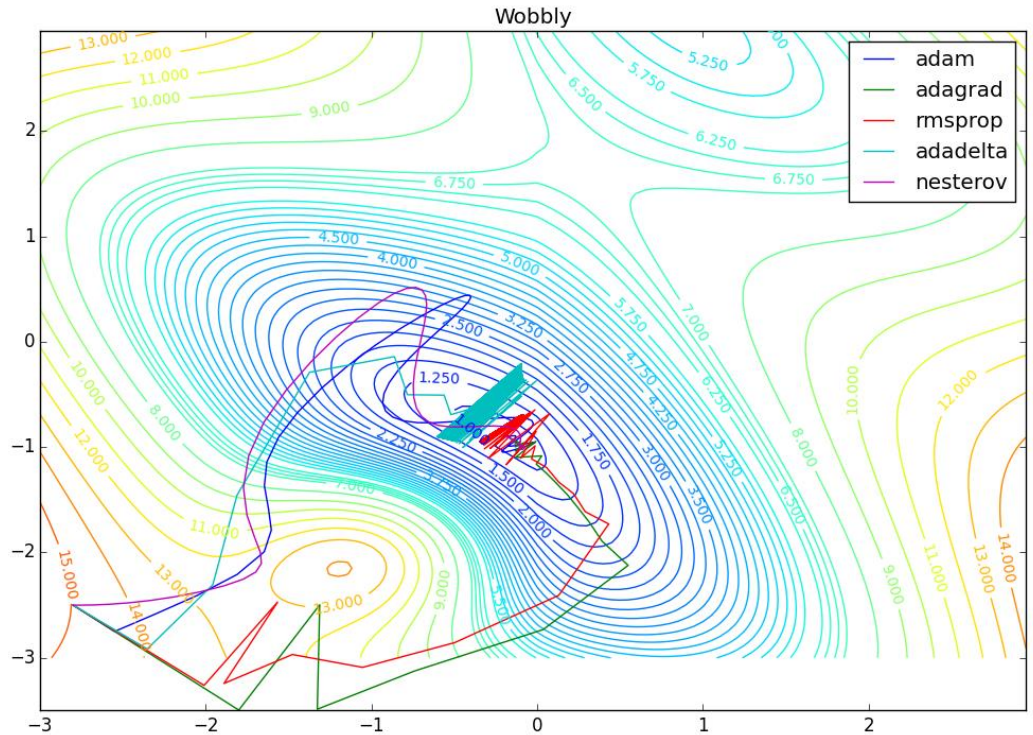

Giant_plateu
adam
adagrad
rmsprop
adadelta
nesterov
0.250
0.500
0.750
1.000
1.250
1.500
1.750
2.000
2.250
2.500
2.750
3.000
3.250
3.500
3.750

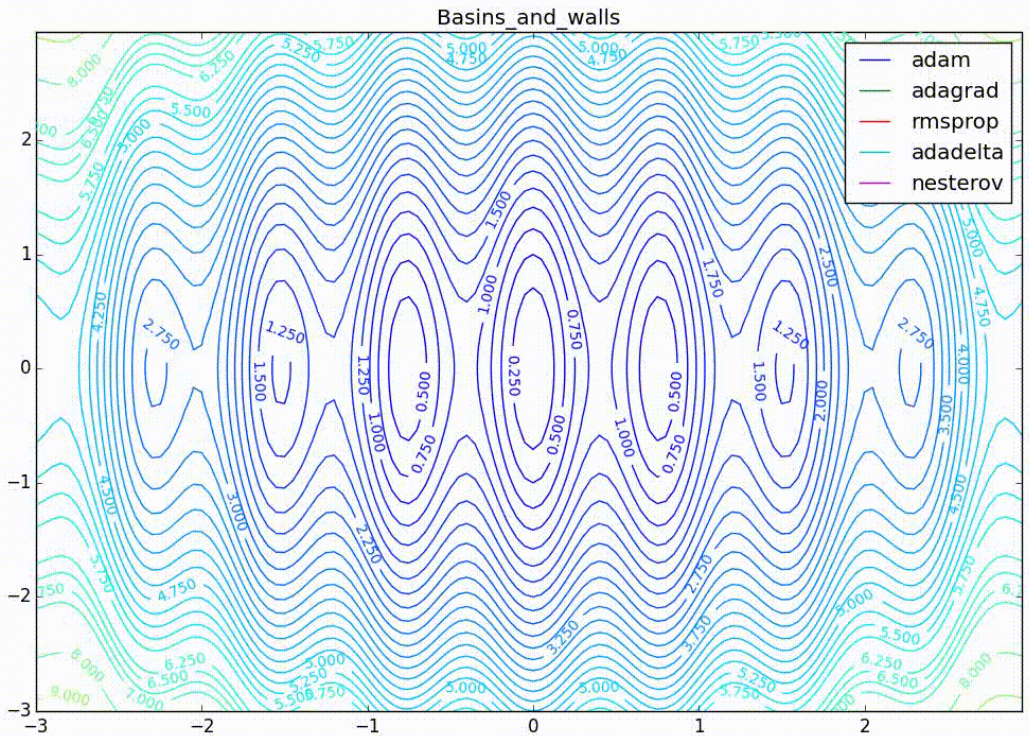

Giant_plateu
adam
adagrad
rmsprop
adadelta
nesterov
0.250
0.500
0.750
1.000
1.250
1.500
1.750
2.000
2.250
2.500
2.750
3.000
3.250
3.500
3.750

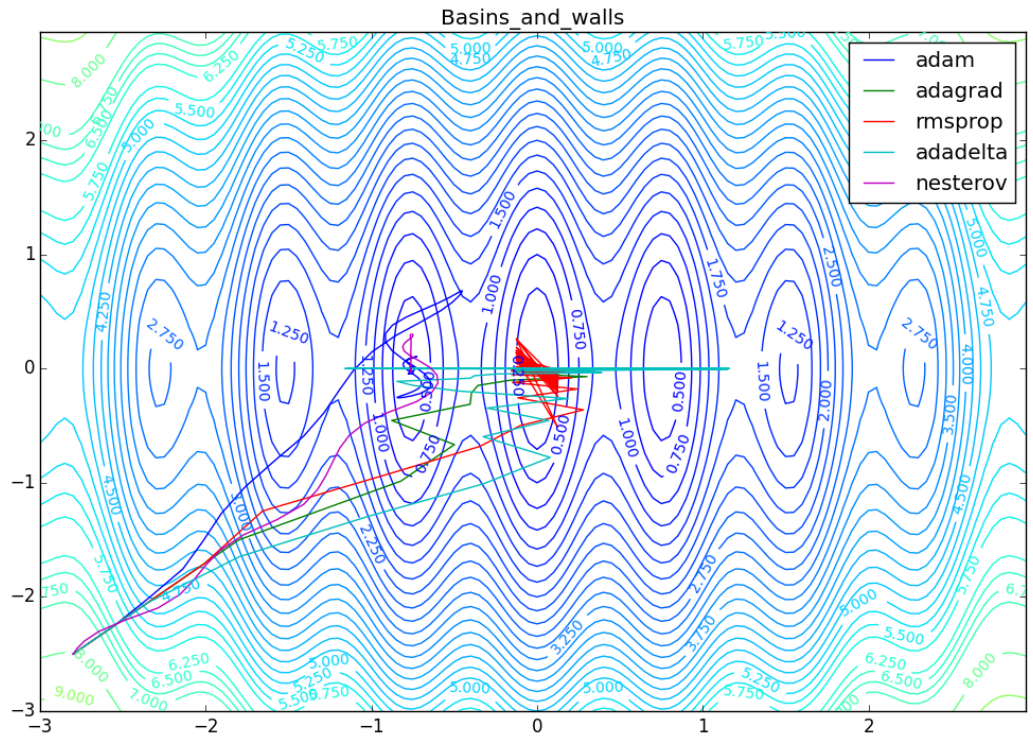

Bad_init
adam
adagrad
rmsprop
adadelta
nesterov
3.500
3.750
2.000
1.250
2.250
1.500
1.750
2.500
2.750
3.000
3.250
3.750
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2

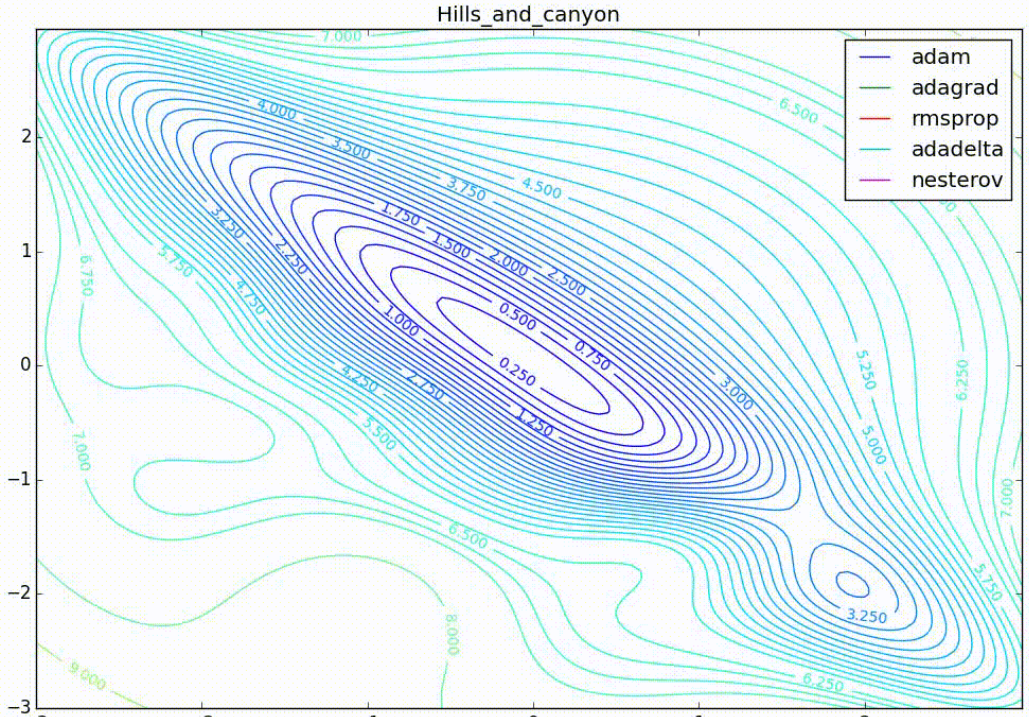

Bad_init
adam
adagrad
rmsprop
adadelta
nesterov
3.500
3.750
2.000
1.250
2.250
1.500
1.750
2.500
2.750
3.000
3.250
3.750
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2

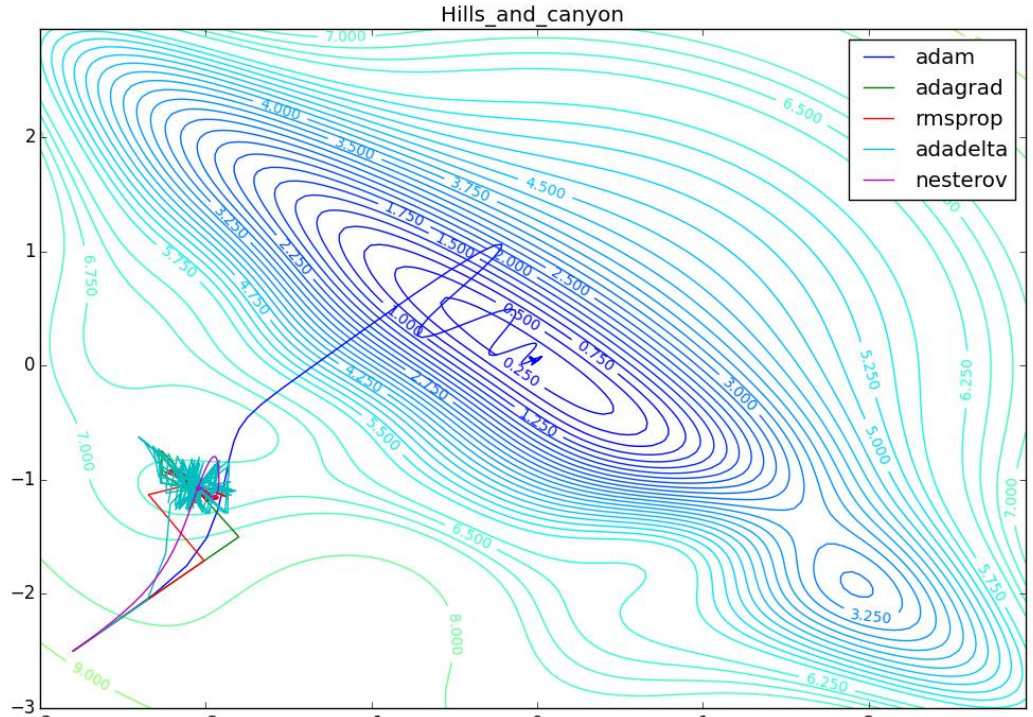

Wobbly
adam
adagrad
rmsprop
adadelta
nesterov


Wobbly
adam
adagrad
rmsprop
adadelta
nesterov


Basins_and_walls
adam
adagrad
rmsprop
adadelta
nesterov


Basins_and_walls
adam
adagrad
rmsprop
adadelta
nesterov


Hills_and_canyon
adam
adagrad
rmsprop
adadelta
nesterov


Hills_and_canyon
adam
adagrad
rmsprop
adadelta
nesterov

Ссылки

[1] An overview of gradient descent optimization algorithms

http://ruder.io/optimizing-gradient-descent/

[2] The 1cycle policy

https://sgugger.github.io/the-1cycle-policy.html#the-1cycle-policy