یادگیری عمیق
بهینه‌سازی ۲

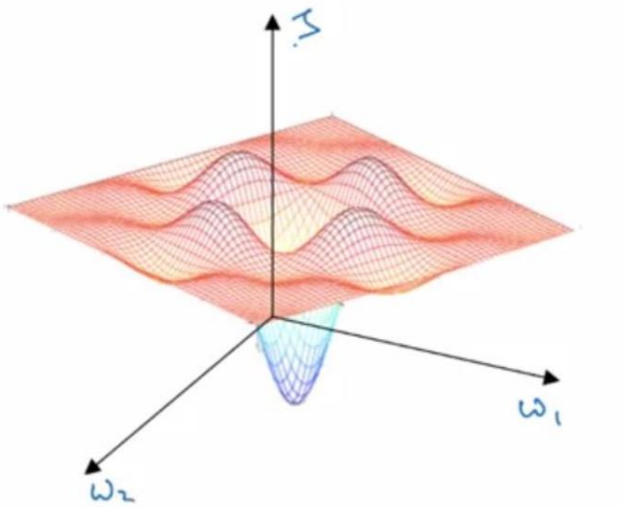
optimization²

دانشگاه شهید بهشتی
بهار ۱۴۰۱
احمد محمودی ازناوه

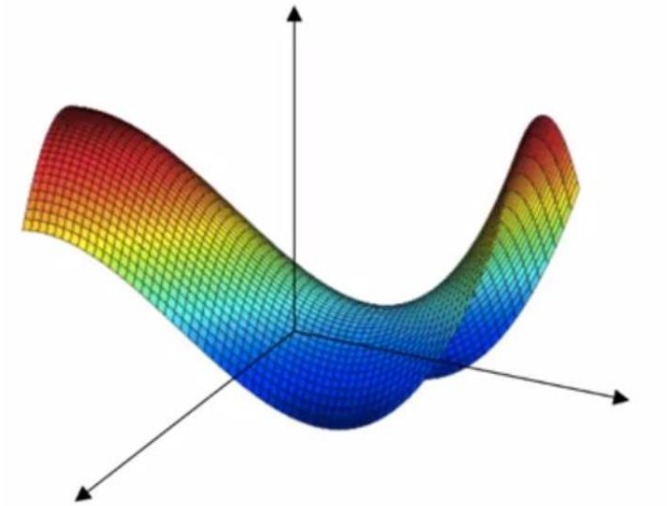
http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

- افزایش سرعت بهینه‌سازی



What about very high-dimensional spaces?



For high dimensional parameter space, most points of zero gradients are not local optima and are saddle points.

Andrew Ng,Deep learning specialization

یادآوری

Gradient descent variants

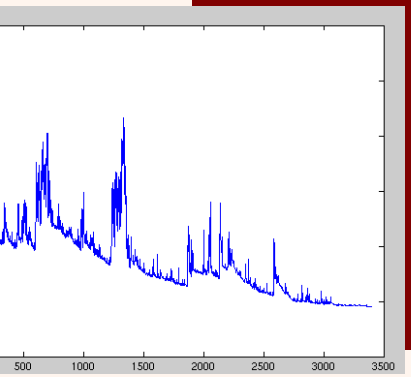
Batch gradient descent (Vanilla gradient descent)

- لازم است برای کل دسته‌داده گرادیان محاسبه و سپس وزن‌ها به روز شود.
- سرعت به روز شدن پایین است.
- برای موارد آنلاین مناسب نیست.
- امکان انجام محاسبات به صورت موازی

Stochastic gradient descent(SGD)

Sequential

- به روز شدن برای هر نمونه آموزشی است(در به روز شدن از تقریب گرادیان استفاده می‌شود). با کاهش تدریجی نرخ آموزش شبیه روش دسته‌ای عمل می‌کند.
- در مدل Batch افزونگی محاسبات وجود دارد.
- در SGD به روز رسانی برای هر نمونه است و به همین دلیل سرعت بالایی دارد
- برای موارد آنلاین مناسب است.
- به دلیل واریانس زیاد پارامترها در روند به روزرسانی نوسانات زیاد دیده می‌شود.



Mini-batch gradient descent

- از مزایای دو روش قبل استفاده می‌شود.
- واریانس به روزرسانی پارامترها کاهش می‌یابد
- معمولا تعداد نمونه ها در هر batchبین ۵۰ الی ۲۵۶ لحاظ می‌شود.

چالش‌های موجود

- تنظیم نرخ آموزش:
  - مقداردهی اولیه نرخ آموزش
  - زمان‌بندی برای کاهش تدریجی
  - نرخ آموزش متغیر برای وزن‌های مختلف
- متوقف شدن در نقاط زینی

Gradient descent optimization algorithms

آیا می‌توان به طریقی میزان نوسانات در رسیدن به نقطه‌ی بهینه را کاهش داد؟

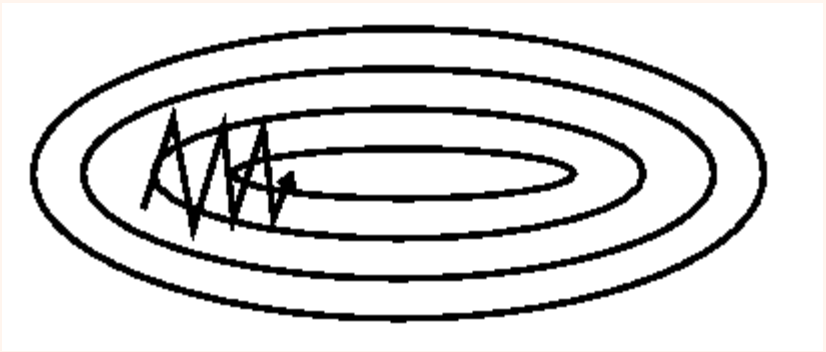
- Momentum
- Nesterov accelerated gradient
- Adagrad
- Adadelta
- RMSprop
- Adam
- Nadam
- AMSGrad

Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

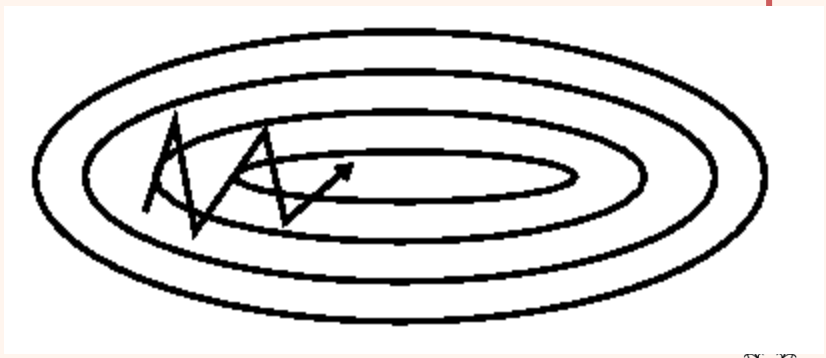
https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

Momentum

- اثر تغییرات وزن در مراحل پیشین در به روزرسانی موثر است.
- اگر تغییرات هم جهت باشد باعث تسریع فرآیند آموزش است
- در غیر این صورت فاکتوری کنترل کننده خواهد بود



SGD without momentum



SGD with momentum

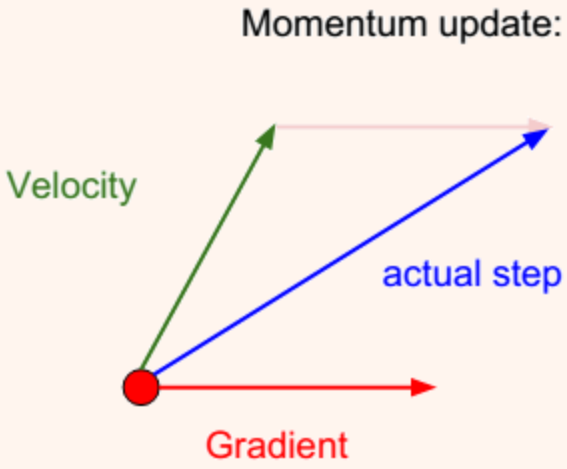
$$\Delta w_{ji}^{l}(n) = \alpha \Delta w_{ji}^{l}(n-1) - \eta \frac{\partial E(n)}{\partial w_{ji}^{l}(n)}$$

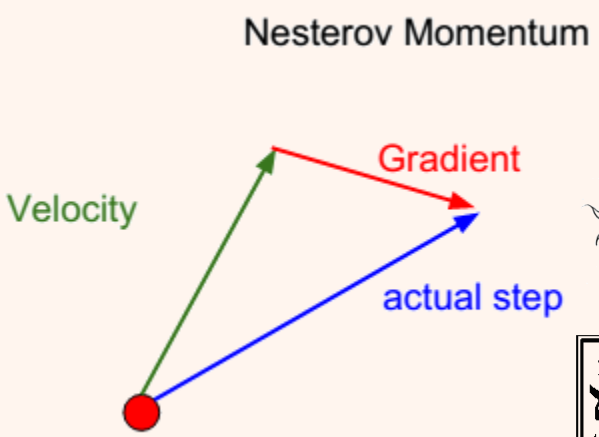
$$w_{ji}^{l}(n+1) = w_{ji}^{l}(n) - \Delta w_{ji}^{l}(n)$$

Nesterov accelerated gradient

Nesterov Accelerated Gradient (NAG), is to measure the gradient of the cost function not at the local position but slightly ahead in the direction of the momentum

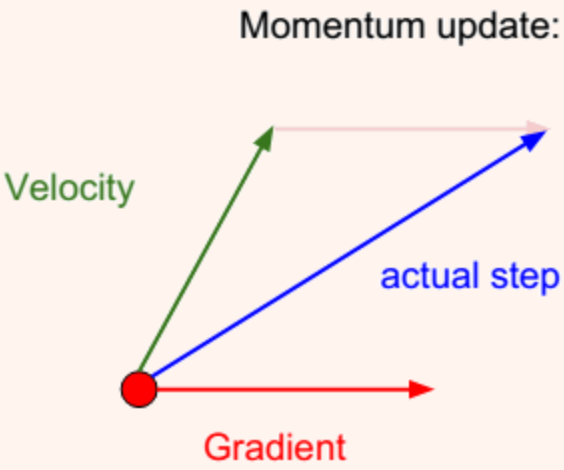
در این شیوه ابتدا بر اساس بردار سرعت موقعیت اصلاح می‌شود و سپس تغییرات گرادیان اعمال می‌شود.

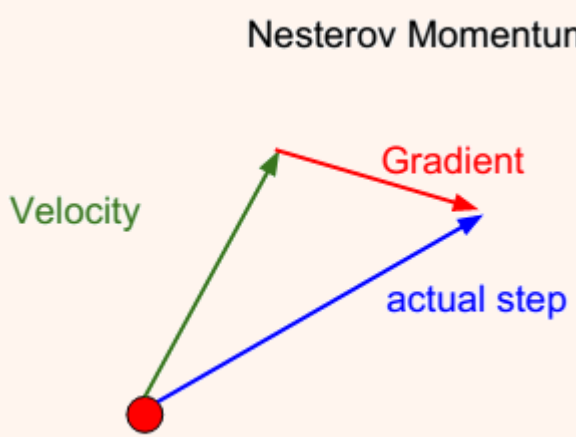


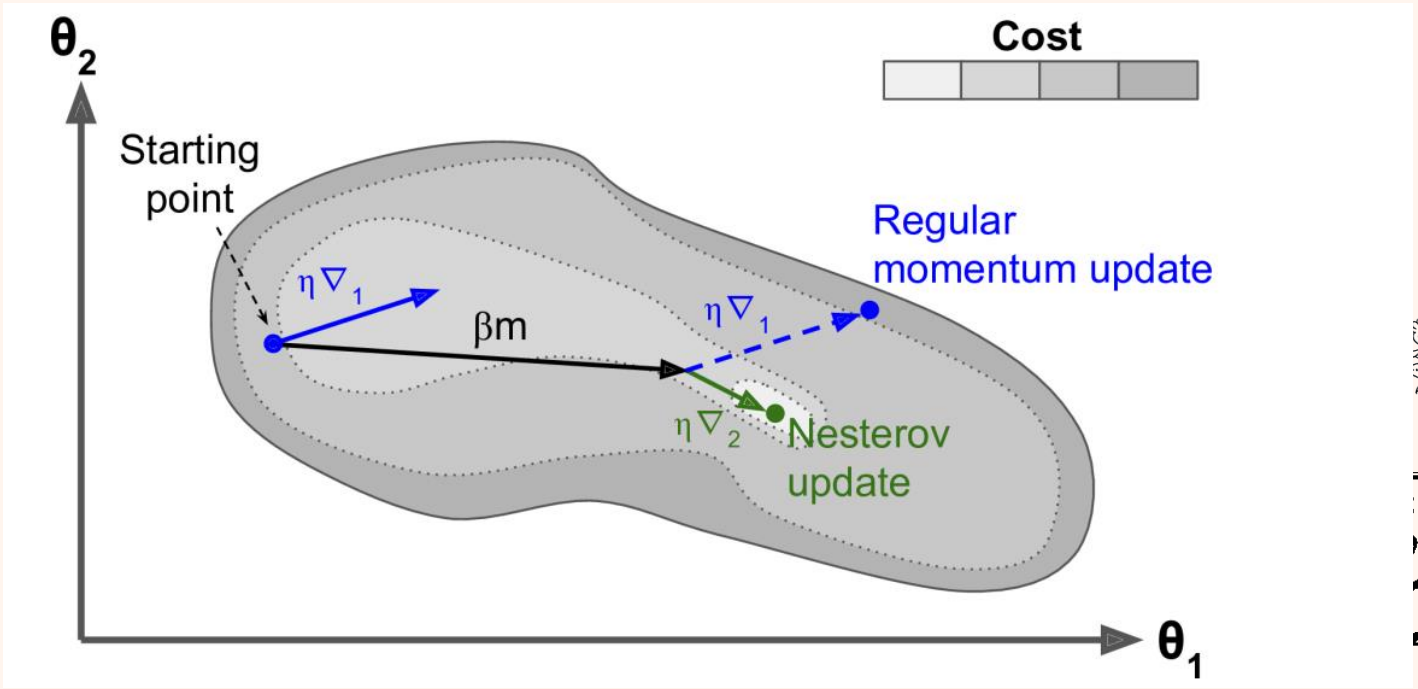


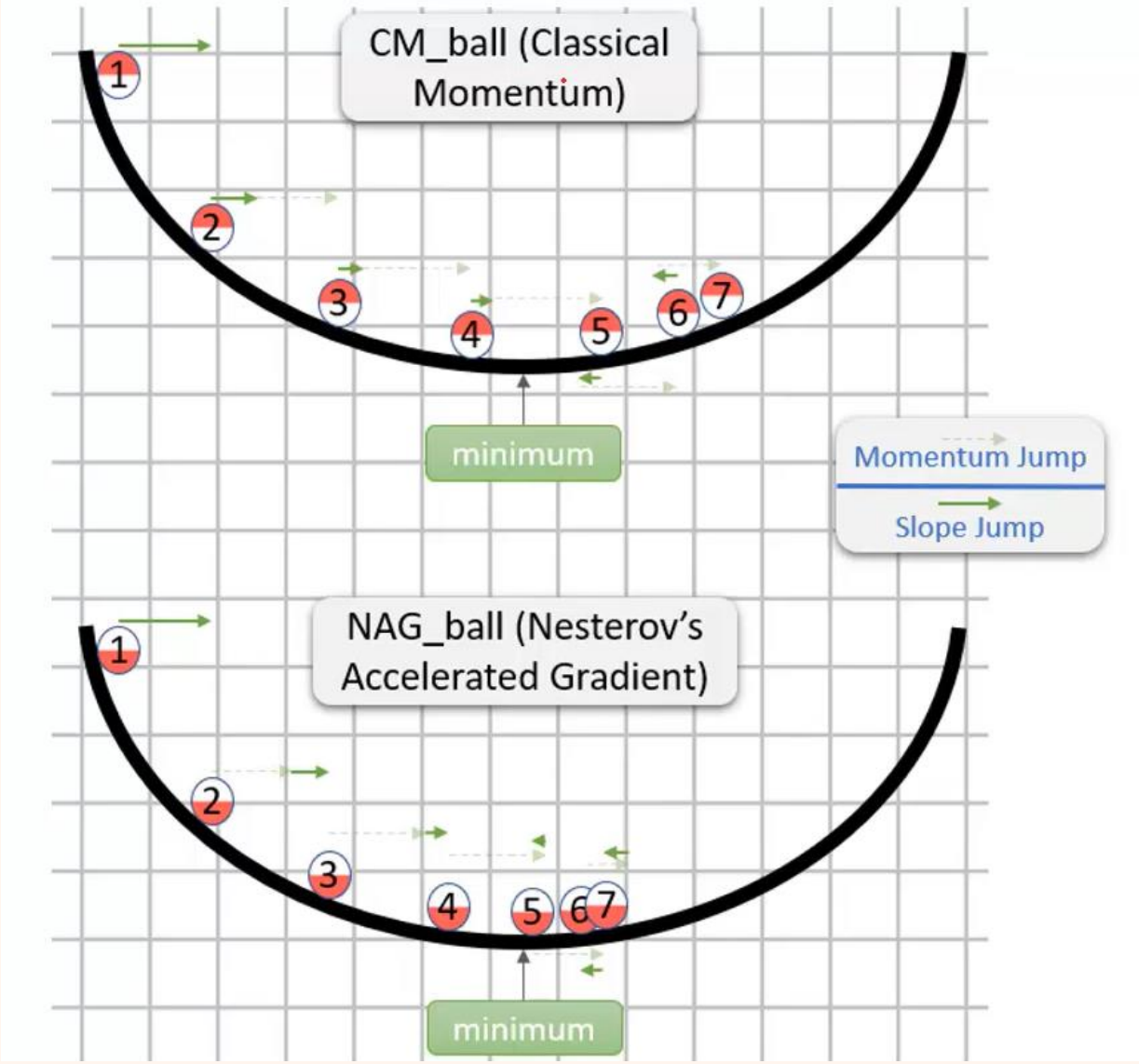
Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2). Doklady an ussr.

```
optimizer = keras.optimizers.SGD(lr=0.001, momentum=0.9, nesterov=True)
```

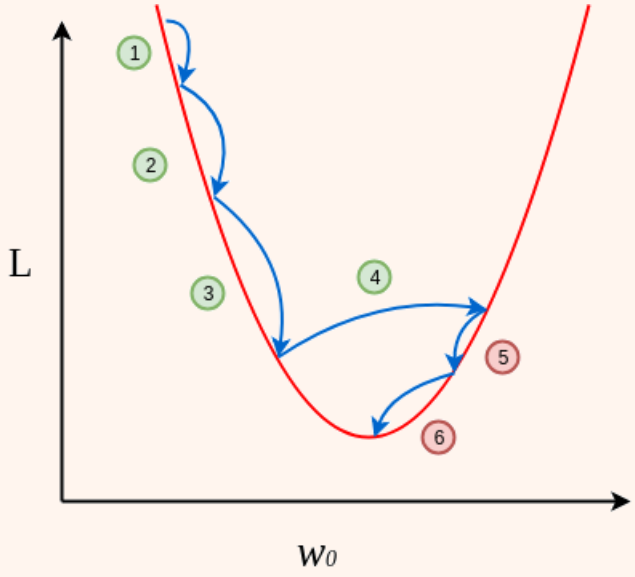




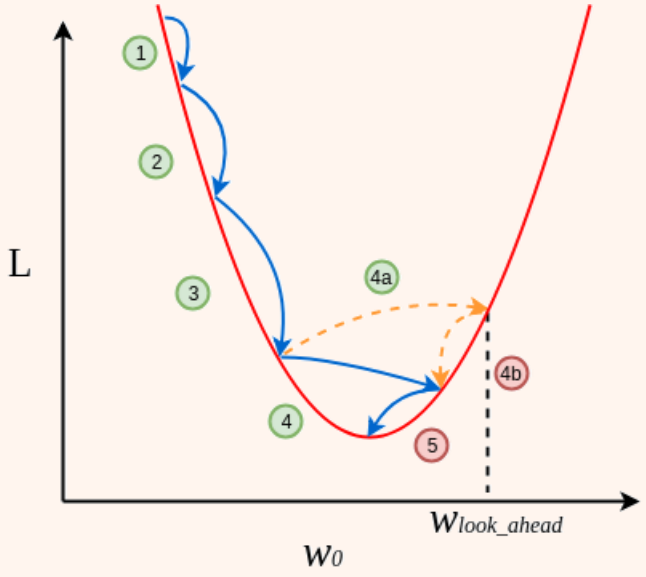



CM_ball (Classical Momentum)
1
2
3
4
5
6
7
minimum
Momentum Jump
Slope Jump
NAG_ball (Nesterov's Accelerated Gradient)
1
2
3
4
5
6
7
minimum

مقایسه



(a) Momentum-Based Gradient Descent



(b) Nesterov Accelerated Gradient Descent

شیوه‌های بهینه‌سازی

SGD

Stochastic gradient descent optimizer.

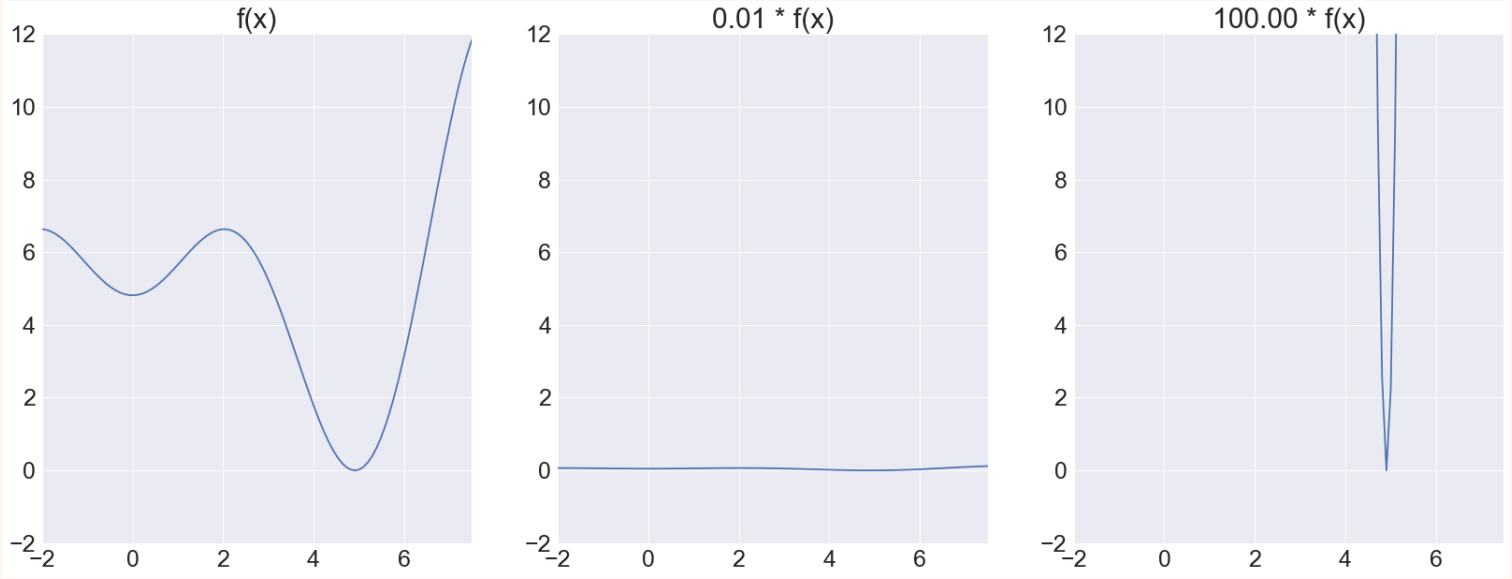
در این شیوه می‌توان از momentum، ضریب یادگیری کاهشی و nestrov momentum استفاده کرد.

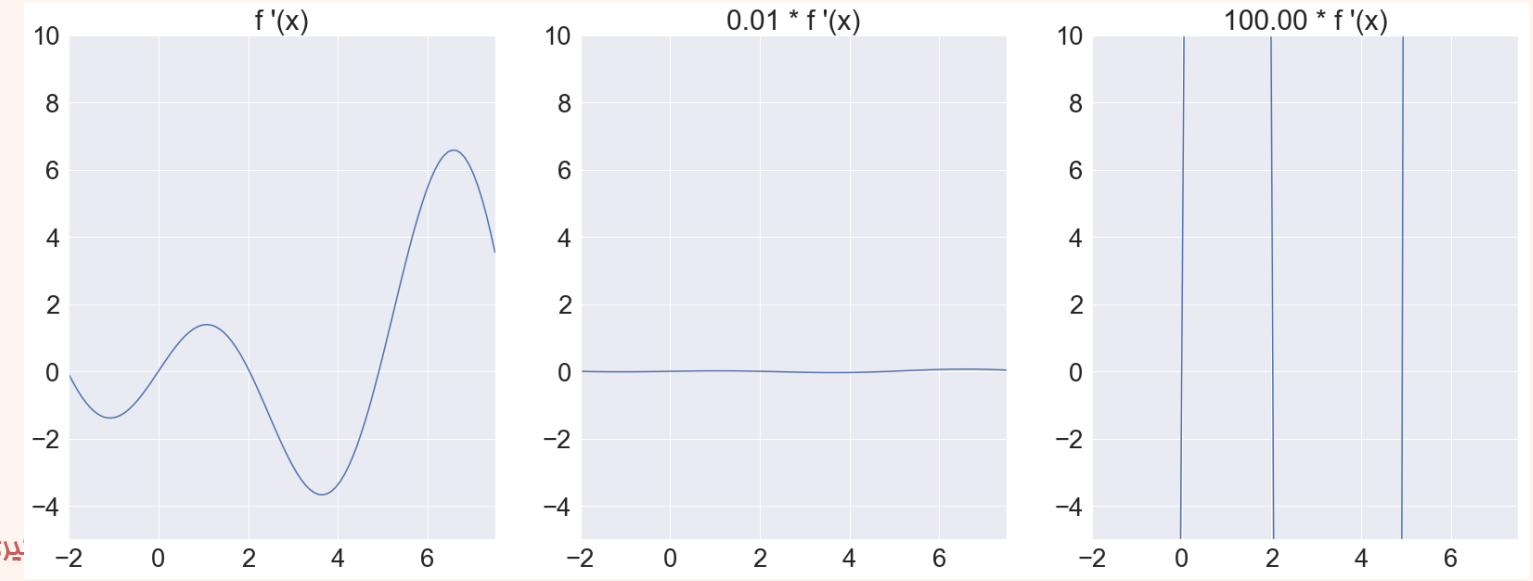
```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

ضریب یادگیری کاهشی

```
lr = self.lr * (1. / (1. + self.decay * self.iterations))
```

$$\eta_{(k)} = \frac{\eta_{(0)}}{1+\frac{k}{T}}$$

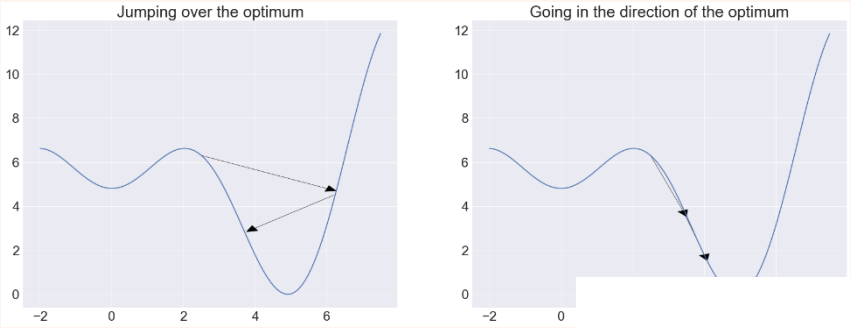




https://florian.github.io/rprop/

resilient backpropagation

اندازه گرادیان برای وزن‌های مختلف می‌تواند بسیار متفاوت باشد.
یک شیوه استفاده از علامت گرادیان به ازای هر وزن و در نظر گرفتن اندازه‌ی یکسان برای وزن‌هاست.
این شیوه در آموزش دسته‌ای به کار می‌رود.
در صورتی که این شیوه به این گونه اصلاح شود که میزان اصلاح بر اساس همسان بودن علامت گرادیان دو مرحله‌ی قبل افزایش/کاهش یابد این شیوه rprop نامیده می‌شود.



$$w_i^{(t)} = w_i^{(t-1)} - \eta_i^{(t-1)} * \mathrm{sgn}\left(\frac{\partial E^{(t-1)}}{\partial w_i^{(t-1)}}\right)$$

Riedmiller, M., and H. Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The Rprop Algorithm." Paper presented at the IEEE International Conference on Neural Networks, 28 March-1 April 1993 1993.

- مزایا:
  - در گرفتن گام با اندازه‌های متفاوت برای پارامترها
  - این روش در شرایطی که پارامترها در یک بعد به مینیم نزدیک و در بعد دیگر دور، خوب عمل می‌کند.
- معایب:
  - به دسته‌های بزرگ احتیاج دارد، چنانچه تغییرات تصادفی زیاد باشد(در تابع خطا)، همگرایی خوبی نخواهد داشت.

$$\eta_i^{(t)} = \begin{cases} \min(\eta_i^{(t-1)} * \alpha, \eta_{\max}) & \text{if } \frac{\partial E^{(t)}}{\partial w_i^{(t)}} * \frac{\partial E^{(t-1)}}{\partial w_i^{(t-1)}} > 0 \\ \max(\eta_i^{(t-1)} * \beta, \eta_{\min}) & \text{if } \frac{\partial E^{(t)}}{\partial w_i^{(t)}} * \frac{\partial E^{(t-1)}}{\partial w_i^{(t-1)}} < 0 \\ \eta_i^{(t-1)} & \text{otherwise} \end{cases}$$

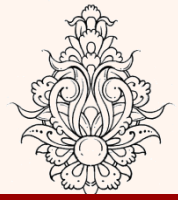
Adagrad

Adaptive gradient

keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)

In its update rule, Adagrad modifies the general learning rate η at each time step t for every parameter based on the past gradients

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \qquad S_t = S_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2$$



Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." *Journal of machine learning research* 12, no. 7 (2011).

RMSprop

```
optimizer = keras.optimizers.RMSprop(lr=0.001, rho=0.9)
```

RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates.
RMSprop as well divides the learning rate by an exponentially Moving average of squared gradients.

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$S_t = \beta S_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial w_t}\right]^2$$

Exponentially weighted averages

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

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

Exponentially Weighted Averages

Exponentially weighted averages

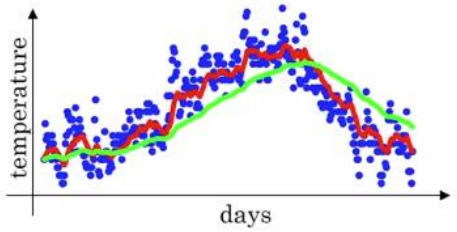
$V_t = \beta V_{t-1} + (1-\beta)\theta_t$

$\beta = 0.9$: $\approx$ 10 days' temperature

$\beta = 0.98$: $\approx$ 50 days

V_t as approximately average over $\approx \frac{1}{1-\beta}$ days' temperature.

$\frac{1}{1-0.98} = 50$



Andrew Ng

Exponentially weighted averages

$v_t = \beta v_{t-1} + (1-\beta)\theta_t$

$v_{100} = 0.9v_{99} + 0.1\theta_{100}$

$v_{99} = 0.9v_{98} + 0.1\theta_{99}$

$v_{98} = 0.9v_{97} + 0.1\theta_{98}$

...

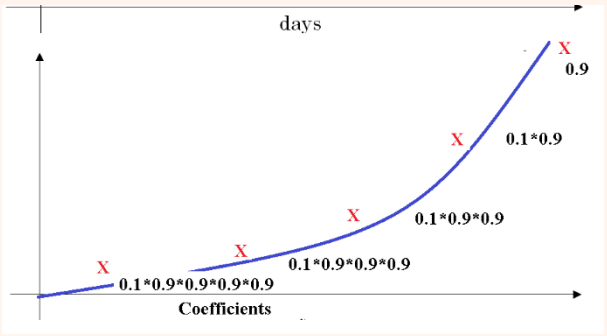
$v_{100} = 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9(0.1\theta_{98} + 0.9v_{97}))$

$= 0.1\theta_{100} + 0.1\times0.9\cdot\theta_{99} + 0.1(0.9)^2\theta_{98} + 0.1(0.9)^3\theta_{97} + \dots$

$0.9^{10} \approx 0.35 \approx \frac{1}{e}$ $\quad (1-\varepsilon)^{1/\varepsilon} = \frac{1}{e}$ $\quad 0.98^{?}$

$\rightarrow 0.98^{50} \approx \frac{1}{e}$

Andrew Ng



RMSprop

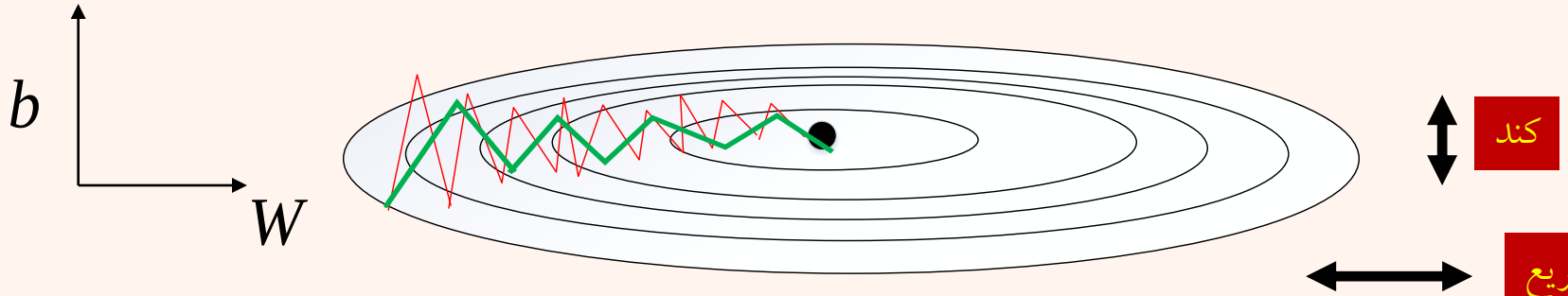
A mini-batch version of rprop

در این شیوه باید دسته‌های داده‌ی مورد استفاده متعادل باشد.

minibatch از نظر محاسباتی به صرفه‌تر است.

گرادیان به دست آمده بر مجذور میانگین مربعات تقسیم می‌شود!

استفاده از نرخ یادگیری بالاتر باعث همگرایی سریع تر میشود.



$$W_{t+1} = W_t - \eta dW$$
$$b_{t+1} = b_t - \eta db$$

ضریب یادگیری برای dw بهتر است بالاتر لحاظ شود

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$S_t = \beta S_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial w_t}\right]^2$$

Adadelta

Accumulation of the squared gradients in the denominator causes the learning rate to shrink and eventually become infinitesimally small, at which point the algorithm is no longer

3.1. Idea 1: Accumulate Over Window

Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w.

keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \epsilon}}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

where

$$D_t = \beta D_{t-1} + (1 - \beta)[\Delta w_t]^2$$

$$v_t = \beta v_{t-1} + (1 - \beta)\left[\frac{\partial L}{\partial w_t}\right]^2$$

with D and v initialised to 0, and

$$\Delta w_t = w_t - w_{t-1}$$

Zeiler, Matthew D. "Adadelta: An Adaptive Learning Rate Method." *arXiv preprint arXiv:1212.5701* (2012).

$$\text{units of } \Delta x \propto \text{units of } g \propto \frac{\partial f}{\partial x} \propto \frac{1}{\text{units of } x}$$

3.2. Idea 2: Correct Units with Hessian Approximation

$$\Delta\theta = -H^{-1}g. \qquad \Delta\theta = \frac{\frac{\partial f}{\partial \theta}}{\frac{\partial^2 f}{\partial \theta^2}}$$

$$\frac{1}{\frac{\partial^2 f}{\partial \theta^2}} = \frac{\Delta\theta}{\frac{\partial f}{\partial \theta}}$$

$$H^{-1} = \frac{\Delta\theta}{\frac{\partial f}{\partial \theta}},$$

$$\Delta\theta = -\frac{\Delta\theta}{\frac{\partial f}{\partial \theta}}g.$$

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \epsilon}}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

where

$$D_t = \beta D_{t-1} + (1 - \beta)[\Delta w_t]^2$$

$$v_t = \beta v_{t-1} + (1 - \beta)\left[\frac{\partial L}{\partial w_t}\right]^2$$

with D and v initialised to 0, and

$$\Delta w_t = w_t - w_{t-1}$$

keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)

https://medium.com/konvergen/continuing-on-adaptive-method-adadelta-and-rmsprop-1ff2c6029133

ADAM

momentum

+

RMSprop

Adaptive moment estimation

keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t} + \epsilon} \cdot \hat{V}_t$$

Bias Correction

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t}$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t}$$

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t} \right]^2$$

Kingma, Diederik P, and Jimmy Ba. "Adam: A Method for Stochastic Optimization." *arXiv preprint arXiv:1412.6980* (2014).

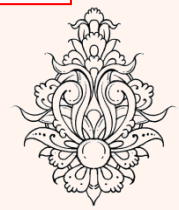
NADAM

Nadam (Dozat, 2015) is an acronym for Nesterov and Adam optimiser.

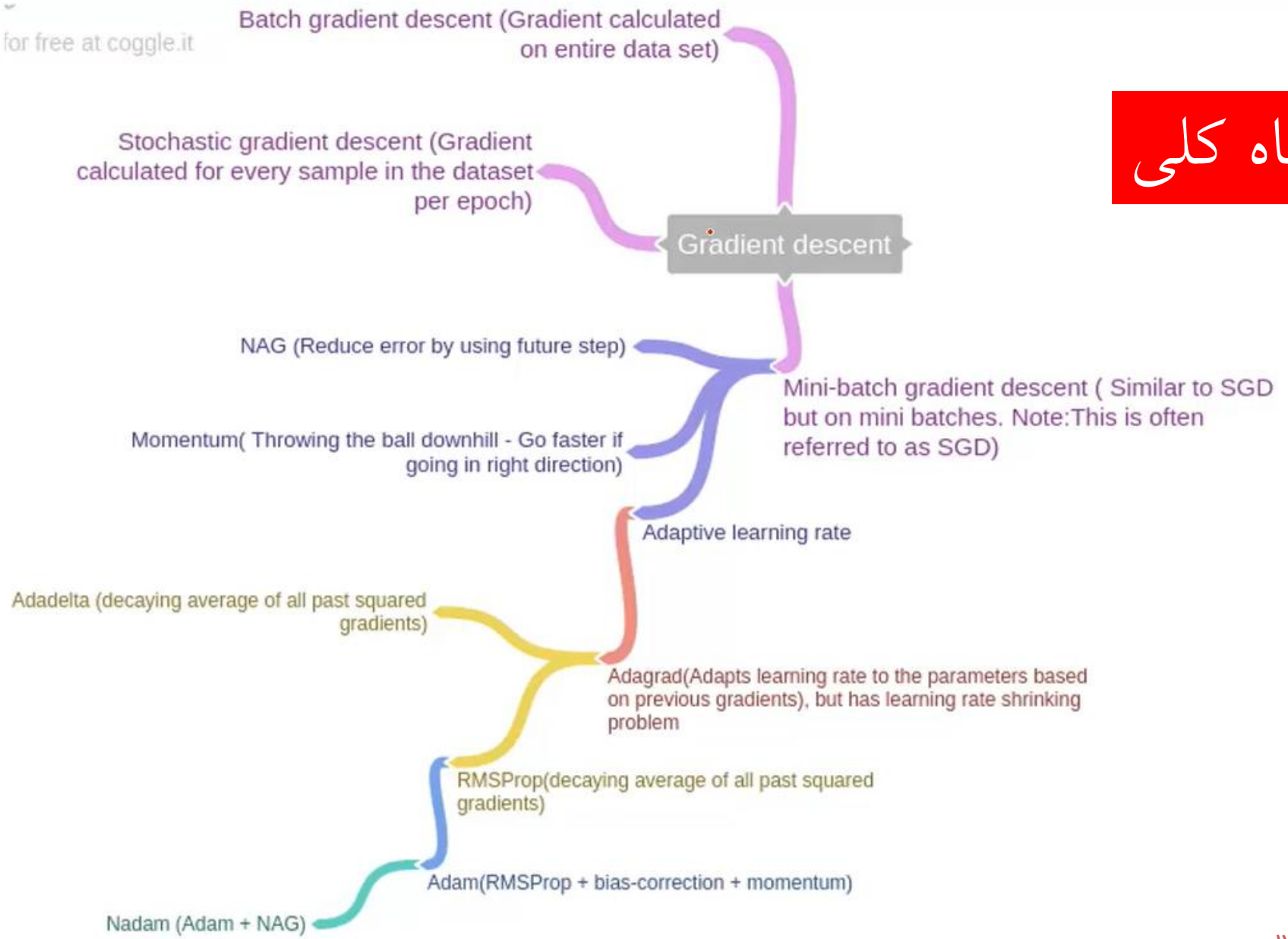
AMSGrad

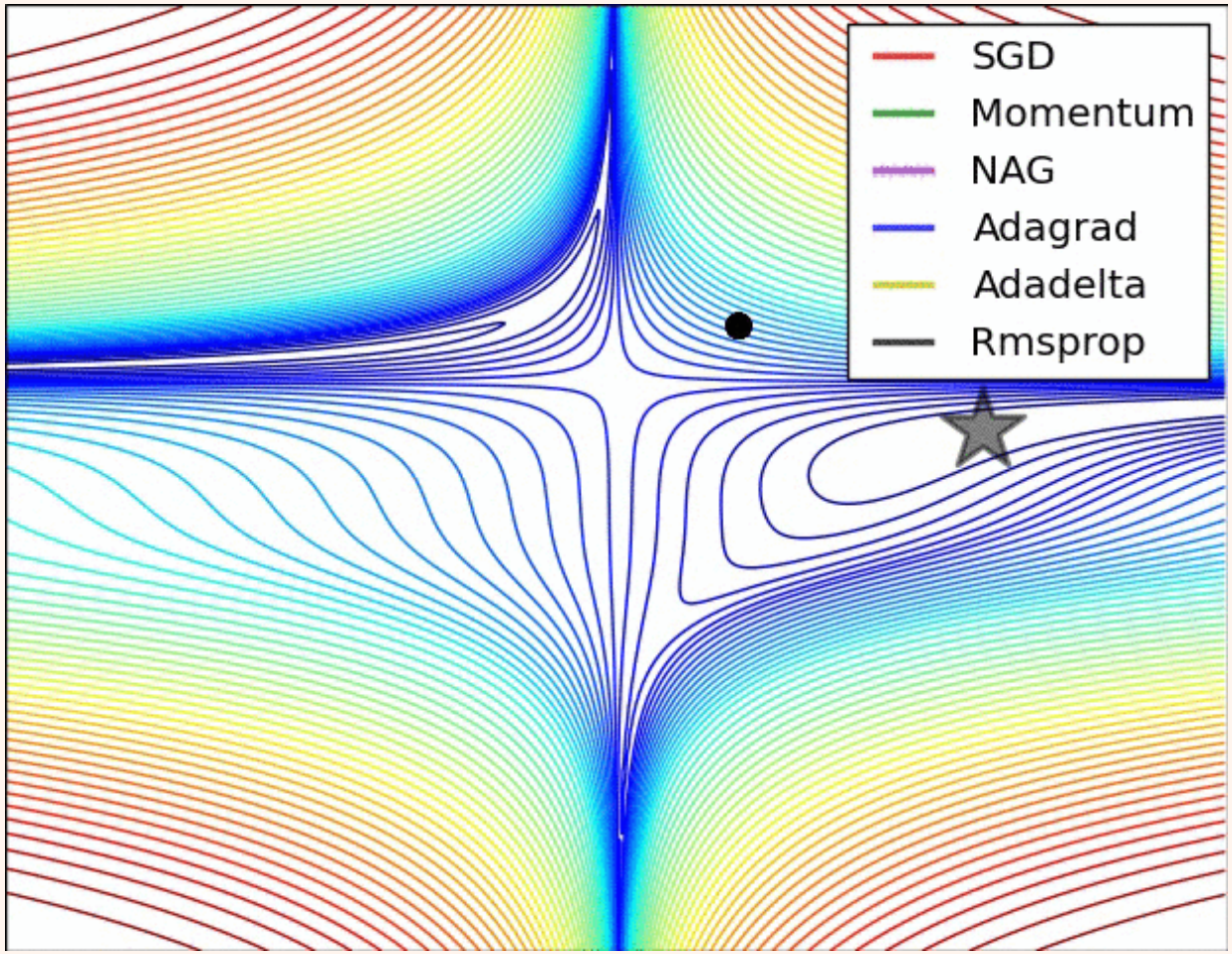
```
avg_grads = beta1 * avg_grads + (1-beta1) * w.grad
avg_squared = beta2 * (avg_squared) + (1-beta2) * (w.grad ** 2)
w = w - lr * avg_grads / sqrt(avg_squared)
```

```
avg_grads = beta1 * avg_grads + (1-beta1) * w.grad
avg_squared = beta2 * (avg_squared) + (1-beta2) * (w.grad ** 2)
max_squared = max(avg_squared, max_squared)
w = w - lr * avg_grads / sqrt(max_squared)
```



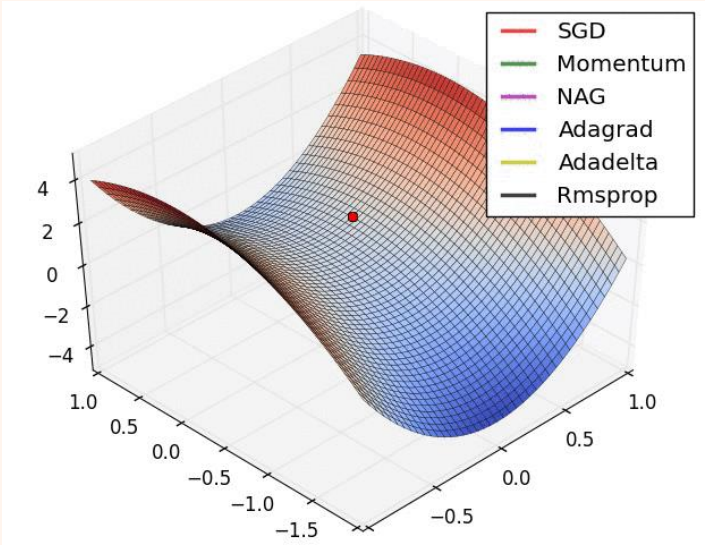
نگاه کلی

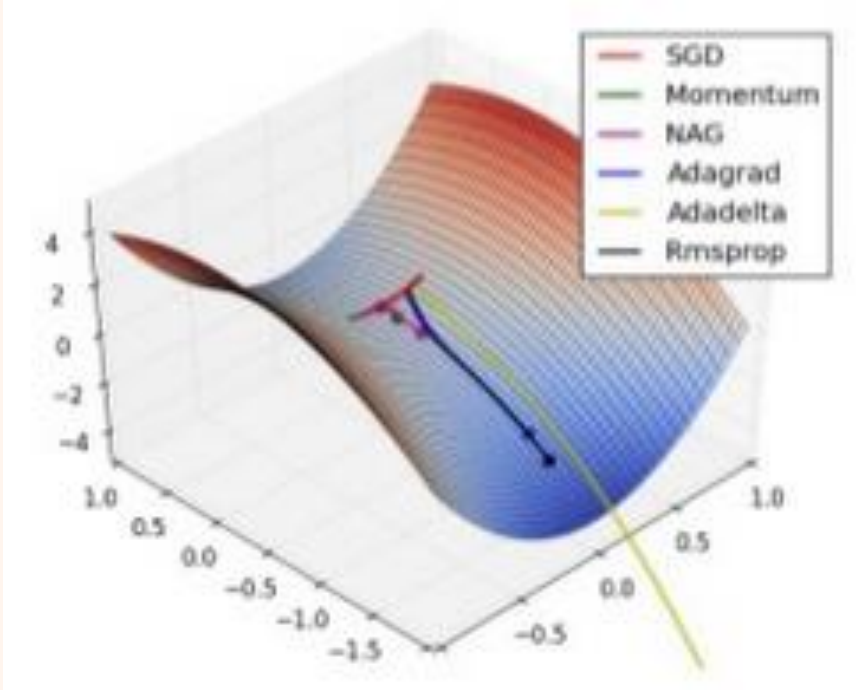

for free at coggle.it
Batch gradient descent (Gradient calculated on entire data set)
Stochastic gradient descent (Gradient calculated for every sample in the dataset per epoch)
Gradient descent
NAG (Reduce error by using future step)
Momentum(Throwing the ball downhill - Go faster if going in right direction)
Mini-batch gradient descent (Similar to SGD but on mini batches. Note:This is often referred to as SGD)
Adaptive learning rate
Adadelta (decaying average of all past squared gradients)
Adagrad(Adapts learning rate to the parameters based on previous gradients), but has learning rate shrinking problem
RMSProp(decaying average of all past squared gradients)
Adam(RMSProp + bias-correction + momentum)
Nadam (Adam + NAG)


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

مقایسه (ادامه)

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

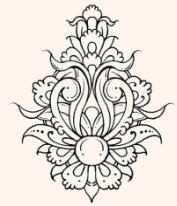




مقایسه

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

- در شرایط یکسان RMSprop, Adadelta, Adam عملکرد خوبی دارند.
- مساله‌ی bias-correction باعث می‌شود عملکرد Adam نسبت به RMSprop کمی بهتر باشد.
- در بسیاری موارد SGD با در نظر گرفتن Annealing انتخاب مناسبی است. هر چند این روش به نقطه مطلوب می‌رسد، سرعت آن پایین است.
- در مواردی که آموزش عمیق مد نظر است یا هدف بالا بردن سرعت همگرایی باشد انتخاب الگوریتمی با نرخ آموزش وفقی مناسب است.



منابع

- Ruder, S. (2016). "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747.