Numerical Optimization 06: 1st order methods

Qiang Zhu

University of Nevada Las Vegas

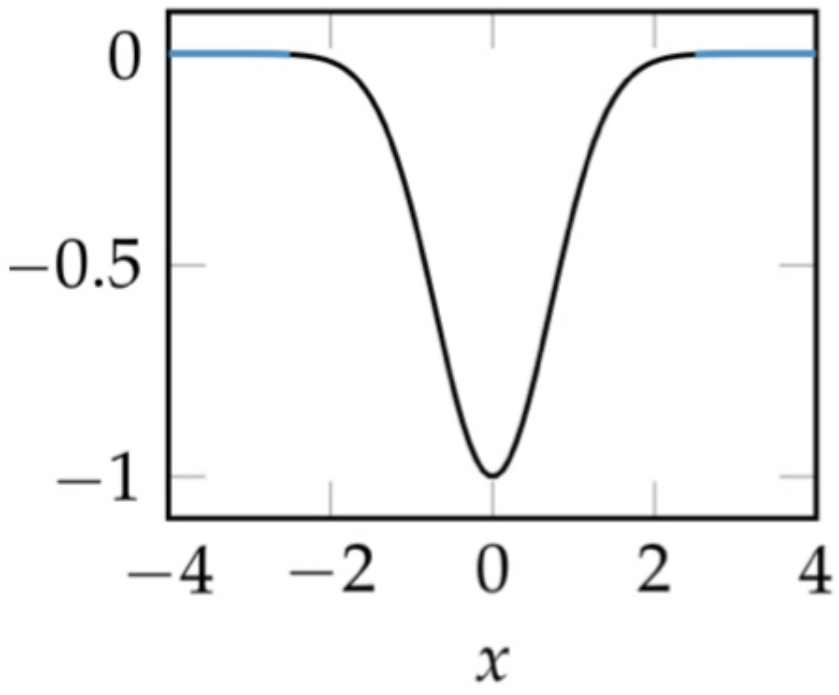
July 2, 2020

Overview

Gradient methods with fixed learning rate

We have talked about steepest descent and conjugate gradient methods, which usually work with the line search methods. Alternatively, it is popular to use the fixed learning rate method based on Gradient descent. However, the standard version will take a long time to traverse a nearly flat surface. Several methods have been proposed. They are commonly used in machine learning neural networks training.

- Momentum
- Nesterov Momentum
- Adagrad
- RMSProp
- Adadelta
- Adam



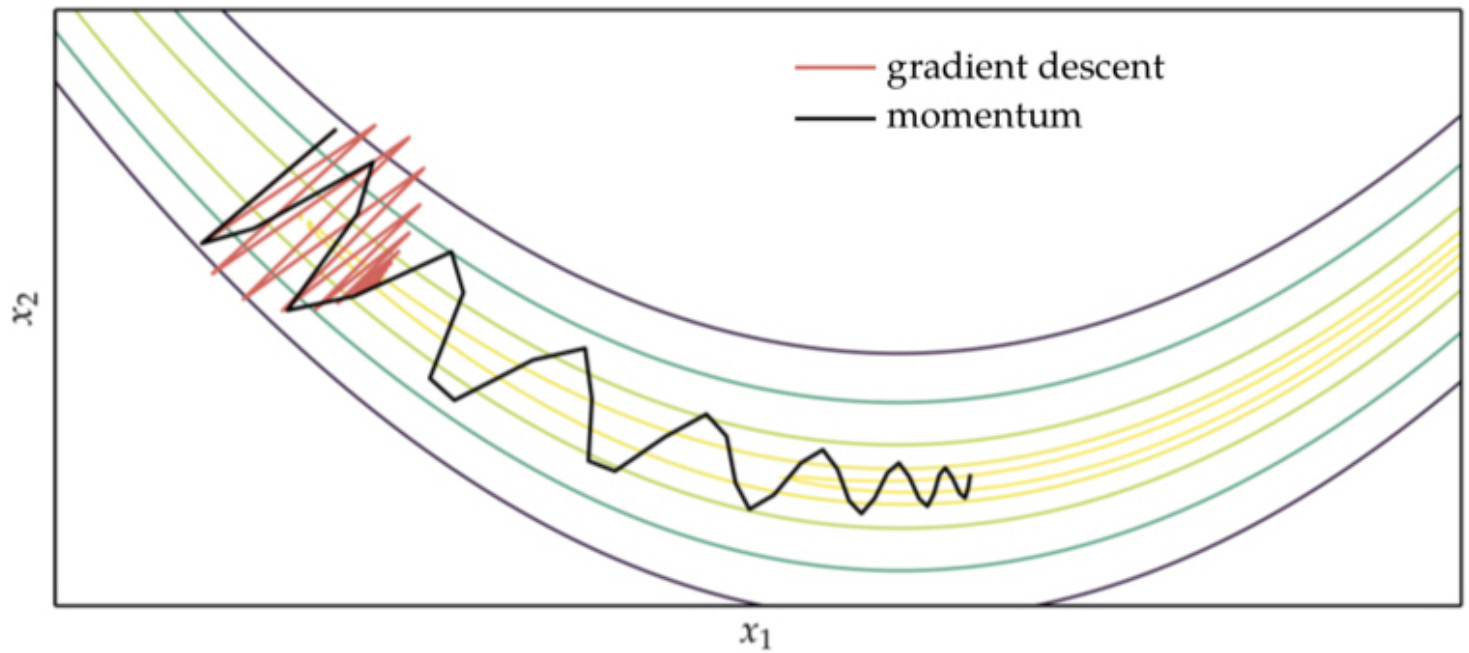
Momentum

Allowing momentum to accumulate is one way to speed the progress. Thus we can modify the gradient descent to incorporate momentum.

$$\boldsymbol{v}^{k+1} = \beta \boldsymbol{v}^k + \alpha \boldsymbol{g}^k$$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{v}^{k+1}$$

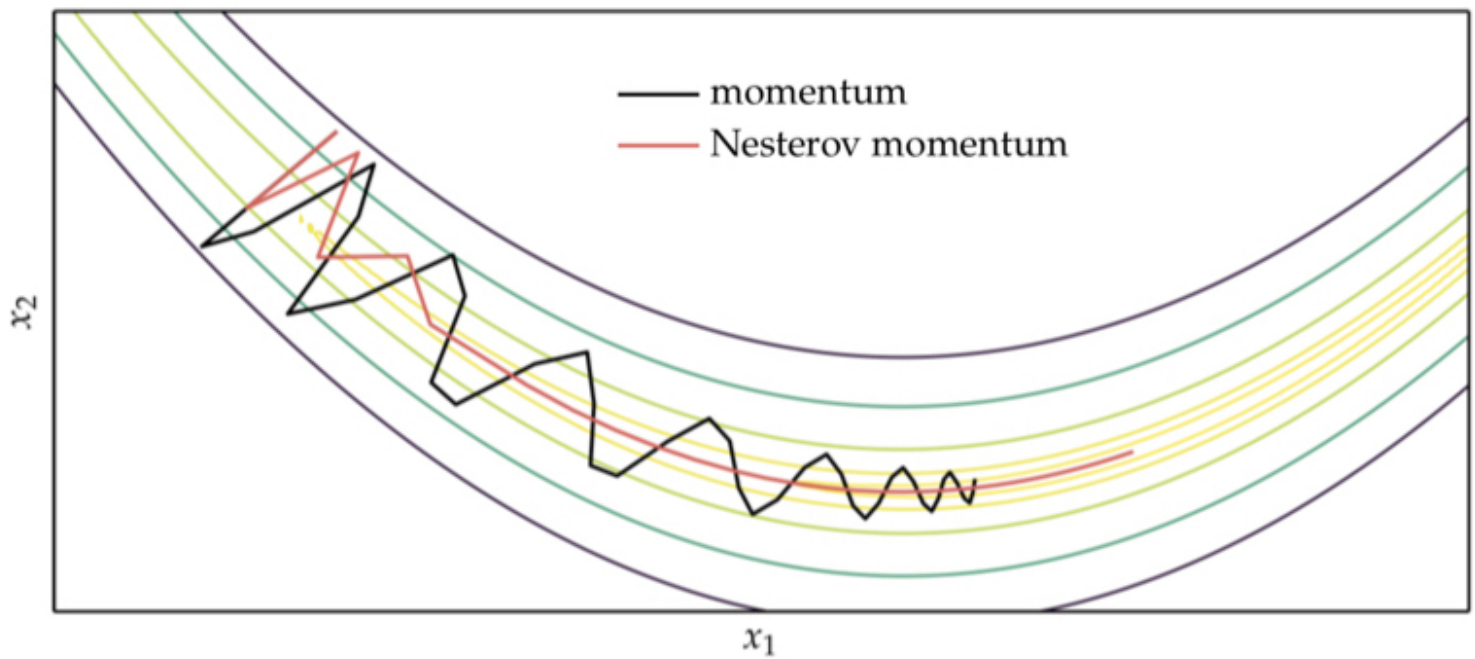
When β=0, it is gradient descent.



Nesterov Momentum

One issue of momentum is that the steps do not slow down enough at the bottom of a valley and it tends to overshoot the valley. Nesterov Momentum remedies the issue by the following updates,

$$\boldsymbol{v}^{k+1} = \beta \boldsymbol{v}^k + \alpha \nabla f(\boldsymbol{x} + \beta \boldsymbol{v}^k)$$
$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{v}^{k+1}$$



Adagrad

Momentum and Nesterov Momentum update all componens of x with the same learning rate. The adaptive subgradient method (Adagrad), adapts a learning rate for each one in x.

$$x_i^{k+1} = x_i^k - \frac{\alpha}{\epsilon + \sqrt{s_i^k}} g^k$$

$$s_i^k = \sum_{j=1}^{k} \left(g_i^j \right)^2$$

where ϵ is a small value on the order of 1e-8, to prevent the case of division by zero. Adagrad is far less sensitive to the learning rate α.

RMSProp and Adadelta

In Adagrad, the learning rate may monotonically decrease. To prevent this, RMPprop maintains a decaying average of squared gradients.

$$\boldsymbol{s}^{k+1} = \gamma \boldsymbol{s}^k + (1-\gamma)(\boldsymbol{g}^k \odot \boldsymbol{g}^k)$$

where γ is between 0 and 1, and usually is 0.9.

$$\begin{aligned} x_i^{k+1} &= x_i^k - \frac{\alpha}{\epsilon + \sqrt{s_i^k}} g_i^k \\ &= x_i^k - \frac{\alpha}{\epsilon + \mathrm{RMS}(g_i)} g_i^k \end{aligned}$$

While in Adadelta, an exponentially decaying average is used,

$$x_i^{k+1} = x_i^k - \frac{\mathrm{RMS}(\delta x_i)}{\epsilon + \mathrm{RMS}(g_i)} g_i^k$$

Adam

The adaptive moment estimation (Adam) is so far the most widely used optimization method in neural network training. It stores both an exponentially decaying squared gradient like RMSProp and Adadelta, but also an exponentially decaying gradient like momentum.

$$\boldsymbol{v}^{k+1} = \gamma_v \boldsymbol{v}^k + (1-\gamma_v)\boldsymbol{g}^k$$

$$\boldsymbol{s}^{k+1} = \gamma_s \boldsymbol{s}^k + (1-\gamma_s)\left(\boldsymbol{g}^k \odot \boldsymbol{g}^k\right)$$

$$\hat{\boldsymbol{v}}^{k+1} = \boldsymbol{v}^{k+1}/(1-\gamma_v^k)$$

$$\hat{\boldsymbol{s}}^{k+1} = \boldsymbol{s}^{\boldsymbol{k+1}}/(1-\gamma_s^k)$$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha\hat{\boldsymbol{v}}^{k+1}/\left(\epsilon + \sqrt{\hat{\boldsymbol{s}}^{k+1}}\right)$$

Summary

- Descent methods with momentum build up progress in favorable directions
- A wide variety of accelerated descent methods use special techniques to speed up descent