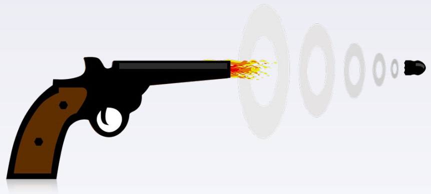
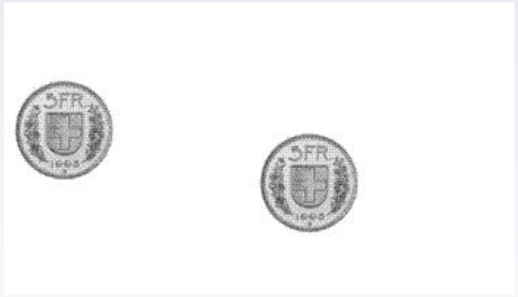
1

Momentum in Machine Learning

Outline

- **What** is Momentum
- **Why** Momentum in Machine Learning.
- **How** to implement momentum in Machine Learning.
- Other optimization algorithms.
- Implementation.
- Conclusion.





source: https://en.wikipedia.org/wiki/Momentum

Here’s a popular story about momentum

Gradient descent is a man walking down a hill. He follows the steepest path downwards; his progress is slow, but steady.

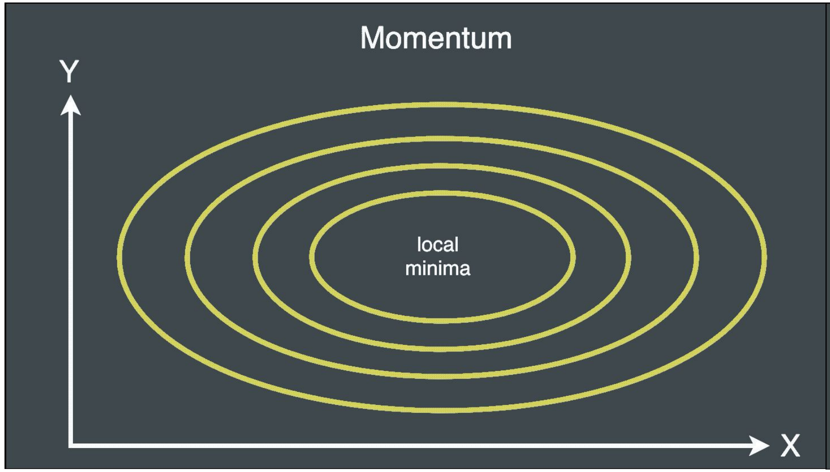
Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima.

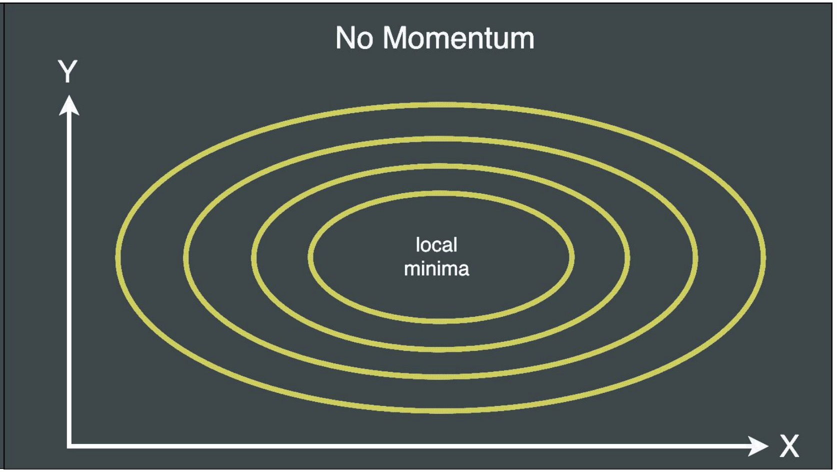
WHY

Gradient Descent is good but slow

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.

Simply put, momentum algorithm helps us progress faster during training, negatively or positively.





HOW

Classical Momentum with Gradient Descent

$$\Delta\theta_t = \eta\nabla J(\theta_t) + \gamma\Delta\theta_{t-1}$$

$$\theta_t = \theta_t - \Delta\theta_t$$

Classical Momentum with Gradient Descent

This is momentum term

$$\Delta\theta_t = \eta\nabla J(\theta_t) + \gamma\Delta\theta_{t-1}$$

$$\theta_t = \theta_t - \Delta\theta_t$$

The momentum term gamma(y) is usually set to 0.9 or similar value.

Nesterov accelerated gradient (NAG)

Classic momentum is like a ball that is rolling down a hill, blindly following the slope. This is not satisfactory for us, we'd like to have a smarter ball, a ball that has a notion of where it is going so that it knows when to slow down before the hill slopes up again.

Nesterov accelerated gradient (NAG) is a way to give our momentum term this kind of superpower.

Nesterov accelerated gradient (NAG)

Can we look ahead in future

$$\Delta\theta_t = \eta\nabla J(\theta_t - \gamma\Delta\theta_{t-1}) + \gamma\Delta\theta_{t-1}$$

$$\theta_t = \theta_t - \Delta\theta_t$$

The momentum term gamma(y) is usually set to 0.9 or similar value.

Adaptive learning rate

With momentum, we are able to adapt our updates to the slope of our error function and speed up SGD in turn, we would also like to adapt our updates to each individual parameter to perform larger or smaller updates depending on their importance.

Other optimization Algorithms

Root Mean Squared Propagation (RMSprop)

RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton in Lecture 6e of his Coursera Class.

$$E[g^2]_t = (1-\gamma)g^2 + \gamma E[g^2]_t$$

Where $$g = \nabla J(\theta_t)$$

$$\theta_t = \theta_t - \frac{\eta}{\sqrt{\epsilon + E[g^2]_t}} \nabla J(\theta_t)$$

Hinton suggests γ(gamma) to be set to 0.9

Adaptive Moment Estimation (Adam)

Adam is another adaptive learning rate that is basically combining the idea of classical momentum with RMSprop.

$$m_t = \gamma m_{t-1} + (1-\gamma) g_t$$

$$v_t = \beta v_{t-1} + (1-\beta) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\gamma_t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_t}$$

Where

$$g_t = \nabla_\theta J(\theta_t)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Implementation

References

- https://ruder.io/optimizing-gradient-descent/index.html#momentum
- https://distill.pub/2017/momentum/
- https://medium.com/@abhinav.mahapatra10/ml-advanced-momentum-in-machine-learning-what-is-nesterov-momentum-ad37ce1935fc
- https://jlmelville.github.io/mize/nesterov.html
- https://mlfromscratch.com/optimizers-explained/#/
- https://www.coursera.org/learn/deep-neural-network/