Module 2

Momentum Based GD

Gradient descent optimization algorithms

In the following, we will outline some algorithms that are widely used by the deep learning community to deal with the aforementioned challenges. We will not discuss algorithms that are infeasible to compute in practice for high-dimensional data sets, e.g. second-order methods such as Newton's method.

Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another [4], 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 as in Image 2.

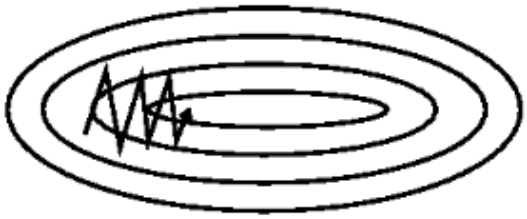

Image 2: SGD without momentum

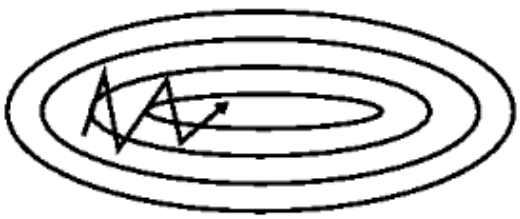

Image 3: SGD with momentum

Momentum [5] is a method that helps accelerate SGD in the relevant direction and dampens oscillations as can be seen in Image 3. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

Module 2

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Note: Some implementations exchange the signs in the equations. The momentum term γ is usually set to 0.9 or a similar value.

Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance, i.e. $\gamma < 1$). The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.