Optimizers

Saeed Reza Kheradpisheh

s_kheradpisheh@sbu.ac.ir

Department of Computer Science
Shahid Beheshti University
Summmer 1398

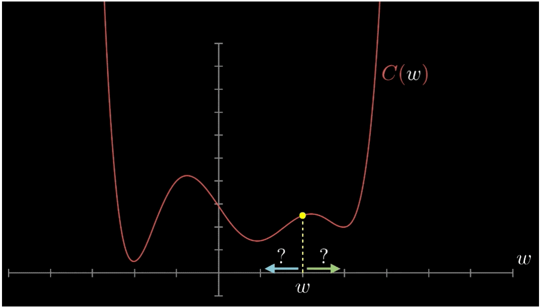
SGD

$$\theta = \theta - \eta \cdot \overbrace{\nabla_\theta J(\theta; x, y)}^{\text{Backpropagation}}$$

- θ is a parameter (theta), e.g. your weights, biases and activations.
- η is the learning rate (eta).
- Δ is the gradient (nabla), which is taken of J
- J is formally known as objective function, but most often it's called cost function or loss function.

SGD

Gif 01



SGD

$$\theta = \theta - \eta \cdot \nabla J(\theta; x, y) \Leftrightarrow \theta = \theta - \eta \cdot \frac{\partial C}{\partial \theta}$$

Pros:

- Relatively fast compared to the older gradient descent approaches
- SGD is comparatively easy to learn for beginners, since it is not as math heavy as the newer approaches to optimizers

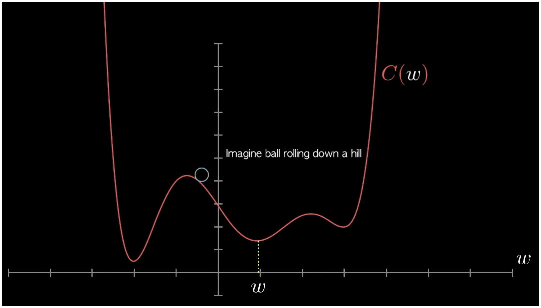
Cons:

- Converges slower than newer algorithms
- Has more problems with being stuck in a local minimum than newer approaches
- Newer approaches outperform SGD in terms of optimizing the cost function

Momentum

The momentum algorithm helps us progress faster in the neural network, negatively or positively, to the ball analogy.

Gif 02

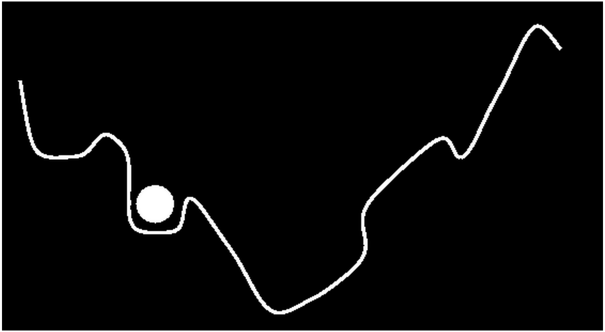


Momentum

Learning rule:

$$\theta_t = \theta_t - \eta \nabla J(\theta_t) + \gamma v_t$$

$$v_t = \eta \nabla J(\theta_{t-1}) + v_{t-1}$$

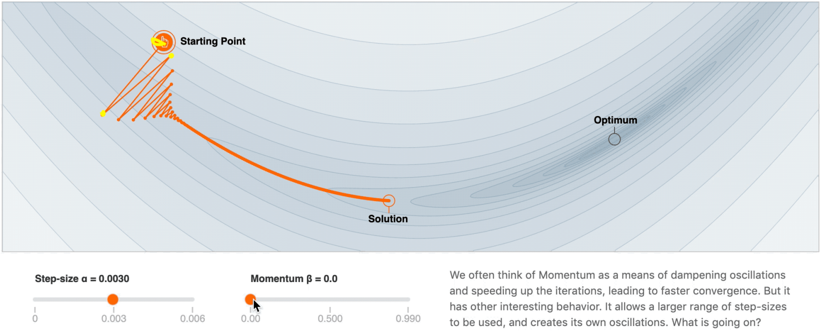


Momentum

Learning rule:

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

Gif 03



Adaptive Learning Rate

An adaptive learning rate can be observed in AdaGrad, AdaDelta, RMSprop and Adam.

AdaGrad: Parameters Gets Different Learning Rates

Learning rule:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\varepsilon + \sum_{\tau=1}^{t} \left(\nabla J(\theta_{\tau,i})\right)^2}} \nabla J(\theta_{t,i})$$

All we added here is division of the learning rate η by the second momentum.

e.g. if $t = 3$:

$$\theta_{4,i} = \theta_{3,i} - \frac{\eta}{\sqrt{\varepsilon + g(\theta_{1,i})^2 + g(\theta_{2,i})^2 + g(\theta_{3,i})^2}} \nabla J(\theta_{3,i})$$

RMSprop

Root Mean Squared Propagation (RMSprop) is very close to Adagrad, except for it does not provide the sum of the gradients, but instead an exponentially decaying average.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\varepsilon + E[g^2]_t}} \nabla J(\theta_{t,i})$$

we have:

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

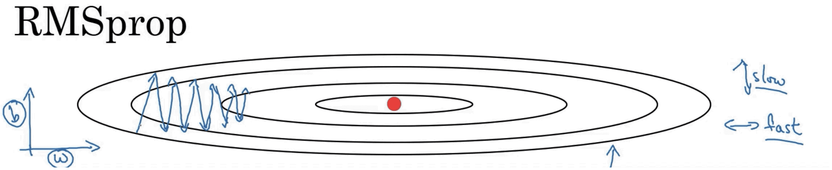
where

$$g = \nabla J(\theta_{t,i})$$

RMSprop

With the AdaGrad algorithm, the learning rate η was monotonously decreasing, while in RMSprop, η can adapt up and down in value, as we step further down the hill for each epoch.

Gif 04



Adam

Root Mean Squared Propagation (RMSprop) is very close to Adagrad, except for it does not provide the sum of the gradients, but instead an exponentially decaying average.

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

where

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

and where

$$m_t = (1 - \beta_1) g_t + \beta_1 m_{t-1}$$

$$v_t = (1 - \beta_2) g_t^2 + \beta_2 v_{t-1}$$

Adam

- Epsilon ε, which is just a small term preventing division by zero. This term is usually 10^{-8}.
- η is the learning rate. A good default setting is $\eta = 0.001$.
- The gradient g, which is still the same thing as before: $g = \nabla J(\theta_{t,i})$
- First momentum term is usually set as $\beta_1 = 0.9$
- Second momentum term is usually set as $\beta_2 = 0.999$

Adam (bias correction)

We want that:

$$E[m_t]=E[g_t] \quad E[v_t]=E[g_t^2]$$

For $t=3$:

$$m_0=0$$
$$m_1=\beta_1 m_0+(1-\beta_1)g_1=(1-\beta_1)g_1$$
$$m_2=\beta_1 m_1+(1-\beta_1)g_2=\beta_1(1-\beta_1)g_1+(1-\beta_1)g_2$$
$$m_3=\beta_1 m_2+(1-\beta_1)g_3=\beta_1^2(1-\beta_1)g_1+\beta_1(1-\beta_1)g_2+(1-\beta_1)g_3$$

thus

$$m_t=(1-\beta_1)\sum_{i=0}^{t}\beta_1^{t-i}g_i$$

and therefore:

$$\hat{m}_t=\frac{m_t}{1-\beta_1^t} \quad \hat{v}_t=\frac{v_t}{1-\beta_2^t}$$