Optimization Algorithms

Dai Bui

Gradient Descent Optimizations

- Gradient descent update

$$W = W - \alpha \frac{\partial L}{\partial W}$$

- Now, consider a function

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$
$$L(W) = 4w_0^2 + w_1^2$$

let us calculate the gradient

$$\frac{\partial L}{\partial w_0} = 8w_0$$

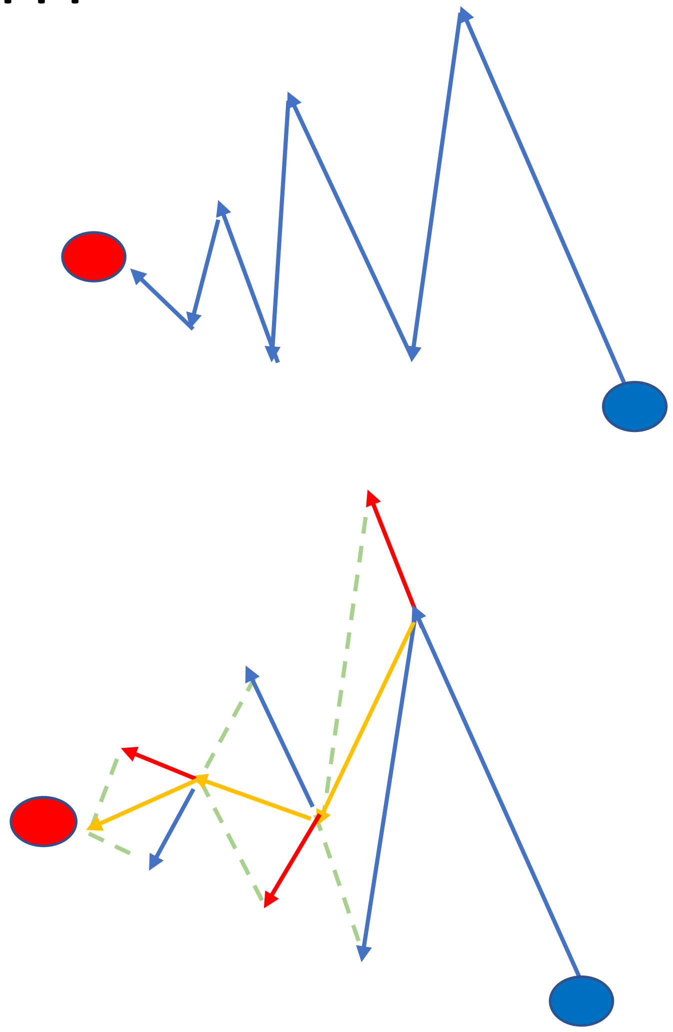
$$\frac{\partial L}{\partial w_1} = w_1$$

SDG with Momentum

- Can mitigate the zig-zag effect?

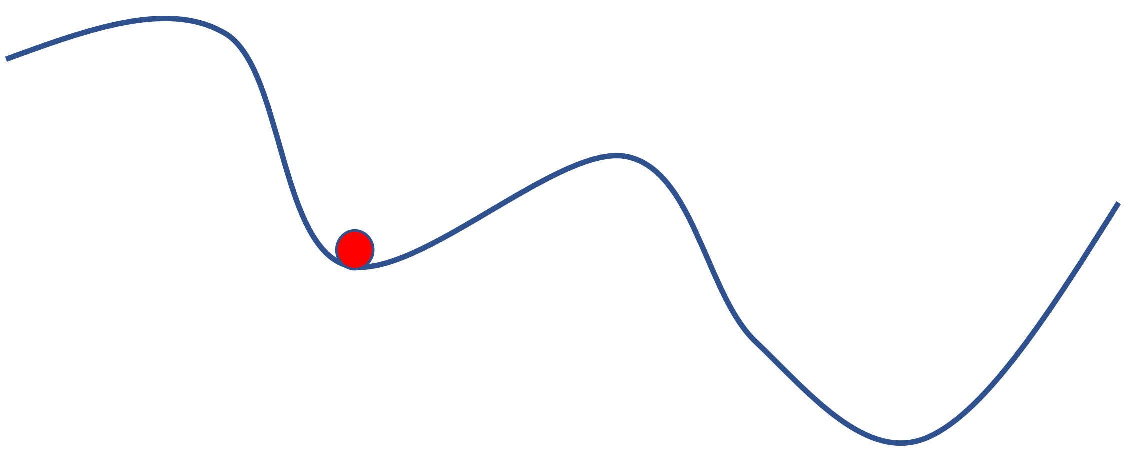
$$v_t = \rho v_{t-1} + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

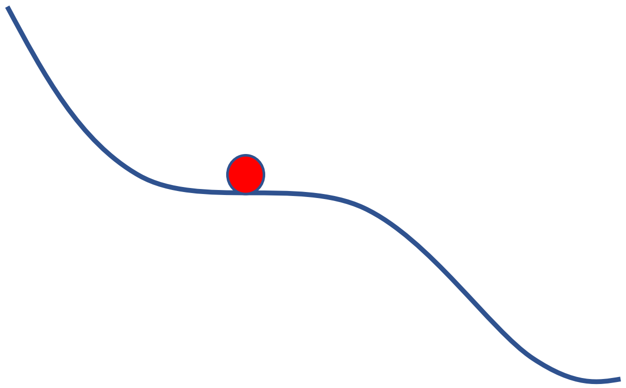
- Velocity is the running mean of the gradients



SGD with Momemtum

- Will it help with loss functions with local minim or saddle points





AdaGrad

$$sum\_squared \mathrel{+}= \left(\nabla f(x)\right)^2$$

$$x_{t+1} = x_t - \frac{\alpha \nabla f(x)}{\sqrt{sum\_squared} + 10^{-7}}$$

- What happens with the step size over time?

RMSProb

$$sum\_squared = \beta \ * \ sum\_squared + (1-\beta)\left(\nabla f(x)\right)^2$$

$$x = x - \frac{\alpha \nabla f(x)}{\sqrt{sum\_squared} + 10^{-7}}$$

- β is the learning rate

Adam (naïve)

$$first\_moment = \beta_1 * first\_moment + (1 - \beta_1)\nabla f(x)$$

Momentum

$$second\_moment = \beta_2 * second\ moment + (1 - \beta_2)\left(\nabla f(x)\right)^2$$

$$x = x - \frac{\alpha * first\_moment}{\sqrt{second\_moment} + 10^{-7}}$$

AdaGrad/RMSProp

- What happens at the first step?

Adam (adjusted)

$$first\_moment = \beta_1 * first\_moment + (1 - \beta_1)\nabla f(x)$$

Momentum

$$second\_moment = \beta_2 * second\ moment + (1 - \beta_2)\left(\nabla f(x)\right)^2$$

AdaGrad/RMSProp

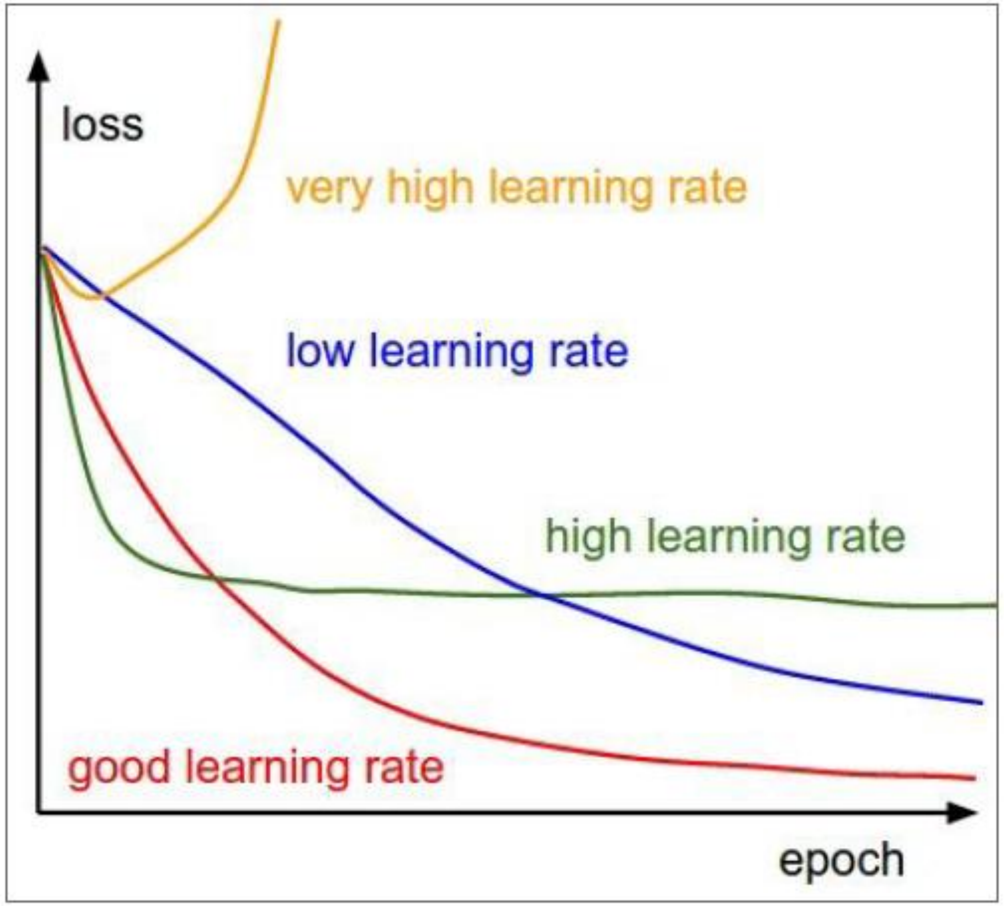
$$first\_unbias = \frac{first\_moment}{(1 - \beta_1^{iteration})}$$

$$second\_unbias = \frac{second\_moment}{(1 - \beta_2^{iteration})}$$

$$x = x - \frac{\alpha * first\_unbias}{\sqrt{second\_unbias} + 10^{-7}}$$

- Bias correction for first and second moment that start at zero

Learning Rate Decay



Which is the best learning rates?

Learning Rate Decay

- Learning rate decay over time, e.g., reduce learning rate by half every few epochs
    - When we are closer to the target, we do not want to "move" too fast
- Exponential decay: $\alpha = \alpha_0 e^{-kt}$
- $\frac{1}{t}$ decay: $\frac{\alpha_0}{(1+kt)}$
- Does AdaGrad, RMSProp and Adam needs this?