5 - Deep Learning Overview

5.5 - Optimization for Model Training

Wendley S. Silva

Setembro, 2020

Index

HUAWEI

Optimizer

- There are various optimized versions of gradient descent algorithms. In object-oriented language implementation, different gradient descent algorithms are often encapsulated into objects called **optimizers**.
- **Purposes** of the algorithm optimization include but are not limited to:
  - Accelerating algorithm convergence.
  - Preventing or jumping out of local extreme values.
  - Simplifying manual parameter setting, especially the learning rate (LR).
- Common optimizers: common GD optimizer, **momentum optimizer**, Nesterov, **AdaGrad**, AdaDelta, **RMSProp**, **Adam**, AdaMax, and Nadam.

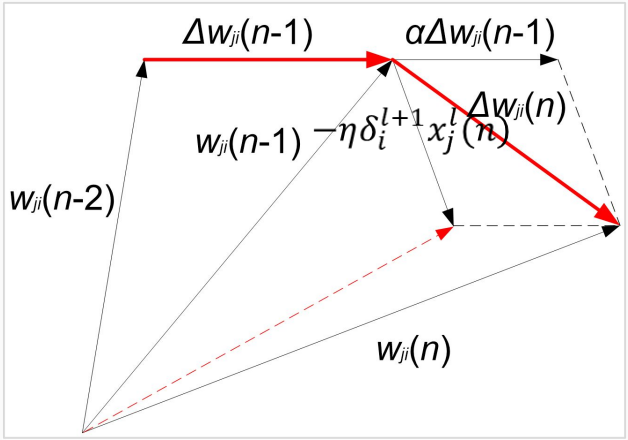
Momentum Optimizer

- A most basic improvement is to add momentum terms for Δw_{ji}. Assume that the weight correction of the n-th iteration is $\Delta w_{ji}(n)$. The weight correction rule is:

$$\Delta w_{ji}^{l}(n) = -\eta \delta_i^{l+1} x_j^{l}(n) + \alpha \Delta w_{ji}^{l}(n-1)$$

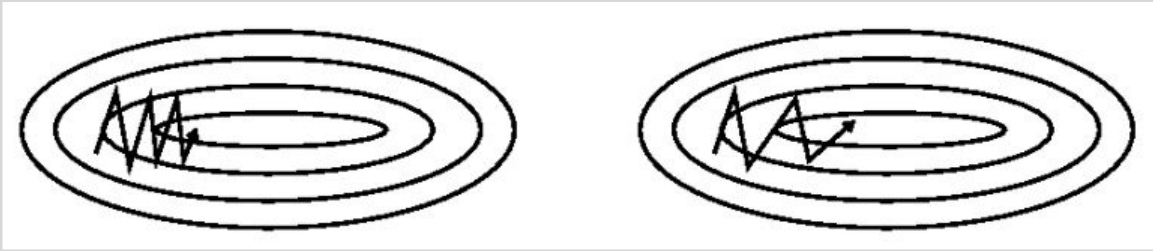
  where α is a constant $(0 \le \alpha < 1)$ called Momentum Coefficient and $\alpha \Delta w_{ji}(n-1)$ is a momentum term.

- Imagine a small ball rolls down from a random point on the error surface. The introduction of the momentum term is equivalent to giving the small ball inertia.



Advantages and Disadvantages of Momentum Optimizer

- Advantages:
  - Enhances the stability of the gradient correction direction and reduces mutations.
  - In areas where the gradient direction is stable, the ball rolls faster and faster (there is a speed upper limit because $\alpha < 1$), which helps the ball quickly overshoot the flat area and accelerates convergence.
  - A small ball with inertia is more likely to roll over some narrow local extrema.
- Disadvantages:
  - The learning rate η and momentum α need to be manually set, which often requires more experiments to determine the appropriate value.



AdaGrad Optimizer (1)

- The common feature of the random gradient descent algorithm (SGD), small-batch gradient descent algorithm (MBGD), and momentum optimizer is that each parameter is updated with the **same LR**.
- According to the approach of AdaGrad, different learning rates need to be set for different parameters.

$$g_t = \frac{\partial C(t,o)}{\partial w_t}$$

Gradient calculation

$$r_t = r_{t-1} + g_t^2$$

Square gradient accumulation

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t$$

Computing update

$$w_{t+1} = w_t + \Delta w_t$$

Application update

- g_t indicates the t-th gradient, r is a gradient accumulation variable, and the initial value of r is 0, which **increases continuously**. η indicates the global LR, which needs to be set manually. ε is a small constant, and is set to about 10^{-7} for numerical stability.

AdaGrad Optimizer (2)

- The AdaGrad optimization algorithm shows that the r continues increasing while the overall learning rate keeps decreasing as the algorithm iterates. This is because we hope **LR to decrease** as the number of updates increases. In the initial learning phase, we are far away from the optimal solution to the loss function. As the number of updates increases, we are closer to the optimal solution, and therefore LR can decrease.
- Pros:
  - The learning rate is automatically updated. As the number of updates increases, the learning rate decreases.
- Cons:
  - The denominator keeps accumulating so that the learning rate will eventually become very

RMSProp Optimizer

- The RMSProp optimizer is an improved AdaGrad optimizer. It introduces an attenuation coefficient to ensure **a certain attenuation ratio** for r in each round.
- The RMSProp optimizer solves the problem that the AdaGrad optimizer ends the optimization process too early. It is suitable for non-stable target handling and has good effects on the RNN.

$$g_t = \frac{\partial C(t,o)}{\partial w_t}$$ Gradient calculation

$$r_t = \beta r_{t-1} + (1-\beta) g_t^2$$ Square gradient accumulation

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t$$ Computing update

$$w_{t+1} = w_t + \Delta w_t$$ Application update

- g_t indicates the t-th gradient, r is a gradient accumulation variable, and the initial value of r is 0, which **may not increase and needs to be adjusted using a parameter**. β is the attenuation factor, η indicates the global LR, which needs to be set manually. ε is a small constant, and is set to about 10^{-7} for numerical stability.

Adam Optimizer

- Adaptive Moment Estimation (Adam): Developed based on AdaGrad and AdaDelta, Adam maintains two additional variables m_t and v_t for each variable to be trained:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- Where t represents the t-th iteration and g_t is the calculated gradient. m_t and v_t are moving averages of the gradient and square gradient. From the statistical perspective, m_t and v_t are estimates of the first moment (the average value) and the second moment (the uncentered variance) of the gradients respectively, which also explains why the method is so named.

Adam Optimizer

- If m_t and v_t are initialized using the zero vector, m_t and v_t are close to 0 during the initial iterations, especially when β_1 and β_2 are close to 1. To solve this problem, we use $\hat{m}_t$ and $\hat{v}_t$:

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

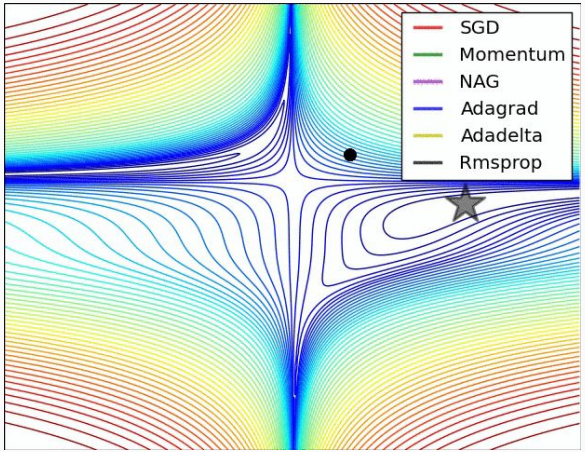
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

- The weight update rule of Adam is as follows:

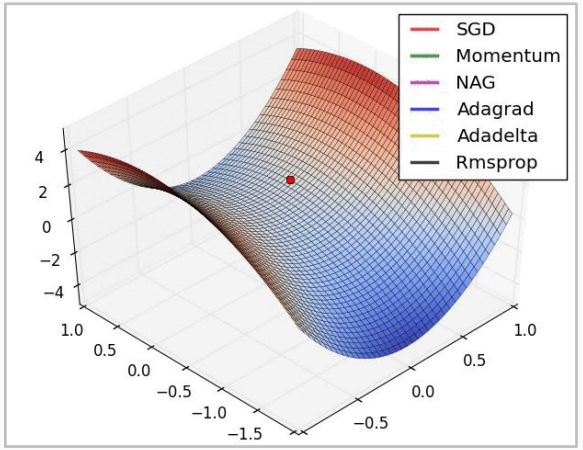
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

- Although the rule involves manual setting of η, β_1, and β_2, the setting is much simpler. According to experiments, the default settings are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and $\eta = 0.001$. In practice, Adam will converge quickly. When convergence saturation is reached, xx can be reduced. After several times of reduction, a satisfying local extremum will be obtained. Other parameters do not need to be adjusted.

Optimizer Performance Comparison



Comparison of optimization algorithms in contour maps of loss functions



Comparison of optimization algorithms at the saddle point

5. Visão geral de Aprendizagem Profunda

Next: 5.6 - Types of Neural Networks

Wendley Silva

Setembro, 2020