TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2020

Stochastic Gradient Descent (SGD)

Momentum as a Running Average

Decoupled Momentum

Momentum

The standard (PyTorch) momentum SGD equations are

$$v_t = \mu v_{t-1} + \eta * \hat{g}_t \quad \mu \text{ is typically .9 or .99}$$

$$\Phi_{t+1} = \Phi_t - v_t$$

We wil refer to η in these equations as the **PyTorch Learning Rate Parameter**.

Momentum

The theory of momentum is generally given in terms of second order structure and total gradients (GD rather than SGD).

But second order analyses are of questionable value for SDG in large dimension.

Still, momentum is widely used in practice.

To better understand momentum we will rewrite it as a running average.

Running Averages

Consider a sequence $x_1, x_2, x_3, \ldots$.

For $t \geq N$, consider the average of the N most recent values.

$$\overline{x}_t = \frac{1}{N} \sum_{\tilde{t}=t-N+1}^{t} x_{\tilde{t}}$$

This can be approximated efficiently with

$$\tilde{x}_0 = 0$$

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right) \tilde{x}_{t-1} + \left(\frac{1}{N}\right) x_t$$

Deep Learning Convention for Running Averages

In deep learning a running average

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right) \tilde{x}_{t-1} + \left(\frac{1}{N}\right) x_t$$

is written as

$$\tilde{x}_t = \beta \tilde{x}_{t-1} + (1 - \beta) x_t$$

where

$$\beta = 1 - 1/N$$

Typical values for β are .9, .99 or .999 corresponding to N being 10, 100 or 1000.

It will be convenient here to use N rather than β.

Momentum as a Running Average

$$
\begin{aligned}
v_t &= \mu v_{t-1} + \eta \hat{g}_t \\
&= \left(1 - \frac{1}{N_g}\right) v_{t-1} + \frac{1}{N_g}(N_g \eta \hat{g}_t)
\end{aligned}
$$

We see that v_t is a running average of $N_g \eta \hat{g}$.

Momentum as a Running Average

v_t is a running average of $N_g \eta \hat{g}$.

Alternatively, we can consider a direct running average of the gradient.

$$\tilde{g}_t = \left(1 - \frac{1}{N_g}\right) \tilde{g}_{t-1} + \frac{1}{N_g} \hat{g}_t$$

Since averaging is a linear operation, we get

$$v_t = N_g \eta \tilde{g}_t$$

Momentum as a Running Average

We have now shown that standard momentum (PyTorch Momentum) can be written as

$$\tilde{g}_t = \left(1 - \frac{1}{N_g}\right) \tilde{g}_{t-1} + \frac{1}{N_g} \hat{g}_t$$

$$\Phi_{t+1} = \Phi_t - \tilde{\eta} \tilde{g}_t$$

$$\tilde{\eta} = N_g \eta$$

Here η is the standard (PyTorch) learning rate parameter for momentum.

Decoupled Momentum

$$\tilde{g}_t = \left(1 - \frac{1}{N_g}\right) \tilde{g}_{t-1} + \frac{1}{N_g} \hat{g}_t$$

$$\Phi_{t+1} = \Phi_t - \tilde{\eta} \tilde{g}_t$$

$$\tilde{\eta} = N_g \eta$$

We will show that $\tilde{\eta}$ is a more rational (decoupled from N_g) learning rate parameter than the PyTorch learning rate η when using momentum.

Decoupled Momentum

We will use the general **matching principle** that the effect of a single batch element gradient $\hat{g}_{t,b}$ on the model should be $\eta_0 \hat{g}_{t,b}$ where η_0 is the optimal learning rate for Vanilla SGD with batch size 1.

This principle should lead to converged loss values with mini-batching and momentum similar to convergence loss values for SGD with batch size 1, learning rate η_0, and no momentum.

Decoupled Momentum

$$\tilde{g}_t = \left(1 - \frac{1}{N_g}\right) \tilde{g}_{t-1} + \frac{1}{N_g} \hat{g}_t$$

$$\Phi_{t+1} = \Phi_t - \tilde{\eta} \tilde{g}_t$$

For $\Delta t >> N_g$, the effect of $\hat{g}_t$ on $\Phi_{t+\Delta t}$ is approximately

$$\frac{\tilde{\eta} \hat{g}_t}{N_g} \sum_{i=0}^{\infty} \left(1 - \frac{1}{N_g}\right)^i = \frac{\tilde{\eta} \hat{g}_t}{N_g} N_g = \tilde{\eta} \hat{g}_t$$

So the effect of $\hat{g}_t$ on the model is $\tilde{\eta} \hat{g}_t$ independent of N_g.

Decoupled Momentum

Using the same value of $\tilde{\eta}$ independent of N_g gives

$$\tilde{\eta} = B\eta_0 = N_g\eta$$

where, as before, η_0 is the optimal learning rate for Vanilla SGD with $B = 1$.

Solving for the PyTorch learning rate η we get

$$\eta = \frac{B\eta_0}{N_g}$$

which can then be used with the PyTorch implementation.

Decoupled Momentum

$$\eta = \frac{B\eta_0}{N_g}$$

Emprical evidence for this setting of η is given in

Don't Decay the Learning Rate, Increase the Batch Size, Smith et al., 2018

Decoupled Momentum Summary

By setting the PyTorch learning rate to be

$$\eta = \frac{B\eta_0}{N_g}$$

The optimnal value of η_0 should be fairly insensitive to the choice of B and N_g.

One should set the batch size B to be large enough so as to saturate the hardware (your GPUs should be running at maximum FLOPs).

END