CME 216, ME 343 - Spring 2020

Eric Darve, ICME

Stanford University

In this lecture, we are going to cover a few practical techniques to improve the convergence and accuracy of DNNs.

DNNs are quite difficult to train.

Deep Learning relies on advanced optimization algorithms to minimize the loss function.

The basic strategy to minimize the loss function is to update the weights using

$$\Delta W = -\alpha \nabla_W L$$

where L is the loss function and W are weights.

Choosing the learning rate α is not easy.

Let us take a simple example.

Let us assume that w is a scalar (we will use a lower case for convenience) and that L is quadratic

$$L = aw^2$$

The solution is trivial. The minimum is achieved at $w = 0$.

Using gradient descent, the update formula is

$$\Delta W = -\alpha \nabla_W L = -\alpha 2aw$$

The optimal learning rate in this simple example is

$$\alpha = \frac{1}{2a}$$

More generally, assume that W is a vector and that L is quadratic

$$L = \frac{1}{2} W^T H W$$

H is a matrix.

The gradient is

$$\nabla_W L = HW$$

Then the optimal gradient step is

$$\Delta W = -H^{-1}\nabla_W L = -W$$

Let us use an eigendecomposition of the symmetric matrix H

$$H = U \Lambda U^T$$

where U is orthogonal.

Define $Z = U^T W$.

The update

$$\Delta W = -\alpha \nabla_W L$$

is the same as

$$\Delta Z = -\alpha \Lambda Z$$

Thanks to the eigendecomposition, we can now work with the diagonal matrix $\mathbf{\Lambda}$.

We can now interpret how convergence is happening.

Each "mode" z_i in Z is updated using

$$\Delta z_i = -\alpha \lambda_i z_i$$

Ideally we want to pick $\alpha = \lambda_i^{-1}$ but this is not possible.

Let us assume that the eigenvalues are sorted by their magnitude:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

If we pick the largest value that corresponds to mode n:

$$\alpha = \lambda_n^{-1}$$

we get a very large step.

But this choice is unstable because the mode 1 will be updated using

$$\Delta z_1 = -\frac{\lambda_1}{\lambda_n} z_1$$

with $\lambda_1/\lambda_n \gg 1$. This mode is going to diverge.

So we are forced to choose the smallest learning rate

$$\alpha = \lambda_1^{-1}$$

Now mode 1 converges very fast. Mode n is going to converge very slowly.

So we have a choice between divergence and slow convergence.

The only practical option is to take a small learning rate and iterate for many steps.

We can also see that with this method the number of steps to take is roughly on the order of

$$\frac{\lambda_1}{\lambda_n}$$

Unfortunately, for most learning problems this is a large number. This means that many iterations are typically required to reach convergence.

In practice, it is difficult to estimate a priori what the learning rate should be.

The Hessian is difficult to even estimate and the quadratic approximation we used is only valid locally.