Derek Lee Deep Learning Fall 2020

Professor Curro Quiz #2

1. Explain the key differences between Adam and the basic gradient descent algorithm. Prose or mathematics equally acceptable.

Adam has several advantages over basic gradient descent. The "magnitudes of parameter updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the step size hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing." It updates exponential moving averages of the gradient and the squared gradient. The moving averages estimate the mean and the variance of the gradient. Because the moving averages are initialized to 0, initial timesteps have estimates biased towards 0. Unlike basic gradient descent, which is a first-order algorithm, Adam incorporates estimates of first-order and second-order information by using a short-term memory, similar to momentum, giving it a faster convergence. At the same time, it avoids some of the problems from computing second-order gradients in high dimensions. In addition, it requires little memory and is computationally efficient, giving it another advantage over basic gradient descent.

The update rule involves a careful choice of step sizes. This is in contrast with basic gradient descent, which uses a fixed step size. The step size has two upper bounds. The first upper bound is $\alpha \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}}$, when the second term ($\frac{1-\beta_1}{\sqrt{1-\beta_2}}$) is greater than one. The second upper bound is α, in all other cases. The first case only occurs "in the most severe case of sparsity: when a gradient has been zero at all timesteps except the current timestep." An example of this would be during the initial timesteps, when all moving averages have been initialized to 0. The effective magnitude of steps taken are approximately bounded by α. It is possible to deduce the right order of magnitude of α so the optima can be reached from the initial state within a set number of iterations. The effective step size depends on $\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}}$, with $\widehat{m}_t$ and $\sqrt{\widehat{v}_t}$ dependent on the gradients. Thus, rescaling the gradient will not change the effective step size, because the scaling factor will be cancelled out, which also means Adam is not hindered as much as gradient descent in areas with small gradients.

Adam corrects for the bias caused by initializing the moving average to 0 by introducing bias correction terms. It does this by comparing the expected value of the exponential moving average with the true second moment. The resulting term is used to correct the initialization bias.

2. Why does momentum really work?

Basic gradient descent has a significant weakness: speed. Initially, it has a large decrease in the loss. However, the decrease slows down over time. This can be caused by pathological curvature. Regions of a smooth function *f* may be valleys, trenches, and ravines. Iterations can become stuck in a valley or take small steps towards the optimum. Momentum introduces a 'short-term memory' to gradient descent. Its computational cost is almost nonexistent. This additional term is referred to as 'acceleration'. This helps gradient descent maintain its speed and can give a "quadratic speedup on many functions". It reduces the noise from gradient descent by using that short-term memory, pushing it closer to the optimum.

The paper derived the ranges for step size α and memory weight β that would still allow for convergence. Notably, α can be doubled, compared to basic gradient descent, before divergence. It also describes finding optimal values for α and β. Basic gradient descent is a special case of momentum, where $\beta = 0$. The paper compared momentum to a weight suspended on a spring. If β is too large, underdamping occurs, and the spring oscillates forever, missing the optimal value. If β is too small, overdamping occurs, and the weight is "immersed in a viscous fluid which saps… kinetic energy at every timestep." With the optimal value of β, the weight converges to the optimum.

With the optimal values of both α and β, the convergence rate is $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, essentially the square root of the convergence rate of basic gradient descent. The formulas for the optimal values reveal that "the optimal α is approximately twice that of gradient descent, and the momentum term is close to 1." The authors recommend setting β as close to 1 as possible, and then finding the highest α that still results in convergence. Being close to divergence is "a good place to be", just like in basic gradient descent.