Summary 13: On Optimization Methods for Deep Learning

Susan Cherry

The most common method for training in deep learning is Stochastic Gradient Descent (SGD). SGD is simple to implement and fast for problems with many training examples. SGD does have some weaknesses, however. They require a lot of manual tuning of optimization parameters. If the task isn't well known, it can be difficult to determine a good learning rate and good convergence criterion and using validation to choose parameters is computationally expensive. The sequential nature of SGD also makes it hard to parallelize.

This paper investigates two alternatives to SGD: Limited memory BFGS (L-BFGS) and Conjugate Gradient Descent (CGD). These methods are usually easier to check for convergence and are easy to parallelize. The benefits of L-BFGS come from using approximated second-order information. while the CG benefits come from using conjugacy information during optimization. The weakness of both the L-BFGS and CG is that they require the computation of the gradient on the entire dataset before they make an update so they don't scale well with the number of examples. This paper builds off of previous optimization research to empirically study the pros and cons of SGD, CGD, and L-BFGS in the context of deep learning and unsupervised learning.

Next the paper introduces several deep learning algorithms: TRestricted Boltzmann Machines (which aren't used because it is difficult to use methods besides SGD for them), Autoencoders and denoising autoencoders, Sparse RBMs and Autoencoders, and Tiled and locally connected networks.

Finally, the discuss their experiments which are carried out with the MNIST dataset. The experiments are carried out with SGDs, L-BFGS, and CG. First, they compare for training autoencoders. Minibatch L-BFGS and CG with line search converge faster than SGDs. CG performs better than L-BFGS, probably because comping the conjugate information is less expensive than estimating the Hessian. Next, we move to sparse autoencoder training. L-BGFS and CG are much faster than SGDs and the difference is much more significant than in the previous experiment. Next, they discuss training autoencoders with GPUs and find that the speed up gains are much higher for CG and L-BGFS than SGD. Then they move to parallel training of dense networks, where they find that parallelizing densely connected networks can result in slower convergence. Then they try parallel training of local networks and find that SGDs are slower when a computer cluster is sued. L-BGFS enjoy an almost linear speed up. Parallel training of supervised CNNs shows that L-BFGS is better than CG and SGDs. The final experiment is classification on standard MNIST. L-BFGS obtains the best result in this experiment.

Overall, the experiments show that different optimization algorithms are better on different problems. L-BFGS is good for low dimensional problems with a small number of problems. CG performs well on high dimensional problems. Both CG and work well for sparsity. Both L-BFGS and CG take better advantage of GPUs. In summary, CG and L-BFGS can offer large improvements over SGD in many cases.