unlike conjugate gradients, the success of the approach is not heavily dependent on the line search finding a point very close to the true minimum along the line. Thus, relative to conjugate gradients, BFGS has the advantage that it can spend less time refining each line search. On the other hand, the BFGS algorithm must store the inverse Hessian matrix, $\boldsymbol{M}$, that requires $O(n^2)$ memory, making BFGS impractical for most modern deep learning models that typically have millions of parameters.

Limited Memory BFGS (or L-BFGS) The memory costs of the BFGS algorithm can be significantly decreased by avoiding storing the complete inverse Hessian approximation $\boldsymbol{M}$. The L-BFGS algorithm computes the approximation $\boldsymbol{M}$ using the same method as the BFGS algorithm, but beginning with the assumption that $\boldsymbol{M}^{(t-1)}$ is the identity matrix, rather than storing the approximation from one step to the next. If used with exact line searches, the directions defined by L-BFGS are mutually conjugate. However, unlike the method of conjugate gradients, this procedure remains well behaved when the minimum of the line search is reached only approximately. The L-BFGS strategy with no storage described here can be generalized to include more information about the Hessian by storing some of the vectors used to update $\boldsymbol{M}$ at each time step, which costs only $O(n)$ per step.

8.7 Optimization Strategies and Meta-Algorithms

Many optimization techniques are not exactly algorithms, but rather general templates that can be specialized to yield algorithms, or subroutines that can be incorporated into many different algorithms.

8.7.1 Batch Normalization

Batch normalization (Ioffe and Szegedy, 2015) is one of the most exciting recent innovations in optimizing deep neural networks and it is actually not an optimization algorithm at all. Instead, it is a method of adaptive reparametrization, motivated by the difficulty of training very deep models.

Very deep models involve the composition of several functions or layers. The gradient tells how to update each parameter, under the assumption that the other layers do not change. In practice, we update all of the layers simultaneously. When we make the update, unexpected results can happen because many functions composed together are changed simultaneously, using updates that were computed under the assumption that the other functions remain constant. As a simple