

6.5.10 Higher-Order Derivatives

Some software frameworks support the use of higher-order derivatives. Among the deep learning software frameworks, this includes at least Theano and TensorFlow. These libraries use the same kind of data structure to describe the expressions for derivatives as they use to describe the original function being differentiated. This means that the symbolic differentiation machinery can be applied to derivatives.

In the context of deep learning, it is rare to compute a single second derivative of a scalar function. Instead, we are usually interested in properties of the Hessian matrix. If we have a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, then the Hessian matrix is of size $n \times n$. In typical deep learning applications, n will be the number of parameters in the model, which could easily number in the billions. The entire Hessian matrix is thus infeasible to even represent.

Instead of explicitly computing the Hessian, the typical deep learning approach is to use **Krylov methods**. Krylov methods are a set of iterative techniques for performing various operations like approximately inverting a matrix or finding approximations to its eigenvectors or eigenvalues, without using any operation other than matrix-vector products.

In order to use Krylov methods on the Hessian, we only need to be able to compute the product between the Hessian matrix \mathbf{H} and an arbitrary vector \mathbf{v} . A straightforward technique (Christianson, 1992) for doing so is to compute

$$\mathbf{H}\mathbf{v} = \nabla_{\mathbf{x}} \left[(\nabla_{\mathbf{x}} f(x))^{\top} \mathbf{v} \right]. \quad (6.59)$$

Both of the gradient computations in this expression may be computed automatically by the appropriate software library. Note that the outer gradient expression takes the gradient of a function of the inner gradient expression.

If \mathbf{v} is itself a vector produced by a computational graph, it is important to specify that the automatic differentiation software should not differentiate through the graph that produced \mathbf{v} .

While computing the Hessian is usually not advisable, it is possible to do with Hessian vector products. One simply computes $\mathbf{H}\mathbf{e}^{(i)}$ for all $i = 1, \dots, n$, where $\mathbf{e}^{(i)}$ is the one-hot vector with $e_i^{(i)} = 1$ and all other entries equal to 0.

6.6 Historical Notes

Feedforward networks can be seen as efficient nonlinear function approximators based on using gradient descent to minimize the error in a function approximation.