where $i, j \in \{1, \ldots, W\}$ and W is the total number of weights and biases. The Hessian matrix arises in several nonlinear optimization algorithms used for training neural networks based on considerations of the second-order properties of the error surface (Bishop, 2006). It also plays a role in some Bayesian treatments of neural networks (MacKay, 1992; Bishop, 2006) and has been used to reduce the precision of the weights in large language models to lessen their memory footprint (Shen *et al.*, 2019).

An important consideration for many applications of the Hessian is the efficiency with which it can be evaluated. If there are W parameters (weights and biases) in the network, then the Hessian matrix has dimensions $W \times W$ and so the computational effort needed to evaluate the Hessian will scale like $\mathcal{O}(W^2)$ for each point in the data set. Extension of the backpropagation procedure (Bishop, 1992) allows the Hessian matrix to be evaluated efficiently with a scaling that is indeed $\mathcal{O}(W^2)$. Sometimes, we do not need the Hessian matrix explicitly but only the product $\mathbf{v}^{\mathrm{T}}\mathbf{H}$ of the Hessian with some vector $\mathbf{v}$, and this product can be calculated efficiently in $\mathcal{O}(W)$ steps using an extension of backpropagation (Møller, 1993; Pearlmutter, 1994).

Exercise 8.6

Since neural networks may contain millions or even billions of parameters, evaluating, or even just storing, the full Hessian matrix for many models is infeasible. Evaluating the inverse of the Hessian is even more demanding as this has $\mathcal{O}(W^3)$ computational scaling. Consequently there is interest in finding effective approximations to the full Hessian.

One approximation involves simply evaluating only the diagonal elements of the Hessian and implicitly setting the off-diagonal elements to zero. This requires $\mathcal{O}(W)$ storage and allows the inverse to be evaluated in $\mathcal{O}(W)$ steps but still requires $\mathcal{O}(W^2)$ computation (Ricotti, Ragazzini, and Martinelli, 1988), although with further approximation this can be reduced to $\mathcal{O}(W)$ steps (Becker and LeCun, 1989; LeCun, Denker, and Solla, 1990). In practice, however, the Hessian generally has significant off-diagonal terms, and so this approximation must be treated with care.

A more convincing approach, known as the *outer product approximation*, is obtained as follows. Consider a regression application using a sum-of-squares error function of the form

$$E = \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 \tag{8.38}$$

Exercise 8.8

where we have considered a single output to keep the notation simple (the extension to several outputs is straightforward). We can then write the Hessian matrix in the form

$$\mathbf{H} = \nabla\nabla E = \sum_{n=1}^{N}\nabla y_n(\nabla y_n)^{\mathrm{T}} + \sum_{n=1}^{N}(y_n - t_n)\nabla\nabla y_n \tag{8.39}$$

where ∇ denotes the gradient with respect to $\mathbf{w}$. If the network has been trained on the data set and its outputs y_n are very close to the target values t_n, then the final term in (8.39) will be small and can be neglected. More generally, however, it may be appropriate to neglect this term based on the following argument. Recall from Section 4.2 that the optimal function that minimizes a sum-of-squares loss is