Deep Learning - Foundations and Concepts

Chapter 8. Backpropagation

nonlineark@github

February 27, 2025

Outline

Single-layer networks

Consider a simple linear model:

$$y_k = \sum_i w_{ki} x_i$$

together with a sum-of-squares error function:

$$E_n = \frac{1}{2} \sum_k (y_k^n - t_k^n)^2$$

The gradient of this error function with respect to a weight w_{ji} is given by:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{1}{2} \sum_k 2(y_k^n - t_k^n) \frac{\partial y_k^n}{\partial w_{ji}} = (y_j^n - t_j^n) x_i^n$$

This is a local computation involving:

- An error signal $y_j^n - t_j^n$ associated with the output end.
- The variable x_i^n associated with the input end.

General feed-forward networks

Consider a unit in a feed-forward network:

$$a_j = \sum_i w_{ji} z_i \qquad z_j = h(a_j)$$

Now consider the evaluation of the derivative of E_n with respect to a weight w_{ji}:

$$\frac{\partial E_n}{\partial w_{ji}} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

If we introduce $\delta_j = \frac{\partial E_n}{\partial a_j}$, and use the fact that $\frac{\partial a_j}{\partial w_{ji}} = z_i$, we have:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$

This takes the same form as that found for the simple linear model.

General feed-forward networks

To evaluate the derivatives, we need calculate only the value of δ_j for each hidden and output unit. For output units:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial y_k}\frac{\partial y_k}{\partial a_j} = y_j - t_j$$

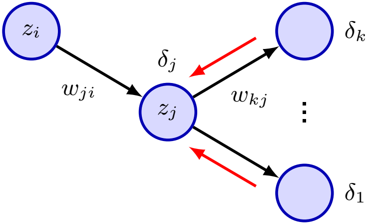
provided we are using the canonical link as the output-unit activation function. For hidden units:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial z_j}\frac{\mathrm{d} z_j}{\mathrm{d} a_j} = h'(a_j)\sum_k w_{kj}\delta_k$$

which tells us that the value of δ for a particular hidden unit can be obtained by propagating the δ's backwards from units higher up in the network.

General feed-forward networks

Figure: Illustration of the calculation of δ_j for hidden unit j



General feed-forward networks

Algorithm 1: Backpropagation

for $j \in$ *all hidden and output units* **do**
    $a_j \leftarrow \sum_i w_{ji} z_i$;
    $z_j \leftarrow h(a_j)$;
end
for $k \in$ *all output units* **do**
    $\delta_k \leftarrow \frac{\partial E_n}{\partial a_k}$;
end
for $j \in$ *all hidden units* **do**
    $\delta_j \leftarrow h'(a_j) \sum_k w_{kj} \delta_k$;
    $\frac{\partial E_n}{\partial w_{ji}} \leftarrow \delta_j z_i$;
end
return ∇E_n;

Numerical differentiation

One of the most important aspects of backpropagation is its $\mathcal{O}(W)$ computational efficiency. Consider an alternative approach to use finite differences with:

$$\frac{\partial E_n}{\partial w_{ji}} \approx \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji})}{\epsilon}$$

or use symmetrical central differences for better accuracy of the approximation:

$$\frac{\partial E_n}{\partial w_{ji}} \approx \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon}$$

There are W weights in the network each of which must be perturbated individually, so that the overall computational cost is $\mathcal{O}(W^2)$.

The Jacobian matrix

Here we consider the network outputs as a function of the network inputs, and calculate the Jacobian of this function:

$$J_{ki} = \frac{\partial y_k}{\partial x_i} = \sum_j \frac{\partial y_k}{\partial a_j}\frac{\partial a_j}{\partial x_i} = \sum_j w_{ji}\frac{\partial y_k}{\partial a_j}$$

$$\frac{\partial y_k}{\partial a_j} = \sum_l \frac{\partial y_k}{\partial a_l}\frac{\partial a_l}{\partial a_j} = \sum_l \frac{\partial y_k}{\partial a_l}\frac{\partial a_l}{\partial z_j}\frac{\mathrm{d}z_j}{\mathrm{d}a_j} = h'(a_j)\sum_l w_{lj}\frac{\partial y_k}{\partial a_l}$$

where in the first equation, the sum runs over all units j to which the input unit i sends connections, while for the second equation, the sum runs over all units l to which unit j sends connections. This backpropagation starts at the output units, for which the required derivatives can be found directly from the functional form of the output-unit activation function.

The Hessian matrix

Here we consider all the weight and bias parameters as elements w_i of a single vector w, and calculate the second derivatives of the error function E with respect to w:

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$$

Extension of the backpropagation procedure allows the Hessian matrix to be evaluated efficiently in $\mathcal{O}(W^2)$ steps. But because the huge number of parameters, evaluating the Hessian matrix for many models is infeasible. Thus there is interest in finding effective approximations to the full Hessian.

The Hessian matrix

The outer product approximation. Consider a regression application using a sum-of-squares error function of the form:

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

$$D^2E(w) = \sum_{n=1}^{N}(Dy_n(w))^2 + \sum_{n=1}^{N}(y_n - t_n)D^2y_n(w)$$

We can ignore the second term because the quantity $y_n - t_n$ is a random variable with zero mean, which leads us to:

$$H \approx \sum_{n=1}^{N}\nabla y_n \nabla y_n^T = \sum_{n=1}^{N}\nabla a_n \nabla a_n^T$$