Neural Network / Multilayer Perceptron

Derek Xiang, Chi Chen, Jiayi Fang

Spring 2022

1 Introduction

The feed-forward neural network, also known as the multilayer perceptron, is a model for regression and classification that is comprised of adaptive basis functions. The model is comprised of multiple layers of logistic regression models that are classified by nonlinear activation functions. The general linear model for regression and classification is as follows.

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right) \qquad (*)$$

For the MLP (Multilayer Perceptron) model, $\phi_j(x)$ depends on parameters that can be adjusted along with $\{w_j\}$. In this case, $\phi_j(x)$ represent the basis functions and $\{w_j\}$ represent the weights corresponding to $\{x_j\}$, the data that the model is trained on. The general linear model, (*), can be extended to an MLP using the following composition.

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right)$$

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

h and σ are understood to be arbitrary differentiable nonlinear activation functions, although σ is usually taken to be the logistic sigmoid function.

$$\sigma(a) = \frac{1}{1 + exp(-a)}$$

Now that we have the MLP model, we can apply error backpropagation to efficiently evalute the gradient of the error function associated with the MLP. The error function that we use is a standard least squares error function between the computed model, $y(\mathbf{x}, \mathbf{w})$, and the target values, $\{t_k\}$, that the model is trying to predict with the training data. $E(\mathbf{w})$ is the error function associated with the MLP model.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2$$

We consider the case when the data we are dealing with is i.i.d (independent and identically distributed), which is most practical for our model. The error function can be decomposed into individual error functions for each data point in the training set. We evaluate the gradient for each on of these error functions separately. The procedure is listed out below explicitly.

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) = \sum_{n=1}^{N} \left(\frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \right)$$

$$\nabla E_n(\mathbf{w}) = \frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}$$

These results are obtained by direct computation. By applying the chain rule to the second expression, we get

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

Applying the chain rule to the first partial derivative on the right hand side gives us

$$\frac{\partial E_n}{\partial a_j} = \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \qquad (**)$$

The derivative in the previous equation is evaluated as follows.

$$a_j = \sum_i w_{ji} z_i$$

$$z_j = h(a_j)$$

$$a_k = \sum_j w_{kj} z_j$$

$$\frac{\partial a_k}{\partial a_j} = w_{kj} h'(a_j)$$

$$\frac{\partial E_n}{\partial a_j} = h'(a_j) \sum_k w_{kj} \frac{\partial E_n}{\partial a_k}$$

Note that $\frac{\partial a_j}{\partial w_{ji}} = z_i$ and also that $\frac{\partial E_n}{\partial a_k} = y_k - t_k$. The end result follows immediately. In terms of efficiency, error backpropagation scaling is $O(W^2)$. Backpropagation can be further used to evaluate the second derivatives of the error function. We consider the case when the Hessian of the error function has a diagonal approximation, that is to say when all of its non-diagonal values are set to zero. In this case, the inverse of the Hessian is trivial to compute. The diagonal along the Hessian matrix is evaluated as follows. For each second partial derivative along the diagonal, applying the chain rule, we get

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

$$\frac{\partial^2 E_n}{\partial w_{ji}^2} = \frac{\partial^2 E_n}{\partial a_j^2} \left(\frac{\partial a_j}{\partial w_{ji}} \right)^2 = \frac{\partial^2 E_n}{\partial a_j^2} z_i^2$$

Then evaluating the partial derivative on the right hand side of the previous equation by using the chain rule as in (**) using the same backpropagation method gives us

$$\frac{\partial^2 E_n}{\partial a_j^2} = h'(a_j)^2 \sum_k \sum_{k'} w_{kj} w_{k'j} \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} + h''(a_j) \sum_k w_{kj} \frac{\partial E_n}{\partial a_k}$$

Only counting the diagonal elements, in the case when the Hessian is evaluated using its diagonal approximate, we get

$$\frac{\partial^2 E_n}{\partial a_j^2} = h'(a_j)^2 \sum_k \sum_{k'} w_{kj}^2 \frac{\partial^2 E_n}{\partial a_k^2} + h''(a_j) \sum_k w_{kj} \frac{\partial E_n}{\partial a_k}$$

Using error backpropagation to evaluate the diagonal approximate of the Hessian has scaling $O(W)$.

References

[1] Christopher M. Bishop. Pattern Recognition and Machine Learning. New York, Springer, 2006.