Summary of: Learning Representations by Back-Propagating Errors

Sungji Wang[1]

[1]Seoul National University, Seoul 151-747, Korea

June 3, 2024

Abstract

In this document, I will provide a summary of the 1986 Nature paper "Learning Representations by Back-Propagating Errors". Unofficial source code fetched online can be found at the repository.

1 Mathematical Preliminary

Equations used in the paper should be easy to follow. Just note that ∂ indicates partial derivative, and the chain rule holds for partial derivatives as well.

The concept of gradient descent is dealt with importantly in the paper. The idea of gradient descent is to take repeated steps in the opposite direction of the gradient, which is the steepest direction of change in a multivariate function.

2 Summary

The paper provides a detailed description of the backpropagation algorithm, which is a method for training multilayer neural networks. The neural network described here is simple. It has a layer of input units at the bottom, any number of intermediate layers, and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is basically the states of the input units. Other units in each layer are determined recursively, following equations 1 and 2. Note that x_j is the total input, whereas y_j is the total output. Weight w_{ji} is the weight from y_i to x_j.

$$x_j = \sum_i y_i w_{ji} \tag{1}$$

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{2}$$

As mentioned in the paper, any input-output function which has a bounded derivative instead of equations 1 and 2 works just as fine. It is well-known that the combination of those two can approximate any continuous function. However, the use of a linear function combining the inputs to a unit before applying the nonlinearity is needed for faster (in modern days, GPU) and better approximation.

Now that we have defined the neural network, we must set our goal. Basically, the goal is to minimize the error of prediction. The error function is defined as follows. c is an index over input-output pairs, j is an index over output units, y is the actual state of an output unit and d is its desired state.

$$E = \frac{1}{2}\sum_c \sum_j (y_{j,c} - d_{j,c})^2 \tag{3}$$

To minimize E by gradient descent, one must compute the partial derivative of E with respect to each weight in the network. This is simply the sum of the partial derivatives for each of the input-output cases (because the sum of derivatives is the derivative of the sum). For a given case, the partial derivatives of the error with respect to each weight are computed in two passes. The forward pass in which the units in each layer have their states determined by the input they receive from units in lower layers is shown using equations 1 and 2. The backward pass, which propagates derivatives from the top layer back to the bottom one, is more complicated, and this backpropagation mechanism is the main work of the paper.

The equations are actually quite simple. Differentiating 3 while fixing c gives:

$$\frac{\partial E}{\partial y_j} = y_j - d_j \tag{4}$$

Applying the chain rule gives:

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{\mathrm{d}y_j}{\mathrm{d}x_j} \tag{5}$$

Using equation 2 gives:

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} y_j(1 - y_j) \tag{6}$$

Now applying the chain rule again can give the partial derivative of the sum by weight:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial y_i} \tag{7}$$

Taking into account all the connections emanating from unit i, we have:

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} w_{ji} \tag{8}$$

Note that index i is for the lower layer, whereas j is for the upper layer. This means that the backward computation of the gradient is possible.