Backpropagation Algorithm

Robert Jacobs
Department of Brain & Cognitive Sciences
University of Rochester
Rochester, NY 14627, USA

August 8, 2008

The backpropagation algorithm is a method for computing partial derivatives in a network. In short, it is nothing more (nor less) than the chain rule from calculus. One of its virtues is that it is an extremely efficient computation due to its recursive formulation. Note that while it is most commonly used to compute first derivatives, it may also be used to compute second derivatives (or derivatives of any order), albeit at additional computational expense.

For the purposes of explanation, let's assume that we are training a network with three layers: an input layer which is connected to a hidden layer which, in turn, is connected to an output layer. The first thing to do is to select a log likelihood function that is appropriate for the nature of the task. For simplicity, let's assume that the task is a regression task (in the appendices we will consider binary and multiway classification tasks) in which the target output $\mathbf{y}^*(t)$ is a noise-corrupted version of some function of the input $\mathbf{x}(t)$:

$$\mathbf{y}^*(t) = f[\mathbf{x}(t)] + \epsilon \tag{1}$$

where $\epsilon \sim N(0,1)$ [that is, $\mathbf{y}^*(t) \sim N(f[\mathbf{x}(t)], 1)$]. The log likelihood function is:

$$\begin{aligned} \log L &= \sum_t -\frac{1}{2} \parallel \mathbf{y}^*(t) - \mathbf{y}(t) \parallel^2 && (2) \\ &= \sum_t \sum_i -\frac{1}{2} [y_i^*(t) - y_i(t)]^2 && (3) \end{aligned}$$

where the index t ranges over all training patterns, and the index i ranges over all output units (please make sure that you understand why the log likelihood function takes this form).

We want to compute the derivatives of the log likelihood with respect to the network's weights. Let's assume that we are doing "batch" training, meaning that the weights remain constant within an epoch (a pass through all data patterns) but change between epochs. For notational convenience, two time indexes will be used: n indexes epochs and t indexes steps (pattern presentations within an epoch). Using the chain rule, computation of the desired derivatives may be broken up into three stages. It is easiest if we consider output units and hidden units separately. For an output unit, the stages are represented by the three partial derivatives on the right-hand side of the equation:

$$\frac{\partial \log L(n)}{\partial w_{ij}(n)} = \sum_t \frac{\partial \log L(t)}{\partial y_i(t)} \frac{\partial y_i(t)}{\partial s_i(t)} \frac{\partial s_i(t)}{\partial w_{ij}(n)} \tag{4}$$

where $s_i(t)$ is the weighted sum of unit i's inputs at step t, and $w_{ij}(n)$ is the weight on the connection between hidden unit j and output unit i at epoch n. For a hidden unit, the stages are:

$$\frac{\partial \log L(n)}{\partial w_{ij}(n)} = \sum_t \frac{\partial \log L(t)}{\partial h_i(t)} \frac{\partial h_i(t)}{\partial s_i(t)} \frac{\partial s_i(t)}{\partial w_{ij}(n)} \tag{5}$$

where we denote hidden unit i's activation at step t by $h_i(t)$ (and, similar to before, $s_i(t)$ is the weight sum of unit i's inputs, and $w_{ij}(n)$ is the weight on the connection between input unit j and hidden unit i).

We first consider output units. In the case considered here,

$$\frac{\partial \log L(t)}{\partial y_i(t)} = y_i^*(t) - y_i(t). \tag{6}$$

For a regression task, the output units use a linear activation function:

$$y_i(t) = s_i(t) = \sum_j w_{ij}(n) h_j(t) \tag{7}$$

where j indexes over the hidden units. Therefore,

$$\frac{\partial y_i(t)}{\partial s_i(t)} = 1, \tag{8}$$

and

$$\frac{\partial s_i(t)}{\partial w_{ij}(n)} = h_j(t). \tag{9}$$

For a hidden unit, the derivative of the log likelihood with respect to the hidden unit's activation is a bit tricky to compute. This value is

$$\frac{\partial \log L(t)}{\partial h_i(t)} = \sum_{k=1}^{K} \frac{\partial \log L(t)}{\partial s_k(t)} \, w_{ki}(n) \tag{10}$$

where k indexes over the output units (so there are K output units, $s_k(t)$ is the weighted sum of output unit k's inputs, and $w_{ki}(n)$ is the weight on the connection between hidden unit i and output unit k). Note that the algorithm is sending error signals from the output units back to the hidden units (hence, the name "backpropagation"). For the case considered here (i.e., regression task), this equation may be re-written:

$$\frac{\partial \log L(t)}{\partial h_i(t)} = \sum_{k=1}^{K} [y_k^*(t) - y_k(t)] \, w_{ki}(n). \tag{11}$$

The hidden units typically use the logistic activation function:

$$h_i(t) = \frac{1}{1 + e^{-s_i(t)}} = \frac{1}{1 + e^{-\sum_j w_{ij}(n) x_j(t)}} \tag{12}$$

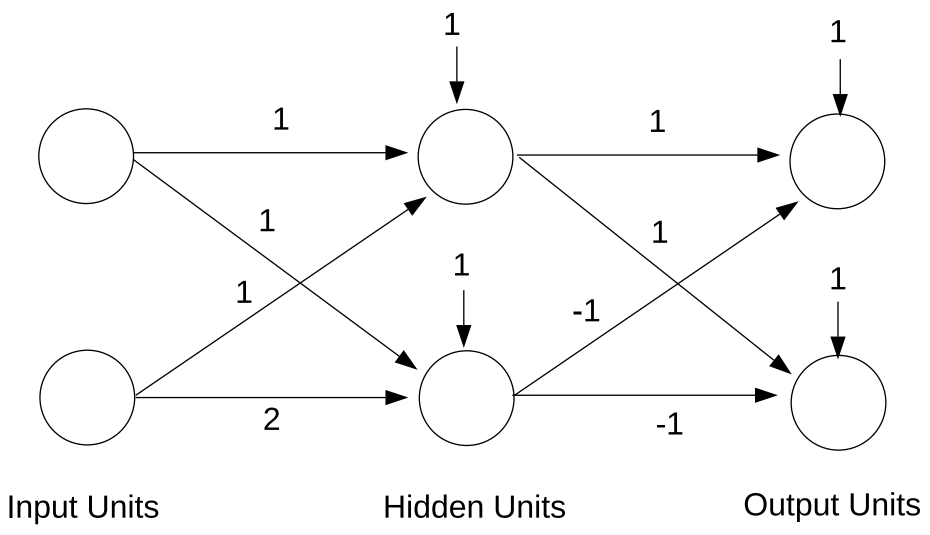

Figure 1: A sample network.

where j indexes over the input units. Therefore,

$$\frac{\partial h_i(t)}{\partial s_i(t)} = h_i(t)\ [1 - h_i(t)], \tag{13}$$

and

$$\frac{\partial s_i(t)}{\partial w_{ij}(n)} = x_j(t). \tag{14}$$

Recall that we are performing gradient ascent on the log likelihood function. The update formula for the weights is

$$w_{ij}(n+1) = w_{ij}(n) + \epsilon\ \frac{\partial \log L(n)}{\partial w_{ij}(n)} \tag{15}$$

where ϵ is a step-size (or learning rate) parameter.

I won't go into it here, but please make sure that you understand how this algorithm is used in the cases of: (i) bias weights; and (ii) on-line training.

For the sake of clarity, let's go through a simple example. Suppose that we have a network with two input units that are connected to two hidden units which, in turn, are connected to two output units. Let's suppose that at some time step, the connection weights on the connections from the first input unit to the two hidden units are each equal to 1, that the weight on the connection from the second input unit to the first hidden unit has a value of 1, and that the weight on the connection from the second input unit to the second hidden unit has a value of 2 (see Figure 1). The two hidden units also have bias weights which are each equal to 1.

Suppose that the connection weights from hidden unit one to the two output units are equal to 1, and that the connection weights from hidden unit two to the two output units are equal to -1. The output units have bias weights equal to 1.

Suppose that the input units have activations of 1 and 2 ($x_1 = 1, x_2 = 2$). Then the weighted sum of the first hidden unit's inputs (plus its bias weight) is equal to 4.0, and this hidden unit has an activation of 0.982; that is, $h_1 = 0.982$ (please make sure that you know how to derive these values). The weighted sum of the second hidden unit is 6.0, and its activation value is 0.998.

The weighted sum of the first output unit is 0.984 (this is also its activation because this is a linear unit). The weighted sum of the second output unit is also 0.984.

Suppose that the output units have target activations of 1.0 and 0.0, respectively. We first consider the derivatives of the log likelihood with respect to the output units' bias weights. The derivative for the first output unit's bias is (1.0 - 0.984); the derivative for the second output unit's bias is (0.0 - 0.984). The derivatives for the hidden-to-output weights are as follows:

$$\begin{aligned}
\frac{\partial \log L(t)}{\partial w_{11}(n)} &= (1.0 - 0.984)(0.982) && (16)\\
\frac{\partial \log L(t)}{\partial w_{12}(n)} &= (1.0 - 0.984)(0.998) && (17)\\
\frac{\partial \log L(t)}{\partial w_{21}(n)} &= (0.0 - 0.984)(0.982) && (18)\\
\frac{\partial \log L(t)}{\partial w_{22}(n)} &= (0.0 - 0.984)(0.998). && (19)
\end{aligned}$$

The derivative for the first hidden unit's bias weight is

$$((1.0 - 0.984)(1.0) + (0.0 - 0.984)(1.0)) * ((0.982)(1.0 - 0.982)). \tag{20}$$

The derivative for the second hidden unit's bias weight is

$$((1.0 - 0.984)(-1.0) + (0.0 - 0.984)(-1.0)) * ((0.998)(1.0 - 0.998)). \tag{21}$$

The derivatives for the input-to-hidden weights are as follows:

$$\begin{aligned}
\frac{\partial \log L(t)}{\partial w_{11}(n)} &= ((1.0 - 0.984)(1.0) + (0.0 - 0.984)(1.0)) * ((0.982)(1.0 - 0.982)) * 1.0 && (22)\\
\frac{\partial \log L(t)}{\partial w_{12}(n)} &= ((1.0 - 0.984)(1.0) + (0.0 - 0.984)(1.0)) * ((0.982)(1.0 - 0.982)) * 2.0 && (23)\\
\frac{\partial \log L(t)}{\partial w_{21}(n)} &= ((1.0 - 0.984)(-1.0) + (0.0 - 0.984)(-1.0)) * ((0.998)(1.0 - 0.998)) * 1.0 && (24)\\
\frac{\partial \log L(t)}{\partial w_{22}(n)} &= ((1.0 - 0.984)(-1.0) + (0.0 - 0.984)(-1.0)) * ((0.998)(1.0 - 0.998)) * 2.0. && (25)
\end{aligned}$$

Please make sure that you understand where all these numbers come from and why they make sense.

Appendix A: Binary Classification

In this appendix we modify what was stated above for the purpose of using the backpropagation algorithm in the context of a binary classification task. That is, we want to classify the input $\mathbf{x}(t)$ as belonging to one of two possible classes. As a matter of notation, the two classes are denoted 0 and 1 (i.e. $y^*(t) = 0$ or $y^*(t) = 1$). The class label (i.e. the target output $y^*(t)$) is a probabilistic function of the input $\mathbf{x}(t)$, and has a Bernoulli distribution:

$$\begin{aligned} p(y^*(t) = 1|\mathbf{x}(t)) &= p = y(t) && (26) \\ p(y^*(t) = 0|\mathbf{x}(t)) &= 1 - p = 1 - y(t). && (27) \end{aligned}$$

In other words,

$$p(y^*(t)|\mathbf{x}(t)) = y(t)^{y^*(t)}(1 - y(t))^{1-y^*(t)}. \tag{28}$$

Note that, as before, we are setting $y(t)$ to the expected value of $y^*(t)$. The log likelihood function is the log of a Bernoulli distribution:

$$\log L = \sum_t y^*(t) \log y(t) + (1 - y^*(t)) \log(1 - y(t)) \tag{29}$$

where t indexes over all training patterns (please make sure that you understand why the log likelihood takes this form).

We want to compute the derivatives of the log likelihood with respect to the output unit's weights (note that, for the sake of simplicity, we are considering a case in which there is only a single output unit). Using the chain rule, we get:

$$\frac{\partial \log L(n)}{\partial w_j(n)} = \sum_t \frac{\partial \log L(t)}{\partial y(t)} \frac{\partial y(t)}{\partial s(t)} \frac{\partial s(t)}{\partial w_j(n)} \tag{30}$$

where $s(t)$ is the weighted sum of the unit's inputs at time t, and $w_j(n)$ is the weight on the connection between hidden unit j and the output unit at epoch n. The first term is given by

$$\frac{\partial \log L(t)}{\partial y(t)} = \frac{y^*(t)}{y(t)} - \frac{1 - y^*(t)}{1 - y(t)}. \tag{31}$$

For a binary classification task, the output unit uses a logistic activation function:

$$y(t) = \frac{1}{1 + e^{-s(t)}} = \frac{1}{1 + e^{-\sum_j w_j(n) h_j(t)}}. \tag{32}$$

Therefore

$$\frac{\partial y(t)}{\partial s(t)} = y(t) * (1 - y(t)), \tag{33}$$

and

$$\frac{\partial s(t)}{\partial w_j(n)} = h_j(t). \tag{34}$$

In regard to the hidden units, their derivatives can be computed in the same way as was discussed in the main body of this document.

Appendix B: Multiway Classification

In this appendix we modify what was stated above for the purpose of using the backpropagation algorithm in the context of a multiway classification task. That is, we want to classify the input $\mathbf{x}(t)$ as belonging to one of several possible classes (where there are three possible classes or more). As a matter of notation, there is a binary indicator variable associated with each class; i.e. $y_i^*(t) = 1$ if the input $\mathbf{x}(t)$ belongs to class i, otherwise $y_i^*(t) = 0$ (note that exactly one indicator variable is set to one, all others are set to zero). The class (that is, the binary indicator variables or the target outputs) given the input $\mathbf{x}(t)$ has a multinomial distribution:

$$p(\mathbf{y}^*(t)|\mathbf{x}(t)) = \prod_{k=1}^{K} y_k(t)^{y_k^*(t)} \tag{35}$$

where k indexes the classes (or output units) and K is the number of possible classes. The log likelihood function is a log of a multinomial distribution:

$$\log L = \sum_t \sum_k y_k^*(t) \log y_k(t) \tag{36}$$

where the index t ranges over al training patterns.

We assume that the output units use the softmax activation function:

$$y_i(t) = \frac{\exp[s_i(t)]}{\sum_k \exp[s_k(t)]} \tag{37}$$

where $s_i(t)$ is output unit i's weighted sum of inputs. We want to compute the derivatives of the log likelihood with respect to the output units' weights. Importantly, note that the softmax function introduces a complication—this function is non-local because the output of unit i depends on the weighted sum of unit $k, k = 1, \ldots, K$. Consequently, we must take these non-local dependencies into account. We compute the relevant derivatives as follows:

$$\frac{\partial \log L(n)}{\partial w_{ij}(n)} = \sum_t \frac{\partial \log L(t)}{\partial s_i(t)} \frac{\partial s_i(t)}{\partial w_{ij}(n)}. \tag{38}$$

The first term is

$$\begin{aligned} \frac{\partial \log L(t)}{\partial s_i(t)} &= \sum_k \frac{\partial \log L(t)}{\partial y_k(t)} \frac{\partial y_k(t)}{\partial s_i(t)} && (39) \\ &= \sum_k \frac{y_k^*(t)}{y_k(t)} y_k(t)(\delta_{ik} - y_i(t)) && (40) \end{aligned}$$

where $\delta_{ik} = 1$ if $i = k$, otherwise it equals 0. (Please make sure that you understand how this equation takes care of the non-local dependencies.) The final term is

$$\frac{\partial s_i(t)}{\partial w_{ij}(n)} = h_j(t). \tag{41}$$

In regard to the hidden units, their derivatives can be computed in the same way as was discussed in the main body of this document.