# Notes on Backpropagation

**Peter Sadowski**
Department of Computer Science
University of California Irvine
Irvine, CA 92697
`peter.j.sadowski@uci.edu`

## Abstract

This document derives backpropagation for some common error functions and describes some other tricks. I wrote this up because I kept having to reassure myself of the algebra.

## 1 Cross Entropy Error with Logistic Activation

Often we have multiple probabilistic output units that do not represent a simple classification task, and are therefore not suited for a softmax activation function, such as an autoencoder network on binary data. The sum of squares error is ill-suited because we still desire the probabilistic interpretation of the output. Instead, we can generalize the two-class case in which we sum of the cross entropy errors for multiple output units, where each output is itself a two-class problem. This is simply the expectation of the loglikelihood of independent targets.

The cross entropy error for a single example is

$$E = -\sum_{i=1}^{nout} \left(t_i \log(o_i) + (1 - t_i)\log(1 - o_i)\right) \tag{1}$$

where $t$ is the target, $o$ is the output, indexed by $i$. Our activation function is the logistic

$$o_i = \frac{1}{1 + e^{-x_i}} \tag{2}$$

$$x_i = \sum_{j=1} o_j w_{ji} \tag{3}$$

Our first goal is to compute the error gradient for the weights in the final layer $\frac{\partial E}{\partial w_{ji}}$. We begin by observing,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_i}\frac{\partial o_i}{\partial x_i}\frac{\partial x_i}{\partial w_{ji}} \tag{4}$$

and proceed by analyzing the partial derivatives for specific data points,

$$\frac{\partial E}{\partial o_i} = \frac{-t_i}{o_i} + \frac{1 - t_i}{1 - o_i} \tag{5}$$

$$= \frac{o_i - t_i}{o_i(1 - o_i)} \tag{6}$$

$$\frac{\partial o_i}{\partial x_i} = o_i(1 - o_i) \tag{7}$$

$$\frac{\partial x_i}{\partial w_{ji}} = o_j \tag{8}$$

where $o_j$ is the activation of the $j$ node in the hidden layer. Combining things back together,

$$\frac{\partial E}{\partial x_i} = o_i - t_i \tag{9}$$

and

$$\frac{\partial E}{\partial w_{ji}} = (o_i - t_i)o_j \tag{10}$$

.

This gives us the gradient for the weights in the last layer of the network. We now need to calculate the error gradient for the weights of the lower layers. Here it is useful to calculate the quantity $\frac{\partial E}{\partial x_j}$ where $j$ indexes the units in the second layer down.

$$\begin{aligned}\frac{\partial E}{\partial x_j} &= \sum_{i=1}^{nout} \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial o_j}\frac{\partial o_j}{\partial x_j} &(11)\\ &= \sum_{i=1}^{nout}(o_i - t_i)(w_{ji})(o_j(1-o_j)) &(12)\end{aligned}$$

Then a weight $w_{kj}$ connecting units in the second and third layers down has gradient

$$\begin{aligned}\frac{\partial E}{\partial w_{kj}} &= \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{kj}} &(13)\\ &= \sum_{i=1}^{nout}(o_i - t_i)(w_{ji})(o_j(1-o_j))(o_k) &(14)\end{aligned}$$

In conclusion, to compute $\frac{\partial E}{\partial w_{ij}}$ for a general multilayer network, we simply need to compute $\frac{\partial E}{\partial x_j}$ recursively, then multiply by $\frac{\partial x_j}{\partial w_{kj}} = o_k$.

## 2 Classification with Softmax Transfer and Cross Entropy Error

For classification problems with more than 2 classes, we need a way of assigning probabilities to each class. The softmax activation function allows us to do this while still using the cross entropy error and having a nice gradient to descend. (Note that this cross entropy error function is modified for multiclass output.) It turns out to be the same gradient as for the summed cross entropy case. The softmax activation of the $i$th output unit is

$$f_i(w,x) = \frac{e^{x_i}}{\sum_k^{nclass} e^{x_k}} \tag{15}$$

and the cross entropy error function for multi-class output is

$$E = -\sum_i^{nclass} t_i \log(o_i) \tag{16}$$

So

$$\begin{aligned}\frac{\partial E}{\partial o_i} &= -\frac{t_i}{o_i} &(17)\\ \frac{\partial o_i}{\partial x_k} &= \begin{cases} o_i(1-o_i) & i = k \\ -o_i o_k & i \neq k \end{cases} &(18)\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial x_i} &= \sum_k^{nclass} \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial x_i} && (19)\\
&= \frac{\partial E}{\partial o_i}\frac{\partial o_i}{\partial x_i} - \sum_{k\neq i} \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial x_i} && (20)\\
&= -t_i(1-o_i) + \sum_{k\neq i} t_k o_k && (21)\\
&= -t_i + o_i \sum_k t_k && (22)\\
&= o_i - t_i && (23)\\
& && (24)
\end{aligned}$$

the gradient for weights in the top layer is thus

$$\begin{aligned}
\frac{\partial E}{\partial w_{ji}} &= \sum_i \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial w_{ji}} && (25)\\
&= (o_i - t_i)o_j && (26)
\end{aligned}$$

and for units in the second lowest layer indexed by $j$,

$$\begin{aligned}
\frac{\partial E}{\partial x_j} &= \sum_i^{nclass} \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial o_j}\frac{\partial o_j}{\partial x_j} && (27)\\
&= \sum_i^{nclass} (o_i - t_i)(w_{ji})(o_j(1-o_j)) && (28)
\end{aligned}$$

Notice that this gradient has the same formula as for the summed cross entropy case, but it is different because the different activation function in last layer means $o$ takes on different values.

## 3 Algebraic trick for cross-entropy calculation

For a single output neuron with logistic activation, the cross-entropy error is given by

$$\begin{aligned}
E &= -\left(t\log o + (1-t)\log\left(1-o\right)\right) && (29)\\
&= -\left(t\log\left(\frac{o}{1-o}\right) + \log(1-o)\right) && (30)\\
&= -\left(t\log\left(\frac{\frac{1}{1+e^{-x}}}{1-\frac{1}{1+e^{-x}}}\right) + \log\left(1-\frac{1}{1+e^{-x}}\right)\right) && (31)\\
&= -\left(tx + \log\left(\frac{1}{1+e^{x}}\right)\right) && (32)\\
&= -tx + \log\left(1+e^{x}\right) && (33)
\end{aligned}$$