Notes on Backpropagation

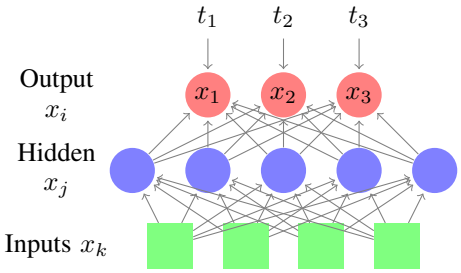
Peter Sadowski
Department of Computer Science
University of California Irvine
Irvine, CA 92697
peter.j.sadowski@uci.edu

Abstract

This document derives backpropagation for some common error functions and describes some other tricks.

1 Cross Entropy Error with Logistic Activation

For classification problems with two classes, the standard neural network architecture has a single output unit that provides a predicted probability of being in one class over another. The logistic activation function combined with the cross-entropy loss function gives us a nice probabilistic interpretation (as opposed to a sum-of-squared loss). We can generalize this to the case where we have multiple, independent, two-class outputs; we simply sum the loglikelihood of the independent targets.



The cross entropy error for a single example with $nout$ independent targets

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

where t is the target, x is the output, indexed by i. The activation function is the logistic function applied to the weighted sum of the neurons inputs,

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

$$s_i = \sum_{j=1} x_j w_{ji}. \tag{3}$$

The backprop algorithm is simply the chain rule applied to the neurons in each layer. The first step of the algorithm is to calculate the gradient of the training error with respect to the output layer weights, $\frac{\partial E}{\partial w_{ji}}$.

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

We can compute each factor as

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

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

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

$$\frac{\partial s_i}{\partial w_{ji}} = x_j \tag{8}$$

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

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

and

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

.

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 $\frac{\partial E}{\partial s_j}$ where j indexes the units in the second layer down.

$$\frac{\partial E}{\partial s_j} = \sum_{i=1}^{nout} \frac{\partial E}{\partial s_i}\frac{\partial s_i}{\partial x_j}\frac{\partial x_j}{\partial s_j} \tag{11}$$

$$= \sum_{i=1}^{nout} (x_i - t_i)(w_{ji})(x_j(1-x_j)) \tag{12}$$

$$\frac{\partial E}{\partial x_j} = \sum_{i=1} \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial S_i}\frac{\partial S_i}{\partial x_j} \tag{13}$$

$$= \sum_i \frac{\partial E}{\partial x_i} x_i(1-x_i)w_{ji} \tag{14}$$

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

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial s_j}\frac{\partial s_j}{\partial w_{kj}} \tag{15}$$

$$= \sum_{i=1}^{nout} (x_i - t_i)(w_{ji})(x_j(1-x_j))(x_k) \tag{16}$$

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

2 Classification with Softmax Transfer and Cross Entropy Error

For classification problems with more than 2 classes, the softmax output layer provides a way of assigning probabilities to each class. The cross-entropy error function is modified, but it turns out to have the same gradient as for the case of summed cross-entropy on logistic outputs. The softmax activation of the ith output unit is

$$x_i = \frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}} \tag{17}$$

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

$$E = -\sum_i^{nclass} t_i \log(x_i) \tag{18}$$

Thus, computing the gradient yields

$$\frac{\partial E}{\partial x_i} = -\frac{t_i}{x_i} \tag{19}$$

$$\frac{\partial x_i}{\partial s_k} = \begin{cases} \frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}} - (\frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}})^2 & i = k \\ -\frac{e^{s_i} e^{s_k}}{(\sum_c^{nclass} e^{s_c})^2} & i \neq k \end{cases} \tag{20}$$

$$= \begin{cases} x_i(1 - x_i) & i = k \\ -x_i x_k & i \neq k \end{cases} \tag{21}$$

$$\frac{\partial E}{\partial s_i} = \sum_k^{nclass} \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial s_i} \tag{22}$$

$$= \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial s_i} - \sum_{k \neq i} \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial s_i} \tag{23}$$

$$= -t_i(1 - x_i) + \sum_{k \neq i} t_k x_k \tag{24}$$

$$= -t_i + x_i \sum_k t_k \tag{25}$$

$$= x_i - t_i \tag{26}$$

$$\tag{27}$$

the gradient for weights in the top layer is thus

$$\frac{\partial E}{\partial w_{ji}} = \sum_i \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial w_{ji}} \tag{28}$$

$$= (x_i - t_i) x_j \tag{29}$$

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

$$\frac{\partial E}{\partial s_j} = \sum_i^{nclass} \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial x_j} \frac{\partial x_j}{\partial s_j} \tag{30}$$

$$= \sum_i^{nclass} (x_i - t_i)(w_{ji})(x_j(1 - x_j)) \tag{31}$$

Notice that this gradient has the same formula as for the summed cross entropy case, but it is different because the activation x takes on different values.

3 Algebraic trick for cross-entropy calculations

We can save some computation when doing cross-entropy error calculations, often an expensive part of training a neural network.

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

$$
\begin{aligned}
E &= -\left(t \log o + (1-t) \log (1-o)\right) && (32)\\
&= -\left(t \log \left(\frac{o}{1-o}\right) + \log(1-o)\right) && (33)\\
&= -\left(t \log \left(\frac{\frac{1}{1+e^{-x}}}{1-\frac{1}{1+e^{-x}}}\right) + \log\left(1-\frac{1}{1+e^{-x}}\right)\right) && (34)\\
&= -\left(tx + \log\left(\frac{1}{1+e^{x}}\right)\right) && (35)\\
&= -tx + \log\left(1+e^{x}\right) && (36)
\end{aligned}
$$

For a softmax output, the cross-entropy error is given by

$$
\begin{aligned}
E &= -\sum_i \left(t_i \log \frac{e^{x_i}}{\sum_j e^{x_j}}\right) && (37)\\
&= -\sum_i \left(t_i \left(x_i - \log \sum_j e^{x_j}\right)\right) && (38)\\
& && (39)\\
& && (40)
\end{aligned}
$$

Also note that in this softmax calculation, a constant can be added to each row of the output with no effect on the error function.