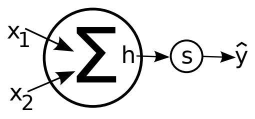
The Usual Derivation of Neural Net Backpropagation

Bart Massey

2011/2/15

This treatment mostly follows `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html`. I used the open source computer algebra package Maxima (`http://maxima.sourceforge.net`) to do and check some of the math.

Consider a neuron that looks like this:



We start with the axiomatic definitions. We will assume that all inputs to a neuron are in the interval $[-1 \ldots 1]$ and that the output will be in the same range. Our squashing function is the symmetric version of the sigmoid. Our first goal is to try to pick weights to minimize the mean squared error E between the desired output y and the actual output $\hat{y}$.

$$x_i, w_i \in [-1 \ldots 1] \tag{1}$$
$$h = \sum_i w_i x_i \tag{2}$$
$$s = \frac{e^h - 1}{e^h + 1} \tag{3}$$
$$\hat{y} = s \circ h \tag{4}$$
$$E(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2 \tag{5}$$

We now compute a couple of derivatives that will be interesting in error analysis. Note how the derivative of the sigmoid squashing function s is itself a simple function of that sigmoid. This is convenient in the implementation, since it means that we can use a single routine for both training and classification.

$$
\begin{aligned}
E_y &= \frac{\partial E}{\partial \hat{y}} && (6)\\
&= \hat{y} - y && (7)\\
E_{\hat{y}} &= \frac{\partial \hat{y}}{\partial h} && (8)\\
&= \frac{\partial s(h)}{\partial h} && (9)\\
&= \frac{\frac{\partial}{\partial h}(e^h - 1)\cdot(e^h+1) - (e^h-1)\cdot\frac{\partial}{\partial h}(e^h+1)}{(e^h+1)^2} && (10)\\
&= \frac{e^h(e^h+1) - (e^h-1)e^h}{(e^h+1)^2} && (11)\\
&= \frac{e^h(e^h+1-e^h+1)}{(e^h+1)^2} && (12)\\
&= \frac{2e^h}{(1+e^h)^2} && (13)\\
&= \frac{1}{2}(1 - s^2(h)) && (14)\\
&= \frac{1}{2}(1-\hat{y}^2) && (15)
\end{aligned}
$$

Next, we compute the error due to weight w_i and due to input x_i. We use the chain rule on the previously calculated derivatives.

$$
\begin{aligned}
E_{w_i} &= \frac{\partial E}{\partial w_i} && (16)\\
&= \frac{\partial E}{\partial \hat{y}}\cdot\frac{\partial \hat{y}}{\partial h}\cdot\frac{\partial h}{\partial w_i} && (17)\\
&= E_y \cdot E_{\hat{y}} \cdot \frac{\partial h}{\partial w_i} && (18)\\
&= (\hat{y}-y)\cdot\frac{1}{2}(1-\hat{y}^2)\cdot x_i && (19)\\
E_{x_i} &= \frac{\partial E}{\partial x_i} && (20)\\
&= \frac{\partial E}{\partial h}\frac{\partial h}{\partial x_i} && (21)\\
&= (\hat{y}-y)\cdot\frac{1}{2}(1-\hat{y}^2)\cdot w_i && (22)
\end{aligned}
$$

At this point, we can use E_{w_i} to adjust the weights, and then use E_{x_i} to find error terms for the prior; that is, for the outputs from the previous network layer.

Another legitimate choice would have been to have chosen to have x and y in the range $[0 \ldots 1]$. This is the more common choice in the literature. The math is the same, except that we now use a slightly different sigmoid for our squashing function, which of course has a slightly different derivative.

$$x, y \in [0 \ldots 1] \tag{23}$$

$$s(h) = \frac{1}{1 - e^{-h}} \tag{24}$$

$$\begin{aligned}
E_{\hat{y}} &= \frac{\partial \hat{y}}{\partial h} && (25)\\
&= \frac{\partial s(h)}{\partial h} && (26)\\
&= \frac{\partial}{\partial h}\left[\left(1 - e^{-h}\right)^{-1}\right] && (27)\\
&= -\left(1 - e^{-h}\right)^{-2} \cdot \frac{\partial}{\partial h}(1 - e^{-h}) && (28)\\
&= \frac{-e^{-h}}{\left(1 - e^{-h}\right)^{2}} && (29)\\
&= \hat{y}(1 - \hat{y}) && (30)
\end{aligned}$$

The adjustment to equations 19 and 22 is presumably obvious.