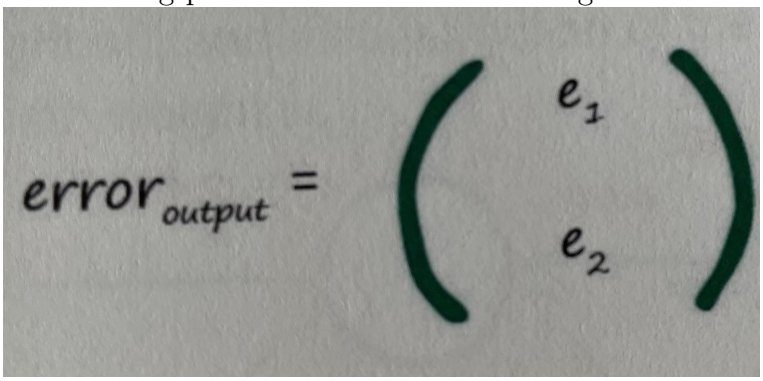
Make Your Own NN p80-p100

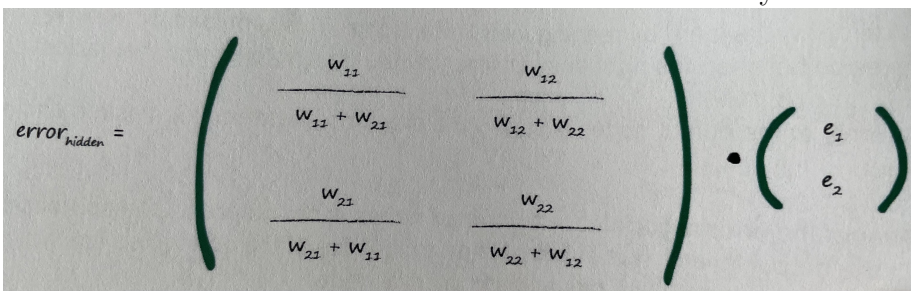
2021.2.2

Backpropagating Errors with Matrix Multiplication

The starting point is the errors that emerge from the NN at the final output layer.



Next we want to construct the matrix for the hidden layer errors.



You can see that the most important thing is the multiplication of the output errors e_n with the linked weights w_{ij}. The larger the weight, the more of the output error is carried back to the hidden layer. If we ignored that factor, we'd only lose the scaling of the errors being fed back. That is, $\frac{w_{11}}{w_{11}+w_{21}} * e_1$ would become the much simpler $w_{11} * e_1$.

The weight matrix we use here is actually the **transpose** matrix which we talked before. Then we have

$$error_{hidden} = W^T_{hidden\_output} \cdot error_{output}$$

Key Points

$\star$ Back-propagating the error can be expressed as a matrix multiplication.

$\star$ Both feeding signals forward and error back-propagation can be made efficient using matrix calculations.

How do we actually update weights: Gradient Descent

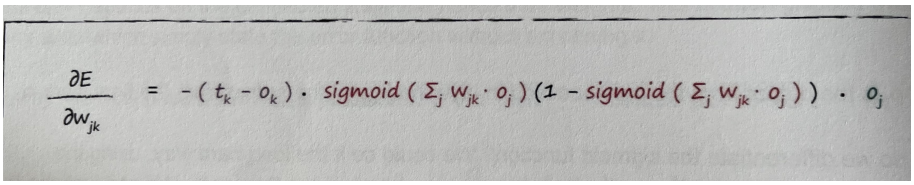
Easy for amath student, skip :)

E is error function, w_{jk} is linked weight.

$$\begin{aligned}
\frac{\partial E}{\partial w_{jk}} &= \frac{\partial}{\partial w_{jk}} \Sigma_n (t_n - o_n)^2 \\
\frac{\partial E}{\partial w_{jk}} &= \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{jk}} \\
&= -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{jk}} \\
&= -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{jk}} sigmoiod(\Sigma_j w_{jk} \cdot o_j)
\end{aligned}$$

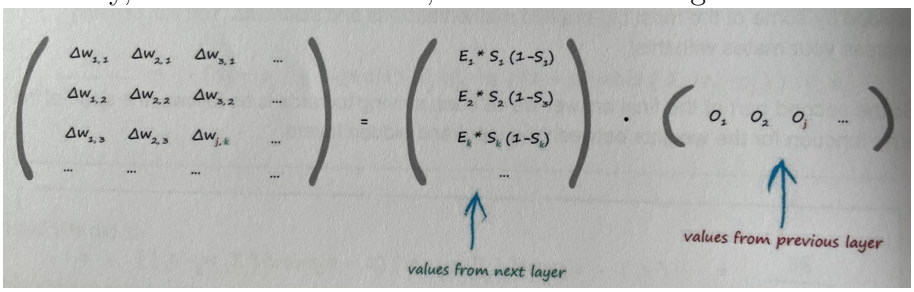
$$where \ \frac{\partial}{\partial x} sigmoid(x) = sigmoid(x) \cdot (1 - sigmoid(x))$$

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) \cdot sigmoid(\Sigma_j w_{jk} \cdot o_j)(1 - sigmoid(\Sigma_j w_{jk} \cdot o_j)) \cdot o_j$$



$$new \ w_{jk} = old \ w_{jk} - \alpha \cdot \frac{\partial E}{\partial w_{jk}}$$

Bascially, it is Newton Method, where α is learning rate.

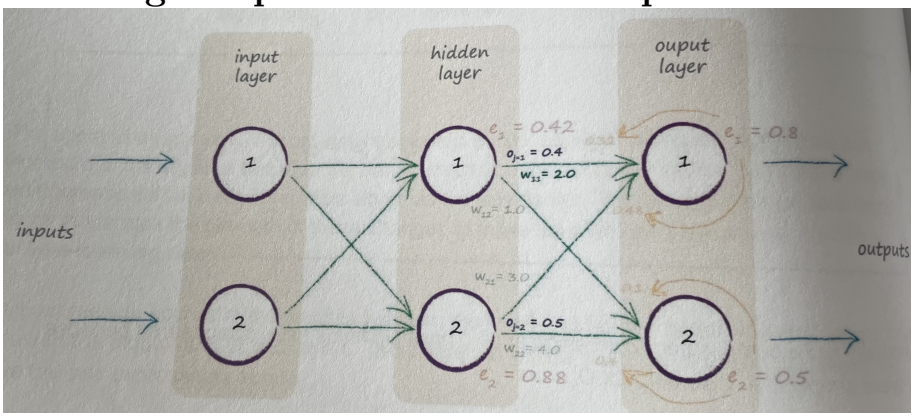


$$\Delta W_{jk} = \alpha \cdot E_k \cdot O_k(1 - O_k) \cdot O_j^T$$

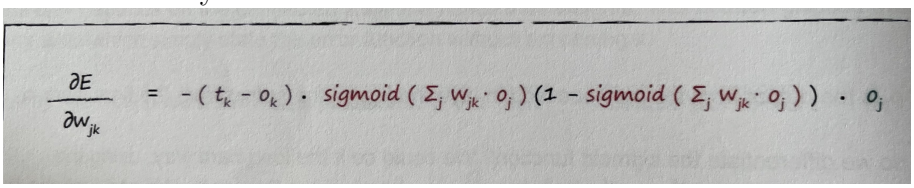
Key Points

⋆ NN's error is a function of the internal link weights

Weight Update Worked Example



Suppose we want to update the weight w_{11} between the hidden and output layers, which currently has the value 2.0



- The first bit $(t_k - o_k)$ is the error $e_1 = 0.8$, just as we saw before.
- The sum inside the sigmoid functions $\mathbf{\Sigma_j w_{jk} o_j}$ is $(2.0 * 0.4) + 3.0 * 0.5 = 2.3$
- The sigmoid $1/(1 + e^{-2.3})$ is then 0.909. That middle expression is then $0.909 * (1 - 0.909) = 0.083$
- $o_{j=1} = 0.4$

Multiplying all these three bits together and not forget minus sign at the start gives us -0.0265. If we have learning rate $\alpha = 0.1$, then $-(0.1 * -0.02650) = +0.00265$. The new w_{11} is $2.0 + 0.00265 = 2.00265$