Backpropgation is Just Steepest Descent with Automatic Differentiation

https://idontgetoutmuch.wordpress.com/2013/10/13/backpropogation-is-just-steepest-descent-with-automatic-differentiation-2/

How to calculate partial derivative

$$\frac{\partial E(\ldots, w, \ldots)}{\partial w} \approx \frac{(\ldots, w+\varepsilon, \ldots) - E(\ldots, w, \ldots)}{\varepsilon}$$

But this will get numerical errors when the ε gets smaller

Backpropgation comes to rescue

$$(g \circ f)'(a) = g'(f(a)).f'(a)$$

in alternative notation

$$\frac{\mathrm{d}(g \circ f)}{\mathrm{d}x}(a) = \frac{\mathrm{d}g}{\mathrm{d}y}(f(a))\frac{\mathrm{d}f}{\mathrm{d}x}(a)$$

where y = f(x). or

$$\frac{\mathrm{d}g}{\mathrm{d}x} = \frac{\mathrm{d}g}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}x}$$

example:

$$\frac{\mathrm{d}}{\mathrm{d}x}\sqrt{3\sin(x)} = \frac{\mathrm{d}}{\mathrm{d}x}(3\sin(x)) \cdot \frac{\mathrm{d}}{\mathrm{d}y}\sqrt{y} = 3\cos(x) \cdot \frac{1}{2\sqrt{y}} = \frac{3\cos(x)}{2\sqrt{3\sin(x)}}$$

Neural Network

$$
\begin{aligned}
a_i^{(1)} &= \sum_{j=0}^{N^{(1)}} w_{ij}^{(1)} x_j \\
z_i^{(1)} &= \tanh(a_i^{(1)}) \\
a_i^{(2)} &= \sum_{j=0}^{N^{(2)}} w_{ij}^{(2)} z_j^{(1)} \\
\ldots &= \ldots \\
a_i^{(L-1)} &= \sum_{j=0}^{N^{(L-1)}} w_{ij}^{(L-1)} z_j^{(L-2)} \\
z_j^{(L-1)} &= \tanh(a_j^{(L-1)}) \\
\hat{y}_i &= \sum_{j=0}^{N^{(L)}} w_{ij}^{(L)} z_j^{(L-1)}
\end{aligned}
$$

cost function

$$
E(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \|(\hat{\boldsymbol{y}} - \boldsymbol{y})\|^2
$$

gradient decent

$$
\Delta w_{ij} = \frac{\partial E}{\partial w_{ij}}
$$

chain rule

$$
\Delta w_{ij} = \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}
$$

since

$$a_j^{(l)} = \sum_{i=0}^{N} w_{ij}^{(l)} z_i^{(l-1)}$$

thus

$$\frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}} = \frac{\sum_{k=0}^{M} w_{kj}^{(l)} z_k^{(l-1)}}{\partial w_{ij}^{(l)}} = z_i^{(l-1)}$$

define

$$\delta_j^{(l)} \equiv \frac{\partial E}{\partial a_j^{(l)}}$$

thus

$$\Delta w_{ij}^{(l)} = \frac{\partial E}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$

output layer

$$\delta_j = \frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j} \left(\frac{1}{2} \sum_{i=0}^{M} (\hat{y}_i - y_i)^2 \right) = \hat{y}_j - y_j$$

a hidden layer using the chain rule

$$\delta_j^{(l-1)} = \frac{\partial E}{\partial a_j^{(l-1)}} = \sum_k \frac{\partial E}{\partial a_k^{(l)}} \frac{\partial a_k^{(l)}}{\partial a_j^{(l-1)}}$$

now

$$a_k^{(l)} = \sum_i w_{ki}^{(l)} z_i^{(l-1)} = \sum_i w_{ki}^{(l)} f(a_i^{(l-1)})$$

thus

$$\frac{\partial a_k^{(l)}}{\partial a_j^{(l-1)}} = \frac{\sum_i w_{ki}^{(l)} f(a_i^{(l-1)})}{\partial a_j^{(l-1)}} = w_{kj}^{(l)}\, f'(a_j^{(l-1)})$$

then

$$\delta_j^{(l-1)} = \sum_k \frac{\partial E}{\partial a_k^{(l)}} \frac{\partial a_k^{(l)}}{\partial a_j^{(l-1)}} = \sum_k \delta_k^{(l)} w_{kj}^{(l)}\, f'(a_j^{(l-1)}) = f'(a_j^{(l-1)}) \sum_k \delta_k^{(l)} w_{kj}^{(l)}$$

Summary:

1. forward calculation for a_j and z_j for each layer
2. evaluate output layer $\delta_j^{(L)}$ using $\delta_j = \hat{y}_j - y_j$
3. backward evaluate δ_j in each layer using $\delta_j^{(l-1)} = f'(a_j^{(l-1)}) \sum_k \delta_k^{(l)} w_{kj}^{(l)}$
4. use $\partial E/\partial w_{ij}^{(l)} = \delta_j^{(l)} z_i^{(l-1)}$ to calculate $\Delta w_{ij}^{(l)}$ in each layer
5. for activation function $\tanh$, f'(a) = $\tanh'(a) = 1 - \tanh^2(a)$
6. thus $w' = w - \gamma \nabla E(w)$