Complete Backpropagation

A Fully Worked Example — 2–2–2 Network, Sigmoid Units

One forward pass, one backward pass, one weight update

@SSRoy

1. The network and the given values

Two inputs i_1, i_2; two hidden neurons h_1, h_2; two outputs o_1, o_2. Every neuron uses the sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$. A single bias node feeds each layer.

Inputs	$i_1 = 0.05,\ i_2 = 0.10$	Targets	$t_1 = 0.01,\ t_2 = 0.99$
Input→hidden	w_1=0.15, w_2=0.20, w_3=0.25, w_4=0.30	Hidden bias	$b_1 = 0.35$
Hidden→output	w_5=0.40, w_6=0.45, w_7=0.50, w_8=0.55	Output bias	$b_2 = 0.60$

Key idea

Backpropagation is *one* idea applied repeatedly: the chain rule, run backward.

Forward: $i \to \text{net} \to \text{out} \to E$ $\quad\Longrightarrow\quad$ **Backward:** $E \to \text{out} \to \text{net} \to w$

2. Part A — Forward propagation

Each neuron computes net $= \sum(\text{weight} \times \text{input}) + \text{bias}$, then out $= \sigma(\text{net})$.

Hidden layer

$$\begin{aligned}
\text{net}_{h_1} &= i_1 w_1 + i_2 w_2 + b_1 = (0.05)(0.15) + (0.10)(0.20) + 0.35 = 0.377500,\\
h_1 &= \sigma(0.377500) = 0.593270;\\
\text{net}_{h_2} &= i_1 w_3 + i_2 w_4 + b_1 = (0.05)(0.25) + (0.10)(0.30) + 0.35 = 0.392500,\\
h_2 &= \sigma(0.392500) = 0.596884.
\end{aligned}$$

Output layer

$$\begin{aligned}
\text{net}_{o_1} &= h_1 w_5 + h_2 w_6 + b_2 = (0.593270)(0.40) + (0.596884)(0.45) + 0.60 = 1.105906,\\
o_1 &= \sigma(1.105906) = 0.751365;\\
\text{net}_{o_2} &= h_1 w_7 + h_2 w_8 + b_2 = (0.593270)(0.50) + (0.596884)(0.55) + 0.60 = 1.224921,\\
o_2 &= \sigma(1.224921) = 0.772928.
\end{aligned}$$

3. Part B — The error

Squared error, summed over both outputs: $E = \frac{1}{2}\sum(t-o)^2 = E_1 + E_2$.

$$\begin{aligned}
E_1 &= \tfrac{1}{2}(t_1 - o_1)^2 = \tfrac{1}{2}(0.01 - 0.751365)^2 = \tfrac{1}{2}(0.741365)^2 = 0.274811,\\
E_2 &= \tfrac{1}{2}(t_2 - o_2)^2 = \tfrac{1}{2}(0.99 - 0.772928)^2 = \tfrac{1}{2}(0.217072)^2 = 0.023560,\\
E &= 0.274811 + 0.023560 = \mathbf{0.298371}.
\end{aligned}$$

4. Part C — The core: gradient of one output weight w_5

The path from w_5 to the error is $w_5 \to \text{net}_{o_1} \to o_1 \to E$. So by the chain rule we need three factors:

$$\boxed{\frac{\partial E}{\partial w_5} = \underbrace{\frac{\partial E}{\partial o_1}}_{(1)} \cdot \underbrace{\frac{\partial o_1}{\partial \text{net}_{o_1}}}_{(2)} \cdot \underbrace{\frac{\partial \text{net}_{o_1}}{\partial w_5}}_{(3)}}$$

(1) Error vs. output. From $E_1 = \frac{1}{2}(o_1 - t_1)^2$:

$$\frac{\partial E}{\partial o_1} = o_1 - t_1 = 0.751365 - 0.01 = 0.741365.$$

(2) Output vs. net — the sigmoid derivative. We *derive* it, not memorise it. With $o_1 = (1 + e^{-\text{net}})^{-1}$,

$$\frac{do_1}{d\text{net}} = -(1 + e^{-\text{net}})^{-2} \cdot (-e^{-\text{net}}) = \frac{e^{-\text{net}}}{(1 + e^{-\text{net}})^2}.$$

Now notice $1 - o_1 = \dfrac{e^{-\text{net}}}{1 + e^{-\text{net}}}$, so $o_1(1 - o_1) = \dfrac{1}{1 + e^{-\text{net}}} \cdot \dfrac{e^{-\text{net}}}{1 + e^{-\text{net}}} = \dfrac{e^{-\text{net}}}{(1 + e^{-\text{net}})^2}$. Therefore

$$\boxed{\frac{\partial o_1}{\partial \text{net}_{o_1}} = o_1(1 - o_1)} = 0.751365(1 - 0.751365) = 0.751365 \times 0.248635 = 0.186816.$$

(3) Net vs. weight. Since $\text{net}_{o_1} = h_1 w_5 + h_2 w_6 + b_2$, only the w_5 term survives: $\dfrac{\partial \text{net}_{o_1}}{\partial w_5} = h_1 = 0.593270$.

Combine:

$$\frac{\partial E}{\partial w_5} = (0.741365)(0.186816)(0.593270) = 0.138499 \times 0.593270 = \mathbf{0.082167}.$$

Key idea

Group the first two factors and call it the neuron's **delta**:

$$\delta_o \equiv \frac{\partial E}{\partial \text{net}_o} = (o - t)\, o(1 - o), \qquad \Longrightarrow \qquad \frac{\partial E}{\partial w} = \delta_{(\text{receiving neuron})} \times (\text{input on that weight}).$$

Here $\delta_{o_1} = (0.741365)(0.186816) = 0.138499$ and $\dfrac{\partial E}{\partial w_5} = \delta_{o_1} h_1$.

5. Part D — All four output-layer gradients

$\delta_{o_1} = 0.138499$ (above). For o_2: $\dfrac{\partial E}{\partial o_2} = o_2 - t_2 = -0.217072$, $o_2(1 - o_2) = 0.175510$, so $\delta_{o_2} = (-0.217072)(0.175510) = -0.038098$.

$$\frac{\partial E}{\partial w_5} = \delta_{o_1} h_1 = (0.138499)(0.593270) = \quad 0.082167, \qquad \frac{\partial E}{\partial w_6} = \delta_{o_1} h_2 = (0.138499)(0.596884) = \quad 0.082668,$$
$$\frac{\partial E}{\partial w_7} = \delta_{o_2} h_1 = (-0.038098)(0.593270) = -0.022603, \qquad \frac{\partial E}{\partial w_8} = \delta_{o_2} h_2 = (-0.038098)(0.596884) = -0.022740.$$

6. Part E — Backpropagate into the hidden layer

Intuition

A hidden neuron feeds *both* outputs, so error reaches h_1 along two paths ($h_1 \to o_1 \to E$ and $h_1 \to o_2 \to E$). We must **sum** both contributions before multiplying by the local sigmoid slope.

$$\delta_h = \Big(\textstyle\sum_k \delta_{o_k}\, w_{h\to o_k}\Big)\, h(1-h).$$

For h_1 (weights w_5, w_7 carry error back from o_1, o_2):

$$\sum = \delta_{o_1} w_5 + \delta_{o_2} w_7 = (0.138499)(0.40) + (-0.038098)(0.50) = 0.055399 - 0.019049 = 0.036350,$$
$$h_1(1-h_1) = 0.593270 \times 0.406730 = \mathbf{0.241301},$$
$$\delta_{h_1} = (0.036350)(0.241301) = \mathbf{0.008771}.$$

For h_2 (weights w_6, w_8):

$$\sum = \delta_{o_1} w_6 + \delta_{o_2} w_8 = (0.138499)(0.45) + (-0.038098)(0.55) = 0.062324 - 0.020954 = 0.041370,$$
$$h_2(1-h_2) = 0.596884 \times 0.403116 = 0.240613,$$
$$\delta_{h_2} = (0.041370)(0.240613) = \mathbf{0.009954}.$$

7. Part F — The four hidden-layer gradients

Same rule, $\dfrac{\partial E}{\partial w} = \delta_{\text{receiving}} \times \text{input}$, with inputs i_1, i_2:

$$\frac{\partial E}{\partial w_1} = \delta_{h_1} i_1 = (0.008771)(0.05) = 0.0004386, \quad \frac{\partial E}{\partial w_2} = \delta_{h_1} i_2 = (0.008771)(0.10) = 0.0008771,$$
$$\frac{\partial E}{\partial w_3} = \delta_{h_2} i_1 = (0.009954)(0.05) = 0.0004977, \quad \frac{\partial E}{\partial w_4} = \delta_{h_2} i_2 = (0.009954)(0.10) = 0.0009954.$$

8. Part G — Update the weights (gradient descent)

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial E}{\partial w}, \qquad \eta = 0.5.$$

Intuition

We *subtract* because the gradient points uphill (toward larger error). A positive gradient means "increasing w raises E," so we lower w; a negative gradient (as for w_7, w_8) makes w *rise*. The step size is η.

Weight	Old	Gradient	Change ($-\eta g$)	**New**
w_1	0.150000	+0.0004386	−0.0002193	**0.1497807**
w_2	0.200000	+0.0008771	−0.0004386	**0.1995614**
w_3	0.250000	+0.0004977	−0.0002489	**0.2497512**
w_4	0.300000	+0.0009954	−0.0004977	**0.2995023**
w_5	0.400000	+0.0821670	−0.0410835	**0.3589165**
w_6	0.450000	+0.0826680	−0.0413340	**0.4086660**
w_7	0.500000	−0.0226025	+0.0113013	**0.5113013**
w_8	0.550000	−0.0227402	+0.0113701	**0.5613701**

Biases (single shared bias per layer, as drawn)

The one bias node of a layer feeds *every* neuron in that layer, so its gradient is the *sum* of those neurons' deltas:

$$\frac{\partial E}{\partial b_2} = \delta_{o_1} + \delta_{o_2} = 0.138499 - 0.038098 = 0.100401, \qquad b_2^{\text{new}} = 0.60 - 0.5(0.100401) = \mathbf{0.549800},$$
$$\frac{\partial E}{\partial b_1} = \delta_{h_1} + \delta_{h_2} = 0.008771 + 0.009954 = 0.018725, \qquad b_1^{\text{new}} = 0.35 - 0.5(0.018725) = \mathbf{0.340638}.$$

Intuition

If instead each neuron had its *own* bias (the more common textbook setup), you would *not* sum: $\partial E/\partial b_{o_1} = \delta_{o_1}$, $\partial E/\partial b_{o_2} = \delta_{o_2}$, and so on. Summing is correct *only* because this diagram shares one bias across the layer.

9. Verification — did the error actually drop?

Running one more forward pass with all updated weights and biases:

$$E_{\text{before}} = 0.298371 \longrightarrow E_{\text{after one step}} = 0.285751 \qquad (\Delta E = -0.012620).$$

The error decreased, confirming the update moved in the descent direction. Repeating forward→backward→update many times (epochs) drives $o_1 \to 0.01$ and $o_2 \to 0.99$.

10. The whole algorithm on one page

Key idea

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \text{out}} \cdot \frac{\partial \text{out}}{\partial \text{net}} \cdot \frac{\partial \text{net}}{\partial w}, \qquad \frac{\partial \text{out}}{\partial \text{net}} = \text{out}(1-\text{out}), \qquad \frac{\partial \text{net}}{\partial w} = \text{input}.$$

Hence $\dfrac{\partial E}{\partial w} = \delta \times \text{input}$, with $\delta_{\text{output}} = (o-t)\,o(1-o)$ and $\delta_{\text{hidden}} = \left(\sum_k \delta_k w_k\right) h(1-h)$.

Vectorised form (what libraries actually run). With activations a, targets t, layer output $\mathbf{o}$, and $\odot$ the elementwise product:

$$\boldsymbol{\delta}^L = (\mathbf{o}-\mathbf{t}) \odot \mathbf{o} \odot (1-\mathbf{o}), \quad \boldsymbol{\delta}^\ell = \left(W_{\ell+1}^\top \boldsymbol{\delta}^{\ell+1}\right) \odot a^\ell \odot (1-a^\ell), \quad \frac{\partial E}{\partial W_\ell} = \boldsymbol{\delta}^\ell \, (a^{\ell-1})^\top.$$

Common student mistake

- Using net instead of the sigmoid *output* in $o(1-o)$ — the slope uses the *activation*, not the pre-activation.
- Forgetting to *sum* both output paths when computing a hidden δ.
- Updating weights *during* the backward pass — compute *all* gradients from the *old* weights first, then update together.
- Sign slip on $(o-t)$ vs. $(t-o)$: with $E = \frac{1}{2}(t-o)^2$ the clean result is $\partial E/\partial o = o - t$.
- Dropping η: the update is not unique without a learning rate. Here $\eta = 0.5$ (chosen for this example; it is *not* given by the diagram).

Forward → Error → Backpropagation → Update. Repeat.