Back Propagation: work backwards from the output error,
distributing the blame for that error amongst all the weights
and adjusting those weights proportionately.

Training with a single layer

n_j represents the j^{th} neuron in the output (only) layer.

N is the number of output neurons

f is the activation function, $f(x) = 1 / (1 + e^{-x})$

w_{ij} is the weight on the connection from input i to output j

z^0_i is the value of input i
z^1_j is the value of output j

v_j is the total weighted input to n_j: $\Sigma(z_i \times w_{ij})$

t_j is the target, the correct value for z^1_j

E_j is the error in output i: $z^1_j - t_j$

Total error, by Mean Square Error, $E = \Sigma E_j^2 / N$

To adjust any particular weight w_{ij},
we need to know what effect a change in w_{ij} would have on E: $\delta E/\delta w_{ij}$

In this section, we are always talking about changes, not actual values:
The total error E due to w_{ij} depends on the output z^1_j
$$\delta E/\delta w_{ij} = \delta E/\delta z^1_j \times \delta z^1_j/\delta w_{ij}$$
The output z^1_j depends on its weighted inputs v_j
$$\delta E/\delta w_{ij} = \delta E/\delta z^1_j \times \delta z^1_j/\delta v_j \times \delta v_j/\delta w_{ij}$$

$\delta E/\delta z^1_j = z^1_j - t_j$ ignoring the constant 2

$\delta z^1_j/\delta v_j = \delta f(v_j)/\delta v_j = z^1_j \times (1 - z^1_j)$ $\delta f(x)/\delta x = f(x) \times (1 - f(x))$

$\delta v_j/\delta w_{ij} = z^0_i$

So the correction to w_{ij} should be proportional to $(z^1_j - t_j) \times z^1_j \times (1 - z^1_j) \times z^0_i$

w_{ij} becomes $w_{ij} - \eta \times (z^1_j - t_j) \times z^1_j \times (1 - z^1_j) \times z^0_i$

η is the learning rate

Do that for every single weight

And repeat that many thousands of times through the training set.

For a network with three inputs and two outputs, we would calculate

$\delta E/\delta w_{00} = \delta E/\delta z^1{}_0 \times \delta z^1{}_0/\delta v_0 \times \delta v_0/\delta w_{00}$

$\delta E/\delta w_{10} = \delta E/\delta z^1{}_0 \times \delta z^1{}_0/\delta v_0 \times \delta v_0/\delta w_{10}$

$\delta E/\delta w_{20} = \delta E/\delta z^1{}_0 \times \delta z^1{}_0/\delta v_0 \times \delta v_0/\delta w_{20}$

$\delta E/\delta w_{01} = \delta E/\delta z^1{}_1 \times \delta z^1{}_1/\delta v_1 \times \delta v_1/\delta w_{01}$

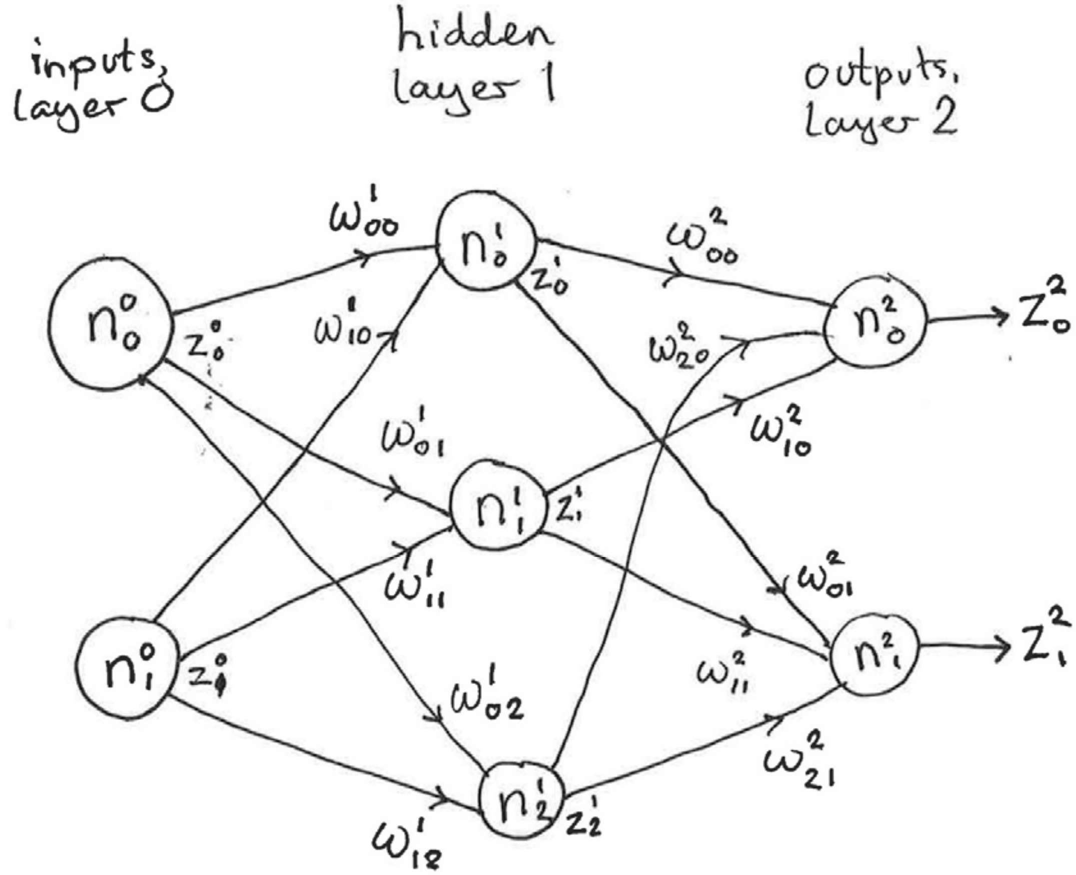
$\delta E/\delta w_{11} = \delta E/\delta z^1{}_1 \times \delta z^1{}_1/\delta v_1 \times \delta v_1/\delta w_{11}$

$\delta E/\delta w_{21} = \delta E/\delta z^1{}_1 \times \delta z^1{}_1/\delta v_1 \times \delta v_1/\delta w_{21}$

Note that the first two terms remain the same for each neuron: precompute.

Training with multiple layers

- Each layer creates a different representation of the inputs
- Inner layers often discover meaningful features of the input
- The output layer usually has one neuron for each feature you want to detect or for each possible classification of the input.



the output from n^{ℓ}_i is called z^{ℓ}_i

so z^0_i is just the i^{th} input value

A slight change in the symbols is required:

the z^0_i are the input values

f is the activation function, $f(x) = 1 / (1 + e^{-x})$

$z^2_0 = f(z^1_0 \times w^2_{00} + z^1_1 \times w^2_{10} + z^1_2 \times w^2_{20})$

in general, where ℓ represents any layer, and L represents the last layer:

$z^{\ell}_j = f(\Sigma(z^{\ell-1}_i \times w^{\ell}_{ij}))$ for $\ell > 1$

And as before, we split this into two parts

$v^{\ell}_j = \Sigma(z^{\ell-1}_i \times w^{\ell}_{ij})$
$z^{\ell}_j = f(v^{\ell}_j)$

t_i is the target value for output i, what v^L_i should be

E_i is the error in output i

$E_0 = z^L_0 - t_0$

$E_1 = z^L_1 - t_1$

Total error, by Mean Square Error, $E = (E_0 + E_1)^2 / 2$

Now for weights in the hidden layer, w^1_{ij}, using $\delta E/w^1_{01}$ as an example

w^1_{01} affects E by two different paths,
from n^0_0 through w^1_{01} to n^1_1 then through w^2_{10} to n^2_0, and
from n^0_0 through w^1_{01} to n^1_1 then through w^2_{11} to n^2_1, and

For the first path we get
$\varepsilon_0 = \delta E/\delta z^2_0 \times \delta z^2_0/\delta v^2_0 \times \delta v^2_0/\delta z^1_1 \times \delta z^1_1/\delta v^1_1 \times \delta v^1_1/\delta w^1_{01}$
and for the second path we get
$\varepsilon_1 = \delta E/\delta z^2_1 \times \delta z^2_1/\delta v^2_1 \times \delta v^2_1/\delta z^1_1 \times \delta z^1_1/\delta v^1_1 \times \delta v^1_1/\delta w^1_{01}$

As before:

$\delta E/\delta z^L_j = z^L_j - t_j$

$\delta z^{\ell}_j/\delta v^{\ell}_j = z^{\ell}_j \times (1 - z^{\ell}_j)$

And a differential we haven't seen before: $\delta v^{\ell}_j/\delta z^{\ell-1}_i = w^{\ell}_{ij}$

$\delta v^{\ell}_j/\delta w^{\ell}_{ij} = z^{\ell-1}_i$

ε_0 and ε_1 must be added together to get $\delta E/\delta w^1_{01}$

$w^1_{01} := w^1_{01} - \eta \times (\varepsilon_0 + \varepsilon_1)$