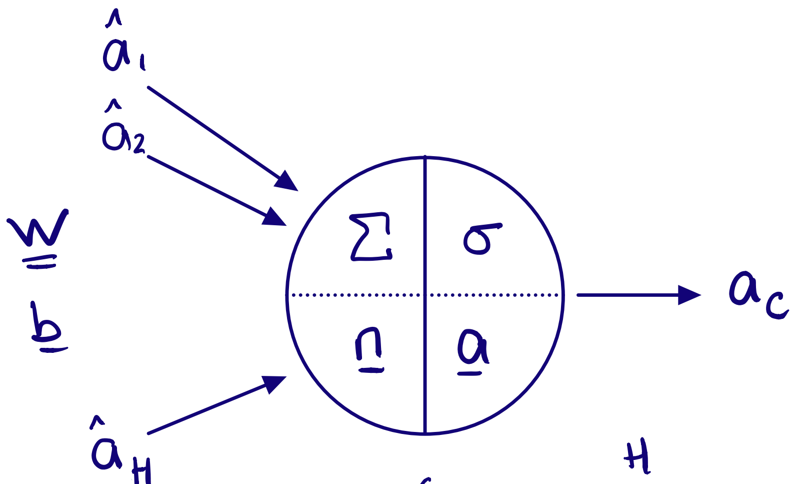
Backpropagation and Learning

We consider a neuron and associated operations on that neuron as follows:



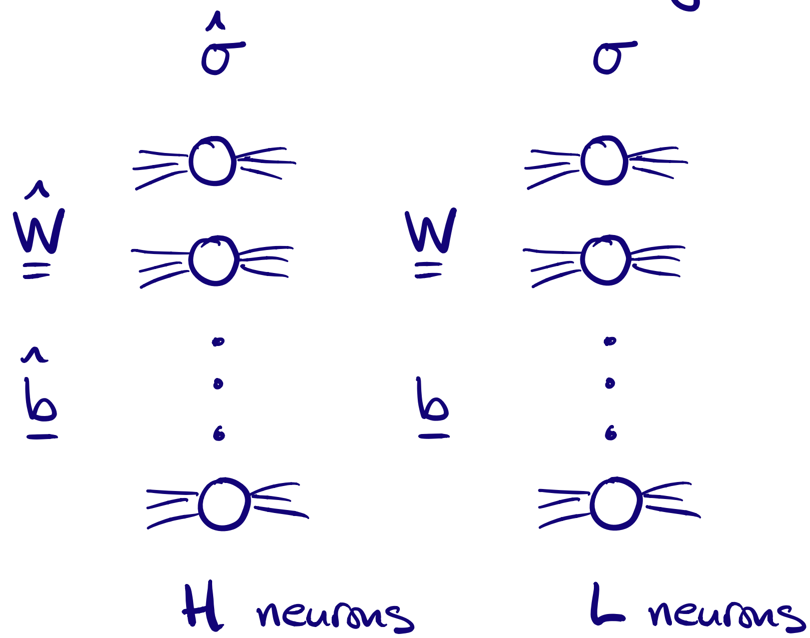
$$\underline{n} = \underline{\underline{W}}^T \underline{\hat{a}} + \underline{b} \iff \begin{cases} n_c = \sum_{l=1}^{H} W_{lc} \hat{a}_l + b_c \\ \text{for } c = 1, 2, \dots, L \end{cases}$$

$\underline{n}$ is the weighted sum of inputs $\underline{\hat{a}}$ with added bias $\underline{b}$

$$\underline{a} = \sigma(\underline{n}) \iff a_c = \sigma(n_c) \text{ for } c = 1, 2, \dots, L$$

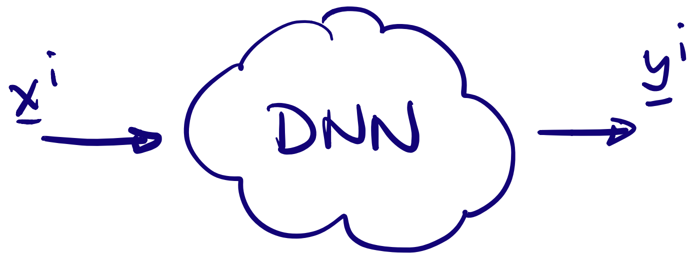
$\underline{a}$ is the activated output.

Consider the general set-up between any two consecutive layers of a DNN



Quantities with hats, $\hat{\sigma}$, $\hat{\underline{\underline{W}}}$, $\hat{\underline{b}}$, refer to the left hand layer.

For the whole network we have a training set of N_t items $(\underline{x}^1, \underline{t}^1), (\underline{x}^2, \underline{t}^2), \ldots, (\underline{x}^{N_t}, \underline{t}^{N_t})$, and each input $\underline{x}^i$ produces an output $\underline{y}^i$



error

$$\underline{e}^i = \underline{t}^i - \underline{y}^i$$

We want to train the network by

updating the weights and biases so as to minimize a cost, or error, or performance index. We can think about

TOTAL SQUARED ERROR:

$$\mathcal{E}_{TSE} = \sum_{i=1}^{N_t} \ell(\underline{t}^i, \underline{y}^i) \quad \text{for } \ell(\underline{t}, \underline{y}) = \|\underline{t} - \underline{y}\|_2^2$$

MEAN SQUARED ERROR:

$$\mathcal{E}_{MSE} = \frac{1}{N_t} \sum_{i=1}^{N_t} \ell(\underline{t}^i, \underline{y}^i) \quad \text{for } \ell(\underline{t}, \underline{y}) \text{ as above}$$

Other choices are possible. For example, we can use a TOTAL CROSS-ENTROPY error, $\mathcal{E}_{TCE}$, which is

$$\mathcal{E}_{TCE} = \sum_{i=1}^{N_t} \ell(\underline{t}^i, \underline{y}^i) \quad \text{for } \underline{t}^i, \underline{y}^i \in \mathbb{R}^d \; \begin{pmatrix}\text{column}\\ \text{vectors}\end{pmatrix}$$

for

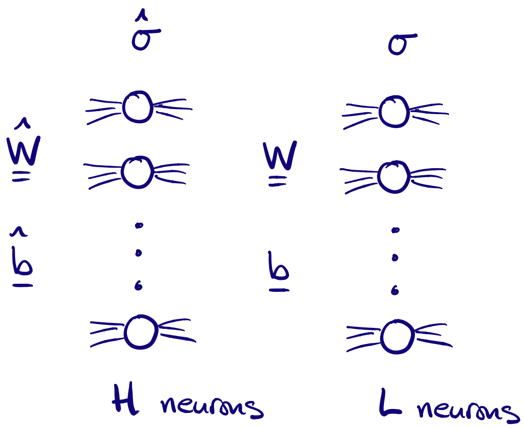
$$\ell(\underline{t}, \underline{y}) = -\sum_{j=1}^{d} t_j \ln(y_j)$$

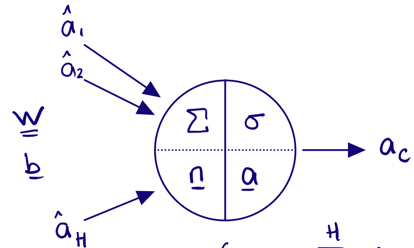
The CROSS ENTROPY loss (cost, error, ...) function.

We'll have more to say about CROSS ENTROPY later. For now we just assume we have a performance index, $\mathcal{E}$, where a function $\mathcal{F}$ of $\underline{t}$ and $\underline{y}$ is summed over the training set.

In SGD we do not take the gradient of $\mathcal{E}$ to implement gradient descent updates on the weights and biases, but just each $\nabla\mathcal{F}(\underline{t}^i, \underline{y}^i)$ (at least in the simplest case) in turn, perhaps chosen at random.

So, we return to the general set-up for a consecutive pair of layers from earlier, recall all the equations in play, and differentiate $\mathcal{F}$ with respect to the weights and biases.





$$\underline{n} = \underline{\underline{W}}^T \hat{\underline{a}} + \underline{b} \iff \begin{cases} n_c = \sum_{l=1}^{H} W_{lc} \hat{a}_l + b_c \\ \text{for } c = 1, 2, \ldots, L \end{cases}$$

In addition to the equations above we also have

$$\hat{\underline{a}} = \hat{\sigma}(\hat{\underline{n}}) \in \mathbb{R}^H, \qquad \underline{\underline{W}} \in \mathbb{R}^{H \times L}$$

$$\underline{a} = \sigma(\underline{n}) \in \mathbb{R}^L, \qquad \underline{b} \in \mathbb{R}^L$$

Given $\not{E} = \not{E}(\underline{t}, \underline{y})$ consider these

$$\frac{\partial \not{E}}{\partial W_{rc}} = \frac{\partial \not{E}}{\partial n_c} \frac{\partial n_c}{\partial W_{rc}}$$

$$\frac{\partial \not{E}}{\partial b_c} = \frac{\partial \not{E}}{\partial n_c} \frac{\partial n_c}{\partial b_c}$$

These will be used in Gradient Descent

$$W_{rc} \leftarrow W_{rc} - \alpha \frac{\partial \not{E}}{\partial W_{rc}}$$

$$b_c \leftarrow b_c - \alpha \frac{\partial \not{E}}{\partial b_c}$$

Now,

$$n_c = \sum_{l=1}^{H} w_{lc}\,\hat{a}_l + b_c$$

and hence,

$$\frac{\partial n_c}{\partial w_{rc}} = \hat{a}_r \quad \text{and} \quad \frac{\partial n_c}{\partial b_c} = 1$$

It follows that

$$\frac{\partial \mathcal{F}}{\partial w_{rc}} = \hat{a}_r S_c \quad \text{and} \quad \frac{\partial \mathcal{F}}{\partial b_c} = S_c$$

where we define

$$S_c = \frac{\partial \mathcal{F}}{\partial n_c} \implies \underline{S} = \begin{bmatrix} \partial \mathcal{F}/\partial n_1 \\ \partial \mathcal{F}/\partial n_2 \\ \vdots \\ \partial \mathcal{F}/\partial n_L \end{bmatrix} \in \mathbb{R}^L$$

These $\underline{S}$ vectors play a crucial role.

With these formulae for individual components we can get

$$\frac{\partial f}{\partial \underline{b}} = \begin{bmatrix} \partial f/\partial b_1 \\ \partial f/\partial b_2 \\ \vdots \\ \partial f/\partial b_L \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_L \end{bmatrix} = \underline{s} \in \mathbb{R}^L$$

and

$$\frac{\partial f}{\partial \underline{\underline{W}}} = \begin{bmatrix} \partial f/\partial W_{11} & \partial f/\partial W_{12} & \cdots & \partial f/\partial W_{1L} \\ \partial f/\partial W_{21} & \partial f/\partial W_{22} & \cdots & \partial f/\partial_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ \partial f/\partial W_{H1} & \partial f/\partial W_{H2} & \cdots & \partial f/\partial W_{HL} \end{bmatrix}$$

$$= \begin{bmatrix} \hat{a}_1 s_1 & \hat{a}_1 s_2 & \cdots & \hat{a}_1 s_L \\ \hat{a}_2 s_1 & \hat{a}_2 s_2 & \cdots & \hat{a}_2 s_L \\ \vdots & \vdots & \ddots & \vdots \\ \hat{a}_H s_1 & \hat{a}_H s_2 & \cdots & \hat{a}_H s_L \end{bmatrix}$$

$$= \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_H \end{bmatrix} [s_1\ s_2 \ldots s_L] = \underline{\hat{a}}\, \underline{s}^T \in \mathbb{R}^{H \times L}$$

Next we introduce the 'Jacobian' matrix,

$$\frac{\partial \underline{n}}{\partial \hat{\underline{n}}} = \begin{bmatrix} \partial n_1/\partial \hat{n}_1 & \partial n_1/\partial \hat{n}_2 & \cdots & \partial n_1/\partial \hat{n}_H \\ \partial n_2/\partial \hat{n}_1 & \partial n_2/\partial \hat{n}_2 & \cdots & \partial n_2/\partial \hat{n}_H \\ \vdots & \vdots & \ddots & \vdots \\ \partial n_L/\partial \hat{n}_1 & \partial n_L/\partial \hat{n}_2 & \cdots & \partial n_L/\partial \hat{n}_H \end{bmatrix} \in \mathbb{R}^{L \times H}.$$

Using $\hat{\underline{a}} = \hat{\sigma}(\hat{\underline{n}})$ we calculate

$$\frac{\partial n_c}{\partial \hat{n}_r} = \frac{\partial}{\partial \hat{n}_r}\left[\sum_{\ell=1}^{H} W_{\ell c}\hat{a}_\ell + b_c\right]$$

$$= W_{rc}\frac{\partial \hat{a}_r}{\partial \hat{n}_r}$$

$$= W_{rc}\,\hat{\sigma}'(\hat{n}_r)$$

We used $\hat{a}_r = \hat{\sigma}(\hat{n}_r)$. $\hat{\sigma}'(n_r)$ denotes differentiation

Now, $\frac{\partial n_c}{\partial \hat{n}_r} = W_{rc}\,\hat{\sigma}'(\hat{n}_r)$ is the element of $\frac{\partial \underline{n}}{\partial \hat{\underline{n}}}$ in row c, column r, So...

$$\frac{\partial \underline{n}}{\partial \underline{\hat{n}}} = \begin{bmatrix} W_{11}\hat{\sigma}'(\hat{n}_1) & W_{21}\hat{\sigma}'(\hat{n}_2) & \cdots & W_{H1}\hat{\sigma}'(\hat{n}_H) \\ W_{12}\hat{\sigma}'(\hat{n}_1) & W_{22}\hat{\sigma}'(\hat{n}_2) & \cdots & W_{H2}\hat{\sigma}'(\hat{n}_H) \\ \vdots & \vdots & \ddots & \vdots \\ W_{1L}\hat{\sigma}'(\hat{n}_1) & W_{2L}\hat{\sigma}'(\hat{n}_2) & \cdots & W_{HL}\hat{\sigma}'(\hat{n}_H) \end{bmatrix}$$

$$= \begin{bmatrix} W_{11} & W_{21} & \cdots & W_{H1} \\ W_{12} & W_{22} & \cdots & W_{H2} \\ \vdots & \vdots & \ddots & \vdots \\ W_{1L} & W_{2L} & \cdots & W_{HL} \end{bmatrix} \begin{bmatrix} \hat{\sigma}'(\hat{n}_1) & & \text{zeros} \\ & \hat{\sigma}'(\hat{n}_2) & \\ & & \ddots & \\ \text{zeros} & & & \hat{\sigma}'(\hat{n}_H) \end{bmatrix}$$

$$\Rightarrow \frac{\partial \underline{n}}{\partial \underline{\hat{n}}} = \underline{\underline{W}}^T \underline{\underline{\hat{A}}} \quad \text{for } \underline{\underline{\hat{A}}} = \text{diag}\left(\hat{\sigma}'(\underline{\hat{n}})\right) \in \mathbb{R}^{H \times H}$$

Now, remember that we wrote

$$\underline{S} = \frac{\partial f}{\partial \underline{n}} \in \mathbb{R}^L.$$

Similarly, we can write

$$\underline{\hat{S}} = \frac{\partial f}{\partial \underline{\hat{n}}} \in \mathbb{R}^H$$

So,

$$\hat{s}_r = \frac{\partial \mathcal{F}}{\partial \hat{n}_r} = \sum_{l=1}^{L} \frac{\partial \mathcal{F}}{\partial n_l} \cdot \frac{\partial n_l}{\partial \hat{n}_r}$$

$$= \sum_{l=1}^{L} \frac{\partial n_l}{\partial \hat{n}_r} s_l$$

$$\Rightarrow \quad \underline{\hat{s}} = \left(\frac{\partial \underline{n}}{\partial \underline{\hat{n}}}\right)^T \underline{s} = \left(\underline{\underline{W}}^T \underline{\underline{\hat{A}}}\right)^T \underline{s}$$

Recall, in general, $(\underline{\underline{P}}\,\underline{\underline{Q}})^T = \underline{\underline{Q}}^T \underline{\underline{P}}^T$, so...

$$\underline{\hat{s}} = \underline{\underline{\hat{A}}}^T \underline{\underline{W}}\, \underline{s} \Rightarrow \boxed{\underline{\hat{s}} = \underline{\underline{\hat{A}}}\, \underline{\underline{W}}\, \underline{s}}$$

RECURSION

because $\underline{\underline{\hat{A}}}^T = \underline{\underline{\hat{A}}}$.

This recursion is the key: if we know $\underline{s}$ at a layer, then we can find $\underline{\hat{s}}$ at the preceding layer. BACK PROP

So: we need $\underline{s}$ at the final (output) layer...

... And that is available - because we know everything we need at that layer.

By definition: $\quad \underline{s} = \dfrac{\partial f}{\partial \underline{n}}$

with

$$\underline{n} = \underline{W}^T \hat{\underline{a}} + \underline{b} \quad \text{and} \quad \underline{y} = \sigma(\underline{n})$$

Let's consider ε_{TSE}, and then

$$f = f(\underline{t}, \underline{y}) = \|\underline{t} - \underline{y}\|_2^2 = \sum_{j=1}^{d} (t_j - y_j)^2$$

Hence,

$$\frac{\partial f}{\partial n_c} = \frac{\partial}{\partial n_c} \sum_{j=1}^{d} (t_j - y_j)^2 = -2 \sum_{j=1}^{d} e_j \frac{\partial y_j}{\partial n_c}$$

for $e_j = t_j - y_j$. However, $\underline{y} = \sigma(\underline{n})$ and so

$$\frac{\partial y_j}{\partial n_c} = \begin{cases} 0 & \text{if } j \neq c \\ \sigma'(n_c) & \text{if } j = c \end{cases}$$

This means that

$$\frac{\partial \phi}{\partial n_c} = -2 e_c \sigma'(n_c).$$

Hence,

$$\frac{\partial \phi}{\partial \underline{n}} = -2 \begin{bmatrix} \sigma'(n_1)e_1 \\ \sigma'(n_2)e_2 \\ \vdots \\ \sigma'(n_L)e_L \end{bmatrix} \in \mathbb{R}^L$$

$$= -2 \begin{pmatrix} \sigma'(n_1) & & \text{zeros} \\ & \sigma'(n_2) & \\ & & \ddots \\ \text{zeros} & & \sigma'(n_L) \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_L \end{pmatrix}$$

$$= -2 \underline{\underline{A}}\, \underline{e}$$

A result we've seen many times before.
So, at the final two layers...

$$\frac{\partial \phi}{\partial \underline{\underline{W}}} = \underline{\hat{a}}\, \underline{S}^T \text{ and } \frac{\partial \phi}{\partial \underline{b}} = \underline{S} \text{ for } \underline{S} = -2 \underline{\underline{A}}\, \underline{e}.$$

Summary

- We can forward prop an input $\underline{x}$ through the DNN to get the output $\underline{y}$
- We can form the error $\underline{e} = \underline{t} - \underline{y}$
- We can form $\underline{S} = -2\underline{\underline{A}}\,\underline{e}$ and then update with

$$\underline{\underline{W}} \leftarrow \underline{\underline{W}} - \alpha \frac{\partial \mathcal{E}}{\partial \underline{\underline{W}}} \quad \& \quad \underline{b} \leftarrow \underline{b} - \alpha \frac{\partial \mathcal{E}}{\partial \underline{b}}$$

where
$$\frac{\partial \mathcal{E}}{\partial \underline{\underline{W}}} = \underline{\hat{a}}^T \underline{S} \quad \& \quad \frac{\partial \mathcal{E}}{\partial \underline{b}} = \underline{S}$$

- We backpropagate through to the previous layer:

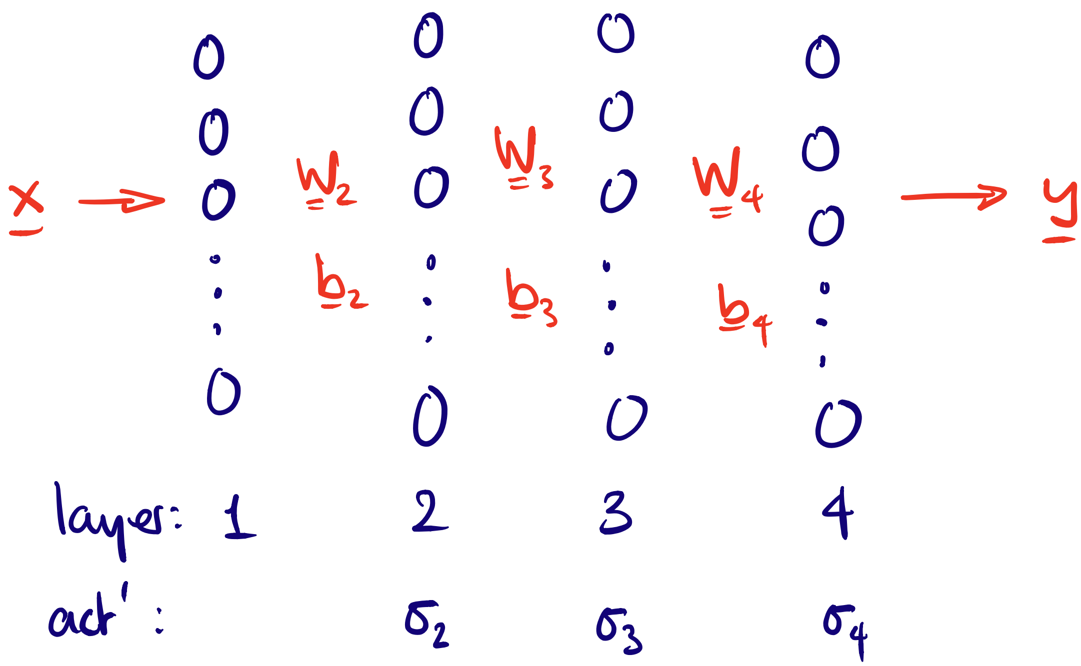
$$\underline{\hat{S}} = \underline{\underline{\hat{A}}}\,\underline{\underline{W}}\,\underline{S}$$

and then
$$\underline{\underline{\hat{W}}} \leftarrow \underline{\underline{\hat{W}}} - \alpha \frac{\partial \mathcal{E}}{\partial \underline{\underline{\hat{W}}}} \quad \& \quad \underline{\hat{b}} \leftarrow \underline{\hat{b}} - \alpha \frac{\partial \mathcal{E}}{\partial \underline{\hat{b}}}$$

- Repeat all the way backwards through the DNN updating weights and biases as we go.

Concrete Example

A 4 layer network



$\underline{e}_4 = \underline{t} - \underline{y}$ for a training pair $(\underline{x}, \underline{t})$

$\underline{S}_4 = -2 \underline{\underline{A}}_4 \underline{e}_4$

$\underline{S}_3 = \underline{\underline{A}}_3 \underline{\underline{W}}_4 \underline{S}_4$

$\underline{S}_2 = \underline{\underline{A}}_2 \underline{\underline{W}}_3 \underline{S}_3$

backpropagating

from output back to input

CALCULUS-BASED