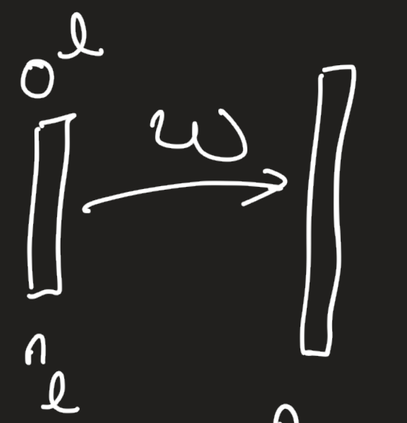
Backpropagation

Determining $\nabla_{o^l} f \in \mathbb{R}^{n_l}$



$$\left[\nabla_{o^l} f\right]_i = \frac{\partial f}{\partial o_i^l} = \sum_{j=1}^{n_{l+1}} \frac{\partial f}{\partial o_j^{l+1}} \frac{\partial o_j^{l+1}}{\partial o_i^l}$$

recall $o^{l+1} = \sigma(a^{l+1}) = \sigma(w^{l+1} o^l + b^{l+1})$

where $w^{l+1} = \begin{bmatrix} w_1^{l+1} \\ \vdots \\ w_{n_{l+1}}^{l+1} \end{bmatrix} \in \mathbb{R}^{n_{l+1} \times n_l}$

so

$$o_j^{l+1} = \sigma(\langle w_j^{l+1}, o^l \rangle + b_j^{l+1}) \quad (a_j^{l+1})$$

$$= \sigma\left(\sum_{k=1}^{n_l} w_{jk}^{l+1} o_k^l + b_j^{l+1} \right)$$

$$\sum_{k \neq i}^{n_l} w_{jk}^{l+1} o_k^l + w_{ji}^{l+1} o_i^l + b_j^{l+1}$$

$$\frac{\partial o_j^{l+1}}{\partial o_i^l} = \sigma'(a_j^{l+1}) \frac{\partial a_j^{l+1}}{\partial o_i^l} = \underbrace{(w^{l+1})^T_{ij}}_{w^{l+1}_{ji}} \sigma'(a_j^{l+1})$$

$$\left[\nabla_{o^l} f\right]_i = \sum_{j=1}^{n_{l+1}} \frac{\partial f}{\partial o_j^{l+1}} \frac{\partial o_j^{l+1}}{\partial o_i^l}$$

$$= \sum_{j=1}^{n_{l+1}} \frac{\partial f}{\partial o_j^{l+1}} \left[(w^{l+1})^T_{ij} \sigma'(a_j^{l+1})\right]$$

$$= \sum_{j=1}^{n_{l+1}} (w^{l+1})^T_{ij} \left[\underbrace{\frac{\partial f}{\partial o_j^{l+1}}}_{\left[\nabla_{o^{l+1}} f\right]_j} \cdot \sigma'(a_j^{l+1})\right]$$

$$= \left[\left(W^{\ell+1}\right)^T \nabla_{o^{\ell+1}} f \odot \sigma'(a^{\ell+1}) \right]_i$$

$\Rightarrow$

(1) $$\nabla_{o^\ell} f = \underbrace{\left(W^{\ell+1}\right)^T}_{\in \mathbb{R}^{n_\ell \times n_{\ell+1}}} \underbrace{\left[\nabla_{o^{\ell+1}} f \odot \sigma'(a^{\ell+1}) \right]}_{\in \mathbb{R}^{n_{\ell+1}}}$$

this is in $\mathbb{R}^{n_\ell}$

Remember if $x \in \mathbb{R}^d$ & $y \in \mathbb{R}^d$ then
$x \odot y \in \mathbb{R}^d$ is called the Hadamard / entrywise /
Schur product and satisfies

$$(x \odot y)_i = x_i y_i \quad \text{for } i = 1, \dots, d$$

Recall "$\frac{\partial f}{\partial w^l} = \frac{\partial f}{\partial o^l}\frac{\partial o^l}{\partial w^l}$"

We will drop the superscript l for convenience since we will be working with quantities on layer l

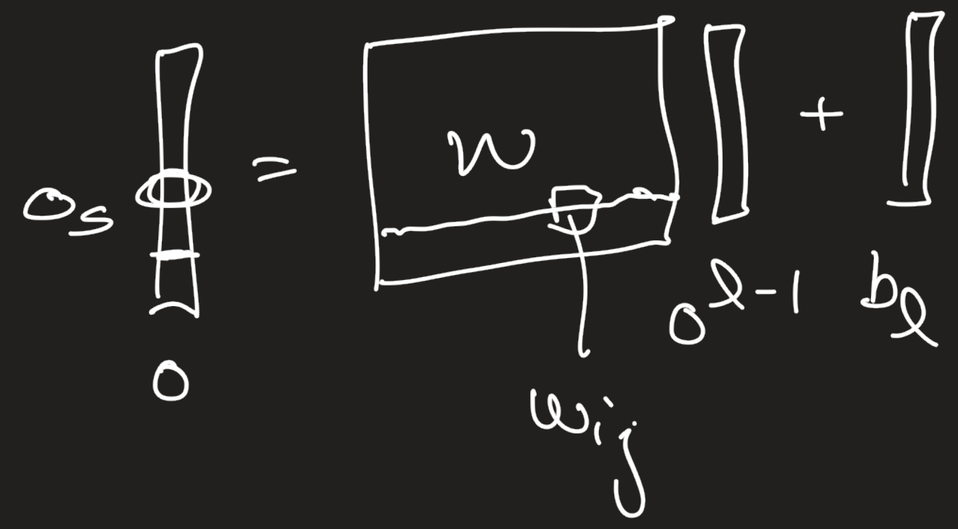
$$\left[\nabla_w f\right]_{ij} = \frac{\partial f}{\partial w_{ij}} = \sum_{s=1}^{n} \frac{\partial f}{\partial o_s}\frac{\partial o_s}{\partial w_{ij}}$$

recall $o = \sigma(a) = \sigma(wo^{l-1} + b)$

where $w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$

so $o_s = \sigma(a_s) = \sigma(\langle w_s, o^{l-1}\rangle + b_s)$

$$= \sum_{j=1}^{\hat{l}-1} w_{sj}\, o_j^{l-1} + b_s$$



$$\frac{\partial o_s}{\partial w_{ij}} = \begin{cases} 0 & i \neq s \\ \sigma'(a_i)\frac{\partial a_i}{\partial w_{ij}} & i = s \end{cases}$$

recall $a_i = \langle w_i, o^{l-1} \rangle + b_i$

$$= \sum_{j=1}^{n^{l-1}} w_{ij} o_j^{l-1} + b_i \quad \Rightarrow \frac{\partial a_i}{\partial w_{ij}} = o_j^{l-1}$$

$$\left[\nabla_w f\right]_{ij} = \sum_{s=1}^{n} \frac{\partial f}{\partial o_s} \frac{\partial o_s}{\partial w_{ij}}$$

$\frac{\partial o_s}{\partial w_{ij}}$: 0 unless $i = s$

$$= \frac{\partial f}{\partial o_i} \frac{\partial o_i}{\partial w_{ij}} = \frac{\partial f}{\partial o_i} \sigma'(a_i)\, o_j^{\ell-1}$$

$$= \sigma'(a_i) \left[\nabla_o f\right]_i \left(o^{\ell-1}\right)_j$$

$$= \sigma'(a_i) \left[\nabla_o f \left(o^{\ell-1}\right)^T\right]_{ij}$$

add back the l superscripts

(2) $$\nabla_{w^l} f = \underbrace{\text{diag}\left(\sigma'(a^l)\right)}_{n_l \times n_l} \underbrace{\nabla_{o^l} f \cdot (o^{l-1})^T}_{n_l \times n_{l-1} \text{ matrix}}$$

$n_l \times n_{l-1}$ matrix as necessary

similarly

(3) $$\nabla_{b^l} f = \underbrace{\nabla_{o^l} f \odot \sigma'(a^l)}_{n^l \text{ dim vector as necessary}}$$

This gives the backprop algorithm for computing the required gradient

Backprop Alg (assuming $R=0$)

Inputs: $\underbrace{a^1, \dots, a^L, o^0, \dots, o^L}$, $W_1, \dots, W_L, b_1, \dots, b_L$

assume computed in a forward pass

Outputs: $\{\nabla_{W^\ell} f\}$, $\{\nabla_{b^\ell} f\}$ for $\ell = 1, \dots, L$

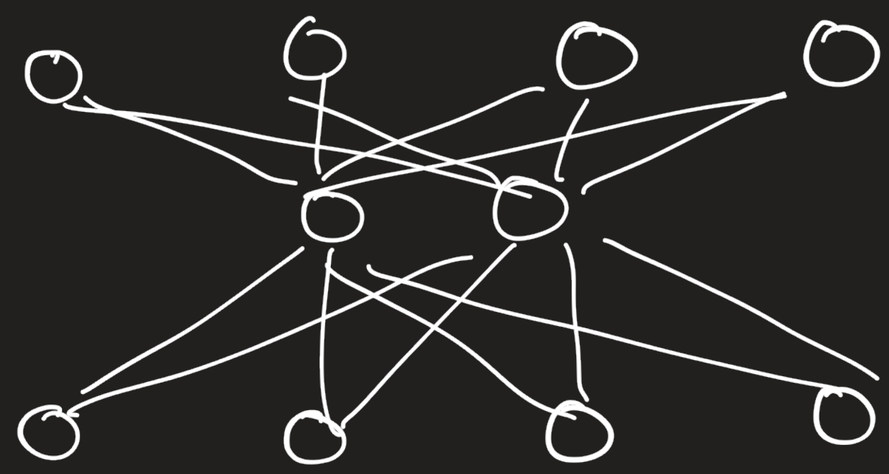
- Compute $\text{temp}\nabla \leftarrow \nabla_{o^L} f$
- Compute $\nabla_{W^L} f$ and $\nabla_{b^L} f$ using eqns (2) & (3) and temp∇

- For $l = L-1, \ldots, 1$
  - Compute temp$\nabla \leftarrow \nabla_{O^l} f$ using eqn (1) and temp∇ (which contains $\nabla_{O^{l+1}} f$)
  - Compute $\nabla_{W^l} f$ and $\nabla_{b^l} f$ using eqns (2) and (3) and temp∇

Autoencoders

- Generalizations of PCA & SVD (nonlinear!)

Typically these are used to create useful machine learning features in an unsupervised way



Goal: learn a (compressed, usually) representation that does a good job of reconstructing the input

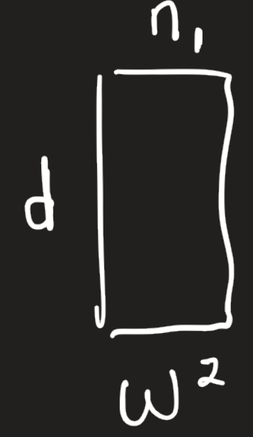
Ex ℓ_2-loss $\quad \ell(v, v) = \|v - v\|_2^2$

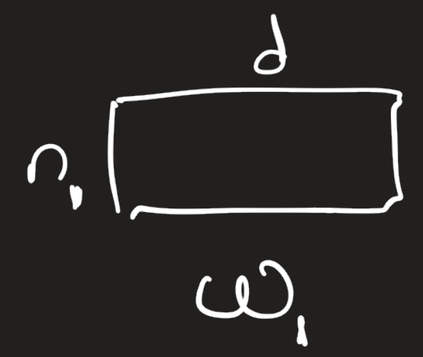
and $\sigma(x) = x$

$$\hat{y}(x) = o_2 = \sigma(w^2 o^1 + b^2)$$
$$= \sigma(w^2 \sigma(w^1 o^0 + b^1) + b^2)$$
$$= \sigma(w^2 \sigma(w^1 x + b^1) + b^2)$$
$$= w^2 w^1 x + w^2 b^1 + b^2$$

fix $b_1 = b_2 = 0$

$$= w^2 w^1 x \quad \text{where}$$





so the ERM is

$$f(W_1, W_2) = \frac{1}{n}\sum_{i=1}^{n} \| \hat{g}(x_i) - x_i \|_2^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \| W_2 W_1 x_i - x_i \|_2^2$$

$$= \frac{1}{n} \| W_2 W_1 X - X \|_F^2$$

where $X = [x_1 | \cdots | x_n]$

so fitting this autoencoder learns matrices W_1, W_2 $\in \mathbb{R}^{d \times n_1}, \mathbb{R}^{n_1 \times d}$ to minimize

$\| W_2 W_1 X - X \|_F^2$, this is the <u>SVD</u>

Deep autoencoders: