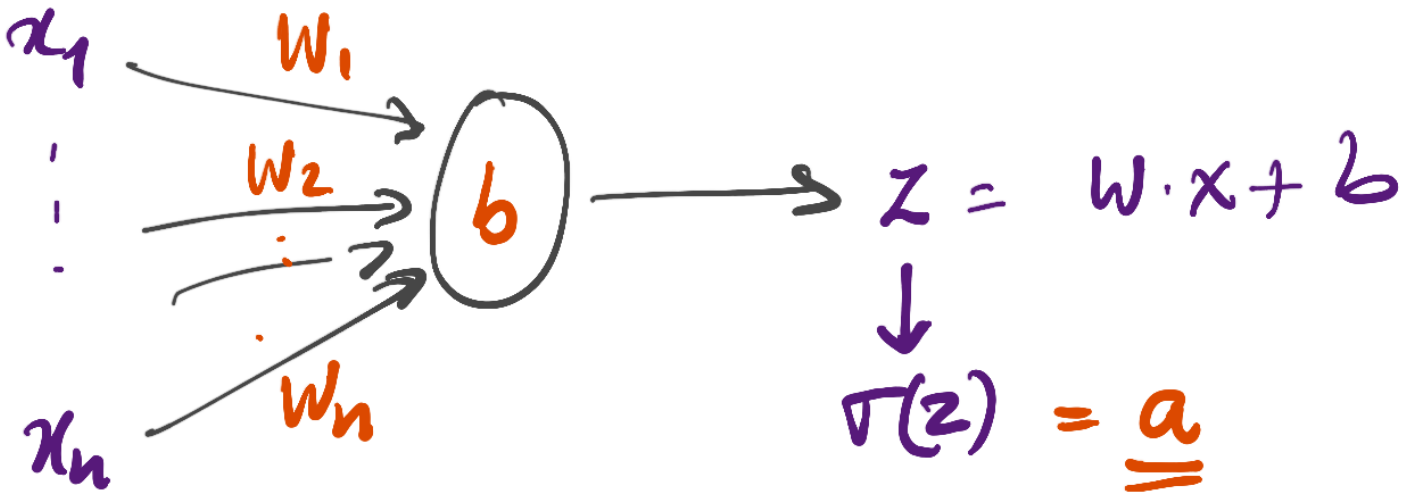
DMML 14 Mar 2019

Neural networks

Acyclic networks of perceptrons

How do we compute weights & biases of individual nodes?



Gradient descent

- Cost function C
- But for each w_j & b in the network,

  compute $\frac{\partial C}{\partial w_j}$, $\frac{\partial C}{\partial b}$

Typical cost function Training data $(\bar{x}_i, y_i)$

$$\frac{1}{2n} \sum_{i=1}^{n} \| y_i - a(\bar{x}_i) \|^2$$

Assumptions about C

1. Depends on $a(\bar{x})$ (and nothing else)

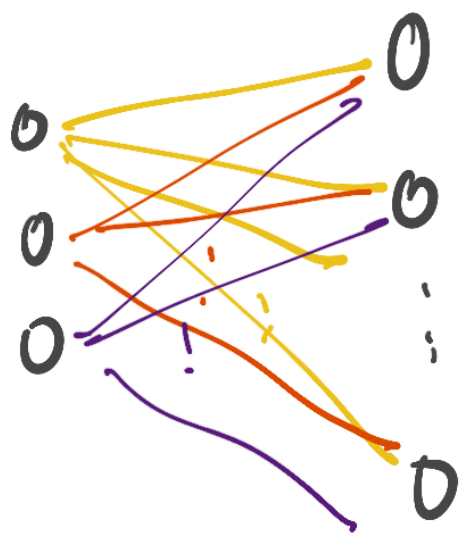
2. Average error

   - Act on each input separately

   - Many parameters to learn — overall training set is large

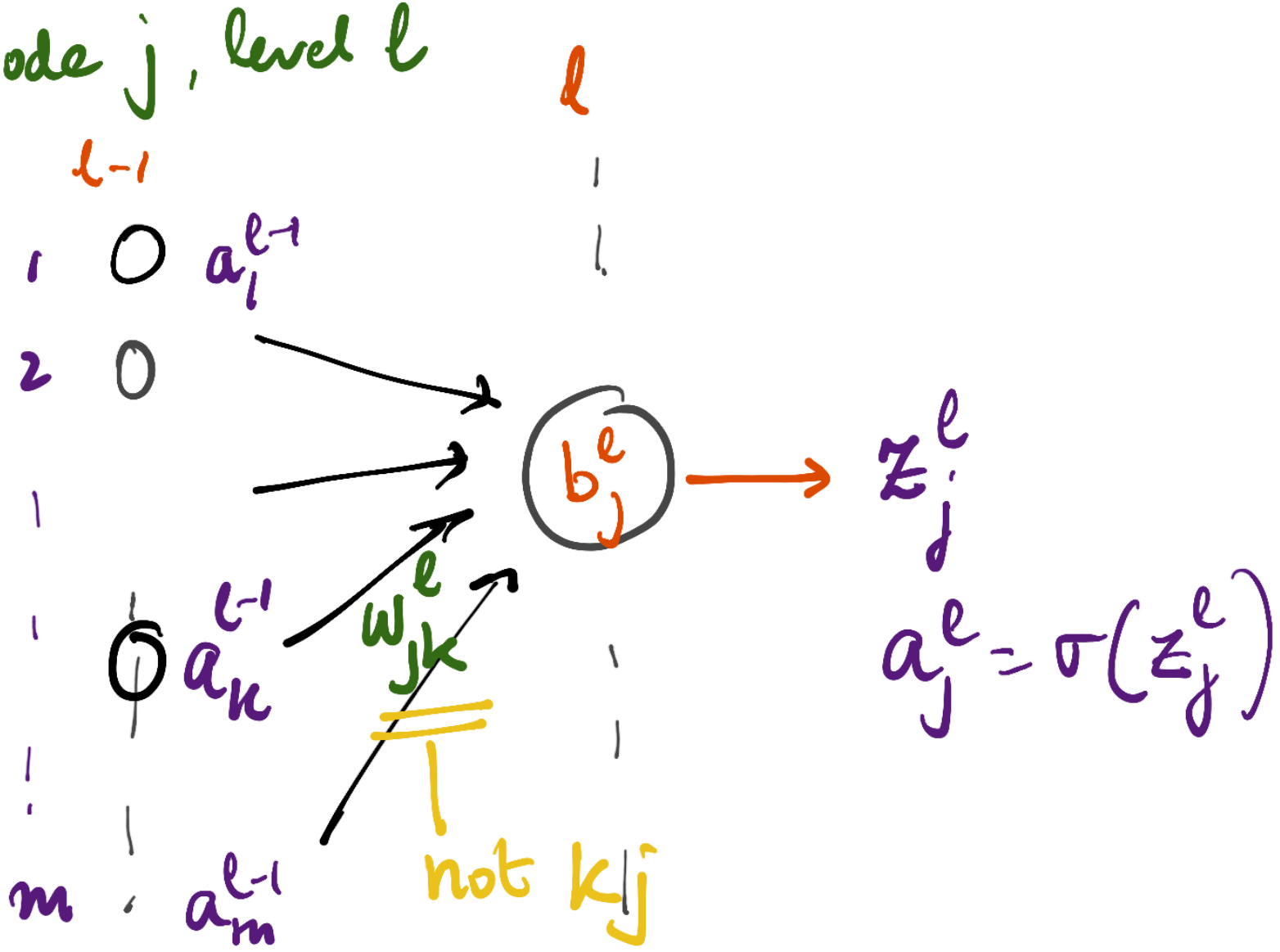
Structural assumption about network

- Assume layers $1, 2, \ldots, L$
  - L is the output layer, single node
- Each layer i is fully connected to layer $i+1$



Notation

Level l, has m nodes



$$z_j^l = w_{j1}^l \cdot a_1^{l-1} + w_{j2}^l \cdot a_2^{l-1} + \cdots w_{jm}^l a_m^{l-1} + b_j^l$$

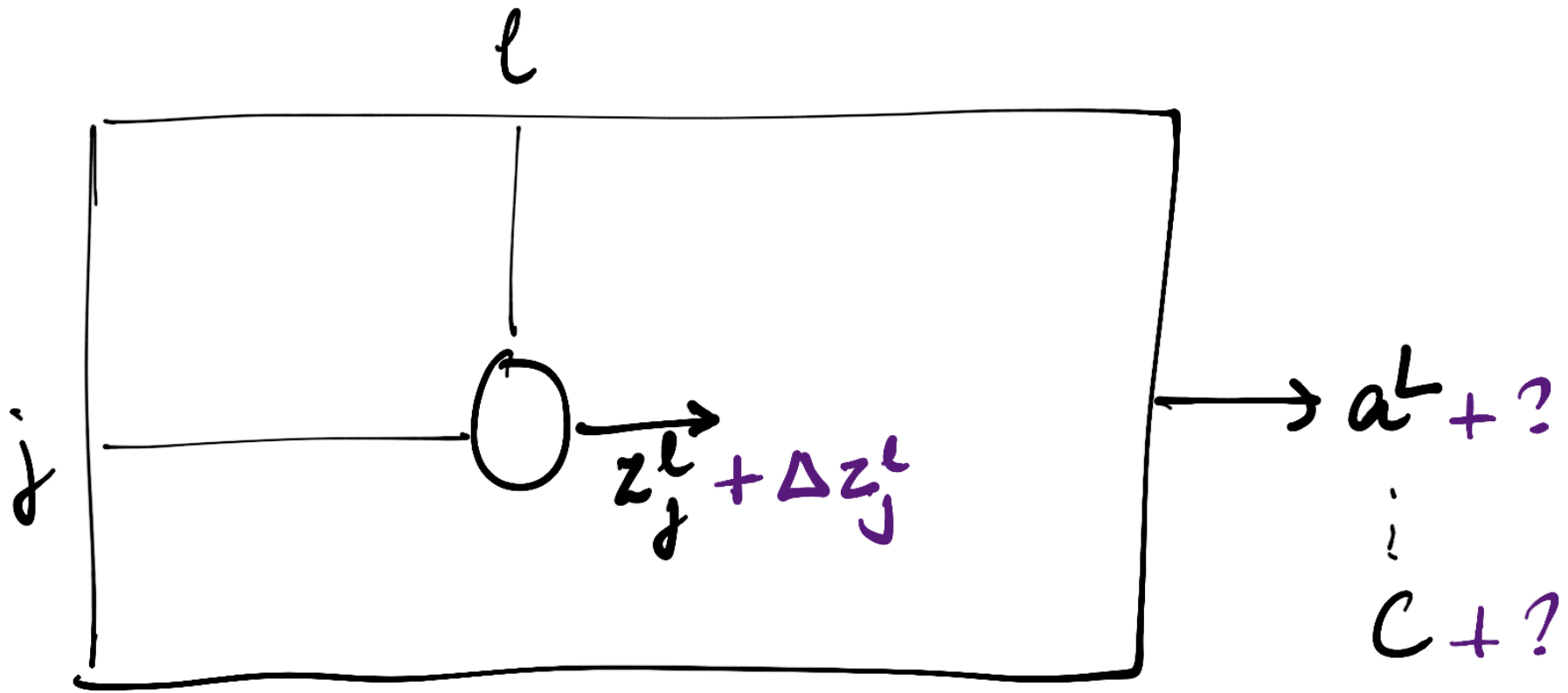
$$= \bar{w}_j^l \cdot \bar{a}^{l-1} + b_j^l$$

Compute

$$\frac{\partial C}{\partial w_{jk}^l}, \quad \frac{\partial C}{\partial b_j^l}$$

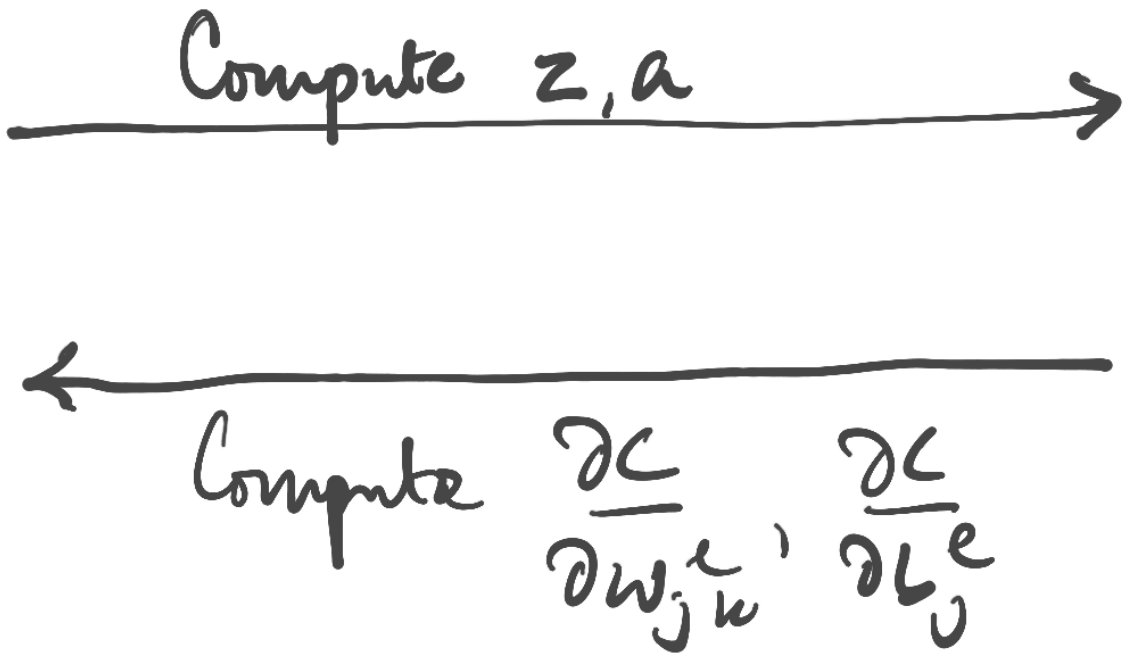
For layer L , only 1 node

$$\frac{\partial C}{\partial w_{11}^L}, \frac{\partial C}{\partial w_{12}^L}, \cdots, \frac{\partial C}{\partial w_{1m}^L}, \frac{\partial C}{\partial b^L}$$



$\frac{\partial C}{\partial z_j^l}$ – artificial quantity δ_j^l

Overall strategy



Recall δ_j^l is $\frac{\partial C}{\partial z_j^l}$

① $\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \boxed{\frac{\partial a_j^L}{\partial z_j^L}}$

$a_j^L = \sigma(z_j^L)$

$\sigma'(z_j^L)$

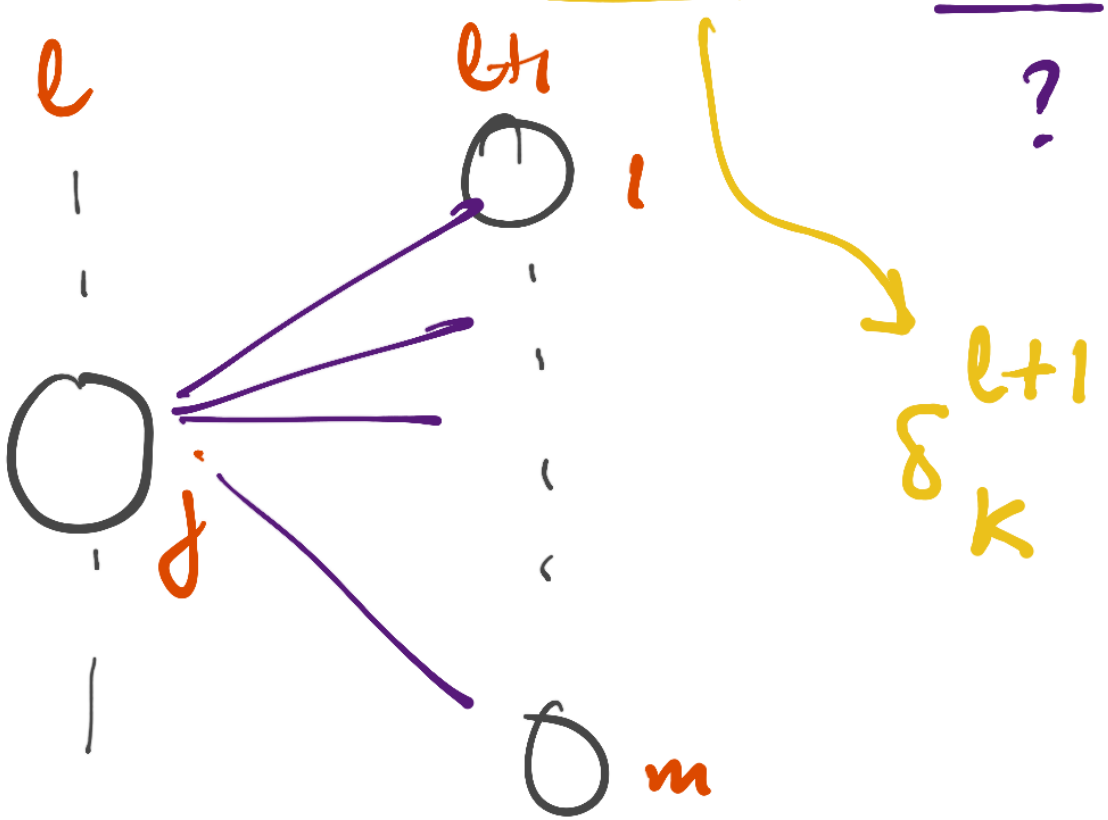
$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$

$a_j^L - y$

$C = \frac{1}{2n} \sum \|y - a\|^2 \rightarrow (y - a_j^L)(-1)$

② Compute δ^l given δ^{l+1}

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_{k=1}^{m} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$



$$\frac{\partial z_k^{l+1}}{\partial z_j^l}$$

level l

$$z_k^{l+1} = \sum_{i=1}^{n} w_{ki}^{l+1} a_i^l + b_k^{l+1}$$

$$a_i^l = \sigma(z_i^l)$$

For $i \neq j$, terms vanish

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \cdot \sigma'(z_j^l)$$

Backward pass computes δ_j^l for all j, l

(3) $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

$\hookrightarrow \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l}$

$\frac{\partial C}{\partial z_j^l} = \delta_j^l$ $\quad \frac{\partial z_j^l}{\partial b_j^l} = 1$?

$z_j^l = \bar{w}_j^l \cdot \bar{a}^{l-1} + b_j^l$

$b_j^l \downarrow$

1

(4) $\frac{\partial C}{\partial w_{jk}^{l}} = a_k^{l-1} \cdot \delta_j^{l}$

L $\frac{\partial C}{\partial z_j^{l}} \cdot \frac{\partial z_j^{l}}{\partial w_{jk}^{l}}$

$= \delta_j^{l}$

$\Rightarrow z_j^{l} = \sum w_{ji} a_i^{l-1} + b_j^{l}$

only $i=k$ term matters

$\downarrow$

a_k^{l-1}

Backpropagation

Forward pass – Compute z_j^l, a_j^l

Backward pass – Compute δ_j^l and

hence $\frac{\partial C}{\partial b_j^l}$, $\frac{\partial C}{\partial w_{jk}^l}$

Backpropagation dates from mid 1980's

Still need raw computing power – large training set, multiple iterations

Use a standard package

- Batch size for stochastic gradient descent update
- No. of iterations