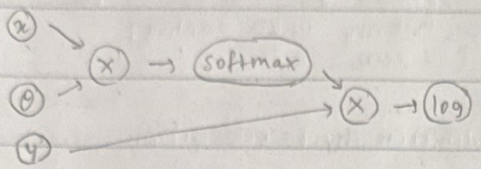
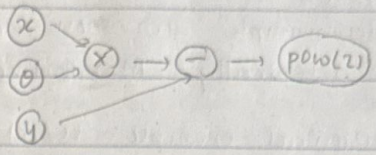
Computation Graph - analogous to a "MATH DAG". NN is one large comp. graph.

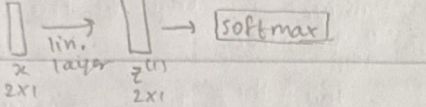
↳ log. Reg.



↳ lin Reg.

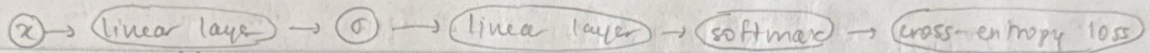


data (x & y) ; parameters (θ) - usually used once.



which layer

How to represent learned features - represent as logistic regression output $(\sigma(w^{(i)}x))$



how much to "fire" based on incoming signals

We can use non-linear functions (activation functions) in slw layers (lin. als. lasagna)

Chain rule: $\frac{d}{dx_i} f(g(x)) = \sum_{j=1}^{m} \frac{dy_j}{dx_i}\frac{dz}{dy_j}$ sum over all dimensions of y $= \frac{dy}{dx_i}\frac{dz}{dy}$ ($1\times m$ · $m\times 1$)

$\frac{d}{dx} f(g(x)) = \frac{dy}{dx}\frac{dz}{dy}$ ($n\times m$ · $m\times 1$) $\left(\frac{dy}{dx}\right)_{ij} = \frac{dy_j}{dx_i}$.

Compute Jacobians right-to-left: saves compute time by factor of n (backprop).

Backpropagation algorithm: ("classic" version)

→ forward pass: calculate each $a^{(i)}$ and $z^{(i)}$

→ backward pass:

initialize $\delta = \frac{dL}{dz^{(n)}}$

for each f w/ input x_f & params θ_f from end to start:

$\frac{dL}{d\theta_f} \leftarrow \frac{df}{d\theta_f}\delta$,

$\delta \leftarrow \frac{df}{dx_f}\delta$

NN arch details:

1. How many layers?
2. How big are the layers?
3. What type of activ. func.? Popular choice: ReLU (efficient).

$z^{(i+1)} = W^{(i)}a^{(i)} + b^{(i)}$ ← additional params in each lin. layer.

$\frac{dz}{dW}\delta = \sum_i \frac{dz_i}{dW}\delta_i$ $\quad \frac{dz_i}{dW_{jk}} = \begin{cases} 0 & \text{if } j \neq i \\ a_k & \text{otherwise} \end{cases} \Rightarrow \delta a^T \cdot \frac{dz}{ds} = \delta \quad \frac{dz}{da}\delta = W^T\delta$