MLP BACK PROPAGATION

EE 541 – UNIT 5

DR. BRANDON FRANZKE

Spring 2023

MLP FORWARD/BACK-PROP TOPICS

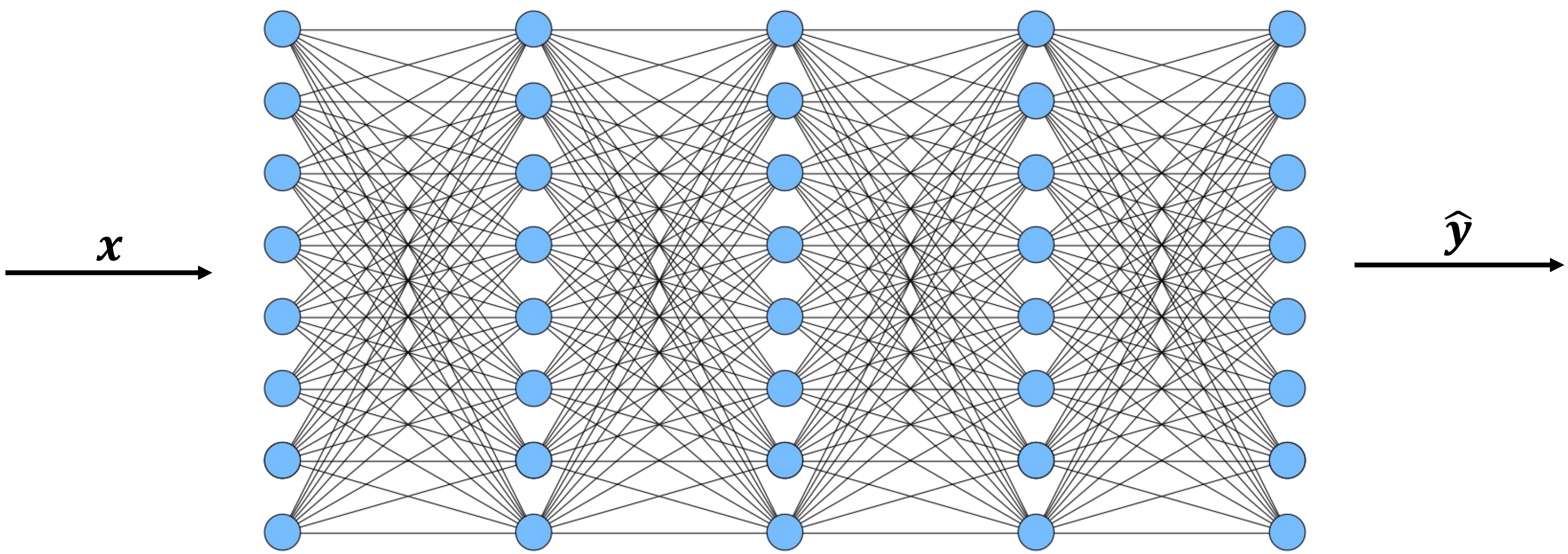
- **MLP feedforward (inference) equations**
- **Back-propagation mathematics**
  - from scalar to the matrix-vector case
  - General rules/conventions for matrix-vector calculus
- Universal Approximation
- Variations on back-prop and MLP training
  - activations and their derivatives
  - regularizers, optimizers (e.g., ADAM) and momentum
  - dropout, batch normalization

MULTILAYER PERCEPTRON NETWORKS (MLPS)

Forward propagation (inference and training)

$\boldsymbol{a}^{(l)} = \underline{h}(\boldsymbol{W}_l \boldsymbol{a}^{l-1} + \boldsymbol{b}_l)$

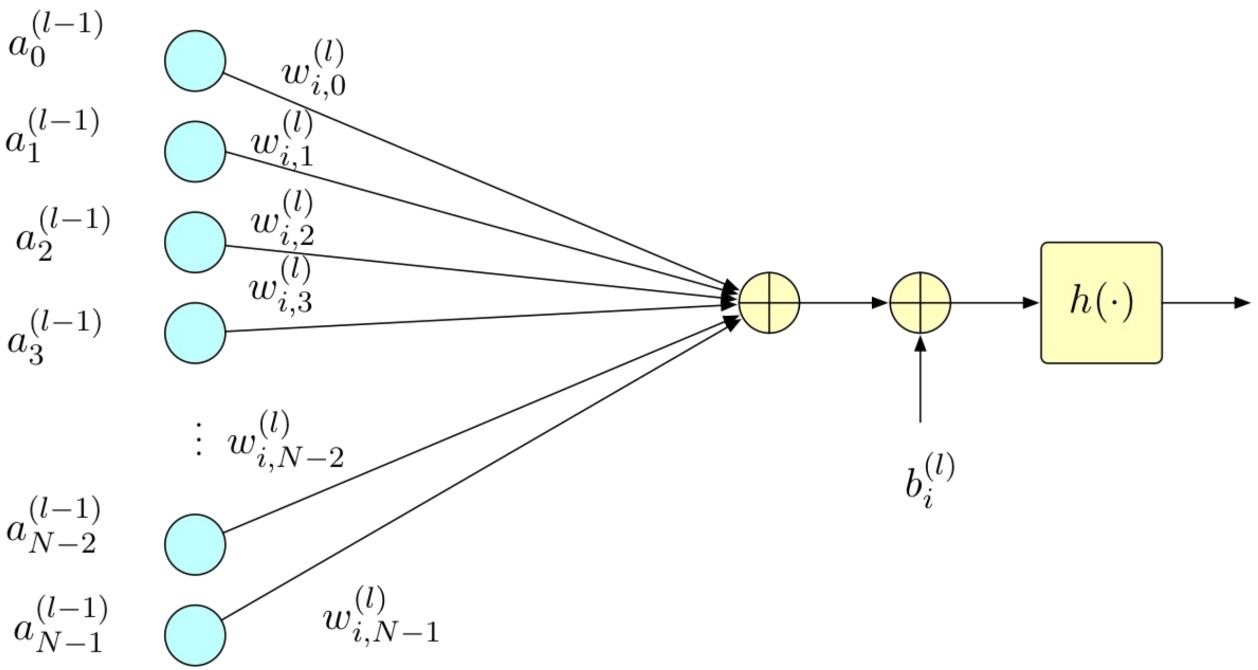
$\Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$ (trainable parameters)



Backward propagation (training)

Learn the trainable parameters using SGD and the chain-rule

MLP FORWARD PROPAGATION DETAILS



$$a_i^{(l)} = h\left(\left[\mathbf{w}_i^{(l)}\right]^{\mathrm{t}} \mathbf{a}^{(l-1)} + b_i^{(l)}\right)$$

processing at the i^{th} neuron (node) at layer l

look familiar?

USC Viterbi School of Engineering

BACK-PROPAGATION

MLP BACKPROP IDEA

do SGD on all trainable parameters:

weight updates: $$w_{i,j}^{(l)} = w_{i,j}^{(l)} - \eta \frac{\partial C}{\partial w_{i,j}^{(l)}}$$

bias updates: $$b_i^{(l)} = b_i^{(l)} - \eta \frac{\partial C}{\partial b_i^{(l)}}$$

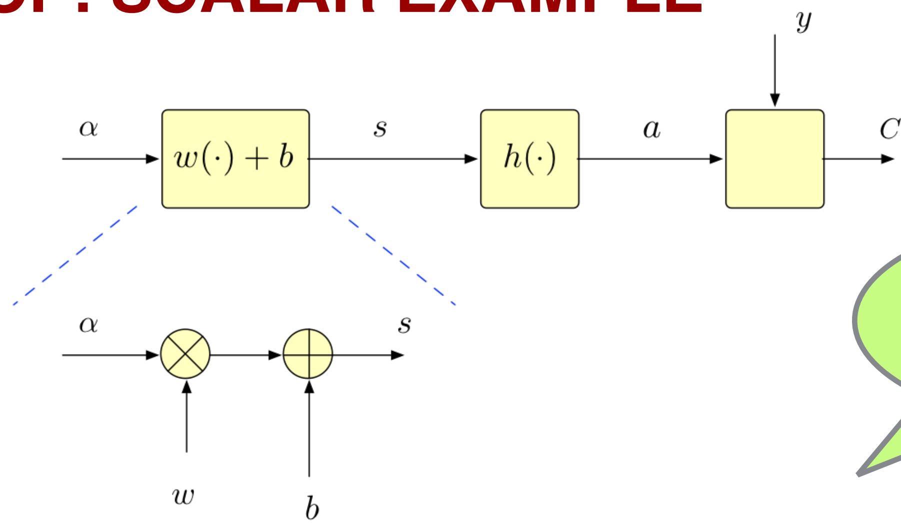
all layers, all indices: $$\forall l \in \{1,2,\dots,L\}; \forall i,j$$

Backprop is an algorithm for computing these, starting at the neural network output and propagating backward to the input layer using the **chain rule**

USC Viterbi School of Engineering

SCALAR BP

MLP BACKPROP: SCALAR EXAMPLE



this is to start the back-prop processing

want to compute: $\dfrac{\partial C}{\partial w}$ and $\dfrac{\partial C}{\partial b}$

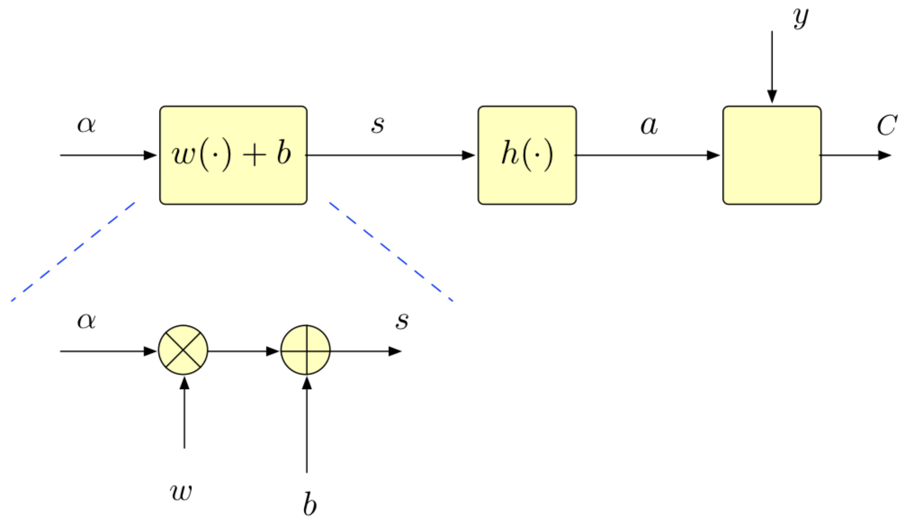
Note that: $\dfrac{\partial s}{\partial w} = \alpha$ $\qquad$ $\dfrac{\partial s}{\partial b} = 1$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial s}\frac{\partial s}{\partial w} = \alpha\frac{\partial C}{\partial s}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial s}\frac{\partial s}{\partial b} = \frac{\partial C}{\partial s}$$

so we can find $\dfrac{\partial C}{\partial s}$ then convert to the desired partials easily by chain rule

MLP BACKPROP: SCALAR EXAMPLE



shorthand

$$\frac{\partial C}{\partial s} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial s}$$

$$= \frac{\partial C}{\partial a}\frac{\partial h(s)}{\partial s}$$

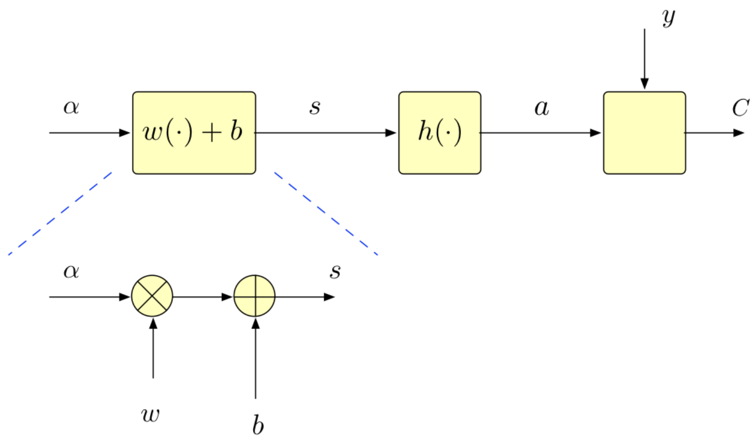
$$= \dot{C}(a)\dot{h}(s)$$

$$\dot{C}(a) = \frac{\partial C}{\partial a}$$

$$\dot{h}(s) = \frac{\partial h(s)}{\partial s}$$

$$\boxed{\begin{aligned} \frac{\partial C}{\partial w} &= \dot{C}(a)\dot{h}(s)\alpha \\ \frac{\partial C}{\partial b} &= \dot{C}(a)\dot{h}(s) \end{aligned}}$$

MLP BACKPROP: SCALAR EXAMPLE



$$\frac{\partial C}{\partial s} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial s} = \frac{\partial C}{\partial a}\frac{\partial h(s)}{\partial s} = \dot{C}(a)\dot{h}(s)$$

$$\frac{\partial C}{\partial w} = \dot{C}(a)\dot{h}(s)\alpha$$

$$\frac{\partial C}{\partial b} = \dot{C}(a)\dot{h}(s)$$

example:

$$C(a) = \frac{1}{2}(y-a)^2$$

$$h(s) = \sigma(s) = \frac{1}{1+e^{-s}}$$

$$\dot{C}(a) = \frac{\partial C}{\partial a} = -(y-a)$$

$$\dot{h}(s) = \frac{\partial h(s)}{\partial s} = \sigma(s)(1-\sigma(s))$$

$$w \leftarrow w + \eta a(1-a)(y-a)\alpha$$

$$b \leftarrow b + \eta a(1-a)(y-a)$$

MLP BACKPROP: SCALAR EXAMPLE

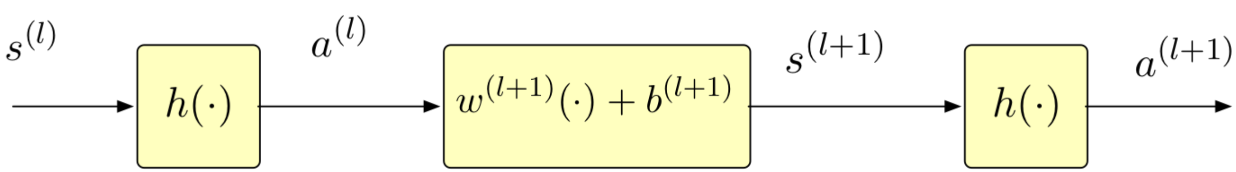
we know how to compute: $$\frac{\partial C}{\partial s}$$

this is to step through the hidden layers

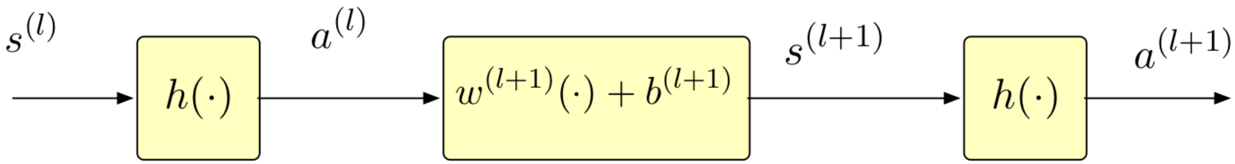
and how to convert to: $$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial s}\frac{\partial s}{\partial w} = \alpha \frac{\partial C}{\partial s}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial s}\frac{\partial s}{\partial b} = \frac{\partial C}{\partial s}$$

now consider:



MLP BACKPROP: SCALAR EXAMPLE



Goal: get a recursion:

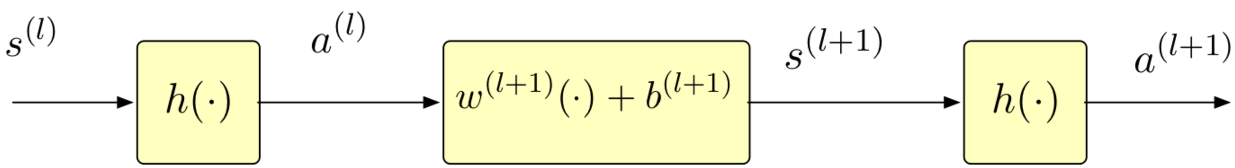
$$\frac{\partial C}{\partial s^{(l)}} \leftarrow \frac{\partial C}{\partial s^{(l+1)}}$$

From previous result:

$$\frac{\partial C}{\partial w^{(l+1)}} = \frac{\partial C}{\partial s^{(l+1)}} \frac{\partial s^{(l+1)}}{\partial w^{(l+1)}} = \frac{\partial C}{\partial s^{(l+1)}} a^{(l)}$$

$$\frac{\partial C}{\partial b^{(l+1)}} = \frac{\partial C}{\partial s^{(l+1)}} \frac{\partial s^{(l+1)}}{\partial b^{(l+1)}} = \frac{\partial C}{\partial s^{(l+1)}}$$

MLP BACKPROP: SCALAR EXAMPLE



Shorthand:

$$\delta^{(l)} \triangleq \frac{\partial C}{\partial s^{(l)}}$$

$$\frac{\partial C}{\partial s^{(l)}} \leftarrow \frac{\partial C}{\partial s^{(l+1)}}$$

$$\boxed{\delta^{(l)} \leftarrow \delta^{(l+1)}}$$

$$\frac{\partial C}{\partial w^{(l)}} = \frac{\partial C}{\partial s^{(l)}} \frac{\partial s^{(l)}}{\partial w^{(l)}} = \frac{\partial C}{\partial s^{(l)}} a^{(l-1)}$$

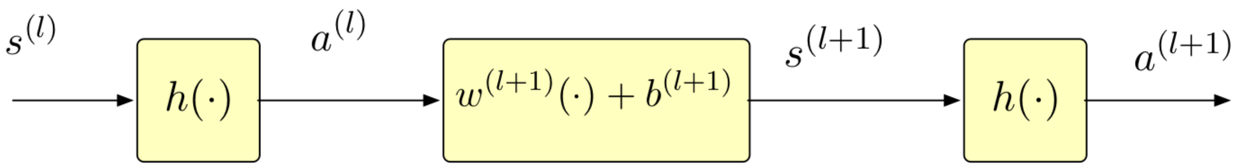
$$\frac{\partial C}{\partial w^{(l)}} = \delta^{(l)} a^{(l-1)}$$

$$\frac{\partial C}{\partial b^{(l)}} = \frac{\partial C}{\partial s^{(l)}} \frac{\partial s^{(l)}}{\partial b^{(l)}} = \frac{\partial C}{\partial s^{(l)}}$$

$$\frac{\partial C}{\partial b^{(l)}} = \delta^{(l)}$$

$$\boxed{\begin{aligned} w^{(l)} &\leftarrow w^{(l)} - \eta \delta^{(l)} a^{(l-1)} \\ b^{(l)} &\leftarrow b^{(l)} - \eta \delta^{(l)} \end{aligned}}$$

MLP BACKPROP: SCALAR EXAMPLE



$$\delta^{(l)} \leftarrow \delta^{(l+1)}$$

$$\delta^{(l)} = \frac{\partial C}{\partial s^{(l)}} = \frac{\partial C}{\partial s^{(l+1)}} \frac{\partial s^{(l+1)}}{\partial s^{(l)}}$$

$$= \delta^{(l+1)} \frac{\partial s^{(l+1)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial s^{(l)}}$$

$$= \delta^{(l+1)} w^{(l+1)} \dot{h}\left(s^{(l)}\right)$$

$$= \dot{h}\left(s^{(l)}\right) w^{(l+1)} \delta^{(l+1)}$$

$$= \dot{a}^{(l)} w^{(l+1)} \delta^{(l+1)}$$

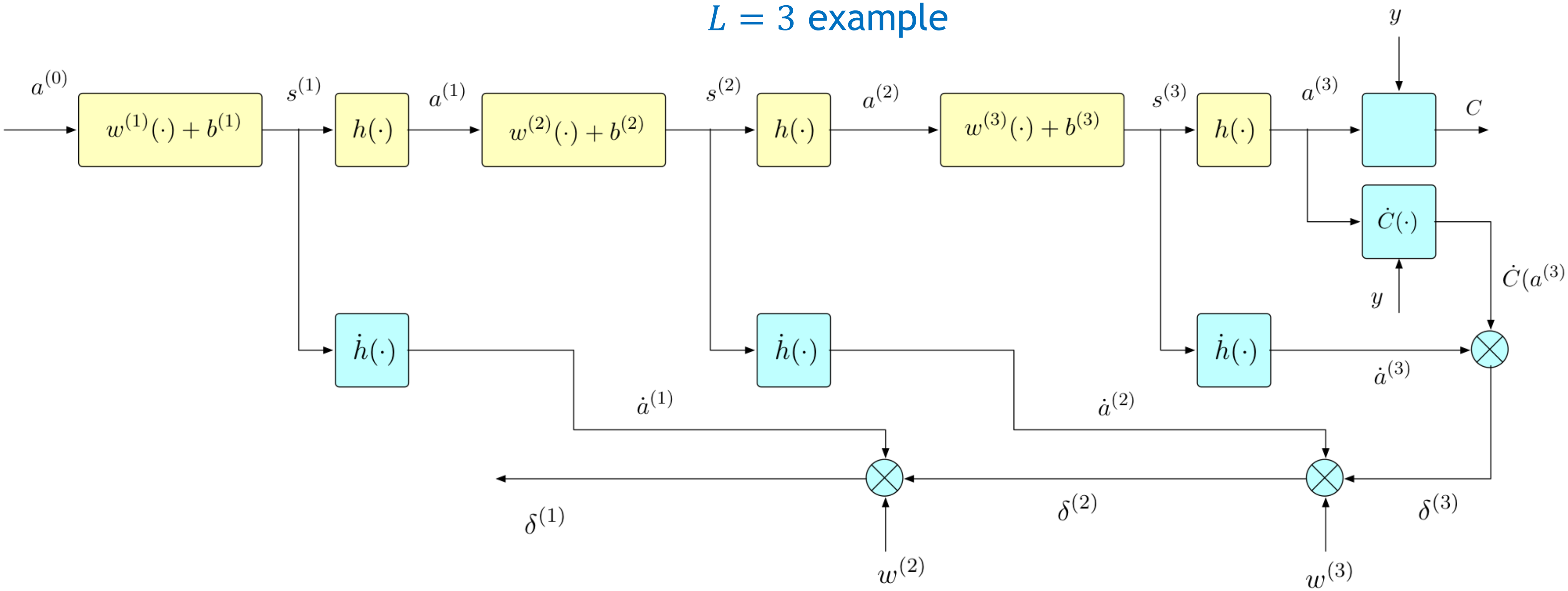
Shorthand:

$$\dot{a}^{(l)} \triangleq \dot{h}\left(s^{(l)}\right)$$

$$\boxed{\begin{aligned} w^{(l)} &\leftarrow w^{(l)} - \eta \delta^{(l)} a^{(l-1)} \\ b^{(l)} &\leftarrow b^{(l)} - \eta \delta^{(l)} \end{aligned}}$$

MLP BACKPROP: SCALAR EXAMPLE

$L = 3$ example



Note: update weights and biases only **after all** the deltas are computed

$$w^{(l)} \leftarrow w^{(l)} - \eta \delta^{(l)} a^{(l-1)}$$

$$b^{(l)} \leftarrow b^{(l)} - \eta \delta^{(l)}$$

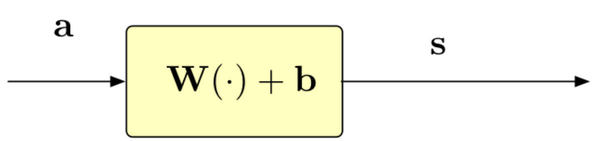
$$\delta^{(l)} = \dot{a}^{(l)} w^{(l+1)} \delta^{(l+1)}$$

VECTOR BP

MLP BACKPROP

now we just need to handle the vector case…

(let’s just repeat what we did)



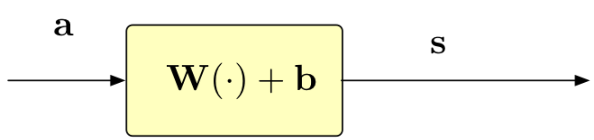
Three Problems:

1. How to initialize the gradient of loss w.r.t. linear/pre activation s
2. How to update gradient w.r.t. s one step backwards
3. How to convert gradient w.r.t. s to gradients on weights and biases

MLP BACKPROP

now we just need to handle the vector case…

(let’s just repeat what we did)



$$\mathbf{s} = \mathbf{W}\mathbf{a} + \mathbf{b}$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$s_n = \left(\sum_m w_{nm} a_m \right) + b_n$$

Suppose we know: $\nabla_{\mathbf{s}} C = \boldsymbol{\delta}$

How to convert to: $\dfrac{\partial C}{\partial w_{ij}}$ $\dfrac{\partial C}{\partial b_i}$

this is “problem 3”

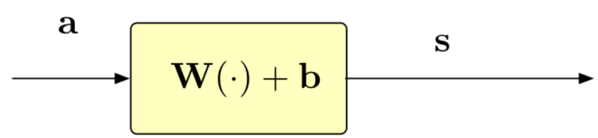
MLP BACKPROP

we use this repeatedly...

$$\frac{\partial C\big(s_0(w_0,w_1),s_1(w_0,w_1)\big)}{\partial w_0} = \frac{\partial C\big(s_0(w_0,w_1),s_1(w_0,w_1)\big)}{\partial s_0}\frac{\partial s_0}{\partial w_0} + \frac{\partial C\big(s_0(w_0,w_1),s_1(w_0,w_1)\big)}{\partial s_1}\frac{\partial s_1}{\partial w_0}$$

$$\frac{\partial C}{\partial w_0} = \frac{\partial C}{\partial s_0}\frac{\partial s_0}{\partial w_0} + \frac{\partial C}{\partial s_1}\frac{\partial s_1}{\partial w_0}$$

MLP BACKPROP



$$\mathbf{s} = \mathbf{W}\mathbf{a} + \mathbf{b}$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$s_n = \left(\sum_m w_{nm} a_m \right) + b_n$$

Suppose we know: $\nabla_{\mathbf{s}} C = \boldsymbol{\delta}$

$$\frac{\partial C}{\partial w_{ij}} = \sum_m \frac{\partial C}{\partial s_m} \frac{\partial s_m}{\partial w_{ij}}$$

$$s_m = \left(\sum_n w_{mn} a_m \right) + b_m$$

$$\frac{\partial s_m}{\partial w_{ij}} = \begin{cases} 0 & m \neq i \\ a_j & m = i \end{cases}$$

$$\frac{\partial C}{\partial w_{ij}} = \frac{\partial C}{\partial s_i} a_j = \delta_i a_j$$

$$\frac{\partial C}{\partial b_i} = \frac{\partial C}{\partial s_i} = \delta_i$$

$$\boxed{\begin{aligned} \mathbf{W} &\leftarrow \mathbf{W} - \eta \boldsymbol{\delta} \mathbf{a}^T \\ \mathbf{b} &\leftarrow \mathbf{b} - \eta \boldsymbol{\delta} \end{aligned}}$$

δ-RECURSION

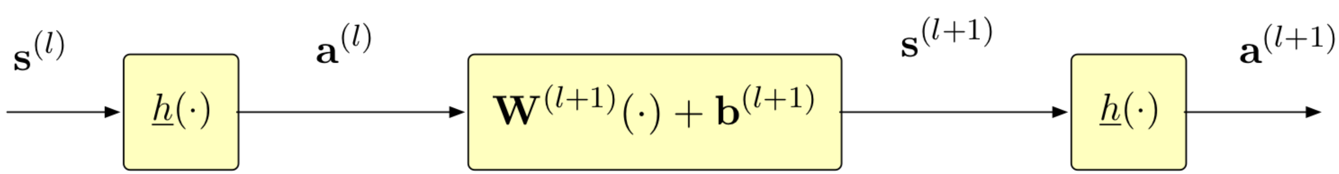
MLP BACKPROP

assume we know how to compute: $\nabla_{\mathbf{s}} C = \boldsymbol{\delta}$

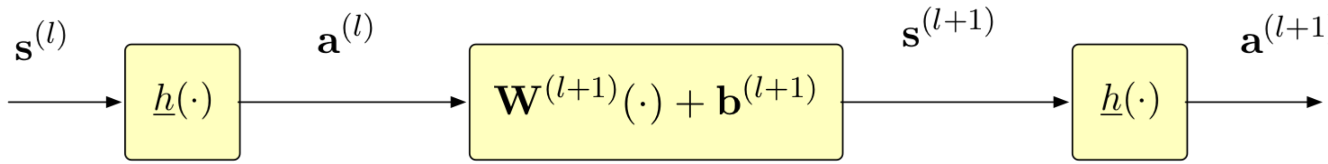
and how to convert to: $\dfrac{\partial C}{\partial w_{ij}} = \dfrac{\partial C}{\partial s_i} a_j \qquad \dfrac{\partial C}{\partial b_i} = \dfrac{\partial C}{\partial s_i}$

this is to step through the hidden layers (problem 2)

now consider:



MLP BACKPROP: DELTA RECURSION



Goal: get a recursion:

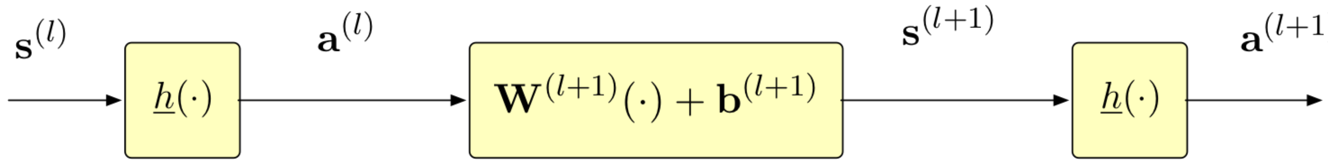
$$\nabla_{\mathbf{s}^{(l)}} C \leftarrow \nabla_{\mathbf{s}^{(l+1)}} C$$

From previous result:

$$\frac{\partial C}{\partial w_{ij}^{(l+1)}} = \frac{\partial C}{\partial s_i^{(l+1)}} a_j^{(l)}$$

$$\frac{\partial C}{\partial b_i^{(l+1)}} = \frac{\partial C}{\partial s_i^{(l+1)}}$$

MLP BACKPROP: DELTA RECURSION



$$\frac{\partial C}{\partial s^{(l)}} \leftarrow \frac{\partial C}{\partial s^{(l+1)}}$$

$$\nabla_{\mathbf{s}^{(l)}} C \triangleq \boldsymbol{\delta}^{(l)}$$

Shorthand: $$\boxed{\delta^{(l)} \leftarrow \delta^{(l+1)}}$$

$$\frac{\partial C}{\partial s^{(l)}} = \delta_i^{(l)}$$

$$= \sum_m \frac{\partial C}{\partial s_m^{(l+1)}} \frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}}$$

$$= \sum_m \delta_m^{(l+1)} \frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}}$$

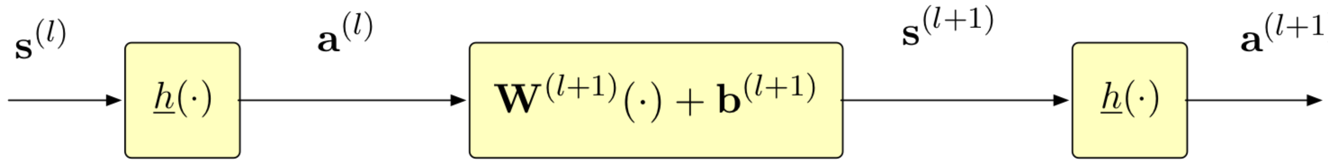
$$\frac{\partial C}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)}$$

$$\frac{\partial C}{\partial b_i^{(l)}} = \delta_i^{(l)}$$

$$\boxed{\begin{aligned} \mathbf{W}^{(l)} &\leftarrow \mathbf{W}^{(l)} - \eta \boldsymbol{\delta}^{(l)} \left[\mathbf{a}^{(l-1)}\right]^T \\ \mathbf{b}^{(l)} &\leftarrow \mathbf{b}^{(l)} - \eta \boldsymbol{\delta}^{(l)} \end{aligned}}$$

(cont. next slide)

MLP BACKPROP: DELTA RECURSION



$$\frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}} = \sum_n \frac{\partial s_m^{(l+1)}}{\partial a_n^{(l)}} \frac{\partial a_n^{(l)}}{\partial s_i^{(l)}}$$

$$\frac{\partial C}{\partial s^{(l)}} = \delta_i^{(l)}$$

$$= \sum_m \frac{\partial C}{\partial s_m^{(l+1)}} \frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}}$$

$$= \sum_m \delta_m^{(l+1)} \frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}}$$

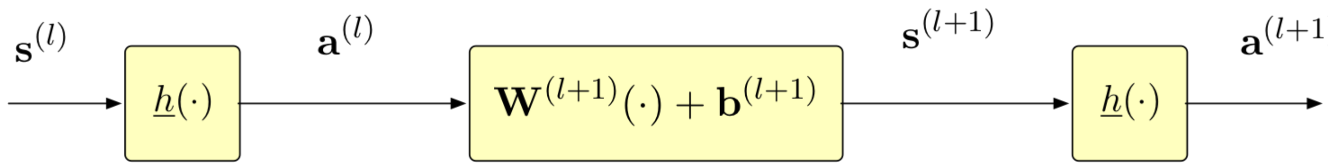
$$\frac{\partial a_n^{(l)}}{\partial s_i^{(l)}} = \begin{cases} \dot{h}\left(s_i^{(l)}\right) & i = n \\ 0 & i \neq n \end{cases}$$

assumes $h(\cdot)$ is component-wise vector function

$$\frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}} = \frac{\partial s_m^{(l+1)}}{\partial a_i^{(l)}} \dot{h}\left(s_i^{(l)}\right)$$

$$= w_{mi} \dot{h}\left(s_i^{(l)}\right)$$

MLP BACKPROP: DELTA RECURSION



$$
\begin{aligned}
\delta_i^{(l)} &= \sum_m \delta_m^{(l+1)} \frac{\partial s_m^{(l+1)}}{\partial s_i^{(l)}} \\
&= \sum_m \delta_m^{(l+1)} w_{mi}^{(l+1)} \dot{h}\left(s_i^{(l)}\right) \\
&= \left(\sum_m \delta_m^{(l+1)} w_{mi}^{(l+1)}\right) \dot{h}\left(s_i^{(l)}\right) \\
&= \left(\sum_m \delta_m^{(l+1)} w_{mi}^{(l+1)}\right) \dot{a}_i^{(l)} \\
&= \left[\left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)}\right] \dot{a}_i^{(l)} \\
\boldsymbol{\delta}^{(l)} &= \left[\left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)}\right] \odot \dot{a}^{(l)}
\end{aligned}
$$

shorthand:

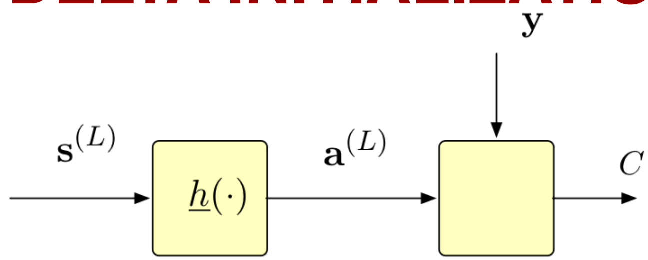
$$\dot{a}_i^{(l)} = \dot{h}\left(s_i^{(l)}\right)$$

$$\dot{\mathbf{a}}_i^{(l)} = \dot{h}\left(\mathbf{s}_i^{(l)}\right)$$

$$\mathbf{v} \odot \mathbf{w} = \begin{bmatrix} v_1 w_1 \\ v_2 w_2 \\ \vdots \\ v_D w_D \end{bmatrix}$$

Hadamard product: .* in Matlab or * for np.array

MLP BACKPROP: DELTA INITIALIZATION



this is to start the back-prop processing (problem 1)

Assume additive costs across components (very common)

$$C_{tot}(\mathbf{y}, \mathbf{a}) = \sum_i C_i(y_i, a_i)$$

$$\frac{\partial C_{tot}}{\partial a_i} = \dot{C}_i(y_i, a_i)$$

$$\frac{\partial C_{tot}}{\partial s_i^{(L)}} = \sum_m \frac{\partial C_{tot}}{\partial a_m^{(L)}} \frac{\partial a_m^{(L)}}{\partial s_i^{(L)}}$$

$$= \dot{C}_i\left(a_i^{(L)}\right) \dot{h}\left(s_i^{(L)}\right)$$

$$\boldsymbol{\delta}^L = \nabla_{\mathbf{s}^{(L)}} C_{tot} = \nabla_{\mathbf{s}^{(L)}} C$$

$$= \dot{C}\left(a^{(L)}\right) \odot \dot{h}\left(s^{(L)}\right)$$

$$= \dot{C}\left(a^{(L)}\right) \odot \dot{a}^{(L)}$$

MLP BACKPROP: SUMMARY

$$\mathbf{a}^{(l)} = \underline{h}\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

$$\dot{\mathbf{a}}^{(l)} = \underline{\dot{h}}\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}_i\right)$$

$$\boldsymbol{\delta}^{(L)} = \dot{\mathbf{a}}^{(L)} \odot \underline{\dot{C}}\left(\mathbf{y}, \mathbf{a}^{(L)}\right)$$

$$\boldsymbol{\delta}^{(l)} = \dot{\mathbf{a}}^{(l)} \odot \left[\left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)}\right]$$

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta\boldsymbol{\delta}^{(l)}\left[\mathbf{a}^{(l-1)}\right]^T$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta\boldsymbol{\delta}^{(l)}$$

specialized to vectorized output activation and additive cost

MLP BACKPROP: SUMMARY

$$\mathbf{a}_l = \mathrm{act}(\mathbf{W}_l \mathbf{a}_{l-1} + \mathbf{b}_l)$$ (activations)

$$\dot{\mathbf{a}}_l = \dot{\mathrm{act}}(\mathbf{W}_l \mathbf{a}_{l-1} + \mathbf{b}_l)$$ (derivative activations)

$$\boldsymbol{\delta}_L = \dot{\mathbf{a}}_L \odot \dot{\mathrm{cost}}(\mathbf{y}, \mathbf{a}_L)$$ (delta initialization)

$$\boldsymbol{\delta}_l = \dot{\mathbf{a}}_l \odot \left[\mathbf{W}_{l+1}^T \boldsymbol{\delta}_{l+1}\right]$$ (delta recursion)

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \boldsymbol{\delta}_l \mathbf{a}_{l-1}^T$$ (weight SGD update)

$$\mathbf{b}_l \leftarrow \mathbf{b}_l - \eta \boldsymbol{\delta}_l$$ (bias SGD update)

specialized to vectorized output activation and additive cost

VARIATIONS ON BP

USING BATCHES

For each data sample: $(\mathbf{x}[n], \mathbf{y}[n])$

Do FF and BP to compute activations, a-dots, and deltas

$$\mathbf{a}_l[n] = \text{act}(\mathbf{W}_l \mathbf{a}_{l-1}[n] + \mathbf{b}_l)$$

$$\dot{\mathbf{a}}_l[n] = \dot{\text{act}}(\mathbf{W}_l \mathbf{a}_{l-1}[n] + \mathbf{b}_l)$$

$$\boldsymbol{\delta}_L[n] = \dot{\mathbf{a}}_L[n] \odot \dot{\text{cost}}(\mathbf{y}[\mathbf{n}], \mathbf{a}_L[n])$$

$$\boldsymbol{\delta}_l[n] = \dot{\mathbf{a}}_l[n] \odot \left[\mathbf{W}_{l+1}^T \boldsymbol{\delta}_{l+1}[n]\right]$$

Do one SGD update after finishing the batch

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \frac{1}{B} \sum_{n=1}^{B} \boldsymbol{\delta}_l[n] \mathbf{a}_{l-1}^T[n]$$

$$\mathbf{b}_l \leftarrow \mathbf{b}_l - \eta \frac{1}{B} \sum_{n=1}^{B} \boldsymbol{\delta}_l[n]$$

EFFECTS OF REGULARIZERS

For typical weight-penalty regularizers (*e.g.*, L1 and L2), these are direct functions of the weights

Example:

$$C_{reg} = C + \lambda \sum_{l} \left\| W^{(l)} \right\|^2 = C + \lambda \sum_{l} \sum_{i,j} \left[W_{i,j}^{(l)} \right]^2$$

$$\frac{\partial C_{reg}}{\partial w_{i,j}^{(l)}} = \frac{\partial C}{\partial w_{i,j}^{(l)}} + 2\lambda w_{i,j}^{(l)}$$

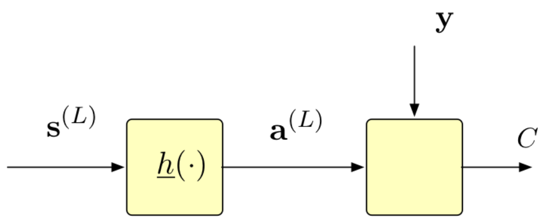
Minor change to gradient update:

$$W^{(l)} \leftarrow W^{(l)} - \eta \left(\delta^{(l)} \left[a^{(l-1)} \right]^T + 2\lambda W^{(l)} \right)$$

NON-VECTORIZED ACTIVATIONS

MLP BACKPROP: NON-VECTORIZED OUTPUT ACTIVATION

recall, assuming the activation is a vectorized scalar function:



$$\frac{\partial C_{tot}}{\partial s_i^{(L)}} = \sum_m \frac{\partial C}{\partial a_m^{(L)}} \frac{\partial a_m^{(L)}}{\partial s_i^{(L)}}$$

$$= \dot{C}_i\left(a_i^{(L)}\right) \dot{h}\left(s_i^{(L)}\right)$$

$$\boldsymbol{\delta}^{(L)} = \underline{\dot{C}}\left(\mathbf{a}^{(L)}\right) \odot \dot{\mathbf{a}}^{(L)}$$

if this final output activation is not a vectorized scalar function:

$$\frac{\partial C_{tot}}{\partial s_i^{(L)}} = \sum_m \frac{\partial C}{\partial a_m^{(L)}} \frac{\partial a_m^{(L)}}{\partial s_i^{(L)}}$$

$$= \sum_m \frac{\partial C}{\partial a_m^{(L)}} \frac{\partial h_m\left(\mathbf{s}^{(L)}\right)}{\partial s_i^{(L)}}$$

$$= \sum_m \dot{A}_{i,m}^{(L)} \frac{\partial C}{\partial a_m^{(L)}}$$

$$\dot{A}_{i,m}^{(L)} = \left[\dot{\mathbf{A}}^{(L)}\right]_{i,m} = \frac{\partial h_m\left(\mathbf{s}^{(L)}\right)}{\partial s_i^{(L)}}$$

(denominator convention)

$$\boldsymbol{\delta}^{(L)} = \dot{\mathbf{A}}^{(L)}\ \underline{\dot{C}}\left(\mathbf{a}^{(L)}\right)$$

USC Viterbi School of Engineering

MLP BACKPROP: NON-VECTORIZED OUTPUT ACTIVATION

$$\mathbf{a}^{(l)} = \underline{h}\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

$$\dot{\mathbf{a}}^{(l)} = \underline{\dot{h}}\left(\mathbf{W}^{(l)}\mathbf{a}_{i-1} + \mathbf{b}_i\right)$$

$$\boldsymbol{\delta}^{(L)} = \dot{\mathbf{A}}^{(L)}\underline{\dot{C}}\left(\mathbf{a}^{(l)}\right)$$

$$\boldsymbol{\delta}^{(l)} = \dot{\mathbf{a}}^{(l)} \odot \left[\left(\mathbf{W}^{(l+1)}\right)^T \boldsymbol{\delta}^{(l+1)}\right]$$

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta\boldsymbol{\delta}^{(l)}\left[\mathbf{a}^{(l-1)}\right]^T$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta\boldsymbol{\delta}^{(l)}$$

similar change could be made if other layers had general activations (not usual in practice)

VECTOR CALCULUS

Always remember this is **just the chain rule…**

But it is tedious…

So there are various conventions of doing derivatives w.r.t. vectors and matrices

$e.g.$, you have seen the Jacobian matrix $d\mathbf{y}/d\mathbf{x}$ for change of variables in multidimensional integration (EE 503)

There are several conventions for keeping track of all the partial derivatives and storing them in tables (vectors, matrices, tensors)

COMPACT TENSOR/MATRIX/VECTOR CALCULUS NOTATION

recall how we started with the vector version of BP…

$$\frac{\partial C}{\partial w_0} = \frac{\partial C}{\partial s_0}\frac{\partial s_0}{\partial w_0} + \frac{\partial C}{\partial s_1}\frac{\partial s_1}{\partial w_0}$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial s_0}\frac{\partial s_0}{\partial w_1} + \frac{\partial C}{\partial s_1}\frac{\partial s_1}{\partial w_1}$$

this is just book-keeping convention for chain rule

This suggests a matrix form:

$$\nabla_{\mathbf{w}} C = \begin{bmatrix} \frac{\partial C}{\partial w_0} \\ \frac{\partial C}{\partial w_1} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial s_0}{\partial w_0} & \frac{\partial s_1}{\partial w_0} \\ \frac{\partial s_0}{\partial w_1} & \frac{\partial s_1}{\partial w_1} \end{bmatrix} \begin{bmatrix} \frac{\partial C}{\partial s_0} \\ \frac{\partial C}{\partial s_1} \end{bmatrix}$$

$$\nabla_{\mathbf{w}} C = \frac{d\mathbf{s}}{d\mathbf{w}} \nabla_{\mathbf{s}} C$$

denominator convention with left-handed chain rule

$$[\nabla_{\mathbf{w}} C]^T = [\nabla_{\mathbf{s}} C]^T \begin{bmatrix} \frac{\partial s_0}{\partial w_0} & \frac{\partial s_0}{\partial w_1} \\ \frac{\partial s_1}{\partial w_0} & \frac{\partial s_1}{\partial w_1} \end{bmatrix}$$

$$[\nabla_{\mathbf{w}} C]^T = [\nabla_{\mathbf{s}} C]^T \frac{d\mathbf{s}}{d\mathbf{w}}$$

numerator convention with right-handed chain rule

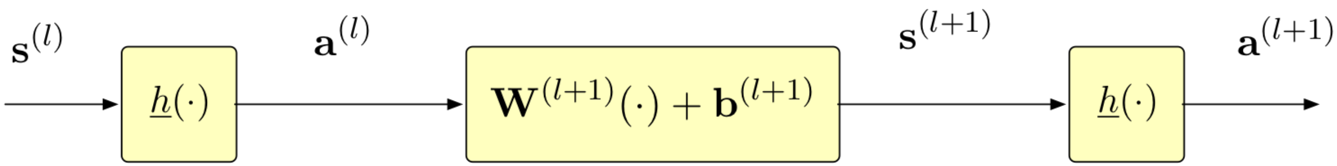
VECTOR DERIVATIVE FORMS

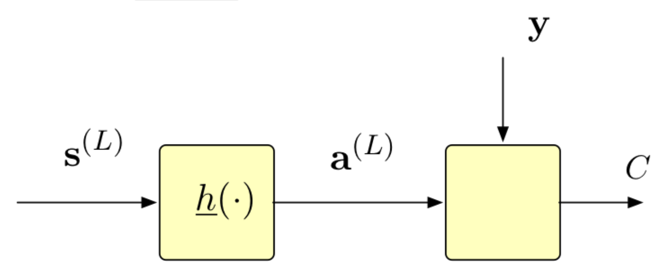
Identities: vector-by-vector $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

Condition	Expression	Numerator layout, i.e. by y and x^T	Denominator layout, i.e. by y^T and x
a is not a function of **x**	$\frac{\partial \mathbf{a}}{\partial \mathbf{x}} =$	$\mathbf{0}$	
	$\frac{\partial \mathbf{x}}{\partial \mathbf{x}} =$	$\mathbf{I}$	
A is not a function of **x**	$\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} =$	$\mathbf{A}$	$\mathbf{A}^\top$
A is not a function of **x**	$\frac{\partial \mathbf{x}^\top \mathbf{A}}{\partial \mathbf{x}} =$	$\mathbf{A}^\top$	$\mathbf{A}$
a is not a function of **x**, **u** = **u**(**x**)	$\frac{\partial a\mathbf{u}}{\partial \mathbf{x}} =$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	
v = *v*(**x**), **u** = **u**(**x**)	$\frac{\partial v\mathbf{u}}{\partial \mathbf{x}} =$	$v\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \mathbf{u}\frac{\partial v}{\partial \mathbf{x}}$	$v\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}}\mathbf{u}^\top$
A is not a function of **x**, **u** = **u**(**x**)	$\frac{\partial \mathbf{A}\mathbf{u}}{\partial \mathbf{x}} =$	$\mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{A}^\top$
u = **u**(**x**), **v** = **v**(**x**)	$\frac{\partial(\mathbf{u}+\mathbf{v})}{\partial \mathbf{x}} =$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$	
u = **u**(**x**)	$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} =$	$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}$
u = **u**(**x**)	$\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{u}))}{\partial \mathbf{x}} =$	$\frac{\partial \mathbf{f}(\mathbf{g})}{\partial \mathbf{g}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{f}(\mathbf{g})}{\partial \mathbf{g}}$

https://en.wikipedia.org/wiki/Matrix_calculus

VECTOR DERIVATIVE DERIVATION OF BP (DENOMINATOR CONVENTION)





(vectorized scalar activation)

$$
\begin{aligned}
\boldsymbol{\delta}^{(L)} &= \frac{\partial C}{\partial s^{(L)}} \\
&= \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial C}{\partial \mathbf{a}^{(L)}} \\
&= \dot{\mathbf{A}}^{(L)} \dot{C}_i\left(a^{(L)}\right) \\
&= \dot{\mathbf{a}}^{(L)} \odot \dot{C}_i\left(a^{(L)}\right)
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{\delta}^{(l)} &= \frac{\partial C}{\partial s^{(l)}} \\
&= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{s}^{(l)}} \frac{\partial \mathbf{s}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \frac{\partial C}{\partial \mathbf{s}^{(l+1)}} \\
&= \dot{\mathbf{A}}^{(l)} \left[\mathbf{W}^{(l+1)}\right]^T \boldsymbol{\delta}^{(l+1)} \\
&= \dot{\mathbf{a}}^{(L)} \odot \left[\mathbf{W}^{(l+1)}\right]^T \boldsymbol{\delta}^{(l+1)}
\end{aligned}
$$

LMS VS. BP

LMS VS MLP-BP

When training an MLP, it is assumed that the target mapping is fixed

— $i.e.$, the probability distributions are not a function of n

$$p_{data}(\mathbf{y}_n|\mathbf{x}_n) \approx p_{model}(\mathbf{y}_n|\mathbf{x}_n;\Theta)$$

if you train a MLP using BP and these data statistics vary with n ($i.e.$, non-stationary) you will get junk!

think of the time-varying LMS example

Can a MLP be used in a non-stationary environment — as a nonlinear adaptive filter?

LMS VS MLP-BP

1. Train an MLP using representative (stationary data)
2. Use on-line learning (batch-size 1) to update the MLP with new data as it becomes available

this could be done on just the last layer if the representative model is good

In this way, an MLP can be used as a direct generalization of the LMS adaptive filter — an on-line (adaptive) non-linear regressor!