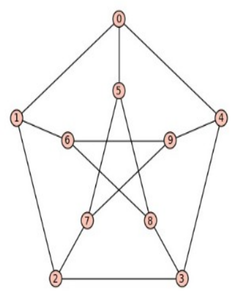
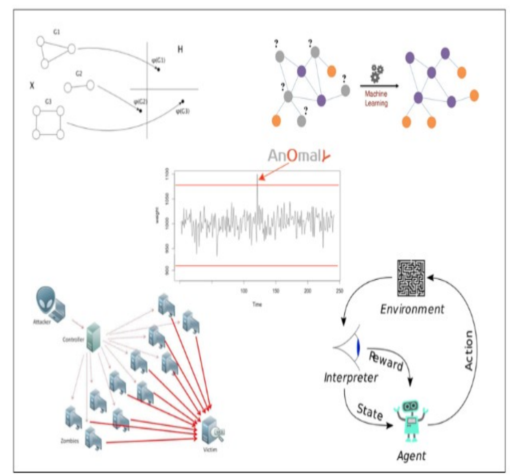
Backpropagation

Pierre Pereira
Université Côte d'Azur / Inria Coati
pierre.pereira@inria.fr





$$
\begin{aligned}
\min \quad & \sum_{e\in\mathcal{E}} y_e \\
\text{s.t.} \quad & \sum_{a\in A_i^+(u)} f_a^i - \sum_{a\in A_i^-(u)} f_a^i = \begin{cases} |V_i| - 1 & \text{if } u = s_i \\ -1 & \text{if } u \neq s_i \end{cases} & \forall u \in V_i,\ V_i \in C \\
& f_a^i \leq |V_i| \cdot x_a, & \forall V_i \in C, a \in A \\
& x_{(u,v)} \leq y_{uv}, & \forall uv \in \mathcal{E} \\
& x_{(v,u)} \leq y_{uv}, & \forall uv \in \mathcal{E}
\end{aligned}
$$

Sources multiples, principalement Polytechnique Montréal

Backpropagation in matrix vector form

- Backpropagation is based on the chain rule of calculus
- Consider the loss for a single-layer network with a softmax output (which corresponds exactly to the model for multinomial logistic regression)
- We use multinomial vectors **y**, with a single dimension $y_k = 1$ for the corresponding class label and whose other dimensions are 0
- Define $\mathbf{f} = [\ f_1(\mathbf{a}), \ \ldots, \ f_K(\mathbf{a})\ ]^T$, and $a_k(\mathbf{x};\boldsymbol{\theta}_k) = \boldsymbol{\theta}_k^{\mathrm{T}}\mathbf{x}$, $\mathbf{a}(\mathbf{x};\boldsymbol{\theta}) = [\ a_1(\mathbf{x};\boldsymbol{\theta}_1), \ \ldots, \ a_K(\mathbf{x};\boldsymbol{\theta}_K)\ ]^T$ where θ_k is a column vector containing the k^{th} row of the parameter matrix
- Consider the softmax loss for **f**(**a**(**x**))

Logistic regression and the chain rule

- Given loss $L = -\sum_{k=1}^{K} y_k \log f_k(\mathbf{x}), \quad f_k(\mathbf{x}) = \frac{\exp(a_k(\mathbf{x}))}{\sum_{c=1}^{K} \exp(a_c(\mathbf{x}))}.$
- Use the chain rule to obtain

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \frac{\partial \mathbf{a}}{\partial \boldsymbol{\theta}_k} \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \frac{\partial L}{\partial \mathbf{f}} = \frac{\partial \mathbf{a}}{\partial \boldsymbol{\theta}_k} \frac{\partial L}{\partial \mathbf{a}}.$$

- Note the order of terms - in vector matrix form terms build from right to left

$$\frac{\partial L}{\partial a_{\mathrm{j}}} = \frac{\partial}{\partial a_{\mathrm{j}}}\left[-\sum_{k=1}^{K} y_k \left[a_k - \log\left[\sum_{c=1}^{K} \exp(a_c)\right]\right]\right]$$

$$= -\left[y_{k=j} - \frac{\exp(a_{k=j})}{\sum_{c=1}^{K} \exp(a_c)}\right] = -\left[y_j - p(y_j \mid \mathbf{x})\right] = -\left[y_j - f_j(\mathbf{x})\right],$$

Matrix vector form of gradient

- We can write $$\frac{\partial L}{\partial \mathbf{a}} = -\left[\mathbf{y} - \mathbf{f}(\mathbf{x})\right] \equiv -\Delta$$

  and since $$\frac{\partial a_{\mathrm{j}}}{\partial \boldsymbol{\theta}_k} = \begin{cases} \dfrac{\partial}{\partial \boldsymbol{\theta}_k} \boldsymbol{\theta}_k^{\mathrm{T}} \mathbf{x} = \mathbf{x} & , j = k \\ 0 & , j \neq k \end{cases}$$

  we have $$\frac{\partial \mathbf{a}}{\partial \boldsymbol{\theta}_k} = \mathbf{H}_k = \begin{bmatrix} 0 & x_1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & x_n & 0 \end{bmatrix}$$

- Notice that we avoid working with the partial derivative of the vector **a** with respect to the matrix θ, because it cannot be represented as a matrix — it is a multidimensional array of numbers (a tensor).

A compact expression for the gradient

- The gradient (as a column vector) for the vector in the *k*th row of the parameter matrix

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \frac{\partial \mathbf{a}}{\partial \boldsymbol{\theta}_k}\frac{\partial L}{\partial \mathbf{a}} = -\begin{bmatrix} 0 & x_1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & x_n & 0 \end{bmatrix}\left[\mathbf{y} - \mathbf{f}(\mathbf{x})\right]$$

$$= -\mathbf{x}(y_k - f_k(x)).$$

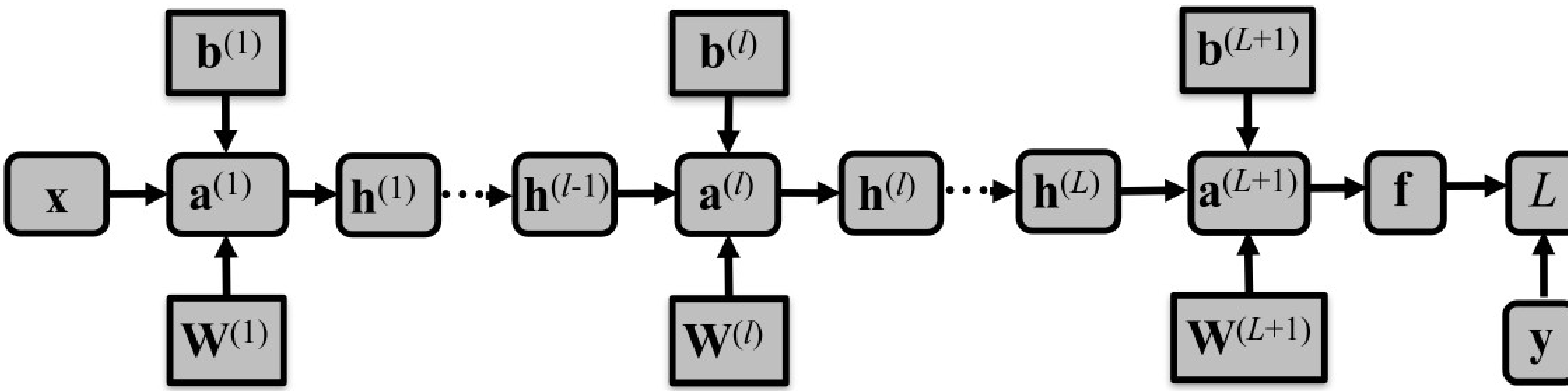
- With a little rearrangement the gradient for the entire matrix of parameters can be written compactly:

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\left[\mathbf{y} - \mathbf{f}(\mathbf{x})\right]\mathbf{x}^{\mathrm{T}} = -\Delta\mathbf{x}^{\mathrm{T}}.$$

Visualizing backpropagation

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \qquad \mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \qquad \mathbf{a}^{(L+1)} = \mathbf{W}^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}$$

$$\mathbf{h}^{(1)} = \mathrm{act}(\mathbf{a}^{(1)}) \qquad \mathbf{h}^{(l)} = \mathrm{act}(\mathbf{a}^{(l)}) \qquad \mathbf{f} = \mathrm{out}(\mathbf{a}^{(L+1)})$$



- In the forward propagation phase we compute terms of the form above

Consider now a multilayer network

- Using the same activation function for all L hidden layers, and a softmax output layer
- The gradient of the k^{th} parameter vector of the L+1$^{\text{th}}$ matrix of parameters is

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k^{(L+1)}} = \frac{\partial \mathbf{a}^{(L+1)}}{\partial \boldsymbol{\theta}_k^{(L+1)}} \frac{\partial L}{\partial \mathbf{a}^{(L+1)}}, \qquad \frac{\partial L}{\partial \mathbf{a}^{(L+1)}} = -\boldsymbol{\Delta}^{(L+1)}$$

$$= -\frac{\partial \mathbf{a}^{(L+1)}}{\partial \boldsymbol{\theta}_k^{(L+1)}} \boldsymbol{\Delta}^{(L+1)}$$

$$= -\mathbf{H}_k^L \boldsymbol{\Delta}^{(L+1)} \qquad \Rightarrow \qquad \frac{\partial L}{\partial \boldsymbol{\theta}^{(L+1)}} = -\boldsymbol{\Delta}^{(L+1)} \tilde{\mathbf{h}}_{(L)}^{\mathrm{T}}.$$

where $\mathbf{H}_k{}^L$ is a matrix containing the activations of the corresponding hidden layer, in column k

Backpropagating errors

- Consider the computation for the gradient of the k^{th} row of the L^{th} matrix of parameters
- Since the bias terms are constant, it is unnecessary to backprop through them, so

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k^{(L)}} = \frac{\partial \mathbf{a}^{(L)}}{\partial \boldsymbol{\theta}_k^{(L)}} \frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L+1)}}{\partial \mathbf{h}^{(L)}} \frac{\partial L}{\partial \mathbf{a}^{(L+1)}}$$

$$= -\frac{\partial \mathbf{a}^{(L)}}{\partial \boldsymbol{\theta}_k^{(L)}} \frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L+1)}}{\partial \mathbf{h}^{(L)}} \Delta^{(L+1)}, \qquad \Delta^{(L)} \equiv \frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{a}^{(L)}} \frac{\partial \mathbf{a}^{(L+1)}}{\partial \mathbf{h}^{(L)}} \Delta^{(L+1)}$$

$$= -\frac{\partial \mathbf{a}^{(L)}}{\partial \boldsymbol{\theta}_k^{(L)}} \Delta^{(L)}$$

- Similarly, we can define $\Delta^{(l)}$ recursively in terms of $\Delta^{(l+1)}$

Backpropagating errors

- The backpropagated error can be written as simply

$$\Delta^{(l)} = \frac{\partial \mathbf{h}^{(l)}}{\partial \mathbf{a}^{(l)}} \frac{\partial \mathbf{a}^{(l+1)}}{\partial \mathbf{h}^{(l)}} \Delta^{(l+1)}, \qquad \frac{\partial \mathbf{h}^{(l)}}{\partial \mathbf{a}^{(l)}} = \mathbf{D}^{(l)}, \qquad \frac{\partial \mathbf{a}^{(l+1)}}{\partial \mathbf{h}^{(l)}} = \mathbf{W}^{\mathrm{T}(l+1)},$$

$$\Delta^{(l)} = \mathbf{D}^{(l)} \mathbf{W}^{\mathrm{T}(l+1)} \Delta^{(l+1)}$$

  where $\mathbf{D}^{(l)}$ contains the partial derivatives of the hidden-layer activation function with respect to the pre-activation input.

- $\mathbf{D}^{(l)}$ is generally diagonal, because activation functions usually operate on an elementwise basis
- $\mathbf{W}^{\mathrm{T}(l+1)}$ arises from the fact that $\mathbf{a}^{(l+1)}(\mathbf{h}^{(l)})=\mathbf{W}^{(l+1)}\mathbf{h}^{(l)}+\mathbf{b}^{(l+1)}$

A general form for gradients

- The gradients for the k^{th} vector of parameters of the l^{th} network layer can therefore be computed using products of matrices of the following form

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k^{(l)}} = -\mathbf{H}_k^{(l-1)} \mathbf{D}^{(l)} \mathbf{W}^{\mathrm{T}(l+1)} \cdots \mathbf{D}^{(L)} \mathbf{W}^{\mathrm{T}(L+1)} \Delta^{(L+1)}, \qquad \frac{\partial L}{\partial \boldsymbol{\theta}^{(l)}} = -\Delta^{(l)} \hat{\mathbf{h}}_{(l-1)}^{\mathrm{T}}$$

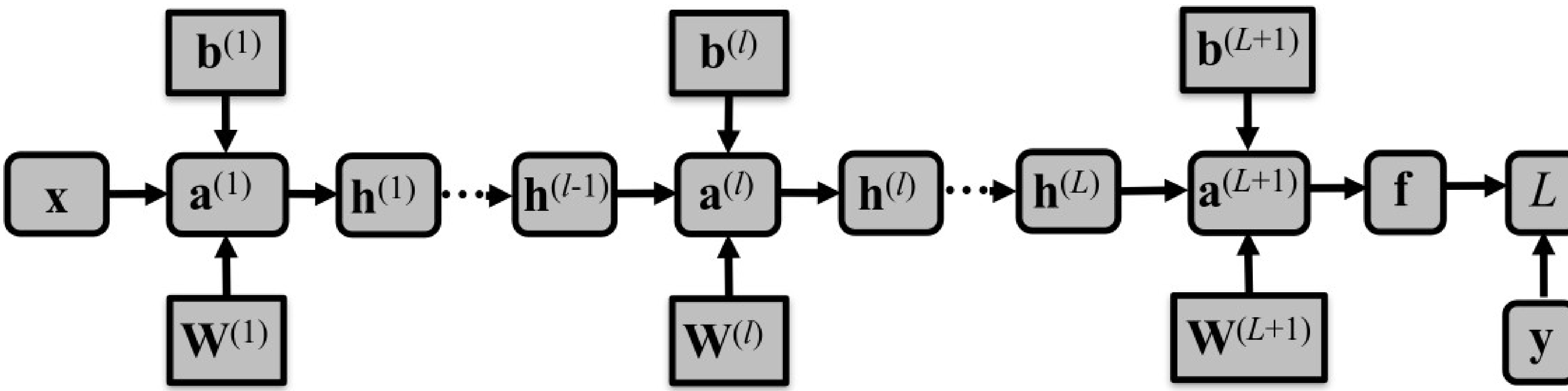
- When l=1, $\hat{\mathbf{h}}_{(0)} = \hat{\mathbf{x}}$, the input data with a 1 appended
- Note: since **D** is usually diagonal the corresponding matrix-vector multiply can be transformed into an element-wise product $\circ$ by extracting the diagonal for **d**

$$\Delta^{(l)} = \mathbf{D}^{(l)} (\mathbf{W}^{\mathrm{T}(l+1)} \Delta^{(l+1)}) = \mathbf{d}^{(l)} \circ (\mathbf{W}^{\mathrm{T}(l+1)} \Delta^{(l+1)})$$

Visualizing backpropagation

$$\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \qquad \mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \qquad \mathbf{a}^{(L+1)} = \mathbf{W}^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}$$

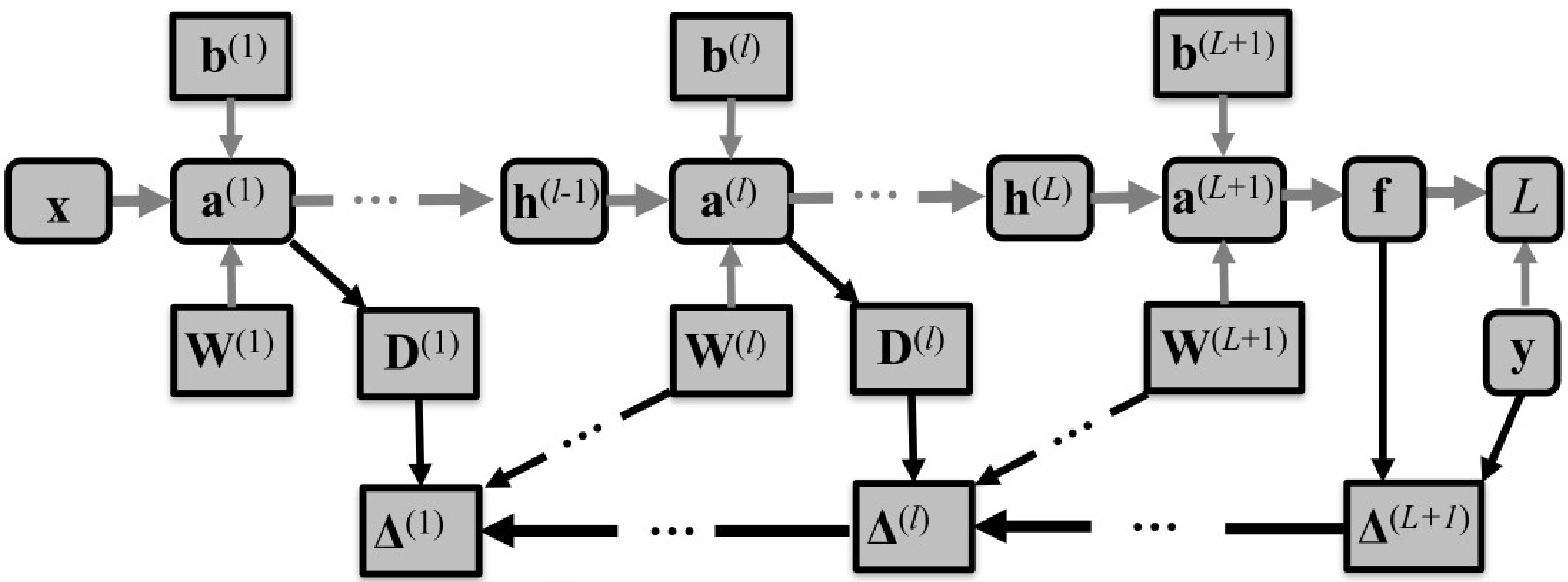
$$\mathbf{h}^{(1)} = \mathrm{act}(\mathbf{a}^{(1)}) \qquad \mathbf{h}^{(l)} = \mathrm{act}(\mathbf{a}^{(l)}) \qquad \mathbf{f} = \mathrm{out}(\mathbf{a}^{(L+1)})$$



- In the forward propagation phase we compute terms of the form above

Visualizing backpropagation

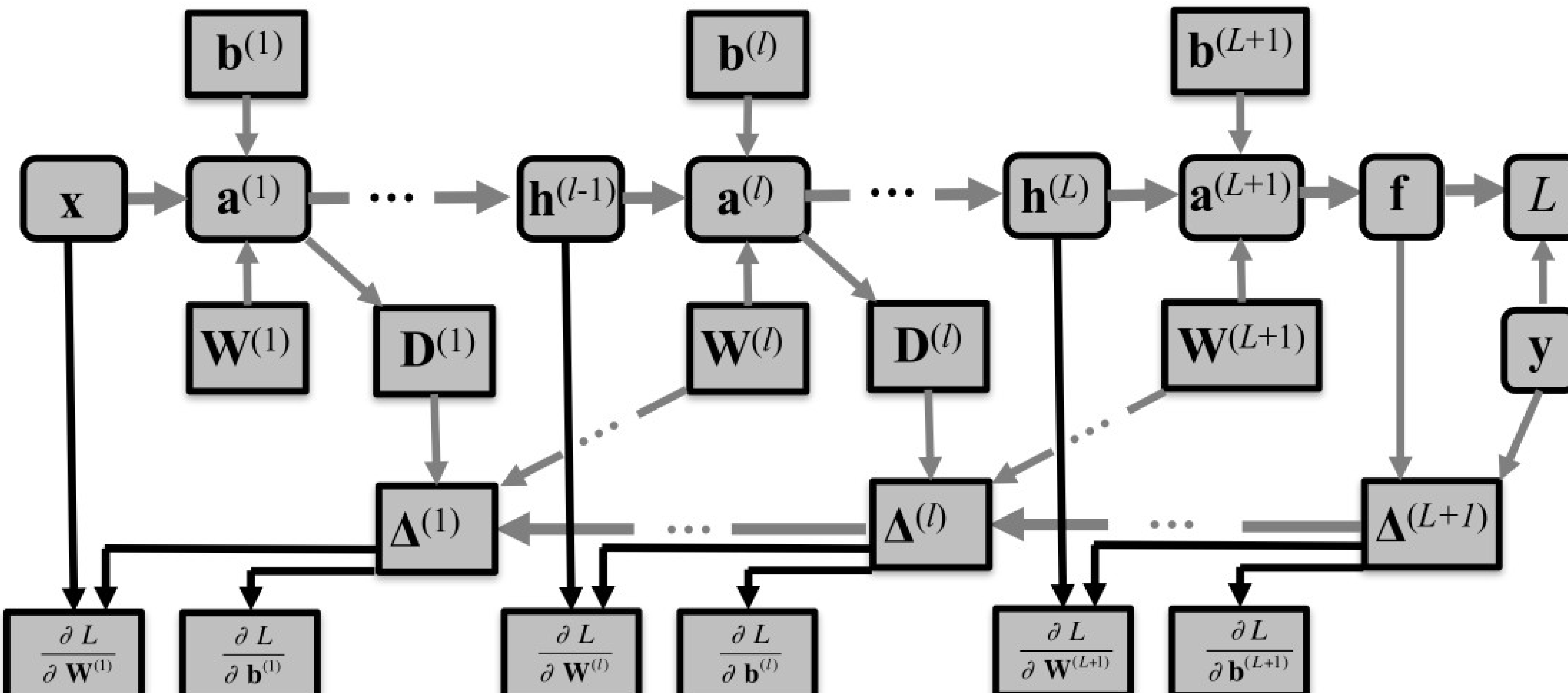
- In the backward propagation phase we compute terms of the form below



$$\Delta^{(l)} = \mathbf{D}^{(l)}\mathbf{W}^{\mathrm{T}(l+1)}\Delta^{(l+1)} \qquad \frac{\partial\, \mathbf{h}^{(l)}}{\partial\, \mathbf{a}^{(l)}} = \mathbf{D}^{(l)} \qquad \frac{\partial\, \mathbf{a}^{(l+1)}}{\partial\, \mathbf{h}^{(l)}} = \mathbf{W}^{\mathrm{T}(l+1)} \qquad \frac{\partial\, L}{\partial\, \mathbf{a}^{(L+1)}} = -\left[\mathbf{y} - \mathbf{f}(\mathbf{x})\right] = -\Delta^{(L+1)}$$

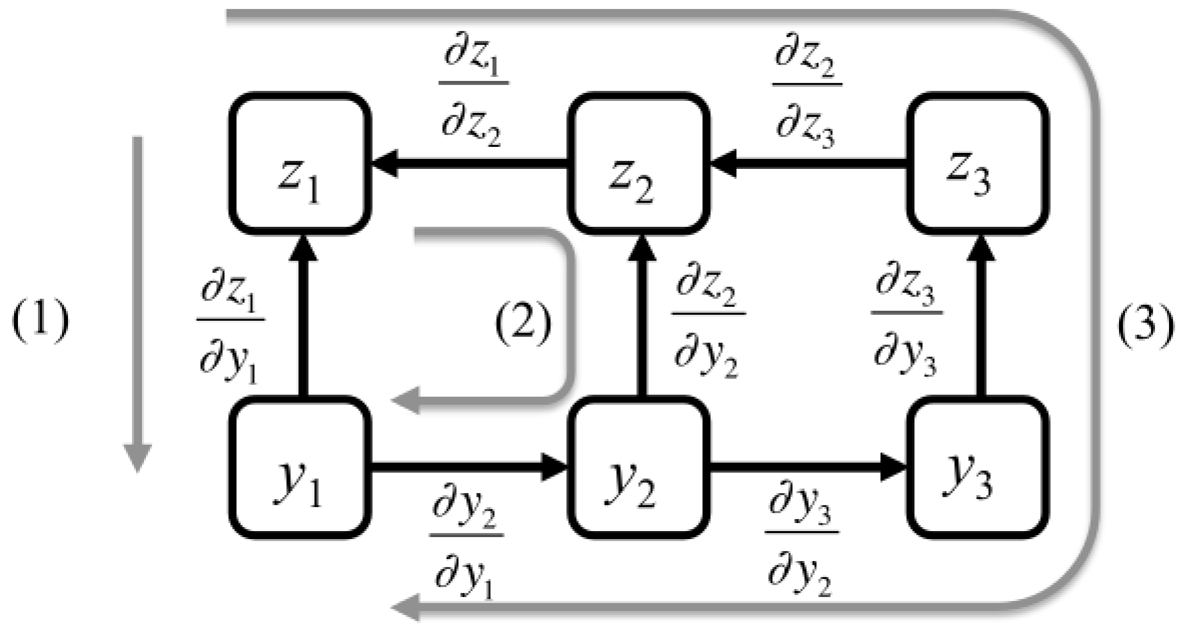
Visualizing backpropagation

- We update the parameters in our model using the simple computations below



$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = -\Delta^{(1)}\mathbf{x}^{\mathrm{T}} \qquad \frac{\partial L}{\partial \mathbf{b}^{(1)}} = -\Delta^{(1)} \qquad \frac{\partial L}{\partial \mathbf{W}^{(l)}} = -\Delta^{(l)}\mathbf{h}_{(l-1)}^{\mathrm{T}} \qquad \frac{\partial L}{\partial \mathbf{b}^{(l)}} = -\Delta^{(l)}$$

Computation graphs



- For more complicated computations, computation graphs can help us keep track of how computations decompose, ex. $z_1 = z_1(y_1, z_2(y_2(y_1), z_3(y_3(y_2(y_1)))))$

$$\frac{\partial z_1}{\partial y_1} = \underbrace{\frac{\partial z_1}{\partial y_1}}_{(1)} + \underbrace{\frac{\partial z_1}{\partial z_2}\frac{\partial z_2}{\partial y_2}\frac{\partial y_2}{\partial y_1}}_{(2)} + \underbrace{\frac{\partial z_1}{\partial z_2}\frac{\partial z_2}{\partial z_3}\frac{\partial z_3}{\partial y_3}\frac{\partial y_3}{\partial y_2}\frac{\partial y_2}{\partial y_1}}_{(3)}$$

Checking an implementation of backpropagation and software tools

- An implementation of the backpropagation algorithm can be checked for correctness by comparing the analytic values of gradients with those computed numerically
- For example, one can add and subtract a small perturbation to each parameter and then compute the symmetric finite difference approximation to the derivative of the loss:

$$\frac{\partial L}{\partial \theta} \approx \frac{L(\theta + \varepsilon) - L(\theta - \varepsilon)}{2\varepsilon}$$

- Many software packages use computation graphs to allow complex networks to be more easily defined and optimized
- Examples include: Theano, TensorFlow, Keras and Torch