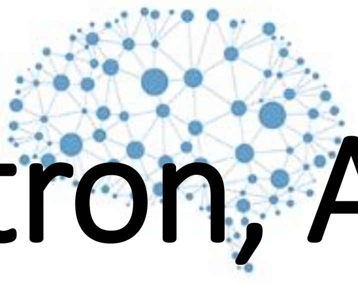


Perceptron, Adaline, Backpropagation

ENEE 4583/5583 Neural Nets

Dr. Alsamman

Slide Credits:

Historical Perspective

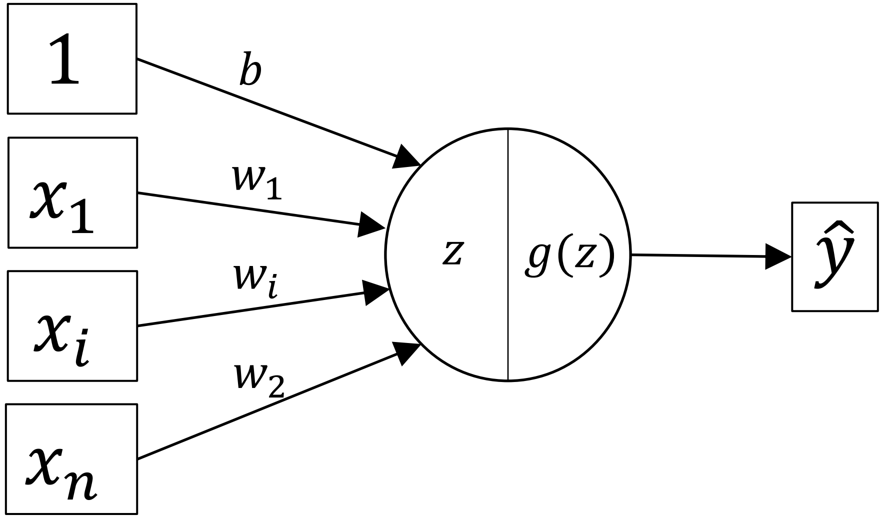
- Computing:
  - Electronic computers (1934)
  - Von Neumann (1945) : architecture of universal computers
  - Turing (1946): universal capabilities of digital machines
  - Shannon (1948): information theory of digital signals
- Binary perceptrons: McCulloch and Pitts (1943)
- Perceptron networks: Rosenblatt (1960)
- Minsky and Papert, *Perceptrons*, (1969)
- Backprop: Rumelhart, Hinton and Williams (1986)
- Deep Learning (2000)

Artificial Neuron

- Weights: $\boldsymbol{w} = w_1, w_2, \dots, w_n$
- Bias: b
- Pre-activation: z

$$z = b + \sum_{i}^{n} w_i x_i = b + \boldsymbol{w^T x}$$

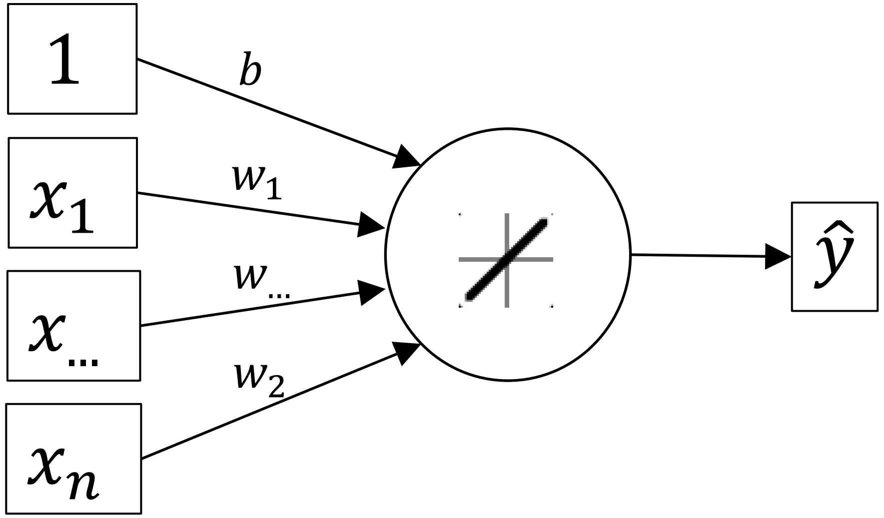
- Activation: $g(z)$
  - Linear
  - Threshold
  - Sigmoid
  - Tanh



Perceptrons

- Early works focused on “neuronal” building blocks to logic circuit: AND, OR, XOR gates
  - Binary thresholds
  - Not very interesting
- Later: “neuronal” logic expressions
  - how to create a logic expression using perceptrons
  - Binary thresholds
  - Algorithms/heuristics for updating weights

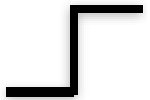
Linear



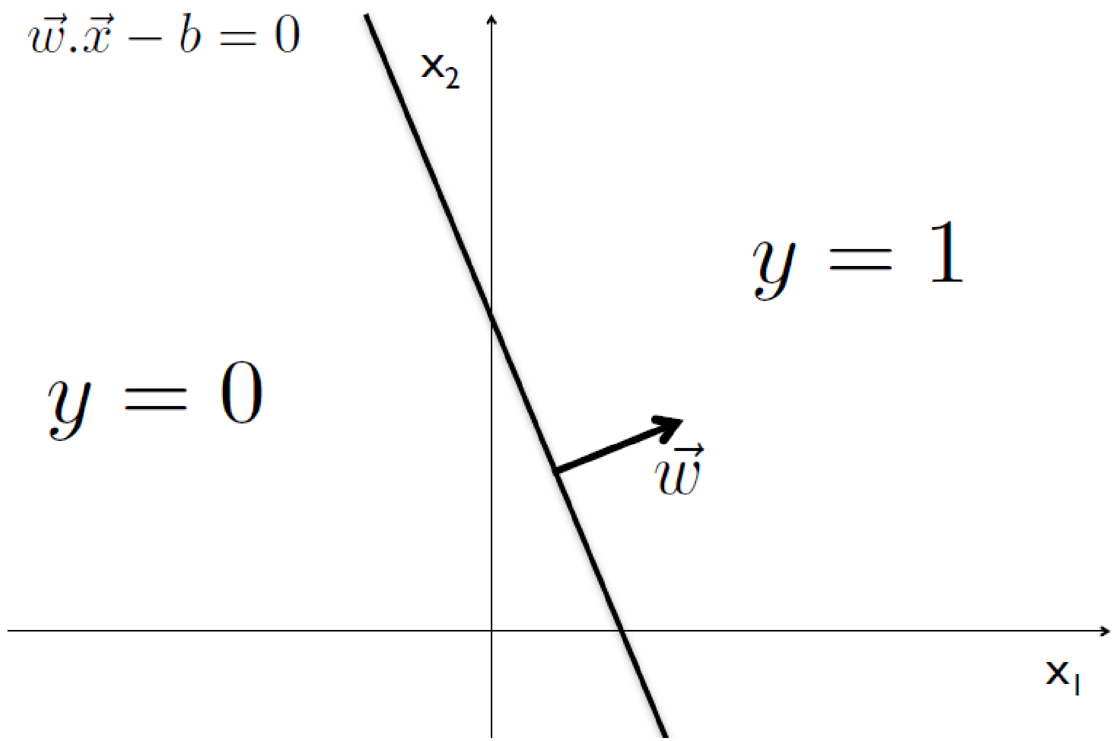
- Linear regression
  - Fitting a line (or a curve)
  - Not suitable for classification

Binary Threshold

$$y = 1 \quad \text{if} \quad \sum_{i}^{n} w_i x_i > T$$

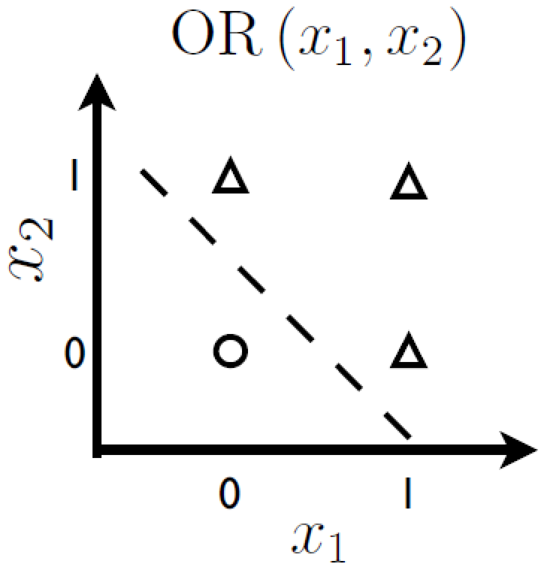


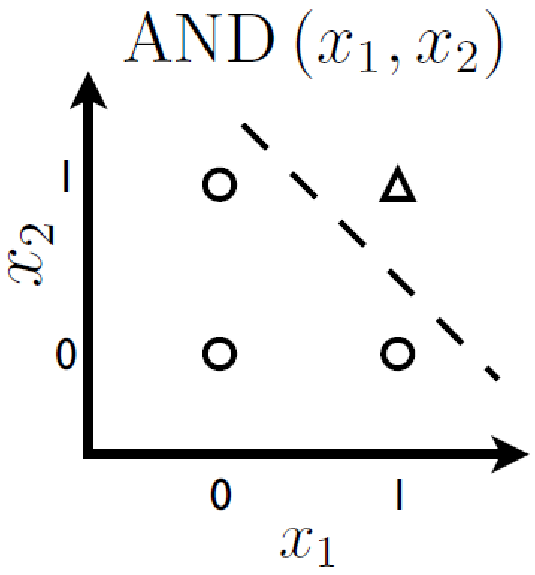
- Aka unit step function
  - $y \ = \ u(x)$
  - $y = u\left(\sum_i^n w_i x_i - b\right) = u\left(\boldsymbol{w^T x} - b\right)$
  - b is threshold
- Inspired by the biological synapse “firing”
- “Linear classifier”
  - Visualize the 1-D binary threshold and 2-D binary threshold.


$\vec{w}.\vec{x} - b = 0$
x_2
$y = 1$
$y = 0$
$\vec{w}$
x_1

Perceptrons as Gates

- Early works focused on “neuronal” building blocks to logic circuit: AND, OR, XOR gates
  - Binary thresholds
  - Not very interesting





Perceptrons as Logic Expressions

- how to create a logic expression using perceptrons
- Binary thresholds
- Algorithms/heuristics for updating weights

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$

Backprop Interest

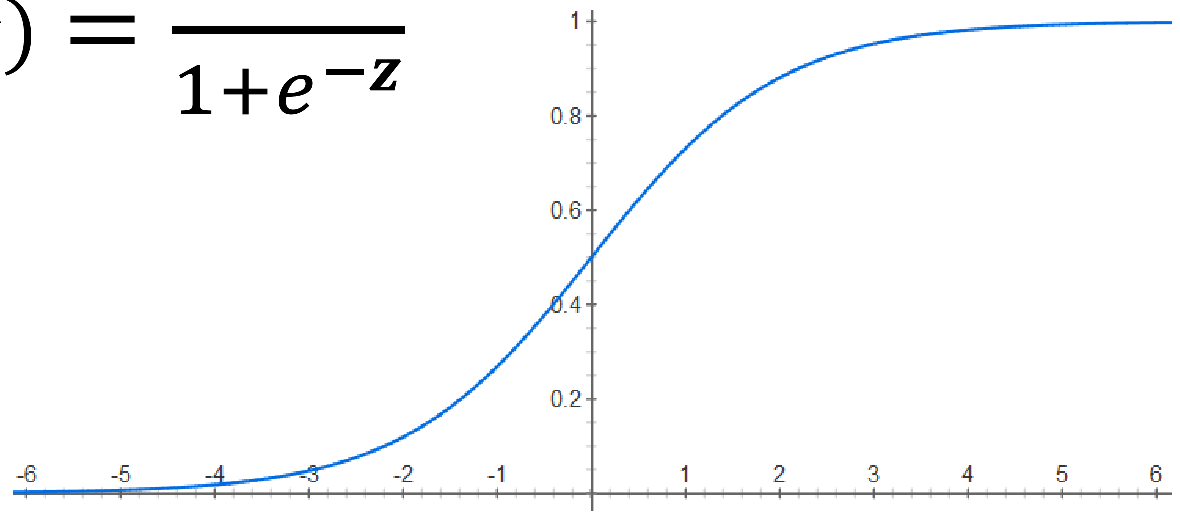
- Given sufficient parameters, nnets can represent any function
- Problem: how to learn parameters
- Binary activation: Simple Biologically inspired
- Problems: Ineffective learning of parameters
- Solution: Rumelhart, Hinton and Williams (1986)
  - use continuous activations functions
  - Apply delta rule to Y
  - Backprop: Chain rule delta learning

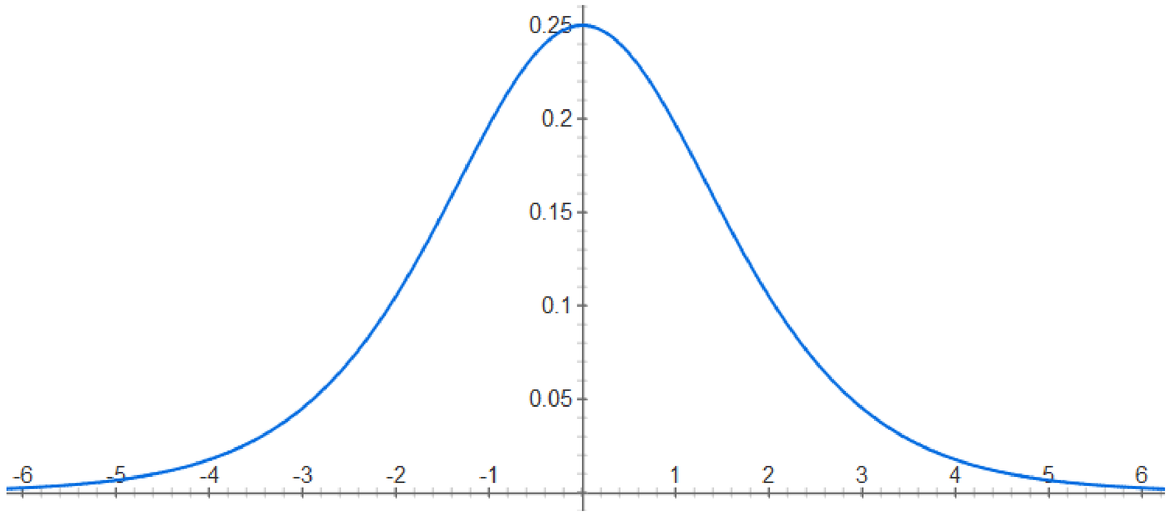
Sigmoid: $g(z) = \dfrac{1}{1+e^{-z}}$

- Relationship to step function:

$$u(x) = \lim_{k\to\infty}\left(\frac{1}{1+e^{-kz}}\right)$$

- Probability driven
- Non-linear
- Derivative: $g'(z) = g(z)(1-g(z))$



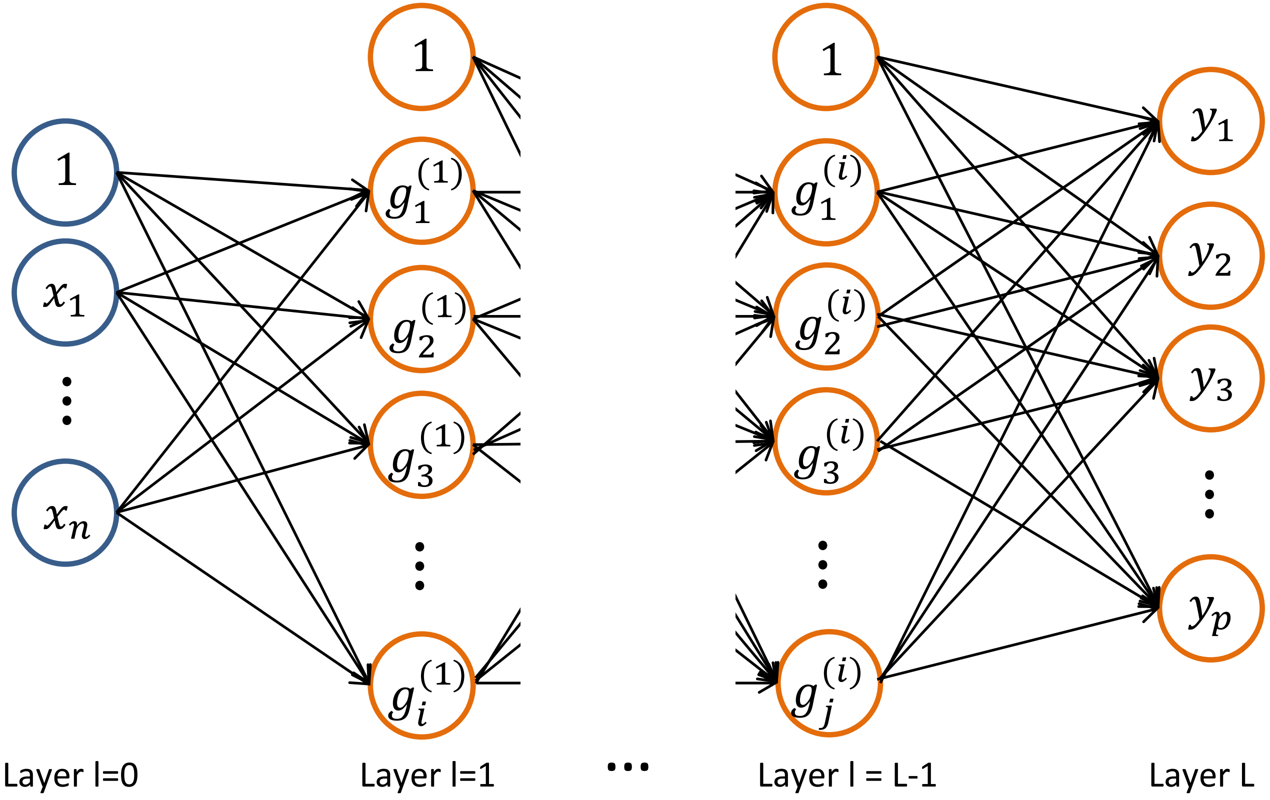


Multi-Layer Feedforward

- AKA directed acyclic graph
- AKA feedforward network
- Depth is L

$$f(\boldsymbol{X}) = f^{(L)}\left(f^{(i)}\left(\dots\left(f^{(2)}\left(f^{(1)}(\boldsymbol{X})\right)\right)\right)\right)$$

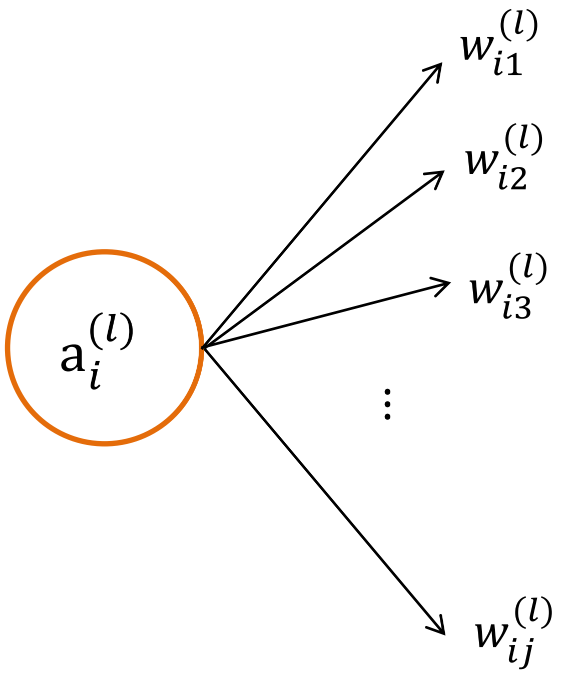
Input: $\boldsymbol{X} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, $g_i^{(l)}$ is the sigmoid function



Feed-forward

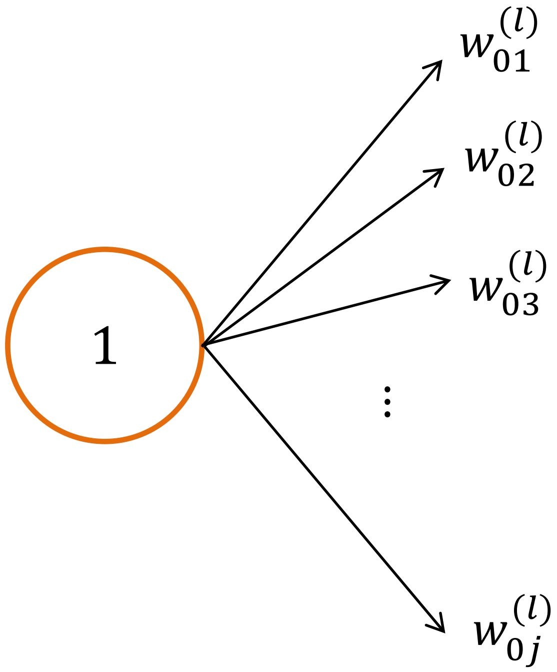
- Layer l : column in network
- neuron i : row number
- j : row number of neuron in next layer
- $w_{ij}^{(l)}$: weight of connection

  between neuron i in layer l and neuron j in layer $l+1$



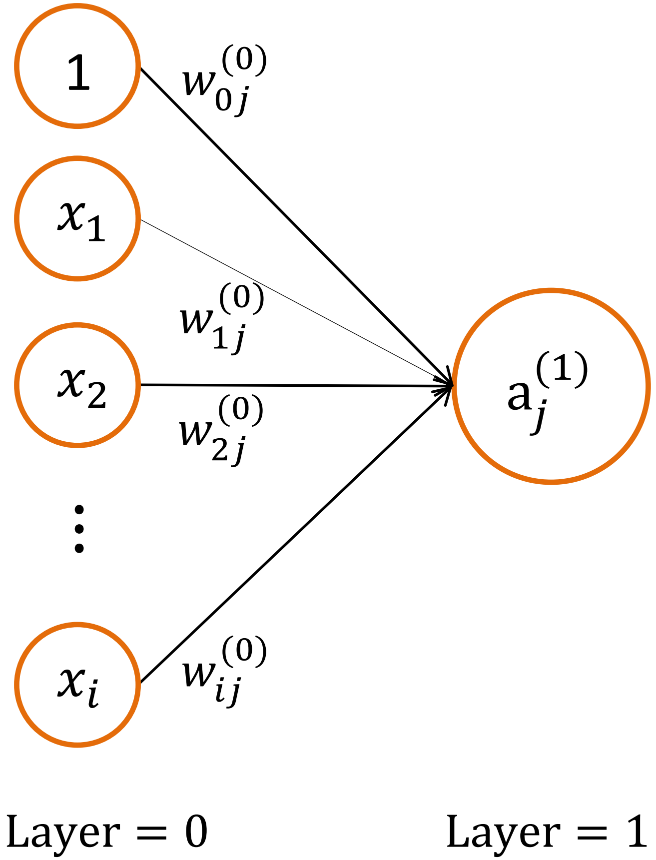
Bias

- In each layer, bias is modeled as a neuron
  - except the output
- $b = w_{0j}^{(l)}$: bias of neuron j in layer $l+1$

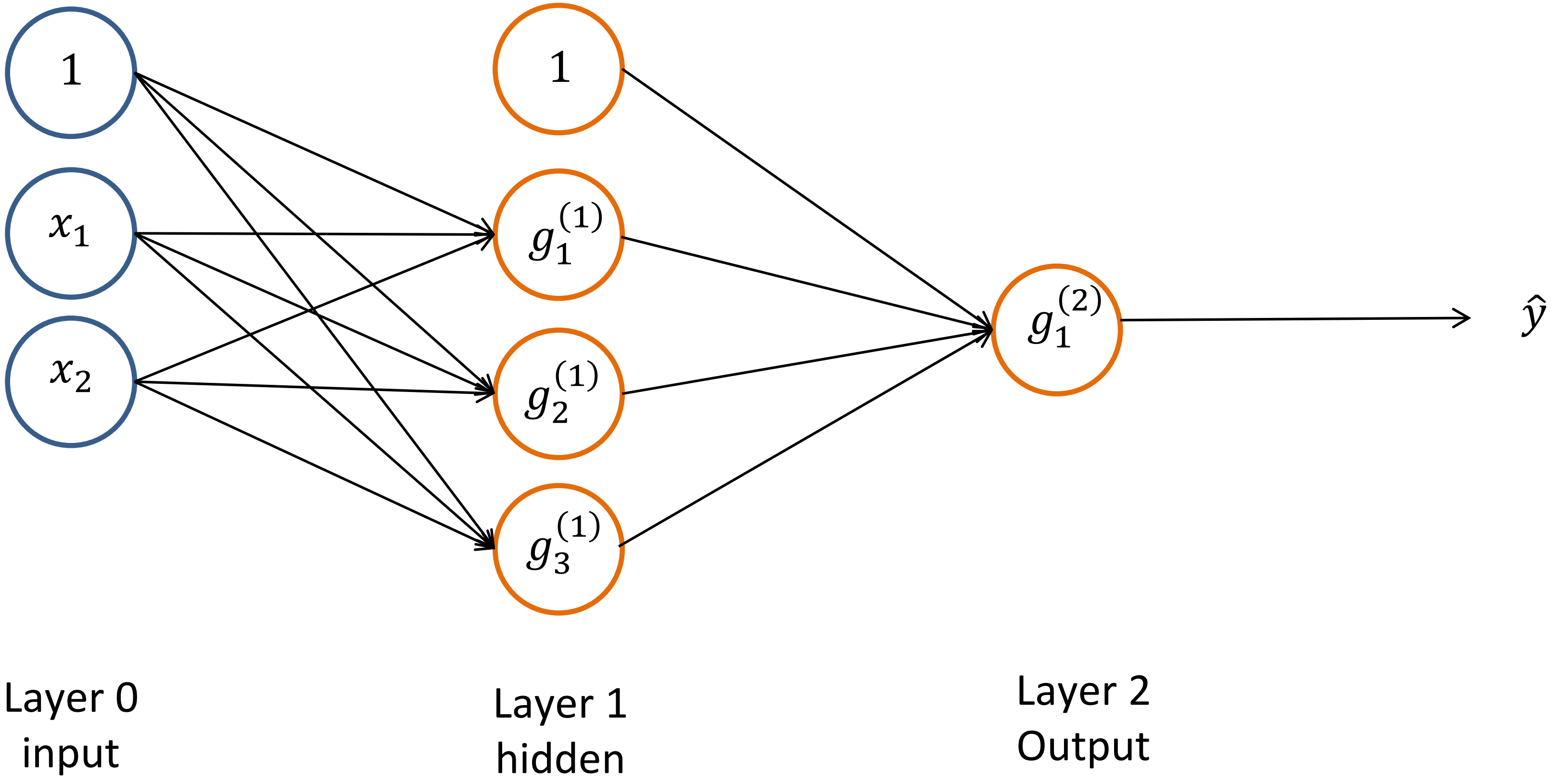


Feed-forward: Layer 1

- $a_j^{(1)}$: activation function in layer 1, row *j*
- $a_j^{(1)} = g\left(\boldsymbol{Z}_{\boldsymbol{j}}^{(\mathbf{1})}\right)$
- $g(\quad)$: sigmoid activation function
- $\boldsymbol{Z}_{\boldsymbol{j}}^{(\mathbf{1})} = \left(\boldsymbol{W}_{\boldsymbol{ij}}^{(\mathbf{0})}\right)^{\boldsymbol{T}} \boldsymbol{X}_{\boldsymbol{i}} = w_{0j}^{(0)} + x_1 w_{1j}^{(0)} + x_1 w_{2j}^{(0)} + \cdots + x_i w_{ij}^{(0)}$



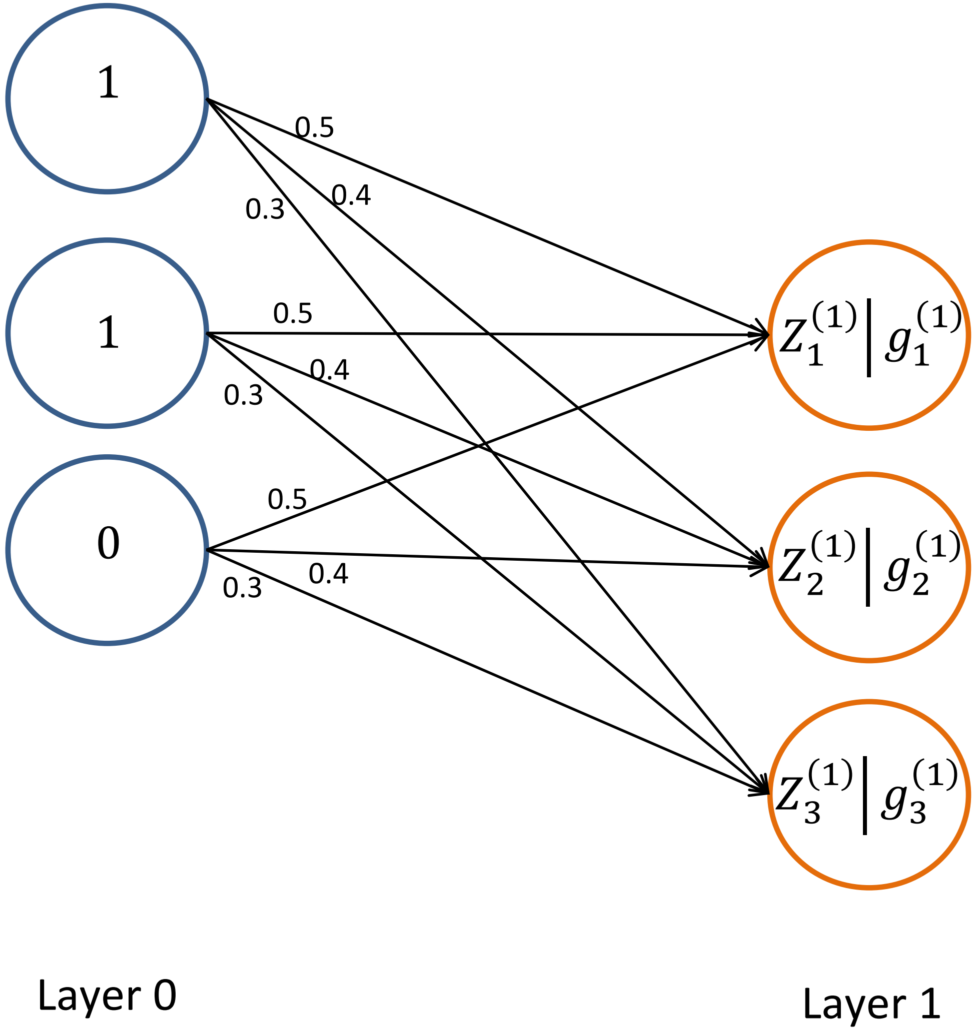
Example Feedforward



Example Feedforward: L0 to L1

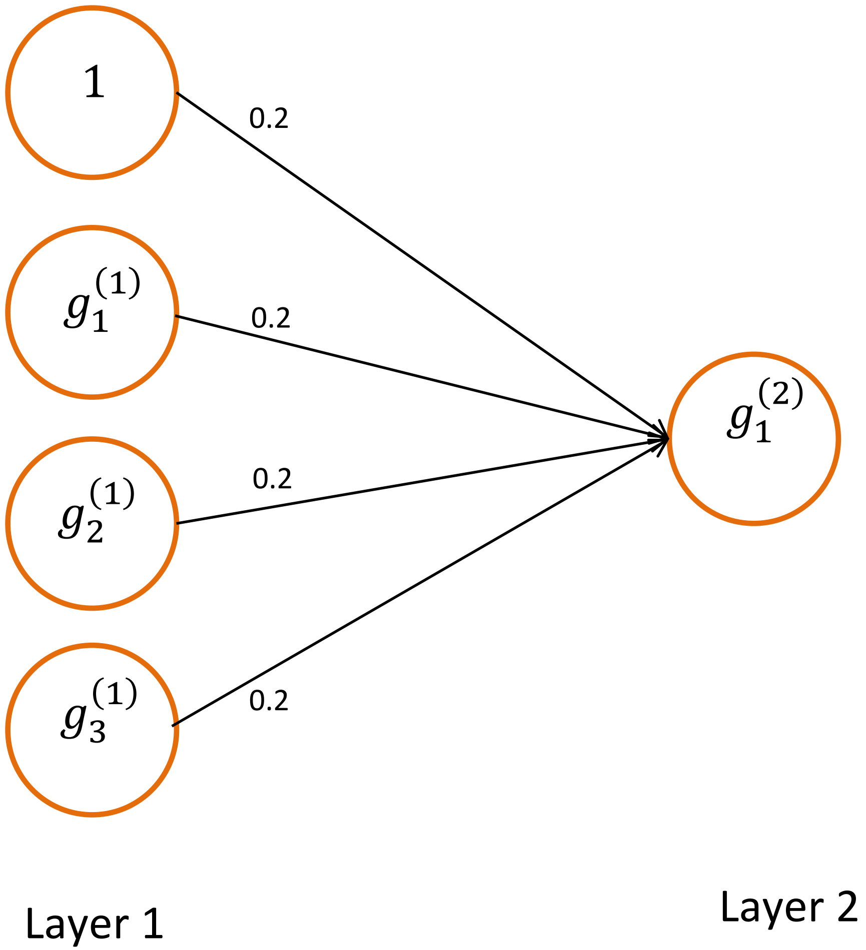
- Weights initialized
  - $w_{01}^{(0)} = 0.5; w_{02}^{(0)} = 0.4; w_{03}^{(0)} = 0.3$
  - $w_{11}^{(0)} = 0.5; w_{12}^{(0)} = 0.4; w_{13}^{(0)} = 0.3$
  - $w_{21}^{(0)} = 0.5; w_{23}^{(0)} = 0.4; w_{33}^{(0)} = 0.3$

- Given a sample: $(x_1, x_2; y) = (1,0;0)$
  - $Z_1^{(1)} = 0.5(0) + 0.5(1) + 0.5 = 1$
  - $Z_2^{(1)} = 0.4(0) + 0.4(1) + 0.4 = 0.8$
  - $Z_3^{(1)} = 0.3(0) + 0.3(1) + 0.3 = 0.6$
  - $g_1^{(1)} = 1/(1 + e^{-1}) = 0.73$
  - $g_2^{(1)} = 1/(1 + e^{-.8}) = 0.69$
  - $g_3^{(1)} = 1/(1 + e^{-.6}) = 0.65$



Example Feedforward: L1 to L2

- Weight matrix: $\boldsymbol{W}^{(1)} = [0.2\ \ 0.2\ \ 0.2\ \ 0.2]$
- $\boldsymbol{G}^{(1)} = [0.73\ \ 0.69\ \ 0.65]$
- $\boldsymbol{Z}^{(2)} = \left(\boldsymbol{W}^{(1)}\right)^T \left[\boldsymbol{b}^{(1)}\ \mid\ \boldsymbol{G}^{(1)}\right] = 0.61$
- $\boldsymbol{G}^{(2)} = g\left(Z^{(2)}\right) = 0.65 = \ \hat{y}$



Feedforward Algorithm

1. $\forall\ i = 0 \rightarrow I, j = 0 \rightarrow J, l = 0 \rightarrow L$: Randomly assign weights $w_{ij}^{(l)}$
2. Initialize: $a_i^{(0)} = x_i$
3. $\forall\ l = 0 \rightarrow L$:

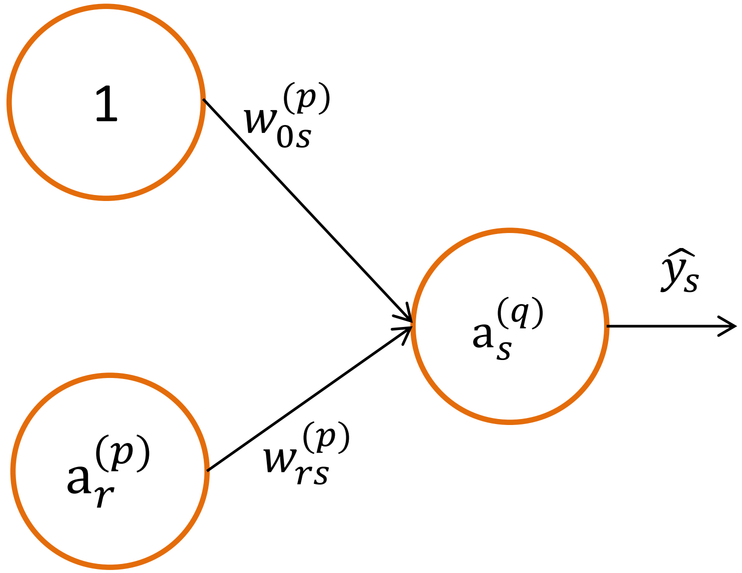
$$\text{compute } \boldsymbol{Z}^{(l+1)} = \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{a}^{(l)}, \quad \boldsymbol{a}^{(l+1)} = g\left(\boldsymbol{Z}^{(l+1)}\right)$$

Error (Delta)

$$Error = E_s = \frac{1}{2}(\widehat{y_s} - y_s)^2 = \frac{1}{2}\delta_s^2$$

$$\widehat{y_s} = a_s^{(q)} = g\left(Z_s^{(q)}\right)$$

$$g\left(Z_s^{(q)}\right) = g\left(w_{0s}^{(p)} + w_{1s}^{(p)}a_1^{(p)} + \cdots + w_{rs}^{(p)}a_r^{(p)} + \cdots\right)$$



Delta Rule

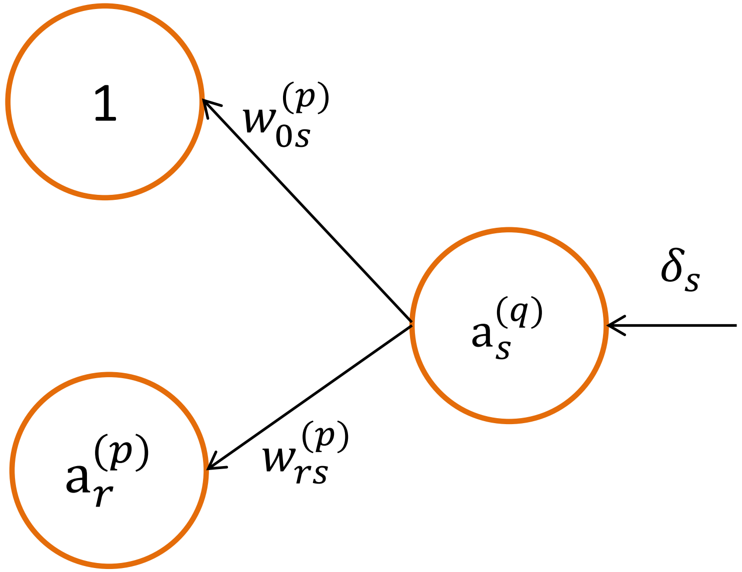
- Find $\frac{\partial E}{\partial w}$:

$$E_s = \frac{1}{2}(\hat{y}_s - y_s)^2 = \frac{1}{2}\delta_s^2$$

$$\hat{y}_s = a_s^{(q)} = g\left(Z_s^{(q)}\right)$$

$$g\left(Z_s^{(q)}\right) = g\left(w_{0s}^{(p)} + w_{0s}^{(p)} a_1^{(p)} + \cdots + w_{rs}^{(p)} a_r^{(p)} + \cdots\right)$$

$$\frac{\partial E_s}{\partial w_{rs}^{(p)}} = \frac{\partial E_s}{\partial \delta_s} \frac{\partial \delta_s}{\partial \hat{y}_s} \frac{\partial \hat{y}_s}{\partial Z_s^{(q)}} \frac{\partial Z_s^{(q)}}{\partial w_{rs}^{(p)}}$$



Chain Rule

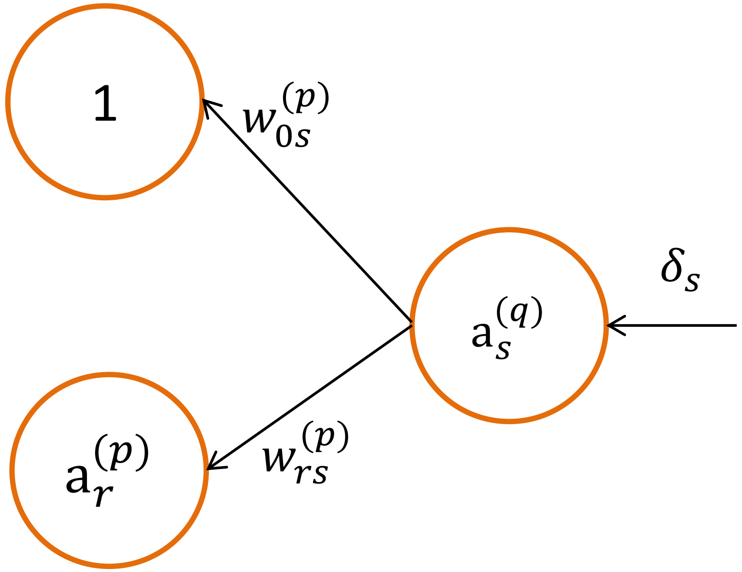
$$E_s = \frac{1}{2}\delta_s^2 \quad \Rightarrow \quad \frac{\partial E_s}{\partial \delta_s} = \delta_s$$

$$\delta_s = \widehat{y_s} - y_s \quad \Rightarrow \quad \frac{\partial \delta_s}{\partial \widehat{y_s}} = 1$$

$$\widehat{y_s} = g\left(Z_s^{(q)}\right) \Rightarrow \quad \frac{\partial \widehat{y_s}}{\partial Z_s^{(q)}} = g'\left(Z_s^{(q)}\right)$$

$$Z_s^{(q)} = w_{0s}^{(p)} + w_{1s}^{(p)} a_1^{(p)} + \cdots + w_{rs}^{(p)} a_r^{(p)} + \cdots \quad \Rightarrow \quad \frac{\partial Z_s^{(q)}}{\partial w_{rs}^{(p)}} = a_r^{(p)}$$

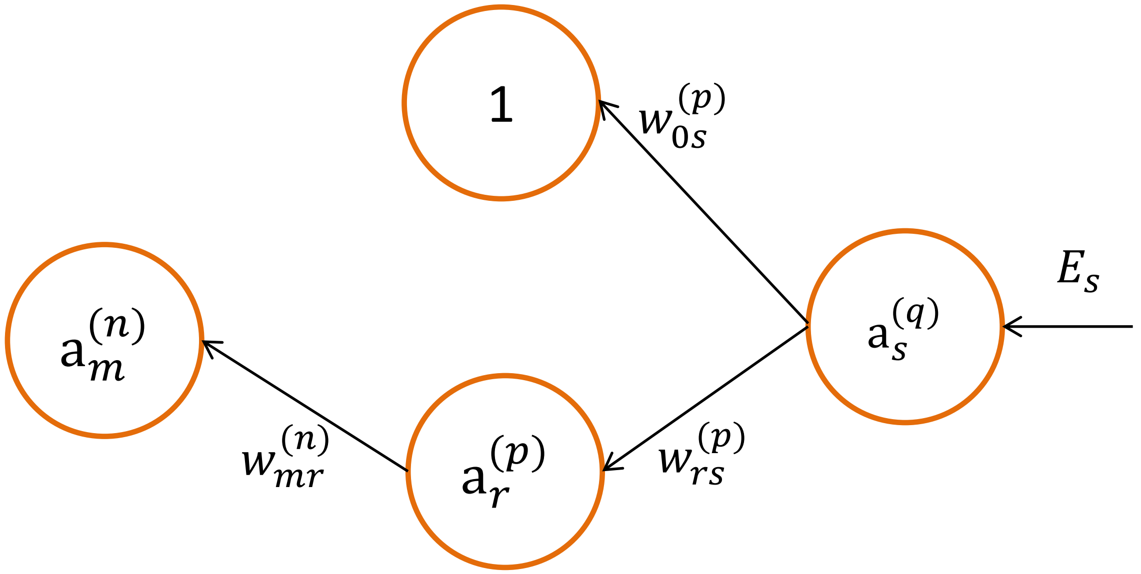
$$\frac{\partial E_s}{\partial w_{rs}^{(p)}} = \frac{\partial E_s}{\partial \delta_s} \frac{\partial \delta_s}{\partial \widehat{y_s}} \frac{\partial \widehat{y_s}}{\partial Z_s^{(q)}} \frac{\partial Z_s^{(q)}}{\partial w_{rs}^{(p)}} = \delta_s \; g'\left(Z_s^{(q)}\right) \; g\left(Z_r^{(p)}\right)$$

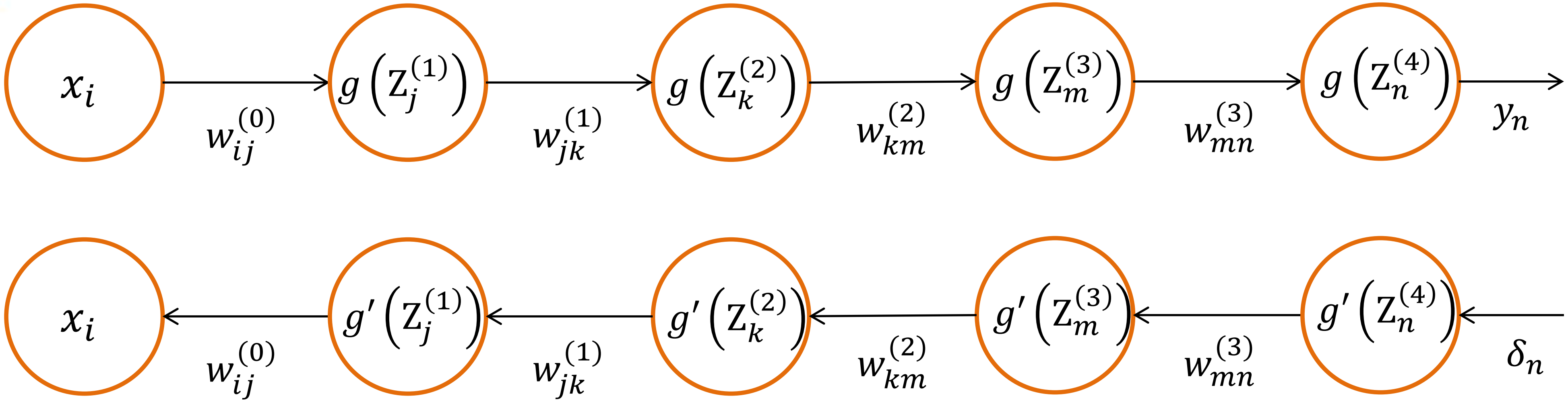


Chain Rule

$$\frac{\partial E_s}{\partial w_{mr}^{(p)}} = \left(\frac{\partial E_s}{\partial \delta_s}\ \frac{\partial \delta_s}{\partial \widehat{y_s}}\ \frac{\partial \widehat{y_s}}{\partial Z_s^{(q)}}\ \frac{\partial Z_s^{(q)}}{\partial a_r^{(p)}}\right) \frac{\partial a_r^{(p)}}{\partial Z_r^{(p)}} \frac{\partial Z_r^{(p)}}{\partial w_{mr}^{(p)}}$$

$$\frac{\partial E_s}{\partial w_{mr}^{(p)}} = \left(\delta\ g'\left(Z_s^{(q)}\right)\ w_{rs}^{(p)}\right) g'^{\left(Z_r^{(p)}\right)} g\left(a_m^{(n)}\right)$$





$$\frac{\partial E_q}{\partial w_{ij}^{(0)}} = \delta_n \, g'\left(Z_n^{(4)}\right) \; w_{mn}^{(3)} g'\left(Z_m^{(3)}\right) \; w_{km}^{(2)} \; g'\left(Z_k^{(2)}\right) \; w_{jk}^{(1)} \; g'\left(Z_j^{(1)}\right) \; x_i$$

Chain Rule Vectorized

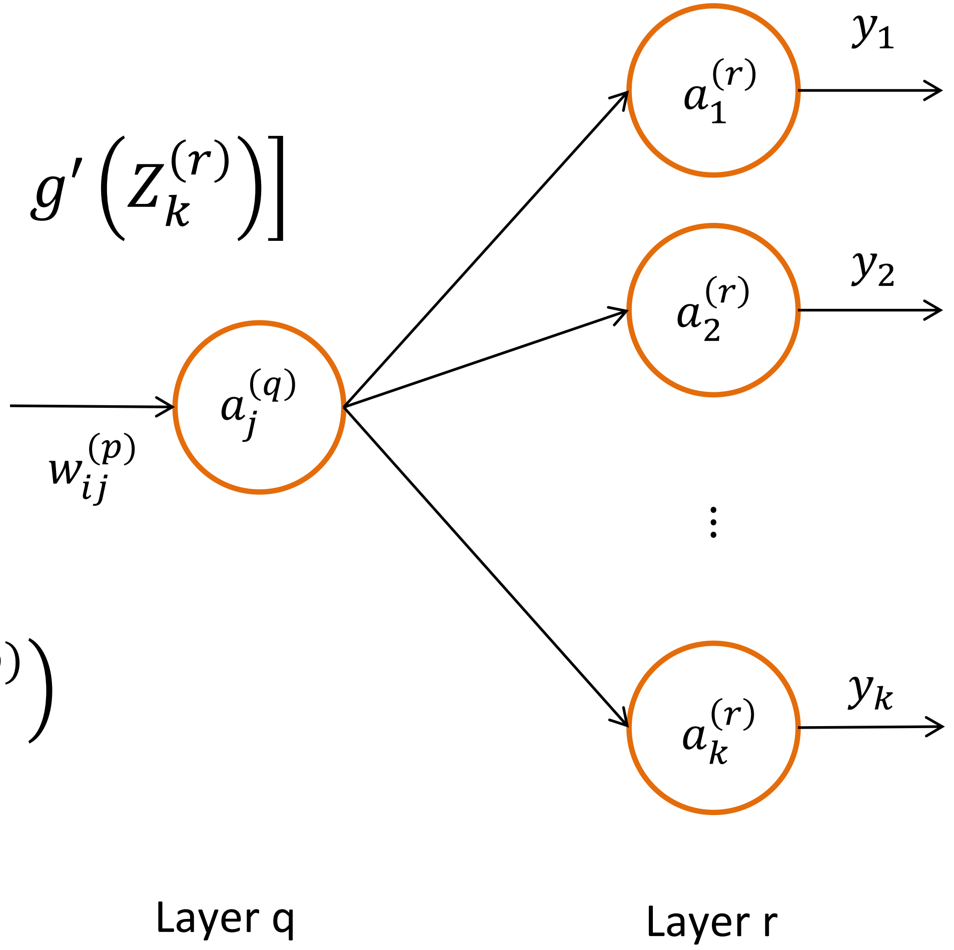
$$a_1^{(r)} = g\left(Z_1^{(r)}\right)$$

$$\boldsymbol{G}'^{(r)} = g'(\boldsymbol{Z}^{(r)}) = \left[g'\left(Z_1^{(r)}\right) \quad g'\left(Z_2^{(r)}\right) \; \dots \; g'\left(Z_k^{(r)}\right)\right]$$

$$W_j^{(q)} = \left[w_{j1}^{(q)} \quad w_{j2}^{(q)} \quad \dots \quad w_{jk}^{(q)}\right]$$

$$\boldsymbol{E} = \frac{1}{2}\left(\widehat{\boldsymbol{Y}} - \boldsymbol{Y}\right)^2 = \frac{1}{2}\boldsymbol{\Delta}^2$$

$$\frac{\partial \boldsymbol{E}}{\partial w_{ij}^{(p)}} = \left(\boldsymbol{\Delta} \circ \boldsymbol{G}'^{(r)}\left(W_j^{(q)}\right)^T\right) g'(Z_j^q)\, g\left(Z_i^{(p)}\right)$$



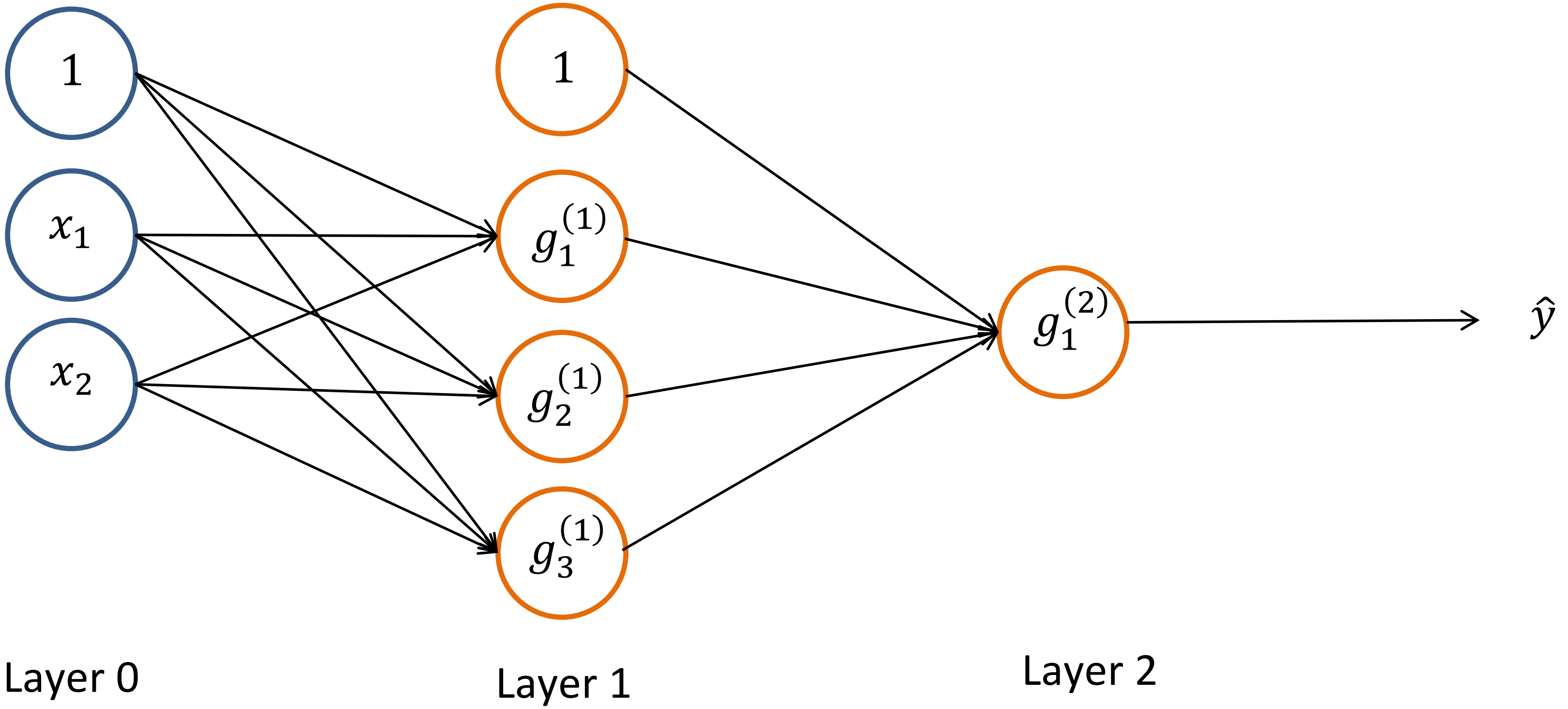
Backprop Algorithm

- Given a weight matrix, $\boldsymbol{W}^{(l)}$ and $\boldsymbol{Z}^{(l)}$ computed for each layer l in feedforward; and Error vector $\boldsymbol{\Delta}$
  1. $\forall$ samples $\left(\boldsymbol{X}^{(m)}, \boldsymbol{Y}^{(m)}\right)$: Feedforward to generate Δ
  2. Initialize $W^{(L)} = \Delta$
  3. $\forall\, l = L \rightarrow 1$:
  4. $\forall\, i, j$:

$$\frac{\partial \boldsymbol{E}}{\partial w_{ij}^{(l)}} = W_j^{(l)} \quad g'\left(Z_j^{(l)}\right) \quad g\left(a_i^{(l-1)}\right)$$

$$w_{ij}^{(l)} = w_{ij}^{(l)} + \alpha \frac{1}{m} \frac{\partial \boldsymbol{E}}{\partial w_{ij}^{(l)}}$$

Backprop Example



- Given a sample: $(x_1, x_2; y) = (1{,}0; 0)$
- Feedforward: $\hat{y} = 0.65$

Example Backprop: Delta

$$\Delta_1 = (0.65 - 1) = -0.45$$

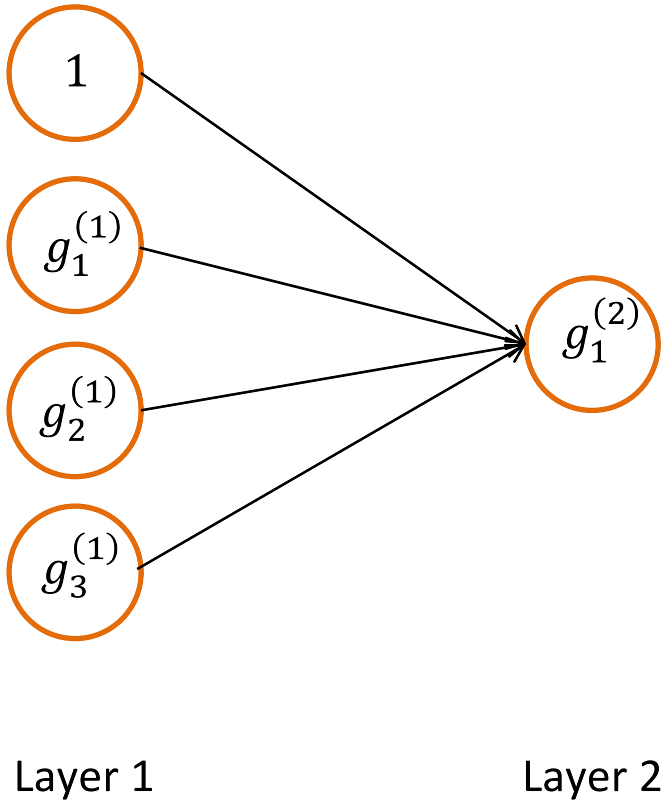
$$g_1^{(2)} \longrightarrow \hat{y}$$

Layer 2

Example Backprop: weights of L1

$$\boldsymbol{Z}^{(2)} = 0.65 \quad \Rightarrow \quad \boldsymbol{G}'^{(2)} = g'(0.65) = 0.23$$

$$\begin{aligned}\frac{\partial E}{\partial W^{(1)}} &= \left[1 \quad \boldsymbol{G}^{(1)}\right]^T \circ \boldsymbol{G}'^{(2)} \circ \boldsymbol{\Delta} \\ &= [1 \quad 0.73 \;\; 0.69 \;\; 0.65]^T \circ 0.23 \circ (-0.44) \\ &= [-0.1 \quad -0.074 \quad -0.070 \quad -.066]\end{aligned}$$



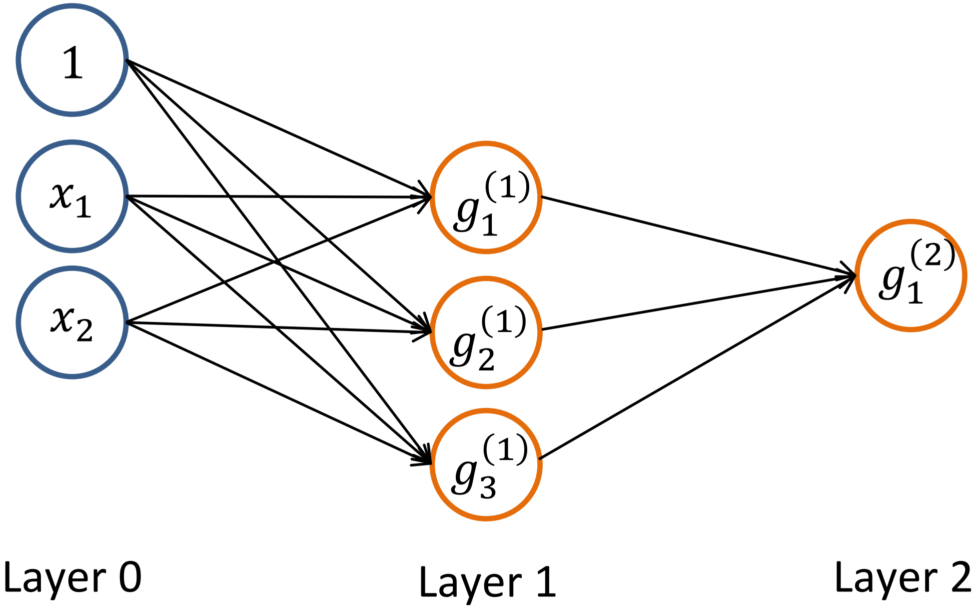
Example Backprop: weights of L0

$$\boldsymbol{Z}^{(1)} = [1 \;\; 0.8 \;\;\; 0.6] \;\Rightarrow\; \boldsymbol{G'}^{(1)} = [0.2 \;\;\; 0.21 \;\;\; 0.23]$$

$$\frac{\partial E}{\partial W^{(0)}} = [1\; X] \circ \boldsymbol{G'}^{(1)}\; \boldsymbol{W}^{(1)} \circ \boldsymbol{G'}^{(2)} \circ \boldsymbol{\Delta}$$

$$\frac{\partial E}{\partial W^{(0)}} = [1 \;\; 1 \;\; 0] \circ \; [0.20 \;\;\; 0.21 \;\;\; 0.23] \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \end{bmatrix} \circ 0.23 \circ -0.44$$

$$\frac{\partial E}{\partial W^{(0)}} = \begin{bmatrix} -.004 & -.0043 & -.0047 \\ -.004 & -.0043 & -.0047 \\ 0 & 0 & 0 \end{bmatrix}$$



Backprop vs Perceptron Learning

- Mathematical no biologically inspired
- Perceptron Learning
  - Biological inspired
  - High Variance: one instance can greatly influence it
    - Obsessed with perfection
  - Fails for complex non-separable dataset
- Backprop
  - Mathematically inspired
  - Works for all data
  - High Bias: new training instance has little effect
    - Consistency over perfection