Classroom **PROCEDURES**

- Get vaccinated
- Provide vaccination proof
- Do the daily COVID screen
- Don't attend when ill
- Wear a mask
- Leave room promptly
- Wash hands frequently
- Don't consume drinks/food

QUartsci.com/Fall2021

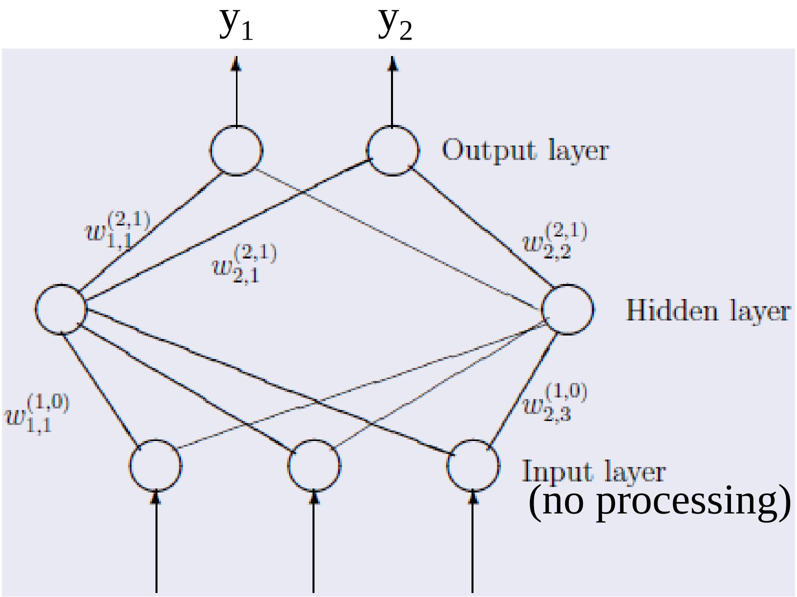
CISC/CMPE452/COGS400
Backpropagation NN

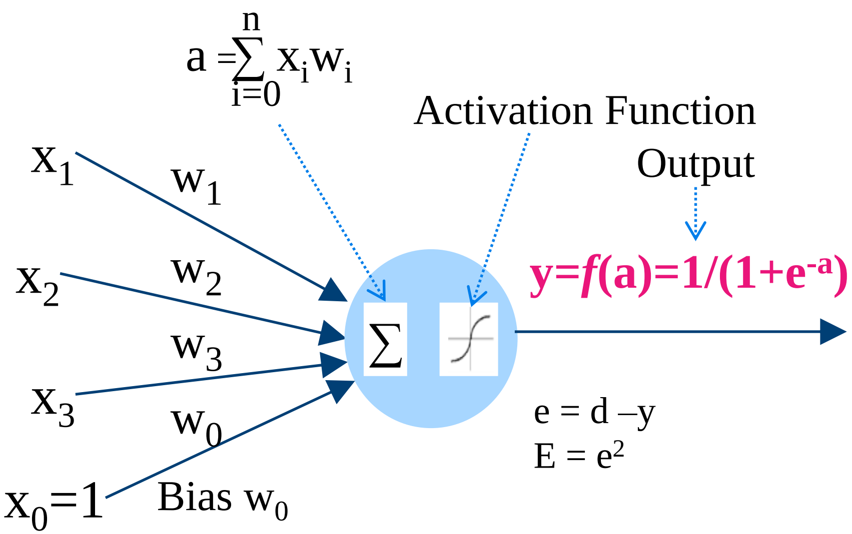
Ch. 3 Text book

Farhana Zulkernine

Supervised Learning using Backpropagation

- Consider a multilayer network with output (layer 2), hidden (layer 1), and input layers (layer 0).
- Reduce MSE (mean squared error).





Goal for Weight Adjustment

- Initially, the weights are assigned random values.
- Our goal will be to find a set of weights and weight adjustment strategy that minimize the *Sum Square Error*, E, for all *P* training data points and *m* output nodes where

$$E = \sum_{k=1}^{P} \sum_{j=1}^{m} E(y_{kj}, d_{kj}) = \sum_{k=1}^{P} \sum_{j=1}^{m} (y_{kj} - d_{kj})^2$$

- E is the Loss Function here. We use *gradient descent optimization* of the ANN model.
- We use Sigmoid activation function.

1. Adjust weights leading to output

- For all layers, we need to apply adjustments to each weight $w^{ll'}_{ji}$, which is the weight from node i in layer l' to node j in layer l.
- **Weights w_{jh} of links from hidden to output layer directly affects error at output node j for all connected nodes h in the hidden layer.**
- So, adjustment to w_{jh} can be calculated using gradient descent

$$\Delta w_{jh} = -\mathrm{c}.\partial E_j / \partial w_{jh}$$

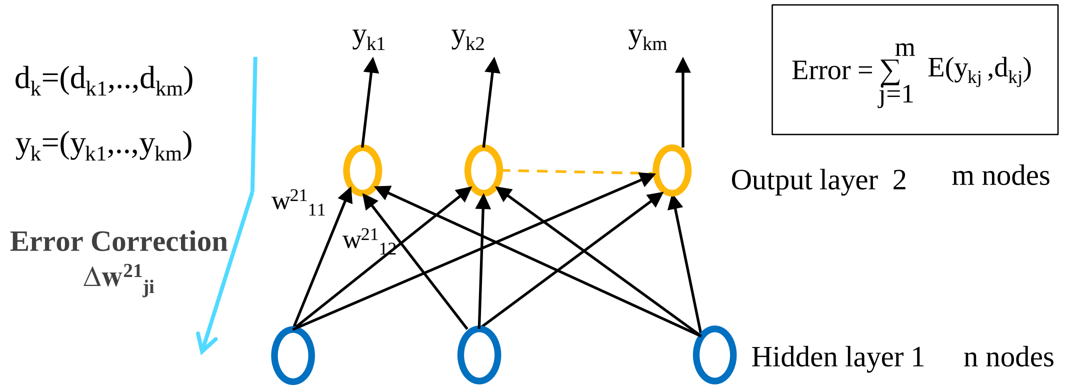
- Where c is learning rate for gradient descent

Weight Adjustment for Δw_{jh}^{21}

- Adjustment for weight w_{jh} from node h in hidden layer to node j in output layer

$\Delta w_{jh} = -c.\partial E_j/\partial w_{jh}$ c is small constant learning rate

1. $E_j = e_j^2 = (d_j - y_j)^2$ where E is total error at node j
2. $y_j = f(a_j) = 1/(1+e^{-a})$ using sigmoid output function
3. $a_j^2 = \sum_h (w_{jh} x_h^l)$ activation at layer 2, x^l_h = input from hidden layer 1 to output layer 2



Calculating Δw_{ji} (hidden to output)

1. $E_j = e_j^2 = (d_j - y_j)^2$ 2. $y_j = f(a_j) = 1/(1+e^{-a})$ 3. $a_j^2 = \sum_i (w_{jh} x_h^1)$

$$\Delta w_{jh} = -c.\frac{\partial E_j}{\partial w_{jh}} = \frac{\partial E_j}{\partial e_j}.\frac{\partial e_j}{\partial yj}.\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial w_{jh}}$$

Applying chain rule

$$= -c.(d_j - y_j).(-1).\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial w_{jh}}$$

$$= c.(d_j - y_j).f'(a_j).\frac{\partial a_j}{\partial w_{jh}}$$

where $f'(a_j) = \frac{\partial y_j}{\partial a_j}$

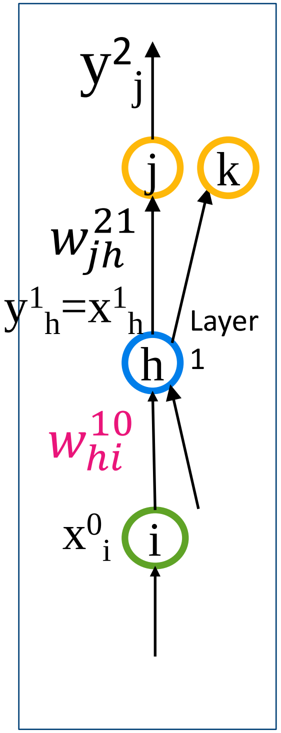
$$= c.(d_j - y_j).f'(a_j).\frac{\partial(w_{j1}x_1 + \cdots + w_{jh}x_h + \cdots + w_{jn}x_n)}{\partial w_{jh}}$$

$$= c.(d_j - y_j).f'(a_j).x_h$$

$$= c.\boldsymbol{\delta_j^o}.x_h \qquad where\ \delta_j^o = (d_j - y_j).f'\left(a_j^2\right) = e_j.f'\left(a_j^2\right)$$

So Δw_{jh} is c.x_h (as before) multiplied by an error term δ_j^o for node *j* in the output layer and (where δ_j^o is error e_j * derivative of output $f'\left(a_j^2\right)$)

Weight Adjustment for Hidden Layer Δw_{hi}^{10}



- A wrong weight w_{hi}^{10} from input node i to one hidden node h affects all the output nodes ($j = 1,\ldots,m$)
- Adjustment for weight w_{hi} therefore must sum up errors at all output nodes when computing error gradient.
- So, using chain rule up to two layers,

$$\Delta w_{hi}^{10} = -\sum_{j=1}^{m} c.\partial E_j / \partial w_{hi}$$

Computing weight to hidden layer Δw_{hi}^{10}

1. $E_j = e_j^2 = (d_j - y_j)^2$ 2. $y_j = f(a_j) = \frac{1}{1+e^{-a}}$ 3. $a_j^2 = \sum_{i=1..n}(w_{ji} x_i^1)$

$$\Delta w_{hi} = -\sum_{j=1}^{m} c.\frac{\partial E_j}{\partial w_{hi}} = -\sum_{j=1}^{m} c.\frac{\partial E_j}{\partial e_j}.\frac{\partial e_j}{\partial y_j}.\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial x_h}.\frac{\partial y_h}{\partial a_h}.\frac{\partial a_h}{\partial w_{hi}} \quad \text{chain rule}$$

$$= -\sum_{j=1}^{m} c.(d_j - y_j).(-1).\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial x_h}.\frac{\partial y_h}{\partial a_h}.\frac{\partial a_h}{\partial w_{hi}} \quad \text{considering } w_{jh} \text{ constant}$$

$$= \sum_{j=1}^{m} c.(d_j - y_j).f'(a_j).\frac{\partial a_j}{\partial x_h}.\frac{\partial y_h}{\partial a_h}.\frac{\partial a_h}{\partial w_{hi}} \quad \text{where } f'(a_j) = \frac{\partial y_j}{\partial a_j}$$

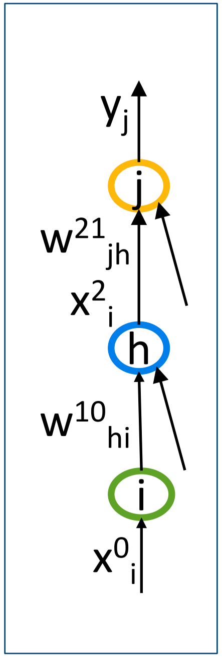
$$= \sum_{j=1}^{m} c.(d_j - y_j).f'(a_j).\frac{\partial(w_{j1}x_1 + \cdots + w_{jh}x_h + \cdots + w_{jn}x_n)}{\partial x_h}.\frac{\partial y_h}{\partial a_h}.\frac{\partial a_h}{\partial w_{hi}}$$

$$= \sum_{j=1}^{m} c.(d_j - y_j).f'(a_j).w_{jh}.f'(a_h).\frac{\partial(w_{h1}x_1 + \cdots + w_{hi}x_i + \cdots + w_{hq}x_n)}{\partial w_{hi}}$$

$$= c.\left\{\sum_{j=1}^{m} \boldsymbol{\delta_j^o}.w_{jh}\right\}.f'(a_h).x_i \quad \textit{where } \delta_j^o = (d_j - y_j^2).f'(a_j^2)$$

$$= \text{c}.\boldsymbol{\delta_h^H}.x_i \quad \textit{where } \delta_h^H = \left\{\sum_j^m \delta_j^o.w_{jh}\right\}.f'(a_h^1)$$

Hidden Layer Δw_{hi}^{10} (cont...)



$$\Delta w_{hi}^{10} = \text{c}.\,\boldsymbol{\delta_h^H}.\,x_i$$

$$\delta_h^H = \left\{ \sum_{j}^{m} \boldsymbol{\delta_j^O}\,.\,w_{jh} \right\}.\,f'\,(a_h)$$

- So Δw_{hi} is c.x$_i$ (as before) multiplied by an error term δ_h^H - sum of weighted error e_j propagated from the output layer to the hidden node h

Compute $f'(a)$

- Output function $y=f(a)$ can be any function but it must be differentiable for Backpropagation
- If $y = f(a) = \frac{1}{1+e^{-a}}$, a sigmoidal function then

$$f'(a) = \frac{e^{-a}}{(1+e^{-a})^2} = \frac{1+e^{-a}-1}{(1+e^{-a})^2} = \frac{1+e^{-a}}{(1+e^{-a})^2} - \frac{1}{(1+e^{-a})^2}$$

$$= \frac{1}{(1+e^{-a})} - \frac{1}{(1+e^{-a})^2} = \frac{1}{(1+e^{-a})}\left[1 - \frac{1}{(1+e^{-a})}\right]$$

$$= y(1-y)$$

Compute Δw

Level 0
input layer

$$\Delta w_{hi}^{10} = \text{c.}\,\delta_h\,.\,x_i$$

$$= \text{c.}\left\{\sum_{j=1}^{m}(d_j - y_j).\,y_j\,.(1-y_j).\,w_{jh}\right\}.\,f'(a_h).\,x_i$$

Level 1
hidden
layer

Level 2
output layer

$$= \text{c.}\left\{\sum_{j=1}^{m}(d_j - y_j).\,y_j\,.(1-y_j).\,w_{jh}\right\}.\,y_h\,.(1-y_h).\,x_i$$

$$\Delta w_{jh}^{21} = \text{c.}\,\delta^{o}\,.\,x_h = \text{c.}(d_j - y_j).\,f'(a_j).\,x_h$$

$$= c.(d_j - y_j).\,y_j\,.(1-y_j).\,x_h$$

Backpropagation 2 Layer Network

1. Start with **randomly chosen weights**;
2. **While** MSE is unsatisfactory and
   computational bounds are not exceeded do
3. **For each input pattern** x_p ,{1 ≤ p ≤ P} do
4. Compute activations at the hidden nodes (a_h^1)
5. Compute outputs from the hidden nodes ($y_h^1 = x_h^2$)
6. Compute activations at the output nodes (a_j^2)
7. Compute the network outputs (y_j^2)
8. Modify weights between input & hidden nodes: Δw_{hi}^{10} first
   Modify weights between hidden & output nodes: Δw_{jh}^{21} next
1. **End for**
2. **End while**