Applied Machine Learning with Big Data
“EE 6973”

Topic:
Deep Learning

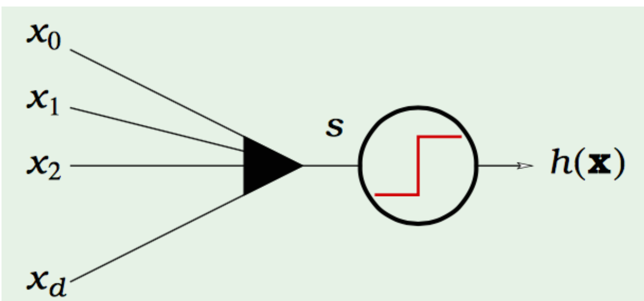
Paul Rad, Ph.D.

Chief Research Officer
UTSA Open Cloud Institute(OCI)
University of Texas at San Antonio

Review Models

Linear Classification

h(x) = Sign ($\sum_{i=0}^{n} w_i \, x_i$)



Sign function:

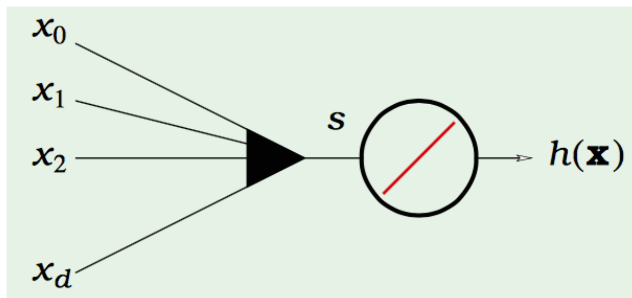
h(s) = 1 s >=0

h(s) = 0 s < 0

Hard Threshold : Certainty

Linear Regression

h(x) = $\sum_{i=0}^{n} w_i \, x_i$

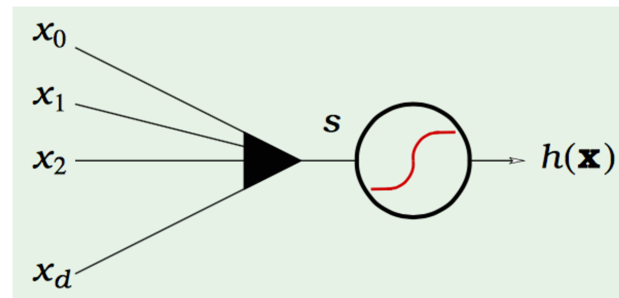


identity function:

h(s) = s

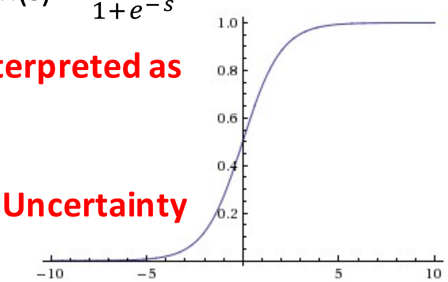
Logistic Regression

h(x) = Sigmoid ($\sum_{i=0}^{n} w_i \, x_i$)



Sigmoid function:

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



The output is interpreted as probability

Soft Threshold : Uncertainty

Probability Interpretation

h(x) = Sigmoid ($\sum_{i=0}^{n} w_i\, x_i$) = θ (s) is interpreted as a probability

Logistic Regression

h(x) = Sigmoid ($\sum_{i=0}^{n} w_i\, x_i$)

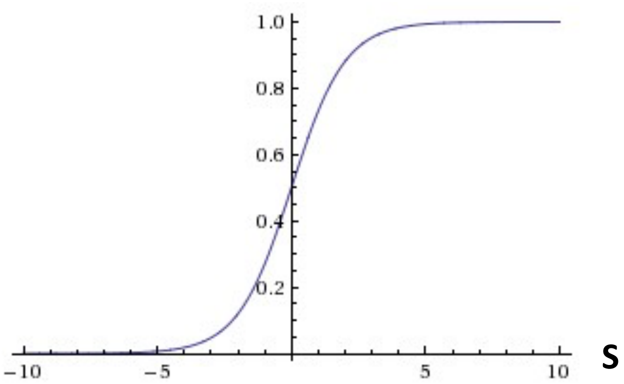
Example: Prediction of heart attacks

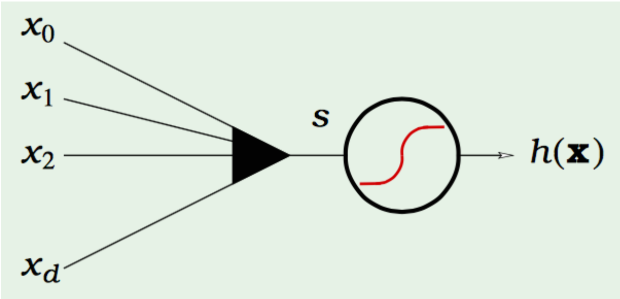
Input **X**: x1 =cholesterol level, x2 =patient age, x3 =patient weight, etc.

θ (s): probability of a heart attack

S= $\sum_{i=0}^{n} w_i\, x_i$ **"risk score"**

θ(s) = $\frac{1}{1+e^{-s}}$





Sigmoid function

Outline

Neural Network Model

Error and Learning

Backpropagation

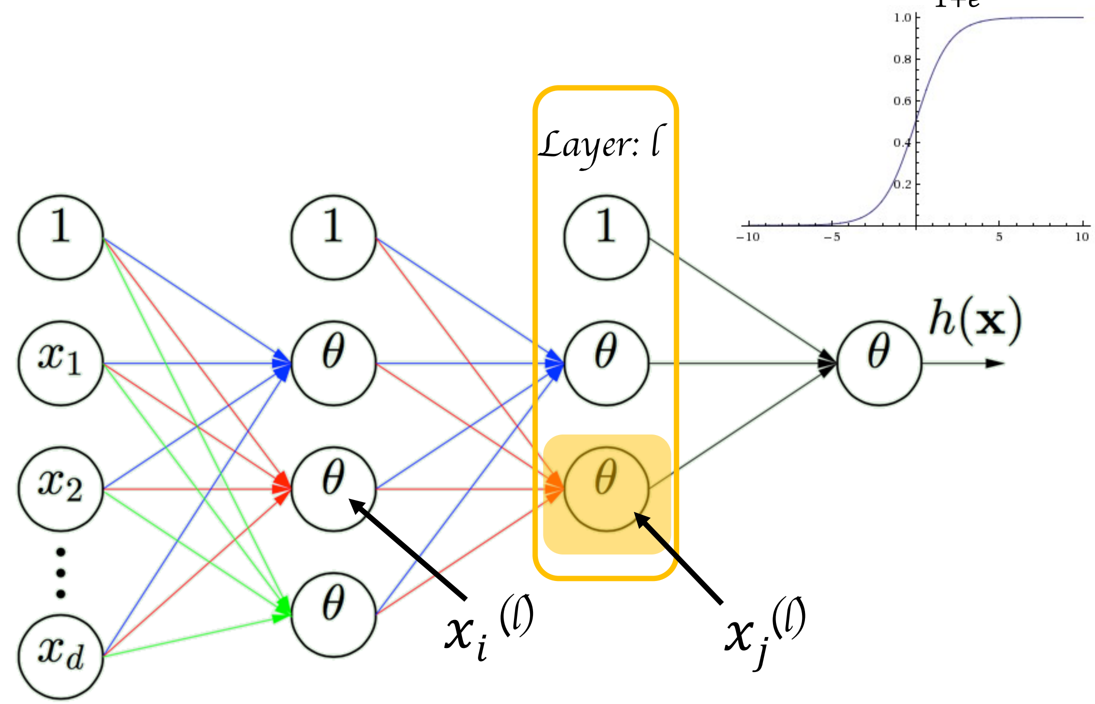
Neural Network Model (forward propagation)

$$
w_{ij}^{(l-1,l)} \begin{cases} 2 =< l =< L & \textit{Layers} \\ 0 =< i =< d^{(l-1)} & \textit{inputs} \\ 1 =< j =< d^{(l)} & \textit{outputs} \end{cases}
$$

$$
x_j^{(l)} = \boldsymbol{\theta}\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l-1,l)} x_i^{(l-1)}\right)
$$

$2 =< l =< L\ ,\ 1 =< j =< d^{(l)}$

$\boldsymbol{\theta}(s) = \frac{1}{1+e^{-s}}$



Input Layer
$l = 1$

Hidden Layers
$1 < l < L$

Output layer
$l = L$

Derivative of Sigmoid Function

$$\frac{ds(x)}{dx} = \frac{1}{1+e^{-x}}$$

$$= \left(\frac{1}{1+e^{-x}}\right)^2 \frac{d}{dx}\left(1+e^{-x}\right)$$

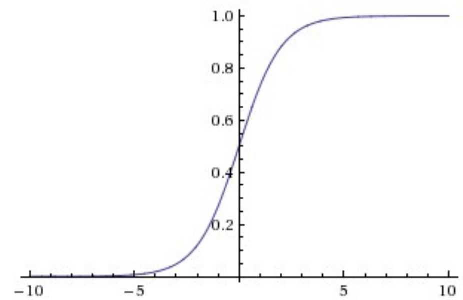
$$= \left(\frac{1}{1+e^{-x}}\right)^2 e^{-x}(-1)$$

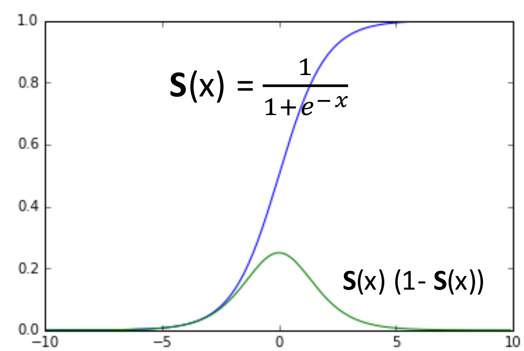
$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)\left(-e^{-x}\right)$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{-e^{-x}}{1+e^{-x}}\right)$$

$$= s(x)(1-s(x))$$

S(x) $= \frac{1}{1+e^{-x}}$





Error

$$x_j^{(l)} = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l-1,l)} x_i^{(l-1)}\right) \qquad h(x) = x_j^{(l)} \text{ when } l = L$$

If we have all the weights $w = \left\{ w_{ij}^{(l-1,l)} \right\}$ *for each layer then we can determine* $h(x)$.

Error on example (X_n, Y_n) *is* $e(h(X_n), Y_n) = e(w)$

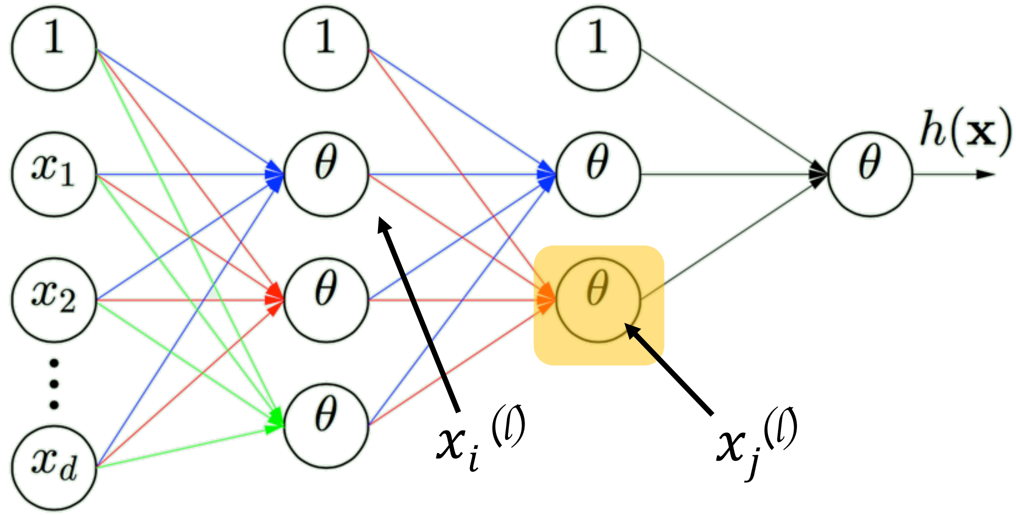
We need to calculate Gradient: $\nabla e(\mathbf{w})$: $\dfrac{\partial\, e(\mathbf{w})}{\partial\, w_{ij}^{(l)}}$ for all i, j, l

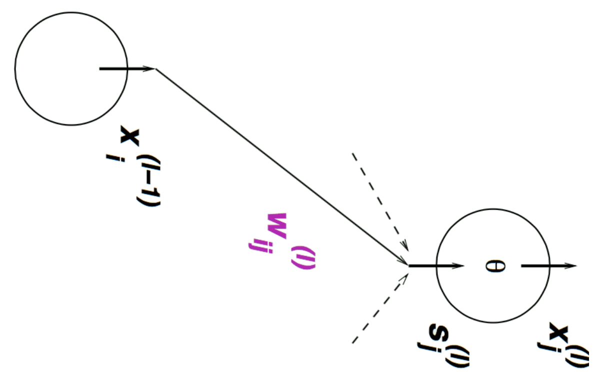
Calculating partial derivatives of $e(\mathcal{W})$

$\nabla \mathrm{e}(\mathbf{w})$: $\dfrac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ w_{ij}^{(l)}}$ for all i, j, l

$$\frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ w_{ij}^{(l)}} = \boxed{\frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ s_j^{(l)}}} \times \underbrace{\boxed{\frac{\partial\ s_j^{(l)}}{\partial\ w_{ij}^{(l)}}}}_{x_i^{(l-1)}}$$

$$\frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ s_j^{(l)}} = \delta_j^{(l)}$$





For the final layer $\delta_j^{(l)}$

$$\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ s_j^{(l)}} = \delta_j^{(l)}$$

For the final layer $l = \mathcal{L}$ and j=1

$$e(w) = (x_1^{(\mathcal{L})} - y)^2 \qquad x_1^{(\mathcal{L})} = \Theta(s_1^{(\mathcal{L})}) \qquad \Theta'(s) = \Theta(x)\,(1 - \Theta(x))$$

$$\frac{\partial \ \mathrm{e}(\mathbf{w})}{\partial \ s_j^{(l)}} = \delta_j^{(l)} \qquad \frac{\partial \ (\Theta(s_1^{(\mathcal{L})}) - y)^2}{\partial \ s_j^{(l)}}$$

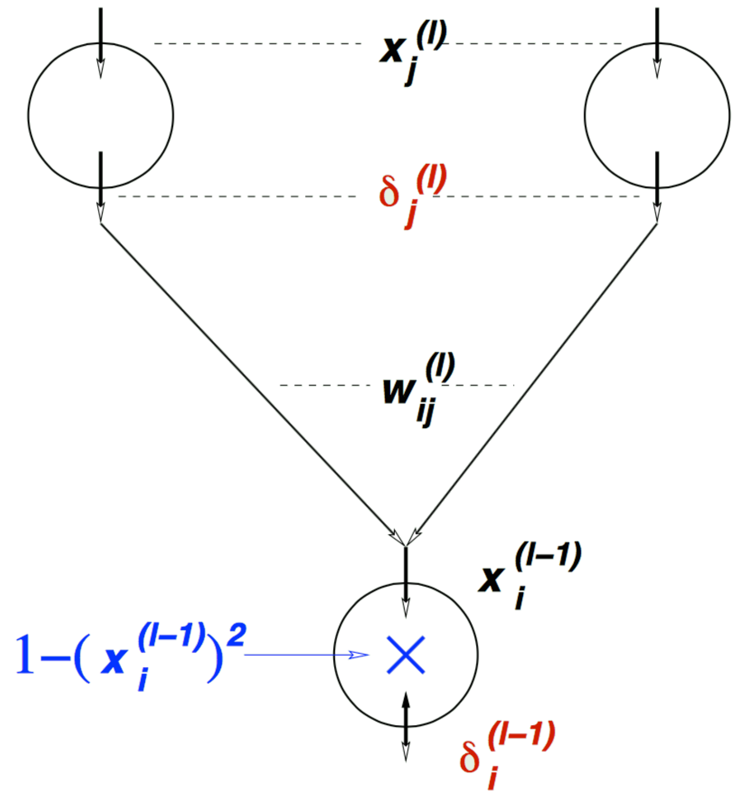
Back propagation of $\delta_j^{(l)}$

$$\delta_i^{(l-1)} = \frac{\partial\ \mathbf{e}(\mathbf{w})}{\partial\ s_i^{(l-1)}}$$

$$= \sum_{j=1}^{d^{(l)}} \frac{\partial\ \mathbf{e}(\mathbf{w})}{\partial\ s_j^{(l)}} \times \frac{\partial\ s_j^{(l)}}{\partial\ x_i^{(l-1)}} \times \frac{\partial\ x_i^{(l-1)}}{\partial\ s_i^{(l-1)}}$$

$$= \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \times \theta'(s_i^{(l-1)})$$

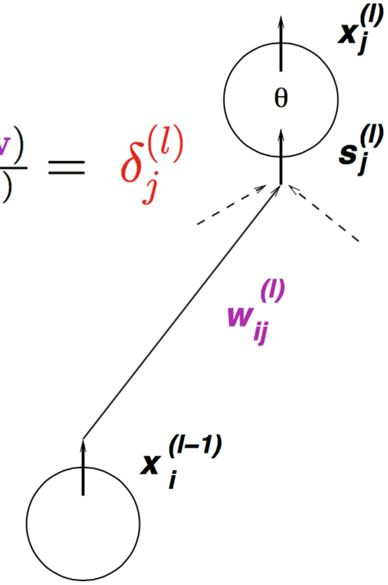
$$\delta_i^{(l-1)} = x_i^{(l-1)}\left(1 - x_i^{(l-1)}\right) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)}\ \delta_j^{(l)}$$



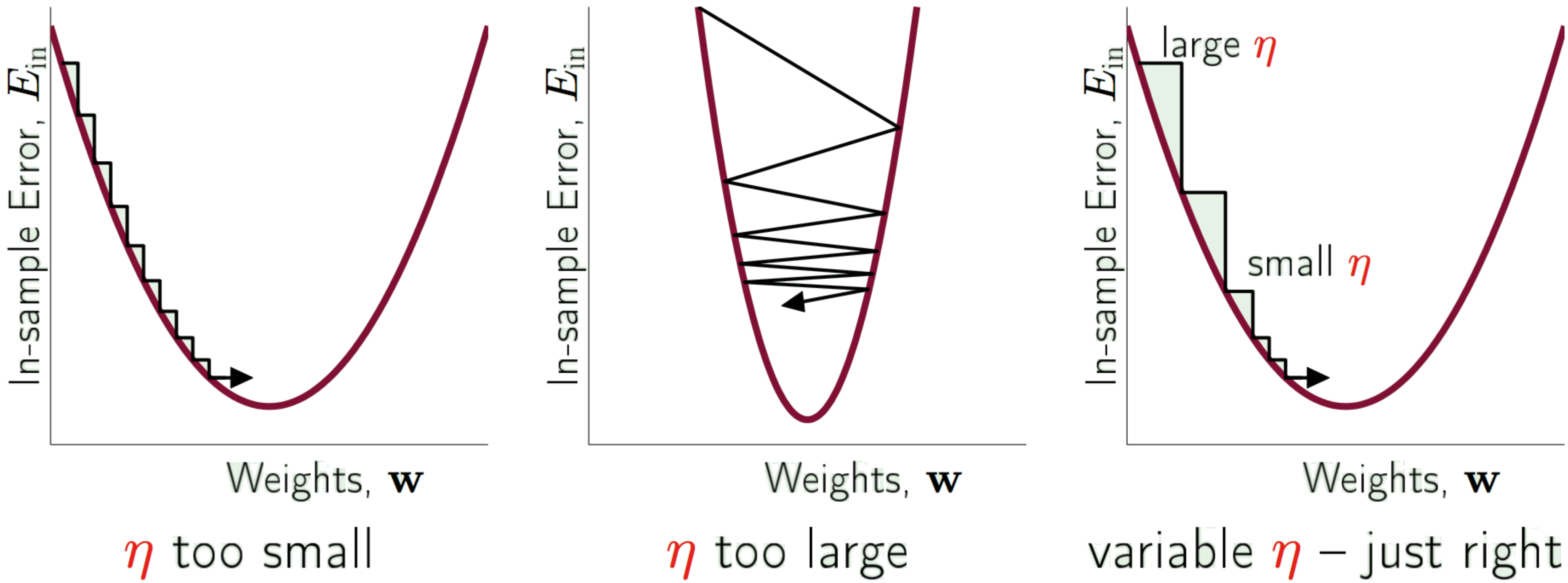
Backpropagation Algorithm

```
Initialize all weights w_ij^(l) at random
for t = 0, 1, 2, ... do
    Pick n ∈ {1, 2, ···, N}
    Forward: Compute all x_j^(l)
    Backward: Compute all δ_j^(l)
    Update the weights: w_ij^(l) ← w_ij^(l) − η x_i^(l−1) δ_j^(l)
    Iterate to the next step until it is time to stop
Return the final weights w_ij^(l)
```

$$\frac{\partial\ \mathrm{e}(\mathbf{w})}{\partial\ s_j^{(l)}} = \delta_j^{(l)}$$



Learning Rate: Steps



Learning Rate should increase with the slope