Code

Neural Networks

Joseph E. Gonzalez

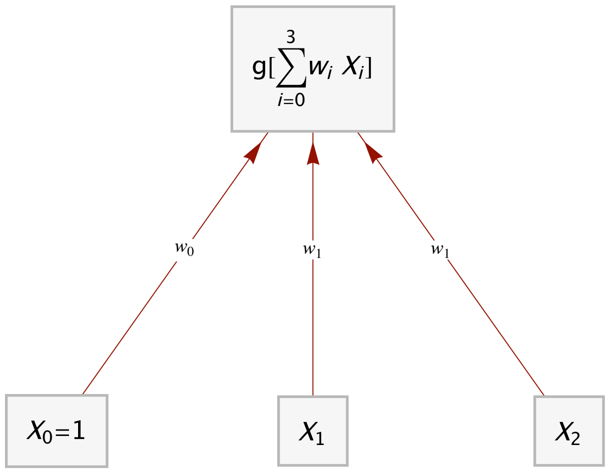
We are going to go through Neural Networks and review the process of back propagation.

Experimental *Mathematica* based presentation.

Single Perceptron

- *The Perceptron*

```
perceptronPlot
```



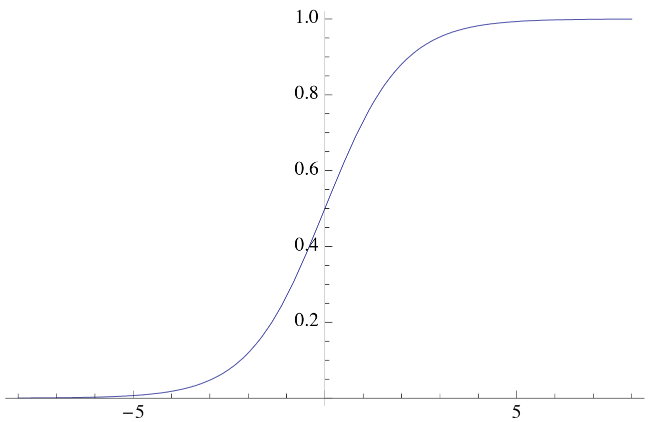
- *There are several parts*

1. Link Function $g[u]$
2. Weights w_i
3. A bias term X_0

Link Function

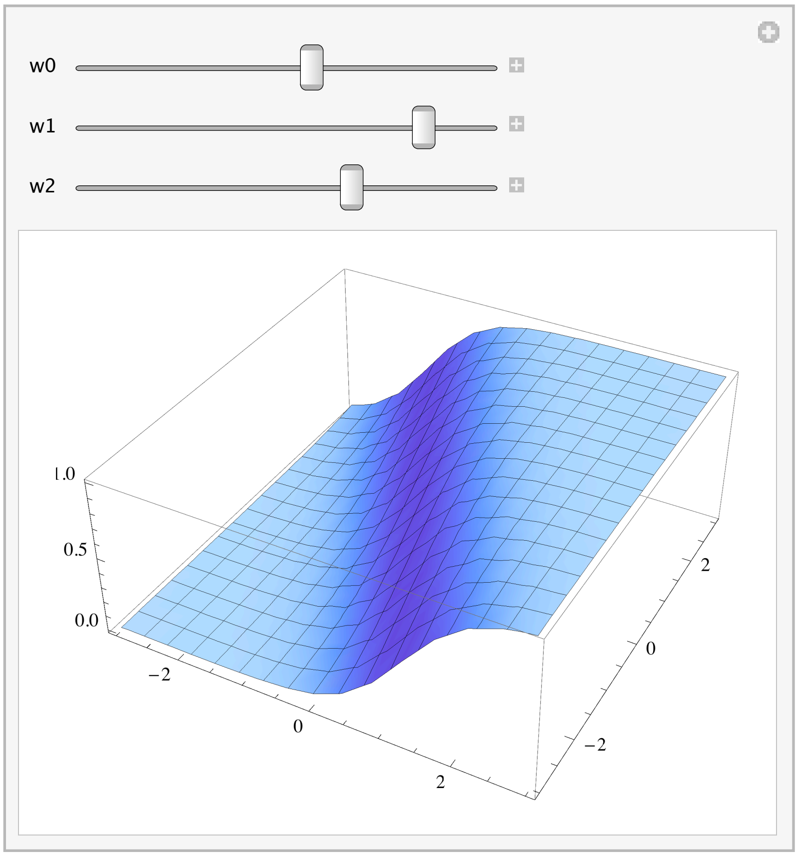
$$g[x] = \frac{1}{1 + e^{-x}} \tag{1}$$

```
g = Function[x, 1/(1 + Exp[-x])];
Plot[g[x], {x, -8, 8}]
```



Demo

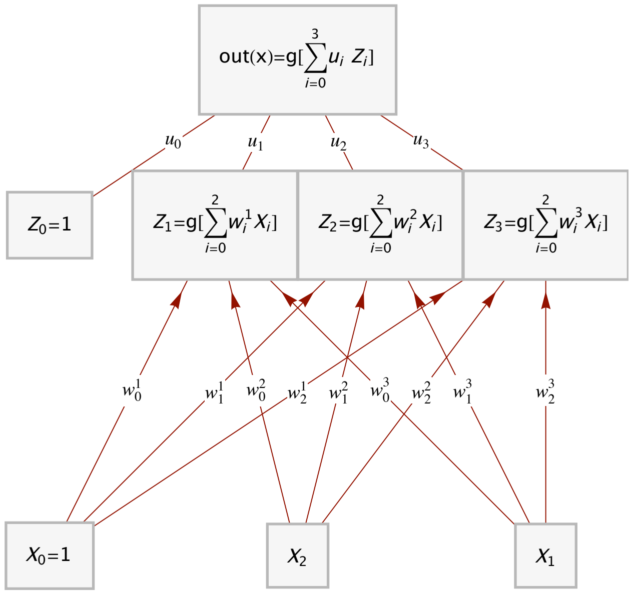
```
Manipulate[
 g = Function[x, 1/(1 + Exp[-x])];
 Plot3D[ g[w0 + w1 x1 + w2 x2], {x1, -3, 3}, {x2, -3, 3}],
 {{w0, 0}, -3, 3}, {{w1, 2}, -3, 3}, {{w2, -2}, -3, 3}
]
```



Neural Network with Multiple Hidden Layers

Lets Consider what this network looks like

```
plt
```



Matlab Style Forward Propagation

Lets define a matrix Was:

$$W = \begin{pmatrix} w_0^1 & w_1^1 & w_2^1 \\ w_0^2 & w_1^2 & w_2^2 \\ w_0^3 & w_1^3 & w_2^3 \end{pmatrix} \tag{2}$$

We can multiply this matrix by X where we have added a 1

$$W \cdot \begin{pmatrix} 1 \\ X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} w_0^1 & w_1^1 & w_2^1 \\ w_0^2 & w_1^2 & w_2^2 \\ w_0^3 & w_1^3 & w_2^3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} w_0^1 + w_1^1\, X_1 + w_2^1\, X_2 \\ w_0^2 + w_1^2\, X_1 + w_2^2\, X_2 \\ w_0^3 + w_1^3\, X_1 + w_2^3\, X_2 \end{pmatrix} \tag{3}$$

Lets define function application as element wise. Then we obtain:

$$g\left[W \cdot \begin{pmatrix} 1 \\ X_2 \\ X_3 \end{pmatrix}\right] = \begin{pmatrix} g\left[w_0^1 + w_1^1\, X_1 + w_2^1\, X_2\right] \\ g\left[w_0^2 + w_1^2\, X_1 + w_2^2\, X_2\right] \\ g\left[w_0^3 + w_1^3\, X_1 + w_2^3\, X_2\right] \end{pmatrix} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} \tag{4}$$

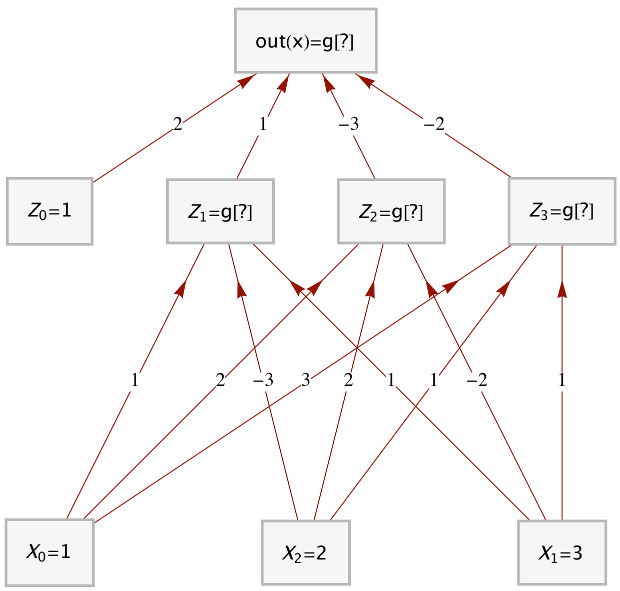
We can then prepend a 1 to the result to obtain:

$$\text{out}(X) = g\left[\begin{pmatrix} u_0 & u_1 & u_2 & u_3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}\right] = g\left[u_0 + \sum_{i=1}^{3} u_i Z_i\right] \tag{5}$$

Forward Propagation (Example) #1

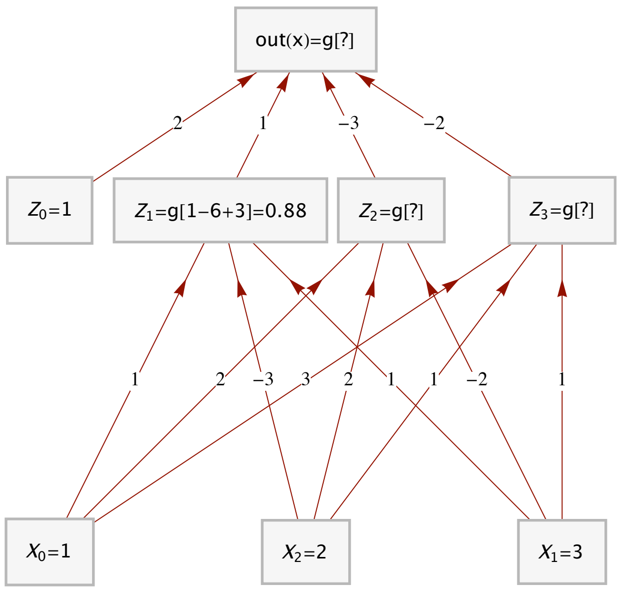
- ***What is the value of Z_1***

```
plotTree[{"out(x)=g[?]", "Z₀=1", "Z₁=g[?]", "Z₂=g[?]", "Z₃=g[?]",
  "X₀=1", "X₂=2", "X₁=3"}]
```



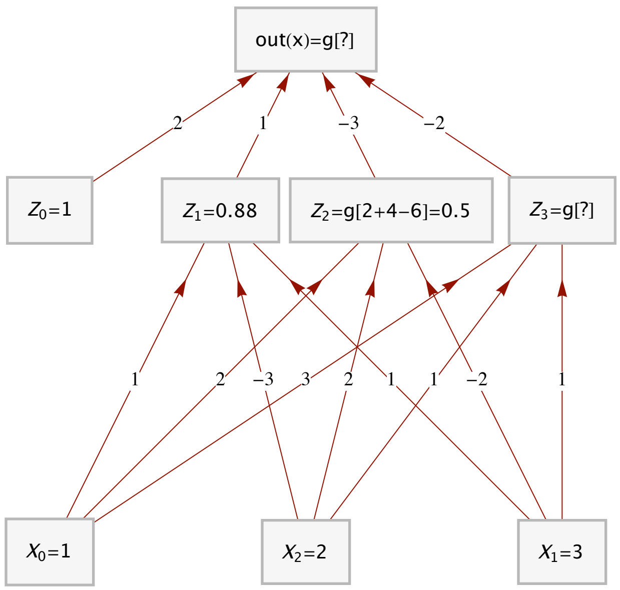
What is the value of Z_2?

```
plotTree[{"out(x)=g[?]", "Z₀=1", "Z₁=g[1-6+3]=0.88", "Z₂=g[?]",
  "Z₃=g[?]", "X₀=1", "X₂=2", "X₁=3"}]
```



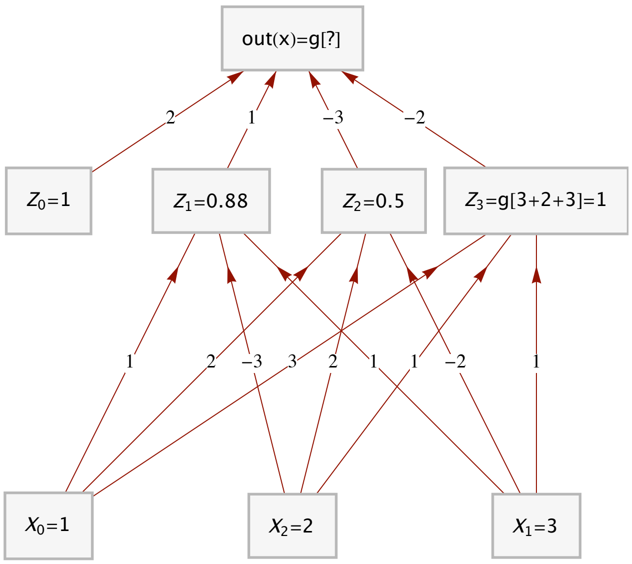
What is the value of Z_3?

```
plotTree[{"out(x)=g[?]", "Z₀=1", "Z₁=0.88", "Z₂=g[2+4-6]=0.5", "Z₃=g[?]",
  "X₀=1", "X₂=2", "X₁=3"}]
```



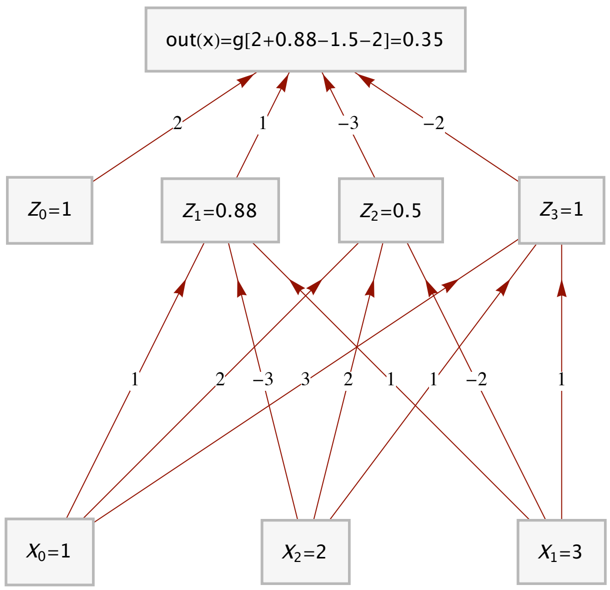
What is the value of out(X)?

```
plotTree[{"out(x)=g[?]", "Z₀=1", "Z₁=0.88", "Z₂=0.5", "Z₃=g[3+2+3]=1",
  "X₀=1", "X₂=2", "X₁=3"}]
```



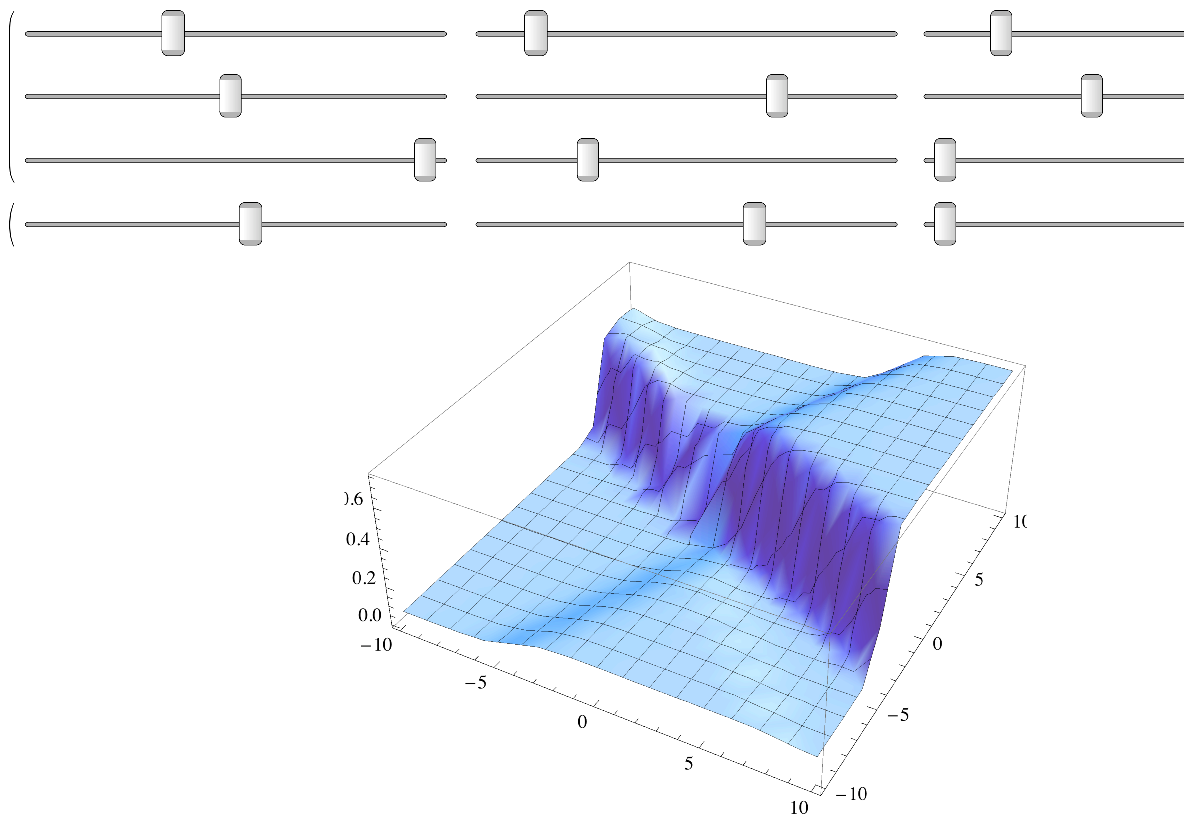
Done!

```
plotTree[{"out(x)=g[2+0.88-1.5-2]=0.35", "Z₀=1", "Z₁=0.88", "Z₂=0.5",
  "Z₃=1", "X₀=1", "X₂=2", "X₁=3"}]
```



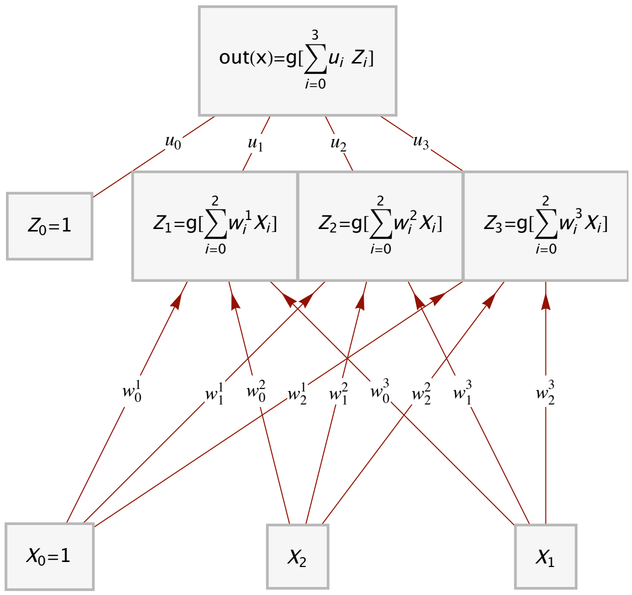
Demo

```
dynamicDemo
```



Generalized Back Propagation

```
plt
```



Suppose we want to find the best model out(x; U, W) with respect to the parameters W and U. How can we quantify best? Lets considered mean squared error.

$$E = \sum_{i=1}^{n} (\text{out}(X_i) - Y_i)^2 \tag{6}$$

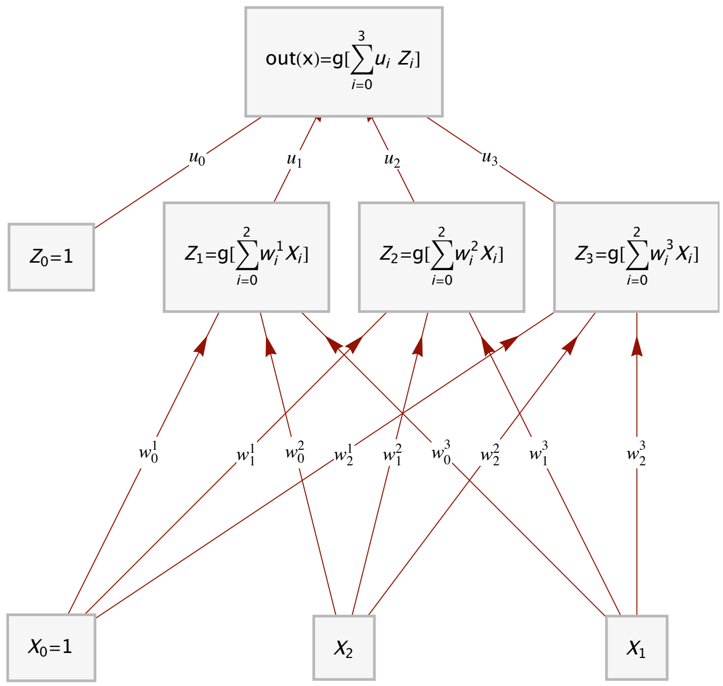
There are many ways to do this. One of the most common (and least effective) methods is to use gradient descent. This corresponds to the update rule:

$$u_i^{(t+1)} \longleftarrow u_i^{(t)} - \eta \left. \frac{\partial E}{\partial u_i} \right|_{u_i^{(t)}} \tag{7}$$

$$w_{\text{ij}}^{(t+1)} \longleftarrow w_{\text{ij}}^{(t)} - \eta \left. \frac{\partial E}{\partial w_{\text{ij}}} \right|_{w_{\text{ij}}^{(t)}} \tag{8}$$

Recall we have the following graph:

```
plt
```



Lets first derive the update rule for U:

$$E = (\text{out}(X) - Y)^2 \tag{9}$$

Taking the derivative we get (stuck?):

$$\frac{\partial E}{\partial u_k} = \frac{\partial}{\partial u_k}(\text{out}(X) - Y)^2 \tag{10}$$

Applying the infamous chain rule:

$$\frac{\partial}{\partial x} f(g(x)) = \left(\frac{\partial}{\partial u} f(u) \Bigg|\, u = g\,(x)\right)\left(\frac{\partial}{\partial x} g(x)\right) \tag{11}$$

$$\frac{\partial E}{\partial u_k} = \underbrace{2\,(\text{out}(X) - Y_i)}_{\frac{\partial}{\partial x} x^2 = 2\,x}\left(\frac{\partial}{\partial u_k}\text{out}(X)\right) \tag{12}$$

$$\frac{\partial}{\partial x} f(g(x)) \;=\; f'(g(x))\, g'(x)$$

Now we need to take the derivative of the neural network. Lets first replace out with the function from the top perceptron

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\left(\frac{\partial}{\partial u_k}\, g\Big[\sum_{i=0}^{3} u_i\, Z_i\Big]\right) \tag{13}$$

Chain rule again

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\, g'\Big[\sum_{i=0}^{3} u_i\, Z_i\Big] \left(\sum_{i=0}^{3} \frac{\partial}{\partial u_k}\, u_i\, Z_i\right) \tag{14}$$

We know that only one term in the Z_i sum will remain and that is $Z_{i=k}$

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\, g'\Big[\sum_{i=0}^{3} u_i\, Z_i\Big]\, Z_k \tag{15}$$

Done thats it!!! Sort of. Lets look at the derivative of $g[x] = \frac{1}{1-\text{Exp}[-x]}$

$$g'[x] = \frac{\partial}{\partial x}\,(1 + \text{Exp}[-x])^{-1} \tag{16}$$

$$g'[x] = -(1 + \text{Exp}[-x])^{-2}\, \frac{\partial}{\partial x}\,(1 + \text{Exp}[-x]) \tag{17}$$

$$g'[x] = -(1 + \text{Exp}[-x])^{-2} \left(\frac{\partial}{\partial x}\,1 + \frac{\partial}{\partial x}\,\text{Exp}[-x]\right) \tag{18}$$

$$g'[x] = -(1 + \text{Exp}[-x])^{-2} \left(0 + \text{Exp}[-x]\, \frac{\partial}{\partial x}\,(-x)\right) \tag{19}$$

$$g'[x] = -(1 + \text{Exp}[-x])^{-2}\,(0 - \text{Exp}[-x]) \tag{20}$$

$$g'[x] = (1 + \text{Exp}[-x])^{-2}\, \text{Exp}[-x] \tag{21}$$

With some manipulation we get:

$$g'[x] = \frac{\text{Exp}[-x]}{(1 + \text{Exp}[-x])}\, \frac{1}{(1 + \text{Exp}[-x])} \tag{22}$$

$$g'[x] = \frac{\text{Exp}[-x]}{1 + \text{Exp}[-x]}\, g[x] \tag{23}$$

$$g'[x] = \frac{1 + \text{Exp}[-x] - 1}{1 + \text{Exp}[-x]}\, g[x] \tag{24}$$

$$g'[x] = \left(\frac{1 + \text{Exp}[-x]}{1 + \text{Exp}[-x]} + \frac{-1}{1 + \text{Exp}[-x]}\right) g[x] \tag{25}$$

$$g'[x] = \left(1 - \frac{1}{1 + \text{Exp}[-x]}\right) g[x] \tag{26}$$

$$\boldsymbol{g'[x] = (1 - g[x])\, g[x]} \tag{27}$$

Recall that we earlier had:

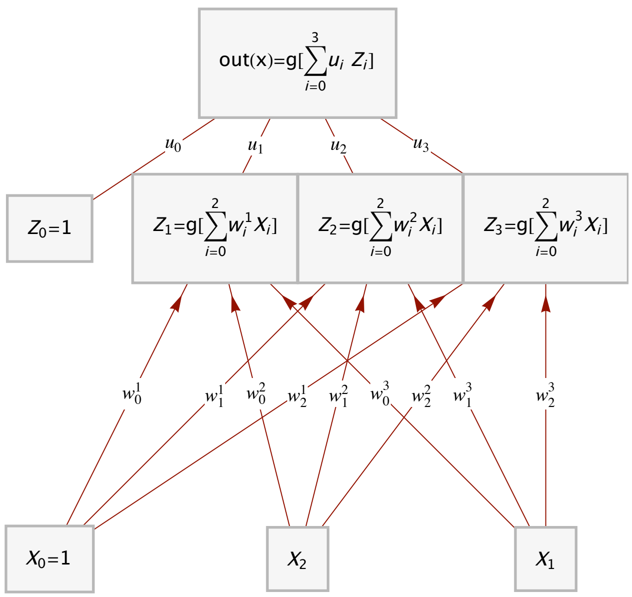
$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\, g'\Big[\sum_{i=0}^{3} u_i\, Z_i\Big]\, Z_k \tag{28}$$

we can make a simple substitution to get:

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\left(1 - g\left[\sum_{i=0}^{3} u_i\, Z_i\right]\right) g\left[\sum_{i=0}^{3} u_i\, Z_i\right] Z_k \tag{29}$$

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X)\, Z_k \tag{30}$$

```
plt
```



Gradient of *W*

That wasn't too bad. How about the next layer. We again start with:

$$E = (\text{out}(X) - Y)^2 \tag{31}$$

Taking the derivative with respect to w_k^r (and applying the chain rule)

$$\frac{\partial E}{\partial w_k^r} = \frac{\partial E}{\partial \text{out}(X)} \frac{\partial \text{out}(X)}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\, \frac{\partial}{\partial w_k^r}\, \text{out}(X) \tag{32}$$

Expanding out we get:

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\, \frac{\partial}{\partial w_k^r}\, g\left[\sum_{i=0}^{3} u_i\, Z_i\right] \tag{33}$$

Chain rule:

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y) \left(g'\left[\sum_{i=0}^{3} u_i\, Z_i\right]\right) \left(\sum_{i=0}^{3} \frac{\partial}{\partial w_k^r}\, u_i\, Z_i\right) \tag{34}$$

Recall that $g'[x] = (1 - g[x])\, g[x]$

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\, \text{out}(X) \left(\sum_{i=0}^{3} \frac{\partial}{\partial w_k^r}\, u_i\, Z_i\right) \tag{35}$$

Remember that each of the Z_i is connected to all the perceptrons from the lower level so we must take the derivative of each.

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\, \text{out}(X) \left(\sum_{i=0}^{3} u_i\, \frac{\partial}{\partial w_k^r}\, Z_i\right) \tag{36}$$

Which becomes:

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\, \text{out}(X) \left(\sum_{i=0}^{3} u_i\, \frac{\partial}{\partial w_k^r}\, g\left[\sum_{s=0}^{3} w_s^i\, X_s\right]\right) \tag{37}$$

Another application of the chain rule:

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\, \text{out}(X) \sum_{i=0}^{3} u_i\, g'\left[\sum_{s=0}^{3} w_s^i\, X_s\right] \frac{\partial}{\partial w_k^r} \sum_{s=0}^{3} w_s^i\, X_s \tag{38}$$

Taking the final derivative we have:

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\, \text{out}(X) \left(\sum_{i=0}^{3} u_i\,(1 - Z_i)\,(Z_i)\right) X_k \tag{39}$$

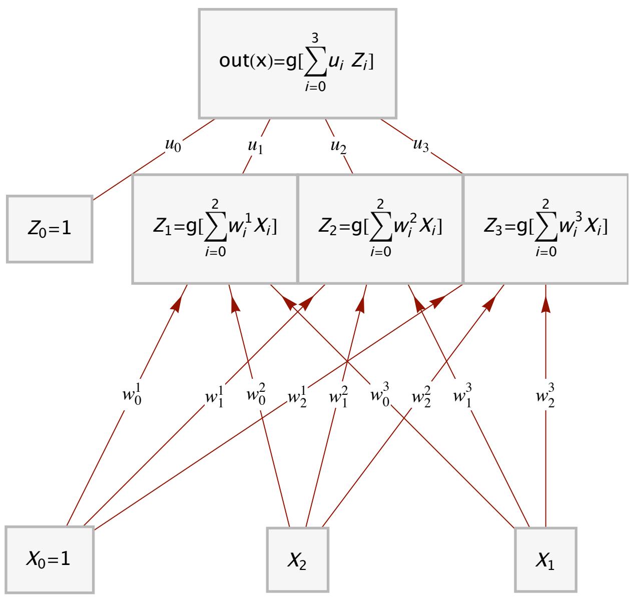
Why is it called back propagation?

Lets look at the following two equations:

$$\frac{\partial E}{\partial u_k} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X)\,Z_k \tag{40}$$

$$\frac{\partial E}{\partial w_k^r} = 2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X)\sum_{i=0}^{3} u_i\,(1 - Z_i)\,(Z_i)\,X_k \tag{41}$$

```
plt
```



We propagate the derivative information backwards to the inputs:

$$\frac{\partial E}{\partial u_k} = (2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X))\,Z_k \tag{42}$$

$$\frac{\partial E}{\partial w_k^r} = (2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X))\sum_{i=0}^{3} u_i\,(1 - Z_i)\,(Z_i)\,X_k \tag{43}$$

$$\frac{\partial E}{\partial w_k^r} = \left((2\,(\text{out}(X) - Y)\,(1 - \text{out}(X))\,\text{out}(X))\sum_{i=0}^{3} u_i\,(1 - Z_i)\,(Z_i)\right) X_k \tag{44}$$

Questions?

- ***Any feedback about Mathematica style presentations is welcome.***