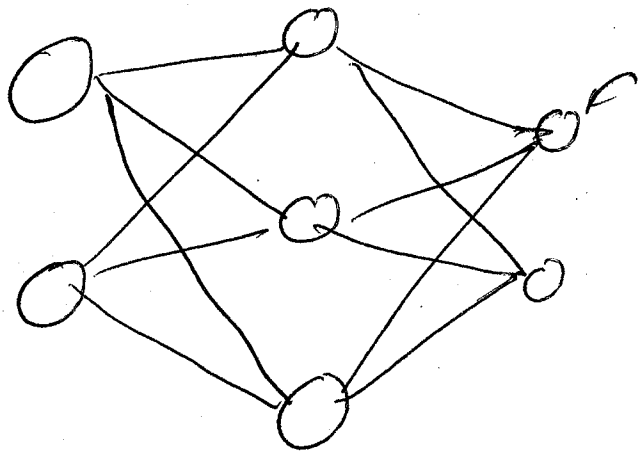
Backprop Notes

Before getting into the details, will give a computational version of backprop, then we'll fill in the theoretical details.

Think of a network as comprised of computational nodes



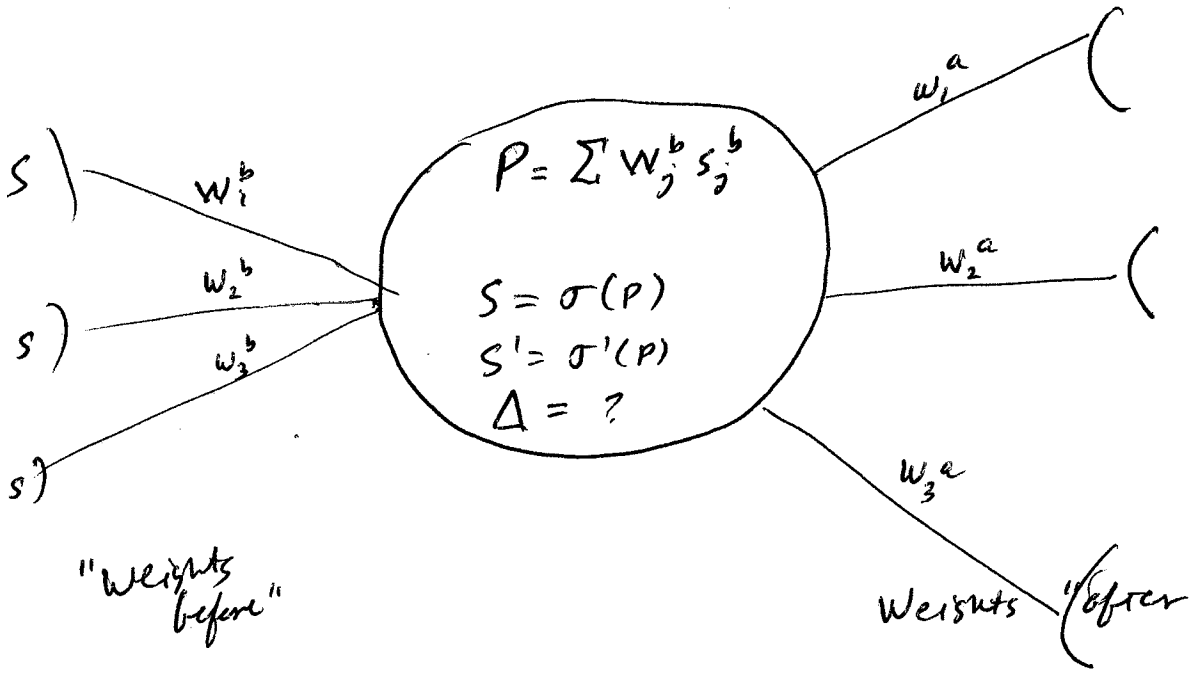
Each neuron has a "node"!

Each node performs computations. In programming, ~~with~~ each node will have 4 numerical values:

$P,\ S,\ S',\ \Delta$

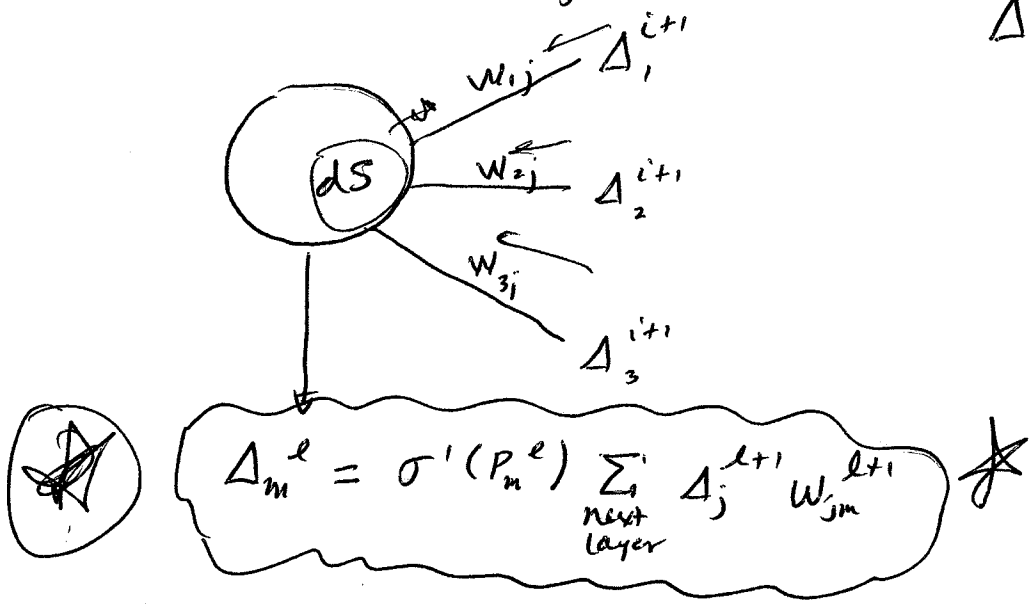
Where P is the preskate, S is the state; $S = \sigma(P)$, S' is the derivative, $\sigma'(P)$, and Δ is a value we'll use to compute some derivatives.

Each node is also connected to other nodes by edges. Where we think of a weight as being attached to the edge.



The "forward phase" for a network is to compute P, S and S' as the information goes forward.

The "backwards" phase is to compute the values of Δ in a recurrent fashion:

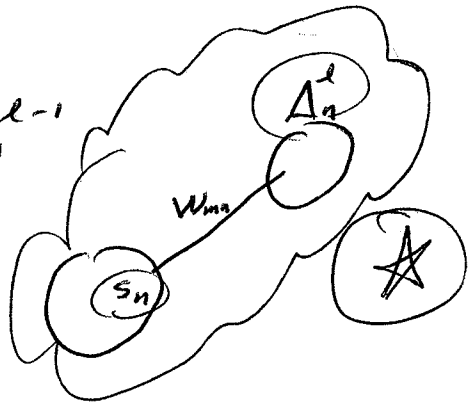


Δ on the output layer is $(t-y)\,\sigma'$

Then $\Delta W_{ij} = S^i \cdot \Delta_1^{i+1}$

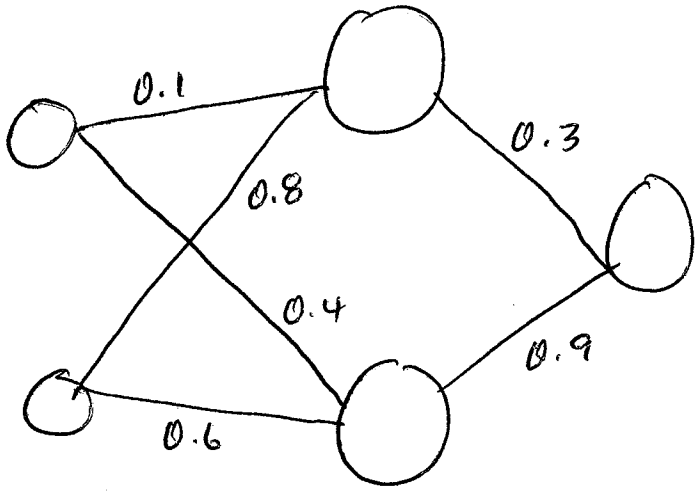
Then we update the weights:

$$W_{ijn}^{new} = W_{ijn}^{old} + \epsilon \, \Delta_m^l \, S_n^{l-1}$$



This is called the **backpropagation of error**

Example in the Matlab script:



Initial Network, Initial training pattern:
$(0.35, 0.9) \rightarrow (0.5)$

Code:

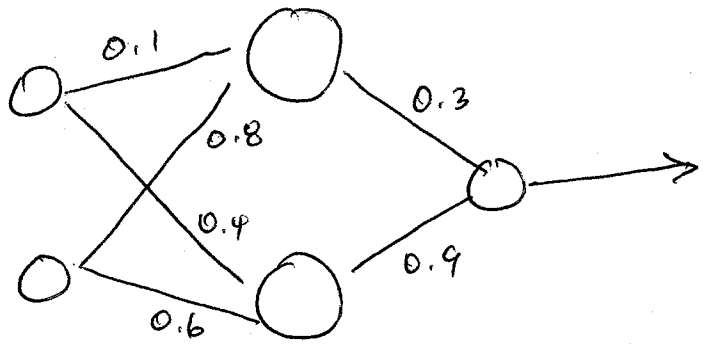
Forward Pass: Construct P, S, ds

Backwards Pass: Construct Δ

Change weights.

Repeat.

```
% Code for a 2 x 2 x 1 network- Training by gradient descent by hand.

s=inline('1./(1+exp(-z))');
STEP=0.1;


% Set up the parameters

x=[0.35; 0.9]; t=0.5;

w11=[0.1; 0.8];
w12=[0.4;0.6];
w2=[0.3;0.9];

for j=1:100

% FORWARD PASS:

P01=x(1);
S01=x(1);
dS01=1;

P02=x(2);
S02=x(2);
dS02=1;

P11=w11'*[S01;S02];
S11=s(P11);
dS11=S11*(1-S11);

P12=w12'*[S01;S02];
S12=s(P11);
dS12=S12*(1-S12);

P2=w2'*[S11;S12];
S2=P2;
dS2=1;

Out(j)=S2;
Err(j)=t-S2;

% BACKWARDS Compute Deltas

% Last layer:
Delta2=(t-S2)*dS2

Delta11=dS11*w2(1)*Delta2;
Delta12=dS12*w2(2)*Delta2;

Delta01=dS01*(w11(1)*Delta11+w12(1)*Delta12);
Delta02=dS02*(w11(2)*Delta11+w12(2)*Delta12);

% Update the weights using the deltas
w11(1)=w11(1)+STEP*Delta11*S01;
w11(2)=w11(2)+STEP*Delta11*S02;
w12(1)=w12(1)+STEP*Delta12*S01;
w12(2)=w12(2)+STEP*Delta12*S02;
```



With sample: $(0.35, 0.9) \rightarrow (0.9)$

Forward Pass: For each of the 5 nodes, Compute P, S, dS.

Backwards Pass: For each of the 5 nodes, Compute the Δ's.

Update the weights... (Gradient Descent)

```
w2(1)=w2(1)+STEP*Delta2*S11;
w2(2)=w2(2)+STEP*Delta2*S12;
end
```

An even simpler example:

Forward Phase: $p = w_1 x$ $\quad$ $\varphi = w_2 s$

$s = \sigma(w_1 x)$ $\quad$ $s = p$

$ds = \sigma'(w_1 x)$ $\quad$ $ds = 1$

Backward Phase:

$\Delta^0 = 1 \cdot w_1 \Delta^1$ $\quad$ $\Delta^1 = w_2 \Delta^2 \cdot ds$ $\quad (t-y) = \Delta^2$

$w_2 (t-y) \sigma'(w_1 x)$

or $w_2 \sigma'(w_1 x)(t-y)$

$= w_1$

$\Delta w_1 = x \cdot \Delta^1 = x \left(w_2 \sigma'(w_1 x)(t-y) \right)$

$\Delta w_2 = \sigma(w_1 x) \cdot \Delta^2 = \sigma(w_1 (x)) \cdot (t-y)$

Now, notice that

$y = f(x) = w_2 \sigma(w_1 (x))$. $\quad E = \frac{1}{2}(t-y)^2$

So $\frac{\partial E}{\partial w_2} = (t-y) \cdot \left(-\frac{\partial y}{\partial w_2}\right)$

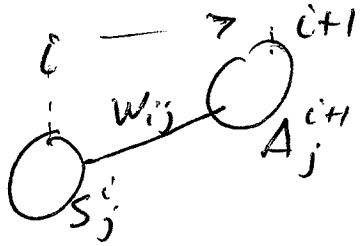
$= (t-y) \left(\sigma(w_1 (x)) \right.$

So $\Delta w_2 = -\frac{\partial E}{\partial w_2}$

Similarly, $\frac{\partial E}{\partial w_1} = (t-y) \left(-\frac{\partial y}{\partial w_1}\right)$

$= (t-y) w_2 \sigma'(w_1 x) \cdot x = -\Delta w_1$

What we should show is that:



$$\Delta W_{ij} = \Delta_j^{i+1} S_j^i$$

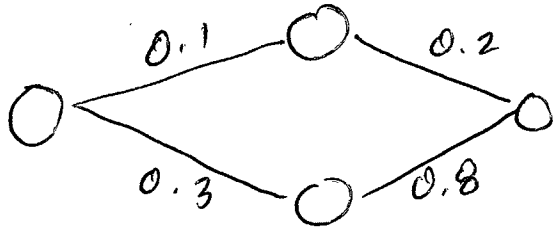
is actually $\frac{\partial E}{\partial W_{ij}}$.

This takes a bit to show - There are multiple indices to keep track of, and it is easy to get lost.

Run the code & try it out.

Hw: Backprop. of Error.

Consider the 1-2-1 network below:



- For input & output layer, $\sigma(x) = x$
- For middle layer, $\sigma(x) = \frac{1}{1+e^{-x}}$

(a) If our desired pairing is $(\frac{1}{2}, \frac{2}{3})$, then

Compute the change in each weight for the first step in gradient descent (Δw).

(b) Show that $\Delta w = \frac{\partial E}{\partial w}$ for each weight w.

You may take $E = \frac{1}{2}(t-y)^2$; where t is the desired target and y is the output of the network.

Also complete exercises 1, 2, 4, 5. Turn in your solns for 2 & 5 together w/ the backprop exercise on this page.

Due: Tuesday