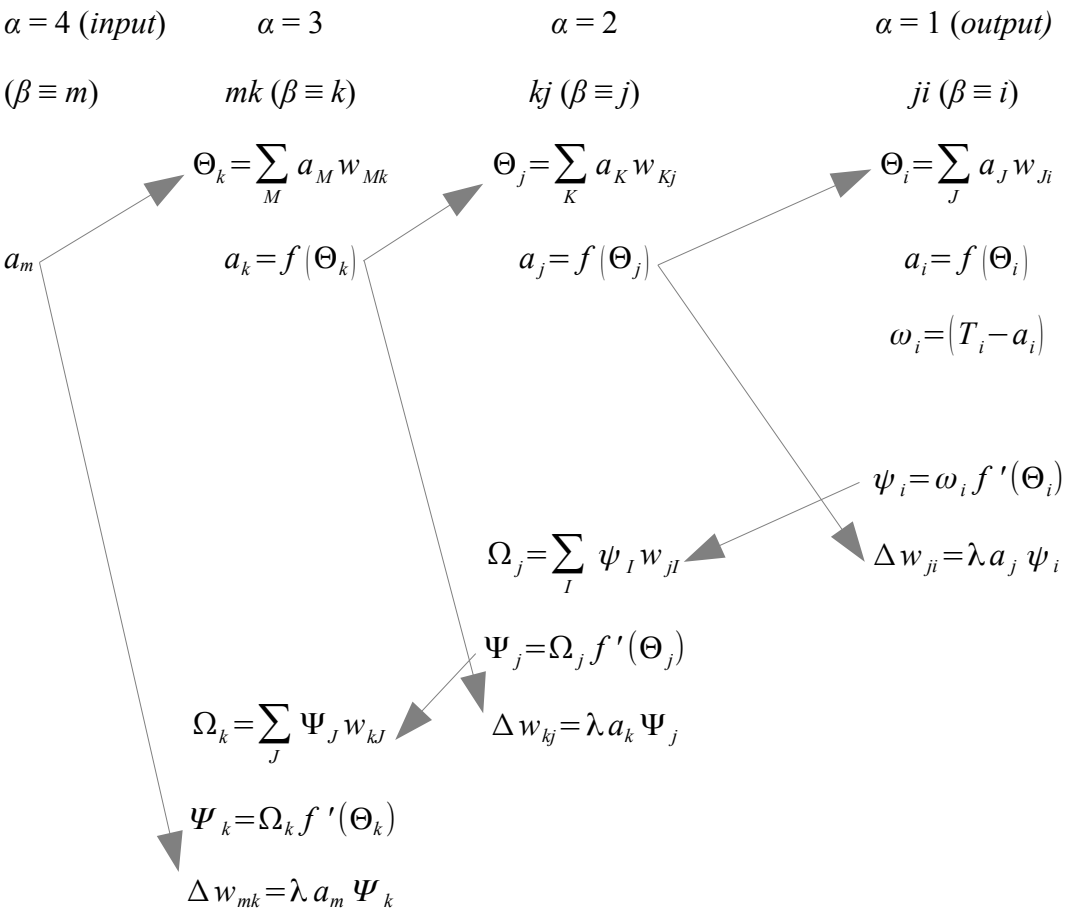
Consider the idea that all we have are different layers with activations *a*, so the error function would be:

$$E=\frac{1}{2}\sum_I \omega_i^2 \text{ with } \omega_i=\left(T_i-F_i\right).$$

Now consider a network with four activation layers. I have identified each layer by an index alpha (α) and have removed the special symbols for the input and output activations that have been used in the previous documentation. The various calculations that were derived for the three activation-layer network can be structured to more clearly show the interdependencies between the layers. I have also introduced a new index beta (β) that will soon be used to increment through the index values that are specific to a given layer. For the moment I am using the same convention as in the previous documents, that the index letter is associated with a specific layer, although I am generalizing the notion.



The brute force path down the network to evaluate it is

```
for i = 1 to Ni
   for j = 1 to Nj
      for k = 1 to Nk
         for m = 1 to Nm
             Θk += am * wmk
          next m
          ak = f(Θk)
          Θj += ak * wkj
      next k
      aj = f(Θj)
      Θi += aj * wji
   next j
   ai = f(Θi)
   ωi = Ti - ai
next i
```

It should be clear from the diagram (and the pseudo code) that we can make a generalized set of constructs to hold the various elements that are required to be propagated down and back up the network. The only layer that has any special elements is the bottom (the output layer) since it is where the training models come in. We need to loop over all the layers starting from the right-most activation layer, which has the outputs. Any variable that previously had a single index value (which implied the layer) will require two and the weights three. The first index is the layer, index α, and the second index is the specific element within that layer, call this index β, so we have activations $a_{\alpha\beta}$. The beta index is what was previously represented by the i, j, k, and m indices that were also associated with a specific activation layer. Using this notation, the above loop becomes

```
for i = 1 to N1i
   for j = 1 to N2j
      for k = 1 to N3k
         for m = 1 to N4m
            Θ3k += a4m * w3mk
         next m
         a3k = f(Θ3k)
         Θ2j += a3k * w2kj
      next k
      a2j = f(Θ2j)
      Θ1i += h2j * w1ji
   next j
   a1i = f(Θ1i)
   ωi = Ti - a1i
next i
```

where $\alpha = 1$ is associated with the *i* index, $\alpha = 2$ with the *j* index, and so on. Look at the associations of the layer index and the element index within that layer and you find that 4*m*, 3*k*, 2*j*, and 1*i* now identify each of the elements in a specific layer. Note that the N values in the above pseudocode represent the highest index and not necessarily the number of elements (zero indexing *vs.* one indexing).

If you look at the inner most loop over the *m* index, it is clear that we need to loop over each of the elements within the α activation layer (index β) and the layer above it (index γ).

```
for α = N_layers-1 to 1      // layers m,k,j,i etc...
   for β = 1 to N_α          // values in layer α
      for γ = 1 to N_α+1     // Values in layer α+1
         Θ_αβ += a_(α+1)γ * w_αγβ
      next γ
      a_αβ = f(Θ_αβ)
   next β
next α
```

When these loops have completed we should have the activations. The abstraction in our mathematical notation should be

$$a_{\alpha\beta} = f\left(\Theta_{\alpha\beta}\right) \text{ where } \Theta_{\alpha\beta} = \sum_{\gamma} a_{(\alpha+1)\gamma} w_{\alpha\gamma\beta}$$

where α is the activation layer indexed over β and index γ is for layer α+1, remembering that the layer indexing goes from right to left, so the left most activations are the input layer (and so are not calculated, but are set as input to the network).

Note that you can no longer easily calculate the ω values when running the network since the output activations are no longer special and the structure of the Ω values are very different than that of ω. You will need to determine the ω values in the training by treating the output layer as a special case. The Ω values can then be determined in a more generalized loop.

Now you need to rewrite the back propagation relationships in the final more generalized notation. The symbolic logic from the previous section is repeated here. You need to rewrite it so that it reflects the use of the α, β, and γ and indices as they are used above. You will then need to write the relevant pseudocode.

Updating the weights in the left most layer for a two layer network:

$$\Delta w_{kj} = \lambda a_k \Psi_j \text{ where } \Psi_j = \Omega_j f'\left(\Theta_j\right),\ \Omega_j = \sum_I \psi_I w_{jI},\ \Theta_j = \sum_K a_K w_{Kj} \text{ and}$$
$$\psi_i = \omega_i f'\left(\Theta_i\right) \text{ where } \omega_i = \left(T_i - F_i\right) \text{ and } \Theta_i = \sum_J h_J w_{Ji}.$$

The network is evaluated with the feed-forward calculations

$$F_i = f(\Theta_i) \text{ where } \Theta_i = \sum_J h_J w_{Ji} \text{ and } h_j = f(\Theta_j) \text{ where } \Theta_j = \sum_K a_K w_{Kj} .$$

Back propagation derivation for three or more layers would have something to the effect of

$$\Delta w_{mk} = \lambda a_m \Psi_k \text{ where } \Psi_k = \Omega_k f'(\Theta_k) , \ \Omega_k = \sum_J \Psi_J w_{kJ} , \ \Theta_k = \sum_M a_M w_{Mk}$$

for the next layer to the left that would have activations a_m. Note that Ψ_j (which is tied to the output layer) is not the same as Ψ_k (which is tied to a hidden layer). You should notice that there will be an array of Θ values for every right-side activation layer, so there will be an array of values for every hidden layer and the output layer. These values can be collected when the activation/weight multiplications are being performed when the network is being evaluated. You will also need to maintain the activation values for every hidden layer.