Here's a brief description of the backpropagation algorithm. The notation is a bit unwieldy but hopefully meaningful.

- Initialize all weights to random values in $[-1, 1]$.
- Make sure to add bias nodes $x_0^{(n)} = 1$, and $z_0^{(n)} = 1$.
- Do A (until $\sum_j |\Delta w_j^{(2)}| + \sum_{ji} |\Delta w_{ji}^{(1)}| \leq \epsilon$): t loop
  - Do B (until number of patterns in training set are exhausted): n loop
    * For input pattern $x^{(n)}$ and current setting of weights $w^{(t)}$, compute $y^{(n)}$.
    * $y^{(n)} = \frac{1}{1+\exp\{-\beta \sum_j w_j^{(2)} z_j^{(n)}\}}$ where $z_j^{(n)} = \tanh(\beta \sum_i w_{ji}^{(1)} x_i^{(n)})$.
    * Compute $\delta^{(n)} = \frac{\partial E^{(n)}}{\partial a^{(n)}}$, where $E^{(n)} = (y^{(n)} - t^{(n)})^2$, $a^{(n)} = \sum_j w_j^{(2)} z_j^{(n)}$, and $y^{(n)} = \frac{1}{1+e^{-\beta a^{(n)}}}$
    * Compute $\frac{\partial E^{(n)}}{\partial w_j^{(2)}} = \delta^{(n)} z_j^{(n)}$
    * Compute $\delta_j^{(n)} = \frac{\partial E^{(n)}}{\partial a_j^{(n)}}$ using $\delta_j^{(n)} = g_1'(a_j^{(n)}) w_j^{(1)} \delta^{(n)}$ where $g_1(a) = \tanh(\beta a)$.
    * Compute $\frac{\partial E^{(n)}}{\partial w_{ji}^{(1)}} = \delta_j^{(n)} x_i^{(n)}$.
  - End B
  - $\frac{\partial E}{\partial w_j^{(2)}} = \sum_n \frac{\partial E^{(n)}}{\partial w_j^{(1)}}$, $\frac{\partial E}{\partial w_{ji}^{(1)}} = \sum_n \frac{\partial E^{(n)}}{\partial w_{ji}^{(1)}}$
  - Initialize $\alpha = 10$.
  - Do C: α loop.
    * $\alpha = \alpha/2$
    * $w_j^{(temporary)} = w_j^{(temporary)} - \alpha \frac{\partial E}{\partial w_j}$, $w_{ji}^{(temporary)} = w_{ji}^{(temporary)} - \alpha \frac{\partial E}{\partial w_{ji}}$
    * If $E^{(temporary)} < E^{(t)}$, accept α.
  - End C
  - $w_j^{(t+1)} = w_j^{(t)} - \alpha \frac{\partial E}{\partial w_j^{(t)}}$, $w_{ji}^{(t+1)} = w_{ji}^{(t)} - \alpha \frac{\partial E}{\partial w_{ji}^{(t)}}$
- End A