Implementation of Back Propagation Algorithm

Andrei Smirnov
andrei.v.smirnov@gmail.com

September 19, 2016

1 Network

L	Number of layers
N_l	Number of nodes on layer l: l=0..$L-1$
N_T	Number of training sets
$T_s = [t^0_i, t^1_j]^s_{i=0..N_0, j=0..N_{L-1}}$	Collection of training sets ($s = 0..N_T - 1$), where 0 corresponds to the input, and 1 - to the output

All indexes start with 0. Because weights occupy only $L - 1$ layers, the weight index l corresponds to layer $l + 1$. ...

2 Method

The errors are defined on all layers except the first one $l = 0$ where the input is provided. Since the indexing of the array of errors starts with 0 we will refer to the errors on layer l as E_{l-1}: Then the errors on layer l $(0 < l < L)$ are given by:

$$E_{l-1} = \frac{1}{2} \sum_{i=0}^{N_l - 1} (a(p_i^{l-1}) - a(v_i^l))^2 \tag{1}$$

where v_i^l is a *variable* assigned to the node i on layer l, $a()$ is the *activation* function, and p_i^l is a *projection* of layer l onto layer $l+1$, which is equal to:

$$p_i^l = \sum_{j=0}^{N_l-1} w_{ij}^l a(v_j^l) + b_j^l \tag{2}$$

Here w_{ij}^l is the strength of connection between node i on layer $l+1$ and node j on layer l, which will be referred to as the *weight* [1], and b_j^l is node's *bias*, which is a constant value assigned to each node.
The objective is to minimize E_l on the set of weights, $W_l = \{w_{ij}^l\}$, and biases, $B^l = \{b_i^l\}$:

$$E_l = E_l(W^l, B^l) \rightarrow \min_{W^l, B^l} \tag{3}$$

During network training the values on the first layer of nodes are set equal to the input values of the trianing set:

$$v_i^0 = t_i^0, \quad {}_{i=0..N_0-1}$$

On the last layer the variables are set to the output values of the training set:

$$v_i^{L-1} = t_i^1, \quad {}_{i=0..N_{L-1}-1}$$

According to (1), reducing all errors to zero will result in $v_i^l = p_i^{l-1}$ or:

$$v_i^l = \sum_{j=0}^{N_{l-1}-1} w_{ij}^{l-1} a(v_j^{l-1}) + b_j^{l-1}$$

The minimization problem (3) can be solved by finding w_{ij}^l, b_i^l which satisfy:

$$dE_l = \frac{\partial\, E_l}{\partial\, w_{ij}^l} dw_{ij}^l + \frac{\partial\, E_l}{\partial\, b_j^l} db_j^l = 0$$

This in turn can be accomplished by repeatedly moving vectors w_{ij}^l, b_i^l in the direction opposite to the gradient of E_l:

$$\nabla\, E_l = \left(\frac{\partial\, E_l}{\partial\, w_{ij}^l}, \frac{\partial\, E_l}{\partial\, b_j^l} \right)$$

[1]The less obvious order of indexes ij is used for numerical efficiency of the algorithm.

i.e.:

$$\begin{aligned} w_{ij}^l &\rightarrow w_{ij}^l + dw_{ij}^l \\ b_i^l &\rightarrow b_i^l + db_i^l \end{aligned} \tag{4}$$

where

$$dw_{ij}^l = -\epsilon \frac{\partial E_l}{\partial w_{ij}^l} \tag{5}$$

$$db_i^l = -\epsilon \frac{\partial E_l}{\partial b_i^l} \tag{6}$$

and ϵ is a relaxation factor selected by trial-and-error as a compromise between the speed of convergence and stability. To find the gradient of E_l we use relations (1) and (2):

$$\begin{aligned} \frac{\partial E_l}{\partial w_{ij}^l} &= \frac{1}{2}\frac{\partial}{\partial w_{ij}^l}\left(\sum_{k=0}^{N_l-1}\left(a(p_k^l) - a(v_k^{l+1})\right)^2\right) \\ &= \sum_{k=0}^{N_l}\left(a(p_k^l) - a(v_k^{l+1})\right) a'(p_k^l)\frac{\partial p_k^l}{\partial w_{ij}^l} \\ &= \sum_{k=0}^{N_l}\left(a(p_k^l) - a(v_k^{l+1})\right) a'(p_k^l)\frac{\partial}{\partial w_{ij}^l}\left(\sum_{p=0}^{N_l-1} w_{kp}^l a(v_p^l) + b_p^l\right) \\ &= \sum_{k=0}^{N_l}\left(a(p_k^l) - a(v_k^{l+1})\right) a'(p_k^l)\left(\delta_{ik}\delta_{jp} a(v_p^l)\right) \\ &= \left(a(p_i^l) - a(v_i^{l+1})\right) a'(p_i^l) a(v_j^l) \end{aligned} \tag{7}$$

where $a'()$ is the derivative of $a()$ and p_i^l is defined by (2). Analogously, for $\partial E_l / \partial b_i^l$ we find:

$$\frac{\partial E_l}{\partial b_i^l} = \left(a(p_i^l) - a(v_i^{l+1})\right) a'(p_i^l) \tag{8}$$

Now equations (5), (6) become:

$$dw_{ij}^l = \epsilon \left(a(v_i^{l+1}) - a(p_i^l) \right) a'(p_i^l) a(v_j^l) \tag{9}$$
$$db_i^l = \epsilon \left(a(v_i^{l+1}) - a(p_i^l) \right) a'(p_i^l) \tag{10}$$

3 Algorithm

L	Number of layers
N_l	Number of nodes on layer l: l=0..$L-1$
N_T	Number of samples in the training set
I_i^t	Input sample of the training set: t=0..N_T-1, i=0..N_0-1
O_i^t	Output sample of the training set: t=0..N_T-1, i=0..$N_{L-1}-1$

```
# Initialize
For l=0..L−1
        For i=0..N_l−1:
                v_i^l = 0
# Initialize weights and biases[2]
For l=0..L−2:
        For i=0..N_{l+1}−1:
                b_i^l = 0
                For j=0..N_l−1:
                        w_ij^l = rand(−1,1)

eps = 1e-6 # minimum error to stop
rel = 0.25 # relaxation factor (learning rate)
err = eps + 1 # error
while err > eps:
        err = 0 # initialize error
        ne = 0 # number of errors
        For s=0..N_T−1: # scan the training sets
                t = T_s^0 # pick the input set
                # Assign input set to the first layer:
                For i=0..N_0−1:
                        v_i^0 = t_i
                # Forward propagate:
```

```
        For l = 0..L − 2:
            For i=0..N_{l+1} − 1:
                v = 0
                For j=0..N_l − 1:
                    v += a(v_j^l) w_ij^l + b_i^l
                End
                v_i^{l+1} = v
            End
        End
        # Compute the error:
        e = 0
        For i=0..N_L − 1:
            e += (a(v_i^{L−1}) − a(o_i^t))^2
            ne += 1
        End
        err += e
        # Assign output to the last layer:
        t = T_s^1 # pick the output set
        For i=0..N_L − 1:
            v_i^{L−1} = t_i
        End
        # Back-propagate:
        For l=L − 2..0:
            For i=0..N_{l+1} − 1:
                p = Σ_{k=0}^{N_l} a(v_k^l) w_ik^l + b_i^l
                d = rel · (a(v_i^{l+1}) − a(p)) · a'(p)
                For j=0..N_l − 1:
                    w_ij^l += d a(v_j^l)
                End
                b_i^l += d
            End
        End
    End
    err = err^{1/2}/ne
End
```