$z = W^{[1]} x + b^{[1]}$

$a = \text{ReLU}(z) \longrightarrow a = g(z)$ — Non-linear link function

$h_\theta(x) = W^{[2]} a + b^{[2]}$

Algo 25 – TWOLAYERNETWORKPREDICT $(W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}, x)$

```
# hidden-units = m
for i=1 to m do
    a_i <- g(w_i^[1] . x + b_i^[1])     // activation of hidden unit i
                                        // x ∈ R^d
end for
return W^[2] a + b^[2]                  // compute output unit
                                        // a = [a_1 a_2 .... a_m]^T ∈ R^m
```

Backpropagation ⟶ GD + chain rule

✓ optimize weights in network to minimize some objective function

Linear predictor	NN non-linear predictor
$\hat{y} = w \cdot x$	$\hat{y} = W^{[2]} \cdot \text{ReLU}(W^{[1]} \cdot x)$

Objective

$$\min_{W^{[1]}, W^{[2]}} \sum_n \frac{1}{2}\Big(\underbrace{y_n - \sum_i w_i^{[2]} g(w_i^{[1]} \cdot x_n)}_{e_n}\Big)^2 \quad \text{— } L(W)$$

Case 1 Gradient w.r.t $W^{[2]}$

✓ Similar to linear case

$$\nabla_{W^{[2]}} = -\sum_n e_n a_n \qquad (6)$$

Case 2 inner layer

$$L(W) = \frac{1}{2}\Big(y - \sum_i w_i^{[2]} \cdot g(w_i^{[1]} \cdot x)\Big)^2$$

$$\frac{\partial L}{\partial w_i^{[1]}} = \frac{\partial L}{\partial g_i} \cdot \frac{\partial g_i}{\partial w_i^{[1]}} = \nabla_{w_i^{[1]}} \qquad \text{// chain rule}$$

$$\frac{\partial L}{\partial g_i} = -\Big(\underbrace{y - \sum_i w_i^{[2]} g(w_i^{[1]} x)}_{\text{error, } e}\Big) w_i^{[2]}$$

$$= -e\, w_i^{[2]}$$

$$\frac{\partial g_i}{\partial w_i^{[1]}} = g'(w_i^{[1]} \cdot x)\, x$$

$$\text{So,} \quad \nabla_{w_i^{[1]}} = -e\, w_i^{[2]} g'(\underbrace{w_i^{[1]} \cdot x}_{z_i})\, x \qquad (11)$$

Algo 26: TWOLAYERNETWORKTRAIN $(D, \alpha, M, \text{MaxIter})$

```
W^[1] <- D x M matrix of small random values
W^[2] <- M-vector of small random values
for iter = 1 ..... MaxIter do
    G <- D x M matrix of zeros
    g <- M-vector of zeros
    for all (x,y) ∈ D do
        for i = 1 to M do
            z_i <- w_i^[1] . x
            a_i <- tanh(z_i)          // g is tanh
        end for
        ŷ <- w^[2] . a
        e <- y - ŷ
        g <- g - e . a                 // from eq. 6
        for i = 1 to M do
            G_i <- G_i - e w_i^[2] (1 - tanh^2(z_i)) . x     // from eq. 11
        end for
    end for
    W^[1] <- W^[1] - α . G
    W^[2] <- W^[2] - α . g
end for
return W^[1], W^[2]
```