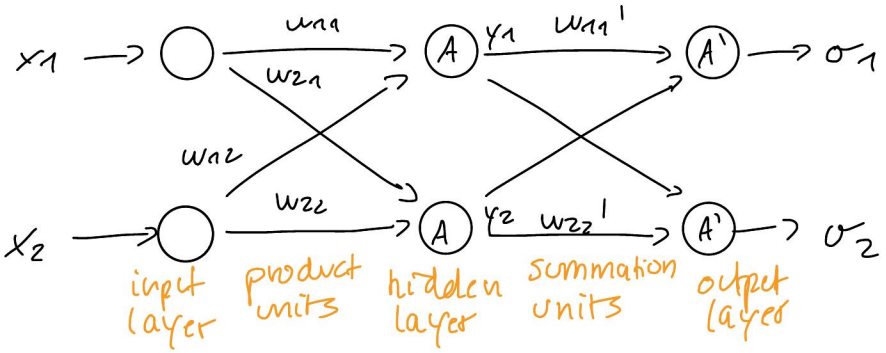
- output o_k:

$$o_k = A'(net_k') = A'\left(\sum_{j=1}^{m+1} w_{kj}' y_j\right) = A'\left(\sum_{j=1}^{m+1} w_{kj}' A(net_j)\right)$$

$$o_k = A'\left(\sum_{j=1}^{m+1} w_{kj}' A\left(\sum_{i=1}^{n+1} w_{ji} x_i\right)\right)$$

2.3.4 Product unit NNs

- weighted product instead of sum of input signals
- advantageous, if couplings between input variables occur



→ product unit:

$$net_j = e^{\sum_{i=1}^{2} w_{ji} \ln(x_i)}$$

$$net_j = e^{w_{j1}\ln(x_1) + w_{j2}\ln(x_2)} = e^{w_{j1}\ln(x_1)} \cdot e^{w_{j2}\ln(x_2)} = x_1^{w_{j1}} \cdot x_2^{w_{j2}}$$

→ old summation unit btw. hidden and output layer:

$$net_k' = \sum_{j=1}^{2} w_{kj}' y_j$$

2.3.5 Gradient descent learning rules

- goal: minimize an error fct. denoting the difference btw. desired and predicted output $d_i - o_i$
- error function:

$$E = \frac{1}{2} \sum_{p=1}^{p_T} (d_p - o_p)^2$$

→ number of desired-predicted output pairs p_T

→ searching for the minimum of $E = E(\underline{w})$

$$\Delta \underline{w} = -\eta \frac{dE}{d\underline{w}}$$

→ learning rate η as proportional constant