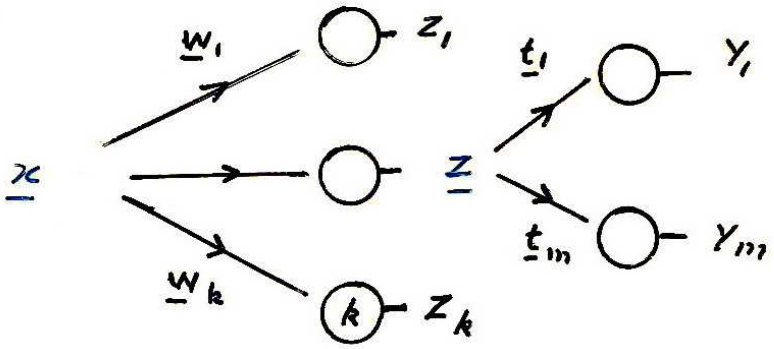
Outras redes feedforward

1 - Rede de Perceptrons - classificadores

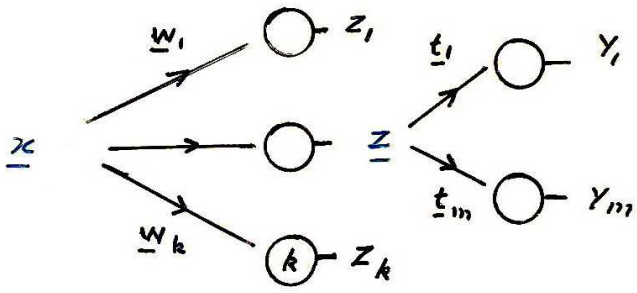


$y_i \longrightarrow \pm 1$ $\quad$ $z_i \longrightarrow \pm 1$

Como saturar os neurônios ?

A rede é treinada

com neurônios tipo tgh (.)



$y \in \{-1, 1\}$ $\quad$ $\tilde{y} \longrightarrow \pm 1$

$\tilde{y} = tgh\ v$ $\quad$ $v \longrightarrow \pm \infty$

$v = \underline{w}^t \underline{x} + b$ $\quad$ $\underline{w}, b \longrightarrow \infty$

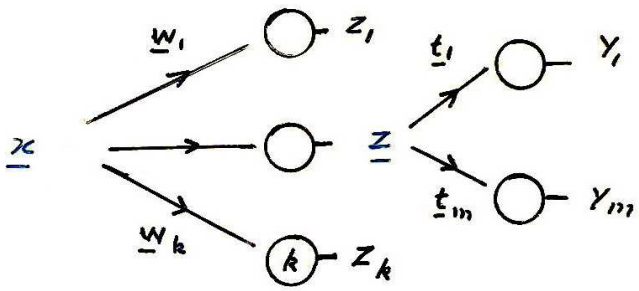
a saída lógica sempre satura !

E a camada intermediária ?

Para $\tilde{y} \to \pm 1 \qquad u_{saída} \to \pm\infty$

logo $\underline{w}_{saída}, b_{saída} \to \pm\infty \qquad e$

$z_i \to \max \quad z_i \to \pm 1,$



logo os neurônios da camada intermediária também saturam,

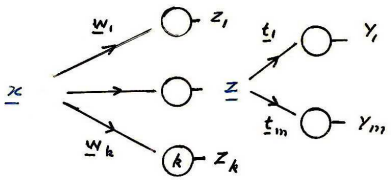
mas muito mais lentamente

Como resolver ?

1 - Usar entropia relativa como função objetivo F

(apresenta os mesmos minimantes que a e.m.q)

$$F = E \sum_l \left\{ (1+y) \ln \frac{1+y}{1+\tilde{y}} + (1-y) \ln \frac{1-y}{1-\tilde{y}} \right\}$$

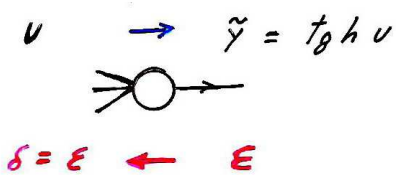


$\Rightarrow$ para a última camada (e apenas esta)

$$\delta_l = \varepsilon_l$$

Camada de saída

A única diferença é que $\frac{d\tilde{y}}{du} = 1$



2 – Alterar a função objetivo

$$F = F_0 + \gamma F_1$$

$$F_1 = \sum_{i=1}^{k} (1 - z_i^2)$$

$$F_1 = \sum_{i=1}^{k} (1 - |z_i|)$$

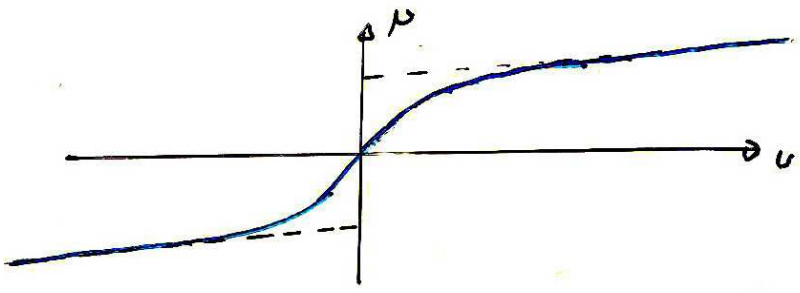
Obs: A rede com perceptrons com entradas e saídas lógicas também é um aproximador universal para funções lógicas. O caminho para uma prova simples é lembrar que qualquer função lógica pode ser realizada por um "ou" de mintermos ("e"s) ou um "e" de maxtermos ("ou"s), isto é, em duas camadas. E que "e"s e "ou"s são realizáveis com neurônios.

2 - Rede sem paralisia de treinamento

Camada intermediária

usar a função de ativação

$$\nu_i = .1\, v_i + .9 \tanh v_i$$



$$\frac{d\nu}{dv} = 1 - .9 \tanh^2 v = 1 - \frac{(\nu - .1\, v)^2}{.9}$$

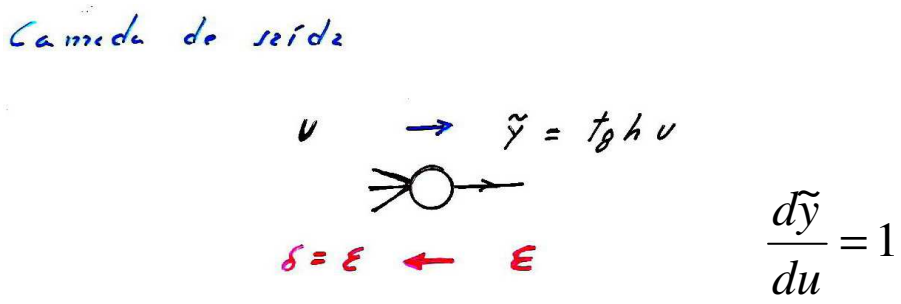
"Quick Prop"

Camada de saída em aproximadores: neurônio linear, não paralisa

Camada de saída em classificadores

Usar entropia relativa como função objetivo.

O neurônio satura mas o treinamento não paraliza

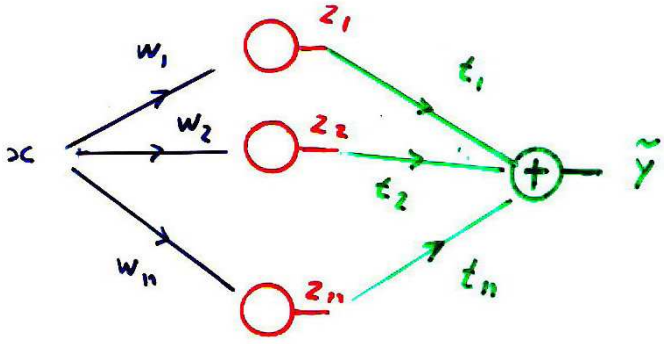


$$\frac{d\tilde{y}}{du} = 1$$

3 – RBF – Redes de Base Radial

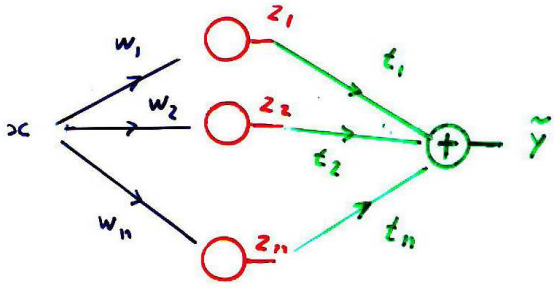
$$u = \left| \underline{x} - \underline{w} \right|^2$$

$$z = e^{-\frac{u}{2\sigma^2}}$$



4 - SVM – Máquinas de Vetor Suporte

$$\tilde{y} = \sum_j \varphi_j(\vec{x})$$



Polinomial $\varphi_j = P_j(\vec{x})$

Perceptrons $\varphi_j = t_j tgh\left(\vec{w}_j^t \vec{x} + w_{j0}\right)$

Base radial gaussiana $\varphi_j = t_j \exp(-\left\|\vec{x} - \vec{w}_j\right\|^2 / 2\sigma_j^2)$

etc. $\varphi_j = \ldots$

mas o treinamento é diferente.

5 – Deep Learning