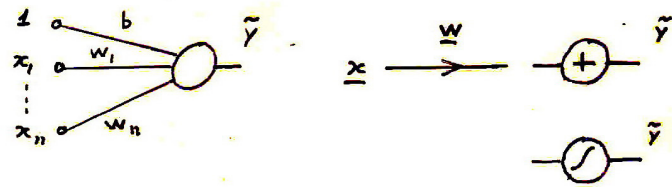
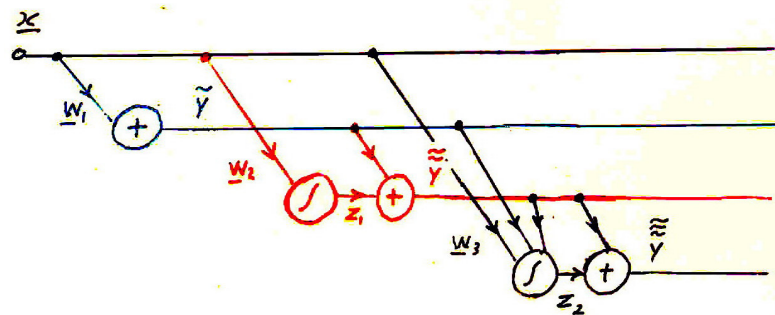


Outras redes feedforward

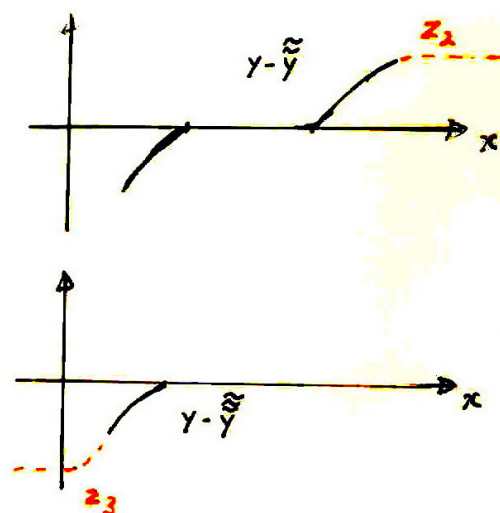
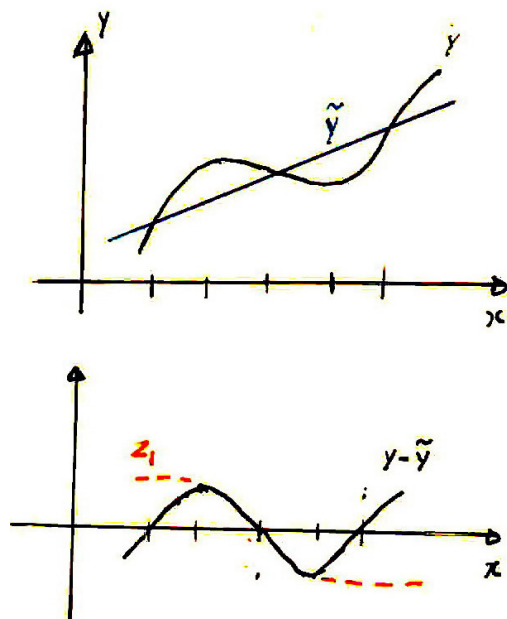
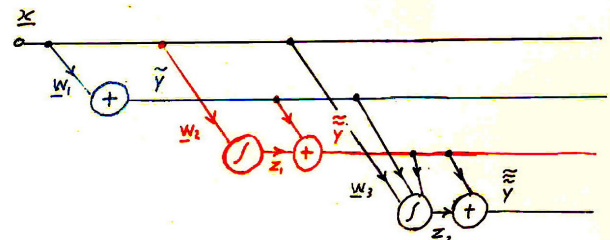
1 - Algoritmos Construtivos – SONN – Self Organizing Neural network



$$y = f(x)$$

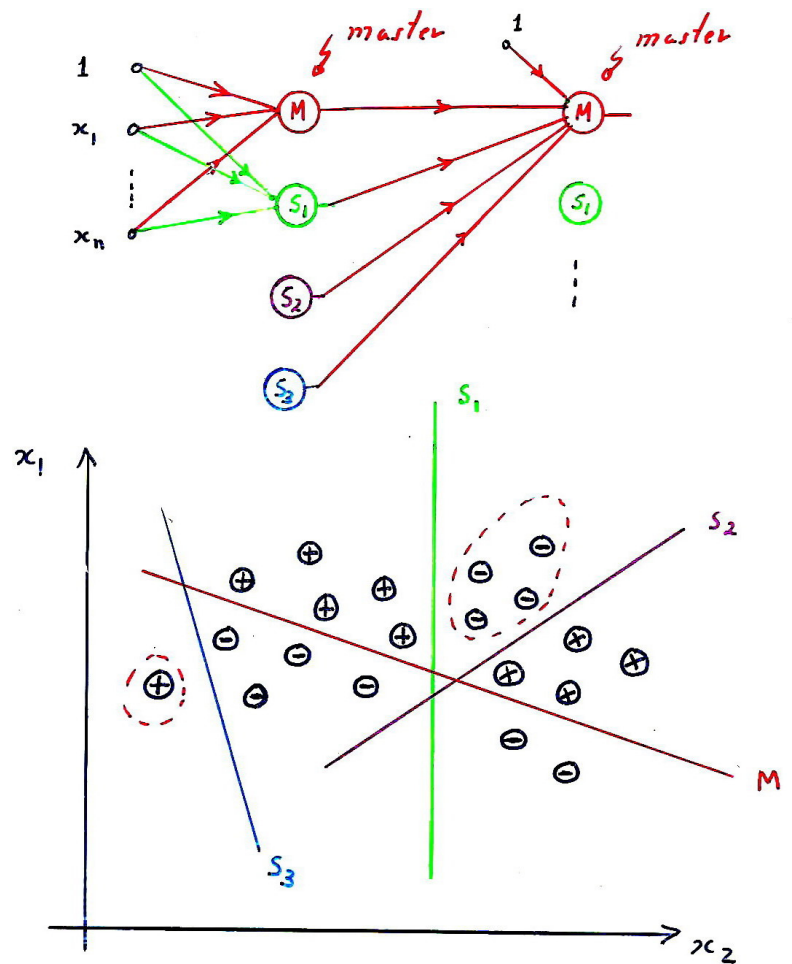


$$y = f(x)$$

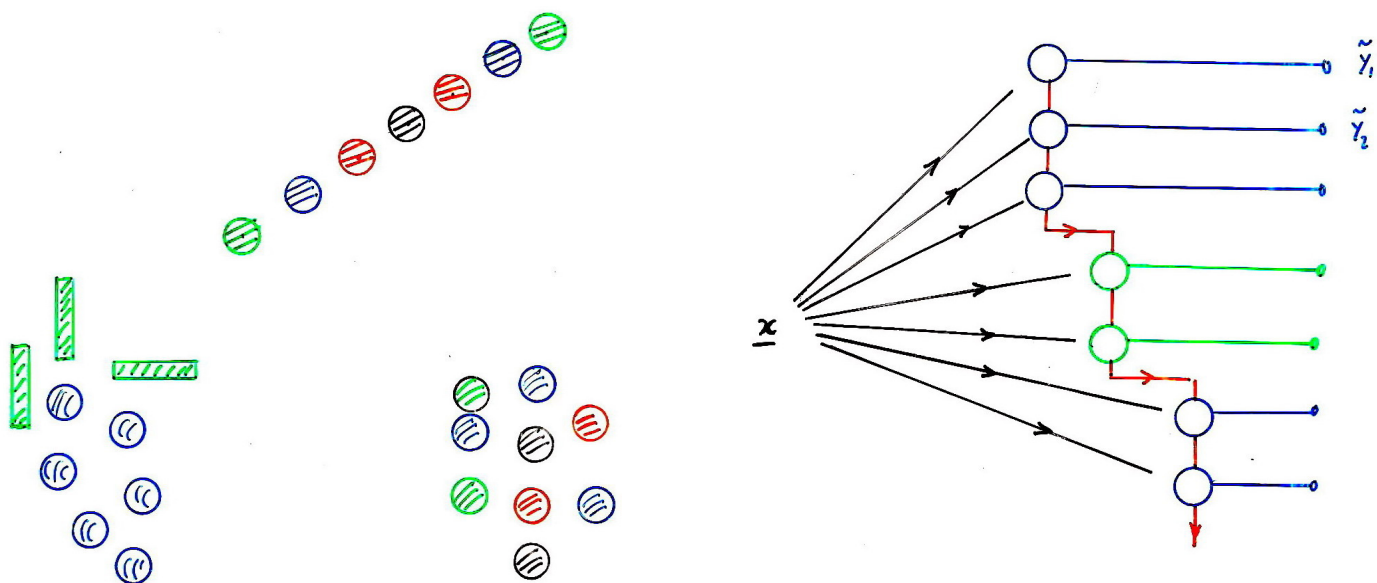


2 - Algoritmos Construtivos

Tiling - classificador



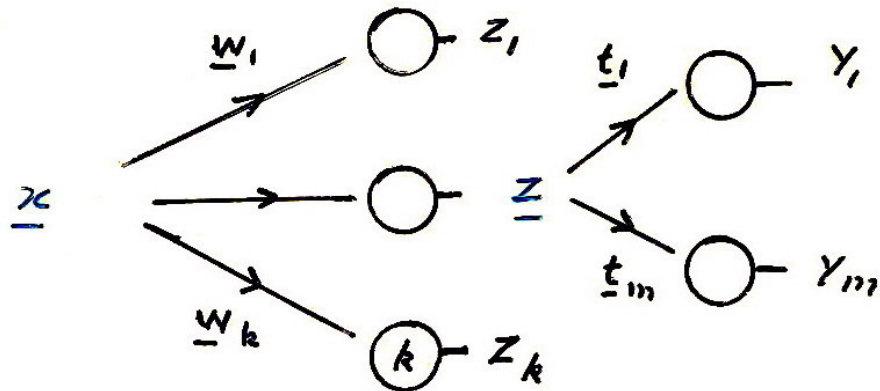
3 - Algoritmos construtivos – Classificador hierárquico



é como descascar uma cebola

4 - Rede de Perceptrons - classificadores

$$f_g h \rightarrow \text{step}$$



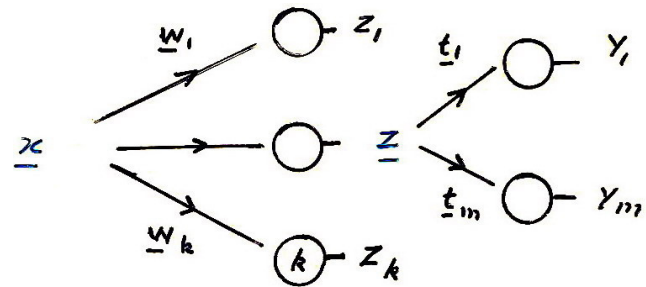
$$y_i \rightarrow \pm 1$$

$$z_i \rightarrow \pm 1$$

Como saturar os neurônios ?

A rede é treinada treinada

com neurônios tipo tgh (.)



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

$$\tilde{y} = f_g h v \quad v \rightarrow \pm \infty$$

$$v = \underline{w}^t \underline{p} + b \quad \underline{w}, b \rightarrow \infty$$

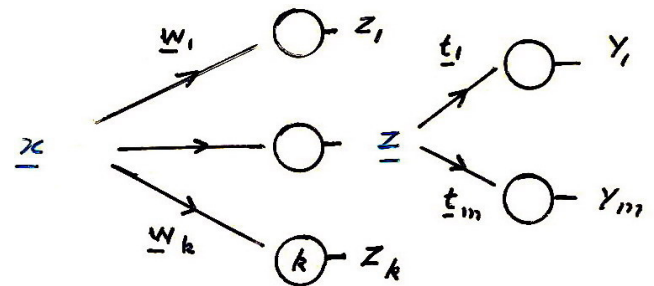
a saída logica sempre satura!

E a camada intermediária ?

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

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

$z_i \rightarrow \max$ $z_i \rightarrow \pm 1$,



logo os neurônios da camada intermediária também saturam,

mas muito mais lentamente

Como resolver ?

1 - Usar entropia relativa como função objetivo F

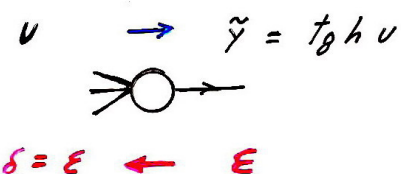
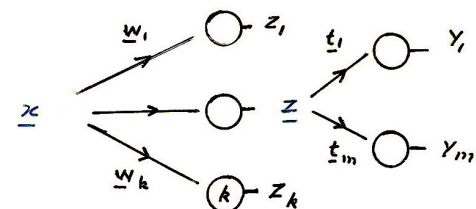
(apresenta os mesmos minimantes que a e.m.q)

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

$\Rightarrow$ 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

(como na poda)

$$F = F_0 + \delta F_1$$

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

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

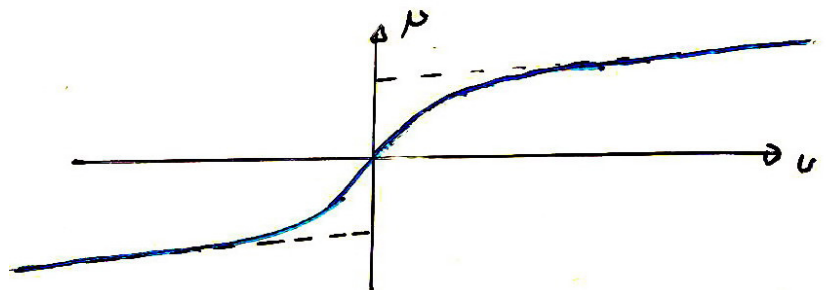
Obs: A rede com perceptrons com entradas e saídas lógicas também é um aproximador universal para funções lógicas. O caminho para uma prova simples é lembrar que qualquer função lógica pode ser realizada por um “ou” de mintermos (“e”s) ou um “e” de maxtermos (“ou”s), isto é, em duas camadas. E que “e”s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

5 - Rede sem paralisia de treinamento

Camada intermediária

usar a função de ativação

$$v_i = .1 u_i + .9 \tanh u_i$$



$$\frac{dv}{du} = 1 - .9 \tanh^2 u = 1 - \frac{(v - .1u)^2}{.9}$$

"Quick Prop"

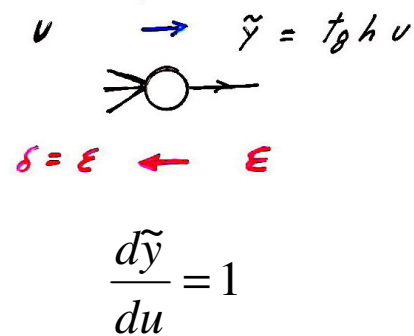
Camada de saída em aproximadores: neurônio linear, não paralisa

Camada de saída em classificadores

Usar entropia relativa como função objetivo.

O neurônio satura mas o treinamento não paralisa

Camada de saída



6 - SVM – Máquinas de Vetor Suporte

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

Polinomial

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

Perceptrons

$$\varphi_j = t_j \tanh(\vec{w}_j^t \vec{x} + w_{j0})$$

**Base radial
gaussiana**

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

etc.

$$\varphi_j = \dots$$

mas o treinamento é diferente.