

# RADIAL BASIS FUNCTION NETWORK

Sono reti neurali che hanno tre layer:

- 1) Input layer
- 2) Hidden layer  $\rightarrow$  Radiale
- 3) Output layer  $\rightarrow$  Lineare

Il livello nascosto presenta una funzione di attivazione di tipo **radial basis**, come la gaussiana. Più l'ingresso al neurone è vicino alla media  $\mu$  con cui è costruita la gaussiana, più l'attivazione del livello successivo è forte.

## RADIAL FUNCTION

In generale questo tipo di funzioni restituiscono un valore reale, proporzionale alla distanza dell'input con un punto detto centro della funzione.

## Gaussiana

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Per quello che ci interessa ci basta usare la formula non normalizzata:

$$g(x, \mu, \sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Chiamiamo ora  $\varepsilon^2 = 1/2\sigma^2$ , così da semplificare la formula:

$$g(x, \mu, \sigma) = e^{-\varepsilon^2 (x - \mu)^2}$$

La gaussiana, vista come funzione di attivazione delle RBFs, ha la seguente espressione:

$$g(x, z) = e^{-\varepsilon^2 z^2} \quad \text{con } z = (x - \mu)^2$$

La  $z$  la possiamo vedere come dipendente da  $w$  e  $x$ , dove il nostro peso è la media  $\mu$  della gaussiana:

$$z(w, x) = (w - x)^2$$

Passando al caso multivariato, dove quindi abbiamo a che fare con vettori in ingresso, abbiamo che:

$$z(w, x) = \sqrt{\sum_{d=1}^D (w_d - x_d)^2}$$

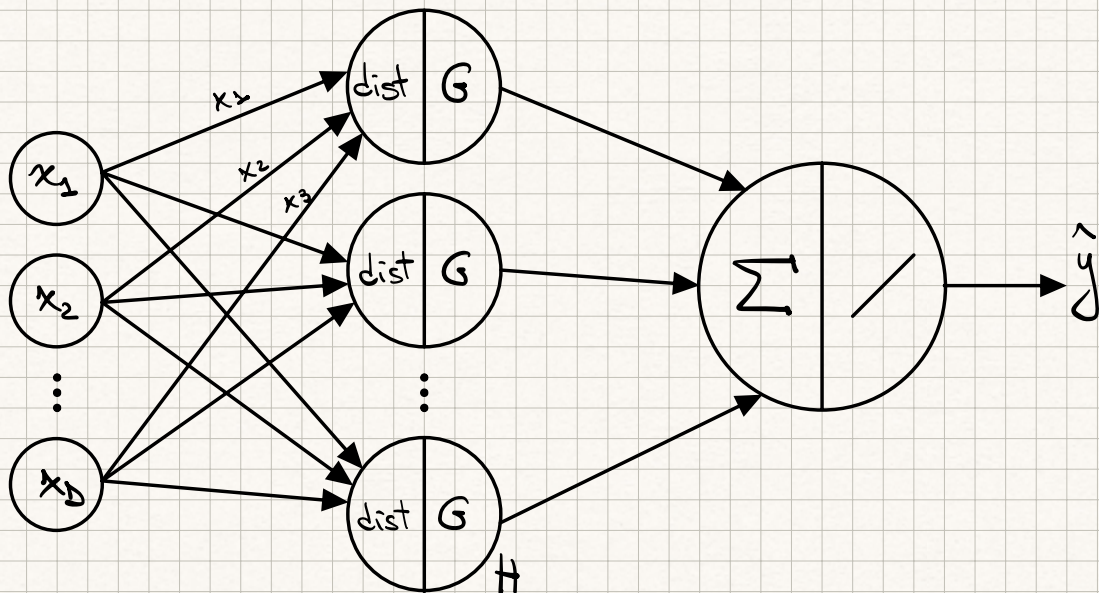
Distanza euclidea.

L'espressione della gaussiana nel caso multivariato diventa:

$$g(\vec{x}, \vec{w}, \varepsilon) = e^{-\varepsilon \sum_{d=1}^D (w_d - x_d)^2}$$



## MODELLO



## Forward

$$zA_0 = x^T$$

$$zZ_2 = \text{dist}(W_L, zA_0) \rightarrow \text{Distanza euclidea}$$

$$zA_1 = e^{-\varepsilon^2 \cdot (zZ_1 \cdot zZ_1)}$$

$$A_1 = \text{extended}(zA_1)$$

$$zZ_2 = W_2 \cdot A_1 \rightarrow \text{Prodotto scalare}$$

$$zA_2 = zZ_2 \rightarrow \text{Layer lineare}$$

## TRAINING

Il training viene fatto su un batch. Dato che i pesi  $W$  sono i centroidi da associare ai punti del dataset

- 1) **Centroidi**: Si trovano con K-Means, per formare  $W_1$ .
- 2)  **$W_2$** : Possiamo usare i minimi quadrati per approssimare una funzione lineare. Possiamo usare il MSE



# RBF-LAZZARINI

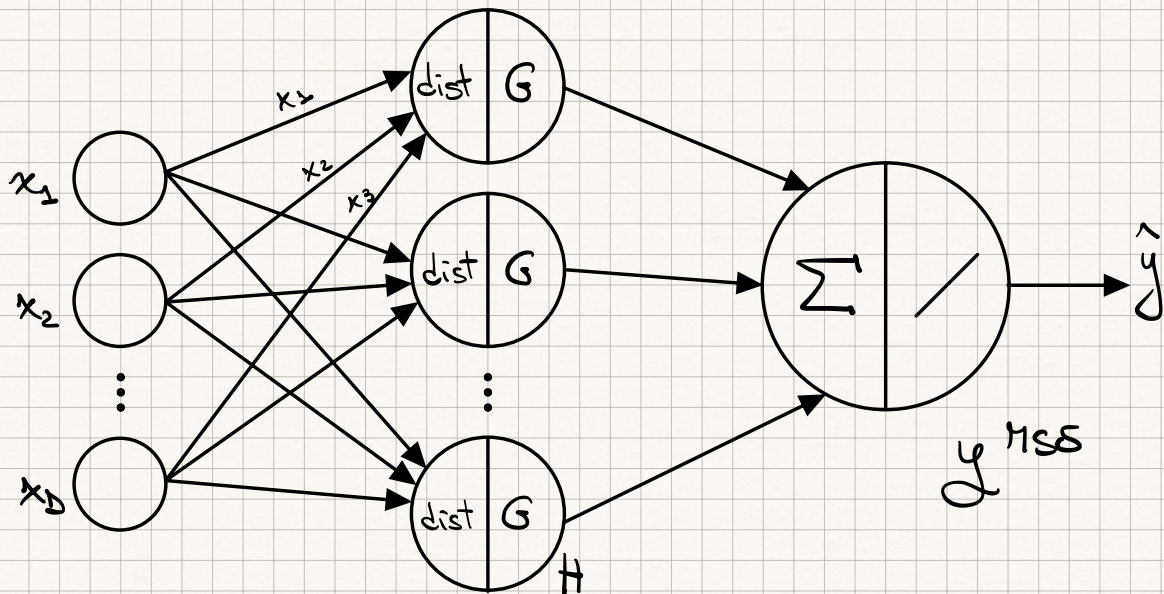
## FUNZIONI RADIALI

Sono funzioni che prendono in ingresso un valore reale e in un'uscita danno un altro valore reale che ci dice quanto l'ingresso è vicino a un punto, detto centro della funzione. Più l'ingresso è vicino al centro, più il valore in uscita è alto.

Esempio: la Gaussiana  $g(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

## RSTI RBF

Sono NN dove l'attivazione dei livelli nascosti è una RBF, mentre l'attivazione in uscita è lineare.




La matrice dei pesi dell'hidden layer fa riferimento ai centri delle RBF, per cui il neurone viene attivato. L'argomento che viene dato in input a una RBF è la **distanza (euclidea)** fra il vettore d'ingresso  $x$  e il centro  $x_i$  di quella RBF:

$$A_j = \|x - w_j\|_2 = \sqrt{\sum_{i=1}^m (x_i - w_{ij})^2}$$

**Note:** Tipicamente in queste reti si ha un solo neurone d'uscita.

### Esempio: IL PROBLEMA XOR

Dove bisogna mappare una coppia di bit in ingresso a un singolo output mostrato in tabella



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

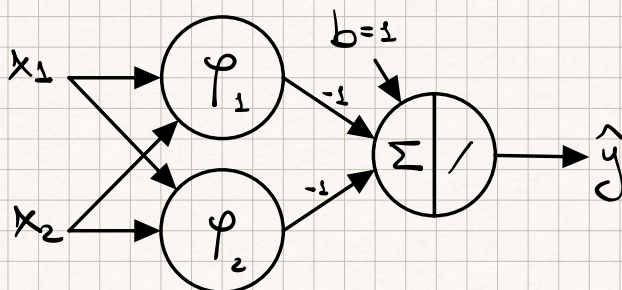
Possiamo risolvere questo problema con una rete RBF che utilizza la Gaussiana.

Usiamo 2 neuroni nascosti, che hanno come centri:

$$t_1 = (1, 1) \quad e \quad t_2 = (0, 0)$$

Da cui costruiamo le due gaussiane con  $\sigma=1$ :

$$\varphi_1(x) = e^{-\|x - t_1\|^2} \quad \varphi_2(x) = e^{-\|x - t_2\|^2}$$



Input x	$\varphi_1(x)$	$\varphi_2(x)$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678
(0,0)	0.1353	1



Per risolvere questo problema, in  $W_2$  per ogni ingresso dell'ultimo layer mettiamo i pesi a -1 così da ottenere come uscita:

$$\hat{y} = -\varphi_1 - \varphi_2 + b = -e^{-\|x-t_1\|^2} - e^{-\|x-t_2\|^2} + 1$$

SE  $\hat{y} > 0$  ALLORA  $(x_1, x_2) \in C_2$  (ovvero 0)  
ALTRIMENTI  $(x_1, x_2) \in C_1$  (ovvero 1).

## LEARNING

Bisogna imparare:

- 1) I centri  $\mu$
- 2) Lo spread  $\sigma$
- 3) I pesi  $W_2$

## LEARNING DEI CENTRI

- 1) Randomicamente nello spazio di training
- 2) Unsupervised - Usando un algoritmo di clustering

## LEARNING DEGLI SPREAD

Una volta noti i centri, lo spread può essere fissato a:

$$\sigma = \frac{\max_{i,j=1,\dots,K} \text{dist}(C_i, C_j)}{\sqrt{2K}}$$

## LEARNING DEI POSSI

Dato che l'uscita di una rete RBF:

$$\hat{y} = \sum w_i \varphi_i(x)$$

Possiamo scriverla in forma vettoriale:

$$\hat{y} = \underbrace{\begin{bmatrix} \varphi_1(x_1) & \dots & \varphi_k(x_1) \\ \vdots & & \vdots \\ \varphi_1(x_m) & \dots & \varphi_k(x_m) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}}_W \approx \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix}}_d$$

$$W = \Phi^{-1} d^T$$

## LEARNING SUPERVISIONATO

Supervised - Usando la discesa del gradiente

$$\delta_{\text{error}} = \frac{1}{2} \sum_{i=1}^m d_i - \underbrace{y(x_i)}_{\text{RBF NN}}$$

$$\text{Centro } t_{i+1} = t_i - \gamma \cdot \frac{\partial \delta_{\text{error}}}{\partial t_i}$$

$$\text{Spread } \sigma_{i+1} = \sigma_i - \gamma \cdot \frac{\partial \delta_{\text{error}}}{\partial \sigma_i}$$

$$\text{Pesi } w_{i+1} = w_i - \gamma \cdot \frac{\partial \delta_{\text{error}}}{\partial w_i}$$