# Redes de funciones de base radial

Luis Ballado

CINVESTAV - UNIDAD TAMAULIPAS

*luis.ballado@cinvestav.mx*

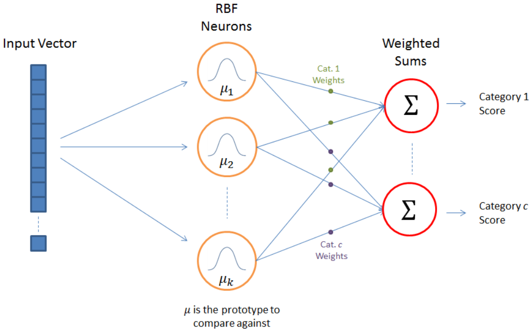
February 26, 2023

# Contenido


1. Introducción
2. Paradigmas de Aprendizaje
3. El proceso de aprendizaje de la red RBF
4. Algoritmo
5. Implementación Python

# Introducción

Las redes neuronales de base radial están compuestas de tres capas: **capa de etrada, capa intermedia ó oculta**, en donde la función de activación de sus neuronas es Gaussiana, y **la capa de salida** con función de activación lineal (Perceptrón)



# Introducción

Las funciones $\phi$ determinan las activaciones de las neuronas de la capa oculta en función de un muestra m.

**Activación de las neuronas $\phi$**

Utilizando una $\phi$ Gaussiana la activación de las nueronas ocultas sería: $\phi = e^{\frac{-\sum_{i=1}^{p}(x_i - c_{ij})^2}{2\sigma_j^2}}$

- p: Total de características
- C: Centroide perteneciente a la neurona oculta j
- $\sum_{i=1}^{p}(x_i(n) - c_{ij})^2$ : Distancia euclidiana del vector de entrada y el centroide $C_j$
- $\sigma^2$ Varianza

## Salida de la red RBF (y)

$$y = \sum_{i=1}^{n} W_i * \phi_i - \theta$$

Donde:

- n: neuronas en la capa oculta
- m: patrón de entrada (seudomuestra)
- $W_i$: pesos sinápticos que conectan las neuronas de la capa oculta y la neurona de salida y
- $\theta$: Umbral de la neurona de salida
- $\phi_i$: Seudomuestra

En resumen:
Para implementar una red RBF necesitamos:

1. Un conjunto de muestras $\rightarrow$ X (sólo para la etapa de aprendizaje)
2. Los centroides $\rightarrow$ C
3. El valor de la variaza $\rightarrow \sigma$
4. Las pseudomuestras $\rightarrow$ Z
5. El valor de los pesos sinápticos $\rightarrow$ w

# El proceso de aprendizaje de la red RBF

Este proceso se suele llamar híbrido ya que se divide en dos fases:

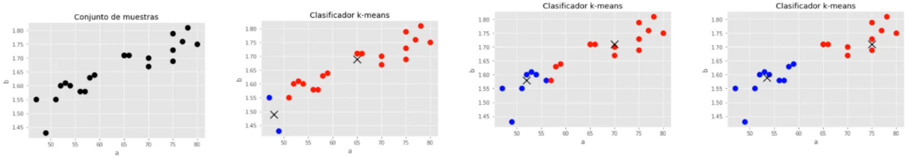
- **Fase no supervisada**
  Se encarga de obtener el valor de los centroides C y la varianza $\sigma$.
  Para esto se suele utilizar el método **k-means**
- **Fase supervisada**
  Consiste en calcular los pesos w y umbral $\theta$ de las neuronas de la capa de salida. Para esto se suele utilizar el método de la **Regla Delta** o bien el método de la **matriz seudoinversa**

Terminología:

- m: Cantidad total de muestras
- X: Conjunto de muestras para el entrenamiento $\rightarrow \{X_0, X_1, X_2, ..., X_m\}$
- Z: Conjunto de seudomuestras $\rightarrow \{Z_0, Z_1, Z_2, .., Z_m\}$
- $\phi$: Salida de una neurona en la capa oculta
- y: Salida de una neurona en la capa de salida
- w: Pesos sinápticos que enlazan la información de las neuronas de la capa oculta con las de la capa de salida.

# Fase no supervisada

**Clasificador k-means**, ejemplo
Estableciendo 2 centroides o bien dos neuronas en la capa oculta $\phi_1$, $\phi_2$:



# Obtener pseudomuestras

Una vez obtenidos los valores de centroides y varianza:

Generar pseudomuestras

- Obtener el conjunto de muestras X
- Generar la salida de las neuronas en la capa oculta por cada muestra del conjunto X

$$\phi_j = e^{\frac{-\sum_{i=1}^{p}(x_i - c_{ij})^2}{2\sigma_j^2}}$$

- $z_i$: Seudomuestra $\rightarrow \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}$
- Z: Seudomuestras $\rightarrow \{Z_0, Z_1, Z_2, ..., Z_m\}$

# Método por Matriz pseudoinversa

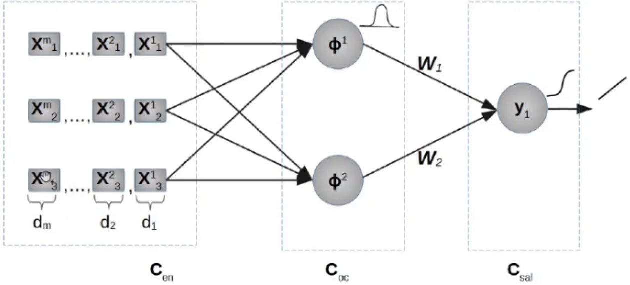
Debido a que la salida de la red depende linealmente de los pesos, es posible proporcionar una solución directa con lo siguiente:

$$w = G^{+} * S \rightarrow (G^{t} * G)^{-1} * G^{t} * S$$

Donde:
$(G^{t} * G)^{-1} * G^{t}$, es la matriz pseudoinversa de G
$G^{t}$, es la matriz transpuesta de G
G, es una matriz de tamaño ($X^{m}$,$n_{oc}$) que contiene las activaciones de las neuronas de la capa oculta pero los patrones de entrada
S, es la matriz de salidas deseadas, de tamaño ($X^{m}$,$n_{csal}$)

# Representación

El indice superior m indica el número de muestras de aprendizaje




$$G = \begin{pmatrix} \phi_1^1 \phi_1^2 \\ \phi_2^1 \phi_2^2 \\ \phi_m^1 \phi_m^2 \end{pmatrix}, S = \begin{pmatrix} d_1 \\ d_2 \\ d_m \end{pmatrix}, w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Suponiendo que se tienen los siguientes valores de G y S:

$$G = \begin{pmatrix} 11 \\ 12 \\ 13 \end{pmatrix}, S = \begin{pmatrix} 2 \\ 2 \\ 4 \end{pmatrix}$$

Paso 1:
Calcular $(G^t * G)$

$$P_1 = \begin{pmatrix} 111 \\ 123 \end{pmatrix} \begin{pmatrix} 11 \\ 12 \\ 13 \end{pmatrix} = \begin{pmatrix} 36 \\ 614 \end{pmatrix}$$

Paso 2:

Calcular la inversa $(G^t * G)^{-1} \rightarrow (P_1^{-1})$
Utilizado el método de Gauss-Jordan:
$R1 * \frac{1}{3} = \begin{pmatrix} 12|\frac{1}{3}0 \\ 614|01 \end{pmatrix} \rightarrow R2 - 6R1 = \begin{pmatrix} 12|\frac{1}{3}0 \\ 02|-21 \end{pmatrix} \rightarrow R1 - R2 = \begin{pmatrix} 10|\frac{7}{3}-1 \\ 02|-21 \end{pmatrix} \rightarrow R2 * \frac{1}{2} = \begin{pmatrix} 10|\frac{7}{3}-1 \\ 01|-1\frac{1}{2} \end{pmatrix}$
Entonces $(G^t * G)^{-1}$ es:

$$P2 = \begin{pmatrix} \frac{7}{3}(-1) \\ (-1)\frac{1}{2} \end{pmatrix}$$

# utilizado el método de determinantes

$$(P_1)^{-1} = \frac{1}{|P_1|} * (P_1^*)^t$$

Si $P_1 = \begin{pmatrix} 36 \\ 614 \end{pmatrix}$ Entonces:

$|P_1| = \begin{pmatrix} 36 \\ 614 \end{pmatrix} \rightarrow 42 - 36 = 6$ $P_1^* = \begin{pmatrix} 14 - 6 \\ -63 \end{pmatrix} \rightarrow (P_1^*)^t = \begin{pmatrix} 14 - 6 \\ -63 \end{pmatrix}$

$\frac{1}{|P_1|} * (P_1^*)^t = \frac{\begin{pmatrix} 14 - 6 \\ -63 \end{pmatrix}}{6} \rightarrow \begin{pmatrix} \frac{7}{3} - 1 \\ -1\frac{1}{2} \end{pmatrix}$

Paso 3: Calcular $((G^t * G)^{-1} * G^t) \rightarrow (P_2 * G^t)$:

$$P_3 = \begin{pmatrix} \frac{7}{3} & -1 \\ -1 & \frac{1}{2} \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} \frac{4}{3} & \frac{1}{3} & \frac{-2}{3} \\ \frac{-1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

Paso 4: Obtener w por medio de $P_3 * S$ :

$$w = \begin{pmatrix} \frac{4}{3} & \frac{1}{3} & \frac{-2}{3} \\ \frac{-1}{2} & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 4 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ 3 \\ 1 \end{pmatrix}$$

Por lo tanto:

$$w = G^{+} * S = \begin{pmatrix} \frac{2}{3} \\ 1 \end{pmatrix}$$

# Aprendizaje de la red RBF, por Regla Delta

Paso 1 - Consiste en obtener la salida de la red ($y$) mediante:

$$y = \sum_{i=1}^{n} (w_i * \phi_i) - \theta$$

Paso 2 - Consiste en corregir el valor de los pesos w dependiendo del error cometido por la red dada una muestra i:

$$e_i = (d_i - y_i)$$

Evaluación: Consiste en evaluar el error global cometido por la red, y en caso de cumplir con un umbral de precisión $\epsilon$ termina el entrenamiento:

$$e_{medio} = \frac{1}{2}(e_i)^2 \quad E = \frac{1}{p}\sum_{k=1}^{p} e_{medio}$$

# Regla Delta

Paso 1: Inicialización

Tomando en cuenta que ya se realizo el proceso de generar las seudomuestras $\phi$ representadas por la matriz Z

- Inicializar el valor de los pesos w con valores aleatorios pequeños
- Inicializar el valor de $\theta = 1$; $\mu, \epsilon$ y épocas.

Paso 2: Generar la salida de la red

- Suponiendo que se tienen los siguientes valores:

$$zi = \begin{pmatrix} 0,5 \\ 0,4 \end{pmatrix}, w = \begin{pmatrix} 0,2 & 0,8 \end{pmatrix} \lambda = 0.2$$

- Obtener la salida de la neurona en la capa de salida (y) dada una seudomuestra $z_i$:

$$I = \sum_{i=1}^{n} (w_i * \phi_i) - \theta \rightarrow \begin{pmatrix} 0,2 & 0,8 \end{pmatrix} \begin{pmatrix} 0,5 \\ 0,4 \end{pmatrix}$$

$$= (0,5 * 0,2) + (0,4 * 0,8) = 0,42 - 1 = -0,58$$

- Aplicar una función de activación (tanh o sigmoide):

$$y = tanh(I) \rightarrow tanh(-0,58) = -0,52$$

# Regla Delta

Paso 3: Ajustar los pesos sinápticos w

- Obtener el error e de la red dada una muestra $\phi$:
  (d-y) $\rightarrow$ (1-(-0,52)) = 1,52
- Obtener $\delta$: $e * \frac{\partial I}{\partial y} \rightarrow e * \frac{1}{1+tanh(y)^2} \rightarrow 1,52 * 0,81 = 1,23$
- Ajustar los pesos w:
  $w + (\lambda * \delta * \phi^t) \rightarrow (0,2 \quad 0,8) + (0,2 * 1,23 * (0,5 \quad 0,4))$

  $$(0,5 \quad 0,8) + (0,122 \quad 0,098) = (0,652 \quad 0,898)$$
- Ajustar $\theta$: $\theta + (\lambda * \delta)$

Paso 4: Evaluación

- Evaluar el error cometido por la red por cada iteración
  $e_i = \frac{1}{2} * (d_i - y_i)^2$

- Al termino de evaluar todas las muestras, calcular el error global:

$$E_g = \frac{1}{p} * \sum e_i$$

**Algorithm 1:** Fase no supervisada, k-means

linenosize= Obtener el conjunto de muestras de entrenamiento
$X \rightarrow \{X^1, X^2, X^3, ..., X^m\}$, donde $X^m \rightarrow \{x_1, x_2, x_2, ...x_n\}$;
Determinar el número de conjuntos k;
Dar un valor inicial aleatorio a los k centroides c, tomando las primeras k muestras de X;
Indicar el umbral de cambio para los centroides u;
**for** *máx iteraciones* **do**
    **for** *todas las muestras de X* **do**
        Calcular la distancia Euclidiana entre cada c y $X^m$;
        Dado un vector $X^m$, determinar a cuál centroide c se encuentra más cercano ;
        Atribuir $X^m$ al grupo $\Omega^c$;
    **end**
    Determinar el cambio de posición de cada centroide c $\rightarrow$ cambio;
    **if** *u>cambio* **then**
        terminar ciclo;
    **end**
**end**
**for** *todo c* **do**
    Calcular la variaza de cada función de activación Gaussiana usando el criterio de los minimos cuadrados:
    $\sigma^2 = \frac{1}{p} \sum_{x^k \Omega^j} \sum_{i=1}^{n} (X_j^k - C_i)^2$, donde p es el número de muestras ;
**end**

## **Algorithm 2:** Fase supervisada, Perceptrón - Regla Delta

linenosize= Obtener las muestras de entrenamiento X;
Obtener las salidas deseadas para cada muestra d;
Inicializar los pesos w con valores aleatorios pequeños;
Especificar la tasa de aprendizaje $\lambda$, el número de épocas máximo y la precisión $\epsilon$ de la red;
**for** *todas las muestras X* **do**

Obtener $\phi$ de cada neurona oculta con respecto a $\Omega^c \rightarrow \phi_{n1} = e^{\frac{-\sum_{i=1}n(x_i(n)-c_{n1})^2}{2\sigma_c^2}}$;
Generr seudomuestras $Z \rightarrow z = \{\phi(1), \phi(2), ..., \phi(n1)\}$;

**end**
**while** $\epsilon > E$ **do**

$E_{prev} = E_g$;
**for** *seudomuestra de Z* **do**

Obtener la salida de la red $\rightarrow y = \sum_i = 1^n(\phi_i * w_i) - \theta$;
Ajustar $w \rightarrow w + (z_i * \delta * \lambda)$;
Obtener salida de la red con los nuevos w y computar el error cuadrático medio $\rightarrow \frac{1}{2} * (d - y)$;

**end**

**end**
Obtener el error global $E_g \rightarrow \frac{E_{medio}}{p}$ donde p es la cantidad de muestras;
Obtener $E \rightarrow |(E_g - E_{prev})|$;

**Algorithm 3:** Fase supervisada, Perceptrón - Regla Delta

linenosize= Obtener las muestras de entrenamiento $X^m$;
Obtener las salidas deseadas para cada muestra $d^m$;
**for** *todas las muestras* $X^m$ **do**

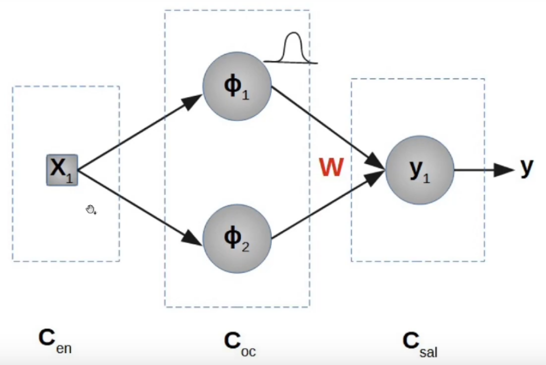
Obtener $\phi$ de cada neurona oculta con respecto a $\Omega^c \rightarrow \phi_{n1} = e^{\frac{-\sum_{i=1} n (x_i(n) - c_{n1})^2}{2\sigma^2_{c_{n1}}}}$;

Generr seudomuestras $G^k \rightarrow$ z = $\{\phi(1), \phi(2), ..., \phi(n1)\}$;

**end**
Obtener matriz de pesos $w \rightarrow G^+ * S$;
**return** w;

# Arquitectura de la red para uso como aproximador de funciones

Figure: Arquitectura de red neuronal de base radial



# Implementación Python

## Aproximador universal de funciones

- Estableciendo $\lambda = 0,015$ y un total de épocas de 5000

Figure: Usando 1 neurona $\phi$

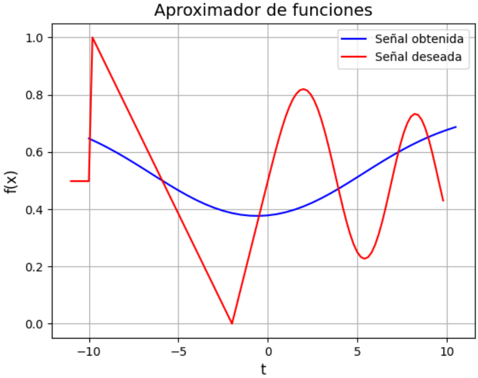


Figure: Usando 3 neuronas $\phi$

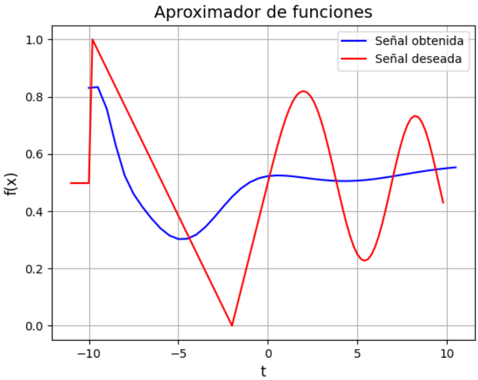


Figure: Usando 4 neuronas $\phi$

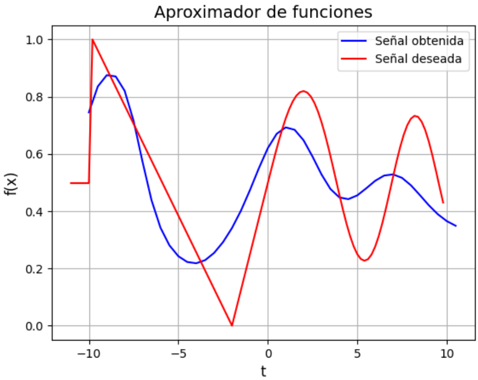


Figure: Usando 6 neuronas $\phi$