# Capítulo 4. Método Propuesto: Self Recurrent Wavelet Neural Network Network - Multidimensional Radial Wavelons (SRWNN-MRW)

En este capítulo se presenta una nueva arquitectura recurrente con wavelets para la implementación de un IDS. En la parte inicial se establecen las características generales del modelo planteado, su arquitectura se describe a través de diagramas del modelo y diagramas de bloque. A partir de esta introducción se desarrollan las características dinámicas del modelo definidas en el algoritmo de aprendizaje, que servirá en la etapa de entrenamiento. Finalmente el algoritmo propuesto es establecido en la parte final de esta sección acompañado de un pseudocódigo y diagrama de flujo.

## 4.1. Arquitectura

La arquitectura propuesta en este trabajo se compone de a partir de nodos de procesamiento llamados MRW (*multidimensional radial wavelon*), estos nodos fueron definidos por Qinghua Zhang [ZHA93] en una arquitectura de alimentación hacia adelante para la aproximación multivariable de funciones. Una de las características principales en la implementación del modelo es el de proponer este tipo de nodo de procesamiento en un contexto recurrente, dando lugar a una nueva arquitectura. Trabajos como [BOR07] exponen las características de aproximación multidimensional de las unidades MRW haciendo que estas sean una mejor elección que las unidades de procesamiento definidas a través el producto tensor de la función wavelet aplicada a cada una de las señales de entrada presentadas al *wavelon*, como por ejemplo, las unidades utilizadas en la capa de producto en el modelo SRWNN. Por esta razón el modelo propuesto incorpora unidades MRW.

En la Figura 4.1 se muestra la arquitectura de una unidad MRW, cada unidad MRW está compuesta por dos módulos, el primero es el de función radial y se denota por la letra $R$, el segundo es una función wavelet unidimensional denotada por $\psi$. El propósito de esta división es procesar el vector de entradas para que se pueda usar por una función wavelet unidimensional [ZHA93].

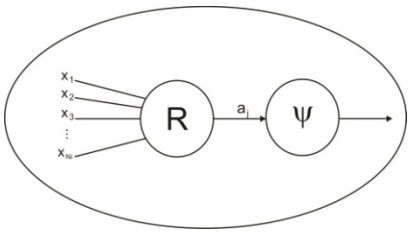


**Figura 4. 1: Unidad MRW**

Dado que hemos definido la unidad elemental de nuestra arquitectura a continuación se presenta en la Figura 4.2 el esquema de conexiones propuesto. La arquitectura se compone de 3 capas, la primera es la capa de entrada, la segunda se trata de la capa multidimensional radial *wavelon* y finalmente la capa de salida compuesta de combinadores lineales.

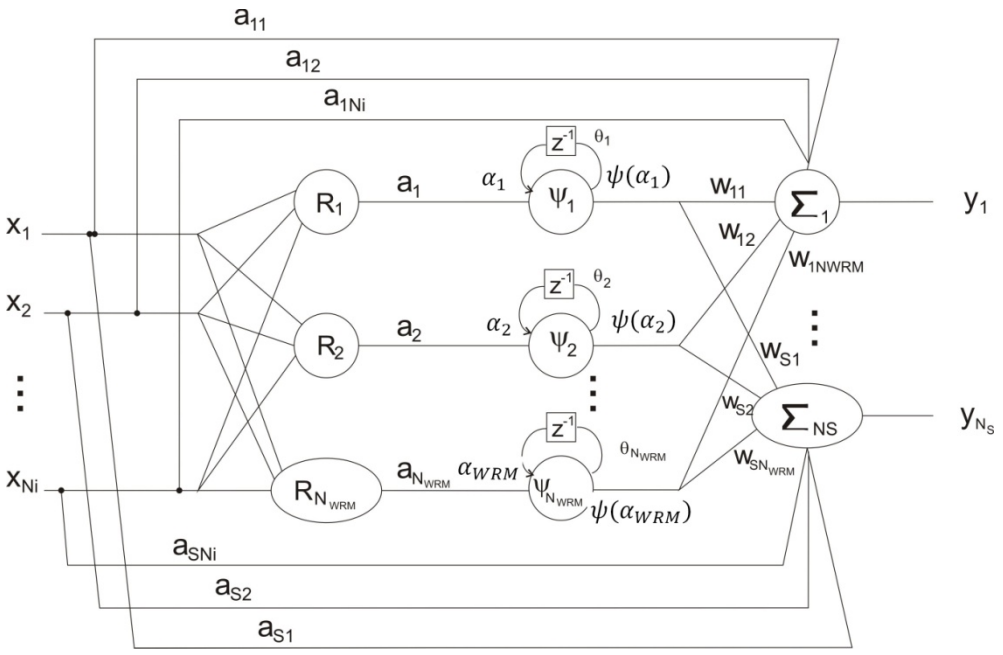


**Figura 4. 2: Arquitectura SRWNN - MRW**

Las entradas externas están representadas por $x_k$, $k = 1 \dots N_i$, siendo $N_i$ el número de entradas definidas para la arquitectura. Las salidas de las unidades $R$ están denotadas por $a_j$, $j = 1 \dots N_{MRW}$, siendo $N_{MRW}$ el número de unidades *wavelon* multidimensionales radiales, estas salidas representan el procesamiento de las líneas de entrada.

Dentro de la capa multidimensional radial, las unidades wavelet poseen una auto conexión que permite introducir el término de memoria $\psi_j(n-1)$ encargado de guardar información sobre estados pasados del modelo, esta conexión recurrente se ve afectada por medio del factor $\theta_j$, el cual representa el peso sináptico de ese enlace. Es decir, la entrada a las unidades wavelet está formada por la salida de las unidades $R$ y el término $\psi_j(n-1)$ multiplicado por un factor $\theta_j$.

Las conexiones directas desde la capa de entrada hasta los combinadores de salida están afectadas por factores $a_{ik}$ $i = 1 \dots N_s$, siendo $N_s$ el número de unidades de salida. Los pesos sinápticos entre las unidades de salida y las unidades wavelet están denotados por $w_{ij}$.

### 4.1.1. Capa de entrada

Esta capa recibe las entradas externas y las transmite directamente a la capa multidimensional radial. Podemos denotar el vector $\boldsymbol{x}(n)$ de entradas externas por medio de:

$$\boldsymbol{x}(n) = [x_1 \dots x_{N_i}] \tag{4. 1}$$

donde $N_i$ representan el número de entradas externas

### 4.1.2. Capa multidimensional radial wavelon

Dentro de esta capa se encuentran las unidades MRW, cada unidad maneja sus propios parámetros de traslación y escalamiento para cada una de las entradas en $\boldsymbol{x}(n)$. Podemos representar a estos parámetros para unidad WRM de forma vectorial como sigue:

$$\boldsymbol{d}_j = [d_{j,1} \dots d_{j,N_i}] \tag{4. 2}$$

$$\boldsymbol{t}_j = [t_{j,1} \dots t_{j,N_i}] \tag{4. 3}$$

$$j = 1 \dots N_{WRM}$$

donde $N_{WRM}$ representa el número utilizado de unidades WRM ubicadas en esta capa.

La principal tarea de las unidades $R$ dentro de esta capa es procesar las múltiples entradas para producir un valor escalar, que sirve en la construcción de la entrada a la unidad wavelet. Para desarrollar el procedimiento de las unidades $R$, necesitamos definir:

$$A(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j) = diag(\boldsymbol{d}_j)(\boldsymbol{x}(n) - \boldsymbol{t}_j)^T \tag{4. 4}$$

donde $T$ significa la operación transpuesta de la diferencia entre en vector de entradas y el vector de traslación, $diag(\boldsymbol{d}_j)$ representa una matriz diagonal construida a partir del vector de escalamiento $\boldsymbol{d}_j$, como se muestra a continuación:

$$diag(\boldsymbol{d}_j) = \begin{pmatrix} d_{j,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{j,N_i} \end{pmatrix} \tag{4. 5}$$

a partir de $A(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j)$ se construye la salida de cada unidad $R$, de la siguiente forma:

$$R(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j) = [A(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j)^T A(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j)]^{\frac{1}{2}} \quad (4.6)$$

$$R(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j) = a_j$$

Habiendo definido las salidas de las unidades $R$ (representadas por valores escalares), se introduce una conexión recursiva en las unidades wavelet. Esta conexión está afectada por el peso sináptico $\theta_j$ de la siguiente manera:

$$\alpha_j = a_j + \psi_j(n-1)\theta_j \quad (4.7)$$

donde $\alpha_j$ representa la entrada a la unidad wavelet $\psi(n)$, finalmente la salida de la unidad wavelet está dada por:

$$\psi(\alpha_j) = \psi(a_j + \psi_j(n-1)\theta_j) \quad (4.8)$$

### 4.1.3. Capa de salida

La capa de salida está constituida por combinadores lineales que representan las distintas salidas del modelo, cada unidad de salida tiene asociada un peso con cada unidad de la capa anterior $(w_{ij})$ y con cada unidad de entrada $(a_{ik})$. Los pesos sinápticos están definidos respectivamente por:

$$w_{ij}, i = 1 \dots N_s, j = 1 \dots N_i \quad (4.9)$$

$$a_{ik}, i = 1 \dots N_{s,} k = 1 \dots N_i \quad (4.10)$$

donde $N_s$ representa el número de salidas definidas en la arquitectura. El valor de cada unidad de salida está dado por:

$$y^i(n) = \sum_{j=1}^{N_{MRW}} w_{ij}\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}x_k \quad (4.11)$$

La Figura 4.3 muestra el diagrama de flujo de señal de las unidades de salida.

## 4.2. Diagrama de bloques del modelo SRWNN-MRW

El diagrama de bloques que se muestra en la Figura 4.4 presenta una descripción funcional de los distintos elementos que componen el modelo SRWNN-MRW. Los módulos descritos por $R$ representan el procesamiento radial del vector de entrada formado por $N_i$ elementos. Los módulos denotados por $\psi$ describen la función de transferencia wavelet seleccionada, la entrada a estas unidades está compuesta por la salida de las unidades $R$

junto con el término de memoria introducido en la conexión recurrente $\psi_j\,(n-1)$ afectado por un factor $\theta_j$.

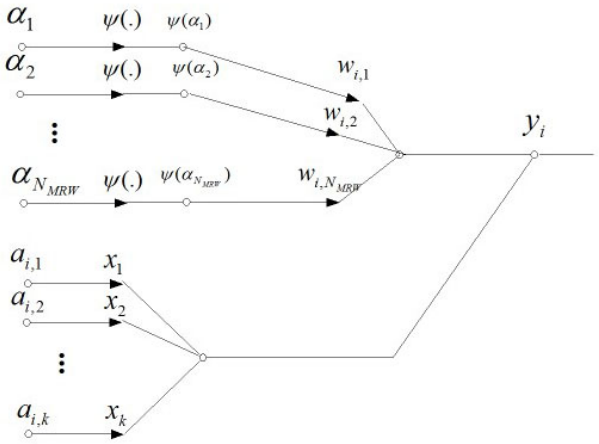


**Figura 4. 3: Diagrama de flujo de señal de la salida**

## 4.3. Algoritmo de aprendizaje

Ya que se ha definido la arquitectura del modelo propuesto, es necesario derivar el algoritmo de aprendizaje que utilizará. El algoritmo de aprendizaje propuesto se basa en el método de gradiente descendente, el objetivo es minimizar la siguiente función de error:

$$J_I = \frac{1}{2}\left[y^d(n) - y^i(n)\right]^2 \tag{4. 12}$$

donde $y^d(n)$ representa el valor deseado para la salida $i$ y $y^i(n)$ es el valor de la salida del modelo. Los parámetros variables del modelo se pueden agrupar dentro de un vector como sigue:

$$W = [a_{ik}\ m_{jk}\ d_{jk}\ \theta_j\ w_{ij}\,] \tag{4. 13}$$

$$i = 1 \ldots N_s$$

$$j = 1 \ldots N_{MRW}$$

$$k = 1 \ldots N_i$$

Una vez que se han definido los parámetros variables durante la etapa de entrenamiento, de acuerdo a la regla delta, se define el cálculo de las derivadas parciales de las salidas respecto a cada uno de estos parámetros.

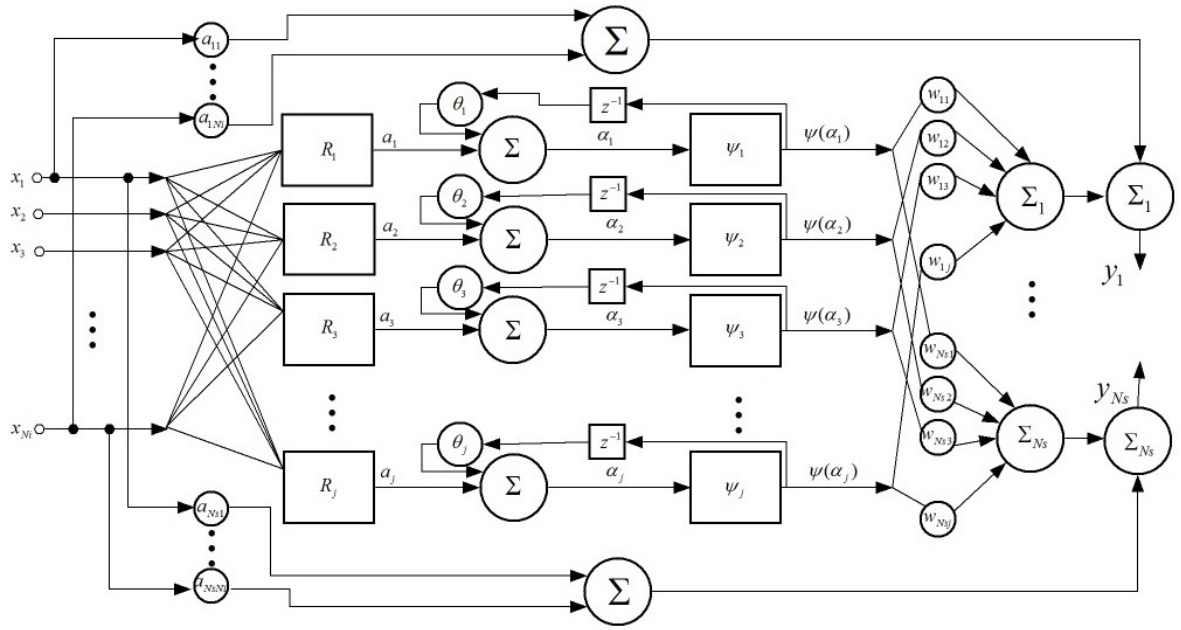

**Figura 4. 4: Diagrama de bloques del modelo SRWNN - MRW**

## 4.3.1. Obtención de $\frac{\partial y^i(n)}{\partial a_{ik(n)}}$

Habiendo definido la salida $i$ del sistema como:

$$y^i(n) = \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\,x_k$$

entonces:

$$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\,x_k}{\partial a_{ik}(n)}$$

debido a que los pesos entre las dos últimas capas no dependen directamente de $a_{ik(n)}$, lo anterior se convierte en:

$$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = \frac{\partial \sum_{k=1}^{N_i} a_{ik}\,x_k}{\partial a_{ik}(n)}$$

lo cual al llevar a cabo la derivación resulta en:

$$\frac{\partial y^i(n)}{\partial a_{ik}(n)} = x_k \tag{4. 14}$$

### 4.3.2. Obtención de $\frac{\partial y^i(n)}{\partial m_{jk}(n)}$

De manera similar a la sección anterior tenemos:

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}x_k}{\partial m_{jk}(n)}$$

debido a que los pesos $a_{ik}$ no dependen de $m_{jk}(n)$, lo anterior se expresa:

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = \frac{\partial \sum_{i=j}^{N_{MRW}} w_{ij}\,\psi(\alpha_j)}{\partial m_{jk}(n)}$$

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial m_{jk}(n)}$$

por medio de la regla de la cadena podemos reescribir lo anterior como:

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\frac{\partial \alpha_j}{\partial m_{jk}(n)}$$

si desarrollamos el tercer término de la multiplicación tenemos:

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\frac{\partial \{d_{j1}(x_1 - m_{j1})^2 \ldots d_{jk}(x_k - m_{jk})^2\}^{\frac{1}{2}}}{\partial m_{jk}(n)}$$

para finalmente obtener:

$$\frac{\partial y^i(n)}{\partial m_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\left[\frac{1}{2a_j}(-2d_{jk}^2 x_k + 2d_{jk}^2 m_{jk})\right] \tag{4. 15}$$

donde

$$R(\boldsymbol{x}(n), \boldsymbol{d}_j, \boldsymbol{t}_j) = a_j$$

### 4.3.3. Obtención de $\frac{\partial y^i(n)}{\partial d_{jk}(n)}$

Inicialmente tenemos:

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}x_k}{\partial d_{jk}(n)}$$

debido a que los pesos $a_{ik}$ no dependen de $d_{jk}(n)$, lo anterior se expresa:

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j)}{\partial d_{jk}(n)}$$

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial d_{jk}(n)}$$

por medio de la regla de la cadena podemos reescribir lo anterior como:

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\frac{\partial \alpha_j}{\partial d_{jk}(n)}$$

si desarrollamos el tercer término de la multiplicación tenemos:

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\frac{\partial\{d_{j1}(x_1 - m_{j1})^2 \ldots d_{jk}(x_k - m_{jk})^2\}^{\frac{1}{2}}}{\partial d_{jk}(n)}$$

al realizar las operaciones finalmente obtenemos:

$$\frac{\partial y^i(n)}{\partial d_{jk}(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\left[\frac{1}{2a_j}(2d_{jk}\,x_k^2 - 4d_{jk}\,x_k t_{jk} + 2d_{jk}\,t_{jk}^2)\right] \qquad (4.16)$$

### 4.3.4. Obtención de $\frac{\partial y^i(n)}{\partial \theta_j(n)}$

Si sustituimos el valor de $y^i(n)$ tenemos:

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\,x_k}{\partial \theta_j(n)}$$

debido a que los pesos $a_{ik}$ no dependen de $\theta_j(n)$ tenemos:

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = \frac{\partial \sum_{i=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j)}{\partial \theta_j(n)}$$

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \theta_j(n)}$$

aplicando la regla de la cadena lo anterior se convierte en:

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\frac{\partial \alpha_j}{\partial \theta_j(n)}$$

para finalmente obtener:

$$\frac{\partial y^i(n)}{\partial \theta_j(n)} = w_{ij}\,\frac{\partial \psi(\alpha_j)}{\partial \alpha_j}\psi_j(n-1) \qquad (4.17)$$

### 4.3.5. Obtención de $\frac{\partial y^i(n)}{\partial w_{ij}(n)}$

Si sustituimos el valor de $y^i(n)$, tenemos:

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j) + \sum_{k=1}^{N_i} a_{ik}\,x_k}{\partial w_{ij}(n)}$$

debido a que los pesos $a_{ik}$ no dependen de $w_{ij}(n)$, lo anterior se convierte en:

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \frac{\partial \sum_{j=1}^{N_{MRW}} w_{ij}\,\psi(\alpha_j)}{\partial w_{ij}(n)}$$

para finalmente obtener:

$$\frac{\partial y^i(n)}{\partial w_{ij}(n)} = \psi(\alpha_j) \qquad (4.18)$$

## 4.4. Pseudocódigo del algoritmo de aprendizaje

A continuación se da un pseudocódigo que establece el funcionamiento general del algoritmo propuesto en el apartado anterior. Las condiciones de paro en el algoritmo se definen por medio del número de épocas establecidas, aunque dependiendo de la naturaleza de la aplicación del algoritmo se pueden considerar otros criterios como error obtenido: MAE (mean absolute error), MSE (mean squared normalized error), SSE(sum squared error).

1. Establecer $N_i$, $N_s$, $N_{MRW}$.
2. Inicializar $w_{ij}$, $a_{ik}$ a números decimales pequeños
3. Inicializar $m_{jk}$, $d_{jk}$ de acuerdo a los niveles de resolución establecidos.
4. **for** i = 1,…,Numero_Épocas **do**
   **for** j=1,…,Número_Muestras **do**
   Método Ejecutar(muestra_j)
   Método Calcular_Error_Salidas()
   Ajustar $a_{ik(n)}$ por medio de:

$$a_{ik(n)} + tasa\_aprendizaje \times error \times \frac{\partial y^i(n)}{\partial a_{ik(n)}}$$

   Ajustar $m_{jk}(n)$ por medio de:

$$m_{jk}(n) + tasa\_aprendizaje \times error \times \frac{\partial y^i(n)}{\partial m_{jk}(n)}$$

Ajustar $d_{jk}(n)$ por medio de:

$$d_{jk}(n) + tasa\_aprendizaje \times error \times \frac{\partial y^i(n)}{\partial d_{jk}(n)}$$

Ajustar $\theta_j(n)$ por medio de:

$$\theta_j(n) + tasa\_aprendizaje \times error \times \frac{\partial y^i(n)}{\partial \theta_j(n)}$$

Ajustar $w_{ij}(n)$ por medio de:

$$w_{ij}(n) + tasa\_aprendizaje \times error \times \frac{\partial y^i(n)}{\partial w_{ij}(n)}$$

**end for**

**end for**

## 4.5. Diagrama de flujo del algoritmo

El diagrama de flujo correspondiente al algoritmo de aprendizaje se muestra en la Figura 4.5. En este diagrama se definen dos métodos importantes, el primero es el de Ejecutar(muestra_j) el cual procesa el patrón presentado a la capa de entrada y obtiene las salidas de los combinadores lineales, el segundo método es el de Calcular_Error_Salidas() el cual encuentra la diferencia entre los valores esperados y los obtenidos.

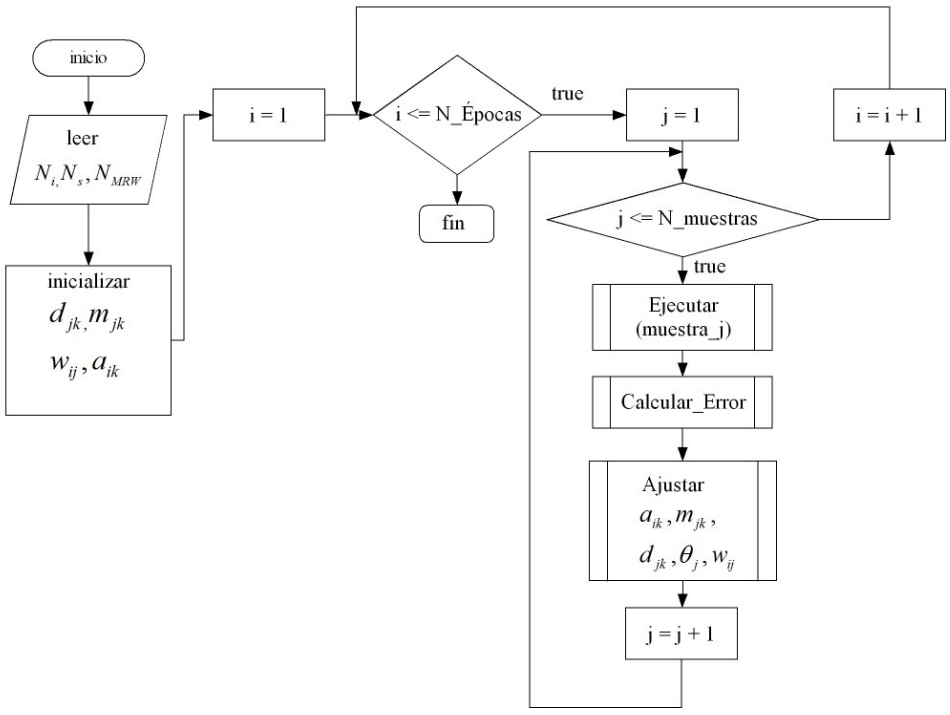


**Figura 4. 5: Diagrama de flujo del algoritmo SRWNN – MRW**

## 4.6 Complejidad computacional del modelo propuesto

Tomando en cuenta el pseudocódigo de la sección 4.4, podemos establecer el costo de cada operación basados en el número de unidades que se establecen en el modelo. El número de unidades está definido por la letra $N$. El costo computacional para operaciones básicas como operaciones aritméticas, asignación, pruebas lógicas, lectura y escritura se consideran con un costo constante $C$. Definimos a las cantidades $N_{\pi_1} = N_s N_i$, siendo $N_s$ el número de salidas del sistema y $N_i$ el número de entradas. $N_{\pi_2} = N_{MRW} N_i N_s$, donde $N_{MRW}$ es el número de unidades multidimensionales radiales. Finalmente $N_{\pi_3} = N_{MRW} N_s$.

La Tabla 4.1 sintetiza el costo computacional de cada derivación en la etapa de entrenamiento, como se puede observar, la cota superior $\mathcal{O}(N_i^2)$ establece la complejidad del algoritmo SRWNN – MRW, siendo $N_i$ el número de unidades de entrada establecidas en el sistema.

**Tabla 4. 1: Complejidad computacional de SRWNN - MRW**

| Ecuación | Multiplicaciones | Sumas | Costo |
|---|---|---|---|
| $\frac{\partial y^i(n)}{\partial a_{ik(n)}}$ | 0 | 0 | $O(CN_{\pi_1})$ |
| $\Delta a_{ik}(n) = \eta_{a_{ik}} e^i(n) \frac{\partial y^i(n)}{\partial a_{ik}}$ | $N_s N_i\{2\}$ | $N_s N_i\{1\}$ | $O(2N_{\pi_1})$ |
| $\frac{\partial y^i(n)}{\partial m_{jk}(n)}$ | $N_{MRW} N_i N_s\{2N_i + 7\}$ | $N_{MRW} N_i N_s\{2N_i\}$ | $O(N_i^2)$ |
| $\Delta m_{jk}(n) = \eta_{m_{jk}} \sum_{i=1}^{N_s} e^i(n) \frac{\partial y^i(n)}{\partial m_{jk}}$ | $N_{MRW} N_i\{N_s + 1\}$ | $N_{MRW} N_i\{N_s - 1\}$ | $O(N_{\pi_2})$ |
| $\frac{\partial y^i(n)}{\partial d_{jk}(n)}$ | $N_{MRW} N_i N_s\{2N_i + 10\}$ | $N_{MRW} N_i N_s\{2N_i + 1\}$ | $O(N_i^2)$ |
| $\Delta d_{jk}(n) = \eta_{d_{jk}} \sum_{i=1}^{N_s} e^i(n) \frac{\partial y^i(n)}{\partial d_{jk}}$ | $N_{MRW} N_i\{N_s + 1\}$ | $N_{MRW} N_i\{N_s + N_s - 1\}$ | $O(N_{\pi_2})$ |
| $\frac{\partial y^i(n)}{\partial \theta_j(n)}$ | $N_{MRW} N_s\{2\}$ | 0 | $O(N_{\pi_3})$ |
| $\Delta \theta_j(n) = \eta_{\theta_j} e^i(n) \frac{\partial y^i(n)}{\partial \theta_j}$ | $N_{MRW}\{2\}$ | $N_{MRW}\{1\}$ | $O(2N_{MRW})$ |
| $\frac{\partial y^i(n)}{\partial w_{ij}(n)}$ | 0 | 0 | $O(CN_{\pi_3})$ |
| $\Delta w_{ij}(n) = \eta_{w_{ij}} \sum_{i=1}^{N_s} e^i(n) \frac{\partial y^i(n)}{\partial w_{ij}}$ | $N_s N_{MRW}\{N_s + 1\}$ | $N_s N_{MRW}\{N_s + N_s - 1\}$ | $O(N_s^2)$ |

## 4.7. Discusión

En este capítulo se presentó el modelo *Self Recurrent Wavelet Neural Network - Multidimensional Radial Wavelons*, este modelo implementa unidades wavelet de tipo radial para llevar a cabo el procesamiento de las señales de entrada. Las unidades tipo MRW surgieron como una opción para implementar modelos donde cada unidad *wavelon* tiene múltiples conexiones de entrada, el enfoque adoptado por otras arquitecturas como la SRWNN para manejar este tipo de conexiones es considerar el producto tensor de las wavelets de una dimensión. Dado que la implementación de unidades MRW derivó es un nuevo esquema de interconexiones fue necesario definir el algoritmo de aprendizaje para esta nueva topología, cada una de las derivaciones realizadas para los parámetros libres servirá para actualizar sus valores de acuerdo a la regla delta.

Dentro de la definición de las nuevas conexiones, era necesario introducir las características recurrentes al modelo, por lo que se estableció una conexión auto recurrente en el nodo wavelet dentro de la capa multidimensional radial. De la misma forma que el modelo SRWNN, se necesita la elección de una wavelet con fórmula explicita para llevar a cabo la aplicación del algoritmo propuesto.