Capítulo 3. Redes Neuronales Wavelet

En este capítulo comienza con una breve introducción a la teoría wavelet, se presentan los conceptos básicos que servirán como base para definir a las redes neuronales con wavelets. Se inicia con una introducción que detalla las características generales de este procesamiento y las implicaciones que tiene sobre la señal de entrada, la notación que representa sus componentes y los procesos que la definen. A continuación se describen las características de las redes neuronales wavelet, los esquemas de interconexión, la representación matemática y su proceso de entrenamiento. En la parte final se hace un énfasis en la red neuronal wavelet auto recurrente y sus propiedades.

3.1. Introducción al procesamiento wavelet

El análisis wavelet consiste en la descomposición de una señal en un conjunto jerárquico de aproximaciones y detalles. En cada nivel de la jerarquía se obtiene una señal de aproximación y una señal de detalle, la señal de aproximación obtenida es una aproximación de la señal original que toma en cuenta las frecuencias bajas mientras que la señal de detalles corresponde a los componentes de alta frecuencia. Las wavelets son funciones que permiten descomponer una señal en distintos componentes de frecuencia y después analizar cada uno en una resolución acorde a su escala. La teoría de las funciones wavelet fue desarrollada en distintos campos del conocimiento como las matemáticas, física e ingeniería eléctrica [WAL08].

Las funciones wavelet deben satisfacer ciertas requerimientos matemáticos y son usadas para representar datos o aproximar otras funciones. La idea de aproximar funciones mediante la superposición de otras tiene sus orígenes en el trabajo de Joseph Fourier quien introdujo el concepto de aproximación de funciones mediante la suma de funciones seno y coseno. Las funciones wavelet surgen como una respuesta a las demandas de los científicos de aproximar señales discontinuas mediante funciones más apropiadas que senos y cosenos dado que estas son no locales y se extienden hasta infinito, teniendo como consecuencia un desempeño pobre en la detección de cambios repentinos. Tomando en cuenta las limitaciones señaladas anteriormente surge la transformada wavelet la cual permite una mejor localización en el espacio frecuencia [GRA95].

Dentro del análisis wavelet se adopta una función prototipo llamada madre wavelet. El análisis temporal es llevado a cabo mediante una versión de alta frecuencia de la función wavelet madre mientras que el análisis de frecuencia es desarrollado por una versión de baja frecuencia [GRA95].

Desde la aparición de la base Haar, renombrada años después como la wavelet Haar, han surgido a través de los años distintas familias como la Gaussiana, Meyer y Daubechies. Durante los primeros 10 años de historia de la teoría wavelet aparecieron numerosas familias pero recientemente la creación se ha vuelto menos regular dado que se ha limitado la construcción a contextos específicos [MIS07]. Las diferencias entre una familia y otra definen propiedades distintas como: soporte de las funciones, velocidad de convergencia a cero, simetría, número de momentos cero, regularidad, existencia de una función de escalamiento, ortogonalidad y la existencia de una formula explicita.

Dentro de cada familia wavelet existen subclases que se distinguen por el número de coeficientes y el nivel de iteración. Las wavelets son clasificadas dentro de una familia a menudo por el número de momentos de desvanecimiento.

3.1.1. Transformada Wavelet Continua

En la transformada continua wavelet se efectúa una comparación entre la función que se quiere aproximar $f(t)$ y varias versiones trasladadas y escaladas de la función wavelet ψ seleccionada. La transformada wavelet de una función $f(t)$ con parámetros de escalamiento a y traslación b está definida por [MIS07]:

$$C\big(a,b;f(t),\psi(t)\big) = \int_{-\infty}^{+\infty} f(t)\,\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)dt \qquad (3.1)$$

los coeficientes de la transformada son obtenidos al variar los parámetros a y b. Esta operación puede considerarse como una comparación entre la señal $f(t)$ y ψ en diversas escalas y posiciones.

3.1.2. Parámetro de escalamiento

El factor de escalamiento denotado por la letra a es una cantidad entera positiva que tiene un efecto compresor mientras más pequeño sea o un efecto de estiramiento al adoptar un valor grande. Los valores pequeños en el factor de escalamiento sirven para detectar cambios rápidos en la función que se está analizando mientras que los valores grandes provocan una baja frecuencia que detecta cambios suaves en la señal.

3.1.3. Parámetro Traslación

El parámetro de traslación tiene un efecto de retraso o avance en la señal. Cuando se efectúa el cálculo de los coeficientes, el factor de traslación se modifica para que cubra toda la señal con el valor de escalamiento específico.

3.1.4. Transformada Wavelet Discreta

En [MAL89] se implementa un algoritmo de filtrado que puede considerarse como una versión rápida de la transformada que funciona como una caja negra donde la señal original es procesada para que en la salida se obtengan los coeficientes de manera rápida. La transformada continua Wavelet continua es redundante dado que al calcular los coeficientes con cada valor de la escala convierte una señal de una dimensión en una señal de dos dimensiones. Para llevar a cabo un procesamiento más efectivo, la transformada discreta wavelet solo toma en cuenta escalas y traslaciones en potencias de dos. Estos valores se llaman escalas y posiciones diádicas, dando como resultado que los parámetros puedan ser expresados como:

$$\{a = a_0^j : j \in \mathbb{Z}\} \tag{3. 2}$$

$$\{b = nb_0 a_0^j : j, n \in \mathbb{Z}\} \tag{3. 3}$$

donde los valores $a_0 = 2$ y $b_0 = 1$, por lo tanto, la familia de funciones wavelet queda descrita por:

$$\psi_{j,n}(t) = 2^{-\frac{j}{2}} \psi\left(2^{-j} t - n\right) \tag{3. 4}$$

ver el apéndice A para mas detalles acerca del procesamiento wavelet.

3.2. Redes neuronales wavelet

Las redes neuronales wavelet combinan la teoría de procesamiento wavelet con las redes neuronales. La arquitectura de una red neuronal wavelet está basada en la arquitectura (1+1/2). En el trabajo de [CYB89] se estudia la arquitectura (1+1/2) como una herramienta de aproximación universal de funciones y se comprueba que cualquier función continua puede ser aproximada por medio de sumas de la forma:

$$g(x) = \sum_{i=1}^{N} w_i \sigma(\boldsymbol{a}_i^T \boldsymbol{x} + b_i) \tag{3. 5}$$

donde $w_i, b_i \in \Re$, $\boldsymbol{a}_i \in \Re^n$, σ es una función continua discriminatoria.

La estructura de una red neuronal wavelet es similar a la red neuronal 1+1/2. Existe una capa de entrada que conecta con una o más entradas del ambiente externo, una capa

oculta donde se efectúa el procesamiento de las entradas y finalmente una capa de salida que consiste en uno o más combinadores lineales que toman la salidas de la capa oculta.

En [ZHA92] se propone la siguiente parametrización para una red neuronal wavelet, donde se introduce el término g para lidiar con funciones que no tengan valor promedio de cero en dominios finitos. Se establece que la combinación de una traslación, un escalamiento y una función wavelet se llame *wavelon*.

$$g(x) = \sum_{i=1}^{N} w_i \psi[D_i(\boldsymbol{x} - \boldsymbol{t}_i)] + g \tag{3. 6}$$

donde $\boldsymbol{t}_i$ representa el vector que contiene los parámetros de traslación $\boldsymbol{t}_i = (t_i^1 \dots t_i^n)$ y D_i representa una matriz diagonal construida a partir de los vectores de escalamiento de la forma $D_i = diag(\frac{1}{s_i^1} \dots \frac{1}{s_i^n})$.

Existen dos enfoques principales para la creación de redes neuronales wavelet. En el primero el procesamiento wavelet y la red neuronal son desarrollados de manera independiente, es decir que la señal de entrada es descompuesta por la transformada wavelet en los nodos de la capa oculta y los coeficientes resultantes son enviados a los sumadores cuyos pesos son modificados de acuerdo a algún algoritmo de aprendizaje en específico esta estructura se llama *wavenet*. El segundo enfoque es combinar ambas teorías y dejar que los parámetros de traslación y escalamiento sean también ajustados en el proceso de aprendizaje, este tipo de estructura se conoce como red neuronal wavelet.

3.3. Arquitectura básica de una Red Neuronal Wavelet

La arquitectura más sencilla de una red neuronal wavelet consiste en tener una sola entrada y una sola salida[1]. En esta arquitectura la capa oculta está formada por *wavelons* cuyos parámetros de entrada están definidos por peso, parámetro de traslación y escalamiento. La salida de la red es una combinación lineal de las funciones wavelet de cada *wavelon*. La salida de un *wavelon* con una entrada sencilla está dada por [RAO94]:

$$\psi_{\lambda,t}(u) = \psi\left(\frac{u-t}{\lambda}\right) \tag{3. 7}$$

La arquitectura de una red neuronal wavelet se muestra en la Figura 3.1, la capa oculta consiste en *M wavelons* y la capa de salida consiste en un sumador.

[1] Llamados también modelos SISO (single input single output)

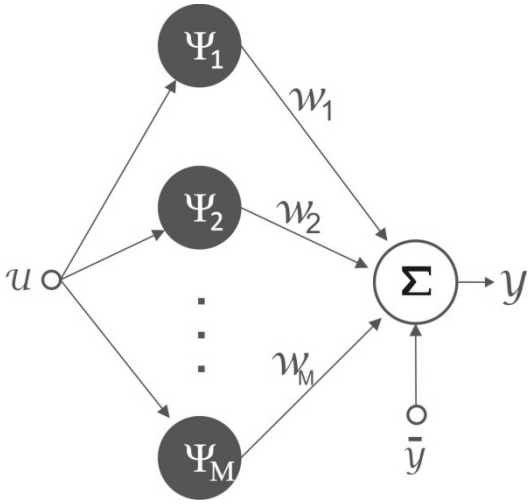


Figura 3. 1: Arquitectura 1 entrada 1 salida

la salida de esta arquitectura está dada por:

$$y(u) = \sum_{i=1}^{M} w_i \psi_{\lambda i,ti}(u) + \bar{y} \tag{3. 8}$$

3.4. Wavenet

La arquitectura de una wavenet es similar a la de red neuronal wavelet, a excepción de que los parámetros de traslación y escalamiento se mantienen fijos desde la inicialización y no son alterados durante el proceso de aprendizaje. Su salida está dada por:

$$y(u) = \sum_{i=1}^{M} w_i \phi_{\lambda i,ti}(u) \tag{3. 9}$$

donde $\phi_{\lambda i,ti}(u)$ representa la función de escalamiento y parámetros de traslación y escalamiento mantienen las restricciones mencionadas anteriormente.

3.5. Red Neuronal Wavelet Multidimensional

En este caso la entrada es un vector multidimensional, por lo tanto los *wavelons* consisten en funciones wavelet multidimensionales. Las salidas de estos nodos están dadas por la multiplicación de wavelets unidimensionales correspondientes a cada una de las entradas que llegan a él, Figura 3.2:

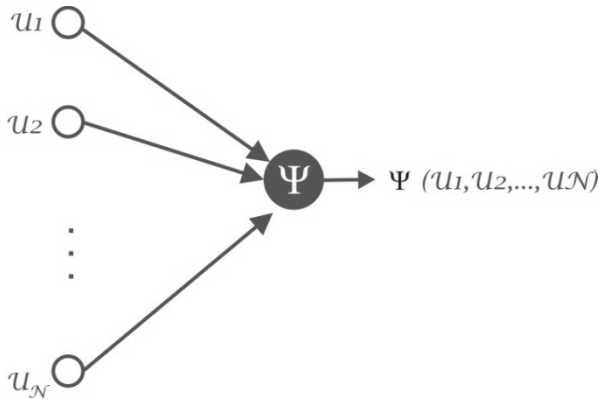


Figura 3. 2: Unidad con múltiples entradas

La salida de cada wavelon multidimensional está dada por [ZHA92]:

$$\Psi(u_1, \dots, u_N) = \prod_{i=1}^{N} \psi_{\lambda i,ti}(u_i) \tag{3. 10}$$

La arquitectura de la red neuronal multidimensional wavelet se muestra en la Figura 3.3, consta de M wavelons multidimensionales y la capa de salida consta de k sumadores que recolectan los resultados de la capa oculta.

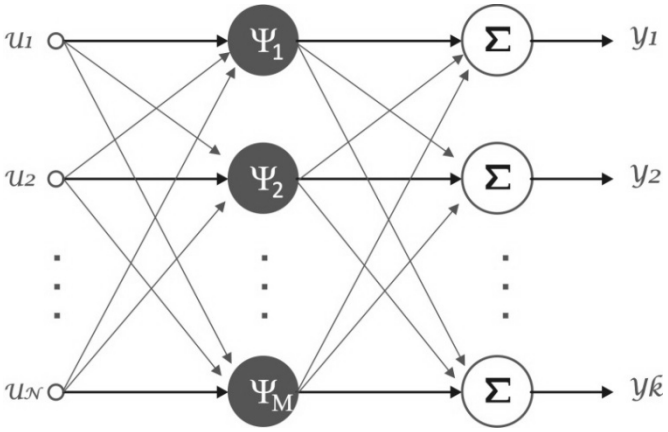


Figura 3. 3: Arquitectura con múltiples entradas y múltiples salidas

la salida de la red de cada combinador está dada por:

$$y_j = \sum_{i=1}^{M} w_i \Psi_i(u_1, \dots, u_n) + \bar{y},\ para\ j = 1 \dots k \tag{3. 11}$$

3.6. Algoritmo de gradiente estocástico

Zhang y Benveniste [ZHA92] propusieron un algoritmo de aprendizaje para redes neuronales wavelet, este algoritmo planteaba el ajuste de los parámetros de la red para la aproximación de una función. Mediante este algoritmo el aprendizaje es llevado a cabo mediante la observación de parejas { u, f(u) = g(u) + ε } donde g es la función a ser aproximada y ε es una medida del ruido en la señal. Si se considera la arquitectura de la red de una entrada una salida entonces la salida de la red con parámetros $\bar{y}, wi, ti, \lambda i$ agrupados en θ es:

$$y_\theta(u) = \sum_{i=1}^{M} w_i \psi\left(\frac{u-t_i}{\lambda_i}\right) + \bar{y} \tag{3. 12}$$

La función objetivo a ser minimizada es:

$$C(\theta) = \frac{1}{2}\{y_\theta(u) - f(u)\}^2 \tag{3. 13}$$

El gradiente para cada parámetro de θ se calcula pro medio de derivadas parciales de C:

$$\frac{\partial c}{\partial \bar{y}} = e_k \tag{3. 14}$$

$$\frac{\partial c}{\partial w_i} = e_k \psi(z_{ki})$$

$$\frac{\partial c}{\partial t_i} = -e_k w_i \frac{1}{\lambda_i} \psi^{|}(z_{ki})$$

$$\frac{\partial c}{\partial \lambda_i} - e_k w_i \left(\frac{u_k - t_i}{\lambda_i^2}\right) \psi^{|}(z_{ki})$$

donde:

$$z_{ki} = \frac{u_k - t_i}{\lambda_i} \tag{3. 15}$$

$$\psi'(z) = \frac{d\psi(z)}{dz} \tag{3. 16}$$

Para implementar el algoritmo es necesario definir una tasa de aprendizaje que es un valor en (0,1], este valor define la rapidez con que el algoritmo intenta converger. El

número de iteraciones de aprendizaje determina las veces que los datos de entrenamiento se presentan a la red durante el proceso de entrenamiento.

3.6.1. Restricción de los parámetros ajustables

Zhang y Benveniste [ZHA92] propusieron un conjunto de restricciones a los valores que se dan a los parámetros ajustables en una red neuronal wavelet para ayudar al algoritmo a converger. Las restricciones se listan a continuación:

1. Las wavelets deben mantenerse cerca del dominio D de la función que se quiere aproximar. Para lograr esto, se escoge un dominio $\mathcal{E}$ tal que $D \subset \mathcal{E} \subset \mathbb{R}$, y se requiere que todos los parámetros de traslación estén en $\mathcal{E}$.

$$t_i \in \mathcal{E}$$

2. Las wavelets deben tener un límite de compresión, definiendo $\varepsilon > 0$, todos los parámetros de escalamiento deben cumplir:

$$\lambda_i > \varepsilon,$$

3.6.2. Inicialización de los parámetros ajustables

En una red neuronal wavelet los parámetros necesitan ser inicializados. El parámetro $\bar{y}$ es inicializado al promedio de las observaciones disponibles del conjunto de entrenamiento. Todos los pesos sinápticos son inicializados a cero.

La inicialización de los parámetros de escalamiento y traslación utilizan el dominio de la función que se quiere aproximar, sea [a,b], entonces se toma un punto p en este intervalo y se define:

$$t_1 = p,\ \lambda_1 = \varepsilon(b - a) \qquad (3.17)$$

donde $\varepsilon > 0$ y comúnmente es asignado el valor de 0.5. El proceso se repite recursivamente tomando los sub intervalos y llevando a cabo las asignaciones para $t_2, \lambda_2, t_3, \lambda_3, etc.$

3.7. Algoritmo Real Time Recurrent Learning (RTRL)

El algoritmo RTRL define una estructura recurrente en la que todas las unidades pueden recibir entradas externas y también pueden ser entrenadas para producir una salida deseada en cualquier ciclo. La arquitectura se muestra en la Figura 3.4:

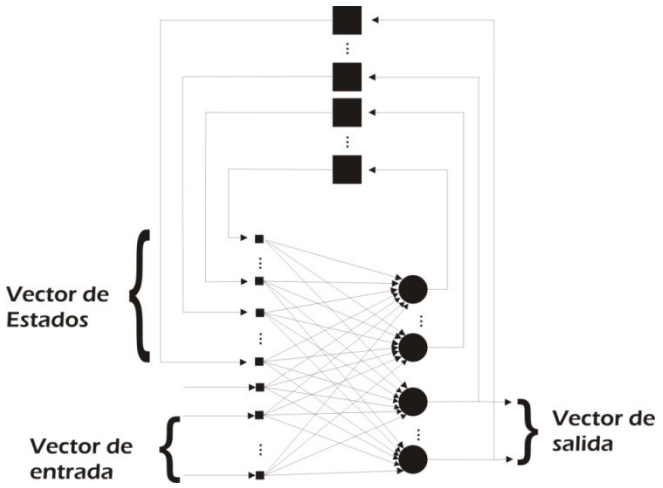


Figura 3. 4. Arquitectura recurrente totalmente conectada

Si se define una estructura con n unidades, m líneas de entradas externas, entonces $y(t)$ denota la n – tupla formada por las salidas de las n unidades en el tiempo t. Definimos a $x(t)$ como la m – tupla formada por las señales externas. También definimos $z(t)$ como la concatenación de $y(t)$ y $x(t)$. Para distinguir a los componentes de $z(t)$ establecemos a U como el conjunto de índices que describen a las unidades del modelo, de la misma forma I denota el conjunto de índices para las líneas de entrada, como se muestra en la siguiente fórmula.

$$z_k = \begin{cases} x_k(t) si\ k \in I \\ y_k(t) si\ k \in U \end{cases} \quad (3.\ 18)$$

Ahora la entrada a cada unidad está definida por la siguiente fórmula:

$$s_k(t) = \sum_{l \epsilon U} w_{kl} y_l + \sum_{l \epsilon I} w_{kl} x_l = \sum_{l \epsilon U \cup I} w_{kl} z_l(t) \quad (3.19)$$

La salida de cada unidad está dada por:

$$y_k(t+1) = f_k[s_k(t)] \tag{3.20}$$

donde f_k es la función de activación.

3.7.1. Aprendizaje en RTRL

De la misma forma que los algoritmos de retro propagación, el algoritmo de RTRL lleva a cabo el proceso de adaptación de sus pesos por medio del cómputo del gradiente en el espacio de pesos. La medida de desempeño es una función de las salidas a través del tiempo, la función define el error como:

$$e_k(t) = d_k(t) - y_k(t) \tag{3.21}$$

La medida de error de la red en el tiempo t se establece como:

$$E(t) = \frac{1}{2}\sum_{k\,\epsilon\,U} {e_k}^2 \tag{3.22}$$

Se busca un algoritmo que compute

$$\frac{\partial E(t)}{\partial w_{ij}} = \sum_{k\epsilon U} e_k(t)\frac{\partial y_k(t)}{\partial w_{ij}} \tag{3.23}$$

para resolver el término $\frac{\partial y_k(t+1)}{\partial w_{ij}}$ se sustituye $y_k(t+1)$ por $f_k[s_k(t)]$. Aplicando la regla de la cadena, da como resultado:

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = {f_k}'[s_k(t)]\left[\sum_{l\epsilon U\epsilon I} w_{kl}\frac{\partial z_l}{\partial w_{ij}} + \delta_{ik}\mathrm{z_j(t)}\right] \tag{3.24}$$

como las señales de entrada no dependen de los pesos sinápticos, la ecuación anterior se convierte en:

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = {f_k}'[s_k(t)]\left[\sum_{l\epsilon U\epsilon I} w_{kl}\frac{\partial y_l}{\partial w_{ij}} + \delta_{ik}\mathrm{z_j(t)}\right] \tag{3.25}$$

donde el término δ_{ik} (Kronecker Delta) indica que se debe sumar $\mathrm{z_j(t)}$ cuando se trate de la derivada de una salida respecto a uno de sus pesos sinápticos.

En un estado inicial de la red neuronal no se tiene dependencia inicial de los pesos, por lo que:

$$\frac{\partial y_k(t_0)}{\partial w_{ij}} = 0 \tag{3.26}$$

Finalmente se puede crear un sistema dinámico con variables $\{p_{ij}{}^k\}$ donde $p_{ij}{}^k = \frac{\partial y_k(t)}{\partial w_{ij}}$, para tener:

$$y(t+1) = f_k{}'[s_k(t)]\left[\sum_{l\epsilon U \epsilon I} w_{kl} p_{ij}{}^l + \delta_{ik} z_j(t)\right] \tag{3.27}$$

Habiendo definido las operaciones matemáticas para la actualización de los pesos sinápticos, el algoritmo consiste en el cómputo de las cantidades $p_{ij}{}^k$ y usar las diferencias de error en cada unidad para el cálculo de Δw_{ij} y así calcular los nuevos pesos.

3.8. Algoritmo Self Recurrent Wavelet Neural Network (SRWNN)

El modelo de la red neuronal wavelet recurrente considera una estructura con múltiples entradas y múltiples salidas. Ésta estructura está dividida en cuatro capas: capa de entrada, capa *wavelon*, capa de producto y finalmente la capa de salida [SUN05], (ver Figura 3.5).

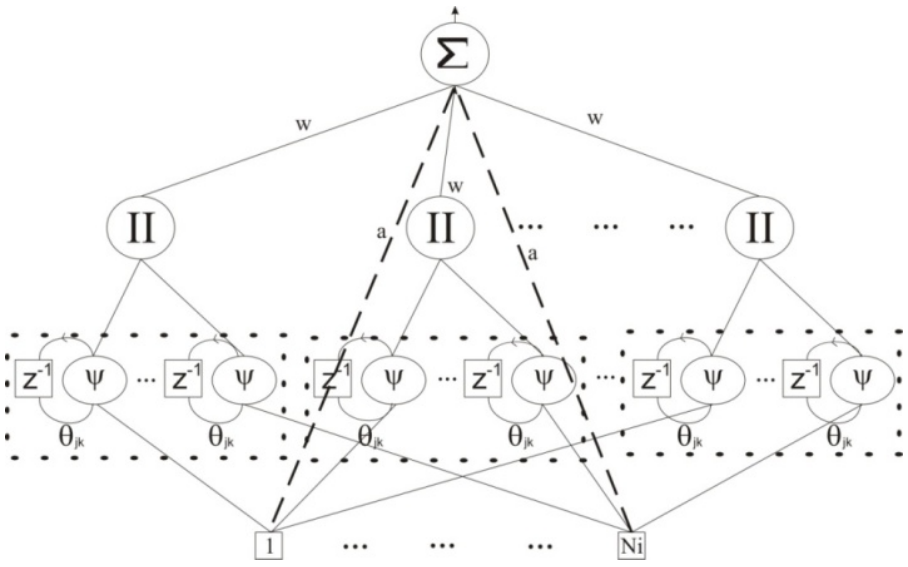


Figura 3. 5: Arquitectura auto recurrente con wavelets

Capa de entrada. En la capa de entrada se aceptan las instancias de entrada y transmite estas entradas directamente a la siguiente capa.

Capa *wavelon*. En esta capa se encuentran las unidades *wavelon*, unidades de procesamiento que utilizan una función wavelet como función de activación. Cada unidad *wavelon* tiene una conexión recurrente hacia ella misma como se muestra en la Figura 3.6.

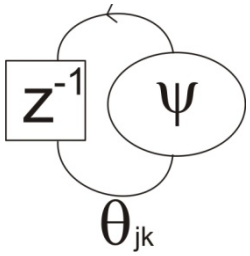


Figura 3. 6: Conexión auto recurrente en el wavelon

La función wavelet ψ_{jk} perteneciente a cada unidad wavelon es obtenida a partir de la función madre wavelet de la siguiente manera:

$$\psi_{jk}(z_{jk}) = \psi\left(\frac{u_{jk} - m_{jk}}{d_{jk}}\right) \tag{3.28}$$

$$z_{jk} = \frac{u_{jk} - m_{jk}}{d_{jk}}$$

donde m_{jk} y d_{jk} son los parámetros de traslación y escalamiento respectivamente. En este trabajo se consideró la primera derivada de la función Gaussiana como madre wavelet. Esta función wavelet tiene la siguiente fórmula:

$$\psi(x) = -xe^{-\frac{1}{2}x^2} \tag{3.29}$$

Otras wavelets con formula explicita se consideran en Tabla 3.1.

Los subíndices j y k indican el k-ésimo termino de entrada asociado con la j-ésima wavelet. Las entradas a esta capa en el tiempo discreto n están definidas por:

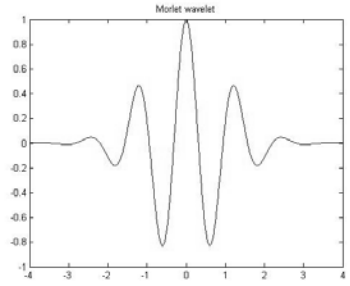
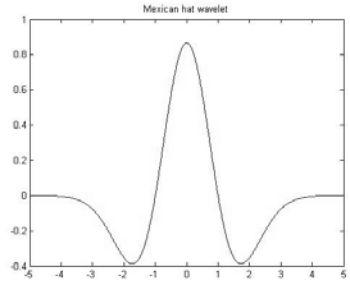
$$U_{jk}(n) = x_k(n) + \psi_{jk}(n-1)\theta_{jk} \tag{3.30}$$

donde $\psi_{jk}(n-1)$ representa el término de memoria el cual guarda la información pasada de la red. θ_{jk} Indica el peso sináptico de la conexión recursiva de ese *wavelon* y $x_k(n)$ es el estimulo externo que llega a ese *wavelon*.

Capa de producto. En esta capa se lleva a cabo la multiplicación de las salidas de los *wavelons*, en algunos trabajos como [RAO94] se considera al producto de *wavelons* unidimensionales como unidades multidimensionales. La salida de cada unidad en esta capa está dada por:

$$\Phi_j(\boldsymbol{x}) = \prod_{k=1}^{Ni} \psi(z_{jk}) \tag{3.31}$$

Tabla 3. 1: Conexión auto recurrente en el wavelon

Nombre	Fórmula explicita	Gráfica
Morlet	$\psi(x) = e^{\frac{-x^2}{2}}\cos(5x)$	
Mexican Hat	$\psi(x) = \frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}(1-t^2)e^{-\frac{t^2}{2}}$	

Capa de salida. Esta capa está formada por varios combinadores lineales cuyas entradas se forman por las salidas de la capa anterior (capa de producto). Además, los nodos de salida aceptan valores de la capa de entrada directamente. La salida de cada combinador está definida por la siguiente fórmula:

$$y(n) = \sum_{j=1}^{Nw} w_j \Phi_j(\boldsymbol{x}) + \sum_{K=1}^{Ni} a_k x_k \tag{3.32}$$

donde w_j son los pesos sinápticos que representan la unión entre la capa de salida y las unidades de producto y a_k es el peso sináptico entre las unidades de entrada y las unidades de salida.

3.8.1 Aprendizaje en SRWNN

Una vez que ha sido definida en la sección anterior la estructura de la SRWNN, es momento de establecer el proceso de aprendizaje mediante el cual se busca minimizar el error entre las salidas obtenidas por el sistema y las salidas deseadas. El método de entrenamiento para la SRWNN se basa en el gradiente descendente y busca minimizar la función cuadrática de costo [YOO05]:

$$J_I = \frac{1}{2}\left[y^{des}(n) - y^{obtenida}\ (n)\right]^2 \tag{3.33}$$

donde $y^{des}(n)$ describe el valor objetivo deseado para esa salida en el tiempo n y $y^{obtenida}\ (n)$ es el valor obtenido por el sistema en el tiempo n. Al llevar a cabo el ajuste de los valores $a_k, m_{jk}, d_{jk}, \theta_{jk}, w_j$ de la SRWNN iterativamente se busca minimizar el error obtenido después de cierto número de ciclos de entrenamiento.

El método de gradiente descendente se define como:

$$W(n+1) = W(n) + \Delta W(n) \tag{3.34}$$

$$\Delta W(n) = \eta_I \left(-\frac{\partial J_I}{\partial W(n)}\right)$$

donde η_I es la tasa de aprendizaje y W representa los parámetros ajustables durante el proceso de entrenamiento, es decir:

$$W = [a_k\ m_{jk}\ d_{jk}\ \theta_{jk}\ w_j] \tag{3.35}$$

La derivada parcial de la función de costo J_I con respecto W es:

$$\frac{\partial J_I(n)}{\partial W(n)} = -e_I \frac{\partial y^{obt}(n)}{\partial W(n)} \tag{3.36}$$

Para la derivación de nuestro algoritmo es necesario entonces calcular la derivada de la función de costo respecto a cada uno de los parámetros libres. Es decir, necesitamos computar la derivada parcial de la salida del sistema respecto a los pesos de la capa de entrada y la capa de salida, los factores de traslación y escalamiento de las funciones wavelet, los pesos de las conexiones auto re cursivas y por último los pesos de las conexiones entre la capa de producto y la capa de salida: $\frac{\partial y^{obt}(n)}{\partial a_{k(n)}}$, $\frac{\partial y^{obt}(n)}{\partial m_{jk}(n)}$, $\frac{\partial y^{obt}(n)}{\partial d_{jk}(n)}$, $\frac{\partial y^{obt}(n)}{\partial \theta_{jk}(n)}$, $\frac{\partial y^{obt}(n)}{\partial w_j(n)}$. La definición de cada una de estas derivadas se encuentra desarrolladas a detalle en el apéndice B y se sintetizan en la Tabla 3.2.

Tabla 3. 2: Derivadas parciales para SRWNN

Derivada parcial de la salida respecto a los pesos entre la capa de entrada y la capa de salida $a_{k(n)}$.	$\frac{\partial y^{obt}(n)}{\partial a_{k(n)}} = x_k$
Derivada parcial de la salida respecto al parámetro de traslación m_{jk}.	$\frac{\partial y^{obt}(n)}{\partial m_{jk}(n)} = -\frac{w_j}{d_{jk}}\frac{\partial \Phi_j}{\partial z_{jk}}$
Derivada parcial de la salida respecto al parámetro de escalamiento d_{jk}.	$\frac{\partial y^{obt}(n)}{\partial d_{jk}(n)} = -\frac{w_j}{d_{jk}} z_{jk} \frac{\partial \Phi_j}{\partial z_{jk}}$
Derivada parcial de la salida respecto al peso de la conexión auto recursiva θ_{jk} en la capa wavelet.	$\frac{\partial y^{obt}(n)}{\partial \theta_{jk}(n)} = -\frac{w_j}{d_{jk}} \psi_{jk}(n-1) \frac{\partial \Phi_j}{\partial z_{jk}}$
Derivada parcial de la salida respecto a los pesos entre la capa de producto y la capa de salida w_j.	$\frac{\partial y^{obt}(n)}{\partial w_j(n)} = \Phi_j(\boldsymbol{x})$

3.9. Discusión

Dentro de este capítulo fue presentado el concepto de redes neuronales wavelet, estas arquitecturas hacen uso de una herramienta poderosa de procesamiento llamada transformada wavelet. Los primeros intentos de incorporar el procesamiento wavelet a las redes neuronales estuvieron relacionados con la aproximación de funciones continuas [ZHA92] y han ido evolucionando hasta definir distintos tipos de esquemas de interconexión y algoritmos de aprendizaje. La unidad fundamental de las redes neuronales wavelet es el *wavelon*, el cual consta de un parámetro de traslación y uno de escalamiento los cuales sirven para modificar a la wavelet madre escogida y aplicar esta función a la señal de entrada. En la investigación de las distintas topologías y algoritmos de entrenamiento de redes neuronales se contempla a las redes neuronales wavelet como un nuevo tipo de red neuronal cuyos nodos de procesamiento utilizan funciones wavelet. Este tipo de redes neuronales combinan la teoría de la transformada wavelet con los conceptos de las redes neuronales y se presentan como una alternativa a las redes con alimentación hacia adelante con funciones de activación sigmoide.

Dentro de las distintas topologías de redes neuronales wavelet encontramos al modelo con múltiples entradas y múltiples salidas, el cual requiere de unidades de procesamiento llamadas *wavelons* multidimensionales. Las salidas de estas unidades se manejan por medio del producto tensor de las funciones wavelets aplicadas a las señales de entrada, resulta de especial interés para este trabajo la manera en la que se construyen estas unidades de

procesamiento pues la definición de nuestro problema demanda una arquitectura de este tipo.

En la parte final de esta sección se presentó la especificación matemática y características de tres distintos algoritmos de aprendizaje. El algoritmo de gradiente estocástico para entrenar redes neuronales wavelet, aunque este algoritmo no funciona para arquitecturas recurrentes, sirve como una introducción o punto de partida para entender la manera en la que los parámetros de la red llevan a cabo sus ajustes después de cada época de entrenamiento.

A continuación se presentó el algoritmo RTRL utilizado en la topología recurrente totalmente conectada, en esta arquitectura cualquier unidad tiene conexiones con las demás, además de que cualquier unidad puede recibir estímulos externos. Este algoritmo abarca otras arquitecturas más simples como la de alimentación hacia adelante y la estructura de red planteada busca tener un mayor índice de certeza en problemas donde no se tenga periodos temporales fijos. Se dice que las redes neuronales descritas por este algoritmo se ejecutan continuamente dado que se lleva a cabo un muestreo de las entradas en cada ciclo de actualización y cualquier unidad puede recibir señal de entrenamiento en cualquier ciclo [WIL89]. Aunque el algoritmo RTRL tiene muchas ventajas frente a algoritmos como retro propagación hacia atrás, tiene inconvenientes como el porcentaje de computación necesario en cada ciclo de actualización y su característica de no localidad. La primera desventaja se entiende como el alto porcentaje de almacenamiento y cómputo necesarios que dependen directamente de la cantidad de unidades en la red, para n unidades es necesario una capacidad de almacenamiento de n^3 y un tiempo de computo en cada ciclo del orden de n^4. La segunda desventaja es la característica de no localidad, esto quiere decir que cada unidad debe tener disponible la matriz de pesos recurrentes completa y del vector de errores.

Finalmente se presentó la arquitectura de la red neuronal wavelet SRWNN, la cual busca obtener una aproximación a un valor deseado y^{des} a partir de la combinación de wavelets hijas ψ_{jk} que son generadas al aplicar una traslación y un escalamiento a la madre wavelet. Los parámetros de la red $a_k, m_{jk}, d_{jk}, \theta_{jk}, w_j$ pueden ser optimizados para minimizar una función de costo mediante sus gradientes y de esta manera actualizar los cambios incrementales para cada parámetro en particular. La estructura de red neuronal wavelet tratada en esta investigación considera conexiones recurrentes en los nodos de la capa wavelet.