Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

UNIDAD ZACATENCO

DEPARTAMENTO DE MATEMÁTICAS

Diccionarios Wavelets y Redes Neuronales para el Análisis Técnico de Series de Tiempo Financieras

TESIS

Presentada por

DANIELA MARTÍNEZ MADRID

Para obtener el grado de

MAESTRA EN CIENCIAS

Director de la Tesis:

Dr. Feliú Davino Sagols Troncoso

Ciudad de México Agosto, 2024

Agradecimientos

Primero quiero agradecer al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV), por abrirme sus puertas y brindarme la oportunidad de continuar mi formación académica como estudiante de maestría. Especialmente, quiero expresar mi sincero agradecimiento a Roxana Martínez, cuyo apoyo y dedicación han sido invaluables para todos los estudiantes.

Quiero expresar mi más sincero agradecimiento al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por su apoyo financiero durante mi maestría. Su beca fue fundamental para mi formación académica y para la realización de esta tesis.

Agradezco sinceramente al Dr. Sagols Troncoso por permitirme adentrarme en temas de actualidad y relevancia como el análisis de datos financieros. Su mentoría ha sido inspiradora y me ha motivado a explorar nuevas áreas de investigación en este fascinante campo. Nuevamente gracias por su confianza, paciencia, apoyo y orientación.

A mi amada familia, a mi querido novio y a mis leales amigos, así como a todas las personas que han estado presentes en mi vida personal, les expreso mi más profundo agradecimiento. No puedo encontrar palabras suficientes para destacar la importancia que han tenido en la realización de este trabajo. Más allá de los desafíos académicos, este camino estuvo plagado de obstáculos personales que logré superar con ayuda de todos.

A todos los mencionados, gracias, gracias, g r a c i a s.

Dedicatoria

"Educar no es dar carrera para vivir, sino templar el alma para las dificultades de la vida"

Pitágoras

Resumen

Las ganancias que se pueden conseguir en los mercados financieros dependen principalmente de las decisiones de compra y venta de activos que se toman a lo largo del tiempo. Por esta razón, es importante contar con buenas herramientas de análisis. El análisis técnico se enfoca en identificar patrones gráficos en las series de precios y en interpretar indicadores económico-financieros para predecir movimientos favorables o desfavorables en los mercados. Desde hace ya algún tiempo, se han utilizado técnicas de inteligencia artificial para reconocer este tipo de patrones, y principalmente, se han usado redes neuronales.

En esta tesis, se presenta una adaptación de un ensamble de redes neuronales que toma como entrada la serie de precios de una acción bursátil particular, junto con algunos indicadores técnicos, y produce como salida la probabilidad de que el instrumento en el futuro próximo vaya al alza o a la baja. En el modelo implementado se incluyen técnicas de análisis multiresolución basadas en wavelets.

Las wavelets son funciones que permiten una descomposición más completa y flexible de las señales en términos de frecuencia y localización en el tiempo. Esto resulta especialmente útil en el análisis de series de tiempo no estacionarias y no periódicas, ya que las wavelets pueden capturar características locales con un alto nivel de detalle.

La integración de wavelets y redes neuronales permite una mejora significativa en la predicción de series de tiempo financieras. Los diccionarios wavelet, que contienen una colección de funciones obtenidas a partir de las descomposiciones wavelet, representan eficientemente las señales y facilitan el proceso de aprendizaje de las redes neuronales.

Palabras Clave: Diccionario, inteligencia artificial, redes neuronales, wavelets.

Abstract

The profits achievable in financial markets largely depend on the buying and selling decisions made over time. Therefore, having effective analysis tools is crucial. Technical analysis focuses on identifying graphical patterns in price series and interpreting economic-financial indicators to predict favorable or unfavorable market movements. Artificial intelligence techniques have been employed for some time to recognize these patterns, primarily through neural networks.

This thesis presents an ensemble of neural networks that take as input the price series of a particular stock, along with selected technical indicators, to output the probability of the instrument moving up or down in the near future. The proposed model incorporates multiresolution analysis techniques based on wavelets.

Wavelets are functions enabling a more thorough and flexible decomposition of signals in terms of frequency and time localization. This capability is particularly beneficial in analyzing non-stationary and non-periodic time series, as wavelets can capture local features with high detail.

The integration of wavelets and neural networks significantly enhances the prediction of financial time series. Wavelet dictionaries, containing a collection of functions derived from wavelet decompositions, efficiently represent signals and facilitate the neural networks' learning process.

Keywords: Artificial intelligence, dictionary, neural networks, wavelets.

Índice general

Capítulo 1

Introducción

En el ámbito de las finanzas, la toma de decisiones de inversión constituye un proceso complejo que requiere la integración de múltiples factores y herramientas analíticas. Entre las metodologías predominantes, destacan el análisis fundamental y el análisis técnico. Mientras que el análisis fundamental se centra en el estudio exhaustivo de los aspectos económicos, políticos y financieros de las empresas para realizar predicciones a largo plazo, el análisis técnico se basa en el examen de los movimientos históricos de precios y volúmenes en el mercado con el fin de efectuar pronósticos a corto plazo.

Una herramienta esencial en el análisis técnico es el empleo de patrones gráficos y de indicadores técnicos, que son utilizados por los inversionistas para identificar oportunidades de compra y venta. La integración de técnicas avanzadas de inteligencia artificial, como las redes neuronales, ha mejorado significativamente este enfoque, permitiendo el análisis de grandes volúmenes de datos y la detección de patrones complejos que podrían no ser evidentes a simple vista.

Entre los enfoques y estudios relevantes se encuentra una serie de experimentos para probar la capacidad predictiva de las redes neuronales sobre el rendimiento financiero: incorporando tanto el análisis fundamental como el análisis técnico [16]. Además, se ha utilizado la inteligencia artificial en la selección y gestión de carteras, integrando redes neuronales [25] y sistemas neuro-difusos para mejorar la precisión y fiabilidad de las predicciones [1]. También se ha propuesto un sistema de trading inteligente basado en la predicción de cajas de oscilación [28].

Los estudios centrados en el reconocimiento de patrones gráficos generalmente emplean redes neuronales convolucionales (CNN) y redes de memoria a largo-corto plazo (LSTM) para identificar patrones seleccionados [26]. Además, se han explorado combinaciones de técnicas de aprendizaje profundo con análisis de multiresolución de wavelets para mejorar la precisión de las predicciones. Un artículo notable en este ámbito propone un modelo basado en la transformada wavelet empírica, que ha demostrado superar a la transformada wavelet estacionaria tradicional en la captura de fluctuaciones de precios en diferentes escalas de tiempo [2].

1.1. Objetivo de la Investigación

En este contexto, la presente tesis se basa en el artículo [9], el cual propone una estrategia de generación de señales de compra y venta mediante la combinación de patrones gráficos y de indicadores técnicos, utilizando un conjunto de redes neuronales para predecir los movimientos futuros del mercado. Este enfoque ha demostrado ser eficaz en la obtención de retornos superiores en comparación con métodos tradicionales de análisis técnico.

Esta tesis tiene como objetivo principal contribuir al campo de las matemáticas aplicadas mediante la adaptación de estas estrategias. Este objetivo se alcanzará mediante un estudio, análisis y comprensión de los conceptos y algoritmos discutidos en dicho artículo, con un enfoque particular en el análisis de wavelets y el ensamblaje de redes neuronales aplicados al contexto financiero.

1.2. Estructura de la Tesis

Esta investigación se estructura en cinco capítulos. El primer capítulo introduce la implementación y revisa la literatura sobre el análisis técnico y las redes neuronales en el contexto financiero. En el segundo capítulo se detallan los fundamentos teóricos para los indicadores técnicos y los patrones gráficos. El tercer capítulo es crucial, ya que desarrolla la teoría del análisis de wavelets para realizar aproximaciones de las series de precios. El cuarto capítulo presenta la propuesta del ensamble de estas herramientas en el contexto financiero. Finalmente, el quinto capítulo aborda la implementación, discute los resultados experimentales obtenidos, presenta las conclusiones y sugiere posibles líneas futuras de investigación.

Capítulo 2

Análisis Técnico

El análisis técnico se centra en los movimientos de precio y volumen de las acciones. Típicamente, los traders utilizan indicadores como el Promedio Móvil (Moving Average), Banda de Bollinger (Bollinger Band), Índice de Fuerza Relativa (Relative Strength Index), Convergencia y Divergencia del Promedio Móvil (Moving Average Convergence Divergence - MACD) y el Oscilador Estocástico (Stochastic Oscillator) para determinar señales de compra y venta. Además, pueden emplear patrones gráficos como los patrones de movimiento de precio y los patrones de velas (candlestick chart patterns) para analizar datos de trading pasados y predecir precios y tendencias futuras [9].

Indicadores técnicos. Son valores obtenidos de fórmulas matemáticas y estadísticas aplicadas a datos históricos de precios y volúmenes de un activo financiero.

Gráficas de línea. Son representaciones visuales que muestran el comportamiento de los precios de un activo financiero a lo largo del tiempo mediante una línea continua. Estas gráficas conectan una serie de puntos de datos, que suelen ser los precios de cierre de cada período.

Figura 2.1: Un gráfico de líneas de Intel. Este tipo de gráfico produce una línea sólida al conectar los sucesivos precios de cierre. Tomada de [19].

Los gráficos de barras son una herramienta común y efectiva para identificar patrones gráficos de líneas en el análisis técnico, el gráfico está representado por una barra vertical en periodo de tiempo determinado, que generalmente es un día, muestra los precios de apertura, máximo, mínimo y cierre. Este tipo de gráficos es uno de los más utilizados en el análisis técnico.

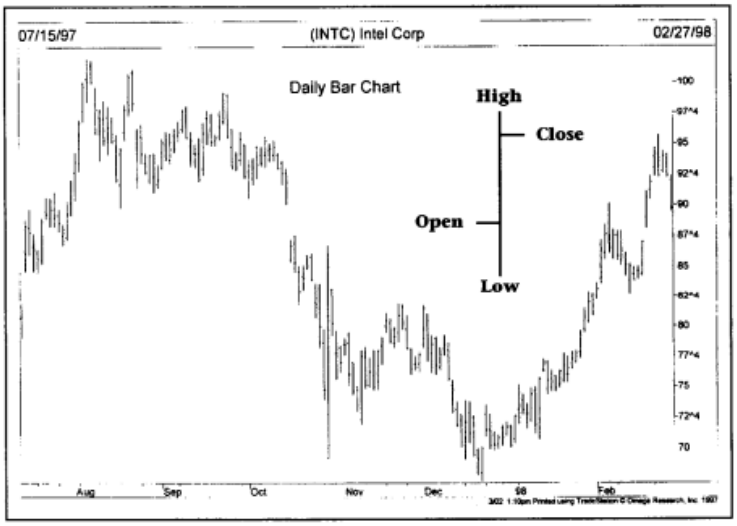


Figura 2.2: Un gráfico de barras diario de Intel. Cada barra vertical representa la información de un día. Tomada de [19].

Gráficas de vela. Es un tipo de gráfico similar al gráfico de barras, pero estas son llamadas velas, también ocurren en un periodo de tiempo especifico, que generalmente es un día. En el gráfico de velas, una línea delgada (llamada sombra) muestra el rango de precios del día desde el máximo hasta el mínimo. Una porción más ancha de la barra (llamada cuerpo

real) mide la distancia entre la apertura y el cierre. Si el cerrado es más alto que el abierto, el cuerpo real es blanco (positivo). Si el cierre es menor que la apertura, el cuerpo real es negro (negativo).

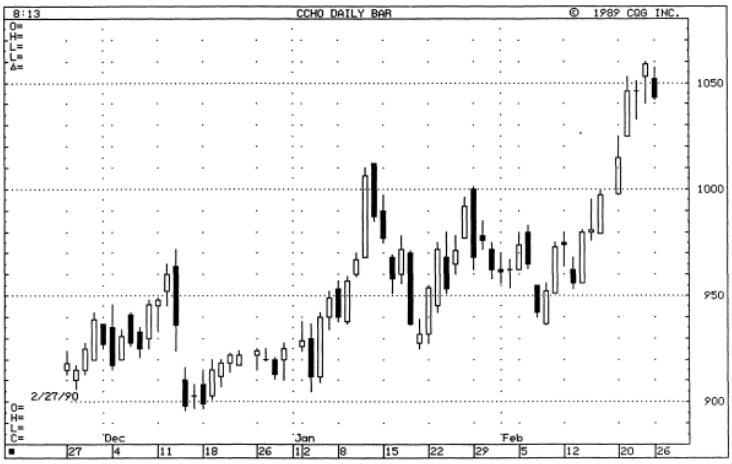


Figura 2.3: Gráfico de velas diarias, Cacao-marzo de 1990. Tomada de [20].

2.1. Indicadores Técnicos

En esta sección, definiremos y describiremos cada uno de los indicadores técnicos utilizados en este estudio. Estos indicadores fueron seleccionados debido a su uso en el artículo [9] y a su popularidad en la práctica.

Media Móvil (MA)

La Media Móvil (MA) representa el promedio móvil de los datos a lo largo del tiempo, se calcula como la suma de los valores de los últimos n datos dividida entre n. Para un periodo de n días, la media móvil en el tiempo actual t se define como:

$$\mathrm{MA}(n,t) = \frac{\mathrm{P}_t + \mathrm{P}_{t-1} + \cdots + \mathrm{P}_{t-n+1}}{n} = \frac{\sum_{i=1}^{n} \mathrm{P}_{t-i+1}}{n}, \quad n \leq t$$

donde P_t representa el precio en el periodo t.

Media Móvil Exponencial (EMA)

La Media Móvil Exponencial (EMA) se distingue de la media móvil simple por otorgar mayor peso a los valores más recientes y menor peso a los valores más antiguos. Este enfoque permite detectar tendencias y variaciones en los precios con mayor sensibilidad a las condiciones actuales del mercado.

La fórmula para calcular la EMA es:

$$\text{EMA}(n,t) = \text{SF} \cdot \text{P}_t + (1 - \text{SF})\text{EMA}(n, t-1)$$

donde P_t representa el precio en el periodo t y SF es un hiperparámetro llamado factor de suavidad, que se toma en el intervalo de $(0, 1)$.

Bandas de Bollinger (BB)

Las Bandas de Bollinger se representan mediante tres líneas trazadas sobre un gráfico de precios. La línea central corresponde a una media móvil, mientras que las dos líneas exteriores están ubicadas a una distancia de k veces la desviación estándar de dicha media móvil. Estas bandas forman un canal alrededor del precio y son útiles para detectar la volatilidad y posibles puntos de inflexión en los movimientos del mercado.

$$\text{MB}(n,t) = \text{MA}(n,t)$$

$$\text{UB}(n,t) = \text{MA}(n,t) + k \cdot \sigma(n,t)$$

$$\text{LB}(n,t) = \text{MA}(n,t) - k \cdot \sigma(n,t)$$

Donde $\text{MB}(n,t)$ es la Banda Media (Middle Band), $\text{UB}(n,t)$ es la Banda Superior (Upper Band), $\text{LB}(n,t)$ es la Banda Inferior (Lower Band), y σ es la desviación estándar de los precios en ese periodo.

Las Bandas de Bollinger proporcionan señales de compra y venta basadas en la relación del precio con respecto a las bandas superior e inferior.

- **Señal de compra**: Una señal de compra se produce cuando el precio de un activo toca o cruza la Banda Inferior (LB) y luego comienza a moverse hacia arriba. Esto sugiere que el activo puede estar sobrevendido y podría revertirse al alza.

- **Señal de venta**: Una señal de venta se produce cuando el precio de un activo toca o cruza la Banda Superior (UB) y luego comienza a moverse hacia abajo. Esto indica que el activo puede estar sobrecomprado y podría revertirse a la baja.

- **Confirmación de tendencia**: Cuando el precio se mueve constantemente entre la Banda Media (MB) y la Banda Superior (UB), indica una tendencia alcista. Por el contrario, cuando el precio se mueve entre la Banda Media y la Banda Inferior (LB), indica una tendencia bajista.

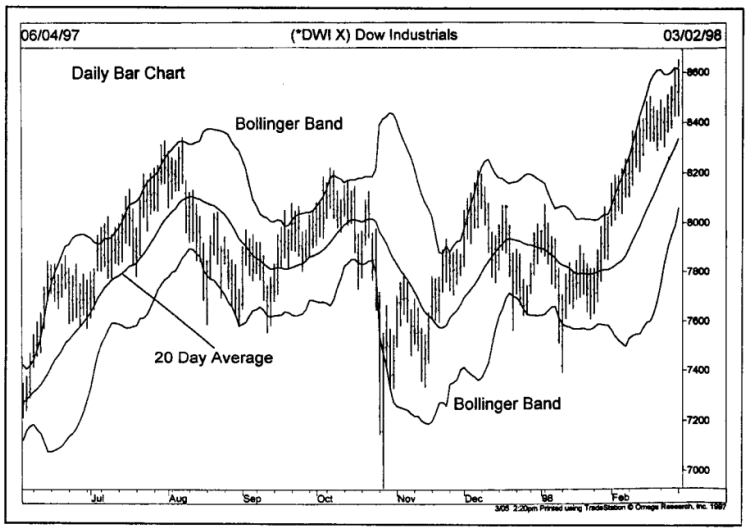


Figura 2.4: Bandas de bollinger graficadas con un promedio móvil de 20 días. Tomada de [19]

.

Media Móvil de Convergencia/Divergencia (MACD)

Es un indicador oscilador que muestra la relación entre dos medias móviles de los precios de un activo. La fórmula del MACD para el día actual t es:

$$\text{MACD}(t) = \text{EMA}(12, t) - \text{EMA}(26, t)$$

La elección de los valores específicos de los períodos (12 y 26 días) es en gran medida una convención histórica. El EMA de 12 días se considera una representación más sensible de los movimientos de precios a corto plazo, mientras que el EMA de 26 días es más suavizado y representa una visión más amplia a mediano plazo.
La línea de señal se construye entonces como la media móvil exponencial de la línea MACD:

$$\text{Línea de señal}(t) = \text{EMA}(9, \text{MACD}(t))$$

El MACD también incluye un histograma, que puede ayudar a visualizar los cruces y las divergencias entre el MACD y su línea de señal:

$$\text{Histograma MACD}(t) = \text{MACD}(t) - \text{Línea de señal}(t)$$

Las señales que emite este indicador son:

- **Señal de Compra**: Ocurre cuando la línea MACD cruza por encima de la línea de señal.

- **Señal de Venta**: Ocurre cuando la línea MACD cruza por debajo de la línea de señal.

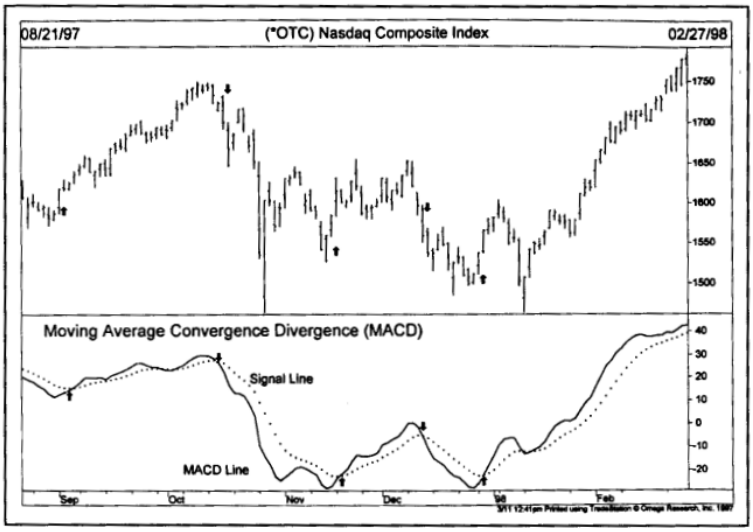


Figura 2.5: Las flechas muestran cinco señales como se indican arriba . Tomada de [19]

El indicador que usaremos para analizar el conjunto de datos de **volumen** es OBV. Para ser precisos en esta definición, el volumen de un activo se refiere al número de unidades negociadas durante un período de tiempo específico; en nuestro estudio, nos enfocaremos en el período diario.

Volumen de Equilibrio (OBV)

El Volumen de Equilibrio (OBV, por sus siglas en inglés "On-Balance Volume") muestra una correlación que incluye la cantidad de volumen que viene con un cambio de precio, multiplicado por la suma del volumen de operaciones, en comparación con el volumen total del periodo.

$$\mathrm{OBV}(n,t) = \frac{\sum_{i=0}^{n-1} \mathrm{sign}\left[\mathrm{C}(t-i) - \mathrm{C}(t-i+1)\right] \cdot \mathrm{V}(t-i)}{\sum_{t=0}^{n-1} \mathrm{V}(t-i)}$$

donde para un tiempo t dado, $\mathrm{C}(t)$ es el precio de cierre, $\mathrm{V}(t)$ es el volumen del activo y n el número de días en el periodo.

El indicador OBV emite señales de compra y venta basadas en la relación entre el volumen y los movimientos del precio. Las señales del OBV se interpretan principalmente a través de la divergencia entre el OBV y el precio del activo.

- **Divergencia alcista**: Ocurre cuando el precio del activo está bajando o es constante mientras el OBV está subiendo. Esto sugiere que el volumen está entrando en el activo a pesar de que el precio no está subiendo, lo que podría anticipar un cambio en la tendencia del precio hacia el alza.
- **Divergencia bajista**: Ocurre cuando el precio del activo está subiendo o es constante mientras el OBV está bajando. Esto sugiere que el volumen está saliendo del activo a pesar de que el precio no está bajando, lo que podría anticipar un cambio en la tendencia del precio hacia la baja.

Entonces las señales de compra y venta emitidas por este indicador están dadas por:

- **Señal de compra**: Se produce cuando el OBV cruza por encima de su media móvil o cuando el precio del activo subyacente sube mientras que el OBV también sube.
- **Señal de venta**: Se genera cuando el OBV cruza por debajo de su media móvil o cuando el precio del activo subyacente baja mientras que el OBV también baja.

A continuación, definiremos los indicadores técnicos tipo **osciladores**. Estos indicadores son especialmente valiosos en mercados sin tendencia, donde los precios oscilan dentro de un rango horizontal. Generan valores generalmente entre 0 y 100, lo que los hace útiles para evaluar condiciones de sobrecompra y sobreventa en un activo financiero. Por ejemplo, un valor cercano a 0 podría indicar una condición de sobreventa, mientras que un valor cercano a 100 podría indicar una situación de sobrecompra.

Índice de fuerza relativa (RSI)

El Índice de Fuerza Relativa (RSI) es un indicador tipo oscilador utilizado en el análisis técnico para evaluar la fuerza del precio mediante la comparación de los movimientos al alza o a la baja de los sucesivos precios de cierre. Para cada periodo de tiempo, como por ejemplo diariamente, se calculan los cambios al alza U y a la baja D utilizando la variable P_t como el precio de cierre del día t:

- Si el periodo de tiempo fue al alza:

$$\mathrm{U} = \mathrm{P}_t - \mathrm{P}_{t-1}$$

$$\mathrm{D} = 0$$

- Si el periodo de tiempo fue a la baja:

$$\mathrm{U} = 0$$

$$\mathrm{D} = \mathrm{P}_{t-1} - \mathrm{P}_t$$

- Si los precios no cambian:

$$\mathrm{U} = \mathrm{D} = 0$$

Así la suma de los valores absolutos de los cambios de precio a la alza AU y a la baja AD durante n días definen el cociente fuerza relativa:

$$\mathrm{RS} = \frac{\mathrm{AU}}{\mathrm{AD}}$$

Esto define el índice RSI (Relative Strength Index) con valores entre 0 y 100,

$$\mathrm{RSI} = 100 - 100 \cdot \frac{1}{1 + \mathrm{RS}} = 100 \cdot \left(\frac{\mathrm{RS}}{1 + \mathrm{RS}}\right) = 100 \cdot \left(\frac{\mathrm{AU}}{\mathrm{AU} + \mathrm{AD}}\right)$$

El RSI se traza en una escala de 0 a 100 y se utiliza para identificar condiciones de sobrecompra y sobreventa en el mercado. Emite las siguientes señales de compra o venta:

- **Señal de Compra**: Cuando el RSI cruza de abajo hacia arriba el nivel de 30 (o 20 en mercados bajistas), indicando una posible reversión al alza.

- **Señal de Venta**: Cuando el RSI cruza de arriba hacia abajo el nivel de 70 (o 80 en mercados alcistas), indicando una posible reversión a la baja.

Oscilador Estocástico (SO)

Es un indicador técnico tipo oscilador. Compara la diferencia de los precios de cierre entre los precios más altos y más bajos en un periodo corto de tiempo. Se calcula utilizando dos líneas principales %K y %D.

%K (línea rápida): Determinar la posición actual del precio de cierre en relación con su rango durante un período de tiempo específico

$$\%\mathrm{K} = 100 \cdot \frac{\mathrm{C}_t - \mathrm{L}_n}{\mathrm{R}_n}$$

Donde C_t es el precio de cierre actual, L_n es el precio de cierre más bajo durante el periodo tiempo y R_n es la diferencia de tiempo de más alto y más bajo.

%D (línea lenta): Es una media móvil de %K y suaviza las fluctuaciones. Se puede calcular como:

$$\%\mathrm{D} = \frac{\sum_{i=t-n}^{t} \%\mathrm{K}}{n}$$

El indicador sugiere que el activo esta en periodo de sobre compra cuando %K o %D están por encima de 80, y en un periodo de sobreventa cuando %K o %D están por debajo de 20.

Las señales que emite este indicador son:

- **Señal de Compra**: Ocurre cuando %K cruza por encima de %D en la zona de sobreventa.

- **Señal de Venta**: Ocurre cuando %K cruza por debajo de %D en la zona de sobrecompra.

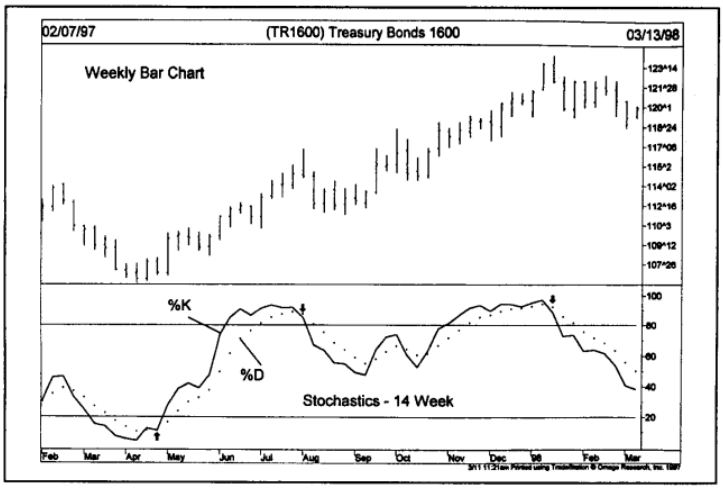


Figura 2.6: Las flechas hacia abajo indican dos señales de venta, y la flecha hacia arriba una señal de compra. Tomada de [19].

2.2. Patrones Gráficos de Línea

Los patrones gráficos de líneas en el mercado bursátil se refieren a formas o configuraciones específicas que se pueden observar en los gráficos de precios de un activo financiero. Estos patrones proporcionan información útil sobre la dirección futura de los precios, lo que los convierte en señales recurrentes en la tendencia del precio de las acciones. Se pueden clasificar en tres categorías principales: reversión, continuación y consolidación.

Los patrones gráficos en el análisis técnico se interpretan en conjunto con la información del volumen debido a que este proporciona información crucial sobre la actividad de compra y venta que respalda los movimientos de precio observados en el mercado. Esta integración permite a los analistas técnicos confirmar, validar y evaluar con precisión la fuerza y la viabilidad de las señales ofrecidas por los patrones gráficos. Por lo tanto, todas las representaciones gráficas de las series de precios en esta sección incluirán datos de los volúmenes correspondientes.

2.2.1. Patrones de reversión

Son patrones que sugieren un cambio significativo en la tendencia actual del mercado. Estos patrones sugieren que la tendencia predominante, ya sea alcista o bajista, está a punto de invertirse. Entre los patrones mas comunes están: cabeza y hombros, doble techo y doble suelo, triple techo y triple suelo y broading tops.

Doble techo. Este patrón de reversión sugiere entrar en una posición bajista después de una tendencia alcista. Se caracteriza por dos picos al mismo nivel respecto a un valle situado en medio y que crea el escote. Los tipos de doble techo varían según la forma de sus picos:

aquellos con picos puntiagudos y estrechos se denominan Adán, mientras que los picos más anchos y planos se llaman Eva.

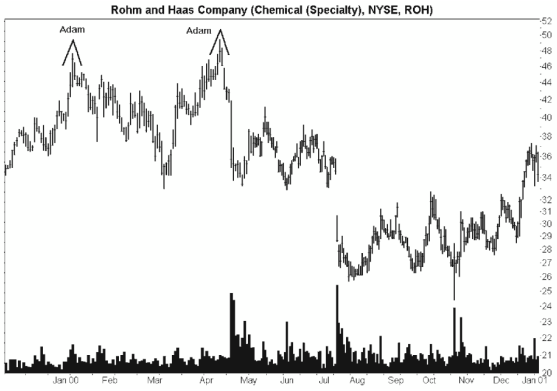


Figura 2.7: Doble Techo Adán y Adán.

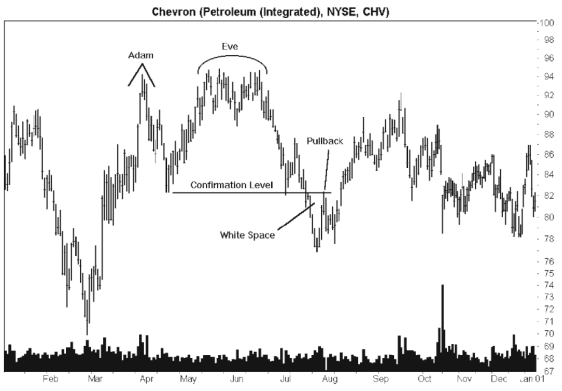


Figura 2.8: Doble Techo Adán y Eva.

Figura 2.9: Tomadas de [6]

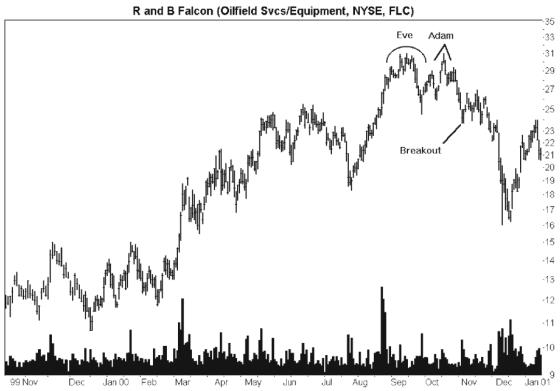


Figura 2.10: Doble Techo Eva y Adán.

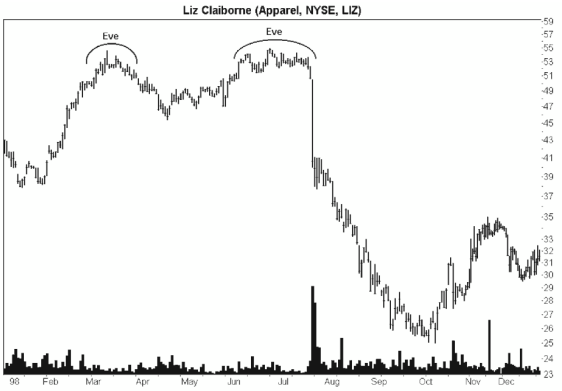


Figura 2.11: Doble Techo Eva y Eva.

Figura 2.12: Tomadas de [6]

Doble suelo. Este patrón de reversión sugiere entrar en una posición alcista después de una tendencia bajista. Se caracteriza por dos mínimos que están prácticamente al mismo nivel, formando un pico situado en medio. Igualmente como en el doble techo los subtipos de patrones están determinados por valles Adán y Eva.

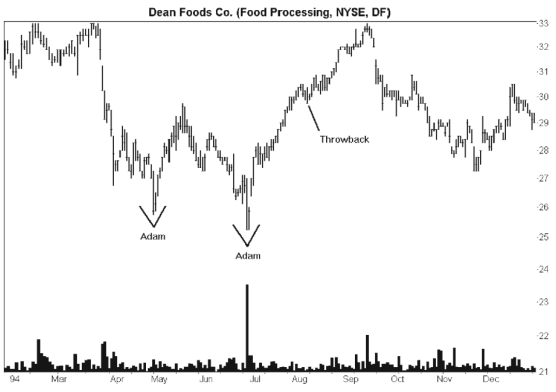


Figura 2.13: Doble Suelo Adán y Adán.

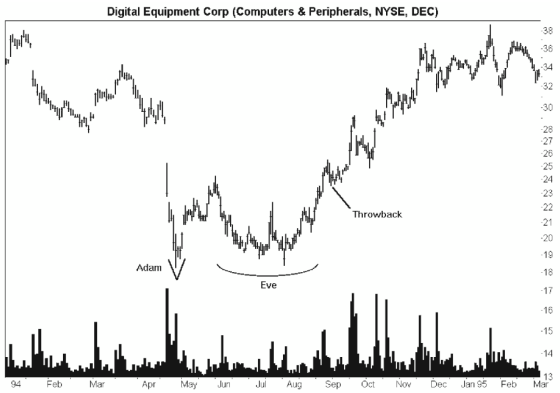


Figura 2.14: Doble Suelo Adán y Eva.

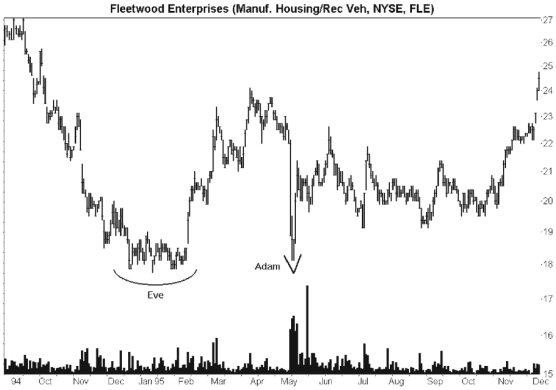


Figura 2.15: Doble Suelo Eva y Adán.

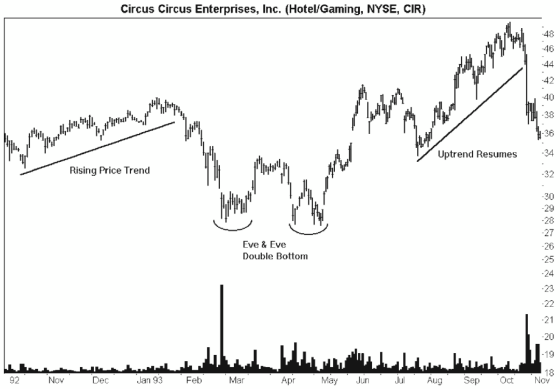


Figura 2.16: Doble Suelo Eva y Eva.

Figura 2.17: Tomadas de [6]

Cabeza y Hombros. Este patrón de reversión indica que debemos de entrar en una posición bajista tras una tendencia alcista. Como su nombre lo indica, el patrón consiste en tres techos en el que el máximo más alto se encuentra en el medio, los picos laterales se denominan hombros y el central cabeza. La línea que conecta los dos valles es la línea del cuello ("neckline"), cuando esta línea cruza la serie de precios aparece la ruptura, la cual sugiere a los traders entrar en operaciones en la dirección de la nueva tendencia.

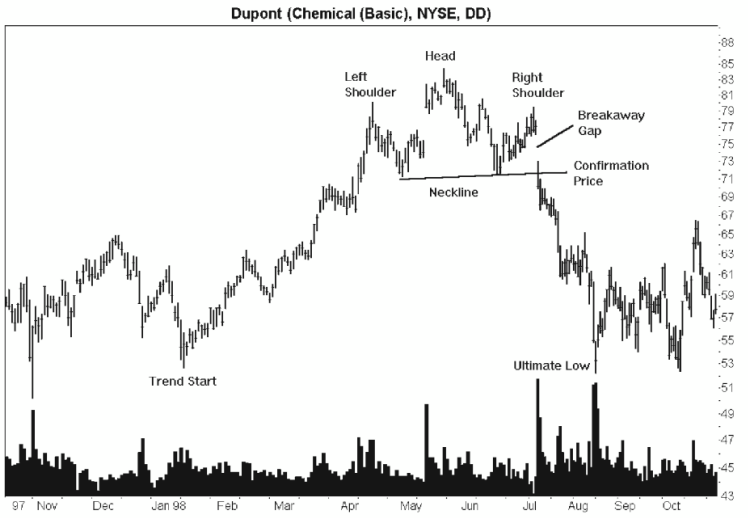


Figura 2.18: Patrón de Cabeza y Hombros, las barras negras son los volumen del activo que suele ser mayor en el hombro izquierdo que en el derecho. Tomada de [6].

Cabeza y Hombros (Invertido). Este patrón de reversión indica que debemos entrar en una posición alcista tras una tendencia bajista. Como su nombre lo indica, el patrón consiste en tres valles en el que el mínimo más bajo se encuentra en el medio, los valles laterales se denominan hombros y el central cabeza. La línea que conecta los dos picos es la línea del cuello ("neckline"), cuando esta línea cruza la serie de precios aparece la ruptura, la cual sugiere a los traders entrar en operaciones en la dirección de la nueva tendencia.

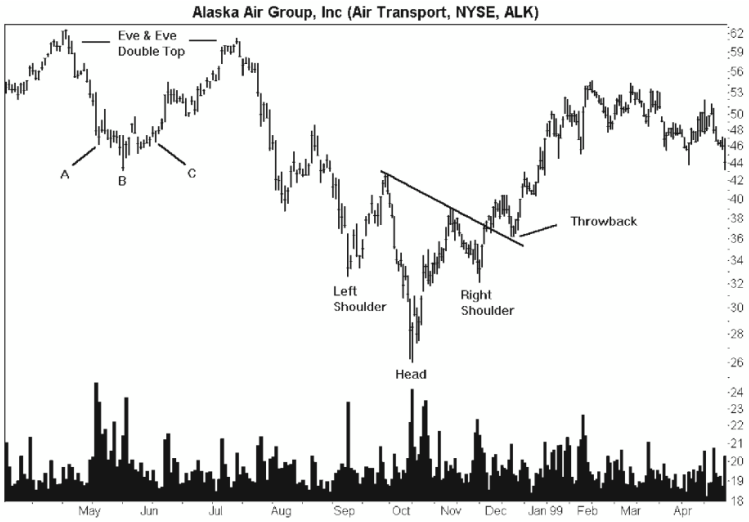


Figura 2.19: Patrón de Cabeza y Hombros invertido, las barras negras son los volumen del activo que suele ser más alto en el hombro izquierdo o cabeza, pero menor en el hombro derecho. El patrón A–B–C no es un patrón de cabeza y hombros invertido porque los puntos A y C no son puntos mínimos. Tomada de [6].

2.2.2. Patrones de Continuación

Son patrones que sugieren que la tendencia actual continuará después de una pausa temporal. Son usualmente usados para identificar posibles puntos de entrada en el mercado durante una tendencia existente. Entre los patrones mas comunes están: los rectángulos y los scallops.

Rectángulos. Este patrón se puede clasificar como patrón de continuación o de reversión, dependiendo de las tendencias y de cómo se hacen la ruptura del precio al salir del rectángulo. Su forma se caracteriza por un movimiento de precios lateral entre dos niveles horizontales de soporte y resistencia.

El siguiente ejemplo ilustra la continuación de una tendencia alcista consolidada. En este caso, se observa un gap de ruptura alcista, que se produce cuando el precio supera un nivel de resistencia previamente establecido. Este movimiento, reforzado por la aparición de un gap de continuación y un alto volumen de negociación, incrementa la probabilidad de que la tendencia alcista continúe.

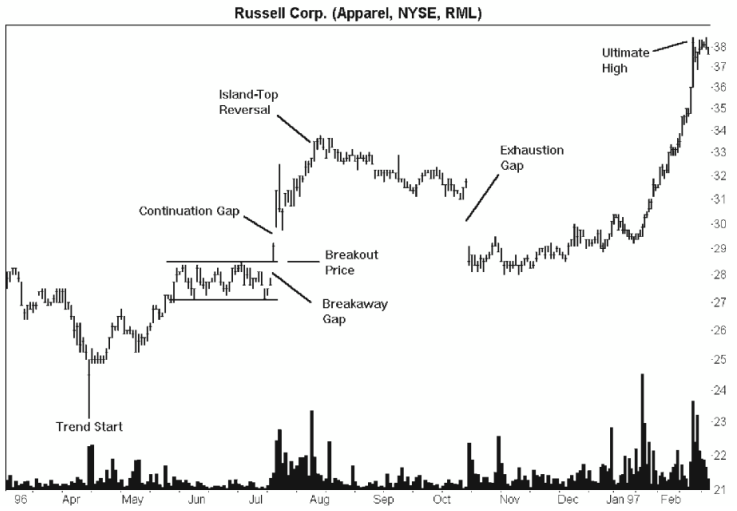


Figura 2.20: Continuación de una tendencia alcista tras la aparición de patrón rectángulo. Tomada de [6].

Scallop Ascendente. Es un patrón gráfico que puede considerarse como una continuación alcista cuando se forma dentro de una tendencia alcista establecida. Este patrón muestra una serie de mínimos ascendentes que conforman el "bowl"o cuenco inicial. Posteriormente, el precio alcanza un máximo más alto conocido en un camino conocido como el "handle". La confirmación del patrón como continuación alcista ocurre cuando el precio rompe por encima del máximo más alto, indicando una posible continuación de la tendencia alcista subyacente.

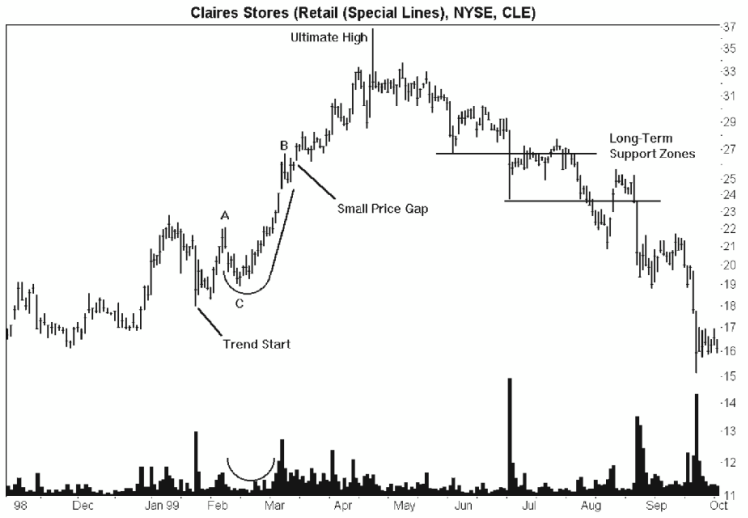


Figura 2.21: Patrón Scallop Ascendente. Tomada de [6].

Triángulos. Hay tres tipos de triángulos: simétricos, ascendentes y descendentes.

El patrón **triángulo simétrico** se caracteriza por la convergencia de dos líneas de tendencia. La línea de tendencia superior une una serie de máximos descendentes, mientras que la línea de tendencia inferior conecta una serie de mínimos ascendentes. Generalmente, este patrón sugiere la continuación de la tendencia existente, aunque también puede señalar una reversión. La dirección en la que se produce la ruptura (ya sea por encima o por debajo de las líneas de tendencia) es fundamental para predecir el movimiento futuro del precio.

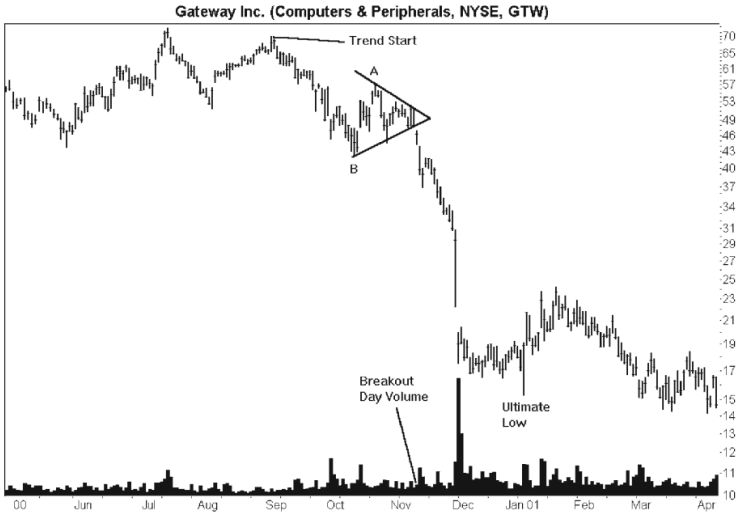


Figura 2.22: Patrón triángulo simétrico de una tendencia bajista. Tomada de [6].

El **triángulo ascendente** se distingue por tener una línea de tendencia inferior que sube y una línea de tendencia superior que es horizontal. Este patrón refleja que los compradores están ganando fuerza, ya que están dispuestos a pagar precios cada vez más altos. Normalmente, se considera un patrón de continuación alcista. Si el precio rompe por encima de

la línea horizontal superior, es una señal fuerte de que la tendencia alcista probablemente continuará.

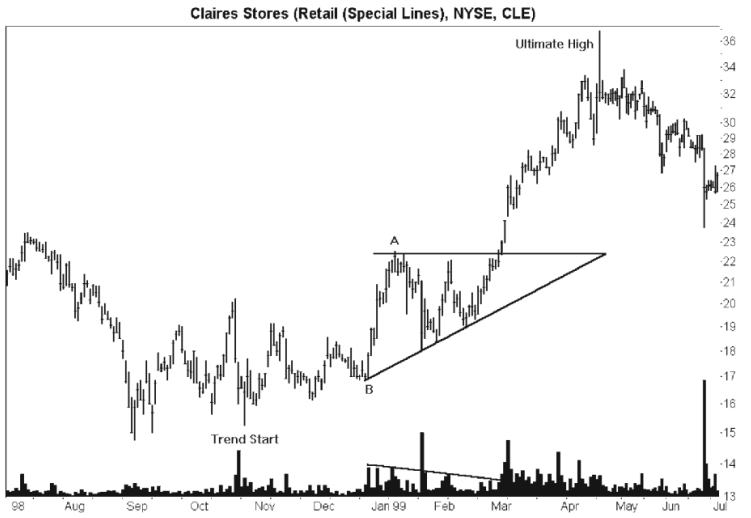


Figura 2.23: Patrón triángulo ascendente de una tendencia alcista. Tomada de [6].

El **triángulo descendente** se caracteriza por una línea superior descendente y una línea inferior plana o horizontal. Este patrón indica que los vendedores están ganando fuerza al mostrar disposición para vender a precios más bajos. Por lo general, se considera un patrón de continuación bajista. Una ruptura por debajo de la línea horizontal inferior suele señalar una alta probabilidad de que la tendencia bajista continúe.

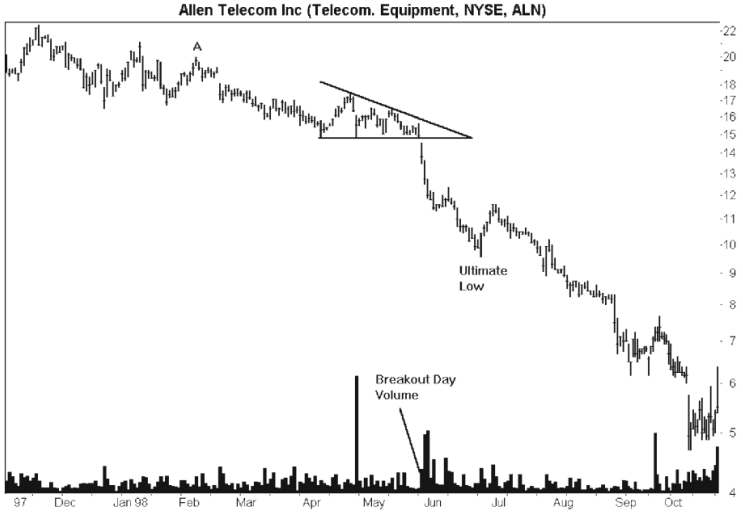


Figura 2.24: Patrón triángulo descendente de una tendencia bajista. Tomada de [6].

En la implementación, el ensamble de redes neuronales diseñado en este trabajo no recibirá reglas específicas sobre los patrones gráficos estudiados en el análisis técnico. Sin embargo, aprenderá a identificarlos mediante aproximaciones de las señales del conjunto de datos. Este enfoque permite que el modelo capture y reconozca patrones complejos sin depender de una

programación explícita de reglas, lo que mejora su capacidad para adaptarse a diferentes condiciones de mercado.

En el siguiente capítulo, nos ocuparemos de toda la teoría de wavelets, que es fundamental para la construcción de estas aproximaciones.

Capítulo 3

Análisis Cuantitativo

3.1. Análisis de Multiresolución Wavelet

El análisis de multiresolución wavelet (WMRA) nos permitirá capturar patrones gráficos en diferentes niveles y escalas. Surgió con el propósito de capturar de manera más precisa las características locales de las señales e imágenes, especialmente en aquellas que no son estacionarias o periódicas y que requieren un mayor nivel de detalle.

El término **"wavelet"** fue creado a partir de la combinación de las palabras "wave"(onda) y "let"(pequeño), haciendo referencia a que estas funciones matemáticas se comportaran como pequeñas ondas que están localizadas en el tiempo con duración limitada y con un valor promedio cero. Esto se puede expresar matemáticamente a través de dos igualdades, una función wavelet $\psi(t) \in L^2(\mathbf{R})$ debe de satisfacer las siguientes condiciones:

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0 \qquad \int_{-\infty}^{\infty} \psi^2(t)\, dt = 1$$

Los investigadores en el campo han propuesto y desarrollado funciones wavelets con el objetivo de abordar diferentes problemas y aplicaciones. Algunas de las wavelets más populares son las wavelets de Haar [21], Daubechies [12], Coiflet [12], Symlet [3] y Morlet [10].

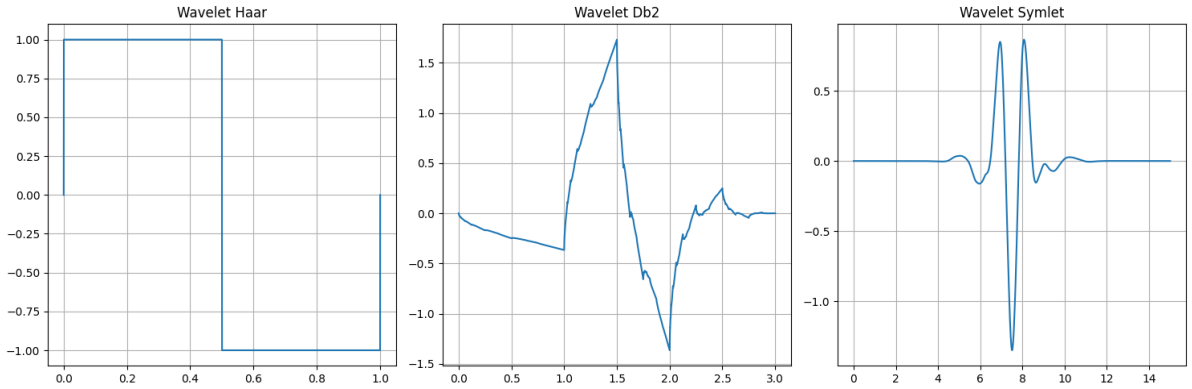


Figura 3.1: Gráficas de las wavelets Haar, Db2 (Daubechies 2) y Symlet generadas con Python y PyWavelets (pywt).

La wavelet de **Morlet** es usada en aplicaciones de diagnóstico de fallas mecánicas [17] y análisis de señales biomédicas [4]. Por otro lado, la wavelet **Coiflet** es usada en el monitoreo y análisis de perturbaciones que pueden afectar la operación de equipos electrónicos y generar problemas de calidad de energía eléctrica [15]. La wavelet **Symlet**, que exhibe una simetría mayor en comparación con la wavelet Daubechies, por ejemplo, resulta útil en métodos avanzados para mejorar la detección automática de episodios epilépticos a partir de señales EEG [27].

Dentro del contexto del WMRA, es crucial seleccionar una familia de wavelets que incluya una función llamada 'escala'. En este capítulo, se definirán los conceptos fundamentales derivados de esta selección: los **espacios de aproximación** y los **espacios de detalle**. Estos espacios se generaran a partir de conjuntos de funciones derivadas de la **función de escala** (también conocida como función padre) y de la **función wavelet** (también llamada función madre).

Definición 3.1.1. *Decimos que una función $\phi(t) \in L^2(\mathbf{R})$ es una* ***función escala*** *si el conjunto de sus traslaciones enteras $\{\phi(t-k);\ k \in \mathbf{Z}\}$, forman una base ortonormal para el subespacio cerrado $V_0 \subset L^2(\mathbf{R})$, donde $V_0 := \overline{gen\{\phi(t-k);\ k \in \mathbf{Z}\}}$. Esto quiere decir que:*

$$\langle \phi_{0,k}(t), \phi_{0,m}(t) \rangle = \int_{-\infty}^{\infty} \phi_{0,k}(t)\phi_{0,m}(t)\ dt = \int_{-\infty}^{\infty} \phi(t-k)\phi(t-m)\ dt = \begin{cases} 1, & si\ k = m \\ 0, & en\ otro\ caso \end{cases}$$

donde $\phi_{0,k}(t) := \phi(t-k)$. Además el subespacio V_0 es conocido como el espacio de aproximación de escala unitaria.

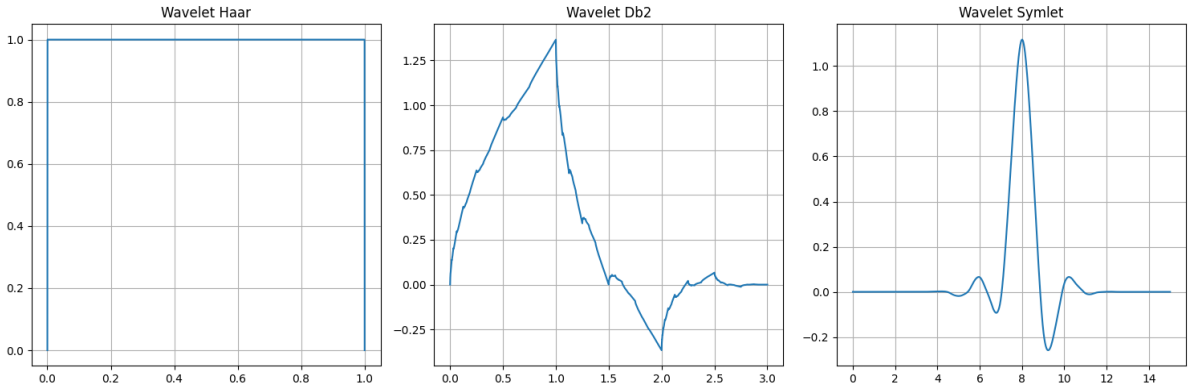


Figura 3.2: Gráficas de las funciones escalas Haar, Db2 (Daubechies 2) y Symlet generadas con Python y PyWavelets (pywt).

El proceso de realizar dilataciones de la función de escala en potencias de dos facilita un análisis estructurado y eficiente. La variedad de escalas permite la creación de nuevos subespacios de aproximación anidados. Esta estructura jerárquica resultante se convierte en un elemento clave para el análisis multiresolución. La siguiente definición resume formalmente la estructura deseada.

Definición 3.1.2. *Decimos que una sucesión de subespacios cerrados* $\{V_j;\ \ j \in \mathbf{Z}\}$ *de funciones de* $L^2(\mathbf{R})$ *es llamado un* ***análisis de multiresolución wavelet (WMRA)*** *si satisface las siguientes condiciones:*

1. *Los espacios están anidados* $V_j \subset V_{j-1} \quad \forall j \in \mathbf{Z}$.
2. *La sucesión es densa en* $L^2(\mathbf{R})$, *es decir,* $\overline{\bigcup V_j} = L^2(\mathbf{R})$.
3. *La intersección de los subespacios es el subespacio trivial* $\bigcap V_j = \{0\}$.
4. *La descomposición en múltiples niveles de resolución establece una relación entre las diferentes escalas:*
   - $f(t) \in V_0$ *si y sólo si para todo* $k \in \mathbf{Z}, \quad f(t-k) \in V_0$.
   - $f(t) \in V_0$ *si y sólo si para todo* $k \in \mathbf{Z}, \quad 2^{-j/2} f(2^{-j}t) \in V_j$.
5. *Existe una función escala* $\phi(t) \in V_0$ *y por tanto el conjunto* $\{\phi(t-k);\ k \in \mathbf{Z}\}$ *es una base ortonormal para este subespacio.*

El subespacio V_j *es conocido como* ***espacio de aproximación*** *para la escala* $\lambda_j = 2^j$.

Observación 1. Podemos analizar cualquier función $f(t) \in L^2(\mathbf{R})$ con respecto a V_0 haciendo proyecciones sobre su base.
Si $f(t) \in V_0$,

$$f(t) = \sum_{k=-\infty}^{\infty} a_k^0 \phi_{0,k}(t) = \sum_{k=-\infty}^{\infty} a_k^0 \phi(t-k)$$

donde $a_k^0 := \langle f(t), \phi_{0,k}(t)\rangle = \int_{-\infty}^{\infty} f(t)\phi_{0,k}(t)\ dt = \int_{-\infty}^{\infty} f(t)\phi(t-k)\ dt$ son los llamados **coeficientes de aproximación** en escala unitaria.

En el caso de que $f(t) \notin V_0$, la expresión anterior representa una aproximación a $f(t)$. Dependiendo de la complejidad de la función, esta aproximación puede mejorar si proyectamos $f(t)$ en subespacios anidados con otras escalas.

Ejemplo 3.1.3. Consideremos la función escala Haar dada por la expresión:

$$\phi^H(t) = \begin{cases} 1, & \text{si } 0 < t \leq 1 \\ 0, & \text{otro caso.} \end{cases}$$

Entonces las funciones bases que componen al espacio V_0 tienen la forma:

$$\phi_{0,k}^H(t) = \phi^H(t-k) = \begin{cases} 1, & \text{si } k < t \leq k+1 \\ 0, & \text{otro caso,} \end{cases}$$

de las cuales fácilmente se puede verificar la condición de ortogonalidad. Además si $f(t) \in V_0$ podemos enontrar los coeficientes de aproximación en la escala unitaria:

$$f(t) = \sum_{k=-\infty}^{\infty} a_k^0 \phi_{0,k}^H(t) = \sum_{k=-\infty}^{\infty} a_k^0 \phi^H(t-k) = a_k^0, \quad \text{para } k < t \leq k+1.$$

Esto nos dice que el **espacio de aproximación** de escala unitaria consiste de todas las funciones en $L^2(\mathbf{R})$ que son constantes sobre cada intervalo de forma: $k < t \leq k+1$, para todo $k \in \mathbf{Z}$, es decir intervalos de longitud uno. En el caso $j = -1$ serán funciones constantes en intervalos de longitud $1/2$, y así sucesivamente.

Teorema 3.1.4. *El conjunto* $\{2^{-j/2}\phi(2^{-j}t - k);\ k \in \mathbf{Z}\}$ *es una base ortonormal para el espacio de aproximación* V_j *con escala* λ_j*. Estas funciones se denotan como* $\phi_{j,k}(t) = 2^{-j/2}\phi(2^{-j}t - k)$*.*

Demostración. Gracias a la existencia de una base ortonormal en V_0 las siguientes igualdades se mantienen:

$$\int_{-\infty}^{\infty} \phi_{0,k}(t)\phi_{0,m}(t)\ dt = \int_{-\infty}^{\infty} \phi(t-k)\phi(t-m)\ dt = \begin{cases} 1, & \text{si } k = m \\ 0, & \text{en otro caso.} \end{cases}$$

Por otra lado, podemos deducir que calcular producto interno en la escala unitaria es equivalente a calcular el producto interno en en la escala λ_j haciendo una dilatación de 2^j de la siguiente forma:

$$\langle \phi_{0,k}(t), \phi_{0,m}(t)\rangle = \int_{-\infty}^{\infty} \phi_{0,k}(t)\phi_{0,m}(t)\ dt = \int_{-\infty}^{\infty} \frac{\phi_{0,k}(2^{-j}t)\phi_{0,m}(2^{-j}t)}{\sqrt{2^j}\sqrt{2^j}}\ dt = \langle \phi_{j,k}(t), \phi_{j,m}(t)\rangle$$

Entonces la ortonormalidad en la escala λ_j se hereda de la escala unitaria. □

Observación 2. Por definición de WMRA los subespacios V_j son cerrados, entonces por el teorema anterior podemos concluir que $V_j = \overline{\text{gen}\{\phi_{j,k}(t);\ k \in \mathbf{Z}\}}$. Esto garantiza que la aproximación de una función en un subespacio de aproximación pueda ser representada de manera única en una misma escala.

Ahora presentaremos **el teorema de la relación de dos escalas**, el cual establece una relación recursiva fundamental para las funciones de escala en el análisis de multiresolución.

Teorema 3.1.5 ([5]). *Supongamos que $\{V_j;\ j \in \mathbf{Z}\}$ es un análisis de multiresolución con función escala $\phi(t)$. Entonces se tiene las siguientes igualdades:*

$$\phi(t) = \sum_{k=-\infty}^{\infty} p_k \phi(2t-k),\ \textit{donde}\ p_k = 2\int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt$$

Además, se cumple también:

$$\phi(2^{j-1}t - l) = \sum_{k=-\infty}^{\infty} p_{k-2l}\phi(2^j t - k)$$

ó equivalentemente

$$\phi_{1-j,l}(t) = 2^{-1/2}\sum_{k} p_{k-2l}\phi_{-j,k}(t)$$

La teoría del análisis de espacios de Hilbert garantiza la existencia del complemento ortogonal de cualquier subespacio cerrado. Por lo tanto, para cada subespacio V_{j+1}, deseamos encontrar su complemento ortogonal $V_{j+1}^{\perp}$ dentro de V_j. Estos espacios nos ayudarán a mejorar las aproximaciones de las funciones en $L^2(\mathbf{R})$. Para ser más precisos estos espacios se definen como:

Definición 3.1.6. *Sea $\{V_j;\ j \in \mathbf{Z}\}$ un análisis multiresolución, entonces cada subespacio V_{j+1} tiene su complemento ortogonal en V_j, el cual denotaremos por W_{j+1}, este espacio es llamado el* ***espacio detalle para la escala*** *λ_j .*

Así, escribimos V_j como una suma directa:

$$V_j = V_{j+1} \oplus W_{j+1}$$

que por iteración se puede reescribir como:

$$V_j = W_{j+1} \oplus W_{j+2} \oplus \cdots$$

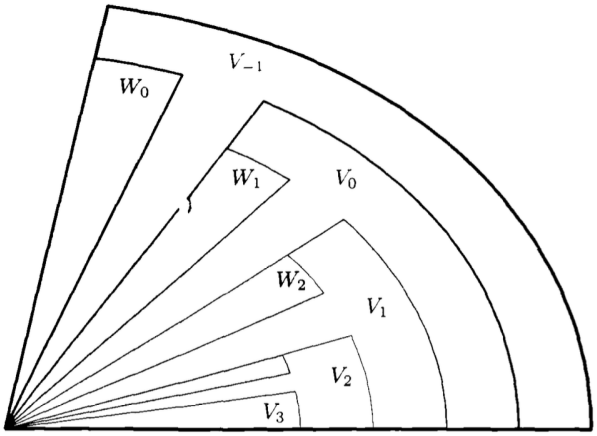


Figura 3.3: En la figura se presenta un diagrama de Venn que ilustra los espacios anidados V_j y W_j [21].

Teorema 3.1.7 ([5]). *Supongamos que* $\{V_j;\ j \in \mathbf{Z}\}$ *es un análisis de multiresolución con función escala dada por:*

$$\phi(t) = \sum_k p_k \phi(2t-k) \quad donde \quad p_k = 2\int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt$$

Entonces la expresión:

$$\psi(t) = \sum_k (-1)^k p_{1-k}\phi(2t-k)$$

cumple las condiciones de una wavelet y los espacios ortogonales W_j *son los subespacios* $\overline{gen\{\psi_{j,k}(t);\ k \in \mathbf{Z}\}}$, *donde* $\psi_{j,k}(t) := 2^{-j/2}\psi(2^{-j}t-k)$.

Ejemplo 3.1.8. La Wavelet de Haar
En este ejemplo, exploraremos la construcción de la función wavelet de Haar $\psi^H(t)$ a partir de su función de escala asociada $\phi^H(t)$ y de su análisis de multiresolución $\{V_j\}_{j\in\mathbb{Z}}$. También veremos cómo construir los espacios de detalle utilizando la definición de complemento ortogonal.

La función de escala de Haar está dada por la fórmula:

$$\phi^H(t) = \begin{cases} 1, & \text{si } 0 \le t < 1, \\ 0, & \text{en otro caso.} \end{cases}$$

Deseamos encontrar el subespacio de detalle para V_j. En otras palabras, queremos expresar V_j como una suma directa de subespacios:

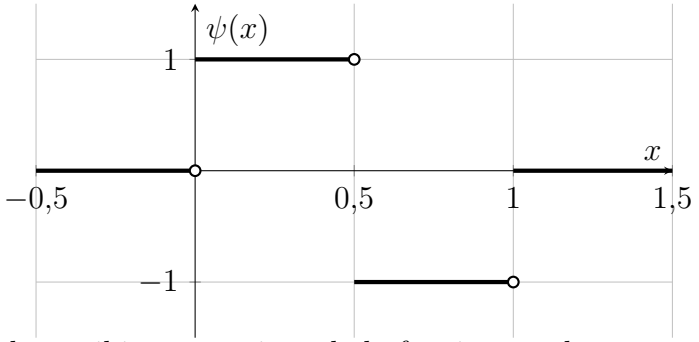
$$V_j = V_{j+1} \oplus W_{j+1}$$

Para ser más ilustrativos, encontraremos el subespacio ortogonal de V_0, que corresponde al caso $j = -1$ y luego haremos una generalización. El subespacio V_0 es generado por las traslaciones de la función de escala $\phi^H(t)$, entonces proponemos que el complemento ortogonal se genere por las traslaciones de una función de soporte compacto que denotamos por $\psi^H(t)$. Por tanto, esta función debe de satisfacer las siguientes condiciones:

- Dado que $\psi^H(t) \in V_{-1}$, se puede expresar como una combinación lineal de funciones en V_{-1}. Es decir, existen escalares p_k tales que $\psi^H(t) = \sum a_k \phi^H(2t-k)$. Gracias a la condición de soporte compacto, podemos garantizar que solo un número finito de estos escalares son diferentes de cero.
- W_0 es ortogonal a V_0, entonces para todos los enteros k, $\int_{-\infty}^{\infty} \psi^H(x)\phi^H(x-k)dx = 0$.

Si tomamos $k = 0$, se puede deducir fácilmente que la función más simple que satisface ambas condiciones es:

$$\psi^H(t) = \begin{cases} 1, & \text{si } 0 \le t < \frac{1}{2}, \\ -1, & \text{si } \frac{1}{2} \le t < 1, \\ 0, & \text{en otro caso.} \end{cases}$$



Observemos que la función wavelet se puede escribir en términos de la función escala:

$$\psi^H(t) = \phi^H(2t) - \phi^H(2t-1) \tag{3.1}$$

entonces los coeficientes de la primera condición son $a_0 = 1$ y $a_1 = -1$.

En el caso general, demostraremos que para cualquier j, el espacio W_{j+1} está constituido por el conjunto de funciones de la forma:

$$\sum_{k\in\mathbb{Z}} a_k \psi(2^{-(j+1)}t - k), \quad a_k \in \mathbf{R},$$

donde solo un número finito de los coeficientes a_k son diferentes de cero.

Nuevamente, para esta demostración debemos verificar dos condiciones:

- Cada función de W_{j+1} es ortogonal a cada función de V_{j+1}.
- Cualquier función de V_j que sea ortogonal a V_{j+1} debe de pertenecer a W_{j+1}.

Sea $g(t) \in W_{j+1}$, entonces $g(t)$ se puede escribir como $g(t) = \sum_{k\in\mathbb{Z}} a_k \psi(2^{-(j+1)}t - k)$. Queremos mostrar que $g(t)$ es ortogonal a cualquier $f(t) \in V_{j+1}$.

$$\langle \sum_{k\in\mathbf{Z}} a_k \psi(t-k), 2^{(j+1)/2} f(2^{j+1}t)\rangle = 0, \quad \text{porque } 2^{(j+1)/2} f(2^{j+1}t) \in V_0$$

$$\int_{-\infty}^{\infty} \sum_{k\in\mathbf{Z}} a_k \psi(t-k) \cdot 2^{(j+1)/2} f(2^{j+1}t)dt = 0$$

$$2^{-(j+1)/2} \int_{-\infty}^{\infty} \sum_{k\in\mathbf{Z}} a_k \psi(2^{-(j+1)}y - k) f(y)dy = 0, \quad \text{con el cambio de variable } y = 2^{j+1}t$$

$$2^{-(j+1)/2} \langle g(y), f(y)\rangle = 0$$

Con esto, hemos demostrado la primera condición. La segunda condición se deriva directamente del resultado presentado en [5] y con esto concluimos el ejemplo.

El siguiente teorema proporciona las fórmulas para calcular los coeficientes en los espacios de detalle y aproximación utilizando la wavelet Haar.

Teorema 3.1.9 ([5]). ***Descomposición de Haar.*** *Supongamos que tenemos un análisis de multiresolución de la wavelet Haar, y sea*

$$f_j(t) = \sum_{k\in\mathbb{Z}} a_k^j \phi(2^{-j}t - k) \in V_j.$$

Entonces, $f_j(t)$ se puede descomponer como

$$f_j(t) = w_{j+1}(t) + f_{j+1}(t),$$

donde

$$w_{j+1}(t) = \sum_{k\in\mathbb{Z}} b_k^{j+1} \psi(2^{-(j+1)}t - k) \in W_{j+1},$$

$$f_{j+1}(t) = \sum_{k\in\mathbb{Z}} a_k^{j+1} \phi(2^{-(j+1)}t - k) \in V_{j+1},$$

con

$$b_k^{j+1} = \frac{a_{2k}^j - a_{2k+1}^j}{2}, \quad a_k^{j+1} = \frac{a_{2k}^j + a_{2k+1}^j}{2}.$$

Durante todo el capítulo hemos trabajado con espacios de funciones de Hilbert en $L^2(\mathbf{R})$ y hemos mencionado que el WMRA se aplica al procesamiento de señales. En términos matemáticos, las señales pueden ser continuas o discretas. Las señales continuas son funciones continuas a trozos. En nuestro caso, estamos interesados en aplicar la teoría a series de precios, que no son más que señales discretas. Estas señales se pueden considerar como sucesiones de elementos en el espacio $\ell^2(\mathbf{R})$.

En el teorema 3.1.9, vimos cómo hacer una descomposición de una señal de forma constructiva. Sin embargo, estas descomposiciones se pueden calcular de forma más simple utilizando filtros y convoluciones sobre sucesiones. A continuación, definiremos estos conceptos principalmente en términos de sucesiones, aunque también tienen su versión continua.

Definición 3.1.10. *Un filtro se define como un operador $\mathcal{F}$ que actúa sobre señales (pueden ser continuas o discretas), estos filtros deben de ser lineales e invariantes en en el tiempo, es decir, si f y g son señales:*

- ***Linealidad.*** $\mathcal{F}(f(t) + cg(t)) = \mathcal{F}(f(t)) + c\mathcal{F}(g(t))$, *para cualquier constante c.*
- ***Invarianza en el tiempo.*** $\mathcal{F}(f_a) = \mathcal{F}_a(f)$, *donde $f_a(t) = f(t-a)$ y $\mathcal{F}_a(f)(t) = (\mathcal{F}f)(t-a)$.*

La operación de traslación en el caso discreto, aplicada a una señal x_t, está definida para cada entrada t-ésima $[\mathcal{F}_a(x)]_t = \mathcal{F}(x)_{t-a}$

Definición 3.1.11. *La **convolución**, denotada por $*$, es una operación que, en el caso discreto, se define para dos sucesiones x_t y y_t de la siguiente manera:*

$$(x * y)_t = \sum_{k=-\infty}^{\infty} x_k y_{t-k},$$

siempre que las series involucradas sean absolutamente convergentes.

Teorema 3.1.12 ([5])**.** *Sea $\mathcal{F}$ un operador que actúa sobre señales discretas x_t. El operador $\mathcal{F}$ es un filtro si y sólo si existe una sucesión f_t tal que para toda sucesión x_t:*

$$\mathcal{F}(x)_t = (f * x)_t,$$

*siempre que las series involucradas sean absolutamente convergentes. La sucesión f_t es denominada **respuesta a impulso**.*

Definición 3.1.13. ***Filtro paso alto y paso bajo.*** *Supongamos que $\{V_j;\ j \in \mathbf{Z}\}$ es un análisis de multiresolución con función escala dada por:*

$$\phi(t) = \sum_k p_k \phi(2t-k) \quad donde \quad p_k = 2\int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt$$

y consideremos las sucesiones g_k y h_k definidas por:

$$g_k := \frac{1}{2}(-1)^k p_{k+1} \quad h_k := \frac{1}{2}p_{-k} \tag{3.2}$$

Definimos filtros discretos paso alto $\mathcal{G}$ y paso bajo $\mathcal{H}$ a partir de sus respuesta de impulso g_k y h_k respectivamente:

$$\mathcal{G}(x)_k = (g * x)_k \qquad \mathcal{H}(x)_k = (h * x)_k$$

El filtro de paso bajo (LPF) se utiliza para suavizar la señal y obtener la aproximación de baja frecuencia en cada escala de descomposición, mientras que el filtro de paso alto (HPF) se emplea para extraer los detalles de alta frecuencia en cada escala de descomposición.

Teorema 3.1.14 ([5])**.** ***Coeficientes de descomposición.*** *Sea $x(t)$ una señal en algún subespacio V_j, y consideremos la suma directa de los subespacios aproximación y detalle $V_j = V_{j+1} \oplus W_{j+1}$, entonces existe una descomposición en la concatenación de bases $\{\phi_{j+1,k}\}_{k\in\mathbb{Z}} \cup \{\psi_{j+1,k}\}_{k\in\mathbb{Z}}$ dada por:*

$$x(t) = \sum_{k\in\mathbb{Z}} a_k^{j+1}\phi_{j+1,k}(t) + \sum_{k\in\mathbb{Z}} b_k^{j+1}\psi_{j+1,k}(t)$$

ó equivalentemente

$$x(t) = \sum_{k\in\mathbb{Z}} a_k^{j+1}\phi(2^{-(j+1)}t-k) + \sum_{k\in\mathbb{Z}} b_k^{j+1}\psi(2^{-(j+1)}t-k)$$

donde los coeficientes están dados por las fórmulas:

$$a_l^{j+1} = 2^{-1} \sum_{k \in \mathbf{Z}} p_{k-2l} a_k^j \tag{3.3}$$

$$b_l^{j+1} = 2^{-1} \sum_{k \in \mathbf{Z}} (-1)^k p_{1-k+2l} a_k^j \tag{3.4}$$

y $a_k^j = 2^{-j/2} \langle x(t), \phi_{j,k}(t) \rangle$.

Definición 3.1.15. *Dada una sucesión* x_n*, el* ***downsampling*** D *por un factor* M *es una operación que genera una nueva sucesión* y_m*, definida por:*

$$y_m = D(x_n) = x_{mM}$$

Esto significa que la sucesión y_m *contiene solo cada* M*-ésima entrada de la sucesión original* x_n*, eliminando las demás entradas. En el caso de WMRA se toma* $M = 2$.

Corolario 3.1.16. *Los filtros de paso alto y paso bajo calculan los coeficientes de la descomposición después de realizar la operación de downsampling en una escala* λ_j.

Demostración. Vamos a tomar $x = a^j$ en la definición de filtros 3.1.13 y calculamos:

$$\mathcal{H}(a^j)_l = (h * a^j)_l = \sum_{k=\infty}^{\infty} h_k a_{l-k}^j = \sum_{k=\infty}^{\infty} \frac{1}{2} p_{-k} a_{l-k}^j = \frac{1}{2} \sum_{k=\infty}^{\infty} p_{k-l} a_k^j$$

$$\mathcal{G}(a^j)_l = (g * a^j)_l = \sum_{k=-\infty}^{\infty} g_k a_{l-k}^j = \sum_{k=-\infty}^{\infty} \frac{1}{2} (-1)^k p_{k+1} a_{l-k}^j = \frac{1}{2} \sum_{k=-\infty}^{\infty} (-1)^{l-k} p_{1-k+l} a_k^j$$

Si comparamos los valores obtenidos con las ecuaciones 3.3 y 3.4, podemos concluir que se obtienen los resultados esperados:

$$a_l^{j+1} = \mathcal{H}(a^j)_{2l} \quad \text{y} \quad b_l^{j+1} = \mathcal{G}(a^j)_{2l}.$$

□

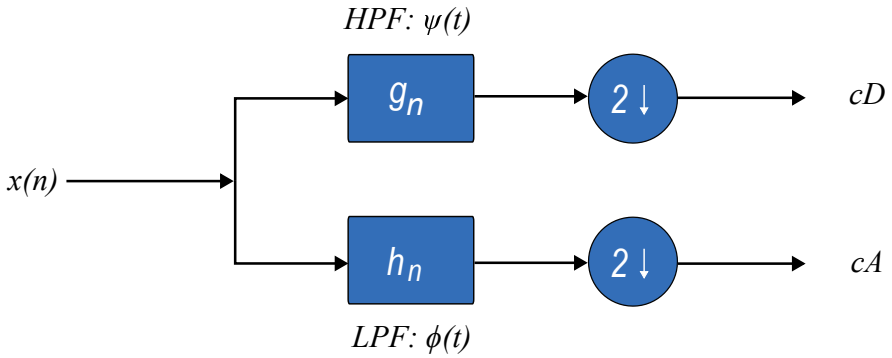


Figura 3.4: Señal descompuesta en componentes de detalle (cD) y aproximación (cA) mediante filtros paso bajo y paso alto.

Los filtros se dividen generalmente en dos categorías principales: filtros de longitud finita (FIR) y filtros de longitud infinita (IIR). En el contexto del análisis de multiresolución, el soporte compacto de la función de escala implica que esta función es diferente de cero solo en un intervalo finito. Esto se traduce directamente en la finitud de los coeficientes del filtro asociados, definiendo **longitud del filtro** a el número de elementos diferentes de cero en la respuesta al impulso del filtro. Familias de wavelets con soporte compacto comunes son las wavelets de Haar, Daubechies, Symlets, y Coiflets.

La cuadratura de espejo es un método utilizado en el diseño de filtros para obtener un filtro de paso alto a partir de un filtro de paso bajo existente, ó para obtener un filtro paso bajo a partir de un filtro paso alto.

Observación 3. Cuadratura de espejo. La relación entre los coeficientes de los filtros (FIR), es decir, la relación entre las respuestas de impulso h_k y g_k, se describe mediante:

$$h_k = (-1)^{k+1} g_{N-1-k}$$

y de manera equivalente:

$$g_k = (-1)^k h_{N-1-k}$$

Donde N es la longitud del filtro.

Demostración. De la definición de filtros 3.1.13 tenemos que para un filtro de longitud N la respuesta de impulso $h_{\tilde{k}} = \frac{1}{2} p_{N-\tilde{k}}$, entonces evaluando $\tilde{k} = N-k-1$ se sigue $p_{k+1} = 2h_{N-k-1}$.

Nuevamente utilizando la definición 3.1.13 la respuesta de impulso g_k:

$$g_k = \frac{1}{2}(-1)^k p_{k+1} = \frac{1}{2}(-1)^k 2h_{N-k-1} = (-1)^k h_{N-k-1}$$

Similarmente se obtiene la otra igualdad. □

Definición 3.1.17. *Sea $\{h_l\}_{l\in\mathbf{Z}}$ una respuesta a impulso de filtro de longitud finita, decimos que los coeficientes del filtro están normalizados si:*

$$\sum_{l\in\mathbf{Z}} h_l^2 = 1$$

Ejemplo 3.1.18. Filtro de Haar. En la práctica, los filtros de las wavelets suelen ser consultados directamente, ya que se emplean wavelets conocidas. Sin embargo, decidimos realizar la tarea de construir el filtro de la wavelet Haar de manera constructiva, con el fin de mejorar la comprensión del concepto.

Como se observó en el ejemplo de la wavelet de Haar 3.1.8, la expresión de la función wavelet en función de la función escala está dada por:

$$\psi^H(t) = \phi^H(2t) - \phi^H(2t-1)$$

Comparando esta expresión con la definición de la función wavelet del teorema 3.1.7 en términos de los coeficientes p_k, obtenemos que $p_0 = p_1 = 1$.

Entonces podemos calcular la respuesta de impulso del filtro paso alto y filtro paso bajo dados por la formulas 3.2:

$$g_k = \{\dots, 0, \underbrace{-\frac{1}{2}, \frac{1}{2}}_{k=-1,0}, 0, \dots\}$$

$$h_k = \{\dots, 0, \underbrace{\frac{1}{2}, \frac{1}{2}}_{k=-1,0}, 0, \dots\}$$

y con esto calculamos los filtros:

$$\mathcal{G}(x)_k = (g * x)_k = \frac{1}{2}x_k - \frac{1}{2}x_{k+1}$$

$$\mathcal{H}(x)_k = (h * x)_k = \frac{1}{2}x_k + \frac{1}{2}x_{k+1}$$

Con esto, al evaluar $x = a^j$ obtenemos los coeficientes de aproximación y detalle haciendo uso de filtros, que correspondes a los mismos coeficientes constructiva en el teorema 3.1.9 calculados de forma constructiva:

$$a_k^{j+1} = D\mathcal{H}(a^j)_k \qquad b_k^{j+1} = D\mathcal{G}(a^j)_k$$

En el caso de la implementación usaremos los coeficientes de los filtros normalizados.

Ejemplo 3.1.19. Wavelets Daubechies Para realizar un análisis de multiresolución en la implementación de este trabajo, seleccionamos la wavelet de Daubechies db2.

Las wavelets de Daubechies, denotadas como dbn, en honor a Ingrid Daubechies, quien las introdujo en 1988 [11], representan una familia de wavelets ortogonales[1] utilizadas para llevar a cabo un análisis multiresolución de señales discretas. Entre ellas, el caso más elemental surge cuando $n = 1$, correspondiendo a las conocidas wavelets de Haar. Por otro lado, para $n \geq 2$, estas wavelets se distinguen por su continuidad y su soporte compacto.

Es importante destacar que las wavelets ortogonales, como las de Daubechies, gozan de amplio uso en aplicaciones de compresión de datos. Su uso en estos trabajos se justifica por la demostrada eficiencia en la representación y reconstrucción precisa de señales, lo que las convierte en herramientas valiosas para nuestro análisis. En particular, se trabaja con $n = 2$, porque es la familia de Daubechies con menor longitud en su filtro, lo que logra obtener mejor tiempo de ejecución.

[1]Lo que significa que las funciones base (tanto de los espacios de aproximación como de los espacios de detalle) son ortogonales entre sí.

Coeficientes de la descomposición db2

Los coeficientes de los filtros paso bajo y paso alto de la wavelet db2 para definir el filtro paso alto $\mathcal{G}$ y el filtro paso bajo $\mathcal{H}$ los consultamos en [5]:

$$h_k = \{\dots, 0, \underbrace{\frac{p_3}{2}, \frac{p_2}{2}, \frac{p_1}{2}, \frac{p_0}{2}}_{k=-3,-2,-1,0}, 0, 0, \dots\}$$

$$g_k = \{\dots, 0, 0, 0, \underbrace{\frac{-p_0}{2}, \frac{p_1}{2}, \frac{-p_2}{2}, \frac{p_3}{2}}_{K=-1,0,1,2}, \dots\}$$

donde:

$$p_0 = \frac{1+\sqrt{3}}{4} \quad p_1 = \frac{3+\sqrt{3}}{4} \quad p_2 = \frac{3-\sqrt{3}}{4} \quad p_3 = \frac{1-\sqrt{3}}{4}$$

Entonces los coeficientes de aproximación a_k^{j+1} y detalle b_k^{j+1} están dados por:

$$a_k^{j+1} = D(h * a^j)_k = \frac{1}{2}\left(p_0 a_{2k}^j + p_1 a_{2k+1}^j + p_2 a_{2k+2}^j + p_3 a_{2k+3}^j\right), \tag{3.5}$$

$$b_{k+1}^{j+1} = D(g * a^j)_{k+1} = \frac{1}{2}\left(p_3 a_{2k}^j - p_2 a_{2k+1}^j + p_1 a_{2k+2}^j - p_0 a_{2k+3}^j\right). \tag{3.6}$$

El articulo de estudio central de esta tesis [9] realiza el análisis multiresolución para obtener los coeficientes que son necesarios en el algoritmo matching pursuit usando la descomposición que se enunciaran en el siguiente teorema. Sin embargo, nosotros empleamos una técnica mucho mas refinada que se explicará después del ejemplo de este teorema.

Teorema 3.1.20 ([21])**.** *Sea $\{V_j,\ j \in \mathbf{Z}\}$ un análisis multiresolución con función escala $\phi(t)$, y función wavelet $\psi(t)$, entonces obtenemos los siguientes resultados:*

- *Para cualquier j, tenemos una suma directa de subespacios:*

$$\overline{V_j \oplus W_j \oplus W_{j-1} \oplus \cdots} = \overline{\bigcup_{l \in \mathbf{Z}} V_l} = L^2(\mathbf{R})$$

  De aquí obtenemos la aproximación a cualquier señal $x(t)$, así:

$$x(t) \approx \sum_{k=-\infty}^{\infty} a_k^j \phi_{j,k}(t) + \sum_{l=-\infty}^{j} \sum_{k=-\infty}^{\infty} b_k^l \psi_{l,k}(t).$$

  Los a_k^j son denominados coeficientes de aproximación y b_k^l coeficientes de detalle.

- *El conjunto $\{\psi_{j,k}(t),\ k \in \mathbf{Z}\}$ es una base ortonormal para W_j.*

- *Además que tenemos una suma directa que depende únicamente de la función wavelet:*

$$\overline{\bigoplus_{l \in \mathbf{Z}} W_l} = L^2(\mathbf{R})$$

  y de aquí cualquier señal $x(t)$ puede ser expresada de forma única como:

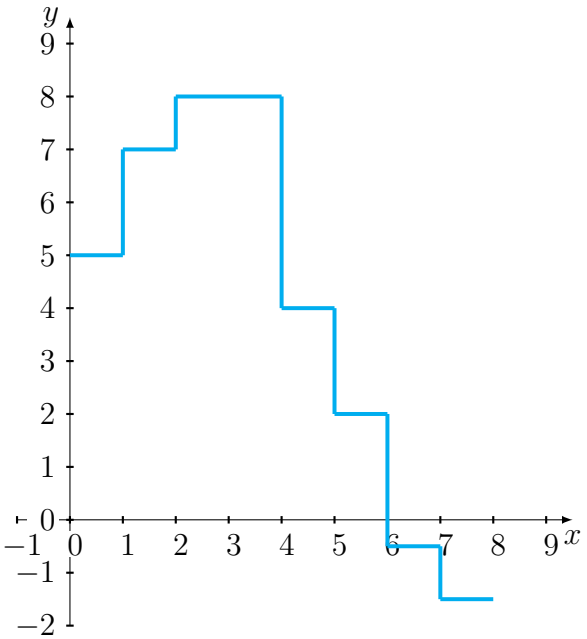
$$x(t) \approx \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} b_k^l \psi_{l,k}(t)$$

Para la demostración de este teorema, refiérase a [21], página 482.

Ejemplo 3.1.21. Realizaremos un análisis multiresolución utilizando la wavelet de Haar. Consideremos la señal discreta $\{x_t\}_{t\in\mathbf{Z}} = \{\ldots, 0, \underbrace{5, 7, 8, 8, 4, 2, -0{,}5, -1{,}5}_{t=0,1,2,3,4,5,6,7}, 0, \ldots\}$, que representaremos gráficamente como una función a trozos para interpretar visualmente los filtros.

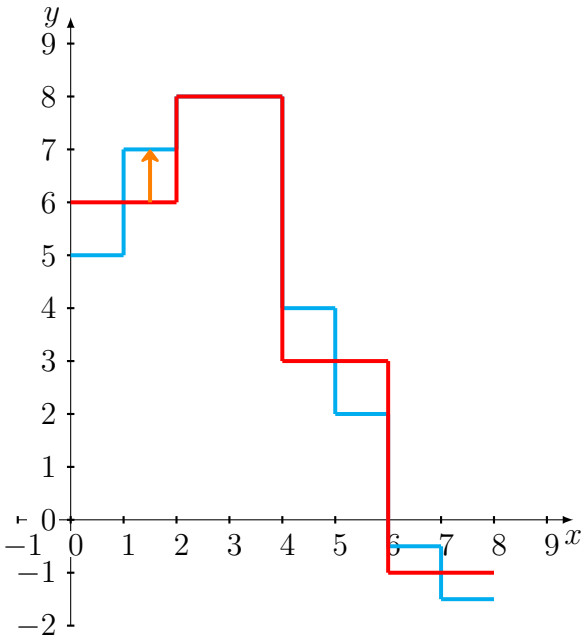
En el teorema 3.1.9 se calcularon los coeficientes de descomposición de la wavelet Haar, de esas ecuaciones podemos interpretar que los coeficientes de aproximación se obtienen mediante un proceso de promedio ponderado, mientras que los coeficientes de detalle representan las diferencias entre la señal original y su versión aproximada (también llamada fluctuación).

En la gráfica, el color rojo representa los coeficientes de aproximación y el azul los coeficientes de detalle correspondiente a cada escala.



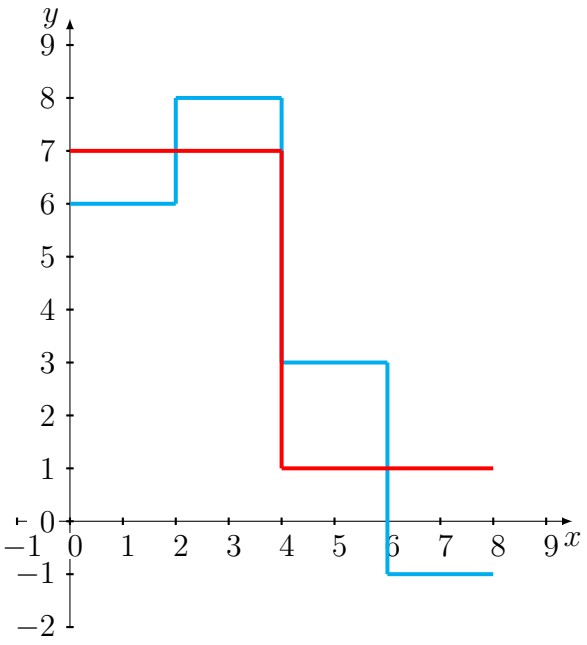
Señal

5	7	8	8	4	2	-0.5	-1.5



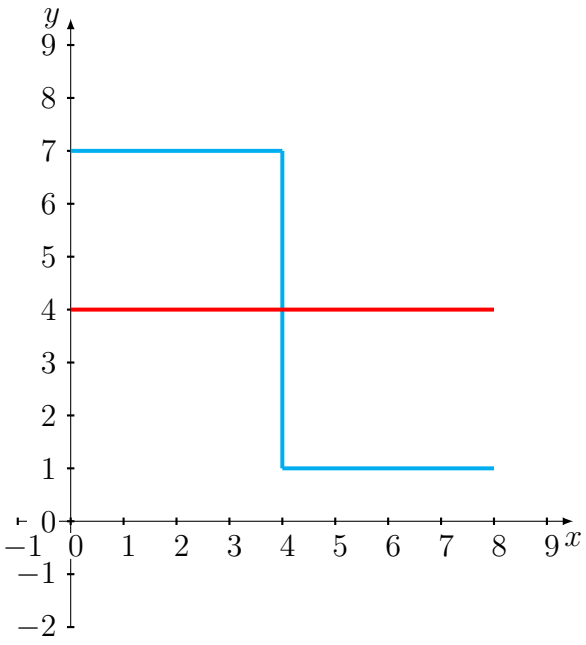
Primera Descomposición λ_1

6	8	3	-1	-1	0	1	0.5



Segunda Descomposición λ_2

7	1	-1	2	-1	0	1	0.5



Tercera Descomposición λ_3

4	3	-1	2	-1	0	1	0.5

3.1.1. Paquetes Wavelet

El concepto de paquetes wavelets fue introducido por primera vez por Coifman, Meyer y Wickerhauser a principios de la década de 1990. La idea principal es proporcionar una herramienta que permita una descomposición más completa y flexible de las señales en términos de frecuencia y localización en el tiempo. Esto se logra mediante una gradación mas fina en las componentes de frecuencia (relacionando estas con los espacios de detalle antes vistos).

La descomposición de paquetes wavelet es usada en aplicaciones como: eliminación de ruido, comprensión de señal e imagen, análisis espectral, clasificación de señales e imágenes, entre otras. En nuestro caso, nos servirá para crear un diccionario [2] que será parte importante en la creación de características de nuestra red neuronal.

Construcción de paquetes wavelet

Anteriormente, en el análisis de multiresolución, para cada escala solo se calcula el espacio de detalle correspondiente al subespacio de aproximación. Sin embargo, en los paquetes wavelets, buscamos encontrar el espacio de detalle a ambos subespacios: aproximación y detalle. Por ejemplos, supongamos que tenemos una señal $x(t) \in V_0$ que se puede descomponer en dos escalas. Las siguientes imágenes ilustran la diferencia entre una descomposición utilizando WMRA y otra utilizando paquetes de wavelets.

[2] Los diccionarios se definen en 3.1.26, sin embargo hago referencia a este concepto para argumentar la construcción de paquetes wavelets

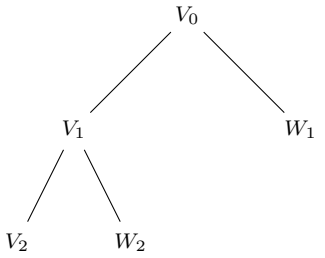


Figura 3.5: Estructura de WMRA

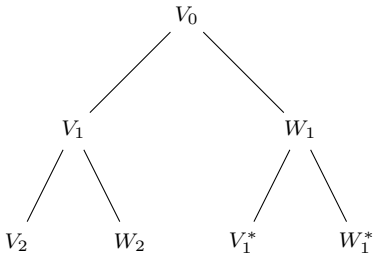


Figura 3.6: Estructura de Paquetes

En el caso de los paquetes wavelets, aparecen nuevos espacios V_1^* y W_1^*, que tienen propiedades similares a los espacios de aproximación y detalle ya construidos en el WMRA. A continuación, construiremos esta nueva estructura, utilizando nuestra herramienta de filtros ya definida.

Cada subespacio estará definido por las bases correspondientes. Estas bases se construirán a partir de las funciones $\{w_n\}_{n\in\mathbf{N}}$ que definiremos utilizando los coeficientes normalizados de los filtros g_k (HPF) y h_k (LPF) de longitud N y de las funciones escala y wavelet seleccionadas.

Definimos la sucesión de funciones $\{w_n\}_{n\in\mathbf{N}}$ de la siguiente forma:

$$w_{2n}(t) = \sqrt{2}\sum_{k=0}^{2N-1} h_{-k} w_n(2t-k) \tag{3.7}$$

$$w_{2n+1}(t) = \sqrt{2}\sum_{k=0}^{2N-1} g_{-k} w_n(2t-k) \tag{3.8}$$

Cuando $n = 0$, las expresiones (3.7) y (3.8) se reducen por definición a las funciones de escala y wavelet:

$$w_0(t) = \sqrt{2}\sum_{k=0}^{2N-1} h_{-k} w_0(2t-k) = \sqrt{2}\sum_{k=0}^{2N-1} h_{-k}\phi(2t-k) \tag{3.9}$$

$$w_1(t) = \sqrt{2}\sum_{k=0}^{2N-1} g_{-k} w_0(2t-k) = \sqrt{2}\sum_{k=0}^{2N-1} g_{-k}\phi(2t-k) \tag{3.10}$$

de las definiciones de los filtros normalizados 3.1.13 obtenemos:

$$h_{-k} = \frac{1}{\sqrt{2}}p_k \quad g_{-k} = \frac{1}{\sqrt{2}}(-1)^k p_{1-k}$$

Reemplazamos en 3.9 y 3.10, luego comparamos con la definición de wavelet presentada en

el teorema 3.1.7:

$$w_0(t) = \sqrt{2}\sum_{k=0}^{2N-1} h_{-k}\phi(2t-k) = \sum_{k=0}^{2N-1} p_k\phi(2t-k) = \phi(t)$$
$$w_1(t) = \sqrt{2}\sum_{k=0}^{2N-1} g_{-k}\phi(2t-k) = \sum_{k=0}^{2N-1} (-1)^k p_{1-k}\phi(2t-k) = \psi(t)$$

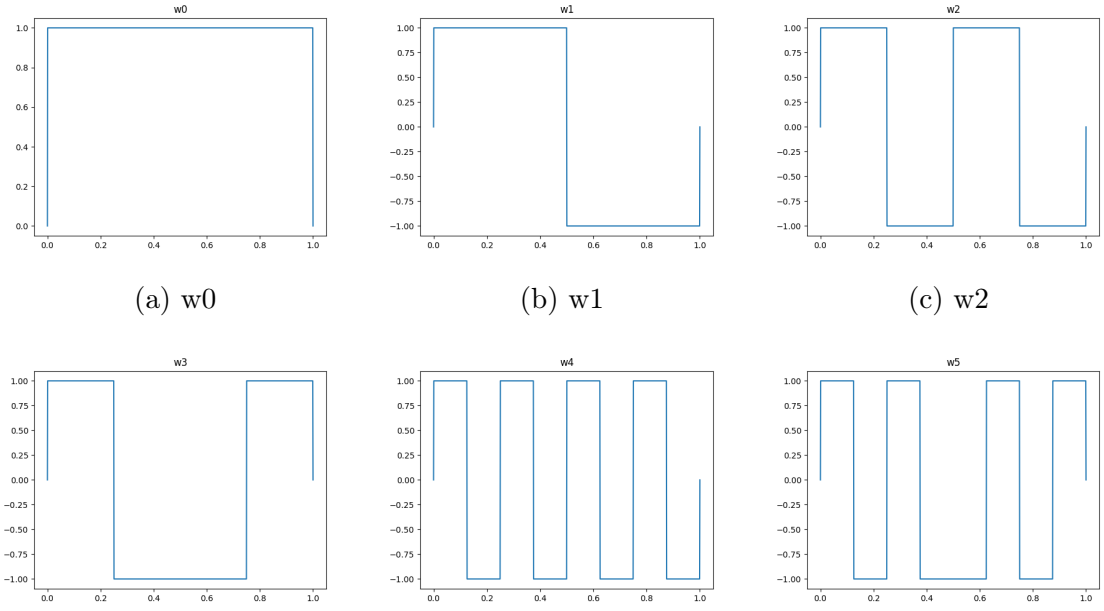
Ejemplo 3.1.22. Consideramos el ejemplo mas simple que es la wavelet Haar, en este caso $N = 1$ y los filtros:

$$h_0 = h_{-1} = \frac{1}{\sqrt{2}} \quad \text{(LPF)} \qquad \text{y} \qquad g_0 = -g_{-1} = \frac{1}{\sqrt{2}} \quad \text{(HPF)}$$

Para los valores de Haar, las ecuaciones (3.7) y (3.8) se reducen a:

$$w_{2n}(t) = w_n(2t) + w_n(2t-1) \tag{3.11}$$
$$w_{2n+1}(t) = w_n(2t) - w_n(2t-1) \tag{3.12}$$

Dado que $w_1(t) = \psi(t)$ para $n = 1$ tenemos que $w_2(t) = \psi(2t) + \psi(2t-1)$.



(a) w0 (b) w1 (c) w2

(d) w3 (e) w4 (f) w5

Figura 3.7: Primeras funciones w_i correspondientes a la wavelet Haar, implementadas en Python.

Implementación de las funciones w_n

Para llevar a cabo esta implementación, nos apoyamos en la biblioteca de código abierto PyWavelets, diseñada en Python.

Esta biblioteca trae consigo una lista de wavelets, donde cada una satisface propiedades particulares:

```
>>> import pywt
>>> pywt.families()
['haar', 'db', 'sym', 'coif', 'bior', 'rbio', 'dmey','gaus', 'mexh', 'morl',
'cgau', 'shan', 'fbsp', 'cmor']
```

De esta biblioteca podemos obtener información básica acerca de las familias de wavelets que ofrece, la descripción para la wavelet de Daubechies db2 es:

```
>>> import pywt
>>> wavelet = pywt.Wavelet('db2')
>>> print(wavelet)

Wavelet db2
  Family name:    Daubechies
  Short name:     db
  Filters length: 4
  Orthogonal:     True
  Biorthogonal:   True
  Symmetry:       asymmetric
  DWT:            True
  CWT:            False
```

En el contexto de la biblioteca PyWavelets (pywt), cuando se establece que DWT es True, implica que se está realizando una Transformada Wavelet Discreta (DWT), lo que significa que se están utilizando funciones de escala para generar un WMRA. Ahora, para invocar las instancias función escala y función detalle debemos indicar el parámetro nivel que controla el número de puntos de la descomposición wavelet, que como ya hemos hablado anteriormente esto determinará el número de etapas de descomposición que se deben realizar. Cuanto mayor sea el nivel, más detalles se extraerán de la señal o la imagen.

```
    phi, psi, x = pywt.Wavelet('db2').wavefun(level=level)
```

El nivel j determina la escala 2^j entonces por cada unidad en el soporte la gráfica será aproximada con $2^j + 1$ puntos.

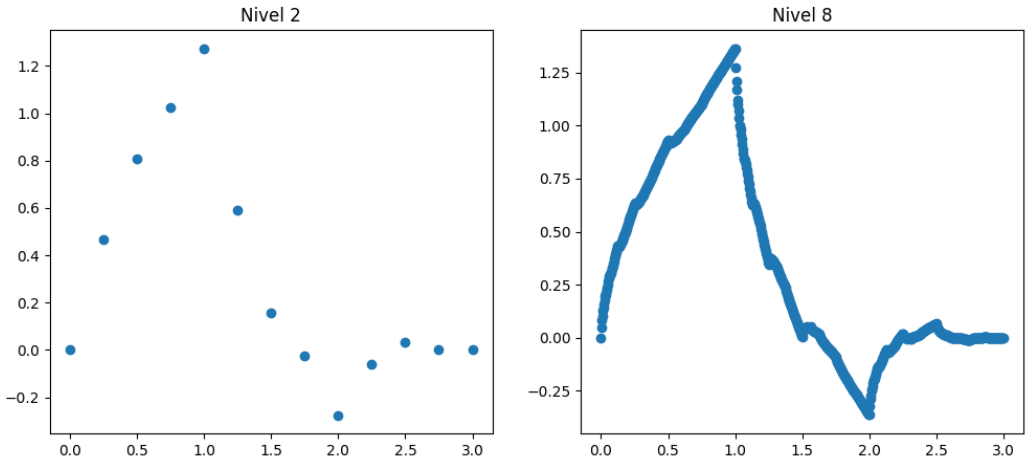


Figura 3.8: Gráficas de la función ϕ (ó función escala) en diferentes escala.

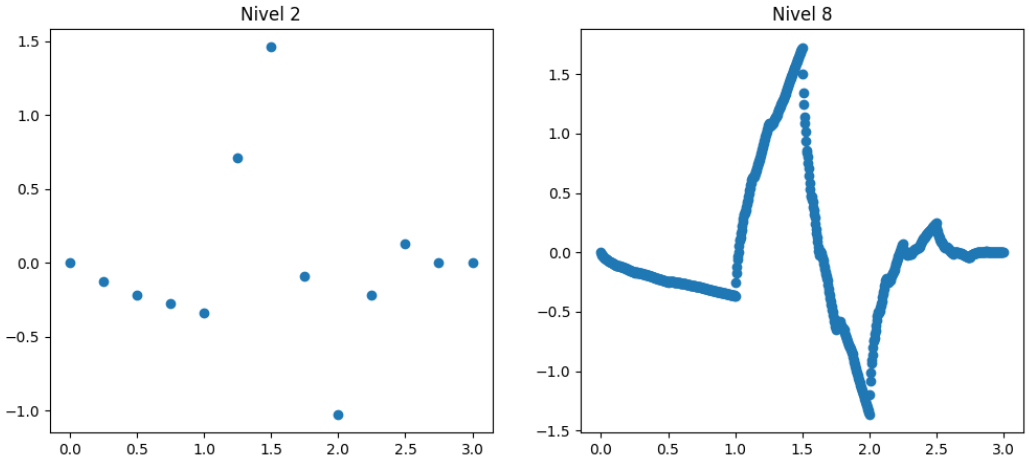


Figura 3.9: Gráficas de la función ψ (ó función wavelet) en diferentes escala.

La biblioteca **pywt** proporciona únicamente las funciones escala y wavelet en diversos niveles, pero no realiza el cálculo completo del paquete wavelets. Para hacer esta implementación, iniciamos creando una clase llamada *Function* para poder manipular de mejor manera las transformaciones[3] que se requieran hacer en el arreglo del dominio y del rango.

```
class Function:

    """
    Esta clase representa una función y tiene los siguientes atributos:
```

[3]Hago referencia a las transformaciones dadas por la definición recursiva 3.7 y 3.8

```
- x (lista): El dominio.
- y (lista): El rango.

Métodos disponibles:
- vcomp(self, factor): Realiza una compresión vertical.
- transform(self, k): Realiza dos transformaciones: la primera es una
  traslación horizontal k (hacia la derecha) y la segunda es una
  compresión horizontal por un factor de 2.
- atom_transform(self, j, k): Realiza dos transformaciones: la primera
  es una traslación horizontal k (hacia la derecha) y la segunda es una
  dilatación horizontal por un factor de 2**j.
- __add__(self, other_function): Calcula la suma entre dos funciones.
"""
```

A continuación, implementamos recursivamente las funciones w_n mediante la definición de la función *package(wavelet, n, level)*. El nombre de esta función se debe a que estas funciones constituyen las bases de lo que llamamos un paquete wavelet. Recordemos que los filtros de la wavelet db2 fueron calculados en la sección 3.1.19:

```
def package(wavelet, n, level):

"""
Calcula el n-ésimo paquete de la wavelet dada.

Args:
    wavelet: Nombre de una wavelet incluida en la biblioteca PyWavelets,
    las más usadas son: db, sym, coif, haar, bior, rbio, dmey, cdf y shan.
    n: parámetro de frecuencia.

Returns: la función n-ésimo paquete.
"""

phi, psi, x = pywt.Wavelet(wavelet).wavefun(level=level)
h = np.sqrt(2) * np.array(pywt.Wavelet(wavelet).dec_lo)[::-1]
g = np.sqrt(2) * np.array(pywt.Wavelet(wavelet).dec_hi)[::-1]
l = len(h)

if n == 0:
    return Function(x, phi)
if n == 1:
    return Function(x, psi)
if (n % 2) == 0 and (n != 0):
    w = package(wavelet, n/2, level).transform(0).vcomp(h[0])
    for k in range(1, l):
        w = w + package(wavelet, n/2, level).transform(k).vcomp(h[k])
```

```
        return w
    if (n % 2) == 1 and (n != 1):
        w = package(wavelet, (n-1) / 2, level).transform(0).vcomp(g[0])
        for k in range(1, l):
            w = w + package(wavelet, (n-1)/2, level).transform(k).vcomp(g[k])
        return w
```

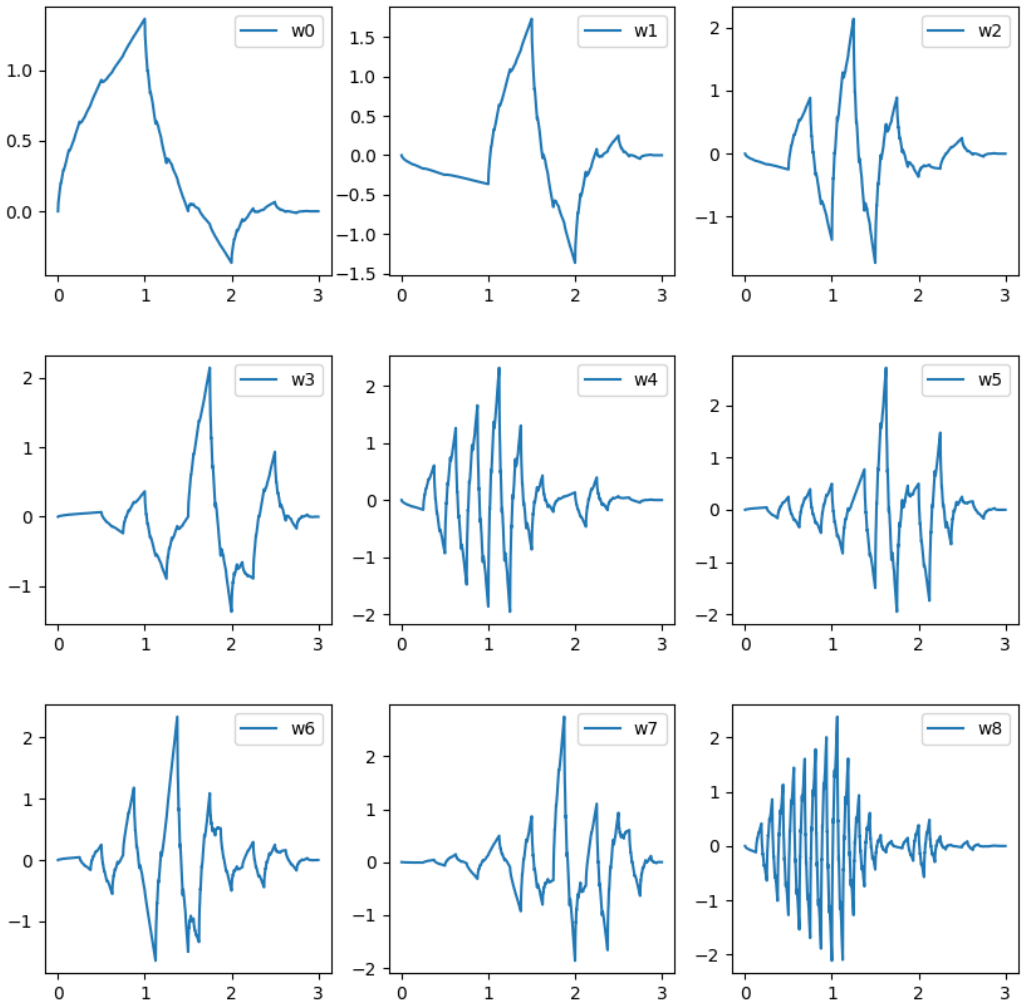


Figura 3.10: Representación gráfica de las primeras nueve funciones w_n de la wavelet db2.

Definición 3.1.23. *El conjunto de funciones $W_{j,n}$ se define como la colección de todas las*

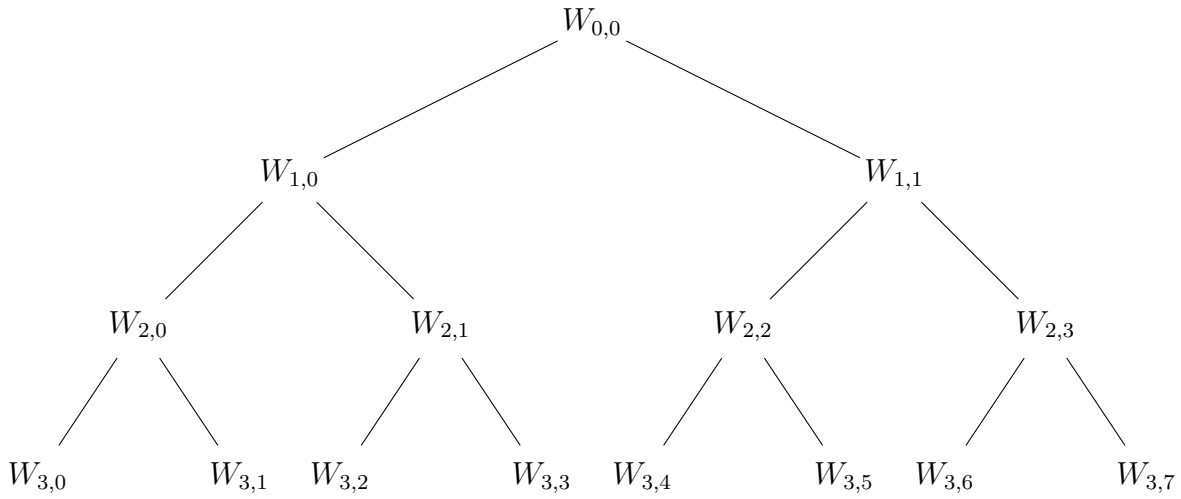
funciones $(w_n)_{j,k}(t)$*, definidas por:*

$$(w_n)_{j,k}(t) = 2^{-j/2} w_n(2^{-j}t - k)$$

donde k *es un número entero. Estas funciones componen* ***(j,n)-paquete wavelet****.*

El ***paquete wavelet****, por lo tanto, es la agrupación de todos los conjuntos* $W_{j,n}$*. Más específicamente, se define como* $\{W_{j,n};\ j \in \mathbf{Z}\ y\ n \in \mathbf{N}\}$*.*

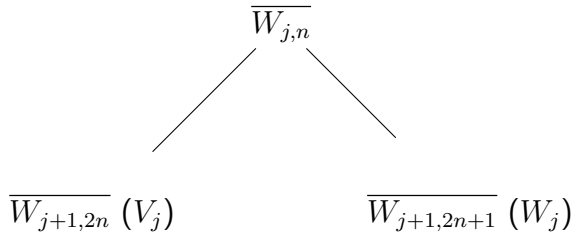
Cuando fijamos una escala, podemos organizar todos los paquetes en una estructura de árbol binario. Por ejemplo, para $j = 2$ los paquetes pueden ser organizados como:



Definición 3.1.24. *Dotamos a todos los paquetes con estructura de espacio generado cerrado* $\overline{W_{j,n}}$*, entonces para cada paquete* (j, n) *el espacio* $\overline{W_{j,n}}$ *se divide es dos subespacios:*

$$V_j := \overline{W_{j+1,2n}} \qquad \textit{llamado } \textbf{\textit{Espacio de Aproximación}}$$
$$W_j := \overline{W_{j+1,2n+1}} \qquad \textit{llamado } \textbf{\textit{Espacio de Detalle}}$$

El espacio $\overline{W_{0,0}}$ *que es el nodo del árbol será llamado* V_0*.*



Teorema 3.1.25. *Los espacios generados por los paquetes wavelets satisfacen las siguientes afirmaciones:*

1. *Para todo $n \in \mathbf{N}$, $W_{j,n}$ es una base ortogonal para $\overline{W_{j,n}}$.*
2. *La concatenación de las bases de V_j y W_j forma una base ortogonal para $\overline{W_{j,n}}$.*
3. *$\overline{W_{j,n}} = V_j \oplus W_j$, esto implica que la condición de ortogonalidad entre las funciones w_n se satisface por nivel*

Demostración. 1. Para demostrar que $W_{j,n}$ es una base ortogonal para $\overline{W_{j,n}}$, necesitamos probar dos cosas:

a) $W_{j,n}$ es un conjunto ortogonal.

b) $W_{j,n}$ genera $\overline{W_{j,n}}$.

Dado $W_{j,n} = \{(w_n)_{j,k} \mid k \in \mathbf{Z}\}$, debemos probar que para todo $n \in \mathbf{N}$:

$$\langle (w_n)_{j,k}, (w_n)_{j,k'} \rangle = 0 \quad \text{para } k \neq k'.$$

Esto se puede verificar haciendo inducción sobre n y desglosando la definición de las funciones $(w_n)_{j,k}$:

Para $n = 0$, $(w_0)_{j,k}(t) = 2^{-j/2} w_0(2^{-j}t - k) = 2^{-j/2}\phi(2^{-j}t - k)$, la condición de ortogonalidad se hereda de la ortogonalidad de los espacios de aproximación vistos en el WMRA.
Para $n = 1$, $(w_1)_{j,k}(t) = 2^{-j/2} w_1(2^{-j}t - k) = 2^{-j/2}\psi(2^{-j}t - k)$, la condición de ortogonalidad se hereda de la ortogonalidad de los espacios de detalle vistos en el WMRA. Ahora supongamos que la hipotesis del paso inductivo se satisface para las funciones w_n con $n < 2m$ (en el caso par), y $n < 2m + 1$ (en el caso impar) para algún $m \in \mathbf{N}$, y veamos que se satisface para $n = 2m$ y $n = 2m + 1$:

$$(w_{2m})_{j,k}(t) = 2^{-j/2} w_{2m}(2^{-j}t - k) = 2^{-j/2}\sqrt{2} \sum_{k=0}^{2N-1} h_{-k} w_m(2t - k)$$

$$(w_{2m+1})_{j,k}(t) = 2^{-j/2} w_{2m+1}(2^{-j}t - k) = 2^{-j/2}\sqrt{2} \sum_{k=0}^{2N-1} g_{-k} w_m(2^{-j}t - k)$$

Para ambos casos la ortogonalidad se hereda de la hipótesis inductiva.

Por definición de espacios cerrados generados claramente los elementos de $W_{j,n}$ generan $\overline{W_{j,n}}$.

2. Para demostrar que la concatenación de las bases de V_j y W_j forma una base ortogonal para $\overline{W_{j,n}}$, se debe de mostrar lo siguiente:

a) Veamos que los espacios V_j y W_j son ortogonales entre sí. Necesitamos demostrar que los conjuntos $W_{j+1,2n}$ y $W_{j+1,2n+1}$ son ortogonales:

$$\int_{-\infty}^{\infty} (w_{2n})_{j+1,k_1}(t) \cdot (w_{2n+1})_{j+1,k_2}(t)\, dt = 0$$

donde

$$(w_{2n})_{j+1,k_1}(t) = 2^{-(j+1)/2} w_{2n}(2^{-(j+1)}t - k_1)$$

y

$$(w_{2n+1})_{j+1,k_2}(t) = 2^{-(j+1)/2} w_{2n+1}(2^{-(j+1)}t - k_2).$$

Consideramos la integral del producto de estas funciones:

$$\int_{-\infty}^{\infty} (w_{2n})_{j+1,k_1}(t) \cdot (w_{2n+1})_{j+1,k_2}(t)\, dt$$

Sustituimos las definiciones de $(w_{2n})_{j+1,k_1}(t)$ y $(w_{2n+1})_{j+1,k_2}(t)$:

$$\int_{-\infty}^{\infty} \left(2^{-(j+1)/2} w_{2n}(2^{-(j+1)}t - k_1)\right) \left(2^{-(j+1)/2} w_{2n+1}(2^{-(j+1)}t - k_2)\right)\, dt$$

Simplificamos el factor constante:

$$2^{-(j+1)} \int_{-\infty}^{\infty} w_{2n}(2^{-(j+1)}t - k_1) \cdot w_{2n+1}(2^{-(j+1)}t - k_2)\, dt$$

Realizamos el cambio de variable $u = 2^{-(j+1)}t$, y la integral se convierte en:

$$\int_{-\infty}^{\infty} w_{2n}(u - k_1) \cdot w_{2n+1}(u - k_2)\, du$$

El cálculo puede continuar aplicando las definiciones de las funciones dadas en las ecuaciones 3.11 y 3.12, De este modo, la integral anterior se expresa en términos de las funciones w_n, que son ortogonales entre sí.

b) Para demostrar que cada función en $W_{j,n}$ puede ser obtenida como una combinación de funciones en $W_{j+1,2n}$ y $W_{j+1,2n+1}$, empecemos analizando la relación entre estos espacios. A partir de las definiciones 3.11 y 3.12, observamos que:

$$w_{2n}(2^{-j}t - k) = 2^{-(j+1)/2} w_n(2^{-(j+1)}t - 2k) + 2^{-(j+1)/2} w_n(2^{-(j+1)}t - 2k - 1)$$

$$w_{2n+1}(2^{-j}t - k) = 2^{-(j+1)/2} w_n(2^{-(j+1)}t - 2k) - 2^{-(j+1)/2} w_n(2^{-(j+1)}t - 2k - 1)$$

Podemos reescribir las funciones w_n en términos de w_{2n} y w_{2n+1} para obtener:

$$w_n(2^{-(j+1)}t - 2k) = \frac{1}{2}\left[w_{2n}(2^{-(j+1)}t - 2k) + w_{2n+1}(2^{-(j+1)}t - 2k)\right]$$

$$w_n(2^{-(j+1)}t - 2k - 1) = \frac{1}{2}\left[w_{2n}(2^{-(j+1)}t - 2k - 1) - w_{2n+1}(2^{-(j+1)}t - 2k - 1)\right].$$

Por tanto, si $w_n(2^{-j}t - k)$ pertenece a $W_{j,n}$, podemos escribir:

$$w_n(2^{-j}t-k)=\frac{1}{2}\left[w_{2n}(2^{-j}t-2k)+w_{2n+1}(2^{-j}t-2k)\right]$$

Así, sumando y restando las funciones de $W_{j+1,2n}$ y $W_{j+1,2n+1}$:

$$w_{2n}(2^{-j}t-2k)=2^{(j+1)/2}w_{j+1,2n}(t)$$

$$w_{2n+1}(2^{-j}t-2k)=2^{(j+1)/2}w_{j+1,2n+1}(t).$$

De esta manera, podemos expresar cualquier función en $W_{j,n}$ como una combinación de funciones en $W_{j+1,2n}$ y $W_{j+1,2n+1}$.

Similarmente podemos demostrar la otra contención.

c) Para demostrar que $\overline{W_{j,n}}=V_j\oplus W_j$, necesitamos probar que cualquier función en $\overline{W_{j,n}}$ puede descomponerse de manera única en una función de V_j y una función de W_j.
Lo pedido se sigue los dos literales anteriores, dado que V_j y W_j son ortogonales entre sí y sus bases concatenadas forman una base para $\overline{W_{j,n}}$.

□

3.1.2. Diccionarios

En esta subsección, exploraremos la creación de diccionarios, y posteriormente implementaremos uno para su uso en el algoritmo de Matching Pursuit.

Definición 3.1.26. *En el contexto del procesamiento de señales, un* ***diccionario*** $\mathcal{D}$ *es un conjunto de elementos* $\{d_i\}_{i\in I}$ *en un espacio de Hilbert* $\mathcal{H}$*. Estos elementos se denominan* ***átomos*** *y se utilizan para representar o descomponer una señal* $f\in\mathcal{H}$*. Es decir, dado un diccionario*

$$\mathcal{D}=\{d_i\in\mathcal{H}:i\in I\}$$

donde I *es un conjunto de índices.*
La representación de una señal $f\in\mathcal{H}$ *se expresa como una combinación lineal de los átomos:*

$$f\approx\sum_{i\in I}c_id_i$$

donde $c_i\in\mathbf{R}$ *son los coeficientes de la combinación.*

Los diccionarios pueden ser completos, sobrecompletos o subcompletos en $\mathcal{H}$:

- **Completo:** Si el diccionario forma una base ortonormal de $\mathcal{H}$.
- **Sobrecompleto o redundante** Si el diccionario contiene más funciones de las necesarias para formar una base de $\mathcal{H}$, es decir, $|I|>\dim(\mathcal{H})$ para espacios de dimensión finita.

- **Subcompleto:** Si el diccionario no contiene suficientes funciones para formar una base de $\mathcal{H}$.

Observación 4. Los **diccionarios redundantes** permiten obtener una representación flexible y detallada, esto quiere decir que se puede tener múltiples opciones de representación de cualquier parte de la señal.

Recordemos que, en la práctica, consideraremos una señal discreta $x(t) \in \ell^2(\mathbf{R})$, la cual representa una serie de precios del mercado bursátil con 64 puntos, que son los precios de cierre ajustados por día. Esto implica que la señal $x(t)$ es cero en la mayoría de sus entradas. Entonces localmente, podemos pensar que la señal está en un subespacio de dimensión finita N y para este espacio podemos considerar un diccionario redundante $\mathcal{D} = \{d_1(t), d_2(t), \cdots, d_L(t)\}$ con $L > N$. Estos diccionarios no garantizan unicidad de la representación, entonces para la señal $x(t)$ pueden existir varios conjuntos de índices $I(x)$, tal que:

$$x(t) \approx \sum_{n \in I(x)} c_i d_i(t)$$

donde c_i son escalares.

Una aproximación con estas características se denomina **no-lineal** y es importante para representar componentes de señales que cambian tanto en el tiempo como en la frecuencia, abarcando una amplia variedad de ubicaciones temporales y rangos de frecuencia. Esta aproximación no lineal la encontraremos con el algoritmo Matching Pursuit 3.2.

En el articulo [22] se presentan técnicas y metodologías para construir diccionarios. Los átomos en un diccionario pueden ser construidos mediante un enfoque matemático o a través de técnicas de aprendizaje automático. En el primer caso, se emplean técnicas como wavelets, paquetes waveletes, contourlets [13] y curvelets [7]. En el segundo caso, se ajustan a conjuntos de ejemplos específicos mediante métodos como MOD y K-SVD, los cuales optimizan la representación basándose en los datos de entrenamiento. Ahora, nos enfocaremos en la construcción de un diccionario utilizando las funciones $\{w_n\}_{n \in \mathbf{N}}$ definidas en los paquetes wavelets.

Definición 3.1.27. *Definimos los átomos como el siguiente conjunto de funciones:*

$$(w_n)_{j,k}(t) := 2^{-j/2} w_n(2^{-j}t - k), \quad (j,k) \in \mathbf{Z} \times \mathbf{Z} \quad y \quad 0 \leq n \leq 2^j - 1.$$

donde j, k y n son los parámetros de escala, posición y frecuencia respectivamente.

Para un valor fijo j, el parámetro de frecuencia n analiza las fluctuaciones de la señal alrededor de la posición $k \cdot 2^j$, es decir, se está analizando cómo las diferentes frecuencias n contribuyen a la señal en una escala particular.

Implementación de las átomos

En esta implementación, se carga el archivo que contiene el paquete db2, el cual fue previamente guardado con un nivel de aproximación de 8, y se realizan las transformaciones correspondientes.

```
def w_jnk(wavelet, j, n, k, level):

    """
    Calcula los átomos del paquete de la wavelet dada.

    Args:
        wavelet: El nombre de la una wavelet incluida en la biblioteca
        PyWavelets.
        j: parámetro de escala.
        n: parámetro de frecuencia.
        k: parámetro de posición.
        level: Nivel de la wavelet.

    Returns: los átomos w_jnk.

    """
    file_path = f'package{wavelet}{level}'
    if os.path.exists(file_path):
        # print(f'The file "{file_path}" exists in the system.')
        with open(file_path, 'rb') as file:
            pack = pickle.load(file)
            y = n+64
            fun = Function(pack[:, n], pack[:, y])
            return fun.atom_transform(j, k).vcomp(2**(-j/2))
```

Implementación del diccionario

Las series de precio con las que estamos trabajando tienen 64 puntos, que son los precios de cierre ajustados por día. Por lo tanto, en la construcción de nuestros diccionarios, hemos optado por una escala de $j = 5$, o que implica que $n \leq 2^5 - 1$. Por consideraciones de eficiencia de memoria, optamos por no almacenar las translaciones de los átomos durante la construcción del diccionario. No obstante, estas translaciones se tendrán en cuenta como parte del diccionario redundante al invocarlo en el algoritmo Matching Pursuit.

Por otro lado, recordemos que las transformaciones que se hacen a la clase función, son tanto para el rango y el dominio, por lo que construiremos dos diccionarios correspondientes. A continuación solo muestro el código de los valores del rango, pero se hace de forma similar para el diccionario del dominio.

```
    def const_dict_y(wavelet, level):

    """
    Esta función construye una matriz que contiene por columna los valores
    de rango de cada átomo.
```

```
"""

j = 6
columns = []
for j in range(j):
    for n in range(64):
        column = w_jnk(wavelet, j, n, 0, level).y
        columns.append(column)
        matrix = np.column_stack(columns)
return matrix
```

3.2. Algoritmo Matching Pursuit en el Análisis de Wavelets

Matching Pursuit es un algoritmo utilizado en el procesamiento de señales y en la representación dispersa de datos. Como su nombre lo indica, dado un diccionario redundante, el algoritmo aproxima una señal como una combinación lineal de los átomos que mejor "coinciden" con ella. Fue introducido en [18].

En general, el algoritmo es planteado para un espacio de Hilbert $\mathcal{H}$, en el caso particular de la implementación es $\ell^2(\mathbf{R})$. Supongamos que tenemos una señal x y un diccionario redundante $\mathcal{D} = \{a_\gamma\}_{\gamma\in\Gamma}$, donde cada a_γ es un átomo con norma unitaria, es decir, $\|a_\gamma\| = 1$. En este espacio, la norma utilizada es la norma euclidiana estándar para sucesiones cuadráticamente sumables de números reales.

Observemos que cuando seleccionamos un átomo a_{γ_0} podemos hacer una proyección ortogonal de la señal sobre el átomo para obtener una descomposición:

$$x = \langle x, a_{\gamma_0}\rangle a_{\gamma_0} + Rx$$

donde Rx es el residuo.

Ahora, gracias a la ortogonalidad de a_{γ_0} y Rx se sigue que:

$$\|x\|^2 = |\langle x, a_{\gamma_0}\rangle|^2 + \|Rx\|^2$$

Por tanto, para encontrar una mejor aproximación de x debemos minimizar $\|Rx\|$ que es equivalente a maximizar $|\langle x, a_{\gamma_0}\rangle|$. En términos computacionales, esto se traduce en seleccionar el átomo que maximiza el producto interno. Luego, los siguientes pasos del algoritmo es iterar la elección de los átomos sobre cada una de las actualizaciones de los residuos y se detendrá al alcanzar una tolerancia predefinida en la norma del residuo o al completar un número especifico de iteraciones.

Teorema 3.2.1. *El algoritmo converge, es decir* $R^n \underset{n\to\infty}{\longrightarrow} 0$, *donde* R^n *es la actualización del residuo en la n-ésima iteración.*

Demostración. Consideremos una señal x y un diccionario $\mathcal{D}$ en un espacio de Hilbert $\mathcal{H}$. Replicando el algoritmo, tenemos lo siguiente:
Inicialmente, definimos $R^0 = x$. En cada iteración n, seleccionamos el átomo a_{γ_n} que maximiza la correlación con el residuo actual:

$$a_{\gamma_n} = \arg\max_{a_\gamma \in \mathcal{D}} |\langle R^n x, a_\gamma \rangle|, \text{ denotamos el máximo } \alpha_n := \langle R^n x, a_{\gamma_n} \rangle.$$

Actualizamos el residuo $R^{n+1}x = R^n x - \alpha_n a_{\gamma_n}$.

La norma cuadrada del residuo se reduce en cada iteración, es decir $\|R^{n+1}x\|^2 \leq \|R^n x\|^2$ esto se debe gracias a las siguientes operaciones:

$$\begin{aligned} \|R^{n+1}x\|^2 &= \|R^n x - \alpha_n a_{\gamma_n}\|^2 \\ &= \|R^n x\|^2 - 2\langle R^n x, \alpha_n a_{\gamma_n} \rangle + |\alpha_n|^2 \\ &= \|R^n x\|^2 - 2|\alpha_n|^2 + |\alpha_n|^2 \\ &= \|R^n x\|^2 - |\alpha_n|^2. \end{aligned}$$

Todo lo anterior demuestra que la sucesión $\{\|R^n x\|^2\}_{n \in \mathbf{N}}$ es monótona decreciente y está acotada inferiormente por cero, lo que implica que $\|R^n x\|^2 \underset{n\to\infty}{\longrightarrow} 0$ y así $R^n x \underset{n\to\infty}{\longrightarrow} 0$.

□

Implementación del Matching Pursuit

Recordemos que el diccionario que creamos anteriormente está guardando la información de la wavelet db2 en una escala de λ_8 en su soporte, esto quiero decir que el diccionario guarda $2^8 + 1$ puntos en un intervalo de $[0, 3]$. Sin embargo, la señal que queremos aproximar tiene 64 puntos, por lo que hace falta hacer una interpolación adecuada de cada átomo para considerarlo en un intervalo de $[0, 64]$, con un total de 16129 puntos.

Para hacer la interpolación se uso la función `np.interp` de la biblioteca NumPy en Python que realiza una interpolación lineal.

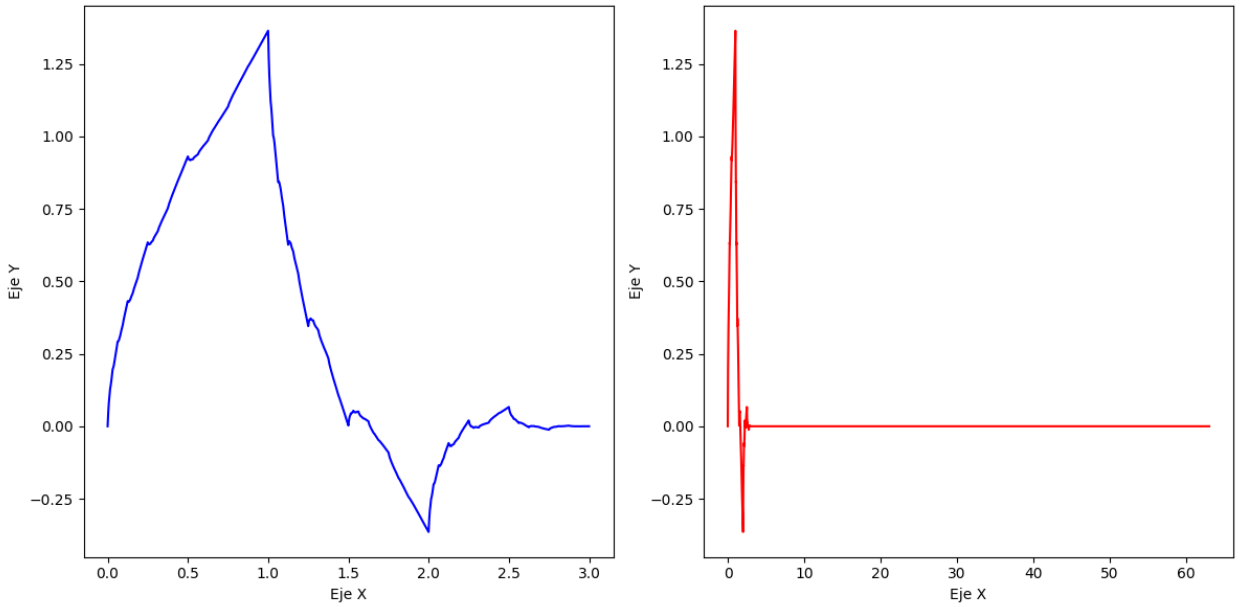


Figura 3.11: Ejemplo de la representación gráfica de wavelet db2 con $n = 0$, a la izquierda utilizando 769 puntos y al lado derecho interpolada con 16129 puntos.

```
def mp(dicc_y, dicc_x_reduced, y_signal, iteration):

"""
Encuentra la "mejor coincidencia" de las proyecciones de los datos sobre
el espacio generado por un diccionario sobrecompleto D.

Args:
    dicc_y: Diccionario que representa los valores y.
    dicc_x: Diccionario que representa los valores x.
    y_signal: La señal que deseas aproximar.
    iteration: Número de iteraciones.

Returns: Una lista con la siguiente información:
Residuo, aproximación y coeficientes después de todas las iteraciones.
"""

delta = int((dicc_x_reduced.shape[0] - 1) / 3)
total_points = int((63 * delta) + 1)
r = y_signal
approx = np.zeros(total_points)
coefficients = np.zeros(25792)

for _ in range(iteration):
    p = {}
    key = 0
```

```
        print(_)

        for a in range(320):
            j = a // 64
            for k in range(-3 * (2 ** j)+1, 63):
                atom_comp = interpolacion(dicc_y, dicc_x_reduced, j, a, k)
                new_value = {'product': (atom_comp @ r) * (1 / delta),
                             'list': [j, a, k]}
                p[key] = new_value
                key += 1

        index_c_max = max(p, key=lambda k: p[k]['product'])
        c = p[index_c_max]['product']
        coefficients[index_c_max] = c

        ### Calculando el residuo
        list_max = p[index_c_max]['list']  # lista: j, a, k
        k_max = list_max[2]
        j_max = list_max[0]
        a = list_max[1]

        a_max = interpolacion_generalizada(dicc_y, dicc_x_reduced, j_max, a,
                                           k_max)

        ### RESIDUO
        r = r - c * a_max

        ### APROXIMACIÓN
        approx = approx + (c * a_max)

    return r, approx, coefficients
```

Capítulo 4

Redes Neuronales

Las redes neuronales son modelos de aprendizaje inspirados en el cerebro humano y otros mamíferos. Están compuestas por capas de neuronas que se conectan entre sí a través de unidades funcionales y estructurales llamadas sinapsis. Estas sinapsis, representadas por pesos sinápticos, determinan la fuerza de las conexiones entre neuronas. Durante el aprendizaje, estas redes ajustan los pesos sinápticos para minimizar una función de costo predefinida, permitiendo así la predicción precisa de resultados basados en datos de entrenamiento. Existen varios tipos de redes neuronales comunes: las redes feedforward, las redes convolucionales, las redes recurrentes, las redes LSTM y las redes generativas. Para más información sobre estas redes, puede consultar [29].

4.1. Perceptrón Multicapa (MLP por sus siglas en inglés)

Un **perceptrón multicapa** es un tipo de red neuronal que pertenece al tipo de **redes feedforward**, también conocida como red neuronal profunda. Estas redes se caracterizan principalmente por su flujo de información unidireccional, donde la información fluye desde la capa de entrada hacia la capa de salida sin la existencia de ciclos o bucles de retroalimentación. Además, se distinguen por su estructura, que incluye al menos una capa oculta intermedia además de las capas de entrada y salida. Son ampliamente utilizadas en aplicaciones de reconocimiento de patrones y clasificación.

En un MLP, las capas ocultas están compuestas por neuronas completamente conectadas a las neuronas de la capa anterior y de la capa siguiente, pero no entre sí dentro de la misma capa oculta, esta configuración se conoce como **"fully connected"** o **"densamente conectada"**.

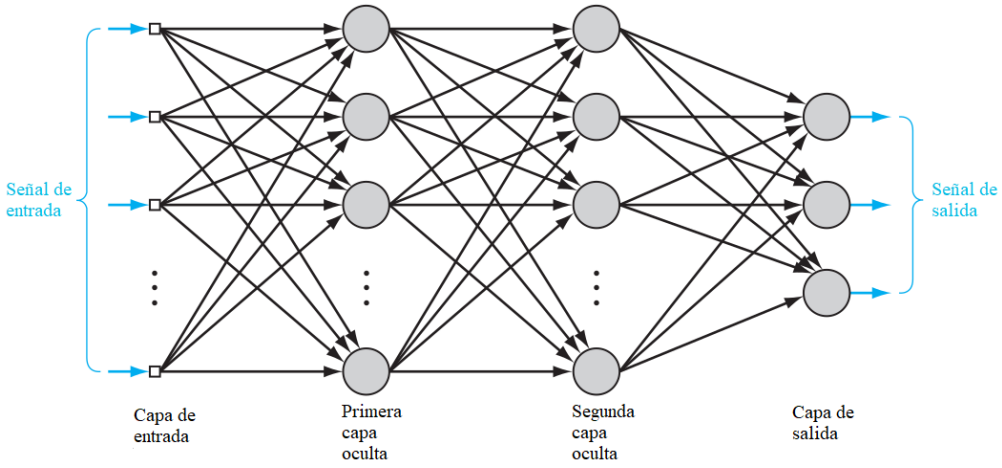
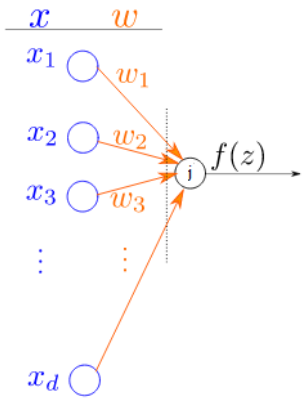


Figura 4.1: Gráfica de la arquitectura de un perceptron multicapa con dos capas ocultas. Adaptada de [14].

Los neuronas en un MLP realizan dos operaciones principales: el cálculo de una suma ponderada de sus entradas y la aplicación de la llamada **función de activación** a esta suma ponderada. La suma ponderada se calcula multiplicando cada entrada por un peso asociado y luego sumando estos productos junto con un término adicional llamado sesgo. Si consideramos una neurona j con entradas x_i, pesos w_{ij} y un sesgo b_j, la salida de la suma ponderada z_j se calcula como:



$$z_j = \sum_{i=1}^{d} w_{ij} x_i + b_j$$

Figura 4.2: Adaptada de [14]

Posteriormente, se aplica una función de activación f a z_j para obtener la salida de la neurona:

$$y_j = f(z_j)$$

Funciones de activación.

Existe una variedad de funciones de activación, estas son las que determinan si una neurona debe activarse, es decir, si debe transmitir una señal hacia adelante. Principalmente, introducen no linealidad en el modelo. Algunas de las funciones de activación más comunes incluyen la función de unidad lineal rectificada (ReLU), la sigmoide y la tangente hiperbólica.

- **Función de unidad lineal rectificada (ReLU).** Esta función se define por la expresión:

$$\text{ReLU}(x) = \max(x, 0)$$

Formalmente, ReLU no es diferenciable en $x = 0$. Entonces, por convención tomamos la derivada del lado izquierdo que es 0. Por tanto, la derivada de la función ReLU es:

$$\frac{d}{dx}\text{ReLU}(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ 1 & \text{si } x > 0 \end{cases}$$

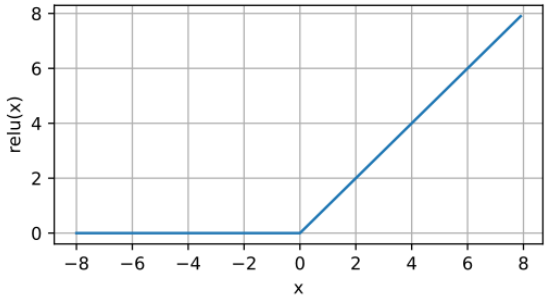


Figura 4.3: Función ReLU. Tomada de [29].

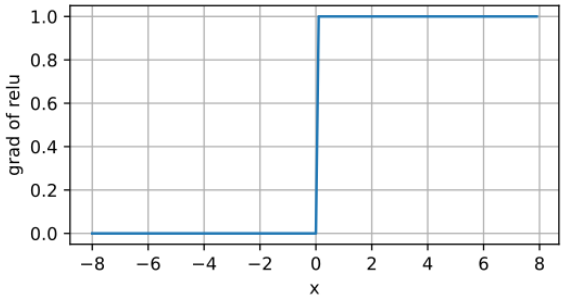


Figura 4.4: Derivada ReLU. Tomada de [29].

- **Función sigmoide.** Esta función se define por la expresión:

$$\text{sigmoide}(x) = \frac{1}{1 + \exp(-x)} \tag{4.1}$$

Observemos que la función sigmoide es diferenciable en todos los $\mathbb{R}$ y su rango se encuentra en el intervalo (0, 1). Esto la hace muy útil para problemas de clasificación binaria, ya que permite interpretar las salidas como probabilidades. Aunque en su mayoría ha sido reemplazada por ReLU en las capas ocultas debido a la simplicidad y eficiencia de esta última, la función sigmoide sigue siendo relevante en ciertos contextos.

La derivada de la función sigmoide esta dada por la ecuación:

$$\frac{d}{dx}\text{sigmoide}(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} = \text{sigmoide}(x)(1 - \text{sigmoide}(x)) \tag{4.2}$$

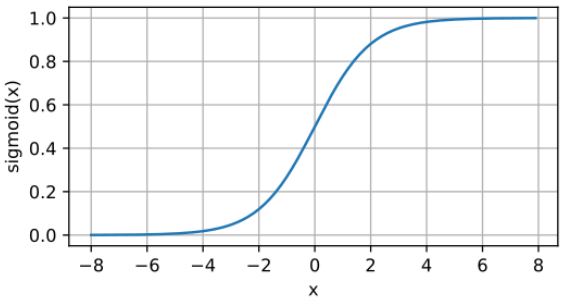


Figura 4.5: Función sigmoide. Tomada de [29].

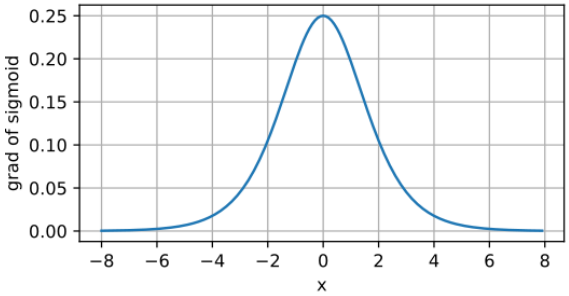


Figura 4.6: Derivada sigmoide. Tomada de [29].

- **Función tangente hiperbólica.** Esta función se define por la expresión:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

Observemos que al igual que la función sigmoide, la función tangente hiperbólica es diferenciable en todos los $\mathbb{R}$, pero su rango se encuentra en el intervalo (-1, 1). La derivada de esta función esta dada por la fórmula:

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

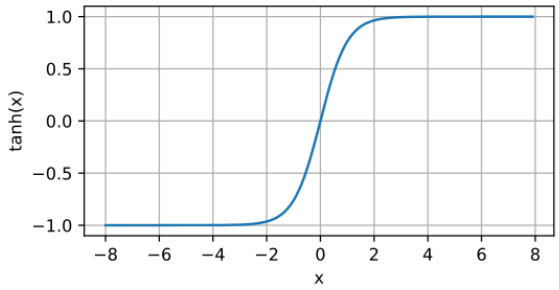


Figura 4.7: Función tangente hiperbólica. Tomada de [29].

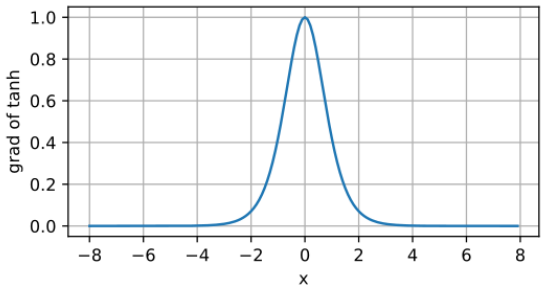


Figura 4.8: Derivada tangente hiperbólica. Tomada de [29].

Función de pérdida.

La función de pérdida, también conocida como función de costo o función objetivo, es una función no negativa $\mathcal{L}(\cdot,\cdot) : \mathbf{R} \times \mathbf{R} \longmapsto [0, \infty)$, que cuantifica la desviación o el error entre valor real (etiqueta) y y el valor predicho $\hat{y}$.

Existen diversas funciones de costo que se adaptan a las características del modelo de predicción. Entre las más comunes se encuentran el error cuadrático medio, el error absoluto medio, la entropía cruzada y la pérdida hinge. Para obtener más información sobre estas funciones, puede consultar [29]. Sin embargo, dado que nuestra red neuronal aborda un problema de

clasificación, utilizamos la entropía cruzada como función de pérdida.

La función de pérdida **entropía cruzada** es ampliamente utilizada en problemas de clasificación donde se considera una distribución de posibles resultados. Se define como:

$$\mathcal{L}(\hat{y}, y) = -\sum_k y^k \log(\hat{y}^k)$$

donde y^k es la etiqueta para la clase k y $\hat{y}^k$ es la probabilidad predicha para la clase k dado un vector de características $\mathbf{x}$.

En los problemas de clasificación donde se utiliza la entropía cruzada como función de pérdida, es común aplicar una función llamada softmax en la última capa de la red neuronal antes de calcular la pérdida.

La **función softmax** transforma las salidas de una red neuronal en probabilidades, garantizando que la suma de todas las salidas sea igual a uno y que cada probabilidad esté en el rango de 0 a 1. Esto es crucial para interpretar las salidas como probabilidades de pertenencia a cada clase en problemas de clasificación multiclase. Formalmente, la función softmax se define como:

Dado un vector de entrada $\mathbf{o} = (o_1, o_2, \ldots, o_k)$, la salida de la función softmax $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k)$ se calcula como:

$$\hat{y}_j = \frac{\exp(o_j)}{\sum_{k=1}^{K} \exp(o_k)}$$

para $j = 1, 2, \ldots, K$, donde K es el número de clases.

Durante el entrenamiento de una red neuronal, el objetivo principal es minimizar la función de pérdida. Esto se logra ajustando gradualmente los pesos y sesgos en la dirección opuesta al gradiente de la función de pérdida. El gradiente no solo indica la dirección en la cual los pesos deben ser ajustados para reducir la pérdida, sino también la magnitud del cambio más rápido en la función de pérdida. Los detalles de cómo se calculan estos gradientes mediante el algoritmo de backpropagation se explicarán más adelante.

Para un Perceptrón Multicapa (MLP), es esencial comprender el proceso detallado de entrenamiento, que implica varias etapas clave, que se realizan utilizando datos de entrenamiento para ajustar efectivamente los parámetros del MLP.

Etapas del Entrenamiento de un MLP

1. **Inicialización de pesos y sesgos.** Los pesos y sesgos de las neuronas, como se define en 4.1, se inicializan de manera aleatoria.
2. Los datos se organizan en mini-lotes para actualizar los pesos y minimizar la función de pérdida. Estos son subconjuntos pequeños y aleatorios del conjunto de datos de

entrenamiento, se usan con el fin de mejorar el proceso de entrenamiento y la gestión de recursos computacionales. La necesidad de usar mini-lotes depende del tamaño del conjunto de datos.

a) **Propagación hacia adelante (forward propagation).** Los datos de entrada se pasan a través de la red, capa por capa.

b) **Cálculo de la pérdida (loss calculation).** Ahora, para evaluar el rendimiento del modelo, hacemos uso de la llamada función de costo empírica que se utiliza para cuantificar la pérdida en todo el conjunto de datos de entrenamiento. Para ser mas precisos, esta función se calcula:

Dado un conjunto de datos de entrenamiento $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ y un modelo parametrizado por θ (pesos y los bias), la función de costo empírica $J(\theta)$ se expresa como:

$$J(\theta) = \sum_{n=1}^{N} \mathcal{L}(\hat{y}_n, y_n)$$

Donde $\mathcal{L}$ es la función de pérdida entropía cruzada entre el valor predicho $\hat{y_n}$ dado un vector $\mathbf{x}_n$ y su etiqueta real y_n.

c) **Propagación hacia atrás (Backpropagation)** [24]. Es un algoritmo que calcula de manera eficiente los gradientes de la función de pérdida con respecto a cada peso en la red neuronal. A continuación, explicaremos detalladamente cómo funciona.
La función de pérdida es una composición de muchas funciones de activación con los parámetros de peso y bias que entran en diferentes etapas, entonces vamos a definir primero la notación de todos los elementos involucrados en estas derivadas.

Sea $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ el conjunto de datos de entrenamiento, donde $\mathbf{x}_n$ corresponde a un vector de características con etiqueta y_n. Supongamos que nuestra red neuronal consiste de L capas, donde cada capa r-ésima contiene k_r neuronas.

Sea θ_j^r es el vector que contiene la información de los pesos y el bais de j-ésima neurona de la r-ésima capa, tiene la forma:

$$\theta_{\mathbf{j}}^{\mathbf{r}} = [\theta_{j0}^r, \theta_{j1}^r, \ldots, \theta_{jk_{r-1}}^r]^T$$

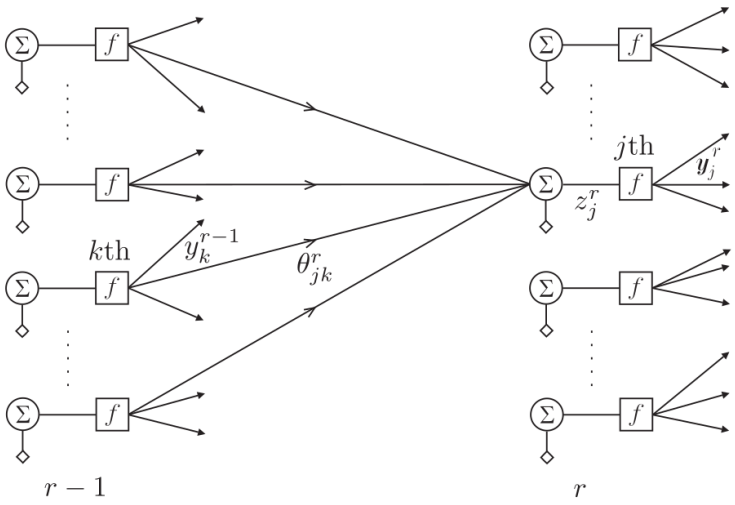


Figura 4.9: Representación gráfica de la red neuronal entre la capa $r-1$ y r. Tomada de [24].

Además recordemos que también en 4.1 habíamos denotado por z_{nj}^r la salida de la combinación lineal de la j-ésima neurona de la r-ésima capa del n-ésimo dato. Entonces:

$$z_{nj}^r = \sum_{m=0}^{k_{r-1}} \theta_{jm}^r y_{nm}^{r-1} = \theta_{\mathbf{j}}^{\mathbf{r}T} {\mathbf{y_n}}^{r-1}$$

donde ${\mathbf{y_n}}^{r-1} = [1, y_{n1}^{r-1}, \ldots, y_{nk_{r-1}}^{r-1}]^T$

La pérdida que deseamos optimizar esta dada por:

$$J(\theta) = \sum_{n=1}^{N} \mathcal{L}(\hat{y}_n, y_n) = \sum_{n=1}^{N} \mathcal{L}_n$$

derivando, tenemos que:

$$\frac{\partial \mathcal{L}_n}{\partial \theta_{\mathbf{j}}^{\mathbf{r}}} = \frac{\partial \mathcal{L}_n}{\partial z_{nj}^r} \frac{\partial z_{nj}^r}{\partial \theta_{\mathbf{j}}^{\mathbf{r}}} = \frac{\partial \mathcal{L}_n}{\partial z_{nj}^r} {\mathbf{y_n}}^{r-1}$$

Entonces, definimos los gradientes

$$\boxed{\delta_{nj}^r := \frac{\partial \mathcal{L}_n}{\partial z_{nj}^r}}$$

Por tanto, el algoritmo de backpropagation se encarga de calcular los gradientes δ_{nj}^r, en un proceso empieza en la capa de salida y se mueve hacia atrás, aplicando la regla de la cadena para calcular las derivadas parciales.
Para $r = L$, el gradiente en la capa de salida está dado por:

$$\delta_{nj}^L = (\hat{y}_{nj} - y_{nj}) f'(z_{nj}^L)$$

Para los gradientes de las capas ocultas:

$$\delta_{nj}^{r-1} = \left(\sum_{k=1}^{k_r} \delta_{nk}^r \theta_{kj}^r \right) f'\left(z_{nj}^{r-1}\right), \quad j = 1, 2, \ldots, k_{r-1}.$$

d) **Actualización de los Pesos y Sesgos** [24]. En este paso hacemos uso de algún método de optimización para actualizar los pesos y los bias. En nuestro caso usamos el método Descenso de Gradiente Estocástico (conocido por sus siglas en inglés SGD):

$$\theta_j^r(\text{new}) = \theta_j^r(\text{old}) + \Delta\theta_j^r, \tag{4.3}$$

$$\Delta\theta_j^r = -\eta \left. \frac{\partial J}{\partial \theta_j^r} \right|_{\theta_j^r(\text{old})} = -\eta \sum_{n=1}^{N} \delta_{nj}^r {\mathbf{y}_\mathbf{n}}^{r-1}, \quad r = 1, 2, \ldots, L. \tag{4.4}$$

Aquí, η es el parámetro denominado tasa de aprendizaje.

3. **Repetición del proceso.** Se repiten los pasos del literal 2 para cada mini-lote de datos en el conjunto de entrenamiento. Una época se define como una pasada completa de todo el conjunto de datos de entrenamiento a través del modelo. Este ciclo se repite durante un número de épocas (el cual es un parámetro en la implementación).

En cuanto a la implementación de la tesis, la aplicación del algoritmo Matching Pursuit genera una gran cantidad de coeficientes y produce señales dispersas, lo que hace necesario una técnica de reducción de dimensión. Por eso, antes de abordar el tema central de este capítulo, dedicaremos una sección al análisis de componentes principales. Esto nos simplificará considerablemente el trabajo en términos computacionales, ya que reducirá la complejidad, optimizará el almacenamiento y disminuirá el tiempo de procesamiento.

4.2. Análisis de Componentes Principales vía SVD

El análisis de componentes principales (ACP o PCA, por sus siglas en inglés) es una técnica de reducción de dimensionalidad utilizada en estadística y aprendizaje automático. Su objetivo es transformar un conjunto de variables posiblemente correlacionadas en un conjunto de variables linealmente no correlacionadas, conocidas como **componentes principales**. Estas componentes permiten proyectar los datos en un subespacio de menor dimensión de manera que la varianza de los datos se maximice en ese subespacio. Para una explicación detallada de las propiedades estadísticas del PCA, se recomienda consultar [24].

A lo largo de esta sección, consideraremos la matriz aleatoria $\mathbf{X}$ cuyas filas son variables aleatorias con media cero. En caso de que alguna fila no tenga media cero, se puede restar la media de cada fila a cada elemento de dicha fila para ajustarlas. El siguiente teorema formaliza la existencia un conjunto ortogonal de vectores que, al proyectar las filas de $\mathbf{X}$ sobre el subespacio generado por este conjunto, maximiza la varianza de los datos. Esto nos da un indicio de donde debemos de buscar las componentes principales.

Teorema 4.2.1. *Sea* $\{x_1, ..., x_N\} \in \mathbf{R}^l$ *un conjunto de vectores de observación del vector aleatorio* $\mathbf{x}$*, entonces existe un conjunto ortogonal de vectores* $\{u_1, ..., u_k\}$*, con* $k < N$ *de manera tal que proyectar los vectores de observación sobre el subespacio generado por este conjunto maximiza la varianza de los datos.*

Demostración. Vamos a seleccionar cada uno de los u_i's en orden. Para determinar u_1, debemos de calcular la varianza de las proyecciones de los vectores $\{x_1, x_2, \ldots, x_N\}$ sobre u_1, la cual está dada por:

$$\text{Var}(\text{Proj}_{\mathbf{u}_1}(\mathbf{x})) = \frac{1}{N}\sum_{n=1}^{N}\frac{(u_1^T x_n)^2}{\parallel u_1 \parallel^2}.$$

Suponiendo que $||u_1|| = 1$, la ecuación de la varianza se simplifica a:

$$\text{Var}(\text{Proj}_{\mathbf{u}_1}(\mathbf{x})) = \frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n)(x_n^T u_1) = u_1^T \hat{\Sigma} u_1$$

donde, $\hat{\Sigma} := \dfrac{1}{N}\sum_{n=1}^{N} x_n x_n^T$ es la matriz de covarianza muestral.

Entonces, planteamos el siguiente problema de optimización con el objetivo de encontrar el u_1 que maximice la varianza:

$$\begin{aligned} & u_1 = \arg\max_{u} \; u^T \hat{\Sigma} u \\ \text{sujeto a} \quad & u^T u = 1 \end{aligned}$$

El Lagrangiano correspondiente a este problema se puede formular de la siguiente manera:

$$\mathcal{L}(u, \lambda) = u^T \hat{\Sigma} u - \lambda(u^T u - 1)$$

Donde u es el vector de variables que queremos optimizar y λ es el multiplicador de Lagrange asociado a la restricción.

Para encontrar los valores óptimos de u y λ, se derivan las ecuaciones de primer orden:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial u} &= 2\hat{\Sigma} u - 2\lambda u \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= u^T u - 1 \end{aligned}$$

Entonces igualando a cero obtenemos el sistema de ecuaciones para encontrar los valores óptimos u y λ:

$$\begin{cases} \hat{\Sigma} u = \lambda u \\ u^T u = 1 \end{cases}$$

Por tanto $u^T \hat{\Sigma} u = \lambda$, esto quiere decir que la varianza se maximiza si elegimos u_1 como el vector propio correspondiente al valor propio máximo λ_1.

Recordemos que la matriz de covarianza muestral ($\hat{\Sigma}$) es simétrica y semidefinida positiva, lo que implica que sus valores propios son reales y no negativos. Suponiendo que $\hat{\Sigma}$ es invertible (lo cual implica necesariamente que $N > k$), los valores propios son todos positivos, es decir, podemos ordenarlos de forma descendente $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq 0$. También asumimos que son distintos para simplificar la discusión.

La segunda componente principal se selecciona para cumplir con las siguientes condiciones: sea ortogonal a u_1 y maximice la varianza después de proyectar los datos en esta dirección y sea ortogonal a u_1. Esto se puede formular como el siguiente problema de optimización:

$$\begin{aligned} & u_2 = \arg\max_u u^T \hat{\Sigma} u \\ \text{sujeto a} \quad & u^T u = 1 \\ & u^T u_1 = 0 \end{aligned}$$

Para resolver este problema de optimización, utilizamos nuevamente el método de los multiplicadores de Lagrange. El Lagrangiano se define como:

$$\mathcal{L}(u, \lambda, \mu) = u^T \hat{\Sigma} u - \lambda(u^T u - 1) - \mu(u^T u_1)$$

Las derivadas parciales del Lagrangiano son:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial u} &= 2\hat{\Sigma} u - 2\lambda u - \mu u_1 \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= u^T u - 1 \\ \frac{\partial \mathcal{L}}{\partial \mu} &= u^T u_1 \end{aligned}$$

Entonces igualando a cero obtenemos el sistema de ecuaciones para encontrar los valores óptimos u, λ y μ:

$$\begin{aligned} \hat{\Sigma} u &= \lambda u - \frac{\mu}{2} u_1 \\ u^T u &= 1 \\ u^T u_1 &= 0 \end{aligned}$$

Por tanto, queremos encontrar u_2 de manera que $u_2^T \hat{\Sigma} u_2 = \lambda$ sea máxima. Ya hemos seleccionado u_1 como el vector propio correspondiente al mayor valor propio λ_1, entonces seleccionamos u_2 como el vector propio correspondiente al segundo mayor valor propio λ_2.

El proceso continúa hasta que se hayan obtenido m ejes principales; estos son los vectores propios correspondientes a los m mayores valores propios. □

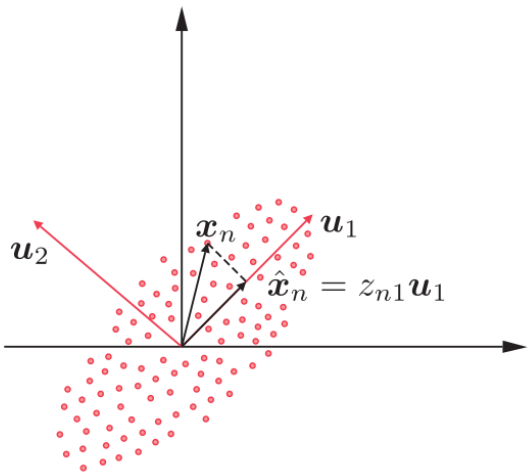


Figura 4.10: Representación grafica de a proyección en las componentes principales. Tomada de [24]

Vamos a calcular las componentes principales usando álgebra lineal, particularmente la descomposición en valores singulares (DVS o SVD, por sus siglas en inglés) que es un tipo de factorización.

Definición 4.2.2. *Si los λ_i son los valores propios de la matriz X^TX (o XX^T), entonces las raíces cuadradas, $\sigma_i := \sqrt{\lambda_i}$, se denominan* ***valores singulares*** *de X.*

Teorema 4.2.3. *Si $X \in \mathbf{R}^{m\times n}$, entonces existen matrices ortogonales:*

$$U = [u_1, ..., u_m] \in \mathbf{R}^{m\times m} \text{ y } V = [v_1, ..., v_n] \in \mathbf{R}^{n\times n}$$

talque $XV = UD$, donde las matriz $D \in \mathbf{R}^{m\times n}$ está formada por los valores singulares de X en su diagonal principal ordenados de mayor a menor, y ceros en el resto de entradas.

Demostración. Sea X una matriz real $m \times n$ de rango r, entonces la matriz $X^TX \in \mathbf{R}^{n\times n}$ es cuadrada, simétrica y semidefinida positiva. Entonces definimos $\{v_1, ..., v_r\}$ como el conjunto de vectores propios ortonormales de la matriz X^TX.

Ahora, definimos los vectores $u_i := \dfrac{Xv_i}{\sigma_i}$, donde σ_i's son los valores singulares de la matriz X ordenados de forma descendente. Quiero mostrar que el conjunto de $\{u_1, ..., u_r\}$ forman un conjunto ortonormal:

$$\langle u_i, u_j\rangle = u_i^Tu_j = \left(\frac{Xv_i}{\sigma_i}\right)^T\left(\frac{Xv_j}{\sigma_j}\right) = \frac{v_i^TX^TXv_j}{\sigma_i\sigma_j} = \frac{v_i^T\sigma_j^2v_j}{\sigma_i\sigma_j} = \frac{\sigma_j^2}{\sigma_i\sigma_j}v_i^Tv_j = \frac{\sigma_j^2}{\sigma_i\sigma_j}\langle v_i, v_j\rangle$$

Las igualdades anteriores demuestran que los vectores u_i's son ortogonales dos a dos.

Solo nos resta calcular su norma:

$$\|u_i\|^2 = \langle u_i, u_i\rangle = \frac{\sigma_i^2}{\sigma_i\sigma_i}\langle v_i, v_i\rangle = \|v_i\|^2 = 1$$

Finalizamos completando los conjuntos $\{u_1, ..., u_r\}$ y $\{v_1, ..., v_r\}$ para obtener bases ortonormales en las dimensiones correspondientes. □

En la práctica aplicamos esta factorización en valores singulares (SVD) para reducir la dimensionalidad de las características en nuestro conjunto de datos $X \in \mathbb{R}^{m\times n}$ de rango r. Esta reducción tiene sentido para valores k, donde $k < r$.

$$X = UDV^T$$

$$X = \underbrace{\begin{bmatrix} | & | & & | & & | \\ u_1 & u_2 & \cdots & u_k & \cdots & u_r \\ | & | & & | & & | \end{bmatrix}}_{m\times r} \times \underbrace{\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \cdots & 0 \\ 0 & 0 & \dots & \sigma_k & \cdots & 0 \\ \vdots & \vdots & & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & \sigma_r \end{bmatrix}}_{r\times r} \times \underbrace{\begin{bmatrix} - & v_1^T & - \\ - & v_2^T & - \\ & \vdots & \\ - & v_k^T & - \\ & \vdots & \\ - & v_r^T & - \end{bmatrix}}_{r\times n}$$

Para obtener una aproximación $\hat{X}$ de X, realizamos el producto matricial $U_k D_k V_k^T$, donde U_k se forma a partir de las primeras k columnas de la matriz U, D_k está compuesta por las primeras k filas y columnas de la matriz D, y V_k^T consiste en las primeras k filas de la matriz V^T.

La aproximación $\hat{X}$ se escribe de esta forma:

$$\hat{X} = \begin{bmatrix} u_1, ..., u_k \end{bmatrix} \begin{bmatrix} \sigma_1 v_1^T \\ \vdots \\ \sigma_k v_k^T \end{bmatrix} = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Entonces cada vector $\hat{x}_n$ de la columna $\hat{X}$ puede ser escrito como:

$$\hat{x}_n = \sum_{i=1}^{k} z_{ni} u_i \quad \text{donde} \quad z_{ni} := u_i^T x_n$$

Definición 4.2.4. *Sea $X \in \mathbf{R}^{m\times n}$ una matriz de rango r, donde las columnas de la matriz representan las variables observadas de una matriz aleatoria $\mathbf{X}$. Podemos reducir la dimensión a $k \leq r$ usando la descomposición SVD. Como acabamos de estudiar, cada columna x se puede escribir de la siguiente forma:*

$$x = \sum_{i=1}^{k} z_i u_i.$$

*Donde $z_i := u_i^T x$, definimos las componentes aleatorias $\mathbf{z}_i$ como las **componentes principales** de la matriz $\mathbf{X}$.*

Teorema 4.2.5. *Las componentes principales son variables no correlacionadas.*

Demostración. Para demostrar que las variables z_i no están correlacionadas, necesitamos demostrar que la covarianza entre z_i y z_j es cero para $i \neq j$.

La covarianza entre z_i y z_j se define como:

$$\text{Cov}(z_i, z_j) = \mathbb{E}[z_i z_j]$$

Sustituyendo $z_i = u_i^T x$ y $z_j = u_j^T x$:

$$\text{Cov}(z_i, z_j) = \mathbb{E}[(u_i^T x)(u_j^T x)] = \mathbb{E}[u_i^T x x^T u_j]$$

Dado que $\Sigma = XX^T$ es la matriz de covarianza de X:

$$\text{Cov}(z_i, z_j) = u_i^T \Sigma u_j$$

En el contexto del análisis de componentes principales, Σ se diagonaliza en la base de U, de manera que:

$$\Sigma = U \Lambda U^T$$

donde Λ es una matriz diagonal que contiene los valores propios. Debido a que U es ortogonal, tenemos $U^T U = I$, por lo que:

$$\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbb{E}[U^T \mathbf{x}\mathbf{x}^T U] = U^T \Sigma U = \Lambda$$

Por lo tanto, para $i \neq j$:

$$\text{Cov}(z_i, z_j) = 0$$

□

Implementación del SVD

```
def redSVD(array, dim):

"""
Reduce la dimensión del arreglo de datos dado.

Args:
    array: arreglo de datos.
    dim: dimensión deseada (número de columnas de los datos).

Returns:
    X_reduced: (mismo número de filas del "array", "dim")
    tsvd: guarda las componentes principales ajustadas de los datos dados.

"""
```

```
    assert dim <= array.shape[1]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(array)

    tsvd = TruncatedSVD(n_components=dim)    # Crea la instancia TruncatedSVD
    X_reduced = tsvd.fit_transform(X_scaled)  # Es U @ Sigma

    return X_reduced, tsvd
```

4.3. Ensamble de Redes Neuronales

Un ensamble de redes neuronales es una técnica de aprendizaje automático que combina múltiples modelos de redes neuronales para mejorar la precisión y la robustez de las predicciones. Entre las técnicas más populares para construir estos ensambles se encuentran bagging y boosting [30].

- **Bagging** Dado un conjunto de datos se genera múltiples conjuntos, mediante muestreo con reemplazo, lo que significa que cada dato puede ser seleccionado más de una vez en la creación de cada subconjunto. Cada red neuronal se entrena independientemente y de manera paralela utilizando uno de estos conjuntos. Finalmente, las predicciones de todas las redes se promedian para obtener el resultado final.
- **Boosting** En este método, las redes neuronales se entrenan secuencialmente. Cada nueva red se centra en los ejemplos que las redes anteriores encontraron difíciles de clasificar. En cada iteración, se asignan pesos más altos a las instancias mal clasificadas o mal predichas por los modelos anteriores, permitiendo que los modelos subsiguientes se centren más en corregir esos errores específicos. Al final, las predicciones se combinan utilizando un esquema ponderado según el rendimiento de cada red.

En el artículo [9], que sirve como base para nuestro estudio, se utiliza la técnica de ensamble bagging. Sin embargo, dado que el objetivo de nuestro ensamble es predecir señales de compra o venta en el mercado bursátil, en este capítulo desarrollaremos un ensamble adaptado específicamente al contexto financiero. Este ensamble estará compuesto por cinco perceptrones multicapa, diseñados para captar de manera efectiva las tendencias del mercado. Por lo tanto, comenzaremos estudiando el concepto de perceptrón multicapa.

4.3.1. Ensamble

En el análisis técnico o en el desarrollo de estrategias de trading aparece el término de sobre-ajuste de datos, conocido en inglés como "data snooping bias", puede ocurrir cuando un analista tiene demasiados parámetros libres en su modelo que son ajustado a patrones de mercado en el pasado y encuentra una estrategia que parece funcionar excepcionalmente bien. Pero es poco probable que un modelo ajustado a estos patrones tenga un gran porcentaje de predicción.

El libro [8] propone el uso de modelos lineales para reducir el sesgo de espionaje de datos y presenta un modelo lineal que utiliza factores predictivos normalizados para estimar los retornos de un índice bursátil. Dado un conjunto de factores predictivos $\{f(i)\}_{i=1}^{n}$ (por ejemplo n indicadores técnicos). Calculamos para cada uno de estos su normalización:

$$z(i) = (f(i) - \text{mean}(f))/\text{std}(f) \tag{4.5}$$

Para luego definir modelo retorno predictivo de la siguiente forma:

$$R = \text{mean}(R) + \text{std}(R) \sum_{i=1}^{n} \frac{\text{sign}(i) \cdot z(i)}{n} \tag{4.6}$$

donde las cantidades mean(f) y std(f) representan el promedio histórico y la desviación estándar de los distintos $f(i)$, sign(i) es el signo de la correlación histórica entre $f(i)$ y R, mientras que mean(R) y std(R) son el promedio histórico y la desviación estándar de los rendimientos de un día, respectivamente.

Las ecuaciones 4.5 y 4.6 son válidas cuando trabajamos con valores continuos. Sin embargo, en nuestra implementación, el conjunto de factores predictivos $\{f(i)\}_{i=1}^{n}$ son redes neuronales diseñadas para clasificación binaria. Entonces, debemos hacer dos adaptaciones:

- La normalización según la ecuación 4.5 no es aplicable en este contexto. En consecuencia, redefiniremos la ecuación 4.6 de la siguiente manera:

$$R = \text{mean}(R) + \text{std}(R) \sum_{i=1}^{n} \frac{\text{sign}(i) \cdot f(i)}{n} \tag{4.7}$$

- En lugar de medir la correlación tradicional, que asume variables continuas y mide la relación lineal entre ellas, evaluaremos el impacto financiero de cada factor predictivo. Esto implica calcular cuánto dinero nos ayuda a ganar cada uno de estos factores en nuestras decisiones de trading.

  Dado un factor predictivo $f(j)$, en un período de n días consecutivos donde r_i denota el rendimiento del día i y $p_j(i)$ es la predicción del factor para el día i $(1 \leq i \leq n)$, el rendimiento compuesto R_j durante este período, al seguir las predicciones de $p_j(i)$, se calcula como:

$$R_j = \sqrt[n]{\prod_{i=1}^{n}(1 + r_i \cdot p_j(i))} - 1$$

  Idealmente, buscamos alcanzar un retorno anual del 20 %, considerando que en los mercados financieros abren su mercado 252 días en promedio en un año, se requiere obtener aproximadamente un rendimiento diario del 0.0724 %.

  Por lo tanto, en la ecuación 4.6, redefiniremos la constante $c(i)$ de la que obtenemos el sign(i) de la siguiente manera:

$$c(i) = \begin{cases} 1, & \text{si } R_j \geq 0{,}0007 \\ -1, & \text{si } R_j < -0{,}0007 \\ 0, & \text{en otro caso} \end{cases} \tag{4.8}$$

Esto establece sign(i) según el rendimiento R_j dentro de una ventana de tiempo, donde R_j es el rendimiento diario.

Capítulo 5

Implementación

El entrenamiento de un ensamble que incluye cinco modelos de redes neuronales del tipo Perceptrón Multicapa (MLP) es un proceso complejo que abarca múltiples etapas. Este proceso de aprendizaje automático incluye la selección de algoritmos adecuados, la extracción y selección de características, la evaluación de modelos y la optimización de hiperparámetros, entre otros aspectos. A continuación, se describen los pasos generales para llevar a cabo este entrenamiento.

Entrenamiento General de un MLP

1. **Obtención de datos.** Recopilar un conjunto de datos representativo del problema.
2. **Preparación de datos.** Limpiar, normalizar y transformar los datos para su uso en el modelo.
3. **División de datos.** Dividir el conjunto de datos en los conjuntos: entrenamiento, validación y prueba.
4. **Entrenamiento de cada MLP.** Realizar los pasos que se describen en 4.1 haciendo uso de los datos del conjunto de entrenamiento.
5. **Validación del modelo.** Evaluar el modelo en el conjunto de validación.
6. **Ajuste de hiperparámetros.** Ajustar hiperparámetros como la tasa de aprendizaje, el número de neuronas, el número de capas, etc.
7. **Evaluación del modelo en el ensamble.** Evaluar el modelo en el conjunto de prueba para obtener una estimación imparcial de su rendimiento en datos no vistos.

5.1. Obtención y preparación de datos

Inicialmente, la implementación descargará los datos necesarios para el entrenamiento y la prueba de la red. Se recopilaron aleatoriamente un total de 3,120 series de precios para

el entrenamiento de la red, distribuidas equitativamente con 520 datos de cada uno de los activos AAPL, AMZN, META, MSFT, TSLA y GOOGL. Para la evaluación del modelo, se descargaron 300 datos consecutivos del activo AAPL.

Los datos obtenidos servirán para formar un conjunto de características concatenadas, que constituirán los datos de entrada de nuestro ensamble de redes neuronales, de la siguiente manera:

- Coeficientes del Matching Pursuit obtenidos tras una reducción de la dimensión de las series de precio mediante el uso del SVD, estos le servirán a la red para que identifique y aprenda patrones particulares presentes en los gráficos de líneas.
- Los indicadores técnicos calculados con las series de precios dadas, estos se enumeran en la tabla 5.1. Estos servirán a la red para el aprendizaje del análisis técnico.

Para resumir, se presenta la tabla 5.1 que muestra la información que obtendrá cada vector de características, recuerde que la notación usada aquí se refiere a los indicadores técnicos descritos en el capitulo 2.1.

5.2. Resultados y Conclusiones

Para el entrenamiento de los modelos de MLP, se utilizan datos de entrenamiento y de validación. Generalmente, los datos de validación representan el 10 % de los datos de entrenamiento y son útiles para ajustar los hiperparámetros de la red. Utilizaremos las gráficas de pérdida para detectar problemas de sobreajuste o subajuste y realizar los cambios necesarios en los hiperparámetros.

Las **gráficas de pérdida** representan la evolución del error del modelo a lo largo del tiempo. En una situación ideal, el modelo muestra una buena capacidad de generalización cuando la curva de pérdida de entrenamiento disminuye constantemente y se estabiliza en un valor bajo, mientras que la curva de pérdida de validación disminuye ligeramente por encima de la curva de pérdida de entrenamiento, pero se mantiene cerca y se estabiliza en un valor bajo similar, sin aumentar posteriormente.

Existe un **sobreajuste** cuando la curva de validación está muy por encima de la curva de entrenamiento hacia el final del entrenamiento, y un **subajuste** cuando ambas curvas (entrenamiento y validación) permanecen altas y cercanas entre sí.

Las gráficas obtenidas en nuestro entrenamiento se muestran en 5.1. La interpretación de estas gráficas indica que los modelos están aprendiendo adecuadamente de los datos de entrenamiento, como se observa en la disminución continua de la pérdida de entrenamiento. La pérdida de los datos de validación también disminuye en todos los casos, lo que sugiere una buena generalización. Sin embargo, los últimos tres modelos muestran una separación entre las curvas de entrenamiento y validación, lo que podría ser un indicio de sobreajuste.

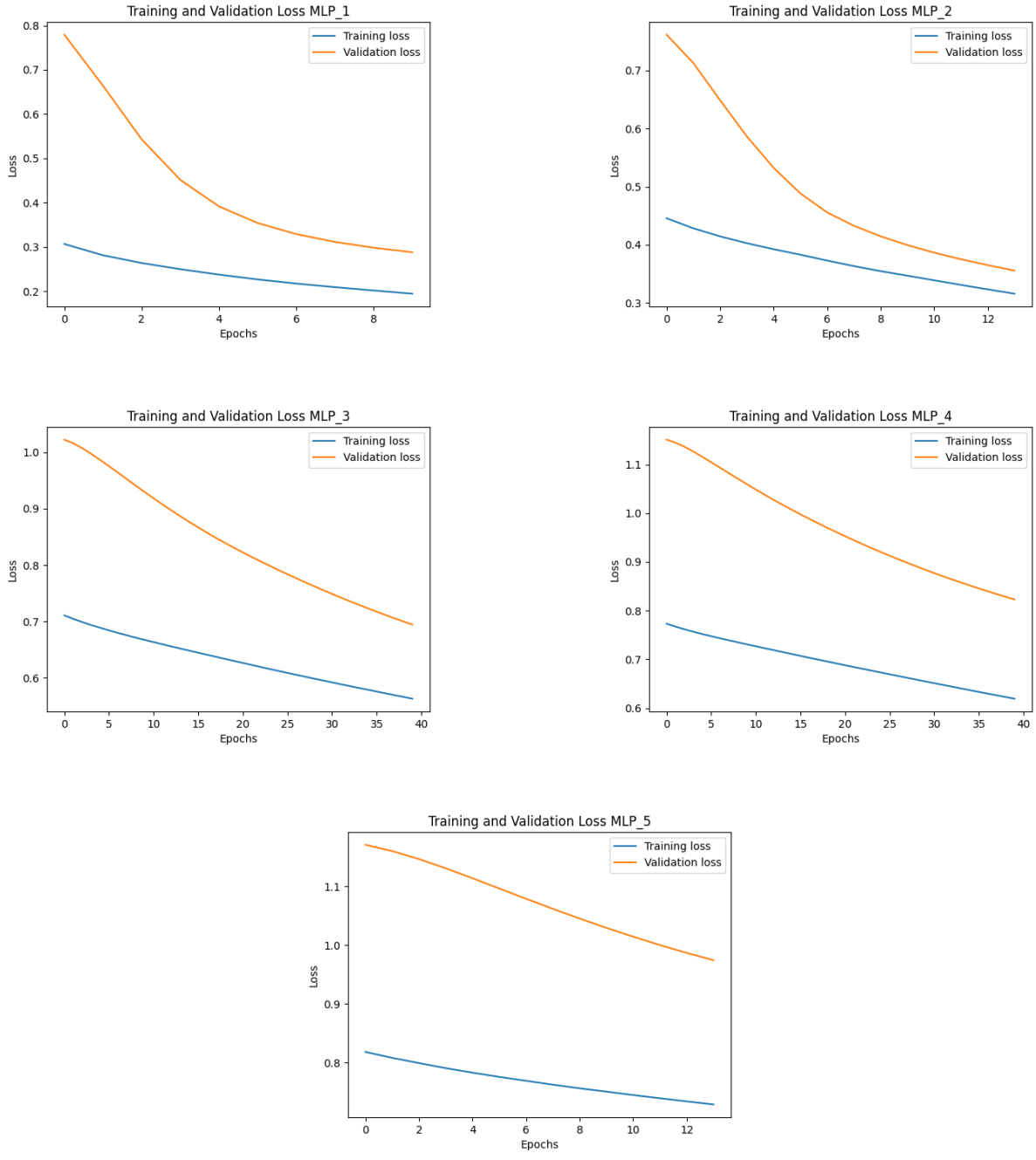


Figura 5.1: Gráficas de la pérdida de cada MLP en función de las épocas.

Para evaluar el resultado de nuestra implementación, se realizó una evaluación del ensamble utilizado en este estudio empleando 281 datos de prueba, obteniendo una precisión del 77 %, equivalente a 195 datos correctamente predichos. Este nivel de precisión es satisfactorio, aunque sería necesario disponer de más datos para realizar un análisis mas adecuado de el sobreajuste. No obstante, este trabajo profundiza en la teoría del modelo descrito en el artículo [9] y abre la puerta a futuros estudios que podrían conducir a una mayor precisión, mayor robustez y una reducción del sobreajuste.

Para trabajos futuros, se pueden considerar las siguientes modificaciones y/o mejoras: Primero, se podría recopilar y utilizar un mayor volumen de datos, particularmente optimizar el código del algoritmo Matching Pursuit, el cual tarda aproximadamente cinco minutos en procesar una sola serie de precios. Además, se pueden explorar aproximaciones diferentes utilizando otras wavelets, ya que en nuestro trabajo solo consideramos la familia de wavelets db2. La optimización de hiperparámetros también es una vía prometedora para mejorar el rendimiento del modelo. Comparar las cifras obtenidas con otros modelos de referencia permitiría evaluar el desempeño relativo y destacar las fortalezas y debilidades de nuestra implementación. Adicionalmente, se sugiere realizar un análisis detallado de las gráficas de velas para capturar patrones más complejos y utilizar otros ensambles de modelos, lo cual podría mejorar la precisión de las predicciones. Estas mejoras y modificaciones abrirán nuevas oportunidades para profundizar y ampliar el alcance de los hallazgos presentados en este trabajo.

Vector de características
Coeficientes del Matching Pursuit obtenidos tras una reducción de la dimensión mediante el uso del SVD
MAs de 5, 10, 25, y 40 días
EMA's de 5, 10, 25, y 40 días
MACD
OBV
RSI's de 4, 9 y 14 días
(precio actual - precio día previo)/precio día previo
(precio actual - MA de 5 días)/MA de 5 días
(precio actual - MA de 10 días)/ MA de 10 días
(precio actual - MA de 15 días)/ MA de 15 días
(precio actual - MA de 20 días)/ MA de 20 días
(precio actual - MA de 25 días)/ MA de 25 días
(precio actual - MA de 30 días)/ MA de 30 días
(precio actual - MA de 35 días)/ MA de 35 días
(precio actual - MA de 40 días)/ MA de 40 días
Considere BB10: $\begin{cases} 0, & \text{si banda inferior} \leq \text{precio actual} \leq \text{banda superior} \\ (\text{precio actual} - \text{banda superior}), & \text{banda superior} \leq \textbf{precio actual} \\ (\text{precio actual} - \text{banda inferior}, & \text{si precio actual} \leq \text{banda inferior} \end{cases}$
Considere BB20: $\begin{cases} 0, & \text{si banda inferior} \leq \text{precio actual} \leq \text{banda superior} \\ (\text{precio actual} - \text{banda superior}), & \text{banda superior} \leq \textbf{precio actual} \\ (\text{precio actual} - \text{banda inferior}, & \text{si precio actual} \leq \text{banda inferior} \end{cases}$
Considere BB30: $\begin{cases} 0, & \text{si banda inferior} \leq \text{precio actual} \leq \text{banda superior} \\ (\text{precio actual} - \text{banda superior}), & \text{banda superior} \leq \textbf{precio actual} \\ (\text{precio actual} - \text{banda inferior}, & \text{si precio actual} \leq \text{banda inferior} \end{cases}$
(RSI15-50)/50 (RSI10-50)/50 (RSI15-50)/50 (RSI20-50)/50
(% K de los últimos 5 días -50)/50
(% K de los últimos 10 días -50)/50
(% K de los últimos 15 días -50)/50
(% K de los últimos 20 días -50)/50
(% K - % D de los últimos 5 días -50)/50
(% K - % D de los últimos 10 días -50)/50
(% K - % D de los últimos 15 días -50)/50
(% K - % D de los últimos 20 días -50)/50
Precio de cierre del día (t-1)
Precio de cierre del día (t-2)
Precio de cierre del día (t-3)
Precio de cierre del día (t-4)
Precio de cierre del día (t-5)

Bibliografía

[1] Ajith Abraham, Baikunth Nath, and Prabhat Kumar Mahanti. Hybrid intelligent systems for stock market analysis. In *Computational Science-ICCS 2001: International Conference San Francisco, CA, USA, May 28—30, 2001 Proceedings, Part II 1*, pages 337–345. Springer, 2001.

[2] Khaled A Althelaya, Salahadin A Mohammed, and El-Sayed M El-Alfy. Combining deep learning and multiresolution analysis for stock market forecasting. *IEEE Access*, 9:13099–13111, 2021.

[3] Sabrine Arfaoui, Anouar Ben Mabrouk, and Carlo Cattani. *Wavelet analysis: basic concepts and applications*. Chapman and hall/CRC, 2021.

[4] Morgan A Belcher, InChan Hwang, Sylvia Bhattacharya, W David Hairston, and Jason S Metcalfe. Eeg-based prediction of driving events from passenger cognitive state using morlet wavelet and evoked responses. *Transportation Engineering*, 8:100107, 2022.

[5] Albert Boggess and Francis J Narcowich. *A first course in wavelets with Fourier analysis*. John Wiley & Sons, 2015.

[6] Thomas N Bulkowski. *Encyclopedia of chart patterns*. John Wiley & Sons, 2021.

[7] Emmanuel Jean Candes, David Leigh Donoho, et al. *Curvelets: A surprisingly effective nonadaptive representation for objects with edges*. Department of Statistics, Stanford University Stanford, CA, USA, 1999.

[8] Ernie Chan. *Algorithmic trading: winning strategies and their rationale*. John Wiley & Sons, 2013.

[9] Chalothon Chootong and Ohm Sornil. Trading signal generation using a combination of chart patterns and indicators. *International Journal of Computer Science Issues (IJCSI)*, 9(6):202, 2012.

[10] Michael X Cohen. A better way to define and describe morlet wavelets for time-frequency analysis. *NeuroImage*, 199:81–86, 2019.

[11] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

[12] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[13] Minh N Do and Martin Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on image processing*, 14(12):2091–2106, 2005.

[14] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[15] Shyh-Jier Huang and Cheng-Tao Hsieh. Coiflet wavelet transform applied to inspect power system disturbance-generated signals. *IEEE Transactions on Aerospace and Electronic Systems*, 38(1):204–210, 2002.

[16] Monica Lam. Neural network techniques for financial performance prediction: integrating fundamental and technical analysis. *Decision support systems*, 37(4):567–581, 2004.

[17] Jing Lin and Liangsheng Qu. Feature extraction based on morlet wavelet and its application for mechanical fault diagnosis. *Journal of sound and vibration*, 234(1):135–148, 2000.

[18] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[19] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[20] Steve Nison. *Japanese candlestick charting techniques: a contemporary guide to the ancient investment techniques of the Far East*. Penguin, 2001.

[21] Donald B Percival and Andrew T Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge university press, 2000.

[22] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[23] Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

[24] Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[25] Robert R Trippi. *Artificial intelligence in finance and investing: state-of-the-art technologies for securities selection and portfolio management*. McGraw-Hill, Inc., 1995.

[26] Marc Velay and Fabrice Daniel. Stock chart pattern recognition with deep learning. *arXiv preprint arXiv:1808.00418*, 2018.

[27] Xiashuang Wang, Guanghong Gong, and Ni Li. Automated recognition of epileptic eeg states using a combination of symlet wavelet processing, gradient boosting machine, and grid search optimizer. *Sensors*, 19(2):219, 2019.

[28] Qinghua Wen, Zehong Yang, Yixu Song, and Peifa Jia. Automatic stock decision support system based on box theory and svm algorithm. *Expert systems with Applications*, 37(2):1015–1022, 2010.

[29] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[30] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.