Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Análisis de dimensión fractal para series de tiempo de expresión génica mediante una red neuronal artificial.

Tesis presentada al

Posgrado en Física Aplicada

como requisito parcial para la obtención del grado de

Maestro en Ciencias

por

Marco Antonio Esperón Pintos

asesorado por

Dr. Jorge Velázquez Castro
Dr. Benito de Celis Alonso

Puebla Pue.
5 de Enero de 2023

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Análisis de dimensión fractal para series de tiempo de expresión génica mediante una red neuronal artificial.

Tesis presentada al

Posgrado en Física Aplicada

como requisito parcial para la obtención del grado de

Maestro en Ciencias

por

Marco Antonio Esperón Pintos

asesorado por

Dr. Jorge Velázquez Castro
Dr. Benito de Celis Alonso

Puebla Pue.
5 de Enero de 2023

Título: Análisis de dimensión fractal para series de tiempo de expresión génica mediante una red neuronal artificial.
Alumno: Marco Antonio Esperón Pintos

COMITÉ

Dr. Javier Miguel Hernández López
Presidente

Dr. José Jacobo Oliveros Oliveros
Secretario

Dr. Eduardo Moreno Barbosa
Vocal

Dr. Andrés Anzo Hernández
Vocal

Dr. Jorge Velázquez Castro
Dr. Benito de Celis Alonso
Asesor

Índice general

Resumen

Las células de los organismos biológicos son, a grandes rasgos, dispositivos integrados por varios miles de tipos de proteínas. Cada proteína es una máquina molecular de tamaño nanométrico que realiza una tarea específica. Gran parte del tiempo las células se enfrentan a diferentes situaciones que requieren del funcionamiento de estas proteínas. Por ejemplo, cuando las células perciben el azúcar, comienzan a producir proteínas que la transportan a su interior. Cuando estas proteínas se dañan, la célula produce proteínas de reparación. Por tanto, la célula vigila continuamente su entorno y calcula la cantidad necesaria de cada tipo de proteína [1]. Esta función de procesamiento de la información, que determina el ritmo de producción de cada proteína, es llevada a cabo por las redes de transcripción. Es de suma importancia comprender esta dinámica, ya que, sirve como un indicador del estado de salud celular y el correcto funcionamiento de las células. Tener métodos eficientes para el análisis de las señales (generadas como series de tiempo de las concentraciones proteicas) es de gran utilidad para el diagnóstico de ciertas enfermedades como el cáncer, que se expresan a nivel celular. De acuerdo a diversas investigaciones de expresión génica que se han realizado [2], se observa que las series de tiempo de concentraciones proteicas tienen un comportamiento fractal. En este trabajo se emulará computacionalmente un modelo dinámico estocástico de una red de regulación genética mínima para simular la dinámica característica de la concentración de proteínas. Al cambiar los parámetros del modelo se podrán simular distintas condiciones celulares que posteriormente serán de utilidad para el entrenamiento de una red neuronal artificial que reconozca las propiedades de la serie en el estado celular correspondiente. En particular se analizará la dimensión fractal de la señal. Esta investigación permitirá evaluar la eficiencia y factibilidad de emplear una red neuronal artificial que diagnostique estados celulares por medio de la dinámica de concentraciones proteicas.

Palabras clave: *Sistemas Complejos (Complex Systems), Redes Neuronales Artificiales (Artificial Neural Networks), Machine Learning, Procesos Estocásticos (Stochastic Processes), Fractales (Fractals), Expresión Génica (Gene Expression), Movimiento Browniano Fractal (Fractal Brownian Motion), Difusión (Diffusion).*

Introducción

Las células de organismos vivos se desarrollan en entornos donde existe constantemente un intercambio de diversas señales, es decir, son parte de un complejo sistema que es afectado por parámetros físicos como la temperatura y la presión osmótica. También perciben nutrientes y sustancias químicas perjudiciales. Las células responden a estas señales externas produciendo un tipo de proteína. Para representar estos estados ambientales, la célula utiliza como insignia unas proteínas llamadas factores de transcripción. Los factores de transcripción [1, 2] están diseñados para transitar rápidamente entre estados moleculares activos e inactivos a una velocidad modulada por una señal ambiental específica. Cada factor de transcripción activo puede unirse al ADN para regular el ritmo de lectura de determinados genes. La cantidad y representación interna, de un conjunto de factores de transcripción, es una descripción compacta de los efectos provocados por agentes del entorno. Los factores de transcripción se encargan de leer (transcribir) genes en ARNm, para posteriormente, ser traducidos en proteínas que puedan interactuar con el ambiente. Por tanto, la tasa de crecimiento y las actividades de los factores de transcripción en una célula pueden considerarse una consecuencia interna debido al entorno. Muchas situaciones diferentes se resumen en la actividad de un factor de transcripción concreto. A la traducción y producción de proteínas se les conoce como expresión génica, es decir, el proceso de síntesis de proteínas a partir de la activación de un gen específico. La célula genera un producto génico funcional que realiza actividades celulares específicas. Podemos concluir que la expresión génica es el proceso por el cual la información del genotipo da lugar al fenotipo (características observables). Es un proceso vital que provoca la diferenciación celular, morfogénesis y la versatilidad y adaptabilidad de cualquier organismo. Las proteínas no sólo le dan forma al organismo, también pueden funcionar como enzimas que catalizan procesos metabólicos específicos. Comprender la "dinámica de expresión genética" nos da una idea de cómo los organismos regulan y controlan la síntesis de proteínas. La regulación de la expresión génica controla la cantidad y tiempo de producción de proteínas objetivo. Investigar la dinámica de expresión génica permite entender mecanismos de organismos biológicos [2]. Por lo tanto, para poder entender la dinámica de expresión génica, es necesario investigar las propiedades estadísticas de los datos (series cronológicas) [3]. Por otro lado, las redes neuronales artificiales han mostrado ser una herramienta muy prometedora para el diagnóstico de enfermedades [4, 5]. En este trabajo se evaluará la factibilidad de emplear una red neuronal artificial para caracterizar series de tiempo de la concentración proteica. Esta caracterización podrá emplearse para

determinar el estado de salud a nivel celular. Para lograrlo se generarán series de tiempo sintéticas con características similares a las series de tiempo generadas por redes de regulación genética. Estas series de tiempo se emplearán para entrenar una red neuronal artificial que ayude a detectar sus parámetros característicos [5, 6]. Este conocimiento puede ser de ayuda para detectar anomalías y enfermedades. También puede ser útil para diseñar células que realicen tareas específicas, por ejemplo, la administración de fármacos para el cáncer (Biología sintética).
Las cuatro características más importantes de variación en la dinámica de la expresión de los genes incluyen:

- La estocasticidad inherente de los procesos bioquímicos que dependen del gran número de moléculas.
- Las diferencias en los estados internos de las células.
- Las sutiles diferencias ambientales.
- Las mutaciones genéticas.

Debido a que los procesos estocásticos son imprscindibles para el análisis y comprensión de la dinámica de expresión génica, se abordará este tema en el primer capítulo.

Capítulo 1

Procesos Estocásticos

Los procesos estocásticos son una rama de las matemáticas encargados del análisis de sistemas que son afectados por magnitudes aleatorias y que no pueden explicarse a través de formalismos matemáticos deterministas. Los procesos estocásticos se describen como una familia de funciones. Cada función representa una realización experimental. Por lo tanto podemos escribir un proceso estocástico $\xi(t)$ de la siguiente manera.

$$\xi(t) = [\xi_1(t), \xi_2(t), ..., \xi_n(t)] \tag{1.1}$$

Para definir un proceso estocástico es necesario asignar una probabilidad o densidad de probabilidad a cada realización del ensamble que se estudia. Cada función de probabilidad nos habla de la ocurrencia de las distintas realizaciones. Denotemos estas funciones de distribución de probabilidad de la siguiente manera

$$F(x;t) = [F_1, F_2, ..., Fn] = P[\xi(t) \leq x] \tag{1.2}$$

Donde $P[\xi(t) \leq x]$ es la probabilidad de que el proceso sea igual o menor a un determinado punto x. Podemos llegar a la conclusión de que $F(\infty;t) = 1$.
Una interpretación frecuentista de un proceso estocástico conlleva los siguientes pasos

- Realizar un experimento n veces
- Observar su trayectoria
- Asignar un par de números (x,t)
- Contar el número de veces que el experimento tuvo un valor igual o menor a x en el tiempo t, es decir, $n_t(x)$.
- Definir una función de distribución de la siguiente manera $F(x;t) \approx \frac{n_t(x)}{n}$

Obteniendo una función de distribución de probabilidad podemos encontrar una función de densidad de probabilidad en un tiempo t de la siguiente manera

$$f(x;t) = \frac{\partial F(x;t)}{\partial x} \tag{1.3}$$

Si quisiéramos obtener la probabilidad de que un proceso tome un valor en la vecindad de x_1 al tiempo t_1 y un valor en la vecindad de x_2 al tiempo t_2 sería necesario definir la distribución de probabilidad conjunta, es decir

$$F(x_1, x_2; t_1, t_2) = P[\xi(t_1) \leq x_1, \xi(t_2) \leq x_2] \tag{1.4}$$

De manera análoga la densidad de probabilidad conjunta es

$$f(x_1, x_2; t_1, t_2) = \frac{\partial^2 F(x_1, x_2; t_1, t_2)}{\partial x_1 \partial x_2} \tag{1.5}$$

y en general la podemos escribir como

$$f(x_1, ..., x_n; t_1, ..., t_n) = \frac{\partial F(x_1, ..., x_n; t_1, ..., t_n)}{\partial x_1 ... \partial x_n} \tag{1.6}$$

Una función importante que nos dice la probabilidad de que ocurra un evento x_2 en un tiempo t_2 habiendo ocurrido el evento x_1 en el tiempo t_1 es la densidad de probabilidad condicional. Esta se define de la siguiente manera

$$f(x_2, t_2|x_1, t_1) = \frac{f(x_1, x_2; t_1, t_2)}{f(x_1, t_1)} \tag{1.7}$$

1.1. Momentos de Procesos Estocásticos

Los momentos son funciones que ayudan a determinar propiedades estadísticas de los procesos estocásticos, es decir, valores numéricos que simplifican el comportamiento global del ensamble que se estudia. Generalmente los momentos abordan medidas de tendencia central. Los momentos más importantes son los siguientes

1.1.1. Media o Promedio (Primer Momento)

$$\mu(t) = \langle \xi(t) \rangle = \int_{-\infty}^{\infty} x f(x; t) dx \tag{1.8}$$

1.1.2. Función de Correlación (Segundo Momento)

La función de correlación nos permite asignar un valor numérico a la dependencia o relación que tiene un punto del proceso con otro en un tiempo posterior. La denotamos de la siguiente manera

$$B(t_1, t_2) = \langle \xi(t_1)\xi(t_2) \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f(x_1, x_2; t_1, t_2) dx_1 dx_2 \tag{1.9}$$

También podemos escribirla de la siguiente forma

$$B(t_1, t_2) = \langle \xi(t_1)\xi(t_2) \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f(x_2, t_2|x_1, t_1) f(x_1, t_1) dx_1 dx_2 \tag{1.10}$$

Si $\langle x_2 \rangle_{x_1}$ es el promedio de x_2 dado que el proceso estuvo en x_1 en el tiempo t_1, entonces

$$\langle x_2 \rangle_{x_1} = \int_{-\infty}^{\infty} x_2 f(x_2, t_2 | x_1, t_1) dx_2 \tag{1.11}$$

Ahora podemos reescribir la ecuación (1.10) como

$$\langle \xi(t_1)\xi(t_2) \rangle = \int_{-\infty}^{\infty} x_1 \langle x_2 \rangle_{x_1} f(x_1, t_1) dx_1 \tag{1.12}$$

Si la probabilidad de que ocurra x_2 después de ocurrir x_1 es pequeña, entonces el proceso estará poco correlacionado.
A esta función también se le conoce como función de autocorrelación, ya que, estamos hablando del mismo proceso estocástico. Si consideramos un par de procesos estocásticos, a $\langle \xi^1(t_1)\xi^2(t_2) \rangle$ se le conoce como correlación cruzada.

1.1.3. Covarianza (Tercer Momento)

Otra medida que ayuda a determinar la correlación del proceso es la covarianza. Esta cantidad nos devolvería un valor igual a cero sino existe correlación. Podemos denotarla así

$$C(t_1, t_2) = \langle [\xi(t_1) - \mu_1][\xi(t_2) - \mu_2] \rangle = \langle \langle \xi(t_1)\xi(t_2) \rangle \rangle \tag{1.13}$$

1.2. Procesos Estacionarios y el Teorema Ergódico

Si un proceso estocástico obedece el teorema ergódico, cumplirá que el promedio en el ensamble es igual al promedio en el tiempo. El teorema ergódico permite hacer una relación de lo que se mide en el experimento y las propiedades estadísticas teóricas.
Un proceso estocástico $\xi(t)$ es estacionario si sus funciones de distribución $F(x;t)$ permanecen constantes en el tiempo, es decir

$$F(x_1, x_2, ..., x_n; t_1 + \tau, t_2 + \tau, ..., t_n + \tau) = F(x_1, x_2, ..., x_n; t_1, t_2, ..., t_n) \tag{1.14}$$

Entonces para un proceso estocástico estacionario

$$\begin{aligned} F(x;t) &= P[\xi(t) \leq x] = F(x) \\ \langle \xi(t)\xi(s) \rangle &= B(t+s) = B(\tau) \end{aligned} \tag{1.15}$$

Por lo tanto, para procesos estacionarios se cumple el teorema ergódico.
Una manera de explicar el teorema es imaginando un conjunto muy grande de experimentos

$$\xi_1(t), \xi_2(t), ..., \xi_n(t) \tag{1.16}$$

Podemos calcular su promedio aritmético de la siguiente manera

$$\langle \xi(t) \rangle \approx \frac{1}{N} \sum_{j=1}^{N} \xi_j(t) \tag{1.17}$$

De forma similar se puede obtener el producto de todos los pares $\xi_j(t)\xi_i(s)$ y promediar para obtener $\langle \xi(t)\xi(s)\rangle = B(t,s)$. Ya que, esto es difícil de realizar. Lo que comúnmente se hace es promediar en el tiempo como se describe en la siguiente ecuación

$$\langle \xi(t)\rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T \xi(t)dt \tag{1.18}$$

La función de correlación, la podemos definir de la siguiente manera

$$\langle \xi(t)\xi(t+\tau)\rangle \approx \lim_{T\to\infty} \frac{1}{T}\int_0^T \xi(t)\xi(t+\tau)dt \tag{1.19}$$

En un experimento real, se hace la siguiente aproximación

$$\int_0^T \xi(t)dt \approx \frac{T}{N}\sum_{k=1}^{N} \xi(k\frac{T}{N}) \ , \ N\to\infty \tag{1.20}$$

Y por lo tanto

$$\langle \xi(t)\rangle \approx \sum_{k=1}^{N} \xi(k\Delta) \ , \ \Delta = \frac{T}{N} \tag{1.21}$$

Se concluye que un proceso estacionario es ergódico, es decir, no importa si se promedia en el tiempo o en el ensamble.

1.3. Movimiento Browniano

En los procesos estocásticos se toman en cuenta las fluctuaciones de un sistema, es decir, todos los grados de libertad o variables que lo afectan.
Es muy común analizar sistemas dinámicos con modelos deterministas pero para algunos sistemas estas fluctuaciones (ruido) no pueden despreciarse. El resumen de las fluctuaciones se consigue analizando propiedades estadísticas deterministas (momentos) como el promedio, varianza, etc.
La teoría de procesos estocásticos se comienza a desarrollar a partir del movimiento browniano.
El físico frances Louis Gouy describe en 1888 al movimiento browniano como

- Una trayectoria muy irregular que, al momento de graficar, parece no tener recta tangente en ningún punto. Esto logra observarlo realizando varios acercamientos a secciones específicas de la trayectoria.
- Las partículas se mueven independientemente entre ellas.
- La composición de la partícula no afecta su comportamiento.
- Si la viscosidad del solvente disminuye, el movimiento es más activo.
- Si el radio de la partícula disminuye, el movimiento es más activo.
- Si la temperatura aumenta, el movimiento es más activo.

- El movimiento nunca termina.

En 1905, partiendo de dos hipótesis, Albert Einstein describe al movimiento browniano.

- **Primera Hipótesis:** El movimiento es causado por los impactos de moléculas.
- **Segunda Hipótesis:** El movimiento puede ser descrito probabilísticamente.

Partiendo de estas hipótesis, Albert Einstein esboza una ecuación diferencial que tenga como solución una función de densidad de probabilidad que represente la densidad de partículas, es decir, las partículas por unidad de volumen. De forma simplificada, los pasos a seguir al desarrollar la ecuación fueron los siguientes

- Introducir un tiempo τ.
- Colocar n partículas brownianas en el liquido.
- El desplazamiento de partículas después del intervalo de tiempo τ se define como Δ.
- Asumir que existe una ley de frecuencias (función de densidad de probabilidad) para Δ que se denota por $\Phi(\Delta)$

Con esta información es posible definir el número dn de partículas que se desplazarán entre Δ y $\Delta + d\Delta$ como

$$dn = n\Phi(\Delta)d\Delta \tag{1.22}$$

Donde

$$\int_{-\infty}^{\infty} \Phi(\Delta)d\Delta = 1 \tag{1.23}$$

Einstein parte de la ecuación de Chapman-Kolmogorov. La cual describe de forma general los procesos de Markov, es decir, procesos para los cuales, el siguiente estado de una realización (experimento) sólo depende del estado actual y no de los pasados. Podría decirse que es un proceso sin "memoria".
La ecuación de Chapman-Kolmogorov tampoco nos da información de la forma que tiene la función de densidad de probabilidad. Esta ecuación tiene la siguiente forma

$$f(x, t+\tau)dx = dx \int_{-\infty}^{\infty} f(x-\Delta, t)\Phi(-\Delta)d\Delta \tag{1.24}$$

Asumiendo que Φ es una función par, es decir, $\Phi(\Delta) = \Phi(-\Delta)$, la ecuación (1.24) se reescribe como

$$f(x, t+\tau)dx = dx \int_{-\infty}^{\infty} f(x-\Delta, t)\Phi(\Delta)d\Delta \tag{1.25}$$

Aplicando una serie de taylor al lado izquierdo de la ecuación (1.25) y suponiendo τ pequeño, consideramos únicamente los dos primeros términos de la expresión y obtenemos lo siguiente

$$f(x, t+\tau) \approx f(x, t) + \tau \frac{\partial f}{\partial t} \tag{1.26}$$

Aplicamos nuevamente una expansión en serie de taylor. Ahora para el término $f(x-\Delta,t)$ ubicado en el lado derecho de la ecuación (1.25)

$$f(x-\Delta,t) \approx f(x,t) - \Delta\frac{\partial f}{\partial x} + \frac{\Delta^2}{2!}\frac{\partial^2 f}{\partial^2 x^2} + ... \tag{1.27}$$

Si suponemos que $\Phi(\Delta)$ es diferente de cero sólo para valores pequeños de Δ podemos concluir que Δ^n para $n = 3, 4, ...$ será muy pequeña. Por lo tanto, sólo consideramos los tres primeros términos de la ecuación (1.27).
Sustituyendo la ecuación (1.26) y (1.27) en (1.25) obtenemos que

$$f + \frac{\partial f}{\partial t}\tau = f\int_{-\infty}^{\infty}\Phi(\Delta)d\Delta - \frac{\partial f}{\partial x}\int_{-\infty}^{\infty}\Delta\Phi(\Delta)d\Delta + \frac{\partial^2 f}{\partial x^2}\int_{-\infty}^{\infty}\frac{\Delta^2}{2}\Phi(\Delta)d\Delta \tag{1.28}$$

Ya que, la suma de los Δ en el intervalo $(-\infty,\infty)$ es igual a cero, el segundo término del lado derecho de la ecuación será cero. Sustituyendo el valor de la ecuación (1.23) en la ecuación (1.28) obtenemos que

$$\frac{\partial f}{\partial t}\tau \approx \frac{\partial^2 f}{\partial x^2}\int_{-\infty}^{\infty}\frac{\Delta^2}{2}\Phi(\Delta)d\Delta \tag{1.29}$$

Sea

$$D = \frac{1}{\tau}\int_{-\infty}^{\infty}\frac{\Delta^2}{2}\Phi(\Delta)d\Delta \tag{1.30}$$

Concluimos que

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{1.31}$$

A esta ecuación diferencial parcial se le conoce como Ecuación de Difusión. El coeficiente de difusión D representa la velocidad de propagación de las partículas. La solución de la ecuación anterior es

$$f(x,t) = \frac{n}{\sqrt{4\pi Dt}}e^{\frac{-x^2}{2Dt}} \tag{1.32}$$

Con $\sigma = \sqrt{\langle x^2\rangle} = \sqrt{2Dt}$ la desviación estandar de $f(x,t)$

1.4. Transformada de Fourier

La transformada de fourier me indica cuáles son las frecuencias que conforman una señal. La transformada de fourier de una función o señal $f(t)$ es

$$\hat{f}(w) = \int_{-\infty}^{\infty}e^{-iwt}f(t)dt \tag{1.33}$$

También es posible obtener la señal original aplicando una transformada inversa, es decir

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}e^{iwt}\hat{f}(t)dw \tag{1.34}$$

1.5. Espectro de Potencias

El espectro de potencias, espectro de fluctuaciones o espectro de frecuencias se define como la transformada de fourier de la función de correlación, es decir

$$S(w) = \int_{-\infty}^{\infty} e^{-iw\tau} B(\tau) dt \tag{1.35}$$

Tomando en cuenta que para procesos estacionarios

$$B(\tau) = \langle \xi(t)\xi(t+\tau) \rangle \tag{1.36}$$

El espectro de potencias indica las frecuencias que tiene el ruido de la señal y de alguna manera el tiempo estocástico del ruido, es decir, si imaginamos una partícula browniana que es golpeada por partículas del medio en donde se encuentra. Podemos caracterizar esos golpeteos como el ruido que obtendríamos al graficar la trayectoria de la partícula browniana.
El espectro de potencias nos da una idea de la frecuencia de esos golpeteos o el tiempo característico de ese ruido.
Sabemos por las propiedades de la delta de dirac que

$$\int_{-\infty}^{\infty} \delta(t-t') f(t) dt = f(t') \tag{1.37}$$

Por lo tanto, la transformada de fourier de la delta de dirac es

$$\int_{-\infty}^{\infty} \delta(t) e^{-iwt} dt = 1 \tag{1.38}$$

Definimos la transformada de fourier de un proceso estocástico como

$$\Xi(w) = \int_{-\infty}^{\infty} \xi(t) e^{-iwt} dt \tag{1.39}$$

Si multiplicamos la transformada de fourier de un proceso estocástico por su conjugado, evaluados en frecuencias diferentes, obtenemos

$$\begin{aligned} \langle \overline{\Xi(w)} \Xi(w') \rangle &= \langle \int_{-\infty}^{\infty} \xi(t) e^{iwt} dt \int_{-\infty}^{\infty} \xi(t') e^{-iw't'} dt' \rangle \\ &= \langle \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \xi(t)\xi(t') e^{iwt} e^{-iw't'} dt dt' \rangle \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \langle \xi(t)\xi(t') \rangle e^{iwt} e^{-iw't'} dt dt' \end{aligned} \tag{1.40}$$

Realizando un cambio de variable donde $t' = t + \tau$, entonces $dt' = d\tau$ y

$$\langle \overline{\Xi(w)} \Xi(w') \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \langle \xi(t)\xi(t+\tau) \rangle e^{iwt} e^{-iw'(t+\tau)} dt d\tau \tag{1.41}$$

Tomando en cuenta la ecuación (2.4)

$$\langle\overline{\Xi(w)}\Xi(w')\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} B(\tau)e^{-iw'\tau}e^{-i(w'-w)t}dtd\tau \tag{1.42}$$

Consideramos la ecuación (2.3) y obtenemos

$$\langle\overline{\Xi(w)}\Xi(w')\rangle = \int_{-\infty}^{\infty} S(w')e^{-i(w'-w)t}dt \tag{1.43}$$

Se concluye que

$$\begin{aligned}\langle\overline{\Xi(w)}\Xi(w')\rangle &= S(w')\delta(w'-w)\\ \langle\overline{\Xi(w)}\Xi(w)\rangle &= S(w')\end{aligned} \tag{1.44}$$

Esto nos dice que el espectro de fluctuaciones es la transformada de fourier de la correlación. En otras palabras, el promedio de la transformada de fourier de la multiplicación del ruido por su conjugado.

Capítulo 2

Funcion de Transferencia

Las funciones de transferencia son ecuaciones que relacionan la señal de salida de un sistema con una señal de entrada.
De acuerdo con [7] la idea básica de la función de transferencia proviene de observar la respuesta en frecuencia de un sistema. Supongamos que tenemos una señal de entrada que es periódica. Entonces podemos descomponer esta señal en la suma de un conjunto de senos y cosenos. La ganancia y la fase en cada frecuencia están determinadas por la respuesta en frecuencia de una transformada de fourier. El modelado de un sistema a través de su respuesta a señales sinusoidales y exponenciales se conoce como modelado en el dominio de la frecuencia. Esta terminología proviene del hecho de que representamos la dinámica del sistema en términos de la frecuencia generalizada en lugar de la variable del tiempo. La función de transferencia proporciona una representación completa de un sistema lineal en el dominio de la frecuencia.
Supongamos una ecuación diferencial lineal no homogénea

$$[a_n \frac{d^n}{dt^n} + a_{n-1} \frac{d^{n-1}}{dt^{n-1}} + ... + a_0]y(t) = \eta(t) \tag{2.1}$$

Que también podemos representar de la siguiente manera

$$\pounds y(t) = \eta(t) \tag{2.2}$$

Donde $\eta(t)$ representa una función estocástica o fuerza estocástica.
Se propone como solución de la ecuación diferencial la siguiente igualdad.

$$y(t) = \int_{-\infty}^{\infty} \eta(t-\tau)h(t)d\tau \tag{2.3}$$

a $h(t)$ se le conoce como función de green. Esta función cumple que

$$\pounds h(t) = \delta(t) \tag{2.4}$$

Con base en esta relación, veamos si $y(t)$ es solución de (2.1)

$$\pounds y(t) = \pounds[\int_{-\infty}^{\infty} \eta(t-\tau)h(\tau)d\tau] \tag{2.5}$$

sea $t' = t - \tau$ entonces $dt' = -d\tau$

$$\pounds y(t) = \pounds[\int_{\infty}^{-\infty} \eta(t')h(t-t')dt'] \tag{2.6}$$

Como $\pounds$ sólo opera sobre t, entonces

$$\pounds y(t) = \int_{\infty}^{-\infty} \eta(t')\pounds h(t-t')dt' \tag{2.7}$$

$$\pounds y(t) = \int_{\infty}^{-\infty} \eta(t')\delta(t-t')dt' \tag{2.8}$$

Y se concluye que

$$\pounds y(t) = \eta(t) \tag{2.9}$$

Por lo tanto, vemos que $y(t)$ es solución de la ecuación diferencial.
Por otra parte, es importante saber lo que ocurre cuando se le aplica la transformada de fourier a un operador diferencial. Si representamos al operador de la transformada como F, entonces

$$\begin{aligned} F[\frac{dy}{dt}](w) &= \int_{-\infty}^{\infty} e^{-iwt}\frac{dy}{dt}dt \\ &= e^{-iwt}y\mid_{-\infty}^{\infty} - \int_{-\infty}^{\infty} y\frac{d}{dt}e^{-iwt}dt \end{aligned} \tag{2.10}$$

Asumiendo que $y(t)$ es una función L^2 se cumple que $y(\pm\infty) \to 0$ y por lo tanto

$$\begin{aligned} F[\frac{dy}{dt}](w) &= -\int_{-\infty}^{\infty} y\frac{d}{dt}e^{-iwt}dt \\ &= iw\int_{-\infty}^{\infty} ye^{-iwt}dt \\ &= iwF[y] \end{aligned} \tag{2.11}$$

Con este resultado se observa que en el espacio de fourier una operación diferencial se convierte en una operación algebraica. En general la ecuación (2.11) la podemos escribir como

$$F[\frac{d^n y}{dt^n}](w) = (iw)^n F[y] \tag{2.12}$$

Es decir que, para la ecuación diferencial

$$[a_n\frac{d^n}{dt^n} + a_{n+1}\frac{d^n}{dt^n} + ... + a_0]h(t) = \delta(t) \tag{2.13}$$

La ecuación algebraica en el espacio de fourier es

$$[a_n(iw)^n + a_{n-1}(iw)^{n-1} + ... + a_0]\hat{h} = 1 \tag{2.14}$$

Por lo tanto

$$\hat{h}(w) = \frac{1}{a_n(iw)^n + a_{n-1}(iw)^{n-1} + ... + a_0} \tag{2.15}$$

A esta ecuación se le conoce como función de transferencia. Basta con realizar la transformada inversa de fourier para poder obtener $h(t)$, es decir

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iwt}\hat{h}(w)dw \tag{2.16}$$

Es importante conocer el espectro de potencias de la solución $y(t)$ de la ecuación diferencial general (2.1). Para poder deducirla, multiplicamos $\eta(t+\tau)$ por la función de green conjugada y haciendo promedio sobre el ensamble obtenemos

$$\begin{aligned}
\langle \eta(t+\tau)\overline{y(t)}\rangle &= \int_{-\infty}^{\infty} \langle\ \eta(t+\tau)\overline{\eta(t-\tau')}\rangle \overline{h(\tau')}d\tau' \\
&= \int_{-\infty}^{\infty} \langle\ \eta(t+\tau)\overline{\eta(t-\tau')}\rangle \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-iw\tau'}\overline{\hat{h}(w)}dwd\tau' \qquad (2.17)\\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} B_{\eta\eta}(\tau'+\tau)e^{-iw\tau'}\hat{h}(w)dwd\tau'
\end{aligned}$$

En este caso nombramos la función de autocorrelación de η como $B_{\eta\eta}$. Haciendo $\overline{\tau} = \tau' + \tau$ entonces $d\overline{\tau} = d\tau'$

$$\begin{aligned}
\langle \eta(t+\tau)\overline{y(t)}\rangle &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} B_{\eta\eta}(\overline{\tau})e^{-iw(\overline{\tau}-\tau)}\overline{\hat{h}(w)}dwd\overline{\tau} \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iw\overline{\tau}}B_{\eta\eta}(\overline{\tau})d\overline{\tau}e^{iw\tau}\overline{\hat{h}(w)}dw \qquad (2.18)\\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{\eta\eta}(w)e^{iw\tau}\overline{\hat{h}(w)}dw
\end{aligned}$$

Donde $S_{\eta\eta}$ es el espectro de potencias de η. Reordenando la ecuación anterior obtenemos que

$$\langle \eta(t+\tau)\overline{y(t)}\rangle = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iw\tau}S_{\eta\eta}(w)\overline{\hat{h}(w)}dw \tag{2.19}$$

Aplicando la transformada de fourier a ambos lados de la igualdad anterior obtenemos que

$$S_{\eta y}(w) = S_{\eta\eta}\overline{\hat{h}(w)} \tag{2.20}$$

Donde $S_{\eta y}$ es la correlación cruzada de η y y. De forma similar se obtiene que

$$S_{yy}(w) = h(w)S_{\eta y}(w) \tag{2.21}$$

Sustituyendo la ecuación (2.20) en la ecuación (2.21) obtenemos la relación fundamental

$$S_{yy}(w) = S_{\eta\eta}(w)h(w)\overline{h(w)} \tag{2.22}$$

O bien

$$S_{yy}(w) = S_{\eta\eta}(w)|h(w)|^2 \tag{2.23}$$

Un ejemplo que aplica los siguientes conceptos es el siguiente.
Consideremos la ecuación diferencial

$$\frac{dy}{dt} = -\beta y + \eta(t) \tag{2.24}$$

con $\langle \eta(t)\eta(t')\rangle = \Gamma\delta(t-t')$. Sea

$$\frac{dh}{dt} + \beta h = \delta(t) \tag{2.25}$$

Primero resolvemos la ecuación (2.25) en el espacio de fourier para encontrar la función de transferencia, es decir

$$iw\hat{h}(w) + \beta\hat{h}(w) = 1 \tag{2.26}$$

Entonces

$$\begin{aligned} \hat{h}(w) &= \frac{1}{iw+\beta} \\ |\hat{h}(w)|^2 &= \frac{1}{w^2+\beta^2} \end{aligned} \tag{2.27}$$

De modo que

$$\begin{aligned} S_{yy}(w) &= \frac{S_{\eta\eta}(w)}{w^2+\beta^2} \\ S_{\eta\eta}(w) &= \int_{-\infty}^{\infty} e^{-iwt}\Gamma\delta(t')dt' \end{aligned} \tag{2.28}$$

Finalmente podemos decir que $S_{\eta\eta}(w) = \Gamma$ y concluir que

$$S_{yy}(w) = \frac{\Gamma}{w^2+\beta^2} \tag{2.29}$$

Haciendo la transformada inversa de fourier de la ecuación anterior obtenemos

$$\langle y(t)y(t-\tau)\rangle = \Gamma\frac{e^{-\beta|\tau|}}{2\beta} \tag{2.30}$$

Capítulo 3

Series Temporales

De acuerdo con [3] las series temporales son un conjunto de datos medidos en determinados momentos y ordenados cronológicamente.
Desde el punto de vista probabilístico, una serie temporal es una sucesión de variables aleatorias indexadas según un parámetro creciente con el tiempo.
Para el análisis de las series temporales se usan métodos que ayudan a interpretarlos y que permiten extraer información representativa, permitiendo extrapolar o interpolar los datos y así predecir el comportamiento de la serie en momentos no observados.
Una serie temporal puede tomarse como solución de una ecuación diferencial estocástica. Una serie temporal estacionaria $y(t)$ puede considerarse como una función solución de una ecuación diferencial bajo la excitación de ruido blanco estacionario $w(t)$. Denotemos por $g(t)$ la función de green o función de impulso de una ecuación diferencial lineal estocástica y escribamos a $y(t)$ de la siguiente manera:

$$y(t) = \int_0^t g(t-\tau)w(\tau)\,d\tau = g * w \tag{3.1}$$

La convolución de g y w se denota por $g * w$ que como se mencionó, es solución de una ecuación diferencial estocástica que podemos escribir de la siguiente forma:

$$\sum_{i=0}^{p} a_i \frac{d^{p-i}y(t)}{dt^{p-1}} = w(t) \tag{3.2}$$

$g(t)$ puede tomar formas diferentes dependiendo de la señal. Por otra parte, si el sistema no es estacionario el ruido $w(t)$ no deberá ser estacionario.
De acuerdo con [5] podemos decir que $y(t)$ es una serie convencional si $y(t)$ va de $\mathbb{R}^1$ a $\mathbb{R}$. Por otro lado, se dice que $y(t)$ es una serie temporal fractal si pertenece a $\mathbb{R}^{1+d}$ para $0 < d < 1$. Será sólo la parte entera perteneciente a $\mathbb{R}^1$ lo que veremos en cualquier gráfica de $y \in R^{1+d}$. Sin embargo, es la parte fraccionaria de $y(t)$ la que la hace sustancialmente diferente de una serie convencional en los aspectos de su Función de Distribución de Probabilidad (PDF), Función de Autocorrelación (ACF) y Función de Densidad Espectral de Potencia (PSD).

3.1. Series de Tiempo Fractal

Según [5] las series temporales fractales difieren sustancialmente de las convencionales en sus propiedades estadísticas. Por ejemplo:

- Puede tener una función de distribución de probabilidad de cola pesada (PDF), es decir, la función de distribución de probabilidad no converge rápidamente a 1 para valores grandes de la variable aleatoria.
- Una función de autocorrelación (ACF) decaída lentamente y una función de densidad espectral de potencia (PSD) de tipo 1/f.
- Puede tener una dependencia estadística, ya sea una dependencia de largo alcance (LRD) o una dependencia de corto alcance (SRD), y una autosimilitud global o local.

Un punto particular, es que normalmente puede no existir media y/o varianza en series de tiempo fractales. Esta puede ser una de las principales razones de tratar de calcular otros valores como la dimensión fractal y el parámetro de Hurst.
Anteriormente se comentó que una serie cronológica convencional puede ser solución de una ecuación diferencial estocástica. De igual forma, una serie cronológica fractal es solución de una ecuación diferencial estocástica fraccionaria.
Sean $\nu > 0$ y $f(t)$ continuas en $(0, \infty)$ e integrables en cualquier subintervalo finito de $[0, \infty)$. Para $t > 0$ denotamos ${}_0D_t^{-v}$ como el operador integral de Riemann-Liouville de orden ν que está definido como

$$ {}_0D_t^{-v} f(t) = \frac{1}{\Gamma(\nu)} \int_0^t (t-u)^{\nu-1} f(u) du \tag{3.3} $$

Donde Γ es la función Gamma y ${}_0D_t^{-v}$ será solución de una ecuación diferencial estocástica fraccionaria.
Para simplificar la notación, denotemos al operador de Riemann-Liouville como D^{-v} y definamos la ecuación diferencial fraccionaria como

$$ \sum_{i=0}^{p} a_{p-i} D^{v_i} f(t) = \sum_{i=0}^{q} b_{q-i} D^{u_i} w(t) \tag{3.4} $$

Con a_i, b_i constantes y $\nu_p, \nu_{p-1}, ..., \nu_0$ y $u_p, u_{p-1}, ..., u_0$ un par de secuencias numéricas decrecientes positivas.
Para fines prácticos, podemos reescribir la ecuación anterior de la siguiente manera

$$ \sum_{i=0}^{p} a_{p-i} D^{v_i} f(t) = w(t) \tag{3.5} $$

3.2. Propiedades Básicas de Series de Tiempo Fractal

Las series temporales fractales tienen sus propiedades particulares en comparación con las convencionales. Una de estas propiedades es la función de autocorrelación.

Si denotamos la función de autocorrelación de $x(t)$ como $r_{xx}(\tau)$ donde $r_{xx}(\tau) = \langle x(t)x(t+\tau)\rangle$ entonces $x(t)$ tiene una dependencia de corto alcance (SRD) si r_{xx} es integrable es decir

$$\int_0^\infty r_{xx}(\tau)d\tau < \infty \tag{3.6}$$

Por otra parte, $x(t)$ tiene una dependencia de largo alcance (LRD) si no es integrable, es decir

$$\int_0^\infty r_{xx}(\tau)d\tau = \infty \tag{3.7}$$

Una forma común para funciones de autocorrelación no integrables es la siguiente

$$r_{xx} \approx c|\tau|^{-\beta} \quad (\tau \to \infty) \tag{3.8}$$

Donde $c > 0$ y $0 < \beta < 1$. La ecuación (3.8) implica que la función de autocorrelación para procesos fractales con dependencia de largo alcance tienen la forma de ley de potencias.
El espectro de potencias de un proceso con dependencia de largo alcance también tiene la forma de una ley de potencias. Usualmente se le conoce como ruido $1/f$. El espectro de potencias de una serie fractal de largo alcance tiene la propiedad de divergir para $w = 0$.
Si r_{xx} no es integrable, se puede observar, como una consecuencia de un proceso de largo alcance, que la PDF será de cola pensada.
Sea $p(x)$ la PDF de $x(t)$, podemos escribir el promedio de $x(t)$ de la siguiente forma

$$\mu_x = \int_{-\infty}^\infty xp(x)dx \tag{3.9}$$

y la varianza de $x(t)$ como

$$\sigma_x = \int_{-\infty}^\infty (x-\mu_x)^2 p(x)dx \tag{3.10}$$

Algo notable en las series de tiempo con dependencia de largo alcance es que la cola de las PDF puede ser tan pesada que la integral de las ecuaciones (3.9) y (3.10) puede no existir.
Es importante notar que μ_x implica una propiedad global de $x(t)$ mientras que σ_x representa una propiedad local de $x(t)$. Desafortunadamente para un proceso LRD, los conceptos de media y varianza son inapropiados para describir la propiedad global y local de $x(t)$. Necesitamos otras medidas para caracterizar la propiedad global y la local de procesos LRD. Para ello se utilizan la dimensión fractal y el parámetro de Hurst.

3.3. Dimensión Fractal y el Parámetro de Hurst

En el caso de las series de tiempo fractales, se utilizan la dimensión fractal y el parámetro de Hurst de $x(t)$ para describir su propiedad local y global respectivamente. Es importante mencionar que el parametro de Hurst está acotado en un

intervalo $[0, 1]$. Si r_{xx} es suficientemente suave en $(0, \infty)$ y si

$$r_{xx}(0) - r_{xx}(\tau) \approx c_1|\tau|^{\alpha} \quad para\ |\tau| \to 0 \tag{3.11}$$

Donde c_1 es una constante y α es el índice fractal de $x(t)$. La dimensión fractal de $x(t)$ se expresa como

$$D = 2 - \frac{\alpha}{2} \tag{3.12}$$

Por otra parte, si expresamos a β en términos del parámetro de Hurst (H) con $0 < H < 1$, entonces

$$\beta = 2 - 2H \tag{3.13}$$

y por lo tanto

$$r_{xx}(\tau) = c|\tau|^{\beta} \quad (\tau \to \infty) \tag{3.14}$$

Como $\alpha/2 = 2H$, podemos llegar a expresar la dimensión fractal como

$$D = 2 - H \tag{3.15}$$

A diferencia de las series convencionales, utilizamos D y H respectivamente, para caracterizar la propiedad local y la global de señales LRD en lugar de la media y la varianza.
Las propiedades que uno puede señalar de las series de tiempo, mediante el valor numérico del exponente de Hurst son los siguientes

- $H = 0{,}5$ **(Ruido Blanco)** Proceso completamente aleatorio e independiente, con ausencia de correlación entre los incrementos de la señal.
- $0{,}5 < H \leq 1$ **(Ruido Negro)** Series de tiempo que muestran procesos persistentes o correlacionados (un periodo de crecimiento es seguido de otro análogo) y presenta un aspecto "suave".
- $0 \leq H < 0{,}5$ **(Ruido Rosa)** Corresponde a un comportamiento de anti-persistencia o anti-correlación en la serie de tiempo (un periodo de crecimiento es seguido de otro de decrecimiento) que se caracteriza por un mayor contenido de alta frecuencia.

Mientras el exponente de Hurst tenga un valor mayor a 0.5 más autosimilar es la serie (Figura 3.1).
la autosimilitud del proceso estacionario es un concepto estrechamente relacionado con las series temporales fractales. El ruido gaussiano fraccional (fGn) es un proceso de incremento estacionario con autosimilitud global. Una serie temporal fractal puede no ser globalmente autosimilar. Sin embargo, una serie que no es autosimilar globalmente puede ser localmente autosimilar.

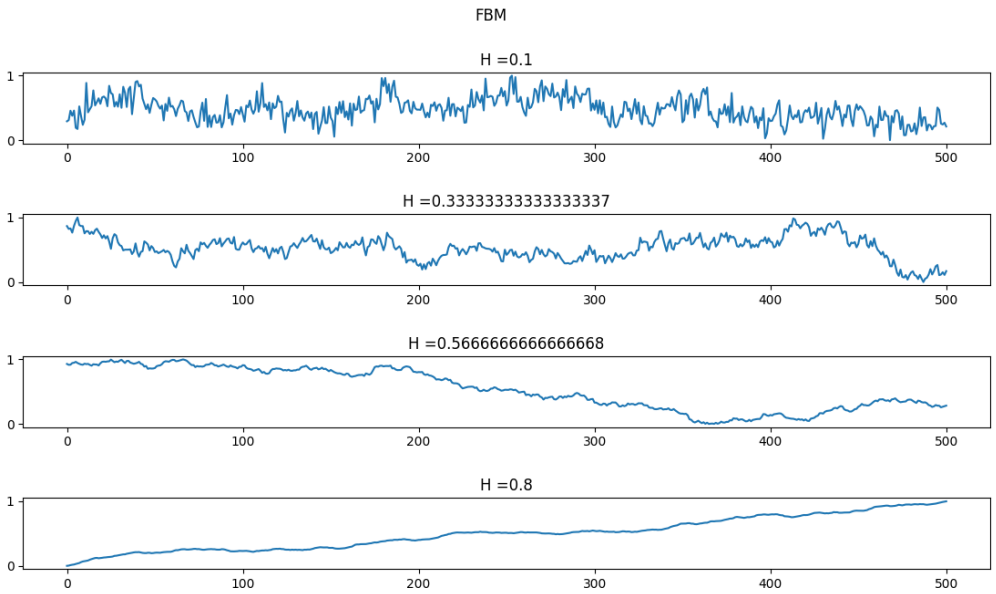

Figura 3.1: Series de tiempo de movimiento browniano fractal con diferentes exponentes de hurst

3.4. Movimiento Browniano Fraccionario

Una serie temporal fractal muy utilizada es el movimiento browniano fraccionario (fBm). Sustituyendo ν por $H + 0{,}5$ en la ecuación (3.3) para $0 < H < 1$, donde H es el parámetro de Hurst, entonces, fBm se define, utilizando el operador integral de Riemann-Liouville como

$$B_H(t) = {}_0D_t^{-(H+\frac{1}{2})} B'(t) = \frac{1}{\Gamma(H+\frac{1}{2})} \int_0^t (t-u)^{(H-\frac{1}{2})} dB(u) \tag{3.16}$$

Capítulo 4

Redes Neuronales Artificiales

En la estadística clásica se suele asumir un modelo y con un conjunto de datos se utilizan técnicas que ayudan a predecir los parámetros del modelo, por ejemplo, Mínimos Cuadrados o Ajuste por Polinomios.
Cuando utilizamos redes neuronales no conocemos ni el modelo ni los parámetros, es decir, las características necesarias para predecir nuevos datos.
Las redes neuronales también son herramientas muy útiles para sistemas de muchas dimensiones.
Las técnicas de machine learning tomaron relevancia cuando comenzamos a almacenar grandes cantidades de datos y los ordenadores personales tuvieron una mayor capacidad de cómputo.

4.1. Los tres tipos de algoritmos de Machine Learning

- **Aprendizaje Supervisado:**
  Se conocen los valores o datos de entrada y los de salida. Se ajustan los parámetros del modelo para que coincida con los resultados que se desean obtener. Un ejemplo es el modelo predictivo de imagenes de gatos y perros.
- **Aprendizaje no supervisado:**
  No tiene datos de salida específicos. Generalmente agrupa datos (Clustering) tomando en cuenta ciertas características. Partiendo de eso, uno puede llegar a sacar ciertas conclusiones (interpretar los datos).
- **Aprendizaje Reforzado:**
  Se basa en un aprendizaje supervisado pero los datos de salida también funcionan como datos de entrada hasta obtener el resultado deseado. Los datos de entrada se consiguen a partir de la interacción (generalmente con humanos).

4.2. Algoritmo de Machine Learning

Para crear un algoritmo de machine learning son necesarios los siguientes inredientes

1. **Conjunto de datos (dataset)**

$$D = (x, y) \quad \text{con} \quad x \rightarrow \text{Datos} \quad y \rightarrow \text{Etiqueta asignada a cada dato}$$

2. **Modelo**

$$f(x;\theta) = \hat{y} \quad f : x \rightarrow y \quad \theta\text{: Parámetros a ajustar}$$

3. **Función de Costo**

$$C(y, f(x;\theta)) = C(y, \hat{y})$$

   Nos dice la distancia entre los datos y los valores que predice el modelo.
   El objetivo es encontrar θ que minimice la función de costo.

Los pasos a seguir para resolver un problema con machine learning son los siguientes

1. Dividir el conjunto de datos en datos de entrenamiento (training data) y datos de prueba (test data). Generalmente los datos de entrenamiento son el 90 % del conjunto de datos.

2. Utilizando los datos de entrenamiento se encuentra

$$\hat{\theta} = argmin_{\theta}\{C(y_{train}, f(x_{train};\theta))\}$$

3. Se prueba la eficiencia del modelo utilizando los datos de prueba

$$C(y_{test}, f(x_{test};\hat{\theta}))$$

4. Se define el error dentro de la muestra como

$$E_{in} = C(y_{train}, f(x_{train};\theta))$$

   y el error fuera de la muestra (estimador sin sesgo de la eficiencia del modelo) como

$$E_{out} = C(y_{test}, f(x_{test};\hat{\theta}))$$

   Generalmente $E_{out} > E_{in}$.

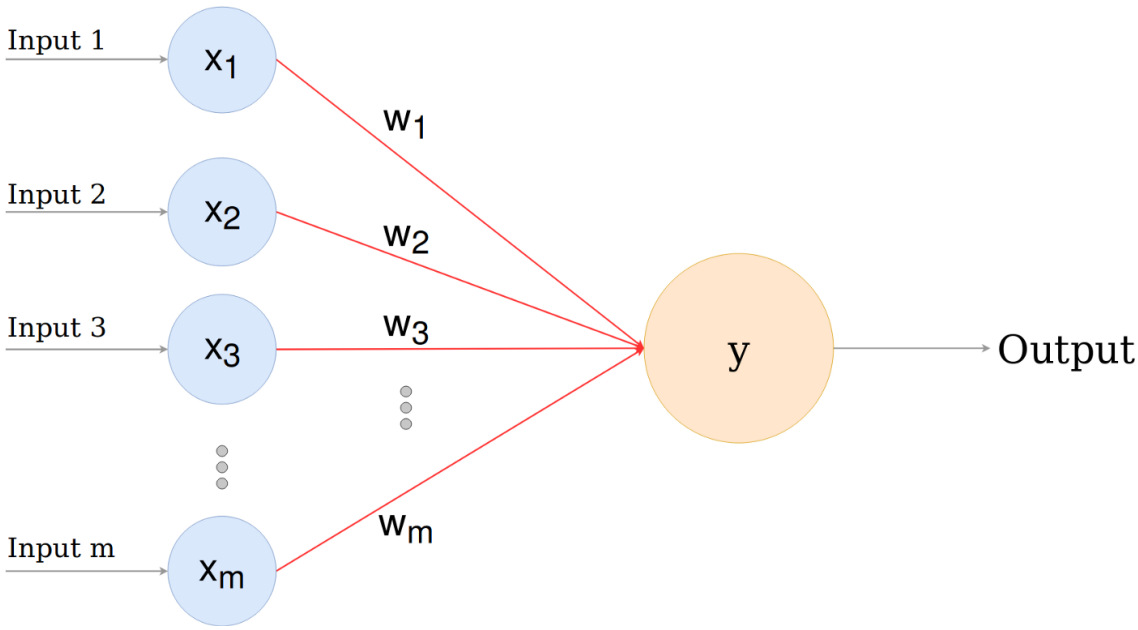

Figura 4.1: Perceptrón

4.3. Perceptrón

El perceptrón es la red neuronal más simple que existe. Cuenta únicamente con una neurona a la cuál se le conoce como perceptrón.
Matemáticamente el perceptrón (Figura 4.1) es una función "**y**" que tiene diferentes variables de entrada (inputs) $x_1, x_2, ..., x_m$ y una salida (output), la cual, únicamente puede tomar como valor 0 o 1.
La salida debe satisfacer las siguientes características.

$$\begin{aligned} 0 \quad & si \quad \sum_j w_j x_j \leq Umbral \\ 1 \quad & si \quad \sum_j w_j x_j > Umbral \end{aligned} \tag{4.1}$$

Donde x_j son las variables de entrada y w_j los pesos (parámetros) asignados a esas variables y el umbral corresponde a un número del cual se hablará posteriormente.
El perceptrón sirve como un modelo simple de toma de decisiones.
A la función que tiene como dominio, tanto a los pesos como a las variables se le conoce como función de activación.
A la función de activación asociada al perceptrón se le conoce como función escalón de Heaviside. Sea $Z = \sum_j w_j x_j$ podemos representar la función escalón como se muestra en la Figura 4.3

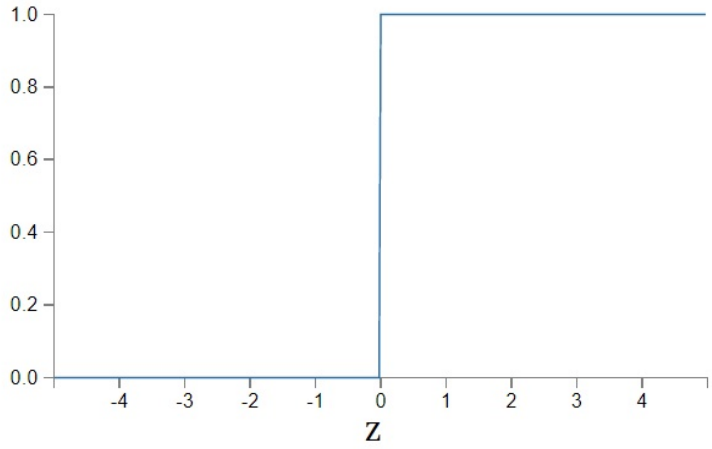

Figura 4.2: Función escalón de Heaviside

4.3.1. Ejemplo

Supongamos que tenemos la disyuntiva de ir o no a una fiesta.
Nuestra desición será sopesada tomando en cuenta las siguientes tres preguntas.

$$\begin{aligned} El\ clima\ es\ bueno \rightarrow w_1 &\begin{cases} x_1 = 0 & Mal\ clima \\ x_1 = 1 & Buen\ clima \end{cases} \\ Vamos\ con\ alguien \rightarrow w_2 &\begin{cases} x_2 = 0 & Solos \\ x_2 = 1 & Acompañados \end{cases} \\ Hay\ transporte\ publico \rightarrow w_3 &\begin{cases} x_3 = 0 & No\ hay \\ x_3 = 1 & Sí\ hay \end{cases} \end{aligned} \tag{4.2}$$

Los valores de los parámetros "w" se asignan dependiendo de la importancia que uno le de a los factores definidos anteriormente, por ejemplo, si nos importa mucho que haya buen clima; queremos ir sólos; y nos importa poco si nos movemos a la fiesta en transporte público, podríamos definir a los parámetros como: $w_1 = 6$; $w_2 = 3$; $w_3 = 2$.
Supongamos que elegimos un umbral de valor cinco. Entonces nuestra función quedaría de la siguiente forma

$$\begin{aligned} No\ vamos\ a\ la\ fiesta\ si \quad & 6x_1 + 3x_2 + 2x_3 \leq 5 \\ vamos\ a\ la\ fiesta\ si \quad & 6x_1 + 3x_2 + 2x_3 > 5 \end{aligned} \tag{4.3}$$

De la ecuación anterior se observa que podemos tolerar cualquier circunstancia excepto el mal clima.
El umbral, por lo tanto, determina el nivel de tolerancia para ciertas circunstancias (decisiones tomadas).

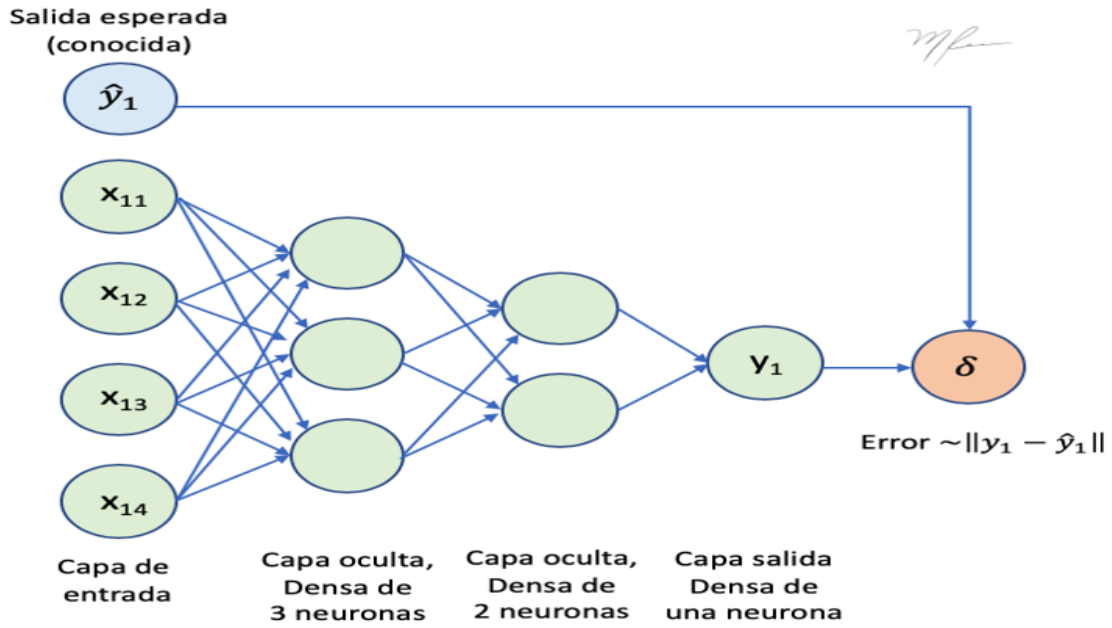

Figura 4.3: Red Neuronal Densa

4.4. Red Neuronal Densa

Las redes neuronales densas pueden definirse como un conjunto de neuronas ordenadas en una o varias capas. Cada una de las neuronas o perceptrones de cada capa es conectada con todas las neuronas de la capa siguiente (Figura 4.3). Este tipo de redes neuronales, en comparación con el perceptrón, sirve para tomar decisiones más sutiles o abstractas.
Tanto los datos de entrada como la salida de cada neurona es única y se conecta con todas los neuronas de la siguiente capa.
De igual manera, como ocurre cuando se tiene un perceptrón, cada una de las neuronas de la red neuronal densa representa una función. Debido a que la red neuronal densa, es útil para resolver problemas más sutiles y complejos, entonces las funciones de activación suelen ser diferentes a la función escalón que se utiliza en el perceptrón. Entre las funciones de activación más utilizadas en problemas resueltos con ayuda de redes neuronales densas se encuentran las funciones "Relu" y "Sigmoide".

4.4.1. El problema de reconocer el exponente de hurst de series de tiempo mediante redes neuronales densas

En la figura 4.4 se muestra un bosquejo de un par de series de tiempo numeradas donde "**FBM**" representa los puntos del movimiento browniano fractal a través del tiempo. En la primera serie los valores que toma la función se representan con la variable "x" seguido de un número.
La gráfica que está debajo de las series muestra una simplificación de lo que sería visualizar en una gráfica de cuatro dimensiones el exponente de hurst **f(x)** asociado a

los tres primeros puntos de las series de tiempo. Cada punto representa el exponente de hurst de las series que van de 1 hasta n.
Para este trabajo en específico se computaron 1000 series de tiempo, cada una con 500 puntos, por lo tanto, este será un problema en el cual se quiere ajustar una función dentro de un espacio de 501 dimensiones.

4.4.2. Explicación matemática de cómo una red neuronal densa puede calcular funciones

Tomando en cuenta la figura 4.3 podemos decir que la capa de entrada tiene los valores del movimiento browniano fractal para la primera serie de tiempo, es decir, $x_{11}, x_{12}, x_{13}, x_{14}$ donde el primer subíndice representa la serie de tiempo y el segundo subíndice representa el valor que toma la función "FBM" en cada punto de la serie. Por lo tanto para la primera capa densa de tres neuronas de la figura 4.3 cada neurona va a tomar un valor que se obtiene de la siguiente manera

$$\begin{aligned} y_1^1 &= \sigma\left(w_{11}^1 x_{11} + w_{12}^1 x_{12} + w_{13}^1 x_{13} + w_{14}^1 x_{14} + b_1^1\right) \\ y_2^1 &= \sigma\left(w_{21}^1 x_{11} + w_{22}^1 x_{12} + w_{23}^1 x_{13} + w_{24}^1 x_{14} + b_2^1\right) \\ y_3^1 &= \sigma\left(w_{31}^1 x_{11} + w_{32}^1 x_{12} + w_{33}^1 x_{13} + w_{34}^1 x_{14} + b_3^1\right) \end{aligned} \tag{4.4}$$

Para la segunda capa oculta de 2 neuronas obtendríamos los siguientes valores para cada neurona

$$\begin{aligned} y_1^2 &= \sigma\left(w_{11}^2 y_1^1 + w_{12}^2 y_2^1 + w_{13}^2 y_3^1 + b_1^2\right) \\ y_2^2 &= \sigma\left(w_{21}^2 y_1^1 + w_{22}^2 y_2^1 + w_{23}^2 y_3^1 + b_1^2\right) \end{aligned} \tag{4.5}$$

Por lo tanto, la capa de salida de una neurona densa obtendría el siguiente valor

$$y_1 = y_1^3 = \sigma\left(w_{11}^3 y_1^2 + w_{12}^3 y_2^2 + b_1^3\right) \tag{4.6}$$

Este valor de salida se aplicará a una función de costo. Esta función casi siempre es el error cuadrático medio definido como

$$mse = \frac{1}{n}\sum_{i=1}^{n}(y_1 - \hat{y_1})^2 \tag{4.7}$$

o el error absoluto medio definido como

$$mae = \frac{1}{n}\sum_{i=1}^{n}\|(y_1 - \hat{y_1})\| \tag{4.8}$$

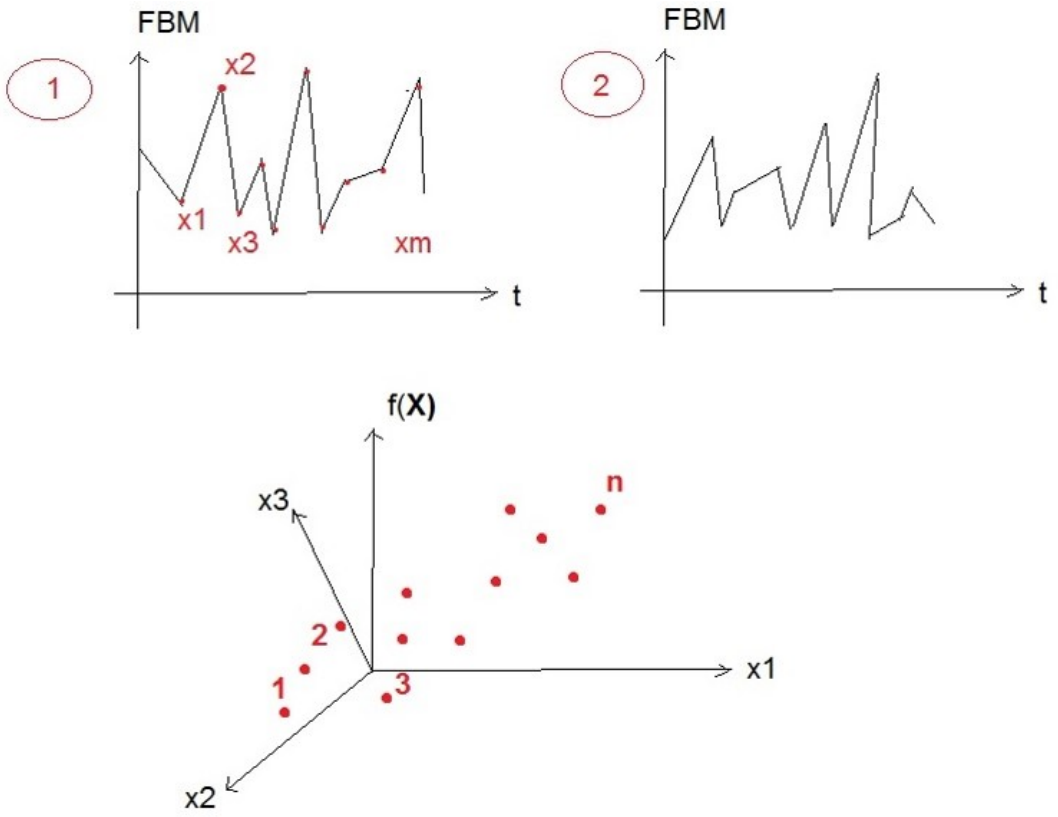

Figura 4.4: Series de tiempo. Gráfica del exponente de hurst $f(x)$ vs los valores de cada serie

El objetivo de los problemas resueltos con redes neuronales artificiales es minimizar la función de costo para cada valor obtenido. Para ello se utiliza el algoritmo de backpropagation que se explica en la siguiente sección.
Por último es importante mencionar que las ecuaciones 4.4, 4.5 y 4.6 pueden resumirse de la siguiente manera.
Sea "w^l_{jk}" el peso desde la "k" ésima neurona en la capa "$l-1$" hacia la "j" ésima neurona en la "l" ésima capa. Podemos escribir las expresiones anteriores como

$$y^l_j = \sigma\left(\sum_k w^l_{jk} y^{l-1}_k + b^l_j\right) \tag{4.9}$$

4.4.3. Algoritmo Descenso del Gradiente

Podemos notar que las funciones de costo expresadas en las ecuaciones 4.7 y 4.8 dependen de los valores de "y" que a su vez dependen de la función de activación "σ" que a su vez depende de la suma ponderada de los valores de las neuronas de las capas anteriores y los pesos asignados. Por lo tanto, si $z^L = w^L y^{L-1} + b^L$ es la suma ponderada y C la función de costo, entonces $C = C(\sigma^L(z^L))$.
Como el objetivo es minimizar la función de costo eligiendo unos parámetros y umbrales adecuados, es necesario derivar la función de costo respecto a los pesos y

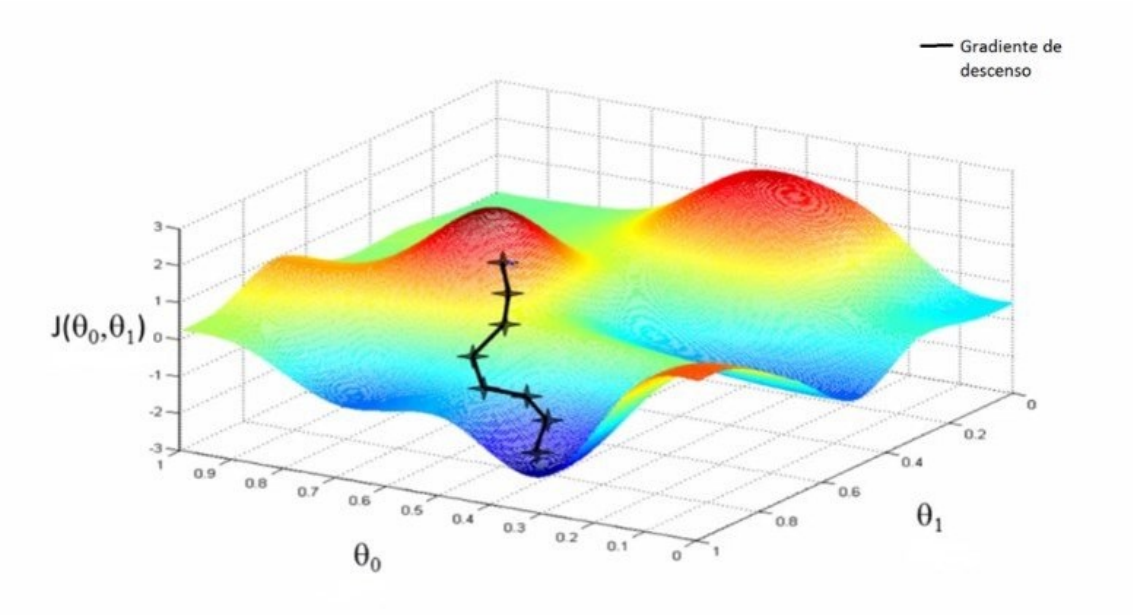

Figura 4.5: Descenso del gradiente

también derivar la función de costo respecto al umbral, es decir

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial \sigma^L}\frac{\partial \sigma^L}{\partial z^L}\frac{\partial z^L}{\partial w^L}$$
$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial \sigma^L}\frac{\partial \sigma^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{4.10}$$

Estas ecuaciones corresponden al gradiente ∇C de la función de costo.
Debido a que el gradiente es la derivada direccional de mayor crecimiento, $-\nabla C$ será la derivada direccional de menor crecimiento.
Por lo tanto en el algoritmo del descenso del gradiente se trata de actualizar los parámetro de cada capa utilizando la siguiente ecuación iterativa

$$w^L = w^L - \alpha \nabla C \tag{4.11}$$

Donde α es el ratio de aprendizaje y determina cuánto se avanza en cada iteración, es decir, cuánto se va descendiendo en el hiperplano de parámetros con el objetivo de llegar a un mínimo o un máximo global como se muestra en la figura 4.5

Capítulo 5

Predicción del exponente de hurst mediante redes neuronales artificiales

Una de las herramientas utilizadas para generar sintéticamente los datos (series temporales) del modelo mínimo de expresión génica es la librería "FBM"que está indexada en el repositorio PyPI de Python. El método "fbm"de la biblioteca "FBM"permite generar series temporales con diferentes exponentes Hurst. Al generar la serie temporal, se implementó un algoritmo que selecciona aleatoriamente los exponentes de hurst sin repetición, para crear una base de datos que simule diferentes estados de la célula, es decir, exponentes Hurst mayores de 0,5 para células sanas y menores de 0,5 para células enfermas. Los valores de cada serie temporal se ordenaron en listas y se etiquetaron con sus respectivos parámetros de Hurst para entrenar posteriormente tres redes neuronales con diferentes configuraciones. Se crearon cuatro listas de series temporales, cada lista contiene 1000 series y cada serie representa una realización experimental que consta de 500 puntos. De las 4 listas, una contiene las series temporales, la siguiente contiene la serie en un espacio de frecuencia, y en las dos listas restantes los datos se normalizaron tanto en el dominio del tiempo como en el dominio de frecuencias. Es importante señalar que cada conjunto de datos es independiente de los demás, es decir, cada uno de los cuatro conjuntos de datos se seleccionó aleatoriamente en diferentes momentos. Una vez obtenidos los datos etiquetados, se procedió a la implementación de una red neuronal densa, una convolucional (CNN) y una LSTM utilizando machine learning. Cada una de las tres redes neuronales se probó con las cuatro listas mencionadas anteriormente, por lo que hubo 12 configuraciones diferentes. La programación de los algoritmos se realizó con la ayuda de Keras versión 2.8.0. Esta interfaz de programación se ejecuta a través de TensorFlow y está enfocada en machine learning y deep learning (aprendizaje profundo), facilitando la programación de redes neuronales artificiales. Los pasos que se llevaron a cabo a la hora de entrenar cada una de las redes neuronales fueron los siguientes.

5.1. Metodología

- Separar los datos de prueba y los de entrenamiento. Tomando en cuenta la cantidad global de datos, la separación fue de un 10 % y 90 % respectivamente.
- Convertir los datos de prueba y entrenamiento en arreglos de Numpy, ya que, este es uno de los diversos objetos con los cuales Keras trabaja.
- Crear el modelo, configurar el número de neuronas, función de activación y estructura de los datos de entrada.
- Definir el número de capas de la red.
- Configurar el entrenamiento del modelo, es decir, definir el optimizador y la función de costo.
- Entrenar el modelo y realizar pruebas con diferentes épocas de entrenamiento.
- Evaluar el error entre los datos de prueba y los predichos por el modelo.
- Graficar la función de costo por cada época de los datos de prueba y entrenamiento.
- Graficar los datos predichos por el modelo contra los datos de prueba.

5.2. Resultados

La red neuronal densa y LSTM que se utilizaron cuentan con 8 neuronas en la capa de entrada, 8 neuronas en la capa intermedia y una neurona en la capa de salida. Para la red neuronal convolucional (CNN) se utilizaron 4 filtros o ventanas unidimesionales de 10 entradas que posteriormente se enlazan a una red densa de 8 neuronas en la capa intermedia y 1 neurona en la capa de salida. La configuración de las tres redes neuronales utilizadas se muestra en las siguientes tablas

Clase	Neuronas en capa de entrada	Primera capa de neuronas	Función de activación	Neuronas de la capa de salida
Dense	8	8	Relu	1
LSTM	8	8	Relu	1

Tabla 5.1

Clase	Ventanas	Filtros de entrada	Función de activación	Red enlazada	Primera Capa	Capa de Salida	Activación de primera capa	Activación de segunda capa
CNN	4	10	Relu	Dense	8	1	Relu	Sigmoid

Tabla 5.2

El número de neuronas asignadas en los algoritmos se determinó tras realizar varias simulaciones. Se observó que en un rango menor y también mayor de ocho neuronas las gráficas de función de costo vs épocas eran similares a las gráficas correspondientes a ocho neuronas. Estos gráficos fueron también el criterio para determinar si era necesario o no entrenar las redes con más épocas. No se asignaron más épocas al entrenamiento cuando los gráficos de función de costo vs épocas se mantuvieron constantes. La función de coste utilizada fue el error medio absoluto (MAE) y los resultados obtenidos se muestran en la tabla 5.3.

Tipo de red	Tiempo de computo para entrenamiento (segundos)	Error absoluto medio de los datos de prueba	Épocas	Dominio de las series
Dense	129.0	0.2	100	Temporal
Dense	226.3	0.13	100	Frecuencias
LSTM	675.9	0.061	20	Temporal
LSTM	673.1	0.026	20	Frecuencias
CNN	82.5	0.03	15	Temporal
CNN	30.9	0.35	15	Frecuencias
CNN	326.8	0.073	200	Temporal
CNN	333.1	0.38	200	Frecuencias

Tabla 5.3

Las gráficas que representan las configuraciones descritas en las tablas son las siguientes

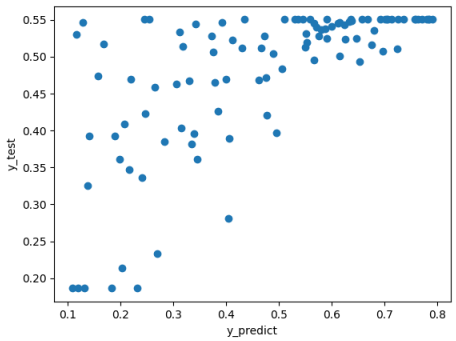

Figura 5.1: Datos predichos contra datos de prueba para una red neuronal densa.

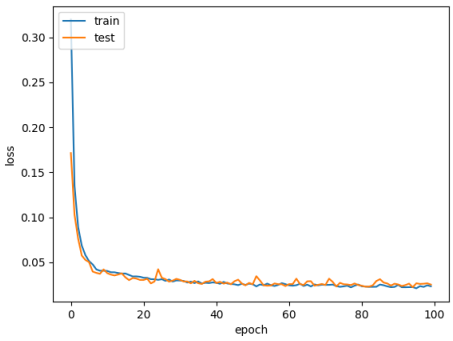

Figura 5.2: Número de épocas versus la función de coste de una red neuronal densa.

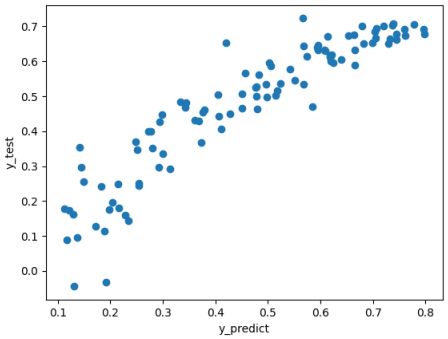

Figura 5.3: Datos predichos contra datos de prueba para una red neuronal LSTM.

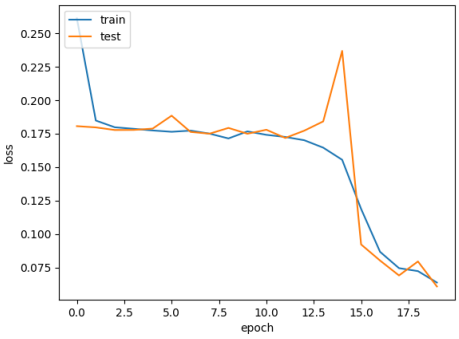

Figura 5.4: Número de épocas versus la función de coste de una red neuronal LSTM

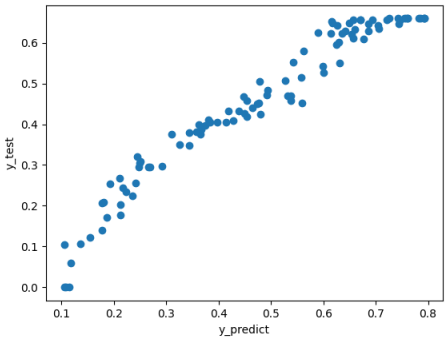

Figura 5.5: Datos predichos contra datos de prueba para una red neuronal Convolucional entrenada con 15 épocas

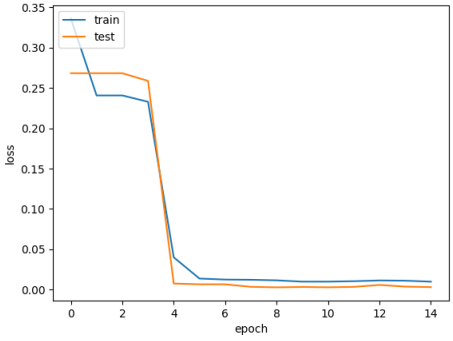

Figura 5.6: Número de épocas versus la función de coste de una red neuronal Convolucional entrenada con 15 épocas

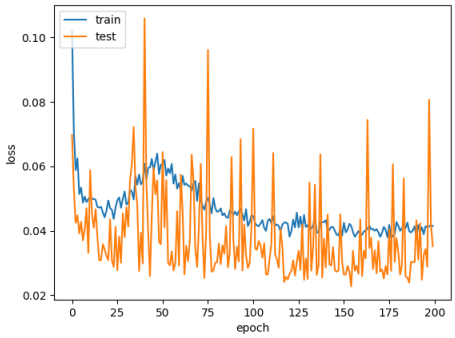

Figura 5.7: Datos predichos contra datos de prueba para una red neuronal Convolucional entrenada con 200 épocas.

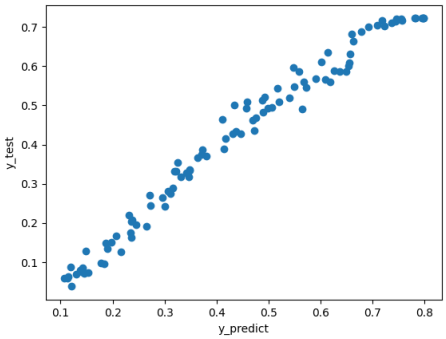

Figura 5.8: Número de épocas versus la función de coste de una red neuronal Convolucional entrenada con 200 épocas.

Conclusión

Una vez generadas las series de tiempo y las tres redes neuronales artificiales, se obtuvieron resultados que permiten presentar las siguientes conclusiones:

1. Las redes neuronales artificiales son eficientes en el análisis de las series temporales de expresión génica y en la predicción del exponente de Hurst.

2. Normalizar las series de tiempo no generó ninguna mejora en las predicciones del exponente de hurst.

3. Las predicciones que se realizaron utilizando las series de tiempo en el dominio de frecuencias sólo generaron una mejora significativa para la red neuronal artificial LSTM.

4. Teniendo en cuenta el error absoluto medio de cada una de ellas, se observa que la más eficiente es la red neuronal convolucional (CNN) cuando se entrena con las series temporales en el dominio del tiempo (ver Fig. 5.6).

5. Tomando en cuenta la rapidez de entrenamiento, los valores obtenidos en la función de costo y la representación gráfica entre datos de prueba y datos predichos . La red neuronal convolucional (CNN) parece ser la mejor candidata para futuras predicciones para las que se podría utilizar más de una capa intermedia (aprendizaje profundo). Esto puede mejorar la precisión en situaciones en las que el ruido de las mediciones puede ser un problema.

6. La red convolucional utilizada, a pesar de su simplicidad, es capaz de determinar el exponente de Hurst y la dimensión fractal de las series temporales de expresión génica. Esto abre la posibilidad de un rápido diagnóstico de enfermedades celulares para la detección temprana de enfermedades como la esteroesclerosis y el cáncer.

Apéndice A

Script en Python de Series de Tiempo de Movimiento Browniano Fractal

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
from fbm import fbm
import pickle
import random

# Numero de realizaciones (Series de Tiempo)
r = 500

# r parametros de hurst entre 0.1 y 0.8
valores_hurst = np.linspace(0.1,0.8,r)

# Devuelve una lista de longitud r de elementos unicos
    elegidos de la secuencia de la poblacion "valores_hurst".
# Se utiliza para el muestreo aleatorio sin reemplazo.
h = random.sample(list(valores_hurst), r)

# Definimos la figura que va a contener las graficas de las
    series

fig, axs = plt.subplots(r, figsize=(15, 50))

# Agregar a "lista" las series de tiempo con su respectivo H
lista = []
for j in range(0,r):
```

```
        serie_tiempo = fbm(hurst=h[j], n=500)
        tupla = (serie_tiempo, h[j])
        lista.append(tupla)
        axs[j].plot(serie_tiempo)
        axs[j].set_title('H = ' + str(h[j]))

# Guardar datos (lista)
with open("NombreSeries.pickle", "wb") as f:
    pickle.dump(lista, f)

# Graficar Series
fig.subplots_adjust(hspace = 1, wspace=.001)
fig.suptitle('FBM')
plt.show()
```

Apéndice B

Script en Python de una Red Neuronal Artificial Densa

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
import pickle
import tensorflow as tf
import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split

# Abrir datos
with open("Nombre.pickle", "rb") as f:
    lista = pickle.load(f)

# Separar datos de Entrenamiento y Prueba
train, test = train_test_split(lista, test_size = 0.1)

# Separar los datos de las etiquetas
x_train = []
y_train = []
for n in range (0, len(train)):
    a = train[n][0]
    x_train.append(a)
    b = train[n][1]
    y_train.append(b)
```

```
x_test = []
y_test = []
for n in range (0, len(test)):
    a = test[n][0]
    x_test.append(a)
    b = test[n][1]
    y_test.append(b)

# Convertimos lista de datos a np.array
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test  = np.array(x_test)
y_test  = np.array(y_test)

# Modelo Monocapa
modelo = Sequential()
modelo.add(Dense(8, activation  = 'relu', input_shape = (
   x_train.shape[1],)))
modelo.add(Dense(8, activation  = 'relu'))
modelo.add(Dense(1))
modelo.summary()

# Optimizador del modelo (configuracion de entrenamiento del
    modelo)
opt = tf.keras.optimizers.Adam(clipvalue=1.0)
modelo.compile(loss = 'mae', optimizer = opt)

# Entrenamiento
entrenamiento = modelo.fit(x_train, y_train, batch_size = 1,
    epochs = 100,
                verbose = 1, validation_data = (x_test,
                   y_test))

# Evaluar la eficiencia del modelo
calificacion = modelo.evaluate(x_test, y_test, verbose = 1)
print('Eficiencia', calificacion)

#Exponente que predice el modelo
H_predict = modelo.predict(x_test)
print('Forma de la prediccion:', H_predict.shape)
print('Valor que se predijo:', H_predict)
print('Valor de prueba:', y_test)

#Valoracion de los datos
```

```
Forma_x_train     = ['Forma del x_train', x_train.shape]
Forma_y_train     = ['Forma del y_train', y_train.shape]
Num_Datos_train   = ["Ejemplos usados para entrenar: ", len(
    train)]
Num_Datos_test    = ["Ejemplos usados para test:", len(test)]
Forma_Prediccion = ['Forma de la prediccion:', H_predict.
    shape]
Valor_Prediccion = ['Valor que se predijo:', H_predict]
Valor_Prueba      = ['Valor de prueba:', y_test]
Eficiencia        = ['Eficiencia', calificacion]

# Grafica de funcion de costo vs Numero de epocas
plt.plot(entrenamiento.history['loss'])
plt.plot(entrenamiento.history['val_loss'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Grafica y_test vs H_predict
plt.scatter(y_test, H_predict)
plt.ylabel('y_test')
plt.xlabel('y_predict')
plt.show()

# Guardar el Modelo
modelo.save(r'Direccion\NombreModelo.h5')
```

Apéndice C

Script en Python de una Red Neuronal Artificial LSTM

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
import pickle
import tensorflow as tf
import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from keras.layers import LSTM
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split

# Abrir datos
with open("Series_Frec(r=1000,n=500).pickle", "rb") as f:
    lista = pickle.load(f)

# Separar datos de Entrenamiento y Prueba
train, test = train_test_split(lista, test_size = 0.1)

# Separar los datos de las etiquetas
x_train = []
y_train = []
for n in range (0, len(train)):
    a = train[n][0]
    x_train.append(a)
    b = train[n][1]
    y_train.append(b)
```

```
x_test = []
y_test = []
for n in range (0, len(test)):
    a = test[n][0]
    x_test.append(a)
    b = test[n][1]
    y_test.append(b)

# Convertimos lista de datos a np.array
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test  = np.array(x_test)
y_test  = np.array(y_test)

# Forma y longitud de los datos de prueba y entrenamiento
print('Forma del x_train', x_train.shape)
print('Forma del y_train', y_train.shape)
print("Ejemplos usados para entrenar:", len(train))
print("Ejemplos usados para test:", len(test))

# Se modifica la forma de los arreglos x_train y x_test
x_train = x_train.reshape(x_train.shape[0], x_train.shape
    [1], 1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], 1)

# Modelo Recurrente
modelo = Sequential()
modelo.add(LSTM(8, activation = 'relu', return_sequences =
    True, input_shape = (x_train.shape[1], 1)))
modelo.add(LSTM(8, activation = 'relu'))
modelo.add(Dense(1))
modelo.summary()

# Optimizador del modelo (configuracion de entrenamiento del
    modelo)
opt = tf.keras.optimizers.Adam(clipvalue=1.0)
modelo.compile(loss = 'mae', optimizer = opt)

# Entrenamiento
entrenamiento = modelo.fit(x_train, y_train, batch_size = 7,
    epochs = 20, verbose = 1, validation_data = (x_test,
    y_test))

# Evaluar la eficiencia del modelo
```

```
calificacion = modelo.evaluate(x_test, y_test, verbose = 1)
print('Eficiencia', calificacion)

# Exponente que predice el modelo
H_predict = modelo.predict(x_test)
print('Forma de la prediccion:', H_predict.shape)
print('Valor que se predijo:', H_predict)
print('Valor de prueba:', y_test)

# Grafica funcion de costo vs numero de epocas
plt.figure(1)
plt.plot(entrenamiento.history['loss'])
plt.plot(entrenamiento.history['val_loss'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Grafica y_test vs H_predict
plt.figure(2)
plt.scatter(y_test, H_predict)
plt.ylabel('y_test')
plt.xlabel('y_predict')
plt.show()

# Guardar el Modelo
modelo.save(r'Direccion\NombreModelo.h5')
```

Apéndice D

Script en Python de una Red Neuronal Convolucional

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
import pickle
import tensorflow as tf
import keras
from keras import layers
from keras import Sequential
from keras.layers import MaxPooling1D, Dropout, Dense,
    Flatten
from keras.layers import Convolution1D as Conv1D
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam

# Abrir datos
with open("Series_Frec(r=1000,n=500).pickle", "rb") as f:
    lista = pickle.load(f)

# Separar datos de Entrenamiento y Prueba
train, test = train_test_split(lista, test_size = 0.1)

# Separar los datos de las etiquetas
x_train = []
y_train = []
for n in range (0, len(train)):
    a = train[n][0]
    x_train.append(a)
    b = train[n][1]
```

```
    y_train.append(b)

x_test = []
y_test = []
for n in range (0, len(test)):
    a = test[n][0]
    x_test.append(a)
    b = test[n][1]
    y_test.append(b)

# Convertimos lista de datos a np.array
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test  = np.array(x_test)
y_test  = np.array(y_test)

# Forma y longitud de los datos de prueba y entrenamiento
print('Forma del x_train', x_train.shape)
print('Forma del y_train', y_train.shape)
print("Ejemplos usados para entrenar:", len(train))
print("Ejemplos usados para test:", len(test))

# Modelo CNN
modelo = Sequential()
modelo.add(Conv1D(4, 10, activation = 'relu', input_shape =
   (x_train.shape[1],1)))
modelo.add(MaxPooling1D(pool_size = 1))

# Capa que une la CNN a la red Densa
modelo.add(Flatten())

# Modelo Denso
modelo.add(Dense(8, activation = 'relu'))
modelo.add(Dense(1, activation = 'sigmoid'))
modelo.summary()

# Optimizador del modelo (configuracion de entrenamiento del
    modelo)
opt = tf.keras.optimizers.Adam(clipvalue=1.0)
modelo.compile(loss = 'mae', optimizer = opt)

# Entrenamiento
entrenamiento = modelo.fit(x_train, y_train, batch_size = 1,
    epochs = 200, verbose = 1, validation_data = (x_test,
```

```
    y_test))

# Evaluar la eficiencia del modelo
calificacion = modelo.evaluate(x_test, y_test, verbose = 1)
print('Eficiencia', calificacion)

# Exponente que predice el modelo
H_predict = modelo.predict(x_test)
print('Forma de la prediccion:', H_predict.shape)
for i in range (len(H_predict)):
    print(H_predict[i],'---->',y_test[i])

# Grafica funcion de costo
plt.figure(1)
plt.plot(entrenamiento.history['loss'])
plt.plot(entrenamiento.history['val_loss'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Grafica y_test vs H_predict
plt.figure(2)
plt.scatter(y_test, H_predict)
plt.ylabel('y_test')
plt.xlabel('y_predict')
plt.show()

# Valoracion de los datos
Forma_x_train    = ['Forma del x_train', x_train.shape]
Forma_y_train    = ['Forma del y_train', y_train.shape]
Num_Datos_train  = ['Ejemplos usados para entrenar',
                    len(train)]
Num_Datos_test   = ['Ejemplos usados para test', len(test)]
Forma_Prediccion = ['Forma de la prediccion:', H_predict.
    shape]
Valor_Prediccion = ['Valor que se predijo:', H_predict]
Valor_Prueba     = ['Valor de prueba:', y_test]
Eficiencia       = ['Eficiencia', calificacion]

# Guardar el Modelo
modelo.save(r'Direccion\NombreModelo.h5')
```

Apéndice E

Script en Python para probar las Redes Neuronales Artificiales entrenadas

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
import pickle
import tensorflow as tf
import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.optimizers import RMSprop, SGD, Adam
from sklearn.model_selection import train_test_split

# Abrir datos
with open("NombreSeries", "rb") as f:
    lista = pickle.load(f)

# Separar datos de Entrenamiento y Prueba
train, test = train_test_split(lista, test_size = 0.005)

# Separar los datos de las etiquetas
x_test = []
y_test = []
for n in range (0, len(test)):
    a = test[n][0]
    x_test.append(a)
    b = test[n][1]
```

```
    y_test.append(b)

# Convertimos lista de datos a np.array
x_test  = np.array(x_test)
y_test  = np.array(y_test)

# Se modifica la forma del arreglo x_test si la red es una
   LSTM
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], 1)

# Carga el modelo entrenado
new_model = keras.models.load_model(r'Direccion\NombreModelo
   .h5')

# Compara los valores predichos por la red con los datos de
   prueba
new_predictions = new_model.predict(x_test)
print('Forma de la prediccion:', new_predictions.shape)
print('')
print('valor que se predijo','-->','valor de prueba')
print('')
for i in range (len(new_predictions)):
    print(new_predictions[i],'-->',y_test[i])
```

Bibliografía

[1] Uri Alon. *An introduction to systems biology: design principles of biological circuits*. Second edition. Boca Raton, Fla: CRC Press, 2019. ISBN: 978-1-4398-3717-7 978-1-138-49011-6.

[2] Mahboobeh Ghorbani, Edmond A. Jonckheere y Paul Bogdan. "Gene Expression Is Not Random: Scaling, Long-Range Cross-Dependence, and Fractal Characteristics of Gene Regulatory Networks". En: *Frontiers in Physiology* 9 (oct. de 2018), pág. 1446. ISSN: 1664-042X. DOI: 10.3389/fphys.2018.01446. URL: https://www.frontiersin.org/article/10.3389/fphys.2018.01446/full (visitado 27-09-2021).

[3] *Serie temporal*. es. Page Version ID: 131528954. Dic. de 2020. URL: https://es.wikipedia.org/w/index.php?title=Serie_temporal&oldid=131528954 (visitado 28-09-2021).

[4] Lyudmyla Kirichenko, V. Bulakh y T. Radivilova. "Machine learning classification of multifractional Brownian motion realizations". En: *CMIS*. 2020.

[5] Ming Li. "Fractal Time Series—A Tutorial Review". en. En: *Mathematical Problems in Engineering* 2010 (2010), págs. 1-26. ISSN: 1024-123X, 1563-5147. DOI: 10.1155/2010/157264. URL: http://www.hindawi.com/journals/mpe/2010/157264/ (visitado 27-09-2021).

[6] Vern Paxson. "Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic". en. En: *ACM SIGCOMM Computer Communication Review* 27.5 (oct. de 1997), págs. 5-18. ISSN: 0146-4833. DOI: 10.1145/269790.269792. URL: https://dl.acm.org/doi/10.1145/269790.269792 (visitado 27-09-2021).

[7] Karl J. Åström y Richard M. Murray. *Feedback systems: an introduction for scientists and engineers*. OCLC: ocn183179623. Princeton: Princeton University Press, 2008. ISBN: 978-0-691-13576-2.

[8] Weilin Xiao, Weiguo Zhang y Weidong Xu. "Parameter estimation for fractional Ornstein–Uhlenbeck processes at discrete observation". en. En: *Applied Mathematical Modelling* 35.9 (sep. de 2011), págs. 4196-4207. ISSN: 0307904X. DOI: 10.1016/j.apm.2011.02.047. URL: https://linkinghub.elsevier.com/retrieve/pii/S0307904X11001181 (visitado 27-09-2021).

[9] Leonardo Plazas Nossa, Miguel Antonio Ávila Angulo y Germán Moncada Méndez. "Estimación del exponente de HURST y dimensión fractal para el análisis de series de tiempo de absorbancia UV-VIS". En: *Ciencia e Ingeniería Neogranadina* 24.2 (dic. de 2014), pág. 133. ISSN: 1909-7735, 0124-8170. DOI: 10.18359/rcin.397. URL: http://revistas.unimilitar.edu.co/index.php/rcin/article/view/397 (visitado 27-09-2021).

[10] Theodor Bruu Røine y Edvard Kvalø Holter. "Properties of the gold price: an investigation using fractional Brownian motion and supervised machine learning techniqes". Tesis de mtría. 2018.