LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

PREDICCIÓN DE SISTEMAS DINÁMICOS CON REDES NEURONALES PROFUNDAS

Daniel Gerardo Maino

Dr. L. C. Uzal
Director

Dr. P. M. Granitto
Co-director

Septiembre de 2013

Grupo de Aprendizaje Automatizado y Aplicaciones
CIFASIS (CONICET-UNR-UPCAM)
Centro Internacional Franco-Argentino de Ciencias de la Información y de Sistemas

Facultad de Ciencias Exactas, Ingeniera y Agrimensura
Universidad Nacional de Rosario
Argentina

Índice de contenidos

Índice de figuras

Resumen

Existe una diversidad de series temporales que son objeto de estudio en múltiples disciplinas, por ejemplo en la meteorología, la geofísica, la biología, la medicina y la sociología. En esta Tesina se aborda el problema de predicción de series temporales caracterizadas por su naturaleza determinística no-lineal. Se presenta una técnica basada en redes neuronales profundas para la predicción de sistemas dinámicos a partir de una serie temporal. Se sabe que las arquitecturas profundas pueden ser mucho más eficientes a la hora de representar ciertas funciones. Por otro lado, recientemente se han publicado trabajos en los que se encuentra evidencia del beneficio en construir un modelo con salida-múltiple, de manera que este aprenda y preserve las dependencias entre los valores de la predicción. Se evalúa el rendimiento de arquitecturas profundas frente a las redes neuronales convencionales y a su vez el uso de salida-múltiple frente a las redes de salida-simple, en un modelo de predicción para múltiples horizontes. Los resultados muestran un mejor desempeño de las arquitecturas profundas sobre las series temporales consideradas.

Capítulo 1

Introducción

En esta Tesina se presenta una técnica basada en redes neuronales profundas para la predicción de valores futuros de una serie temporal observada. Tales series estarán caracterizadas por su naturaleza determinística y *no-lineal*. Se evalúa el rendimiento de tales arquitecturas profundas frente a las redes neuronales convencionales.

A modo de introducción, y para finalmente poder definir más precisamente el problema abordado en esta Tesina, se presenta en este capítulo una breve introducción a las series temporales, a los sistemas dinámicos y a las redes neuronales profundas.

1.1. Series Temporales

Se denomina serie temporal a un conjunto de mediciones realizadas sobre un sistema a lo largo del tiempo. Son el resultado de un experimento o una observación de fenómenos naturales espontáneos que evolucionan en el tiempo. Existe una gran diversidad de series temporales que son objeto de estudio en múltiples disciplinas, por ejemplo en la meteorología, la geofísica, la biología, la medicina y la sociología .

Existen diversas clases de series temporales que pueden catalogarse según su grado de *no-linealidad* y el nivel de estocasticidad o aleatoriedad. En el extremo lineal y determinista se encuentran las series que son simples oscilaciones. Cuando las series se apartan de este comportamiento simple, existen dos paradigmas para explicar el comportamiento irregular (no periódico). Por un lado, los modelos lineales estocásticos suponen que esta irregularidad es debida a la incidencia de una fuente de ruido que perturba o que es parte de la dinámica que genera la serie. En el otro extremo, el paradigma introducido por la teoría del caos es que sistemas *no-lineales* completamente deterministas son capaces de producir un comportamiento irregular. El concepto de determinismo implica una evolución del estado del sistema regida por leyes. Los métodos de reconstrucción de atractores y de predicción expuestos en esta tesina se ubican en la región *no-lineal* y determinista.

En la siguiente sección se presenta la definición de sistema dinámico, luego en las secciones siguientes algunos conceptos necesarios para el análisis de este tipo de series temporales.

1.2. Breve introducción a los sistemas dinámicos

Los sistemas dinámicos son sistemas cuyos parámetros internos (variables de estado) siguen una serie de reglas para su evolución temporal. La evolución temporal de estas variables de estado está determinada completamente por estas reglas. Para ciertos sistemas reales es posible definir un conjunto de variables y escribir un sistema de ecuaciones de modo de obtener un modelo matemático del sistema real. Un sistema queda determinado unívocamente por un conjunto de D variables que serán las componentes de un vector s en $\mathbb{R}^D$. La transición desde un estado $s(t_0)$ en el tiempo t_0 a un estado en el tiempo (t_0+h) queda completamente determinada al aplicar la función f_h, es decir:

$$s(t_0 + h) = f_h[s(t_0)]. \tag{1.1}$$

Los sistemas dinámicos pueden dividirse en dos grandes clases: aquellos en los que el tiempo varía continuamente y en los que el tiempo transcurre discretamente. Los sistemas dinámicos de tiempo continuo se expresan con ecuaciones diferenciales; éstas pueden ser ecuaciones diferenciales ordinarias (ODEs):

$$\dot{s}(h) = f_{ode}[s(h)]. \tag{1.2}$$

Por otra parte, si el tiempo es discreto los sistemas se describen por medio de ecuaciones de diferencias (DEs), también conocidas como mapas iterados. En tal caso el estado del sistema evoluciona en un tiempo discreto $t = n\triangle t$ de manera que la Ec. (1.1) se reduce a

$$s_{n+1} = f_{map}(s_n). \tag{1.3}$$

Por ejemplo, los sistemas dinámicos más comunes en electrónica son ODEs en el caso de electrónica analógica y DEs en electrónica digital. También se utilizan en distintas áreas de la ciencia como la meteorología, la ecología, la astronomía o la epidemiología, biomecánica deportiva, entre otras.

Como ejemplo consideremos el sistema no lineal

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -\sin x. \end{cases} \tag{1.4}$$

Se trata de un péndulo simple, el cual tiene un conjunto numerable de puntos de

equilibrio sobre el eje x. Aquí la función $\frac{1}{2}y^2 + (1 - \cos x)$ es constante. Dibujando sus curvas de nivel que describen las soluciones, vemos que los puntos de equilibrio son alternadamente sillas y centros. Estos últimos se llaman así por estar rodeados de una región abierta que contiene solo órbitas periódicas. Ver la Fig. 1.1.

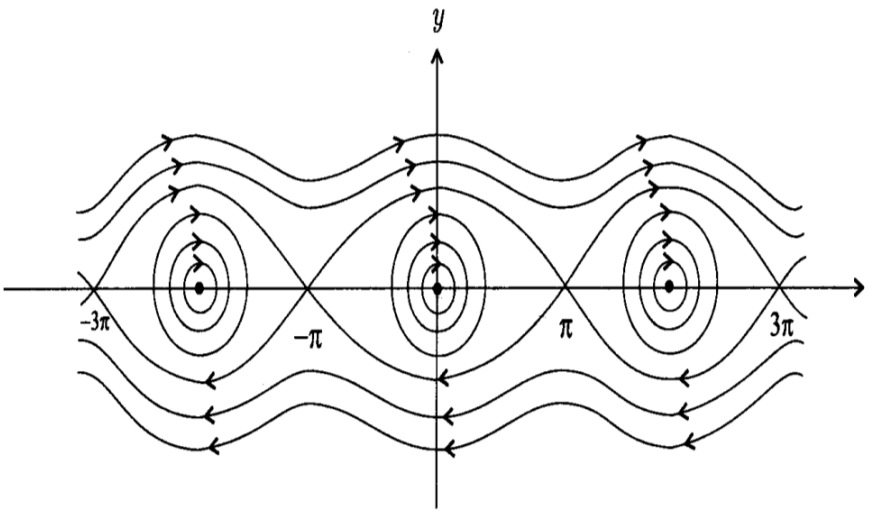

Figura 1.1: Retrato Fase del péndulo

La forma de visualizar el comportamiento de las variables de estado de un sistema dinámico puede ser en forma de serie temporal (gráfica de una variable de estado contra el tiempo), o en forma de espacio de fases. El espacio fases de un sistema n-dimensional (Ec. 1.2) es el espacio donde todos los posibles estados de un sistema son representados, cada variable del sistema se representa como un eje de un espacio multidimensional y cada punto del espacio representa cada posible estado de las variables del sistema. En este tipo de representación el tiempo se vuelve un parámetro implícito.

Un atractor es el conjunto al que el sistema evoluciona después de un tiempo suficientemente largo. Para que el conjunto sea un atractor, las trayectorias que le sean suficientemente próximas han de permanecer próximas incluso si son ligeramente perturbadas. Geométricamente, un atractor puede ser un punto, una curva, una variedad o incluso un conjunto complicado de estructura fractal conocido como atractor extraño. La descripción de atractores de sistemas dinámicos caóticos ha sido uno de los grandes logros de la teoría del caos. En esta Tesina se considerarán sistemas dinámicos que evolucionan sobre atractores extraños.

1.3. Reconstrucción del espacio de fases a partir de una serie temporal

Existe la posibilidad de analizar algunos sistemas dinámicos a partir de series temporales medidas sobre estos sistemas. En determinados casos será posible recuperar el atractor (o al menos un objeto equivalente) a partir de una serie temporal de mediciones. Una forma simple de obtener una reconstrucción del espacio de fases es mediante coordenadas de retraso [1].

1.3.1. Coordenadas de retraso

Cuando se estudia un sistema dinámico, una de sus limitaciones es la imposibilidad de medir todas las variables de estado. En la mayoría de los casos se tiene una serie temporal de registros de un único observable x. Sin embargo, de una secuencia de valores de x dentro de cierta ventana temporal de ancho t_w pueden distinguirse los distintos estados del sistema y la relación causal entre ellos. Definimos el vector de reconstrucción y para un tiempo t como la reconstrucción de las observaciones dentro de la ventana temporal $[t - t_w, t]$. Una forma simple de obtener $y(t)$ es considerando directamente el vector $\bar{x}(t)$ de coordenadas de retraso y que está compuesto por m observaciones equiespaciadas en el tiempo, es decir:

$$\bar{x}(t) = [x(t), x(t-\tau), x(t-2\tau), ..., x(t-(m-1)\tau)] \tag{1.5}$$

donde τ es el tiempo entre observaciones sucesivas, llamado tiempo de retraso (*delay time*). A partir de esta construcción pueden recuperarse órbitas en un espacio de m dimensiones que estarán restringidas a un atractor en este nuevo espacio, el cual no será igual al atractor *original* pero, bajo ciertas condiciones, conservará las propiedades geométricas [1].

El problema de reconstrucción del atractor consiste en encontrar la mejor representación para el atractor construido a partir de la serie temporal de observaciones. Los requisitos básicos que debe cumplir una reconstrucción serán:

- que la dinámica en el espacio de fases propuesto sea determinista y por ende no haya cruce de órbitas. Esto está garantizado por el teorema de Mañé-Takens [1, 2] —generalizado luego por Sauer *et al.*[3]— cuando la dimensión de embeding m es mayor que dos veces la dimensión fractal del atractor

- que el atractor reconstruido conserve las propiedades geométricas del atractor original pudiéndose por lo tanto calcular consistentemente a partir del mismo cantidades como la dimensión de correlación y los exponentes de Lyapunov.

Una serie temporal real será una secuencia limitada de observaciones, estas observaciones tienen asociado una cantidad de ruido observacional. El efecto de estos factores sobre la calidad de la reconstrucción fue estudiado en forma rigurosa por Casdagli *et al* [4]. A partir de modelar presencia de ruido en el proceso de reconstrucción, los autores distinguen efectos que limitan la elección del tiempo de retraso τ, o más precisamente, del ancho de ventana t_w. En esta Tesina se usará la metodología descrita en [5, 6] que tiene en cuenta estos factores para calcular el tamaño óptimo de la ventana t_w.

1.3.2. Dimensión de Correlación

Los sistemas dinámicos caóticos generan típicamente atractores extraños con dimensiones no enteras. Estas dimensiones fraccionarias son asignadas a objetos que presentan algún tipo de autosimilaridad, presentando estructuras recurrentes a diferentes escalas. Una medida de la dimensión que ha cobrado particular interés debido a su aplicabilidad en casos donde la cantidad de datos es limitada [7] es la *dimensión de correlación* introducida por Gassberger y Procaccia [8, 9].

Integral de correlación

Definimos la *suma de correlación* para una colección de puntos $\{x_1, x_2, ..., x_n\}$ como la proporción de todos los posibles pares de puntos que se encuentran a una distancia menor que ϵ. Una definición básica [7] para C es:

$$C(\epsilon) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\|), \tag{1.6}$$

donde Θ es la función de Heaviside:

$$\Theta(x) = \begin{cases} 0 & cuando \quad x \leq 0 \\ 1 & cuando \quad x > 0 \end{cases} \tag{1.7}$$

La suma es sobre los $N(N-1)/2$ pares de puntos del atractor. En el límite de la infinita cantidad de datos ($N \to \infty$) y para $\epsilon \to 0$ se espera encontrar una ley de potencias $C(\epsilon) \propto \epsilon^D$ de manera que la *dimensión de correlación* queda definida por

$$D = \lim_{\epsilon \to 0} \lim_{N \to \infty} \frac{\partial \ln C(\epsilon, N)}{\partial \ln \epsilon}. \tag{1.8}$$

Una dificultad relacionada con la cantidad finita de datos para el cálculo de la dimensión de correlación es que no resulta posible considerar el límite $\epsilon \to 0$ en la Ec. (1.6) ya que la ley $C(\epsilon) \propto \epsilon^D$ solo se observará en un rango de valores de ϵ. La finitud en la cantidad de datos restringe la distancia entre pares arbitrariamente cercanos

imponiendo una cota inferior para ϵ al restringir la distancia entre pares arbitrariamente cercanos.

En una gráfica de $D(\epsilon)$ vs. ϵ (como en las figuras 3.5, 3.12, 3.20 de esta Tesina) se espera observar una zona de *meseta* (también llamado *plateau*) en este rango, en la cual es posible determinar una estimación robusta de la *dimensión de correlación*.

1.4. Predicción de valores futuros de una serie temporal

La predicción de series temporales es un campo de interés creciente que juega un papel importante en casi todos los campos de la ciencia y de la ingeniería, tales como la economía, finanzas, meteorología y telecomunicaciones [10].

La predicción de valores futuros de la serie temporal se realiza basándose en valores previos y el valor actual de la serie temporal, tales valores son usados como entrada en el modelo de predicción:

$$\hat{x}_{t+h} = f_h(x_t, ..., x_{t-t_w}) + \epsilon_h \tag{1.9}$$

donde $\hat{x}_{t+h}$ es la predicción h pasos hacia adelante con respecto al tiempo t, f_h es la función que modela las dependencias entre las observaciones pasadas y futuras, $t_w + 1 = m$ es el tamaño de entrada de la función f, ϵ_h representa el error del modelado.

Predicción sobre espacio de fases

Si se dispone de una reconstrucción del espacio de fases (Sec. 1.3), es posible realizar predicción de valores futuros sobre éste y obtener una observación futura de la serie.

Al realizar una reconstrucción de coordenadas de retraso (Sec. 1.3.1), obtenemos el vector $\bar{x}$ (Ec. 1.5). Luego, obtenida la reconstrucción, se construye un modelo $F_h : \mathbb{R}^n \to \mathbb{R}$, de predicción ($h$ pasos hacia adelante) sobre el espacio de fases. El valor $\hat{x}(t+h)$ obtenido corresponde a una única observación futura de la serie temporal:

$$\hat{x}(t+h) = F_h(\bar{x}(t)) \tag{1.10}$$

1.4.1. Predicción a corto y largo plazo

Uno de los desafíos en la predicción de series temporales es la predicción a *largo plazo*, la cual es en general más compleja en comparación a la predicción a *corto plazo*.

Por ejemplo, en el área de la meteorología, predecir la temperatura ambiente en la próxima hora es un problema distinto de predecir la temperatura para dentro de una semana. Las expectativas sobre el nivel de error y la precisión para las predicciones son completamente distintas.

El término *corto* o *largo plazo*, no tiene una única definición, esto depende de la aplicación y de nuestros intereses sobre la predicción.

Para el caso especial de las series consideradas en esta tesina, el carácter caótico de la serie conlleva a una divergencia exponencial en la evolución de estados vecinos y por ende una complejización de la ley que determina su estado para horizontes H crecientes. Para las series caóticas se puede decir que *largo plazo* corresponde a horizontes cercanos al límite de impredecibilidad de la serie. En estos rangos la función de predicción f_H está dominada por términos *no-lineales*.

Predicción a paso múltiple

Por lo general, cuando una aplicación práctica requiere una predicción de largo plazo de una serie temporal, suele ser necesario predecir, además de x_{t+H}, toda la secuencia de valores entre x_t e x_{t+H}. En otras palabras, la predicción a un *paso múltiple* hacia adelante de una serie temporal $\{x_1, x_2, ..., x_N\}$ compuesta por N observaciones, consiste en predecir los próximos H valores $\{x_{N+1}, x_{N+2}, ..., x_{N+H}\}$, donde $H > 1$ denota el *horizonte de predicción*.

Un ejemplo de aplicación donde se requiere predecir a pasos múltiples proviene del estudio de la física solar. La serie temporal de manchas solares [11] refleja el nivel de actividad en la superficie del sol y presenta un comportamiento irregular con máximos cada 11 años aproximádamente. Resulta de interés científico y tecnológico predecir cuándo va a ocurrir el próximo máximo y cuál va a ser su amplitud. Dado que no se conoce *a priori* cuándo va a ocurrir el próximo máximo, no alcanza con hacer una predicción a un horizonte fijo. Es necesario en cambio predecir toda la secuencia dentro de la cual esperamos que ocurra el máximo y a partir de allí extraer información sobre su tiempo de ocurrencia y amplitud.

1.4.2. Predicción iterativa vs. predicción directa

Existen dos métodos para la predicción en un horizonte H lejano: el *iterativo (o recursivo)* y el *directo*.

Método Recursivo o Iterativo

En este método, una predicción con horizonte H es llevada a cabo por iterar H veces un predictor de *paso simple* (one-step-ahead). Una vez que el predictor ha estimado el valor $\hat{x}_{N+1}$, éste es reinyectado como parte de la entrada de tamaño m, para obtener la siguiente predicción, así hasta realizar H iteraciones de un *paso simple*

$$x_{t+1} = f_1(x_t, x_{t-1}, ..., x_{t-m+1}) \tag{1.11}$$

donde $t \in \{m, ..., N-1\}$. Luego de tener el modelo entrenado $\hat{f}_1$, la predicción queda definida de la siguiente manera:

$$\hat{x}_{N+h} = \begin{cases} \hat{f}_1(x_N, ..., x_{N-m+1}) + \epsilon_h & si \;\; h = 1 \\ \hat{f}_1(\hat{x}_{N+h-1}, ..., \hat{x}_{N+1}, x_N, ..., x_{N-m+h}) + \epsilon_h & si \;\; h \in \{2, ..., m\} \\ \hat{f}_1(\hat{x}_{N+h-1}, ..., \hat{x}_{N+h-m}) + \epsilon_h & si \;\; h \in \{m+1, ..., H\} \end{cases} \tag{1.12}$$

donde $\hat{x}_{N+h}$ es la predicción h pasos hacia adelante con respecto al último valor de la serie (N), f_1 es la función que modela las dependencias entre la observaciones pasadas y la futura, m dimensión de la entrada del modelo, N es la cantidad de datos disponibles de la serie, ϵ_h representa el error del modelado.

Método Directo

Predecir a un horizonte H, es ajustar *directamente* un modelo para predecir este valor. Esta estrategia toma como entrada solamente valores de la serie temporal, sin tomar valores de predicción.

$$x_{t+h} = f_h(x_t, x_{t-1}, ..., x_{t-m+1}) \;\; con \;\; 1 \leqslant h \leqslant H \tag{1.13}$$

donde $t \in \{m, ..., N-H\}$ y $h \in \{1, ..., H\}$. Una vez entrenados los modelos (uno por cada horizonte h) $\hat{f}_h$, obtenemos la predicción como sigue:

$$\hat{x}_{N+h} = \hat{f}_h(x_N, x_{N-1}, ..., x_{N-m+1}) + \epsilon_h \;\; con \;\; 1 \leqslant h \leqslant H. \tag{1.14}$$

Los errores que se cometen en realizar las próximas predicciones no son acumulables porque solo se usan como entrada datos de la serie temporal. Cuando se desean predecir todos los valores de $\hat{x}_{N+1}$ a $\hat{x}_{N+H}$, se deben estimar H modelos diferentes. Dado el carácter caótico de las series temporales, el modelo requiere mayor complejidad que la *estrategia recursiva* (*un paso simple*). Según el estudio teórico y numérico desarrollado en [12] los métodos directos (en particular, modelos polinómicos locales) presentan mayor error que sus equivalentes métodos iterativos.

1.5. Redes neuronales

Las Redes Neuronales Artificiales o simplemente Redes Neuronales (RN) surgieron de la idea de modelar matemáticamente habilidades intelectuales humanas mediante diseños ingenieriles biológicamente plausibles. Intentando ser esquemas de computación en paralelo que imitan el cerebro humano, las RN evolucionaron en herramientas valiosas para clasificación y regresión con una gran influencia teórica y práctica en el

aprendizaje automatizado. El modelo más popular de RN es el de perceptrones multicapas que se muestra en la Figura 1.2. Una RN consta de un conjunto de unidades de procesamiento elementales, llamadas usualmente neuronas, con un esquema de conexión adecuado entre las mismas.

1.5.1. Redes neuronales convencionales

Neuronas

Una neurona es una unidad de cálculo que toma como entradas las variables (*features*) del problema o las salidas de otras neuronas, realiza un promedio ponderado de los mismos y transforma este promedio en su propia salida utilizando una función de activación.

Formalmente, llamamos $u = [u_0, u_1, ..., u_q] \in \mathbb{R}^{q+1}$ el vector de entrada a la neurona y $v \in \mathbb{R}$ a su salida. Llamamos $w = [w_0, ..., w_q]^T \in \mathbb{R}^{q+1}$ al vector de pesos sinápticos que se utilizan para realizar el promedio ponderado. Estos pesos son los parámetros ajustables del modelo. El elemento de procesamiento implementa la función:

$$v = \phi(\xi) \qquad \xi = \sum_{i=0}^{q} w_i u_i \tag{1.15}$$

donde $\phi : \mathbb{R} \to \mathbb{R}$ es la función de activación y ξ es el promedio pesado.

Las funciones típicas de activación son:

Umbral:

$$\phi(\xi) = \begin{cases} 1 & si \;\; \xi \geq 0 \\ 0 & si \;\; no \end{cases} \tag{1.16}$$

Sigmóidea:

$$\phi(\xi) = \frac{1}{1 + \exp(-\xi)} \tag{1.17}$$

Identidad:

$$\phi(\xi) = \xi \tag{1.18}$$

La función *sigmóidea* es la más utilizada debido a que es derivable y puede modelar tanto funciones *lineales* como *no-lineales* con la precisión deseada, ϕ es casi lineal en el origen, mientras que para pesos grandes ϕ es altamente *no-lineal*.
El peso w_0 es utilizado como *bias*, para corregir el valor medio de los inputs. El valor de entrada correspondiente u_0 se fija a una constante.

Recientemente se han aplicado otro tipo de funciones de activación llamadas *Unidades Lineales Rectificadas* (*Rectified Linear Units, ReLUs*) [13, 14] cuya función de activación es $f(x) = max(0, x)$ ó, en su versión diferenciable:

NReLUs:

$$\phi(\xi) = \log(1 + \exp(\xi)). \tag{1.19}$$

Utilizando este tipo de unidades, se reducen los tiempos de entrenamiento con descenso por gradiente, respecto de una red neuronal con unidades de activación *sigmoidea* (Ec. 1.17). El tipo de saturación de estas últimas no favorece un rápido descenso por gradiente. En consecuencia el entrenamiento de redes neuronales profundas con *ReLUs* es aproximadamente 6 veces más rápido que entrenar una red utilizando unidades *sigmoideas* [14]. La utilización de unidades tipo *ReLU* es uno de los elementos clave que ha permitido entrenar en forma supervisada redes profundas aún sin necesidad de recurrir a un pre-entrenamiento no supervisado por capas [14, 15] (ver Sec. 1.5.4).

Modelo Perceprones Multicapas

Conectando neuronas en capas podemos diseñar una RN llamada de perceptrones multicapa. Una capa es una serie de neuronas que no tienen conexiones entre sí y que tienen sus entradas conectadas a la misma serie de inputs. Conectando varias capas se crea una estructura unidireccional (*feedforward*) donde la salida de la capa de entrada y todas las capas intermedias es enviada sólo a las capas superiores. La primer capa o capa de entrada se conecta al vector de variables (*features*) del problema. La capa de entrada utiliza la función identidad como activación, mientras que en cada capa intermedia (o de salida) se utiliza una misma función de activación, usualmente *sigmoidea* o *lineal*.

1.5.2. Salida-Múltiple vs. Salida-Simple

Las redes de *salida-múltiple* permiten predecir simultáneamente múltiples horizontes. Siendo s la cantidad de neuronas de salida, cada una de estas estará especializada en predecir un horizonte distinto.

Recientemente se han publicado trabajos en los que se encuentra evidencia del beneficio en construir un modelo con *salida-múltiple*, de manera que éste aprenda y preserve las dependencias entre los valores de la predicción [16]. Los resultados mencionados corresponden al modelado de series temporales estocásticas. En esta tesina compararemos las dos estrategias, *salida-múltiple* y *salida-simple*, para las series temporales caóticas (deterministas) en estudio. Se quiere evaluar si el entrenamiento minimizando el error sobre las múltiples salidas tenga un efecto regularizador que pueda otorgarle a la red un mejor error de generalización sobre el horizonte más lejano.

Back Propagation

A continuación se describe en forma muy sintetizada el método de *retro-propagación* de errores (*back propagation*). Éste es un método para buscar un mínimo del error de

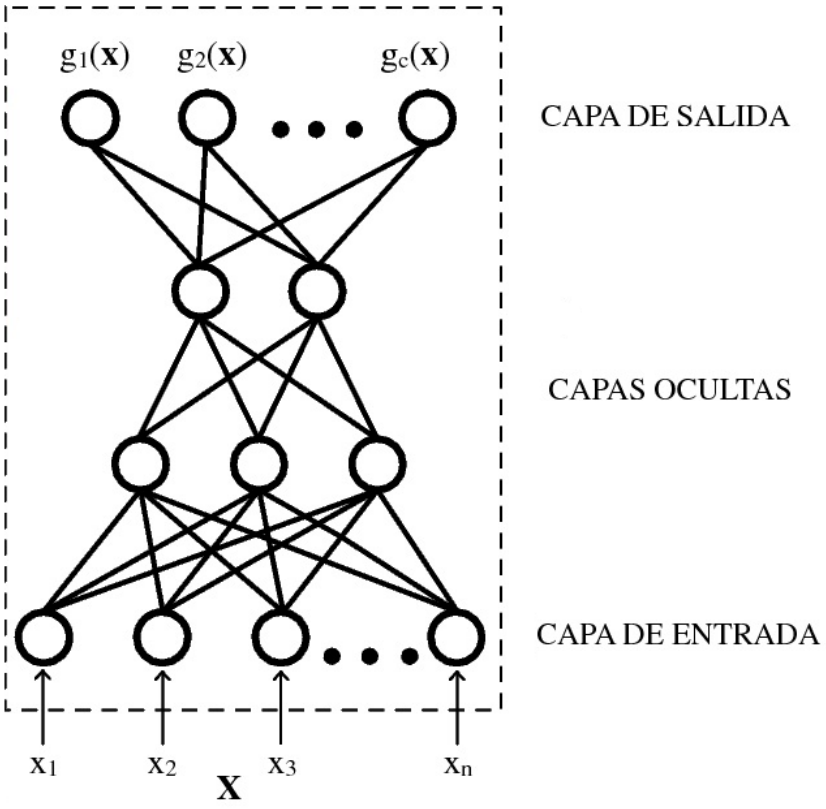

Figura 1.2: Red Neuronal Convencional.

la red basado en el método del gradiente, es decir, es un método de primer orden que sólo utiliza la primer derivada de la función a minimizar.

Sea dada la estructura de la RN y una función de activación diferenciable ϕ. El problema es determinar el valor de los pesos para todos los nodos que minimice una función de error $J(\theta)$. Dado θ, un parámetro de la RN, el método del gradiente lo actualiza con:

$$\theta \leftarrow \theta - \eta \frac{\partial J}{\partial \theta} \tag{1.20}$$

donde el parámetro *learning rate* $\eta > 0$. Si utilizamos como $J(\theta)$ a la función:

$$E = \frac{1}{2} \sum_{d \in D} \left(y_d - g(x) \right)^2 \tag{1.21}$$

donde D es el conjunto de patrones de entrenamiento y si tomamos la función de activación ϕ *sigmóidea*, se obtiene que el error del k-ésimo nodo en una capa oculta h es:

$$\delta_k^h = (\sum_{i=1}^{c} \delta_i^0 \ w_{i,k}^0) \ v_k^h \ (1 - v_k^h) \tag{1.22}$$

para los cuales debemos de tener calculados los errores de la capa superior. Entonces para actualizar el peso $w_{i,k}^h$ en la conexión del nodo k de la capa $h-1$ al nodo i de la

última capa oculta usamos:

$$w_{i,k}^h \leftarrow w_{i,k}^h - \eta\ \delta_i^k v_k^{h-1} \quad k = 0, ..., S \quad i = 1, ..., M \tag{1.23}$$

donde S es el número de nodos en la capa $h-1$ y $v_k^{h-1} \quad k = 1, ..., S$ es la salida de esta capa.

Para evitar que el algoritmo converja a mínimos locales se suele agregar a la corrección un nuevo término con una nueva variable μ llamada *momentum*. Como su nombre lo sugiere, este nuevo término mantiene una inercia en la dirección de descenso que traía del paso anterior. Entonces la actualización de los pesos viene dada por:

$$w_{i,k}^h \leftarrow \mu\ w_{i,k}^h - \eta\ \delta_i^k v_k^{h-1} \tag{1.24}$$

Hay dos formas de implementar el procedimiento de minimización, *batch* y *on-line*. En la versión *batch*, la actualización de pesos se lleva a cabo luego de pasar por todos los objetos del conjunto D, esto es denominado *época*. En cambio de manera *on-line*, los pesos son actualizados para cada objeto que es pasado a la red para entrenar.

El método utilizado en esta Tesina para ajustar los pesos de las redes es el siguiente:

Gradiente Conjugado Buscar en cada iteración la dirección del gradiente puede no representar la mejor elección de dirección de búsqueda. Para mejorar esto se eligen sucesivas direcciones de búsqueda tales que, a cada paso, la componente del gradiente paralela a las direcciones de búsquedas previas son inalteradas. A estas direcciones se las denomina *no interfering o conjugadas* [17].

1.5.3. Redes Neuronales Profundas

Las redes neuronales de una o dos capas ocultas, se encuentran limitadas con respecto a las funciones que pueden representar. Si bien las redes neuronales de tan solo una capa oculta resultan aproximadores universales de funciones continuas [18], al momento de modelar funciones de alta variabilidad estas requieren de un número exponencialmente mayor de neuronas que una arquitectura de mayor profundidad [19]. En consecuencia, surge la necesidad de implementar las *arquitecturas profundas*, las cuales poseen muchas capas de componentes adaptativos *no-lineales*, permitiendo la representación de una amplia familia de funciones de manera más compacta que las arquitecturas poco profundas utilizadas habitualmente.

Se conocen varias funciones simples que son dificiles de representar con una arquitectura poco profunda mientras que con una arquitectura de la profundidad adecuada se pueden representar fácilmente (ver por ejemplo [19]).

Las arquitecturas profundas fueron aplicadas principalmente en problemas de reconocimiento de imágenes [20, 21], reconocimiento de voz [22], modelado acústico [23],

secuencia de video [24], y captura de datos del movimiento de humanos [25].

1.5.4. El problema de entrenar redes profundas

El método estándar de aprendizaje consiste en inicializar los pesos de la red neuronal con valores aleatorios en un intervalo determinado (Por ej: [-0.5 ; 0.5]) y luego ajustar los pesos por descenso de gradiente. Se sabe que de esta forma se obtienen soluciones pobres para redes neuronales profundas debido a que el descenso de gradiente fácilmente puede quedar atrapado en un mínimo local. Puede ocurrir que con más capas, el número o la anchura de tales cuencas (mínimos locales) aumente. Este problema se resolvió a partir de 2007 donde se propuso un pre-entrenamiento de cada capa en forma secuencial, el cual consiste en un entrenamiento *no-supervisado* mediante *Máquinas de Boltzmann Restringidas* [26]. Esto permite inicializar los parámetros de la red profunda en una región cerca del óptimo buscado, para luego aplicar algoritmos de descenso por el gradiente realizando así un *ajuste fino* (*fine-tune*).

Recientemente ha quedado en evidencia que resulta posible entrenar ciertas arquitecturas profundas sin necesidad un pre-entrenamiento, si se dispone una cantidad suficientemente grande de datos supervisados (etiquetados) [15]. Entre los elementos clave para que esto sea posible [14] se encuentra el uso de las unidades de activación *ReLUs* (Ec. 1.19). Con este tipo de unidad de activación, la cantidad de épocas de *backpropagation*, es aproximadamente 6 veces menor que al usar unidades sigmoideas. Además, el tipo de saturación de estas unidades reduce el estancamiento en mínimos locales del descenso por gradiente. Estos resultados motivaron a utilizar las unidades *ReLUs* en esta tesina (Sec. 2.1.2), dado que se dispone de gran cantidad de datos para realizar un entrenamiento supervisado.

1.5.5. Entrenamiento

En la práctica es común utilizar tres conjuntos, uno de *entrenamiento*, uno de *validación* y otro de *testeo*. El conjunto de testeo queda oculto durante el proceso de entrenamiento. El conjunto de validación actúa como un *pseudo-testeo*. Para problemas en los que los datos están contaminados con ruido, se entrena el predictor hasta que la mejora en el desempeño sobre el conjunto de entrenamiento no sea correspondida con la del de validación. En este punto se debe detener el entrenamiento para evitar el *sobreajuste* (ver Sec. 2.1.2). Finalmente el conjunto de testeo se utiliza para cuantificar el error del método y comparar contra otros métodos.

1.6. Modelos Locales Lineales

Una clase de modelos muy simples para realizar la predicción de series temporales son los *modelos locales lineales*. Estos serán utilizados en esta Tesina como referencia al evaluar el error de predicción de las Redes Neuronales propuestas (Cap. 3). Este método es una variante para resolver un problema de aprendizaje supervisado, en este caso un problema de regresión. Se basa en una estrategia *divide-y-vencerás* (divide-and-conquer), que consiste en dividir un problema complejo en varios problemas simples, y luego combinar las soluciones de éstos para obtener la solución del problema original.

Suponemos que $b^* \in \mathbb{R}^m$ representa un estado del sistema. Queremos evaluar $f_h(b^*)$, donde f_h es la función que devuelve el valor de la serie temporal, h unidades de tiempo a futuro. Se buscan en el conjunto de *entrenamiento* los k primeros vecinos $b_1, ..., b_k$ de b^* en $\mathbb{R}^m$. Como cada b_i pertenece al conjunto de *entrenamiento*, también conocemos el valor $Y_i = f_h(b_i)$. Luego con los k valores (b_i, Y_i), se construye un modelo lineal, en este caso una regresión lineal, ajustando un polinomio de grado 1 por mínimos cuadrados :

$$\hat{f}_h(x) = x_1 + x_2 + ... + x_m + c \tag{1.25}$$

Una vez obtenido el polinomio, éste es evaluado sobre el estado b^*, obteniendo la predicción h unidades de tiempo a futuro:

$$\hat{f}_h(b^*) = \hat{Y}^* \tag{1.26}$$

Los parámetros libres de este método son:

- k = cantidad de vecinos usada para ajustar la regresión lineal,
- m = dimensión del modelo

1.7. Estimación del error

Dado un dataset $D = \{d_1, d_2, ..., d_N\}$ donde $d_i \in \mathbb{R}^n$, y su correspondiente conjunto objetivo $Y = \{y_1, y_2, ..., y_N\}$ donde $y_i \in \mathbb{R}$, una estimación del error del predictor f es:

$$error(f) = \frac{1}{N} \sum_{i=1}^{N} (f(d_i) - y_i)^2 \tag{1.27}$$

A este error así definido se lo denomina *Error Cuadrático Medio* (*Mean Square Error*, MSE).

1.8. Objetivos de esta Tesina

En esta Tesina se presenta una técnica basada en redes neuronales profundas para la predicción de valores futuros de una serie temporal observada, con la finalidad de encontrar respuestas a las siguientes preguntas: ¿El uso de una arquitectura *profunda* ayuda para este problema en particular? ¿La *salida-múltiple* tiene algún beneficio frente a la *salida-simple*? ¿En los casos anteriores, resulta mejor el método iterativo o el directo?

Capítulo 2

Predicción con Redes Neuronales Profundas

En este capítulo se propone el uso de redes neuronales *profundas* (Sec. 1.5.4) y evaluar el desempeño de éstas frente a las redes neuronales *no-profundas* en el problema de predicción de valores futuros de una serie temporal. A su vez, evaluar el beneficio considerar una arquitectura de *salida-múltiple* frente a la *salida-simple* para realizar predicción de series temporales. Se utilizarán ambas estrategias, *directa* e *iterativa*, descriptas en el capítulo previo.

2.1. Metodología propuesta

En esta sección se describe la metodología propuesta que luego será evaluada en el capítulo siguiente.

2.1.1. Preparación de los Datos

Antes de empezar a trabajar con las RN, los datos son normalizados a *media* cero y *desvío estándar* $\sigma = 10$, este último valor fue el mejor candidato de distintos que se probaron, $2, 4, 6, \ldots, 20$, y que además es sugerido en [27], donde utilizan unidades ReLus (Ec. 1.19) en redes profundas para modelar y reconocer voz a partir de las ondas sonoras.

Como primer paso se busca obtener una reconstrucción del espacio de fases a partir de considerar coordenadas de retraso (Sec. 1.3.1). Sobre este espacio de fases se espera que la dinámica del sistema en estudio sea determinística, siendo la ley de la dinámica sobre este espacio la función que se quiere aproximar con las redes neuronales propuestas. Para determinar las coordenadas de retraso es necesario encontrar el valor óptimo para el tiempo de retraso τ y la dimensión del espacio de fases m. Estos parámetros se determinaron con la metodología propuesta en [5, 6].

Para entrenar un modelo f_h, en nuestro caso una red neuronal, que prediga valores de la serie temporal a h pasos futuros, definimos la matriz de datos de entrada X y la matriz de datos de salida Y_h^s de la red, de la siguiente manera: con el valor τ se construye la matriz de coordenadas de retraso, X (Sec. 1.3.1) de la serie temporal $[x_1, x_2, \ldots, x_N]$ donde cada patrón de entrenamiento, fila i de dicha matriz, está asociado al vector reconstruido correspondiente al tiempo $t = (m-1)\tau + i$ de la serie temporal.

$$X = \begin{bmatrix} x_{(m-1)\tau+1} & \cdots & x_{\tau+1} & x_1 \\ x_{(m-1)\tau+2} & \cdots & x_{\tau+2} & x_2 \\ \vdots & \ddots & \vdots & \\ x_{N-h} & \cdots & x_{N-(m-2)\tau-h} & x_{N-(m-1)\tau-h} \end{bmatrix} \tag{2.1}$$

Como salida de la red se toma la misma serie, h pasos hacia adelante, por lo tanto ambas matrices tendrán $N-(m-1)\tau-h$ filas. Para el caso de *salida-múltiple*, definimos la dimensión de la salida de la red $s = m$, siendo ésta igual al tamaño de la entrada y a la dimensión del espacio de fases.

$$Y_h^m = \begin{bmatrix} y_{(m-1)\tau+1+h} & \cdots & y_{\tau+1+h} & y_{1+h} \\ y_{(m-1)\tau+2+h} & \cdots & y_{\tau+2+h} & y_{2+h} \\ \vdots & \ddots & \vdots & \\ y_N & \cdots & y_{N-(m-2)\tau+h} & y_{N-(m-1)\tau} \end{bmatrix} \tag{2.2}$$

Para el caso *salida-simple* consideramos la dimensión de la salida de la red $s = 1$, teniendo la matriz de datos de salida una única columna:

$$Y_h^1 = \begin{bmatrix} y_{(m-1)\tau+1+h} \\ y_{(m-1)\tau+2+h} \\ \vdots \\ y_N \end{bmatrix} \tag{2.3}$$

2.1.2. Construcción del Modelo

Una vez obtenidos los conjuntos X y Y_h^s, como se indica en la Sec. 2.1.1, se consideran tres particiones disjuntas que serán el conjunto *entrenamiento*, el de *validación* y un conjunto de *test*. Luego comenzamos con el entrenamiento, de las RN a fin de construir un modelo predictor.

Entrenamiento

Se realizaron pruebas preliminares haciendo un pre-entrenamiento con RBMs (Sec. 1.5.4). Los errores obtenidos con el pre-entrenamiento fueron mayores que utilizando un entrenamiento puramente supervisado. Lo cual indica que los parámetros de la

red quedan en una región lejana del óptimo buscado. La posibilidad de entrenar una red profunda sin un pre-entrenamiento previo ya ha sido reportado por [14, 15] para problemas donde se dispone de gran cantidad de datos supervisados. Por lo tanto, en este trabajo no se utiliza el pre-entrenamiento debido a que no resulta beneficioso.

Busqueda de Arquitectura

No hay un método definitivo para decidir a priori la cantidad de capas ocultas necesarias ni la cantidad de neuronas por cada capa oculta. Estos parámetros están relacionados a la complejidad de las funciones no-lineales que puede ser generada por la red neuronal. Comenzamos el entrenamiento barriendo sobre el parámetro h (predicción a h pasos futuros), con una red simple de una capa oculta, inicialmente con una cantidad de neuronas igual a la cantidad de variables, luego se incrementa en potencia de dos, hasta observar que el incremento de éstas no mejore el resultado (en términos del error de validación), por lo cual la incorporación de nuevas neuronas estaría añadiendo complejidad a la red innecesariamente. Luego se realiza un paso más fino dentro del intervalo donde se obtuvo buen resultado en el paso anterior (por ej.: $[2^7; 2^8]$), obteniendo así la cantidad óptima de neuronas en la capa oculta. Por otro lado, para las redes profundas, empezamos con una red de 3 capas ocultas, todas con la misma cantidad de neuronas, de manera que la red tenga la misma cantidad de pesos (parámetros) que la red no-profunda. Luego se incrementa la cantidad de neuronas en las tres capas, con igual número de neuronas en ellas, siguiendo el mismo procedimiento utilizado para las redes no-profundas. Por simplicidad se buscará una cantidad de neuronas que sea conveniente para todos los posibles valores de h.

Criterio de corte

Tendremos dos tipos de series temporales a estudiar, las sintéticas, que están libres de ruido y las series temporales reales que presentan cierto nivel de ruido. Para el caso de las series temporales sintéticas, al estar libres de ruido, durante el entrenamiento no observaremos sobreajuste en el conjunto de entrenamiento (el error de validación no comienza a aumentar a partir de una determinada época). Por lo tanto es necesario algún criterio de corte adicional, ya sea por cantidad de épocas o disminución del error en un lapso determinado de épocas.

Se adoptan los siguientes criterios de corte de entrenamiento, para series sintéticas:

- Se impone un máximo de 400.000 iteraciones de entrenamiento.
- Si en 1.000 iteraciones el error de validación desciende menos que 10^{-7}:

  $$Eval_{(it-1000)} - Eval_{it} \leqslant 10^{-7}$$

  o se llega al máximo de iteraciones, según lo que suceda primero, el entrenamiento se detiene.

Para las series temporales reales, al estar contaminadas con ruido, podríamos tener *sobreajuste* en el conjunto de entrenamiento, por lo tanto en el error de validación se observará un mínimo absoluto en una época determinada de entrenamiento, y después comenzará a aumentar, perdiendo eficacia para predecir datos no vistos. El método más habitual para evitar el *sobreajuste* es el de *detención temprana* (*early stopping*). Utilizando un conjunto de validación, se aplica la red a esos elementos y se obtiene el correspondiente error de validación, similar a un error de test. A lo largo del entrenamiento se calcula este error y al final del mismo se busca la época en donde el error de validación fue mínimo. Debido a que la red no utilizó estos elementos para entrenar es una buena estimación de cuándo ésta comienza a sobreajustar el conjunto de entrenamiento.

2.1.3. Implementación

Por cuestiones de eficiencia, el algoritmo de descenso por gradiente está implementado en C++, utilizando la biblioteca *alglib* [28], más precisamente la función

$$mincgoptimize(state, function1\_func),$$

la cual recibe el estado inicial *state* del problema, en este caso los valores de los pesos (conexiones entre neuronas), y un puntero a la función que calcula el gradiente *function1_func*. Los parámetros de la función *mincgoptimize* que determinan cuándo debe finalizar el algoritmo GC son:

- EpsF: ≥ 0, finaliza en la $(k+1)-$ésima iteración, si se cumple la condición: $\|F(k+1) - F(k)\| <= EpsF * max\{\|F(k)\|, \|F(k+1)\|, 1\}$
- G: la norma del gradiente
- F: el cambio de la función en iteraciones consecutivas
- StepMax: Tamaño máximo del paso, si es 0, no tiene límite.
- Iter. máx.: la cantidad de iteraciones máxima a realizar.

Al dejarlos en 0, (excepto el número máximo de iteraciones, que fue limitado según el problema), el método de detección es automático (según la implementación de *alglib*) y selecciona un epsilon pequeño. Luego, se optimizan los parámetros de la red (los pesos), con el objetivo de ajustar mejor los ejemplos de entrenamiento.

A nivel más abstracto, utilizamos el lenguaje de alto nivel R para manejar todo a lo que respecta a la preparación de los datos, posee un buen manejo en alto nivel de objetos matemáticos como vectores, matrices y conjuntos.

Las corridas se generaron un *cluster* ("grupo" de computadoras que se comportan como si fuesen una única computadora), en paralelo, con la intención de barrer los parámetros libres a ajustar (cant. de capas, cant. de neuronas por capa, etc), y también sobre H (horizonte de predicción).

2.2. Simulación de la dinámica

Al disponer de un modelo (en este caso una red neuronal) entrenado a partir de los datos disponibles, el mismo se puede utilizar para generar nuevos datos. La red neuronal debiera poder simular la dinámica original que generó los datos. Se puede entonces generar una nueva secuencia completa de la serie temporal. Esta serie temporal no será idéntica a la serie temporal original, pero puede compartir ciertas propiedades estadísticas. Comparando estas propiedades se puede evaluar el modelo obtenido, resultando esta evaluación complementaria a evaluar el error de predicción sobre un conjunto de test.

El procedimiento para generar la serie sintética es el siguiente. Se parte de un vector de tiempos de retraso construido a partir de datos de la serie temporal original. Se utiliza para la producción de una secuencia de datos, una red entrenada para predecir en un horizonte $H = 1$. La salida puede ser simple o múltiple. Si el vector de entrada corresponde a tiempo t interesa predecir el valor de la serie a tiempo $t + 1$. En el caso de la red de salida múltiple, las salidas corresponden a los tiempos $t - m\tau + 1, t - (m - 1)\tau, \ldots, t + 1$ y sólo se retiene el último valor. Con el valor de la serie obtenido para t+1 combinado con las correspondientes coordenadas de la secuencia original se obtiene el vector de tiempo de retraso asociado al tiempo $t + 1$. Este vector se utilizará como entrada para la siguiente iteración utilizando la misma red neuronal. De esta forma se obtendrá el valor de la serie a tiempo $t + 2$. Repitiendo este proceso, tras m pasos, el vector de entrada estará compuesto sólo por valores generados por la misma red neuronal. El proceso se repite n veces para generar una secuencia completamente generada por la red neuronal. De modo de eliminar la influencia de la porción original de la serie se descartan los primeros 1000 valores generados. El procedimiento descripto se sintetiza en el Algoritmo (1).

La secuencia de datos de la serie temporal se genera con el siguiente algoritmo:

- tw: tamaño óptimo de ventana.
- V_{ini}: vector con tw elementos.
- RN: red neuronal entrenada con $H = 1$.
- m: dimensión de entrada de la red.

Algorithm 1 Algoritmo Iterativo de Producción de datos

Require: $V_{ini}, tw, tau, RN, m, s, n$
Ensure: sec
1: $i \leftarrow 0$
2: $sec \leftarrow <>$
3: $D \leftarrow u.delays(m, tau)$
4: **while** $i < n$ **do**
5: **if** $i <= tw$ **then**
6: $X \leftarrow <V_{ini} + +sec>[D + i]$
7: **else**
8: $X \leftarrow sec[D + i - tw]$
9: **end if**
10: $\overline{Y} \leftarrow propagar(X, RN)$
11: $sec \leftarrow < sec + +(\overline{Y})[s] >$
12: $i \leftarrow i + 1$
13: **end while**
14: **return** sec

- s: dimensión de la salida de la red. Donde $s \in \{1, m\}$ según el tipo de salida, *simple* o *múltiple*.
- n: cantidad de elementos a producir.
- tau: tiempo de retraso.
- sec: secuencia de datos creada con la *Red*, utilizando el método iterativo.
- $u.delays(m, tau)$: vector de índices de las coordenadas de retraso, donde m es la cantidad de componentes (dimensión de embeding) y tau es el tiempo de retraso.
- $<>$: secuencia de elementos.

El Algoritmo (1) recibe el vector V_{ini} el cual es utilizado para iniciar la producción de una secuencia de datos, propagando las coordenadas de retraso de éste por la *Red* ya entrenada para predecir el siguiente dato de la secuencia (es decir $H = 1$). Luego, de la salida de la red sólo nos interesa el valor más lejano (última neurona de salida) para concatenarlo con los datos generados en los pasos anteriores. En la siguiente iteración, se avanza el tiempo en una unidad, extrayendo las coordenadas de retraso de la secuencia generada correspondiente al instante $i + 1$, propagando éste por la red y concatenando nuevamente el dato más alejado de la salida a la secuencia de datos previa. Este procedimiento es repetido n veces. Luego, la secuencia es devuelta en el vector *sec*. Para ambos casos, *salida-simple* y *salida-múltiple* se utiliza el mismo algoritmo, con $s = 1$ para el caso *simple* y $s = m$ para el caso *múltiple*.

Una vez generada la secuencia de datos, creamos la reconstrucción en el espacio de fase mediante *coordenadas de retraso* (Sec. 1.3.1) para obtener el atractor correspondiente. La visualización del atractor obtenido (o una proyección bidimensional del

mismo) permite obtener una primer idea cualitativa de la calidad de la red neuronal como modelo de la dinámica. Para pasar a una evaluación más cuantitativa se puede recurrir al cálculo de propiedades invariantes del atractor tales como los exponentes de Lyapunov o la dimensión fractal (ver Sec. 1.3.2). Se evaluará entonces la dimensión de correlación para el conjunto de *test* y para la serie sintética generada. A partir de las curvas $D(\epsilon)$ vs. ϵ (Sec. 1.3.2) se podrá evidenciar diferencias entre los atractores a distintas escalas que pueden no observarse a simple vista a partir de gráficas del atractor.

Capítulo 3

Experimentos

En este capítulo se describen los pasos del entrenamiento de las redes neuronales para luego analizar los resultados obtenidos con la metodología propuesta en el Cap. 2 para la predicción sobre ejemplos prácticos (series temporales sintéticas y experimentales).

3.1. Entrenamiento

Para entrenar un predictor se define implícitamente un espacio de entrada. La dimensión de éste espacio queda determinada por la dimensión del espacio de fases reconstruido (ver Sec. 1.3). En particular, se utilizaron *coordenadas de retraso* (Sec. 1.3.1) utilizando tamaños óptimos de la ventana t_w y de la dimensión de embedding (los cuales varían para cada serie temporal), obtenidos a partir de la metodología propuesta en [5, 6]. Luego comenzamos a trabajar con la metodología propuesta en la Sec. 2.1.

Para las series temporales utilizadas en los experimentos, se particiona el dataset en 3 conjuntos (Sec. 2.1.2), cada uno contiene datos consecutivos, o sea que no son tomados al azar. Se utilizaron 8.000 datos para el conjunto de *entrenamiento*, 2.000 datos para el de *validación*, y 10.000 datos, para la tarea de predicción (conjunto de *test*).

3.1.1. Búsqueda de la arquitectura

En el caso de las redes *no-profundas*, utilizando la metodología descripta en la Sec. 2.1.2 se encontró un valor óptimo de cantidad de neuronas alrededor de 200, para todas las series temporales, por lo tanto se adoptó ese valor para realizar los experimentos.

Por cuestiones relacionadas con el tiempo de ejecución y el consumo de recursos, sólo se utilizaron redes *profundas* de tres capas ocultas. Para dichas redes, se utilizó la metodología descripta en la Sec. 2.1.2 para buscar un cantidad conveniente de neuronas.

Obteniendo que 20 neuronas son suficientes para representar la información en todas las series temporales utilizadas, por ende, empleamos esta cantidad para realizar los experimentos.

3.1.2. Caracterización del desempeño

Para evaluar el desempeño de los métodos sobre el conjunto de *test* (de 10.000 datos), tanto *directo* como *iterativo* (Sec. 1.4.2), variamos el valor del parámetro H (horizonte de predicción), donde $H \in \{\lceil 10^x \rceil \mid x \in \{1.1,\ 1.2,\ \ldots\ ,\ 2.9,\ 3.0\ \}\}$ (escala logarítmica), utilizando la siguiente metodología, dependiendo del método:

Directo: Para cada valor de H, tenemos una red entrenada a H pasos futuros ($RN_{h=H}$). Se propaga cada fila de la matriz del conjunto de *test* por la red ya entrenada, obteniendo la matriz de salida $\hat{P}$. Mido el error cometido entre la última columna de la matriz P (original de salida del conjunto de *test*) y la matriz $\hat{P}$ con la formula (1.27).

Iterativo: Utilizamos la red entrenada a un paso futuro ($RN_{h=1}$) como entrada del algoritmo 1 para construir la predicción de n datos donde $n = H$. Este procedimiento se repite (10.000-H) veces, variando el instante $t_{ini} = t_0 + i$ (del conjunto de *test*) donde se inicia la predicción, con $i \in \{0, \ldots, $10.000-$H\}$. Luego se calcula el error, en este caso se calcula utilizando la última columna de la matriz P (original de salida del conjunto de *test*) y la predicción $\hat{P}$, que resulta ser un vector de longitud n, con la Fig. 1.7.

Para ambos métodos el error es dividido por *sigma* (utilizamos *sigma= 10* ver Sec. 2.1.1), para que se encuentre a la misma escala que el método LLM (*Modelos Locales Lineales*, Sec. 1.6).

Para el método LLM se utilizan los parámetros m igual a la dimensión del espacio de fases y $k = 15$. Este último valor fue obtenido de variar k de 1 a N, siendo N el tamaño del conjunto de entrenamiento, y mediante un conjunto de validación se buscó el valor de k donde se encuentra el mínimo error del conjunto de validación.

Otra forma de caracterizar el desempeño de las RN es analizando la dinámica generada por éstas. Para cada una de las series analizadas se siguió la metodología expuesta en la Sec. 2.2.

3.2. Resultados

En esta sección se exponen los resultados obtenidos aplicando la metodología propuesta en el Cap. 2, a series temporales, tanto sintéticas como reales.

3.2.1. Serie de Mackey-Glass

En el primer caso de estudio se considera la serie de Mackey-Glass (ver Apéndice A.1). Para esta serie, el tamaño de ventana óptimo obtenido es $t_w = 30$ y dimensión de embedding $m = 4$ (tiempo de retraso $\tau = 10$) [6].

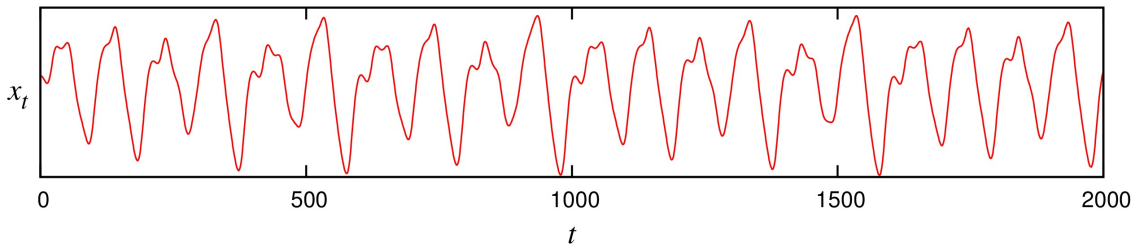

Figura 3.1: Serie temporal de Mackey-Glass. Primeros 2.000 puntos del conjunto de *test* .

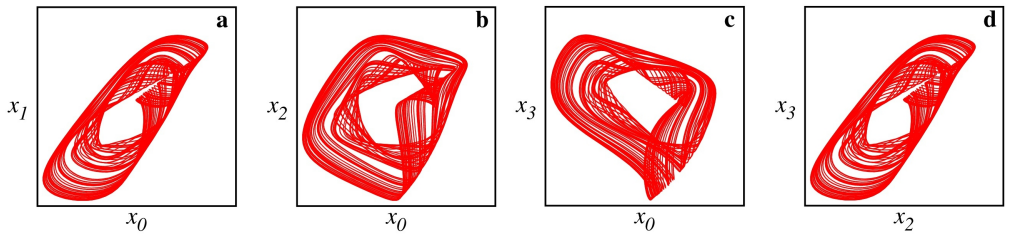

Figura 3.2: Coordenadas de las reconstrucciones de la serie de Mackey-Glass (conjunto de *test* completo, 10.000 puntos) con tiempo de retraso 10 y dimensión de embedding 4. Donde $x_i = x(t - 10i)$, $i = 0 \ldots 3$ indica la i-ésima componente de la reconstrucción correspondiente.

En la Fig. 3.1 se observan los primeros 2.000 puntos correspondientes al conjunto de *test*. Luego en la Fig. 3.2 se muestran las coordenadas de retraso, correspondientes al mismo conjunto de *test*, esta vez considerando el conjunto completo (10.000 datos). Donde $x_i = x(t - 10i)$, $i = 0 \ldots 3$ es la i-ésima componente de la reconstrucción.

Se realiza el entrenamiento descripto en la Sec. 3.1, barriendo sobre el parámetro H para las diferentes estructuras de RN, tanto *profundas* como *no-profundas*, para los métodos *iterativo* y *directo*, de *salida-simple* y *salida-múltiple*, tomando como referencia los modelos locales lineales (LLM Sec. 1.6), con el fin de comparar los errores (Error Cuadrático Medio, Sec. 1.7).

El resultado más sobresaliente de la Fig. 3.3 es que la red profunda de *salida-simple* (4-20-20-20-1) es la red con mejor comportamiento en un balance sobre el rango completo horizontes: presenta un mínimo o comparable error con respecto al resto de los métodos evaluados.

Es posible, a partir de la gráfica, distinguir una predicción a *corto plazo* para $h \lessapprox 10^2$ y otra de largo plazo $h \gtrapprox 10^2$ donde el comportamiento de los distintos métodos varía sensiblemente.

En las predicciones a corto plazo las mayores diferencias entre métodos ocurre entre las estrategias *directa* vs. *iterativa* y entre *salida-simple* vs *salida-múltiple*, imponiéndose las primeras sobre las segundas respectivamente. La profundidad de la red no parece ser un elemento clave en la predicción a *corto plazo*. Esto se debe a que la predicción a corto plazo no requiere gran no-linealidad para esta serie.

En el caso *iterativo* se observa un resultado desfavorable para las predicciones a *corto plazo* frente al método *directo*. Esta relación se invierte en un pequeño rango dentro los horizontes de *largo plazo*.

Para el caso de *salida-múltiple* se observa sistemáticamente un peor desempeño independientemente de la profundidad de la red o estrategia *iterativa* o *directa*. Esto sugiere que entrenar estas redes minimizando el error sobre las cuatro neuronas de salida deteriora el desempeño sobre el conjunto de test sobre la neurona correspondiente al horizonte más lejano (que es el que se evalúa en la figura para comparar con los métodos de *salida-simple*).

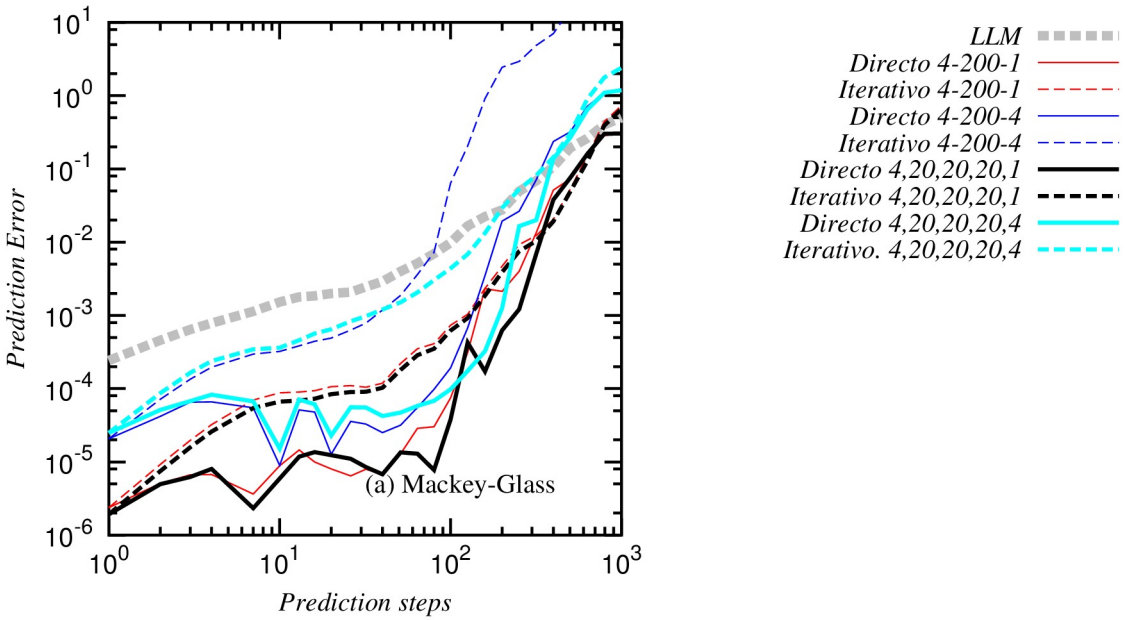

Figura 3.3: Serie temporal de Mackey-Glass. Error de predicción en función del horizonte H (ambos en escala logarítmica), para los métodos *iterativo* y *directo*, tanto en RN *profundas* como *no-profundas*, tomando como referencia los modelos locales lineales (LLM).

Una diferencia sistemática entre las variantes de los métodos, se encuentra entre los métodos *iterativos* y *directos* (Sec. 1.4.2), donde se puede observar que, en general, los métodos *directos* dan mejores resultados para esta esta serie. Pero para este método, si se quiere la predicción completa hasta un horizonte H usando como entrada solamente datos de entrenamiento, se deben entrenar H predictores (RN) (Sec. 1.4.2). Lo cual implica consumir una gran cantidad de recursos, frente al método *iterativo* que sólo necesita entrenar una única RN para H=*1*.

Simulación de la dinámica

Comparamos el atractor original y los producidos por las RNs con el método *iterativo* ($H = 1$) (Sec. 2.2) para las distintas arquitecturas de RN, *profundas* y *no-profundas*, y para la *salida-simple* y *salida-múltiple*.

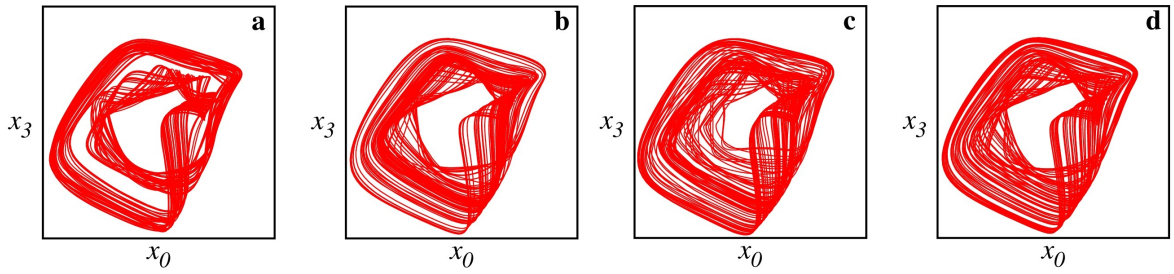

Figura 3.4: Datos generados por las redes neuronales. Se muestran las reconstrucciones (como en la Fig. 3.2 b) de la serie de Mackey-Glass generada por las redes neuronales consideradas. Las predicciones fueron generadas con el método *iterativo* ($h = 1$): (a) con una RN de estructura 4-200-4 (b) con una RN de estructura 4-200-1 (c) RN de estructura 4-20-20-20-4 (d) RN de estructura 4-20-20-20-1. Para cada caso se muestra únicamente la proyección sobre las coordenadas x_0 y x_3, primera y última componentes de la reconstrucción respectivamente, $x(t)$ y $x(t-30)$.

Utilizando las RN entrenadas para h=1, se generan con el método iterativo 10.000 datos de la serie de Mackey-Glass, para las distintas estructuras de RN, ya sea *profundas* como *no-profundas*, con *salida-simple* y *salida-múltiple*. En la Fig. 3.4 podemos observar que la estructura *salida-múltiple*, paneles (a) y (c), tiene inflexiones en la trayectoria de las órbitas. Esto se debe a que la predicción es generada con la última neurona de las 4 neuronas de salida, y en ocasiones ésta puede tener mayor o menor error que en las neuronas restantes.

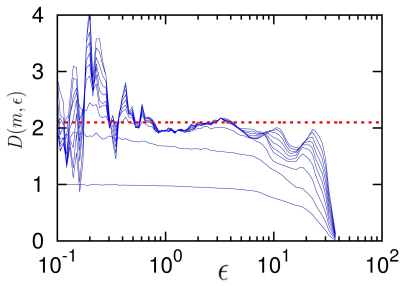

Figura 3.5: Dimensión de correlación del conjunto de *test* original, serie de Mackey-Glass. Con dimensión $D(m, \epsilon)$ en el eje y, ϵ en el eje x

En la Fig. 3.5 vemos graficadas las curvas de dimensión de correlación (Sec. 1.3.2) del conjunto de *test*, con $m = 2 \ldots 10$, desde la curva inferior hacia la superior, y $\epsilon \in [10^{-1}, \ldots, 10^{2}]$ donde se sabe que la dimensión fractal (Eq. 1.8) es 2.1 (Ver [29], pag. 379), valor que se señala en la gráfica con una línea de trazos en color rojo .

En la Fig. 3.6 se grafican como en la Fig. 3.5, las respectivas curvas de dimensión de correlación correspondientes a las series generadas por cada una de las RN consideradas.

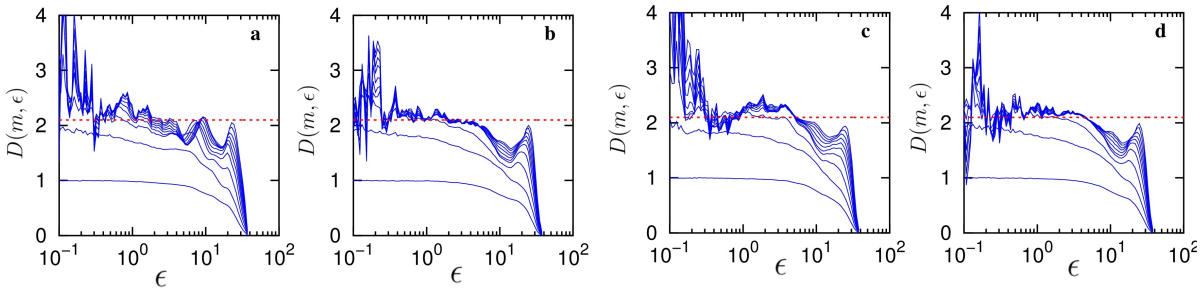

Figura 3.6: Dimensión de correlación de las series generadas con las distintas arquitecturas de RN utilizadas. (a) con una RN de estructura 4-200-4, (b) con una RN de estructura 4-200-1, (c) RN de estructura 4-20-20-20-4 y (d) RN de estructura 4-20-20-20-1. Línea de trazo en color rojo en $D(m,\epsilon) = 2$, dimensión de correlación de la serie original.

En todas las gráficas vemos una línea roja en trazo interrumpido, en $D(m,\epsilon)$=2.1, para tener una referencia de donde se encuentra la dimensión de correlación con respecto a la serie original. En los paneles (b) (RN de estructura 4-200-1) y (d) (RN de estructura 4-20-20-20-1) se nota una mejor definición del plateau (meseta) alrededor de $\epsilon = 10$, con dimensión cercana a 2.1. En cambio en los paneles (a) y (c) se observa una distorsión mayor del perfil de curvas. En el panel (a) (RN de estructura 4-200-4) no se observa ninguna meseta, existiendo además una separacion de la secuencia de curvas. Por su parte el panel (c) (RN de estructura 4-20-20-20-4) presenta una meseta más acotada y en un valor por encima del original.

3.2.2. Serie de Lorenz

En el segundo caso de estudio se presenta la serie correspondiente al sistema de Lorenz (ver Apéndice A.2). Para esta serie, el tamaño de ventana óptimo obtenido es $t_w = 12$, siendo su dimensión $m = 3$ ($\tau = 6$ el tiempo de retraso) [6].

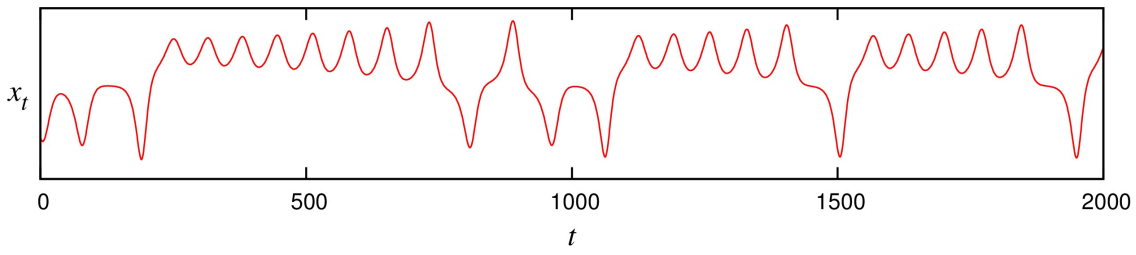

Figura 3.7: Primeros 2.000 datos del conjunto de *test* de la serie temporal de Lorenz.

Luego de realizar el entrenamiento indicado en la Sec. 3.1, barriendo sobre el parámetro H para las diferentes estructuras de RN y para los métodos *iterativo* y *directo* se obtiene la siguiente gráfica de *Error de predicción* vs. *Horizonte de predicción* H sobre el conjunto de *test*.

En esta gráfica (Fig. 3.9) observamos que la red profunda es la mejor opción en

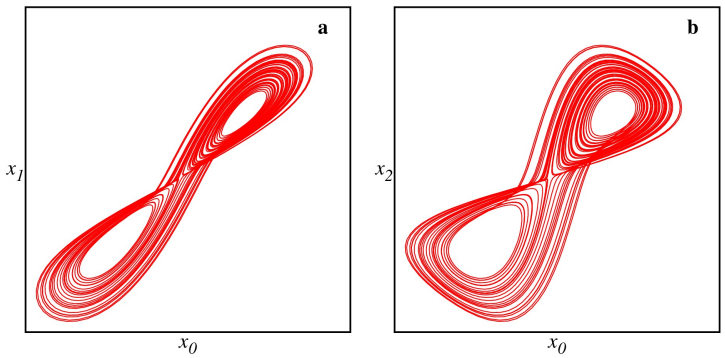

Figura 3.8: Coordenadas de las reconstrucciones de la serie de Lorenz con tiempo de retraso 6 y dimensión de embedding 3. Donde $x_i, i = 0 \ldots 2$ indica la i-ésima componente de la reconstrucción correspondiente: $x(t - 6i)$.

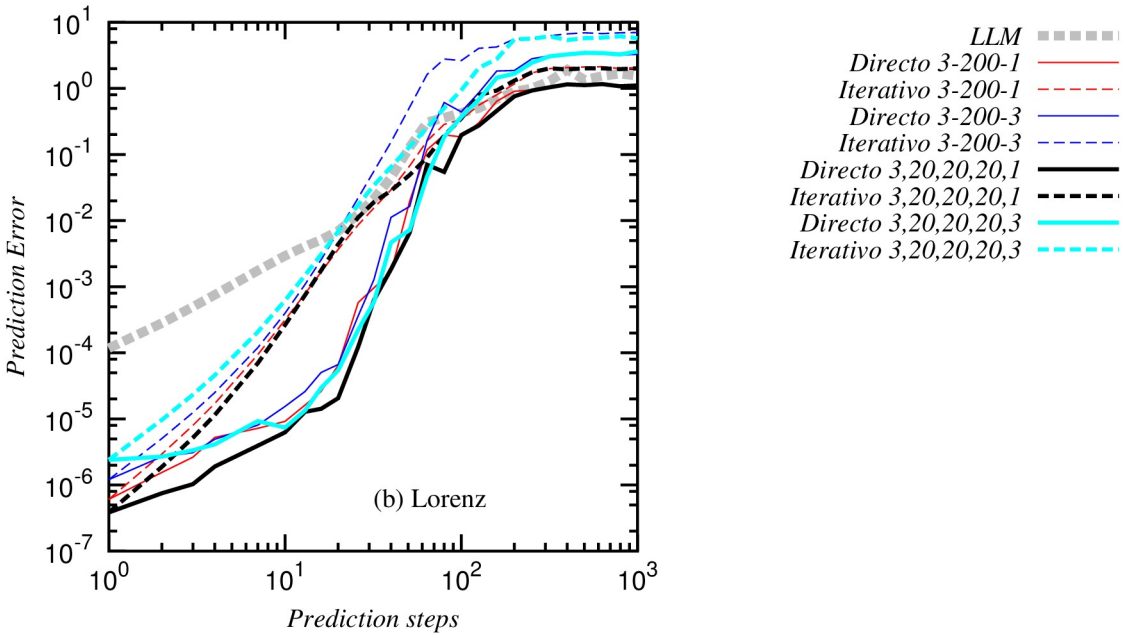

Figura 3.9: Serie temporal de Lorenz. Error de predicción en función del horizonte H, ambos en escala logarítmica, para las distintas arquitecturas y métodos. Tomando el método LLM como referencia

cuanto a las arquitecturas de redes en general para todo el rango de horizontes H. En cuanto a los métodos tenemos una diferencia significativa a favor del método *directo*. Entre las redes con *salida-multiple* y *salida-simple* se observa una pequeña diferencia a favor de las redes de *salida-simple* que es ligeramente mayor para las arquitecturas profundas.

En cuanto a los los métodos iterativos, se comportan muy parecidos entre sí sin importar la estructura de la red (con una leve mejoría para los de salida-simple). La pendiente de crecimiento del error para los métodos iterativos es más constante. Esto se debe a la amplificación del error en $H = 1$ al iterar. En cambio el error a $H > 1$

para los métodos directos no depende del error a $H = 1$, ya que cada cálculo de H es independiente de los demás. Ambos métodos *iterativo* y *directo* inician con el mismo error, ya que no difieren en $H = 1$ para una misma estructura.

Al igual que para la serie de Mackey-Glass es posible distinguir una predicción a corto plazo y otra de largo plazo donde el comportamiento de los distintos métodos varía sensiblemente. En este caso el umbral se identifica en $h \approx 20$. También en esta gráfica alcanza a verse con mayor claridad que en la Fig. 3.3 la saturación de los errores alrededor de 10^0 cuando $h \gtrapprox 300$. Éste es el umbral de impredecibilidad de la serie.

Simulación de la dinámica

Comparamos el atractor original y los producidos por las redes neuronales para la serie temporal de Lorenz, con el método iterativo ($h = 1$), para las distintas arquitecturas y tipo de salida (*salida-simple* y *salida-multiple*).

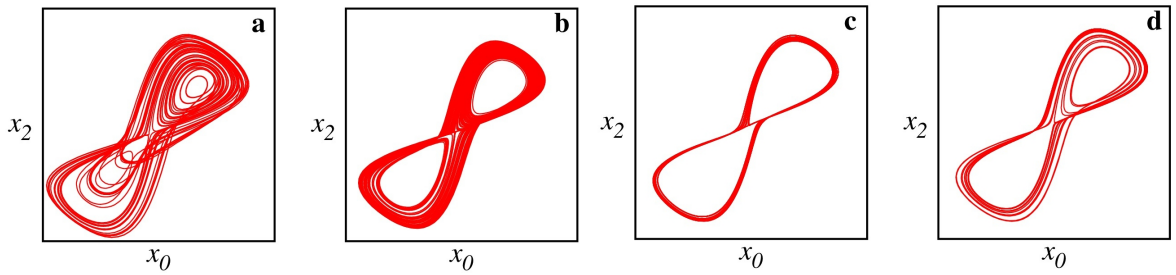

Figura 3.10: Datos generados por las redes neuronales. Se muestran Coordenadas de las reconstrucciones (como en la Fig. 3.8 b) de los datos producidos para la serie de Lorenz por las redes neuronales con el método *iterativo* ($h = 1$): (a) con una RN de estructura 3-200-3 (b) con una RN de estructura 3-200-1 (c) RN de estructura 3-20-20-20-3 (d) RN de estructura 3-20-20-20-1. Para cada caso se muestra únicamente la proyección sobre las coordenadas x_0 y x_2, primera y última componentes de la reconstrucción respectivamente, $x(t)$ y $x(t-12)$.

Vemos en la Fig. 3.10 las reconstrucciones de los datos generados, utilizando coordenadas de retraso, con el tiempo de retraso $\tau = 5$. En el panel (a) tenemos la reconstrucción según la RN 3-200-3, donde podemos observar que la trayectoria no conserva la dinámica del atractor original (Fig. 3.8), en ocasiones la parte no-lineal de la predicción, o sea decidir si la trayectoria pasa o no a la otra espiral, no esta resuelta adecuadamente. Para la arquitectura *profunda* 3-20-20-20-3, panel (c), las órbitas están demasiado juntas, o sea que no se desplegó por completo el atractor, por lo cual es de esperar que presente una dimensión fractal menor a la del atractor original. Para las RN de *salida-simple* tenemos mejores resultados observando que en el atractor del panel (d) (RN de estructura 3-20-20-20-1) las órbitas están más separadas que en el panel (c). pero aun se presentan colapsadas de a grupos. Finalmente, en el panel (b) (RN de estructura 3-200-1) tenemos el mejor resultado de los cuatro, desplegándose el atractor casi en su totalidad y respetando el paso de las órbitas de una espiral a la otra.

A partir de las figuras 3.11 y 3.12 (dimensión de correlación de la serie original y de las series generadas, respectivamente), se puede obtener una medida más precisa de lo observado sobre la apariencia de las reconstrucciones. Los paneles se presentan en el mismo orden que la figura 3.10 de los atractores reconstruidos. En todas las gráficas tenemos una línea roja en trazo interrumpido, representando la dimensión fractal del atractor original. Vemos en los paneles (a), (c) y (d) que la dimensión fractal es menor que 2, mientras que en el panel (b) es cercana a 2, siendo por lo tanto éste el mejor resultado (3-200-1).

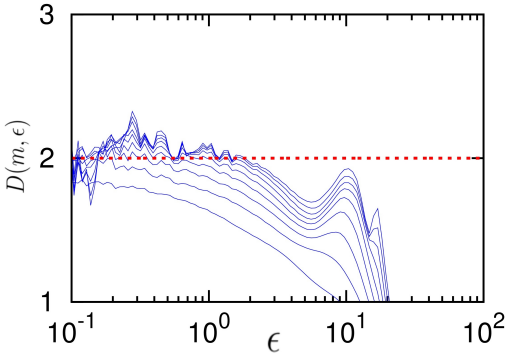

Figura 3.11: Curvas de dimensión de correlación de la serie temporal de Lorenz, conjunto de *test*. Con dimensión $D(m, \epsilon)$ en el eje y, ϵ en el eje x, siendo 2 su dimensión

.

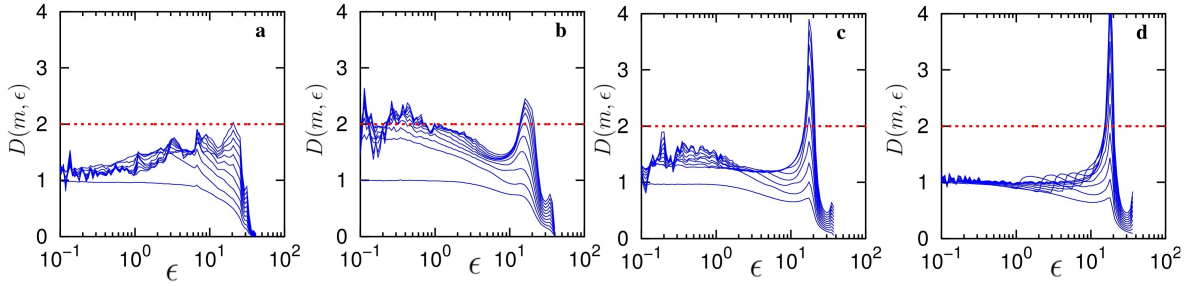

Figura 3.12: Dimensión de correlación, serie temporal de Lorenz. $D(m, \epsilon)$ en el eje y, ϵ en el eje x para las diferentes arquitecturas de RN: (a) con una RN de estructura 3-200-1 (b) con una RN de estructura 3-20-20-20-1 (c) RN de estructura 3-200-3 (d) RN de estructura 3-20-20-20-3.

3.2.3. Serie de Rössler

En este caso de estudio se presenta la serie correspondiente al sistema de Rössler (ver Apéndice A.3). Para ésta serie, el tamaño de ventana óptimo obtenido es $t_w = 10$, siendo su dimensión $m = 3$, y $\tau = 5$ el tiempo de retraso [6].

En las figuras 3.13 y 3.14 se muestran 2.000 datos de la serie temporal de Rössler y el atractor reconstruido a partir de un tramo de la serie temporal, con 8.000 datos, mediante coordenadas de retraso, con tiempo de retraso $\tau = 5$ y dimensión de embedding 3.

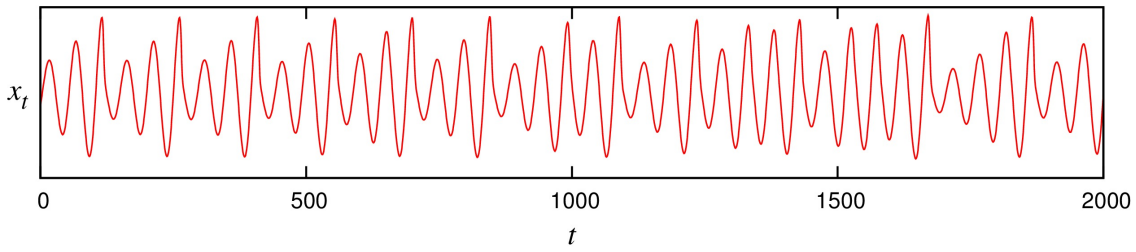

Figura 3.13: Serie temporal de Rössler

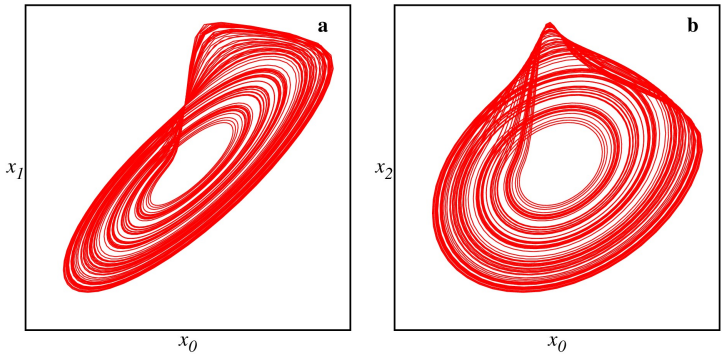

Figura 3.14: Reconstrucción serie de Rössler con tiempo de retraso 5 y dimensión de embedding 3.

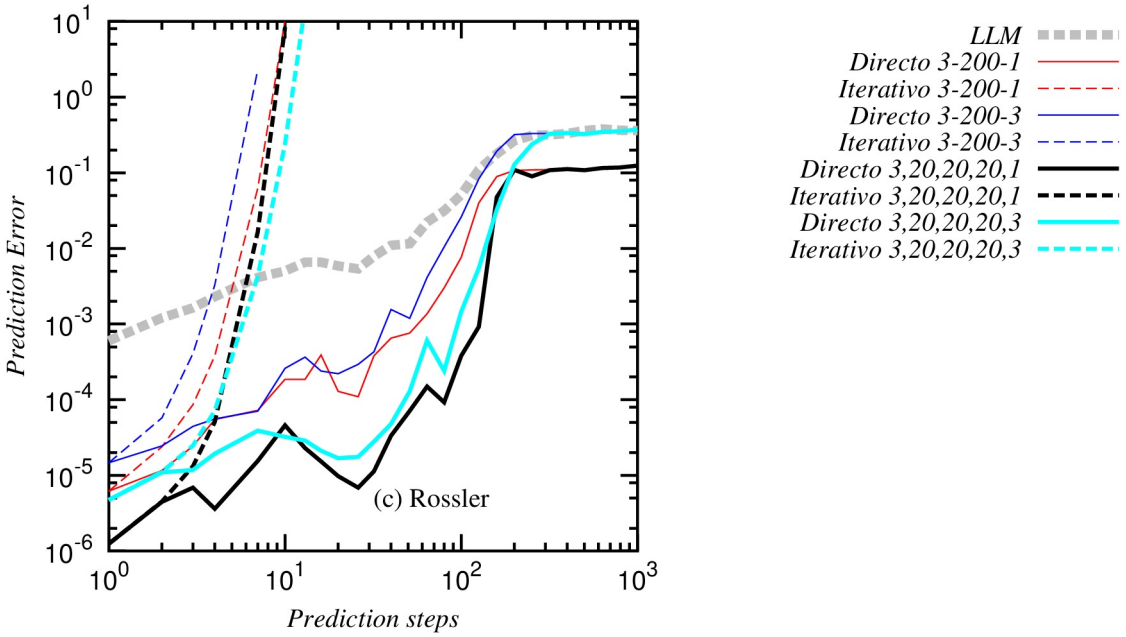

Figura 3.15: Serie temporal de Rössler. Error de predicción vs. Horizonte de predicción H, ambos en escala logarítmica

En la Figura 3.15, se puede observar que se obtiene un mejor resultado con una RN *profunda* (3-20-20-20-N) que con una RN *no-profunda* (3-200-N), donde $N \in \{1, 3\}$. En cuanto a los métodos *directo* e *iterativo*, se puede observar que el *directo* es superior, ya

que la predicción para el método *iterativo* toma valores infinitos al llegar a *H=10*. Esto puede estar relacionado con la forma del atractor de Rössler (Fig. 3.15) , donde la parte de la espiral (zona lineal del atractor), es relativamente fácil de predecir por un modelo con $H = 1$. Sin embargo la zona donde la órbita vuelve a re-inyectarse en la espiral requiiere observar horizontes más alejados para poder predecir su comportamiento.

Simulación de la dinámica

En este caso de estudio no podemos generar una serie temporal a partir de iterar ninguna de las redes (siguiendo la metodología propuesta en el capítulo 2) debido a que estas toman valores infinitos al llegar a *H=10*.

3.2.4. Circuito de Chua

En esta sección se muestra el uso de la metodología propuesta para la serie temporal del circuito de Chua. Esta serie temporal corresponde a mediciones de la corriente en el inductor realizadas por Aguirre *et al.* [30], cuyos datos están disponibles en [31]. Los datos presentan una cantidad considerable de ruido observacional debido a la resolución del conversor A/D y al sensor de efecto Hall utilizado para medir la corriente sobre el inductor [30]. Es el más simple circuito electrónico que satisface los criterios de comportamiento caótico.

Dado que la serie temporal contiene ruido, se procede a suavizar los datos con un filtro de Savitzky-Golay [32]. Este filtro tiene dos parámetros libres: el intervalo de la ventana donde se ajustan los polinomios y el orden de los mismos. Se utilizó [27, 0] (es decir que la ventana toma 27 valores hacia atrás y ninguno hacia adelante), y polinomios de orden $p = 2$ (parábola). Este filtro se puede obtener utilizando el siguiente comando del paquete TISEAN [33]:

```
$> sav_gol data.dat -n27,0 -p2
```

Para esta serie, el tamaño de ventana óptimo obtenido es $t_w = 81$ y su dimensión $m = 3$ [6], resultando $\tau = 27$ el tiempo de retraso[1].

En la Fig. 3.16 se muestra la serie temporal del circuito de Chua luego de ser suavizada con con un filtro de Savitzky-Golay. Luego en la Fig. 3.17 se grafica el atractor reconstruido a partir de la serie temporal, utilizando coordenadas de retraso, con tiempo de retraso $\tau = 27$ y dimension $m = 3$.

En la Fig. 3.18, tenemos las curvas de *Error de Predicción* (eje y) vs. *Horizonte de Predicción H* (eje x). Para esta serie a diferencia de las anteriores el mejor desempeño

[1] Para definir el tamaño de la ventana en este caso, se tuvo en cuenta que el filtro aplicado incorpora a cada observación información proveniente de 27 observaciones previas incrementando el tamaño efectivo de la ventana.

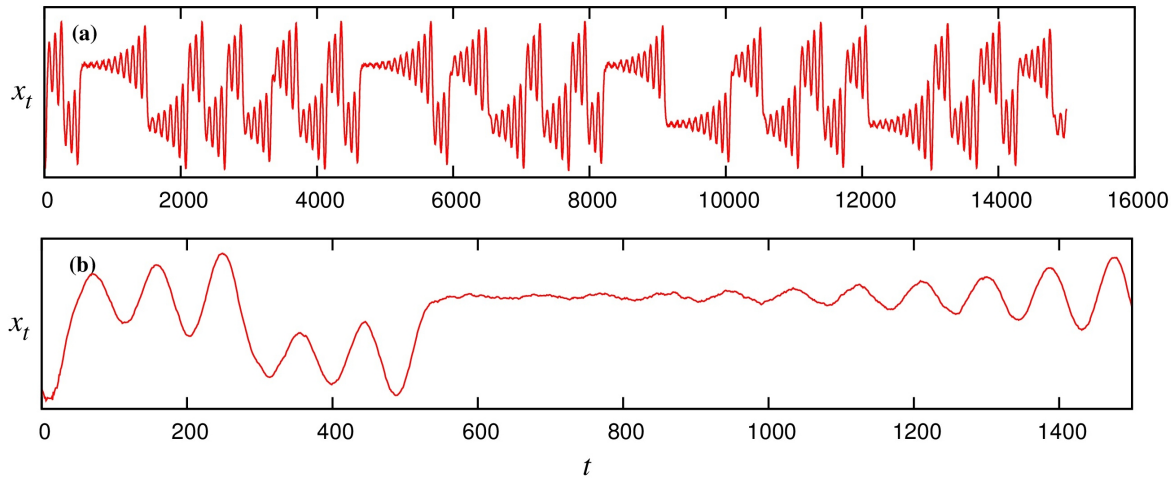

Figura 3.16: Serie temporal del circuito de Chua. (a) Serie temporal completa. (b) Primeros 1500 datos de la serie temporal, ambas suavizadas con un filtro de Savitzky-Golay.

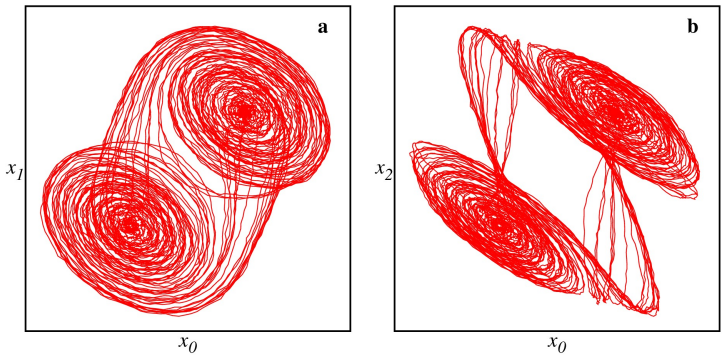

Figura 3.17: Reconstrucción serie de Chua suavizada con filtro Savitzky-Golay, conjunto de *test* 7.000 puntos. Tiempo de retraso 27 y dimensión de embedding 3.

se obtiene con un método *iterativo*. Sin embargo sigue siendo una red profunda la que presenta menor error en todo el rango de horizontes H. Las redes neuronales estudiadas se comportan de manera similar hasta $H = 10^1$, luego las curvas de error se separan y finalmente alcanzan valores similares a partir de $H = 400$. De los distintos comportamientos vemos que las RN (3-200-N) directas son las primeras en aumentar su error con el horizonte (en $H \approx 40$). En $H \approx 10^2$ diverge el error de las RN (3-20-20-20-N) directas. Las redes de *salida-simple* iteradas son las que alcanzan mayores horizontes manteniendo un bajo error, logrando la red profunda 3-20-20-20-1 iterada un error significativamente menor que el caso no-profundo en este rango. Finalmente las redes iteradas de *salida-múltiple* son las que peor desempeño presentan en las predicciones a largo plazo.

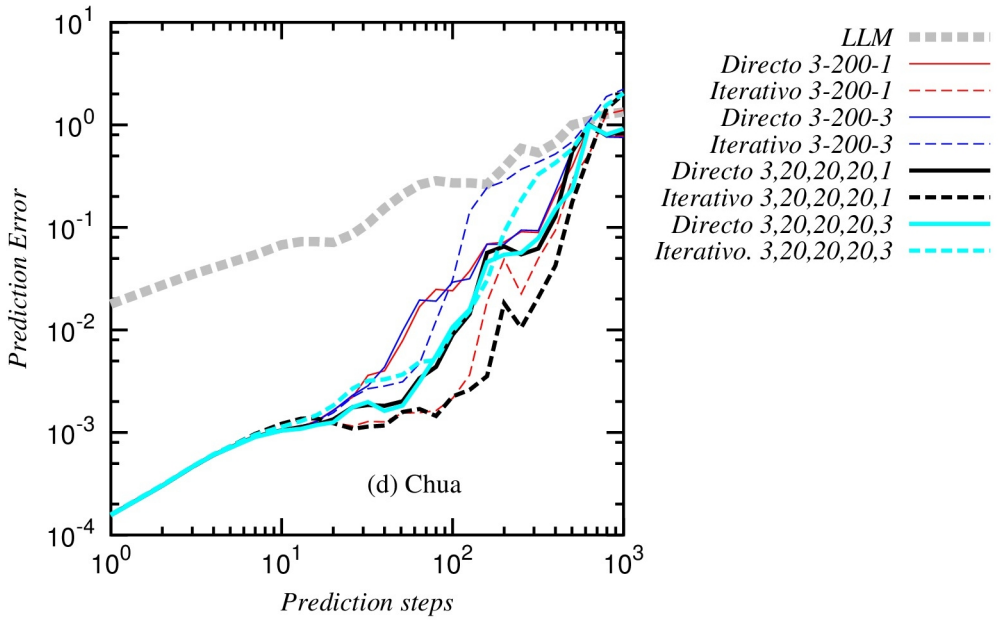

Figura 3.18: Serie temporal de Circuito de Chua. Error de predicción en función del horizonte H, ambos en escala logarítmica.

Simulación de dinámica

Comparamos el atractor original y los producidos por las RN con el método iterativo ($h = 1$) para las distintas arquitecturas de RN, y además la *salida-simple* y *salida-multiple*.

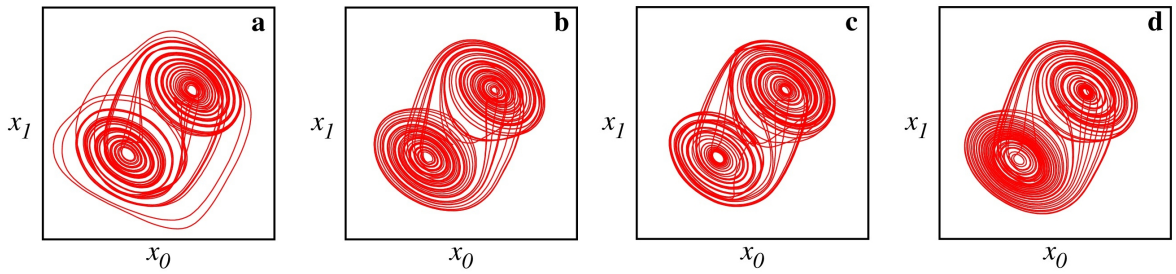

Figura 3.19: Datos generados por las redes neuronales. Se muestran las reconstrucciones (como en la Fig. 3.17 a) de la serie de Chua generada por las redes neuronales consideradas. Las predicciones fueron generadas con el método *iterativo* ($h = 1$): (a) con una RN de estructura 3-200-3 (b) con una RN de estructura 3-200-1 (c) RN de estructura 3-20-20-20-3 (d) RN de estructura 3-20-20-20-1. Para cada caso se muestra únicamente la proyección sobre las coordenadas x_0 y x_1, primera y segunda componentes de la reconstrucción respectivamente, $x(t)$ y $x(t-27)$.

En la Fig. 3.19 se observan las reconstrucciones de las series generadas con el método propuesto. En una RN, la primer capa oculta actúa como filtro pasabajo, limpiando el ruido de la señal, como se puede ver, se obtiene así una reconstrucción más suave que la original en la Fig. 3.17. Vemos que para los casos de *salida-simple* (paneles b y d) se tiene una dinámica más estable a comparación de la *salida-múltiple* (paneles a y c).

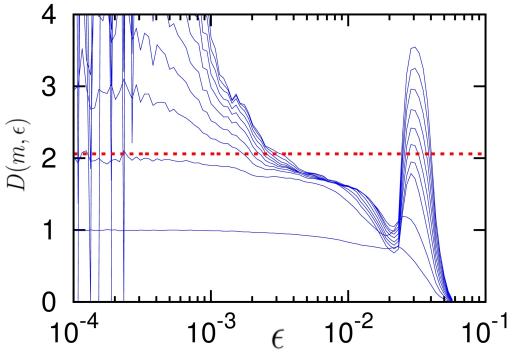

Figura 3.20: Curvas correspondientes a la dimensión de correlación de la serie temporal de Chua. Según Aguirre *et al.* [34] la dimensión de este atractor es 2.06 y este valor se indica con la línea de trazos roja.

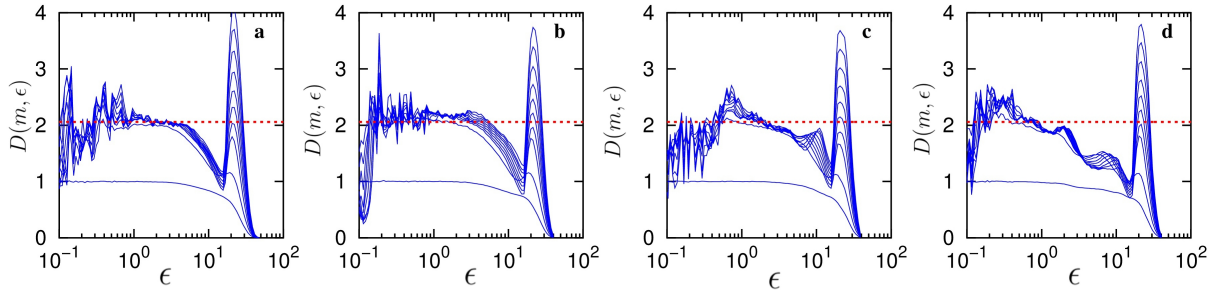

Figura 3.21: Dimensión de correlación de las reconstrucciones generadas por las RN, correspondientes a la figura (a) 3-200-3 (b) con una RN de estructura 3-200-1, (c) RN de estructura 3-20-20-20-3, (d) RN de estructura 3-20-20-20-1.

En la Fig. 3.21, tenemos las respectivas gráficas de dimensión de correlación de las reconstrucciones vistas en la Fig. 3.19. La línea roja en trazo interrumpido, nos da una guía aproximada de donde se encuentra la dimensión de correlación (d=2.06). Por lo visto, todas tienen dimensión fractal cercana a $d = 2$. Descartando las redes de *salida-múltiple* (paneles a y c) según lo analizado en el párrafo anterior resta comparar los paneles (b) y (d) correspondientes a las redes de *salida-simple*. Comparando con la Fig. 3.20 vemos que el comportamiento mas parecido al conjunto de test, corresponde a la RN *profunda* de estructura 3-20-20-20-1.

Capítulo 4

Conclusiones

En esta Tesina se evaluaron redes neuronales profundas para la predicción de series temporales caóticas. Se comparó su desempeño con el de las redes neuronales convencionales (de una única capa oculta) para realizar predicción en distintos horizontes barriendo desde el corto plazo hasta el límite de predecibilidad de cada serie. Los resultados sobre series temporales sintéticas y reales sugieren que las arquitecturas profundas superan a las redes neuronales convencionales para todo el rango de horizontes.

Se realizaron además experimentos para evaluar la dinámica capturada (o aprendida) por cada red neuronal. Con tal finalidad se compararon los atractores obtenidos y las curvas de estimación de la dimensión de correlación correspondientes a los mismos. Estos experimentos dieron como resultado que resulta una dinámica más parecida a la original utilizando iteraciones de redes con *salida-simple*. En términos de la profundidad de las mismas, el resultado depende de las series consideradas. En cuanto al tiempo de ejecución, al considerar redes profundas con igual cantidad de parámetros que las redes no-profundas, no representó una diferencia considerable, por lo tanto vale considerar el uso de estas arquitecturas para la tarea de predicción. Con respecto a la *salida-múltiple*, no se encontró beneficio frente a un modelo con *salida-simple*.

Finalmente, respecto al desempeño de métodos iterativos vs. directos (discusión inicialmente abordada en [12]), no se pueden sacar mayores conclusiones de los resultados obtenidos ya que para las series sintéticas los métodos directos presentaron menor error y lo contrario ocurrió para la serie real del circuito de Chua.

Trabajos futuros

En relación al párrafo anterior resta verificar si la diferencia de comportamiento entre las series sintéticas y la serie real se debe a la presencia de ruido en esta última. Para identificar esto, una posibilidad es repetir los experimentos agregando ruido a las series sintéticas. Esto permitiría verificar si el método iterativo en presencia de ruido supera al método directo. Esto implicaría que predecir a un horizonte de largo plazo en

forma directa es resolver un problema no-lineal de mayor dificultad y mayor error que el que se genera al amplificar el ruido de la misma serie temporal al iterar predicciones de corto plazo. Esta conclusión fue la obtenida por [12] utilizando modelos locales polinómicos.

Como tarea pendiente, también queda evaluar la metodología utilizando el vector de coordenadas de retraso completo que contiene todas las observaciones dentro de la ventana ($\tau = 1$ y $m = t_w + 1$). Esta elección implica aumentar la cantidad de neuronas de entrada y por lo tanto la cantidad de parámetros a entrenar. Se realizaron pruebas preliminares y se obtuvo mejor resultado con la serie de Rössler en cuanto a la estabilidad de la dinámica de predicción. Se pudieron obtener modelos iterativos estables (que no presentan la divergencia observada en la Fig. 3.15).

Otro trabajo interesante, en la dirección de obtener dinámicas más estables, sería utilizar la metodología propuesta en [35], denominada *denoising autoencoders*. Esta metodología consiste en agregar un cierto nivel de ruido en la entrada de la red al momento del entrenamiento pero no en los datos de salida. Esto permitiría que la red neuronal aprenda a filtrar el ruido y conservar la dinámica dentro del atractor.

Apéndice A

Series temporales sintéticas

A.1. Serie de Mackey-Glass

Se consideró la ecuación de Mackey-Glass [36]:

$$\frac{dx}{dt} = \frac{ax(t-\overline{\tau})}{1+x^c(t-\overline{\tau})} - bx$$

con parámetros $a = 0{,}2$, $b = 0{,}1$, $c = 10$, y $\overline{\tau} = 17$. Se utilizó como condiciones iniciales $x(t < 0) = 0$ y $x(t = 0) = 1{,}2$ y se integró numéticamente esta ecuación diferencial de dimensión infinita. Se descartaron las primeras 2000 iteraciones. Finalmente se submuestreó la serie con un tiempo de muestreo $\delta t = 0{,}5$. La serie resultante tiene 20000 elementos de los cuales los primeros 10000 son los utilizados al implementar los métodos de reconstrucción y se grafican en la Fig. 3.1 y los restantes 10000 son guardados para prueba de los algoritmos.

A.2. Serie de Lorenz

El sistema de Lorenz [37] se describe con las siguiente ecuaciones

$$\begin{aligned} dx/dt &= \sigma(y-x), \\ dy/dt &= x(\rho-z)-y, \\ dz/dt &= xy-\beta z \end{aligned}$$

cuyos parámetros son $\sigma = 10$, $\rho = 28$ y $\beta = 8/3$. Se inicializó el sistema en $x = y = 1$, $z = 50$ y se integrró numéricamente con algoritmo de Runge-Kutta de $4^{\underline{o}}$ orden con paso $\delta t = 0{,}01$. La serie corresponde a la secuencia de valores de la coordenada x. Luego de un transiente de 1000 puntos, se conservan 10000 puntos para implementación y 10000 para prueba. Los puntos utilizados para la implementación de los métodos de

reconstrucción se grafican en la Fig. 3.7.

A.3. Serie de Rössler

Para el sistema de Rössler [38] se integraron las ecuaciones

$$\begin{aligned} dx/dt &= -(y+z), \\ dy/dt &= x+\alpha y, \\ dz/dt &= \beta + z(x-\gamma) \end{aligned}$$

con parámetros $\alpha = 0{,}15$, $\beta = 0{,}2$ y $\gamma = 10$. Las condiciones iniciales consideradas son $x = y = z = 1$. Se utilizó un algoritmo de Runge-Kutta de 4^{o} orden para la integración numérica de las ecuaciones. Se descartaron los primeros 8000 puntos y se consideró la coordenada x con una frecuencia de muestreo $\delta t = 0{,}125$ para generar una serie de 10000 puntos para implementación y otros 10000 para prueba. Los puntos utilizados para la implementación de los métodos de reconstrucción se grafican en la Fig. 3.13.

Bibliografía

[1] Takens, F. Detecting strange attractors in turbulence. En: D. Rand, L.-S. Young (eds.) Dynamical Systems and Turbulence, Warwick 1980, tomo 898 de *Lecture Notes in Mathematics*, págs. 366–381. Springer Berlin / Heidelberg, 1981. 4

[2] Mañé, R. On the dimension of the compact invariant sets of certain non-linear maps. En: D. Rand, L.-S. Young (eds.) Dynamical Systems and Turbulence, Warwick 1980, tomo 898 de *Lecture Notes in Mathematics*, págs. 230–242. Springer Berlin / Heidelberg, 1981. 4

[3] Sauer, T., Yorke, J., Casdagli, M. Embedology. *Journal of Statistical Physics*, **65** (3-4), 579–616, 1991. 4

[4] Casdagli, M., Eubank, S., Farmer, J. State space reconstruction in the presence of noise. *Physica D: Nonlinear Phenomena*, **51**, 52–98, 1991. 5

[5] Uzal, L. C., Grinblat, G. L., Verdes, P. F. Optimal reconstruction of dynamical systems: A noise amplification approach. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, **84** (1), 2011. 5, 17, 25

[6] Uzal, L. C. Tesis de doctorado en física, técnicas de reconstrucción de sistemas dinámicos, 2012. 5, 17, 25, 27, 30, 33, 35

[7] Kantz, H., Schreiber, T. Nonlinear time series analysis, tomo 7 de *Cambridge Nonlinear Science Series*. Cambridge: Cambridge University Press, 1997. 5

[8] Grassberger, P., Procaccia, I. Characterization of strange attractors. *Physical Review Letters*, **50**, 346–349, ene. 1983. 5

[9] Grassberger, P., Procaccia, I. Measuring the strangeness of strange attractors. *Physica D Nonlinear Phenomena*, **9**, 189–208, oct. 1983. 5

[10] Shumway, R. H., Stoffer, D. S. Time Series Analysis and Its Applications: with R examples. Springer, 2000. 6

[11] Solar Influences Data Analysis Center (SIDC). Monthly and monthly smoothed sunspot number. URL URLhttp://sidc.oma.be/DATA/monthssn.dat. 7

[12] Farmer, J. D., Sidorowich, J. J. Exploiting chaos to predict the future and reduce noise. *Evolution, learning, and cognition*, pág. 277, 1988. 8, 39, 40

[13] Nair, V., Hinton, G. E. Rectified linear units improve restricted boltzmann machines. págs. 807–814, 2010. 9

[14] Krizhevsky, A., Sutskever, I., Hinton, G. E. Imagenet classification with deep convolutional neural networks. págs. 1106–1114, 2012. 9, 10, 13, 19

[15] Hinton, G. E. e. a. Improving neural networks by preventing co-adaptation of feature detectors arxiv preprint arxiv:1207.0580 (2012), 2012. 10, 13, 19

[16] Ben Taieb, S., Bontempi, G., Atiya, A. F., Sorjamaa, A. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Syst. Appl.*, **39** (8), 7067–7083, jun. 2012. URL http://dx.doi.org/10.1016/j.eswa.2012.01.039. 10

[17] Fletcher, R. Practical methods of optimization; (2nd ed.). New York, NY, USA: Wiley-Interscience, 1987. 12

[18] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, **2**, 303–314, 1989. 12

[19] Bengio, Y. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, **2** (1), 1–127, 2009. 12

[20] Hinton, G. E., Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, **313** (5786), 504–507, jul. 2006. URL http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google. 12

[21] Hinton, G. E., Osindero, S., Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, **18** (7), 1527–1554, 2006. 12

[22] rahman Mohamed, A., Sainath, T. N., Dahl, G. E., Ramabhadran, B., Hinton, G. E., Picheny, M. A. Deep belief networks using discriminative features for phone recognition. En: ICASSP, págs. 5060–5063. IEEE, 2011. 12

[23] rahman Mohamed, A., Hinton, G. E., Penn, G. Understanding how deep belief networks perform acoustic modelling. En: ICASSP, págs. 4273–4276. IEEE, 2012. 12

[24] Cook, J., Sutskever, I., Mnih, A., Hinton, G. E. Visualizing similarity data with a mixture of maps. *Journal of Machine Learning Research - Proceedings Track*, **2**, 67–74, 2007. 13

[25] Taylor, G. W., Hinton, G. E., Roweis, S. T. Modeling human motion using binary latent variables. En: NIPS, págs. 1345–1352. MIT Press, 2006. 13

[26] Hinton, G. E., Osindero, S., Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, **18** (7), 1527–1554, 2006. 13

[27] Jaitly, N., Hinton, G. E. Learning a better representation of speech soundwaves using restricted boltzmann machines. págs. 5884–5887, 2011. 17

[28] ALGLIB. Bochkanov, S. and Bystritsky, V. URL http://www.alglib.net/. 20

[29] Farmer, J. Chaotic attractors of an infinite-dimensional dynamical system. *Physica D: Nonlinear Phenomena*, **4** (3), 366–393, 1982. 29

[30] Mendes, L., Aguirre, G., Rodrigues, E. Nonlinear identification and cluster analysis of chaotic attractors from a real implementation of Chua's circuit. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, **7** (6), 1411–1423, 1997. 35

[31] MACSIN. Research Group. Chua's circuit meassured data. URL http://www.cpdee.ufmg.br/~MACSIN/services/data/dsil.zip. 35

[32] Persson, P.-O., Strang, G. Smoothing by savitzky-golay and legendre filters. En: J. Rosenthal, D. Gilliam (eds.) Mathematical systems theory in biology, communications, computation and finance, tomo 134 de *IMA volumes in mathematics and its applications*, págs. 301–315. Springer, 2003. 35

[33] Hegger, R., Kantz, H., Schreiber, T. Practical implementation of nonlinear time series methods: The TISEAN package. *Chaos*, **9** (2), 413–435, 1999. 35

[34] Aguirre, L. A., Maquet, J., Letellier, C. Induced one-parameter bifurcations in identified nonlinear dynamical models. *International Journal of Bifurcation and Chaos*, **12** (01), 135–145, 2002. 38

[35] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders, 2008. 40

[36] Mackey, M., Glass, L. Oscillation and chaos in physiological control systems. *Science*, **197** (4300), 287–289, 1977. 41

[37] Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.*, **20**, 130–141, 1963. 41

[38] Rössler, O. An equation for continuous chaos. *Physics Letters A*, **57** (5), 397–398, 1976. 42