Universitat de les Illes Balears

Facultat de Ciències

Memòria del Treball de Fi de Grau

Máquinas de aprendizaje y aplicaciones

Alejandro Morán Costoya

Grau de Física

Any acadèmic 2015-16

DNI de l'alumne: 43190889V

Treball tutelat per Claudio R. Mirasso Santos
Departament de Física

S'autoritza la Universitat a incloure aquest treball en el Repositori Institucional per a la seva consulta en accés obert i difusió en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor		Tutor	
	Sí	No	Sí	No
	x		x	

Paraules clau del treball:
Aprendizaje artificial, *reservoir computing*, sistemas dinámicos, retraso temporal, memoria, clasificación.

Resumen

Este trabajo trata de dar una visión general sobre la técnica de *reservoir computing*, una técnica de *machine learning* o máquinas de aprendizaje con la cual el sistema aprende a resolver ciertas tareas a base de ejemplos. En particular hemos utilizado como máquina de aprendizaje un sistema dinámico no lineal acoplado con retraso, partiendo de trabajos de investigación recientes [1], [2], [3]. Parte del trabajo ha consistido en adaptar un programa desarrollado en MATLAB por el Dr. Miguel Cornelles Soriano a ciertas tareas, así como comparar los resultados obtenidos empleando regresiones lineales y logísticas en tareas de clasificación.

Tras una breve introducción se explicará el funcionamiento de la máquina de aprendizaje utilizada. El resto del trabajo consiste en poner dicha máquina a prueba mediante la evaluación de tareas sencillas: memoria, predicción de una serie temporal, clasificación bidimensional binaria y múltiple y clasificación binaria de electrocardiogramas. En todos los casos se explora el rango de parámetros que permiten realizar correctamente cada una de dichas tareas.

Mi contribución al programa original desarrollado por el Dr. Cornelles incluye la incorporación de modelos lineales generalizados y clasificación múltiple, así como la definición de tareas sencillas que permiten evaluar la diferencia entre distintos modelos de regresión para el aprendizaje.

Índice

1. Introducción

Actualmente y cada vez más, empleamos ordenadores u otros dispositivos para modelar experiencias reales y ver en qué medida se ajustan nuestros modelos a la realidad. De este modo ampliamos nuestro conocimiento del mundo y podemos aprovechar dicha información para hacer predicciones o crear nuevas aplicaciones. Si lo que uno trata de modelar es la inteligencia (capacidad de generalizar e interpretar, razonamientos lógicos, toma de decisiones, ...) entonces nos podemos estar refiriendo al comportamiento biológico de conjuntos de neuronas en el procesamiento de la información para diferentes tareas, o a la inteligencia artificial (IA), no necesariamente inspirada en sistemas biológicos.

Una de las ramas de la IA emplea máquinas de aprendizaje (del inglés, *machine learning*). Se trata de sistemas capaces de aprender a base de ejemplos. Dentro de esta disciplina nos centraremos en las redes de neuronas artificiales, un paradigma inspirado en las neuronas de los sistemas nerviosos de ciertos animales. Comenzaremos dando una visión muy general de estos conceptos para poner en contexto el caso particular que ocupa el trabajo: *Reservoir Computing* basado en un solo nodo acoplado a una línea de retraso.

1.1. *Machine learning*

Recientemente esta disciplina ha hecho posibles numerosos avances, como la circulación en carretera de coches autónomos [4], entrenamiento de prótesis basadas en sensores neuronales [5], la clasificación de diagnósticos médicos [6], el procesamiento del lenguaje hablado [7], la predicción de series temporales en economía y otras disciplinas, etc.

En *machine learning* uno trata de conseguir comportamientos en la máquina que no han sido programados explícitamente. Un sistema explícitamente programado tiene una estructura interna fija y es muy eficiente realizando cálculos matemáticos, sin embargo no lo es tanto en tareas más complejas tales como reconocer una cara conocida, identificar y manipular objetos o procesar una conversación. En general para las tareas de reconocimiento de patrones nuestro cerebro es más eficiente. Uno puede plantearse resolver este tipo de tareas con técnicas de *machine learning*, en lugar de hablar de arquitecturas de Von Neumann/Turing (programación explícita), características de sistemas expertos. Por *machine learning* nos referimos a arquitecturas de red [8], que tras una fase de entrenamiento (ajuste de parámentros de la red) deben ser algoritmos capaces de generalizar el resultado para un estímulo similar. Este método no compite con las arquitecturas de programación explícita en lo que a cálculos aritméticos se refiere, de hecho normalmente utilizamos métodos de programación explícita para preprocesar la información de entrada y supervisar algoritmos de machine learning (etiquetado, test, error, etc.).

Una de las complicaciones de esta manera de proceder es la necesidad de implementar dos algoritmos, uno para generar la red y otro para entrenarla, siendo por lo general más complejo lo segundo. El entrenamiento o aprendizaje puede ser supervisado o no supervisado.

1.1.1. Aprendizaje supervisado

En este caso los datos entrantes están etiquetados y se provee al sistema una respuesta correcta durante el entrenamiento. Típicamente nos enfrentamos dos tipos de tareas: regresión y clasificación. Ilustremos cada caso con un ejemplo:

- Tarea de regresión: Un ejemplo de regresión podría ser el de tratar de aproximar el producto interior bruto en un instante temporal t_0 a partir de cierta cantidad de datos correspondientes a $t < t_0$. Se trata de aprendizaje supervisado porque conocemos los puntos (t, $PIB(t)$). Las variables que intervienen son continuas.

- Tarea de clasificación: Un ejemplo de clasificación podría ser el reconocimiento de números escritos a mano, donde las variables son discretas y nos referimos a ellas como categorías. Imaginemos que disponemos de una base de datos con números escritos por muchas personas diferentes y etiquetados correctamente (aprendizaje supervisado). Características como el tamaño de la imagen no se tienen en cuenta (tras digitalizarla se redimensionaliza) y lo más habitual es tener en cuenta características como el área, la posición del centroide, la longitud del eje mayor, la longitud del eje menor, etc. En definitiva, lo ideal es tener en cuenta tantas características como sea necesario para que las categorías sean linealmente separables e identificar las características más relevantes para dicho propósito. A continuación, en la figura 1 se muestra un ejemplo de la distribución de probabilidades para un clasificador bidimensional de grupos de puntos símplemente conexos pero sin una clara separación entre ellos. Han sido empleados cinco métodos distintos para ilustrar cualitativamente que diferentes métodos generan diferentes distribuciones. Dichos métodos son: *random forest*, *support vector machine*, *nearest neighbours*, *naive Bayes* y una red de neuronas artificiales.

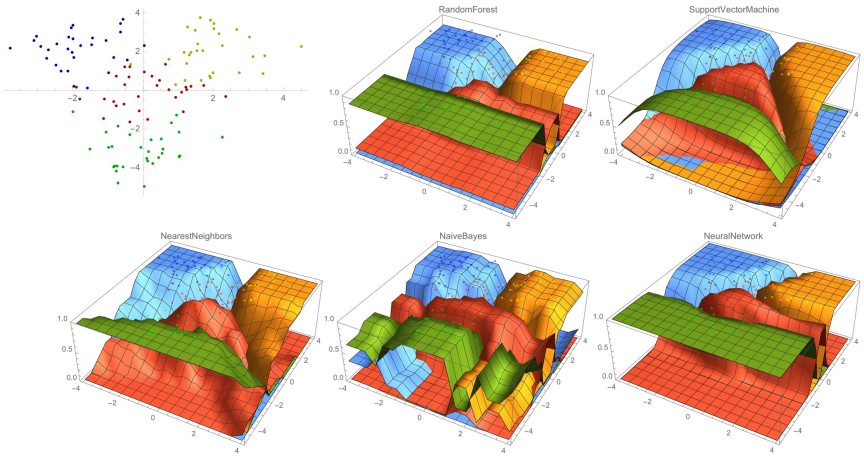


Figura 1: Distribuciones de probabilidad generadas por diferentes métodos de *machine learning* [9].

En otras tareas de clasificación uno puede tener en cuenta como parámetros de entrada muchas más características, normalmente el método de *machine learning* escogido depende del número de características y de la tarea que se trate. En ocasiones lo que se hace es evaluar una tarea con clasificadores distintos para ver cual arroja mejores resultados [10].

1.1.2. Aprendizaje no supervisado

En este caso disponemos de datos de entrada no etiquetados y el algoritmo está diseñado para buscar patrones. Un ejemplo de aplicación puede ser buscar el contorno de un objeto en una imagen o la compresión de datos, entre otros. Además es habitual combinar aprendizaje no supervisado y supervisado para que el error decaiga más rápidamente en tareas de regresión y clasificación. En este trabajo no trataremos ningún caso de aprendizaje no supervisado.

Existen numerosos tipos de máquinas de aprendizaje además de sus variantes. De aquí en adelante trataremos con un tipo más específico: las redes de neuronas artificiales (RNA o ANN por sus siglas en inglés).

1.2. Redes de neuronas artificiales

Basadas en el funcionamiento de las redes neuronales biológicas, las ANNs procesan la información proyectándola en una red de elementos interconectados llamados nodos. Estos nodos y sus conexiones son los análogos artificiales de las neuronas y sus sinapsis, respectivamente.

En cuanto al sistema biológico, la intensidad de una sinapsis no es fija, sino que puede ser modificada en base a la información del medio, de modo que la estructura del cerebro es dinámica y se va modificando por la formación de nuevas conexiones (excitadoras o inhibidoras), la destrucción de conexiones, la modificación de la intensidad de la sinapsis, o incluso por muerte neuronal. Desde un punto de vista funcional las neuronas son procesadores de información sencillos, constan de un subsistema de entrada (dendritas), un núcleo de procesamiento (soma) y un subsistema de salida (axón).

Evidentemente, los nodos de una ANN no presentan todos los comportamientos que puede presentar una neurona real con exactitud pero sí hay métodos con más similitudes biológicas que otros, como puede ser el caso de las redes de neuronas de impulsos [11], [12].

El funcionamiento general es el siguiente (ver figuras 2 y 3): dado un estímulo (información entrante), almacenado en los nodos de la capa de entrada, éste se proyecta en las capas ocultas (puede haber más de una). Algunas neuronas de las capas ocultas (o todas) se conectan con la capa de salida, que es donde se almacena la respuesta del sistema. Las conexiones hacia un nodo determinan la contribución del mismo y cada conexión tiene asignado un peso, es decir, la respuesta es una transformación no lineal del estímulo y diferentes distribuciones de pesos representan transformaciones diferentes.

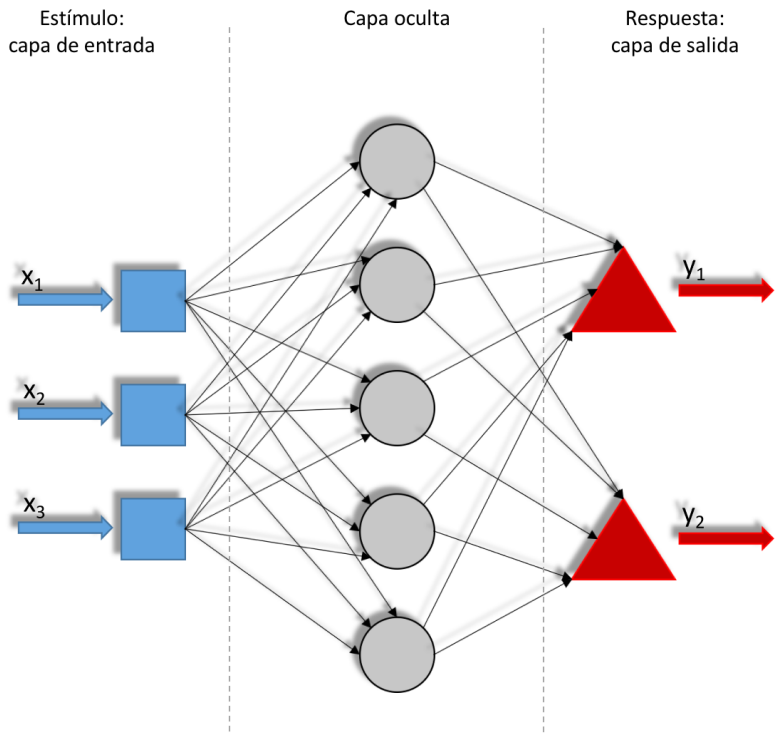


Figura 2: Representación de una red de neuronas artificiales con una sola capa oculta y sin realimentaciones.

El caso más general puede incluir retroalimentación en los nodos, es decir, conexiones que se propaguen hacia nodos anteriores o conexiones desde un nodo hacia el mismo. La figura 3 representa esquemáticamente esta situación. Nos referimos ahora a redes neuronales recurrentes (RNN).

En el caso del aprendizaje supervisado la tarea que se debe llevar a cabo en la fase de entrenamiento de la red es un ajuste de pesos que minimice el error en la respuesta, un problema de optimización donde debemos encontrar una transformación óptima del estímulo. Aunque tene-

mos claro cual es nuestro problema, en el caso de las RNNs el coste computacional que requiere entrenar los pesos de la red puede ser elevado. Aún así, como se espera que al introducir recurrencias (que se traducen en memoria) en la red ésta mejore el rendimiento en ciertas tareas tras la fase de aprendizaje, vamos a esquivar esta complicación ajustando únicamente los pesos correspondientes a la capa de salida (técnica conocida como *reservoir computing*). Además una RNN puede conseguir mejores resultados con menos recursos debido a esta característica de memoria, al compararse con una FNN (*feed-forward neural network*, figura 2). A continuación se explican las bases del método de *reservoir computing* (RC).

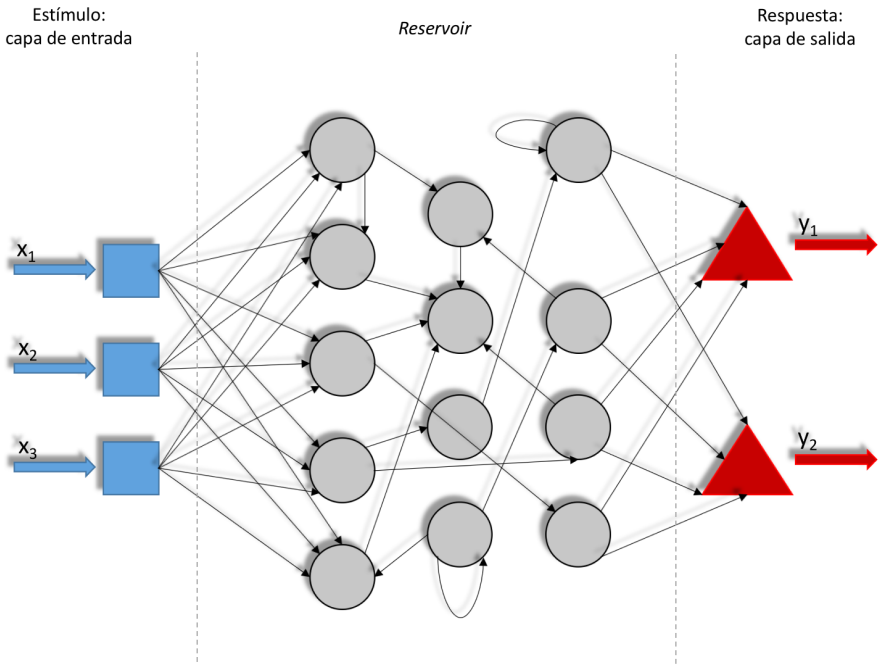


Figura 3: Representación de una red neuronal recurrente.

1.3. *Reservoir computing*

La optimización de una RNN puede requerir un coste computacional elevado debido a problemas como la cantidad de pesos a ajustar, inicialización, que el algoritmo no sea convergente, convergencia prematura, etc. La técnica que usaremos en este trabajo para resolver algunas tareas es conocida como *reservoir computing* y permite esquivar algunos problemas ya mencionados. Esta técnica permite ajustar los pesos con muchos menos recursos a cambio de mantener una zona, llamada *reservoir*, con sus pesos fijos. El modo de proceder en RC es el siguiente. Dada la topología de la red[1], inicializar los pesos de manera aleatoria/pseudoaleatoria[2] y llevar a cabo la fase de aprendizaje modificando únicamente los pesos que comunican el *reservoir* con la capa de salida. En la figura 3 son los pesos de las conexiones intersectadas por la línea discontinua de la derecha. El resto permanecen fijos con sus valores aleatorios iniciales.

Un *reservoir computer* debe tener las siguientes propiedades para realizar tareas correctamente una vez haya pasado la fase de aprendizaje:

- Aproximación: Sean $(\mathbf{X}, \mathbf{Y})$ los datos de entrada y salida, respectivamente. Se debe cumplir que para unos datos de entrada casi idénticos, la respuesta debe de ser casi la misma.

[1]La topología de la red es la disposición espacial de los nodos y sus conexiones.
[2]Podríamos utilizar, por ejemplo, *linear feedback shift registers* (LFSR) [13].

$$(\mathbf{X} \rightarrow \mathbf{Y}) \Leftrightarrow (\mathbf{X} + \Delta\mathbf{X} \rightarrow \mathbf{Y} + \Delta\mathbf{Y}) \tag{1}$$

- Separación: Aunque en general se cumpla la propiedad anterior, la distancia entre los estados del *reservoir* debe ser suficientemente grande para distinguir dos entradas similares pero que pertenecen a categorías distintas.

- Memoria: Una red recurrente presenta retroalimentaciones entre sus nodos, de manera que la respuesta del *reservoir* a una señal de entrada conserva información de señales anteriores y esto es algo que se ve reflejado en la capa de salida. Por eso a esta característica se le llama memoria y dicha información conservada se va perdiendo, es decir, la memoria referida a salidas anteriores se va perdiendo más cuanto mayor es la diferencia de tiempo.

2. *Reservoir computing* basado en un solo nodo acoplado con retraso

Anteriormente se ha introducido el término *reservoir computer* como un tipo de RNN. Sin embargo, veremos que el número de nodos operativos no tiene porque ser el número de nodos físicos (como hemos supuesto hasta ahora) si el comportamiento de un nodo depende del mismo en un instante de tiempo anterior. Veremos que un solo nodo no lineal es suficiente para conseguir las propiedades deseadas en RC. La implementación teórica y experimental de este equivalente a una RNN (en la que no hay realimentación desde los nodos de la capa de salida) fue llevada a cabo con éxito en 2011 (Appeltant et al. [14]) y es interesante por la simplicidad del esquema. RC basado en un solo nodo con retraso tiene prestaciones similares a una RNN.

En la figura 4 se muestra un esquema general del sistema estudiado y la metodología seguida. Se representan varios nodos virtuales separados por un intervalo de tiempo θ, equidistantes entre sí, que son imágenes del nodo no lineal (NL) sujeto a retraso temporal $\tau = N\theta$, donde N es el número de nodos virtuales. Primero la información de entrada es preprocesada con el fin de proyectarla en el *reservoir* a través de un solo canal de entrada. A continuación, la entrada provoca una perturbación en dicho nodo, que inicialmente se encuentra en estado estacionario, de modo que los nodos virtuales reflejen dicha perturbación en diferentes instantes de tiempo durante cada vuelta de duración τ. Finalmente, la capa de salida proporciona una respuesta en función del estado del *reservoir*. De aquí en adelante se muestra una descripción más completa y justificada de cada paso.

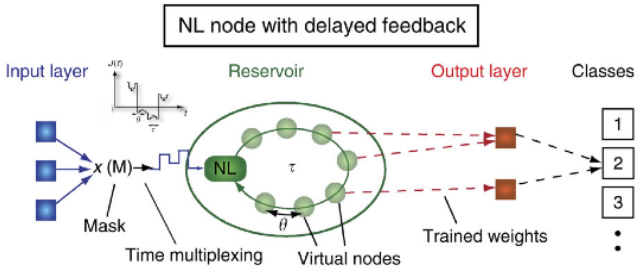


Figura 4: Esquema de RC basado en un solo nodo acoplado con retraso [14].

2.1. Preprocesado en la capa de entrada

Se podría emplear cada entrada de información directamente como entrada del nodo NL, pero si la señal es discretizada en pasos de tiempo τ, y este es suficientemente grande el último no-

do virtual podría encontrarse en el estado estacionario y por tanto no proporcionaría ninguna información. Con tal de evitar esto todas las señales de entrada son preprocesadas combinándolas con una máscara, de modo que la señal que recibe el nodo NL quede discretizada en pasos de tiempo θ y codificada por dicha máscara, por tanto si este parámetro de discretización es suficientemente pequeño los nodos virtuales del *reservoir* nunca se encontrarán en estado estacionario y podremos observar la evolución temporal transitoria en cada nodo virtual, cuantos más haya más información de la evolución temporal del sistema. Sin embargo, si θ es demasiado pequeño el nodo NL no tiene tiempo suficiente para reaccionar a la señal preprocesada. Para obtener prestaciones adecuadas se ha determinado empíricamente que $\theta = 0.2T$ [14], donde T es la escala de tiempo catacterística de dicho nodo.

Supongamos que tenemos M muestras de una entrada de datos de D dimensiones, es decir, una entrada original $\boldsymbol{u}_t$ de dimensiones $M \times D$. Lo que hacemos en nuestro caso es multiplicar por una máscara aleatoria binaria[3] (± 1) de dimensiones $D \times N$. Por ejemplo, en el caso $D = 1$, que es el más fácil de visualizar, tendríamos una entrada $\boldsymbol{u}_t$ que símplemente sería un vector que representa valores escalares de una función para diferentes tiempos. En la figura 5 se ilustra este caso particular, donde la máscara ha sido representada con un diagrama de barras. Nótese que dicha máscara es la misma cada periodo de inyección τ. En general, la matriz de entrada se obtiene como:

$$\boldsymbol{I} = \boldsymbol{u}_t \cdot \boldsymbol{Mask} \tag{2}$$

donde, en nuestro caso particular, tenemos en cuenta desviaciones de la entrada original respecto de su valor medio y en lugar de $\boldsymbol{u}_t$ empleamos $\frac{\boldsymbol{u}_t - \langle \boldsymbol{u}_t \rangle}{\sigma[\boldsymbol{u}_t - \langle \boldsymbol{u}_t \rangle]}$, siendo σ la desviación estándar. Finalmente la entrada secuencializada se obtiene reordenando las filas de la matriz de entrada en una sola. La matriz de entrada es:

$$\boldsymbol{I} = \begin{pmatrix} I_{11} & \dots & I_{1N} \\ \vdots & \ddots & \vdots \\ I_{M1} & \dots & I_{MN} \end{pmatrix} \tag{3}$$

por tanto:

$$\boldsymbol{J} = (I_{11}, \dots, I_{1N}, I_{21}, \dots, I_{2N}, \dots, \dots, I_{M1}, \dots, I_{MN}) \tag{4}$$

donde $\boldsymbol{J}$ es la entrada secuencializada en forma de vector de longitud MN. También puede verse como una función temporal discretizada en pasos de tiempo k de duración θ que denotamos como J_k o $J(t)$ constante entre pasos, donde $t = k\theta$, $k = 1, \dots, MN$.

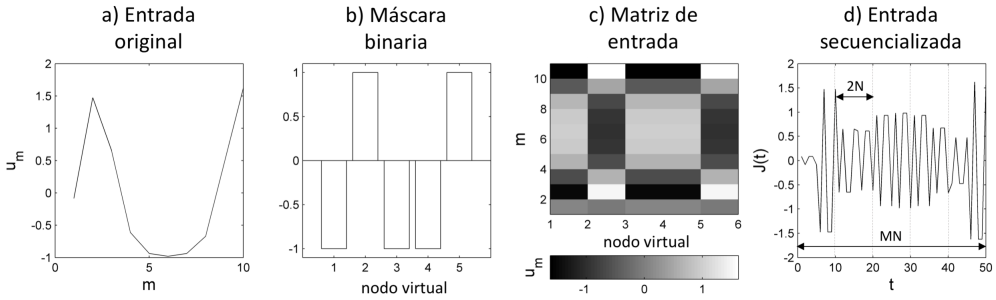


Figura 5: Preprocesado de la señal de entrada para el caso $D = 1$, $M = 10$, $N = 5$.

[3]La máscara aleatoria no tiene que ser necesariamente binaria pero se ha escogido así por ser la más sencilla.

2.2. *Reservoir*

Sea $x(t)$ la respuesta del nodo NL en un instante de tiempo t. Lo que ocurre en este sistema es que la variación de x con el tiempo no depende únicamente de $x(t)$ sino también de los valores de x en un intervalo continuo $[t-\tau, t]$, además de la entrada secuencializada $J(t)$ definida en el apartado anterior. De manera general, adimensionalizando el tiempo (unidades en las que $T = 1$):

$$\frac{dx(t)}{dt} = F(x(t), x(t-\tau), J(t)) \in \mathbb{R} \tag{5}$$

que es la forma general de una ecuación diferencial con retraso teniendo en cuenta la dependencia explícita con $J(t)$. En [14] se considera la siguiente forma para la ecuación anterior:

$$\frac{dx(t)}{dt} = -x(t) + f(x(t-\tau), J(t)) \in \mathbb{R} \tag{6}$$

A partir de las ecuaciones (5) o (6) se pueden comentar cualitativamente las propiedades que se deben satisfacer en RC (apartado 1.3). Si f es tal que la respuesta x tiene un comportamiento caótico, entonces ésta será sensible a pequeñas variaciones de J, favoreciendo así la propiedad de separación. Pero si la respuesta es demasiado caótica no se cumplirá la propiedad aproximación y además también afectaría negativamente a la propiedad de memoria para τ suficientemente grande. En éste y otros tipos de sistemas se ha llegado a la conclusión de que trabajar al borde del caos es óptimo para realizar ciertas tareas [11].

La ecuación (6), con distintas formas de f, ha servido para realizar tareas de RC. Por ejemplo, se han usado las funciones correspondientes al atractor de Mackey-Glass con realimentación retrasada (implementación tanto en software como electrónicamente [3], [1]) y una ecuación tipo atractor de Ikeda de realimentación retrasada (implementación opto-electrónica [15], [2]). Este trabajo se centra en las propiedades del modelo tipo Ikeda:

$$\frac{dx(t)}{dt} = -x(t) + \beta \sin^2[x(t-\tau) + \gamma J(t) + \phi] \tag{7}$$

donde β es el grado de no linealidad, γ es un parámetro que controla la señal de entrada y ϕ la fase. Es una de las no linealidades más sencillas y se ha usado mucho en óptica.

2.2.1. Obtención de la respuesta dinámica

Para determinar la transformación de la entrada en el nodo no lineal a lo largo de la línea de retraso temporal se ha empleado el método de Euler de integración numérica en la ecuación (7); en general el procedimiento es el mismo para cualquier F (ecuación (5)). Dado $\boldsymbol{J}$, lo más inmediato es tomar θ como paso de integración, aunque no es nuestro caso. Los resultados mejoran empleando otro método de integración, por ejemplo Euler mejorado o Runge-Kutta, o símplemente reduciendo el paso de integración para un mejor resultado, de manera que obtengamos la respuesta $x(t)$ en los tiempos que coincidan con los nodos virtuales escogidos. En nuestro caso se ha reducido el paso de integración sobremuestreando la señal $\boldsymbol{J}$, ejemplo de sobremuestreo (*oversampling*) $OS = 3$:

$$\boldsymbol{J_{over}} = (J_1, J_1, J_1, J_2, J_2, J_2, J_3, J_3, J_3, \ldots, J_{MN}, J_{MN}, J_{MN}) \tag{8}$$

donde $\boldsymbol{J_{over}}$ es la señal secuencializada sobremuestreada. Si tomamos $h = \theta/OS$ reducimos el paso de integración y obtenemos una respuesta sobremuestreada, Se obtiene $x(t = n\theta)$ símplemente quedándonos con una de cada OS componentes. Si queremos partir de un punto fijo para

operar en su entorno, antes de inyectar la señal $J(t)$ se deja evolucionar la respuesta dinámica hasta el estado estacionario.

Además distinguimos dos estructuras a la hora de realizar la integración según el tipo de tarea:

- En una **tarea de regresión**, predicción de series temporales por ejemplo, es importante conservar la información de salidas en tiempos anteriores al presente en cada punto y esto requiere memoria, con lo cual en cada vuelta τ se parte del estado anterior para conservar parte de la información computada hasta entonces. Llamamos método continuo al hecho de obtener x de este modo y el sistema resultante es equivalente a una ESN (*echo state network*) [2].

- En una **tarea de clasificación** no es relevante el orden en el que se introduce la información de cada punto D-dimensional y después de cada tiempo de inyección volvemos al estado estacionario dejando evolucionar el sistema sin entrada. Llamamos método separado al hecho de obtener x de este modo y el sistema resultante es equivalente a una ELM (*extreme learning machine*) [2].

Otro factor a tener en cuenta en el caso de contemplar la posibilidad de implementar el sistema experimentalmente es comprobar que la simulación funcione con ruido añadido en la señal de entrada. Experimentalmente la señal se resuelve con un número de bits menor, por lo que se pierde precisión respecto a una simulación por ordenador. Por tanto, se ha añadido ruido (aleatorio) para evitar que pequeños detalles en la curva $J(t)$, no apreciables en un experimento, influyan en la salida $\boldsymbol{y}$.

2.2.2. Puntos fijos de la ecuación de Ikeda

Como ya hemos dicho, partiremos del estado estacionario antes de introducir $J(t)$ en la ecuación (7) y obtener la respuesta dinámica. Para calcular los posibles puntos de partida de la respuesta dinámica sin entrada, como para un punto fijo x_0 no hay variación temporal, la derivada es nula y por tanto la respuesta constante: $x(t) = x(t-\tau) = x_0$. La ecuación resultante es:

$$x_0 = \beta \sin^2(x_0 + \phi) \tag{9}$$

donde si se anula el seno tenemos los mínimos $x_0 = 0$, $\phi = \pm k\pi$ y los máximos en $x_0 = \beta$, $\phi = \pi(\pm k + 1/2) - \beta$ con $k = 0, 1, 2, 3, \ldots$ (ver figura 6).

El hecho de inyectar $J(t)$ perturba el estado estacionario y para cada tarea ($J(t)$ diferente) la

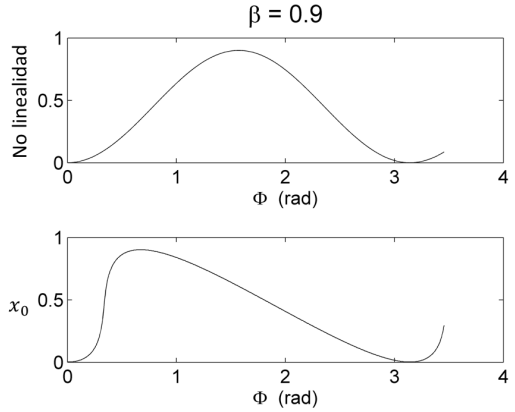


Figura 6: La no linealidad es el segundo término de la ecuación de Ikeda y x_0 son los puntos fijos.

perturbación puede ser más o menos significativa en función de β, γ y ϕ. El hecho de añadir la señal de entrada se puede interpretar como sustituir la fase ϕ por una fase no constante que

oscila alrededor de la que ya teníamos y que viene modulada por el parámetro γ. Cuanto mayor sea γ, mayor rango de fases estamos "explorando" en la curva que describen todos los posibles puntos fijos, pero también nos alejamos cada vez más del punto inicial; y alejarse en exceso no conduce a buenos resultados. Esto se debe a que β modula la amplitud de la no linealidad (seno cuadrado) desplazando la respuesta de la curva de puntos fijos de manera que el comportamiento se vuelve cada vez más caótico al aumentar β. Lo que buscamos es que la desviación del punto fijo debida a estos parámetros sea tal que el proceso se lleve a cabo en su entorno, a una distancia suficiente para considerar diferencias entre diferentes respuestas (separación) y no tan grande como para que diferencias sutiles arrojen resultados muy diferentes. En la figura 7 se ilustran tres ejemplos de respuestas dinámicas en función de la entrada secuencializada.

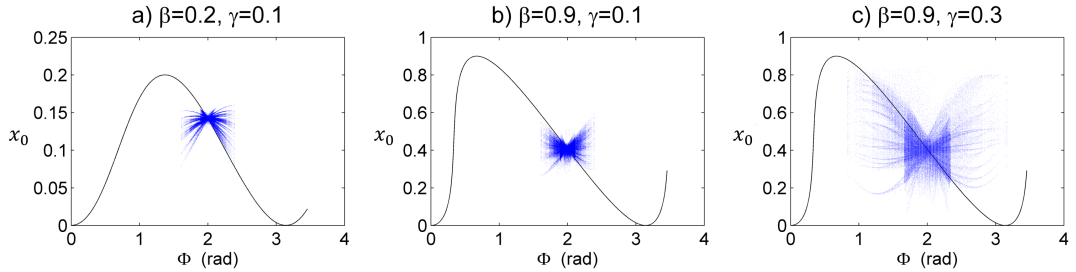


Figura 7: En los tres casos se representan resultados de los puntos perturbados en azul respecto del punto fijo. Se puede observar la influencia de γ en el alcance sobre la curva, de β en la dispersión de los puntos fuera de la curva y de ϕ, que determina la región en la que se opera. Se ha empleado la misma señal de entrada que en el apartado 4 con $\phi = 2$

Por otro lado, en los máximos y mínimos de $x_0(\phi)$, así como en sus proximidades, la evolución respecto al punto fijo queda restringida en comparacióncon el resto, lo cual adelantamos que no conduce a buenos resultados. En la figura 8 se muestra un ejemplo de dicho comportamiento, véase que las respuestas en el máximo y el mínimo representados describen parábolas en las que la densidad de puntos es mucho mayor alrededor de los respectivos puntos fijos, es decir, para la mayoría de señales $J(t)$ se cumple que $x \simeq x_0$, el régimen dinámico no es caótico. En cambio, veremos que suficientemente lejos de los puntos de tangente horizontal de la función $x_0(\phi)$ los resultados son óptimos para una buena elección de los parámetros β y γ.

2.2.3. Matriz de estado

Análogamente a la matriz de entrada $\boldsymbol{I}$, se define la matriz de estado del *reservoir* $\boldsymbol{S}$, que contiene la correspondiente salida secuencializada $x(t)$ y M términos adicionales:

$$\boldsymbol{S} = \begin{pmatrix} x(\theta) & \dots & x(N\theta) & b_1 \\ x((N+1)\theta) & \dots & x(2N\theta) & b_2 \\ x((2N+1)\theta) & \dots & x(3N\theta) & b_3 \\ \vdots & \ddots & \vdots & \vdots \\ x((MN-N)\theta) & \dots & x(MN\theta) & b_M \end{pmatrix} \tag{10}$$

Nos volveremos a referir a esta matriz al comentar cómo se obtienen los pesos (apartado 2.3.1). La columna añadida (*bias*) corresponde a una señal constante adicional y se ha comprobado que el hecho de añadirla hace que la respuesta permanezca óptima para un rango más amplio de fases ϕ de la ecuación de Ikeda, permitiéndonos centrar la función de aprendizaje en un punto diferente del origen.

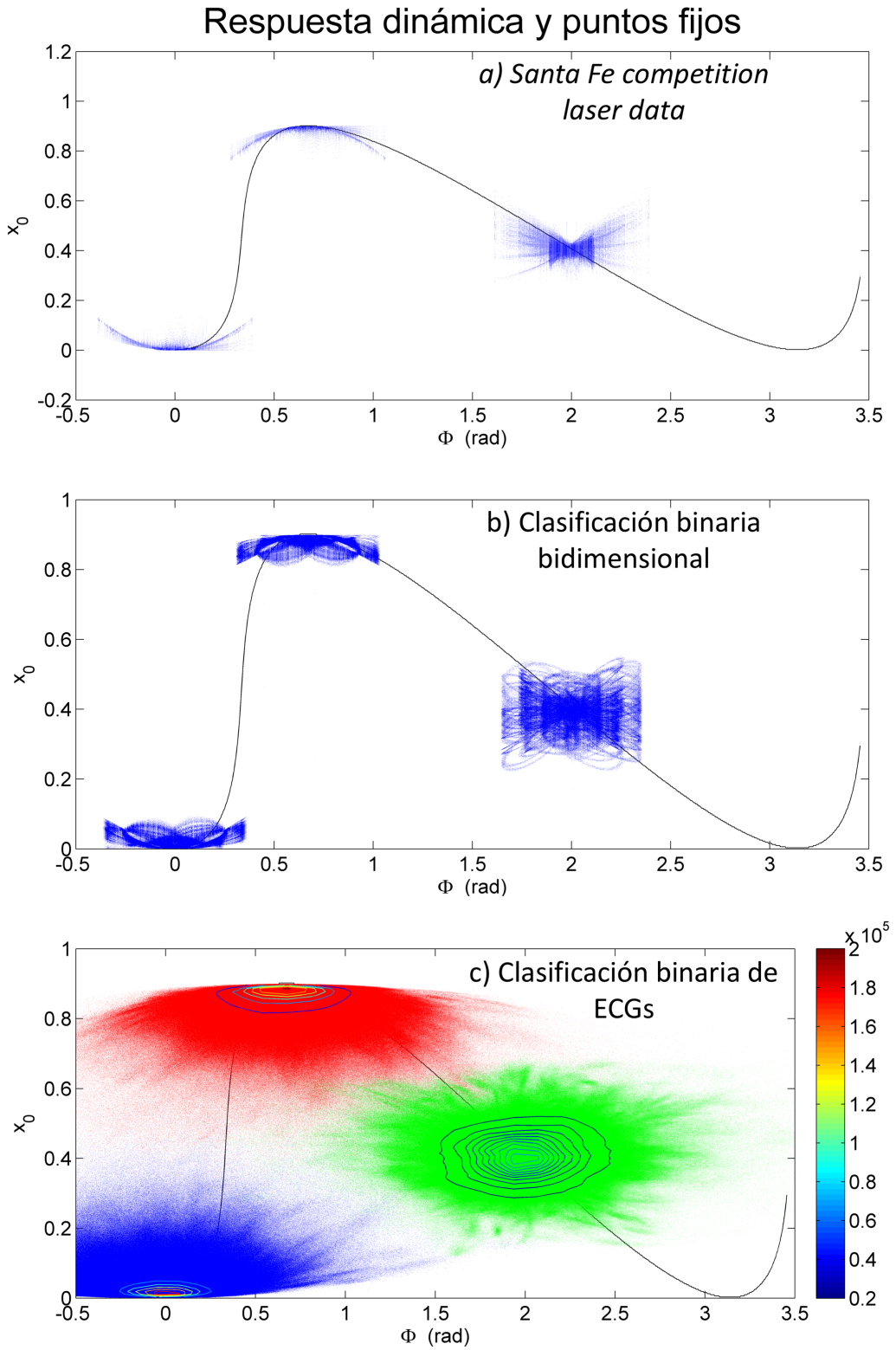


Figura 8: Se han representado tres respuestas superpuestas en los correspondientes puntos fijos para $\beta = 0.9$ y para distintas tareas. a) Apartado 4, $\gamma = 0.1$. b) Apartado 5.1, $\gamma = 0.1$. c) Apartado 6, $\gamma = 0.015$. En cada gráfico dos de las respuestas parten de un mínimo y un máximo de la función $x_0(\phi)$. En (c) se han empleado tres colores para diferenciar mejor las tres respuestas y contornos para indicar las concentraciones de puntos indicadas en la barra de color de la derecha.

2.3. Capa de salida

Con la finalidad de resolver ciertas tareas y comprobar su eficiencia, destinamos un porcentaje de los datos disponibles a la fase de aprendizaje y otra parte distinta de los datos a evaluar los resultados tras el aprendizaje. Es habitual destinar también una parte de los datos a un proceso llamado *cross-validation* [16] para evitar problemas de *overfitting* [17]. En este apartado se explican las fases de aprendizaje y posterior evaluación, es decir, cuán buenos son los resultados obtenidos en una tarea.

2.3.1. Aprendizaje

La fase de aprendizaje termina siendo equivalente a la de un perceptrón simple, compuesto por una sola capa (no hay capa oculta) sin realimentación y en la que cada señal tiene un peso asociado. La respuesta de cada nodo virtual no proporciona directamente la información necesaria para realizar una tarea. Las transformaciones llevadas a cabo a lo largo de la línea de retraso temporal son interpretadas en la capa de salida como una suma ponderada con los pesos de cada transformación. Estos pesos $\boldsymbol{W}$ se pueden interpretar como la importancia relativa de cada contribución y deben ser tales que la salida, y_{rm} en general, sea la óptima, $\hat{y}_{rm}$. Donde $r = 1, \ldots, R$, siendo R el número de salidas. Con salida óptima nos referimos a la más cercana posible a la salida ideal en términos de RC (solamente se optimiza la capa de salida), con lo cual los pesos de la capa de salida serían diferentes para diferentes máscaras aleatorias (conexiones internas tradicionales). Entonces, si la matriz de salida $\boldsymbol{y}$ tiene dimensiones $R \times M$ y la de pesos $R \times (N+1)$:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{S}^T \tag{11}$$

por tanto, teniendo en cuenta (10) y (11), las componentes de $\boldsymbol{y}$ son:

$$\hat{y}_{rm} = \sum_{n=1}^{N} \omega_{rn} \cdot x[m\tau - \frac{\tau}{N}(N-n)] + \omega_{r,N+1} \cdot b_m \tag{12}$$

donde ω_{rn} son las componentes de $\boldsymbol{W}$. Lo más habitual en tareas de aprendizaje supervisado es estimar estos pesos con una regresión lineal múltiple a partir de ejemplos conocidos (*targets*), $\boldsymbol{y}_{tar}$:

$$\boldsymbol{W} = \boldsymbol{y}_{tar}(\boldsymbol{S}^T)^{\dagger} \tag{13}$$

donde † se refiere a la pseudoinversa de Moore-Penrose [18], que generaliza y extiende la inversa convencional. Lo interesante aquí es que como $\boldsymbol{S}^T \in \boldsymbol{\mathcal{M}}(M, N+1; \mathbb{R})$ y $M \geq N+1$, entonces el problema puede ser resuelto con ecuaciones normales:

$$\boldsymbol{W} = \boldsymbol{y}_{tar}(\boldsymbol{S}\boldsymbol{S}^T)^{-1}\boldsymbol{S} \tag{14}$$

la ventaja está en que las dimensiones de $\boldsymbol{S}\boldsymbol{S}^T$, que es lo que se debe invertir, no dependen del volumen de datos destinados al aprendizaje, es una matriz cuadrada $(N+1) \times (N+1)$ y además el método (14) tiene un coste computacional menor que el de aplicar directamente (13). En este trabajo no se ha usado ningún método de regularización y en consecuencia los pesos obtenidos son elevados, sobre todo cuando se emplea la regresión logística que veremos a continuación. Esto es debido a que si el determinante de $\boldsymbol{S}\boldsymbol{S}^T$ es muy cercano a cero su inversa es muy grande.

En un marco más general se emplean modelos lineales generalizados (GLM). Estos modelos constan de tres elementos básicos: una función de distribución perteneciente a la familia exponencial

de distribuciones [19], un predictor lineal y una función de enlace. La familia exponencial de distribuciones incluye, entre otras, la distribución normal y la binomial.

La regresión multilineal es un caso particular típico para respuestas continuas en las que la distribución es normal y su función de enlace es la identidad. Los modelos GLM generalizan esta regresión, también llamada ordinaria, en dos aspectos: permiten distribuciones para $\boldsymbol{y}$ diferentes a la normal e incluyen funciones de enlace diferentes de la identidad, las cuales representan la media de la distribución correspondiente. Por otro lado, permiten la unificación de una amplia variedad de métodos estadísticos y se usa el mismo algoritmo para obtener los estimadores de máxima verosimilitud [20]. En nuestro caso hemos empleado funciones de MATLAB (ver *Statistics and machine learning toolbox*^{TM} [21]).

Otro caso particular en el cual estamos interesados es la regresión logística. En muchos casos las respuestas son binarias (0 y 1), que se suelen interpretar como la probabilidad de éxito (1) o fracaso (0) aleatoria (distribución binomial). El objetivo es conocer las probabilidades de pertenecer a una u otra categoría, que en nuestro modelo son las respuestas $\boldsymbol{y}$. La función de enlace es *logit*, tal que:

$$logit(y_m) = \ln\left(\frac{y_m}{1-y_m}\right) \tag{15}$$

donde el cociente de probabilidades es conocido como *odds ratio* e y_m representa las componentes de $\boldsymbol{y}$ en el caso $R = 1$, ya que es suficiente con conocer la probabilidad de '1'. Para hacer la regresión suponemos dependecias lineales de los $logit(y_m)$ con los pesos:

$$logit(\boldsymbol{y}) = \boldsymbol{W}\boldsymbol{S}^T \tag{16}$$

Una vez determinado $\boldsymbol{W}$ con ejemplos conocidos, obtenemos las probabilidades aproximadas para otro conjunto de datos que nos genere otra matriz de estado similar aislando en la ecuación anterior:

$$\boldsymbol{y} = \frac{\exp(\boldsymbol{W}\boldsymbol{S}^T)}{1+\exp(\boldsymbol{W}\boldsymbol{S}^T)} \tag{17}$$

Al final del trabajo veremos que en tareas de clasificación la regresión logística funciona mejor que la multilineal cuando dos respuestas dinámicas de estímulos que pertenecen a categorías diferentes son muy parecidos. En la figura 8c, debido a la gran cantidad de datos que se emplean en la tarea y la naturaleza de los ECGs, que pertenecen a diferentes pacientes, hay más probabilidades de que dos respuestas que deberían clasificarse en categorías distintas generen respuestas dinámicas muy parecidas. Si tomamos como ejemplo dos componentes de la matriz de estado muy parecidas (δx pequeño):

$$x[(m+i)\tau - \frac{\tau}{N}(N-n)] = x[m\tau - \frac{\tau}{N}(N-n)] + \delta x \tag{18}$$

entonces, para la regresión multilineal:

$$\hat{y}_{r,m+i} = \sum_{n=1}^{N} \omega_{r,m+i} \cdot x[m\tau - \frac{\tau}{N}(N-n)] + \sum_{n=1}^{N} \omega_{rn}\delta x + \omega_{r,N+1} b_{m+i} \tag{19}$$

donde si δx es pequeño, como en muchos casos pequeñas desviaciones respecto de $x[m\tau - \frac{\tau}{N}(N-n)]$ pertenecen a la misma categoría y estamos contemplando el caso en el que no es así. Entonces tras la fase de aprendizaje el *bias* y los pesos serán muy similares en ambos casos

y en consecuencia $\hat{y}_{rm} \simeq \hat{y}_{r,m+i}$, en general no separable. Para solucionar ésto introducimos la regresión logística, que representa una transformación no lineal adicional. A partir de la ecuación (17), es equivalente que:

$$\boldsymbol{y} \simeq \begin{cases} \exp(\boldsymbol{W}\boldsymbol{S}^T) \simeq \mathbf{0} & \text{si } \boldsymbol{W}\boldsymbol{S}^T < -\boldsymbol{\epsilon} \\ \mathbf{1} & \text{si } \boldsymbol{W}\boldsymbol{S}^T > \boldsymbol{\epsilon} \end{cases} \tag{20}$$

donde las componentes de $\boldsymbol{\epsilon}$ pueden ser tan pequeñas como sea necesario y la región intermedia entre $-\boldsymbol{\epsilon}$ y $\boldsymbol{\epsilon}$ es muy abrupta, de modo que aunque dos respuestas dinámicas sean similares hay más probabilidades de que podamos clasificarlas correctamente con la regresión logística que con la multilineal.

2.3.2. Evaluación

En la fase de evaluación se introducen definiciones de cuantificadores para caracterizar las propiedades del modelo, en su mayoría porcentajes que nos indican cuán buena es la aproximación. Algunas de las propiedades son independientes de la tarea que se realice y pueden ser evaluadas por separado, un ejemplo es la capacidad de memoria, relevante cuando la información de los canales de entrada tiene una estructura dependiente del tiempo. En las siguientes secciones se muestran tareas de memoria, predicción de una serie temporal y clasificaciones bidimensionales y de electrocardiogramas (ECG). Al final de cada sección se lleva a cabo la fase de evaluación, donde se muestra el modo de cuantificar el error y otros resultados según el tipo de tarea que se trate.

3. Tarea de memoria

La tarea de memoria consiste en determinar la cantidad de información que conserva el *reservoir* en relación a datos de entrada correspondientes a pasos de tiempo anteriores. Para ello los datos utilizados son números reales (entre 0 y 1) aleatorios porque por definición no deben estar correlacionados, 4000 en total.

3.1. Procedimiento y evaluación

La entrada de datos es unidimensional ($D = 1$), por tanto $\boldsymbol{u}_t$ tiene dimensiones $M \times 1$ y componentes u_m, con $m = 1, \ldots, M$. De nuevo, tomamos y_m como las componentes de $\boldsymbol{y}$ en el caso $R = 1$. En cuanto a la fase de aprendizaje, se ha empleado la regresión multilineal porque los números aleatorios reales son variables continuas. Si se empleara la regresión logística la señal debería de ser normalizada y discretizada, obteniendo así un peor resultado al perder información inicial en la discretización. Además, el uso de clasificadores binarios para tareas de más de dos categorías requiere como mínimo $K-1$ clasificadores (ver apartado 5.2), siendo K el número de categorías, de modo que a no ser que se quieran obtener predicciones de variables discretas o cualitativas, no tiene sentido comparar los resultados de la regresión multilineal con los de la logística.

Sea $mem(i)$ la función de memoria dada por la correlación lineal normalizada entre las predicciones y los valores correctos de los números aleatorios [22], [2]:

$$mem(i) = corr[y_m, u_{m-i}] = \frac{\langle u_{m-i} y_m \rangle_m^2}{\sigma^2[u_m]\sigma^2[y_m]} \tag{21}$$

donde $\langle \cdot \rangle_m$ es la media sobre todos los valores de m. Se define la capacidad de memoria MC como la suma de valores de la función de memoria:

$$MC = \sum_{i=1}^{\infty} mem(i). \tag{22}$$

Como en la práctica la función $mem(i)$ tiende a cero conforme aumenta i, basta con sumar sobre cierta cantidad de puntos.

El método de integración es continuo (ver apartado 2.2.1) y 3000 de los 4000 puntos han sido destinados a la fase de aprendizaje. Para proceder a calcular estos resultados, con los que evaluamos la tarea, lo primero es encontrar un régimen de operación aproximadamente óptimo, es decir, los parámetros β, γ y ϕ de la ecuación (7) que maximizan MC. Lo que hacemos es fijar el número de nodos virtuales ($N = 50$) y determinar los parámetros haciendo barridos para diferentes valores de cada uno.

3.2. Resultados

Un ejemplo de barrido es el de las figuras 9 y 10. Estos resultados expuestos a modo de ejemplo corresponden a una sola ejecución del programa para cada conjunto de parámetros, es decir, no son resultados promedio y por tanto hay un factor aleatorio a tener en cuenta debido a la máscara. Sin embargo se espera que las posibles variaciones no sean muy significativas. Se muestra también la función $mem(i)$ promediada sobre cinco ejecuciones para parámetros que esperamos que arrojen buenos resultados y diferente número de nodos virtuales (ver pie de la figura 11 y comparar con las figuras 9 y 10).

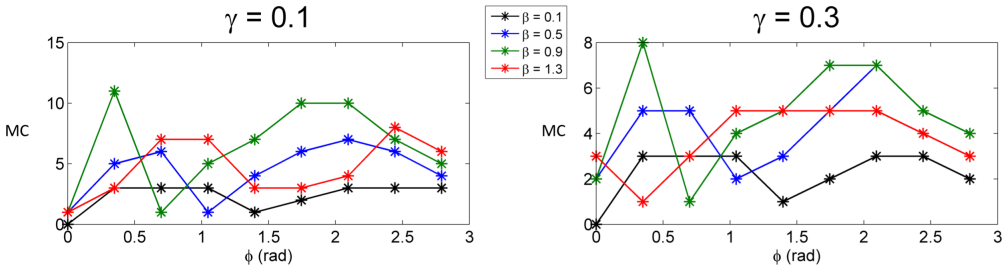


Figura 9: Barrido de β, γ y ϕ para el cálculo de MC con $N = 50$, $\theta = 0.2$ y $OS = 2$.

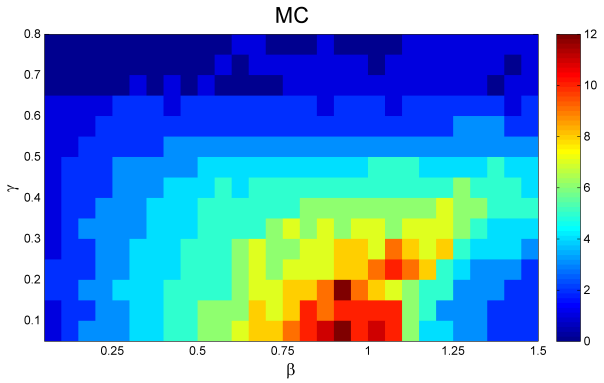


Figura 10: Barrido de β y γ para el cálculo de MC con $N = 50$, $\theta = 0.2$ y $OS = 2$. Es una representación en escala decimal y el paso de malla del gráfico es 0.05.

La razón por la que se ha escogido $\phi = 2$ en lugar de $\phi = \pi/9$ es porque las fases alrededor de $\pi/9$ dan lugar a peores resultados al estar muy cerca de $\phi = 0$ y $\phi = \pi/2 - \beta$, en cambio, para $\phi \simeq 1.6 \pm 0.5$ los resultados parecen ser casi igual de buenos, algo importante en el caso de no poder fijar la fase con mucha precisión. Por otro lado, aunque el conjunto de parámetros

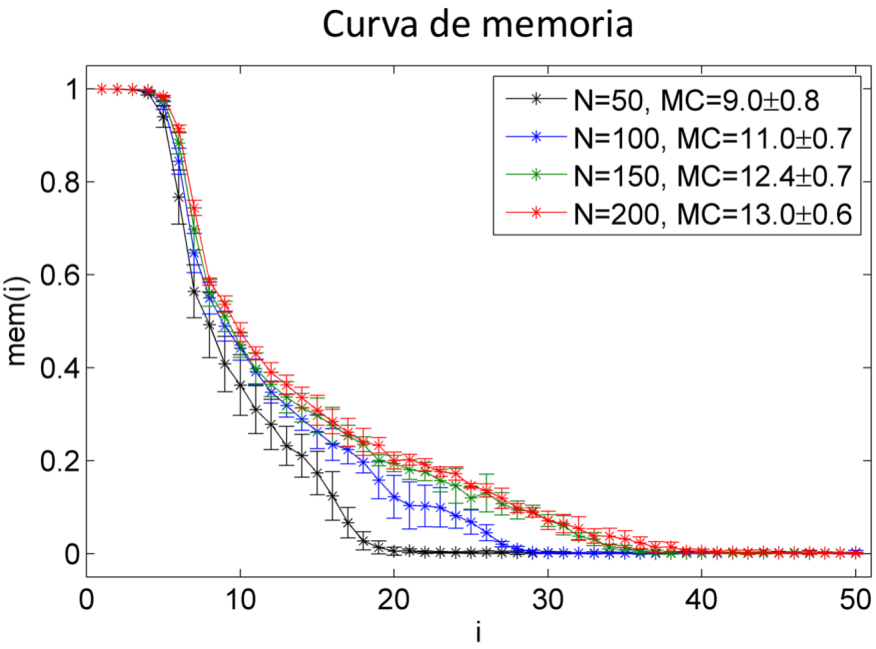


Figura 11: Promedio sobre 5 ejecuciones de la función $mem(i)$ para diferentes números de nodos virtuales con $\beta = 0.9$, $\gamma = 0.1$, $\phi = 2$, $\theta = 0.2$ y $OS = 2$.

β, γ y ϕ ha sido aproximado para $N = 50$, como al incrementarlo aumentamos el número de pesos, se espera un mejor resultado. Se puede observar como MC tiende a un valor asintótico con $N \to \infty$.

4. Tarea de predicción de una serie temporal: *Santa Fe competition laser data*

El objetivo de esta tarea es hacer predicciones para tiempos posteriores al presente, en nuestro caso predicciones a un paso de tiempo. En particular se ha escogido una serie temporal de una tarea de referencia (*benchmark task*) bastante utilizada: *Santa Fe competition laser data* [23]. Los datos empleados representan medidas de la intensidad de un laser de NH_3 (81.5 μm) en función del tiempo operando en un régimen caótico [24].

4.1. Procedimiento y evaluación

Los datos empleados son 4000 puntos divididos en 4 series de 1000 puntos, de las cuales 3 han sido destinadas a la fase de aprendizaje. Igual que en la tarea de memoria tenemos canales de entrada y salida unidimensionales ($D = R = 1$), las respuestas y_m representan predicciones de un punto posterior a partir de resultados anteriores gracias a la memoria introducida integrando la ecuación (7) con el método continuo (ver apartado 3). Además las variables sobre las cuales hacemos predicciones son continuas, $u_m \in \mathbb{R}$, por lo que la fase de aprendizaje se lleva a cabo con la regresión multilineal.

La evaluación se ha llevado a cabo calculando el *normalized mean square error* ($NMSE$):

$$NMSE = \frac{1}{M} \frac{\sum_{m=1}^{M} (y_m^{tar} - y_m)^2}{\sigma^2[y_m^{tar}]} \tag{23}$$

donde y_m^{tar} son las componentes de $\boldsymbol{y}_{tar}$, es decir, los ejemplos usados para la fase de aprendizaje. En este caso, $y_m^{tar} = u_{m+1}$.

Los parámetros de la ecuación (7) que conduzcan a los mejores resultados serán aquellos que minimicen el $NMSE$, por tanto haremos un barrido análogo al del apartado anterior, con el fin de obtener resultados promedio para un buen conjunto de parámetros.

4.2. Resultados

En las figuras 12 y 13 se muestran barridos de algunos valores de los parámetros mencionados, en el que se puede ver que $\beta = 0.9$ y $\gamma = 0.3$ conducen a buenos resultados en comparación al resto de valores. Además los resultados se mantienen en el mismo rango de fases que para la tarea de memoria. En la figura 14 se pueden visualizar los datos originales y sus predicciones, el error es mayor cuanto mayor es la derivada de y_m. Finalmente, en la figura 15 se representa el $NMSE$ para los parámetros mencionados y para $\phi = 2$, aumentando el número de nodos virtuales y promediando sobre 10 ejecuciones.

Se debe mencionar que tanto esta tarea como la de memoria pueden ser llevadas a cabo haciendo la integración con el método separado a cambio de aumentar la dimensión de los datos de entrada del orden de 10 veces para obtener resultados similares [2].

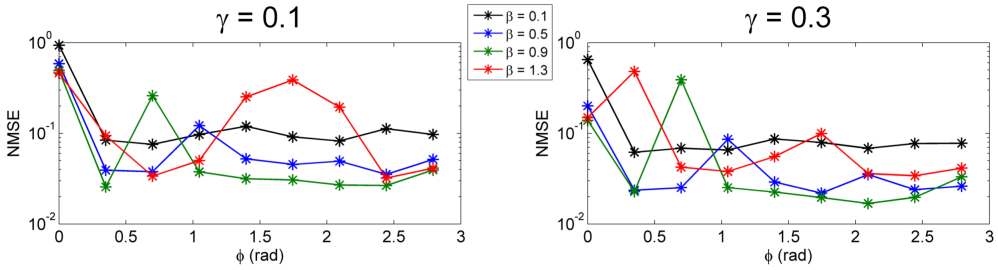


Figura 12: Barrido de β, γ y ϕ para el cálculo del $NMSE$ con $N = 50$, $\theta = 0.2$ y $OS = 2$.

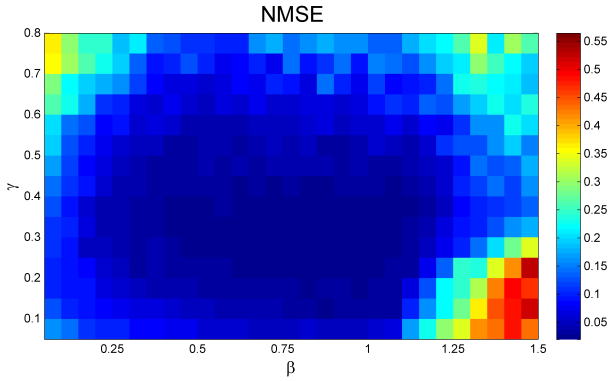


Figura 13: Barrido de β y γ para el cálculo del $NMSE$ con $N = 50$, $\theta = 0.2$ y $OS = 2$. Es una representación del logaritmo del $NMSE$ y el paso de malla del gráfico es 0.05.

5. Tareas de clasificación bidimensional

A modo de ejemplo y para facilitar la visualización de los resultados e introducir parámetros de evaluación del rendimiento típicos en estas tareas, que usaremos en la clasificación de ECGs, se han considerado grupos de puntos en el plano XY, donde cada coordenada es análoga a una característica en una aplicación real. Estas características suelen ser escogidas de modo que se mantengan alrededor de un punto en el espacio D-dimensional, es decir, características relevantes que no tengan en común las categorías y suficientes para que sean separables por la

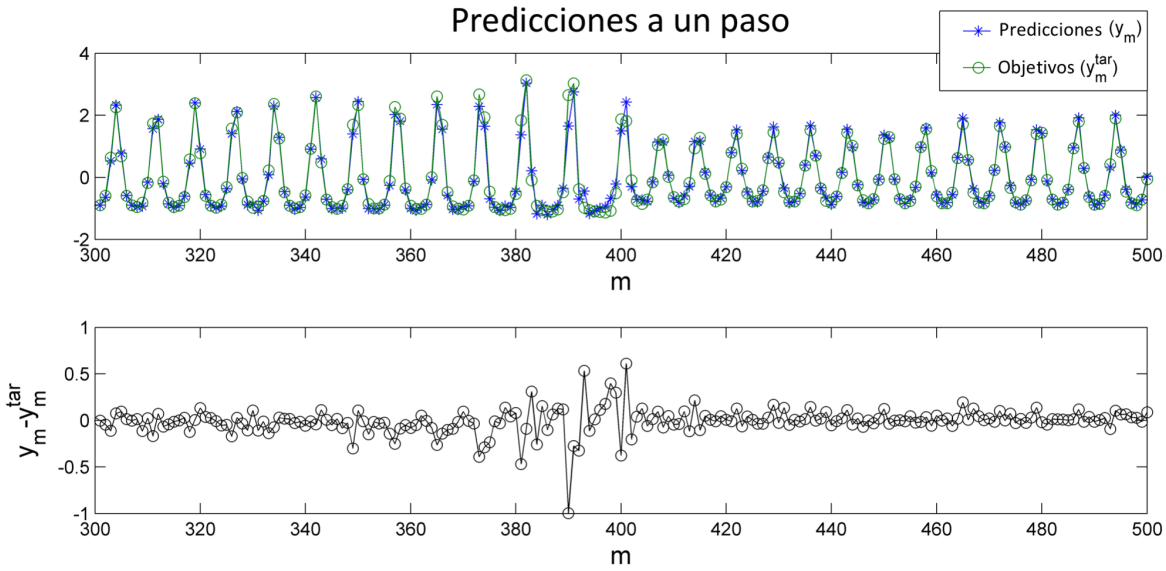


Figura 14: Visualización de la tarea de predicción y el error cometido con $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 2$.

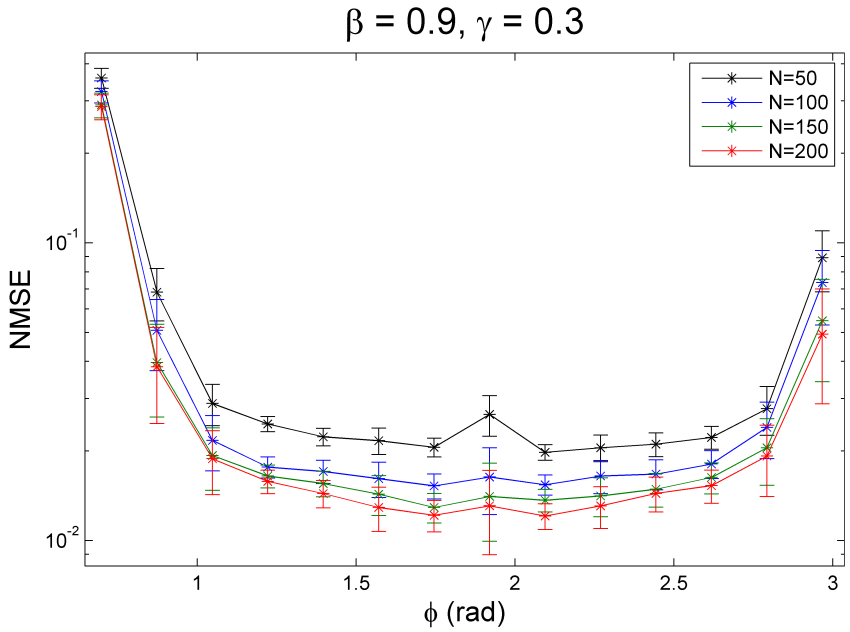


Figura 15: $NMSE$ promediado sobre 10 ejecuciones para diferentes fases y diferentes números de nodos virtuales con $\theta = 0.2$, $OS = 2$, $\beta = 0.9$ y $\gamma = 0.3$.

máquina en dicho espacio.

El método de integración es separado y las variables de entrada y salida son discretas, por lo que contemplaremos dos algoritmos de aprendizaje: la regresión multilineal y la logística.

5.1. Clasificación binaria

En una clasificación binaria la división del espacio viene dada por un solo conjunto de $N + 1$ pesos, puesto que identificar una categoría es suficiente para separar las dos. Cada categoría tiene asignada una etiqueta, pongamos que son '1' y '0', pero las respuestas y_{rm} en principio pueden ser cualquier número real en el caso de la regresión multilineal y entre 0 y 1 para la logística, representan probabilidades, por tanto si se desea decidir si un punto del espacio D-dimensional pertenece a una u otra categoría se hace con el criterio *winner-take-all*: si una respuesta es mayor que 0.5 se toma 1 y si es menor se toma 0. Esto cambiaría en el caso de no disponer del mismo número de ejemplos de cada categoría, cosa que normalmente se debe a la frecuencia con la que ocurre '1' o '0' y en tal caso la frontera entre ambas clases no es necesariamente 0.5. Para ilustrar este hecho podemos poner como ejemplo la tarea de clasificación ECGs (ver apartado 6), en la cual disponemos de más datos de individuos sin irregularidades cardíacas debido a que es más frecuente. Esta falta de simetría en los datos provoca que el hiperplano que separa las categorías se incline hacia el grupo mayoritario, mientras que lo idóneo sería tenerlo en el punto medio. Además, el ratio de acierto no es un buen valor de referencia porque si el 90 % de las muestras destinadas al aprendizaje pertenecen a '0' y se obtiene un 91 % de acierto empleando los mismos datos para el test, significa que la generalización del sistema consiste en atribuir la clase '0' a casi todas las muestras (nuevas o no), o lo que es lo mismo, que el ratio de acierto de la clase '1' es solamente del 10 %. Para solucionar este problema se disminuye el umbral que separa las categorías, ya no es 0.5, sino que es menor. Ésto es habitual si se emplea la regresión logística. Si en cambio se emplea la multilineal, ésta funciona mejor renombrar las clases, ya que las respuestas no representan probabilidades y sus extremos pueden ser ajustados con números reales positivos y negativos, que suelen ser -1 y 1 si el número de muestras es equitativo, posteriormente haríamos corresponder -1 con la clase '0' y 1 con la clase '1'. En ambos casos la idea es que la zona de intersección o confusión de '0' y '1' quede equidistribuida, es decir, que el porcentaje de acierto o error de ambas categorías sea similar, hecho que puede generalizarse a la clasificación múltiple.

5.1.1. Procedimiento y evaluación

La tarea es bidimensional: $D = R = 2$. Como señales de entrada se han tomado dos circunferencias concéntricas con una cierta incertidumbre en el valor de sus respectivos radios, compuestas por 2000 puntos cada una, de los cuales el 75 % han sido destinados a la fase de aprendizaje.

En cuanto a la evaluación, se mostrará la matriz de confusión (tabla 1) teniendo en cuenta los falsos negativos (FN), falsos positivos (FP), verdaderos negativos (VN) y verdaderos positivos (VP). Donde $positivo \equiv 1$ y $negativo \equiv 0$.

P \ O	0	1
0	VN	FN
1	FP	VP

Tabla 1: Matriz de confusión binaria, donde las predicciones (P) se obtienen con el criterio *winner-take-all* y los ejemplos suministrados para el test son los objetivos (O).

5.1.2. Resultados

En la figura 16 se muestra la topología de las dos señales de entrada etiquetadas correctamente junto con las predicciones correspondientes. Se muestra también la división del espacio (generalización) correspondiente a cada regresió en la figura 17. Aunque veremos que para esta tarea el resultado es independiente de si se emplea una regresión u otra sí que son diferentes las separaciones, las fronteras de la logística son mucho más abruptas.

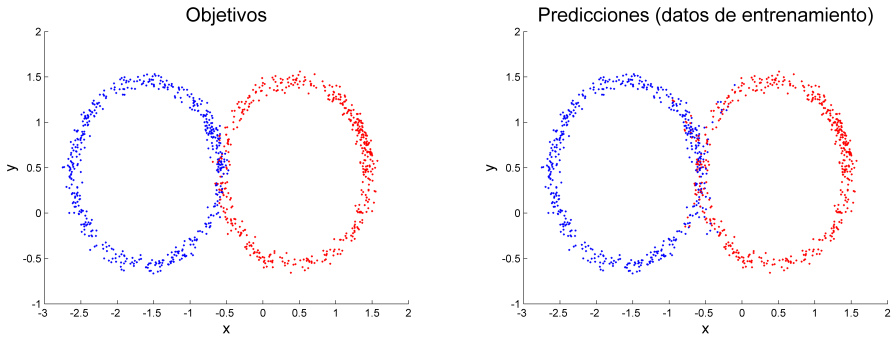


Figura 16: Se han representado los datos de entrada empleados para la fase de aprendizaje junto con las predicciones sobre los mismos datos de entrenamiento. Se han empleado los siguientes parámetros en ambos casos: $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 1.7$.

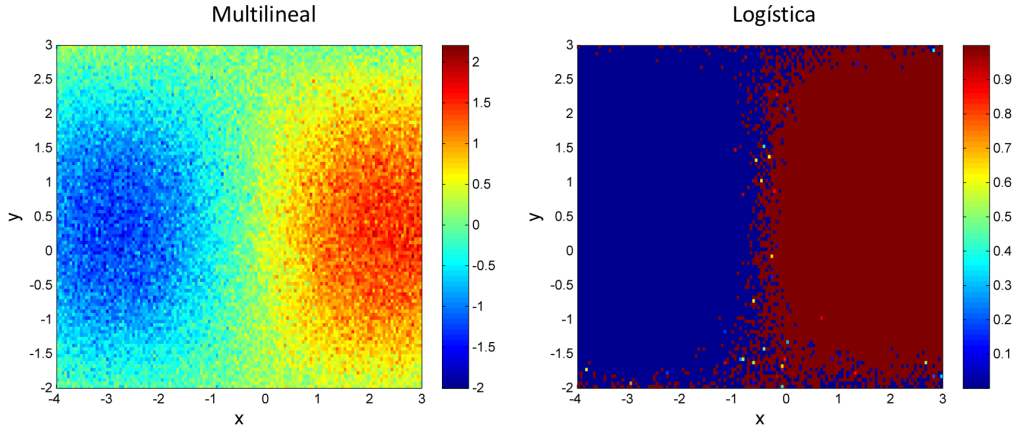


Figura 17: En los gráficos se muestra la separación del espacio según la regresión que ajuste los pesos. Se han empleado los siguientes parámetros en ambos casos: $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 1.7$. Paso de malla del gráfico: 0.05.

Durante la evaluación se obtiene la matriz de confusión de la tabla 2. Solamente se muestra la obtenida con la regresión logística porque que las variaciones respecto a la multilineal son despreciables en promedio y dichas variaciones se deben al uso de máscaras diferentes en cada ejecución.

P \ O	0	1
0	447	53
1	52	448

Tabla 2: Matriz de confusión de la tarea correspondiente al apartado 5.1. Se han empleado los siguientes parámetros: $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 1.7$, aunque los resultados son equivalentes alrededor de esta fase, suficientemente alejada de los máximos y mínimos de $x_0(\phi)$.

5.2. Clasificación múltiple

Una clasificación múltiple es aquella en la que intervienen más de dos categorías. El problema de la clasificación múltiple puede descomponerse en varias clasificaciones binarias, las cuales se pueden llevar a cabo del mismo modo que en el apartado anterior, aunque en este caso la división del espacio está condicionada por la superposición de todas las iteraciones de separación binaria. Mostraremos los resultados correspondientes a dos criterios de separación: *one-versus-all* (OVA) y clasificación jerárquica. También es común referirse a *all-versus-all* (AVA), en la tabla 3 se ilustra de manera esquemática en qué consiste cada uno.

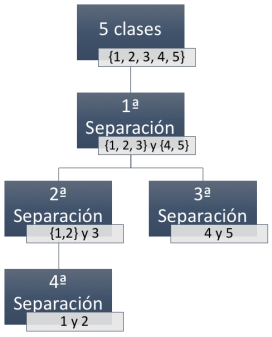
a) OVA	b) AVA	c) Jerárquico
Cada categoría se separa del resto del grupo. $1, 2, 3, \dots, K \Longrightarrow P_1$ $2, 1, 3, \dots, K \Longrightarrow P_2$ $3, 1, 2, \dots, K \Longrightarrow P_3$ $[\dots]$ $(K-1), 1, 2, 3, \dots, K \Longrightarrow P_{K-1}$ $P_K = 1 - \sum_{i=1}^{K-1} P_i$ El número de clasificadores necesarios es $K-1$.	Cada categoría se compara con el resto por parejas. $1, 2 \Longrightarrow P_{12}$ $1, 3 \Longrightarrow P_{13}$ $[\dots]$ $1, K \Longrightarrow P_{1K}$ $2, 3 \Longrightarrow P_{23}$ $2, 4 \Longrightarrow P_{24}$ $[\dots]$ $2, K \Longrightarrow P_{2K}$ $[\dots]$ $(K-1), K \Longrightarrow P_{(K-1)K}$ Donde $P_{ij} = 1 - P_{ji}$, se obtiene: $P_i = \prod_{j \neq i} P_{ij}\,;\;\; P_K = 1 - \sum_{i=1}^{K-1} P_i$ El número de clasificadores necesarios es $K(K-1)/2$.	Es un método de árbol de decisión, se procede por separación de subconjuntos. Ejemplo con 5 categorías:  En la primera separación se obtiene $P'_{123} = 1 - P'_{45}$, en la segunda $P'_{12} = 1 - P'_3$, en la tercera $P'_4 = 1 - P'_5$ y en la cuarta $P'_1 = 1 - P'_2$. Por tanto: $P_1 = P'_{123}P'_{12}P'_1$ $P_2 = P'_{123}P'_{12}P'_2$ $P_3 = P'_{123}P'_3$ $P_4 = P'_{45}P'_4$ $P_5 = P'_{45}P'_5$ El número de clasificadores necesarios es $K-1$.

Tabla 3: a) *One-versus-all*. b) *All-versus-all*. c) *Método jerárquico*. Las P representan distribuciones de probabilidad y K es el número de categorías.

El uso de métodos jerárquicos es habitual en el análisis de cluster con aprendizaje no supervisado, donde se trata de buscar patrones comunes en grupos de puntos para subdividirlos. Por grupo de puntos se entiende que los datos oscilan alrededor de un centro geométrico y el análisis trata de identificar estructuras internas.

El método OVA puede dar resultados tan buenos como el resto según la tarea que se trate. Aquí se mostrará un caso en el que no es posible obtener buenos resultados empleando dicho método y que puede llevarse a cabo con el jerárquico. Algo que también se debe tener en cuenta es el caso en el que las respuestas no proporcionen directamente distribuciones de probabilidad, como ocurre al emplear la regresión multilineal, en tal caso las normalizamos entre 0 y 1 para así obtener algo equivalente a las distribuciones de probabilidad P que aparecen en la tabla 3, aunque ésto no asegura buenos resultados. Sin embargo, en caso de aplicar el criterio OVA para resolver la tarea uno no debe preocuparse de si sus respuestas representan probabilidades o no, puesto que no aparece ningún producto de las mismas y basta con asignar a cada categoría un número diferente en función de las muestras disponibles de cada una.

5.2.1. Procedimiento y evaluación

La tarea escogida consiste en separar 6 circunferencias concéntricas con radios definidos con una cierta incertidumbre y de tamaños distintos, tal y como se muestra en la figura 18. En total se dispone de 2000 puntos de cada categoría, que se permutan de manera aleatoria para

posteriormente emplear el 75 % en la fase de aprendizaje y el resto en la evaluación, de modo que el número de muestras empleadas evaluadas no tiene porque ser el mismo para las seis categorías. Se ha procedido determinando los pesos a partir de la regresión logística y se comparan los resultados de los criterios OVA y jerárquico. En el siguiente apartado se muestra la matriz de confusión correspondiente al segundo, pues con el primero veremos que no es posible separar las categorías.

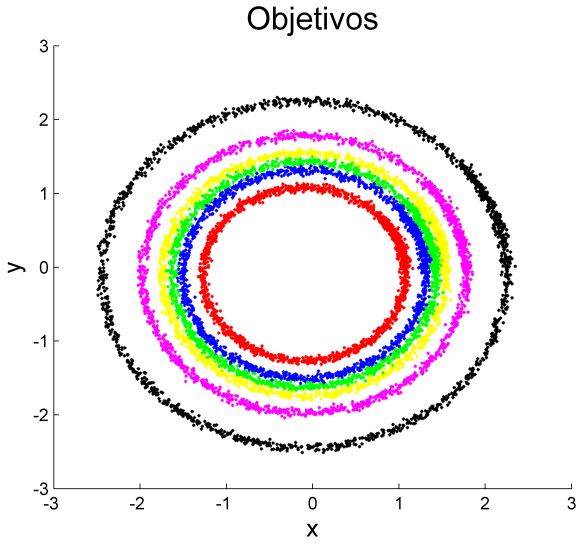


Figura 18: Datos destinados al aprendizaje de la tarea del apartado 5.2 etiquetados correctamente.

5.2.2. Resultados

Cuando tratamos este tipo de problemas normalmente uno no puede saber a priori que criterio de separación debe usar, ya que podemos encontrarnos espacios de muy alta dimensión. En dos dimensiones sí que podemos predecir que el método OVA no va a funcionar. Etiquetemos por ejemplo P_i en orden de radio creciente, al separar '1' del resto obtenemos P_1 y el resultado seguramente sea bueno, pero cuando separemos '2' del resto, la distribución P_2 incluirá '1' y '2', la distribución P_3 incluirá '1', '2' y '3', y así sucesivamente. De modo que al tratar de ajustar los resultados a los datos etiquetados obtendremos lo que se muestra en la figura 19 (OVA). El método jerárquico funciona porque en cada iteración se renombran las categorías, recuperando su etiqueta original al final del proceso. El hecho de que tengamos en cuenta productos de probabilidad cancela las contribuciones de las distribuciones finales (P_i) a categorías que no les corresponde. Se expone el resultado en la figura 19 (jerárquico). Al representar las distribuciones no se ha tenido en cuenta P_6 para mayor claridad, a la que le corresponde el resto del espacio:

$$P_6 = 1 - \sum_{i=1}^{5} P_i \tag{24}$$

Finalmente, en la tabla 4 se muestra la matriz de confusión correspondiente al método jerárquico, que es la generalización de la binaria (tabla 2) a seis categorías. Los resultados pueden variar ligeramente pero siempre se observa que el número de fallos en la clasificación es mayor cuanto más cerca están un grupo de otro.

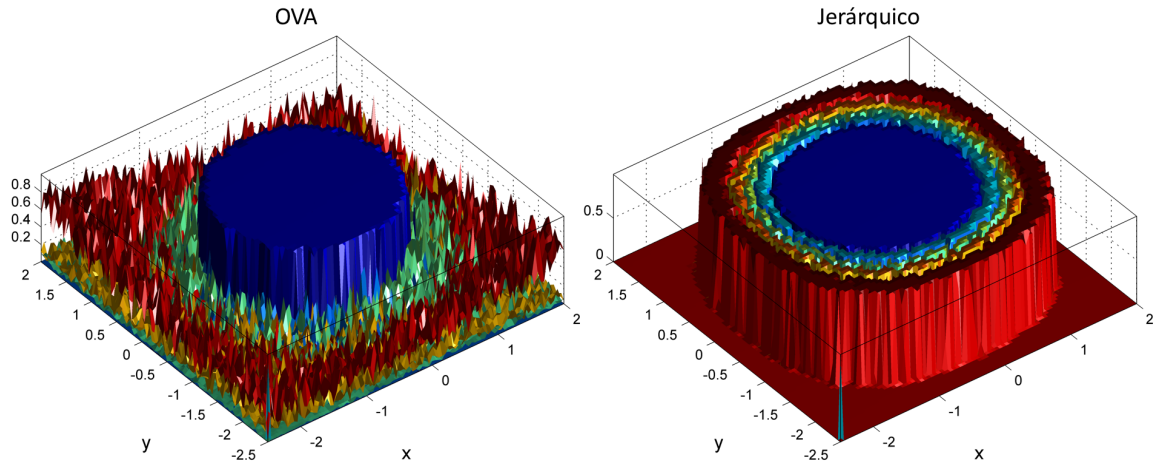


Figura 19: Distribuciones de probabilidad de cada categoría superpuestas, usando los criterios OVA y jerárquico, respectivamente. Dichas distribuciones han sido generadas llevando a cabo la fase de aprendizaje con la regresión logística. Se han empleado los siguientes parámetros: $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 1.7$. Paso de malla del gráfico: 0.05.

P \ O	1	2	3	4	5	6
1	517	3	0	0	0	0
2	1	455	58	0	0	0
3	0	59	382	56	0	0
4	0	0	53	442	6	0
5	0	0	0	12	454	0
6	0	0	0	0	0	501

Tabla 4: Matriz de confusión de la tarea correspondiente al apartado 5.2. Se han empleado los siguientes parámetros: $N = 50$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.3$ y $\phi = 1.7$, aunque los resultados son equivalentes alrededor de esta fase, suficientemente alejada de los máximos y mínimos de $x_0(\phi)$.

6. Clasificación de electrocardiogramas

La clasificación de ECGs es un ejemplo de aplicación real con la que se pone a prueba el método de clasificación empleado en este trabajo. A efectos prácticos, puesto que no hay tratamiento previo de la señal original sino que directamente tomamos las entradas como $\boldsymbol{u}_t$, todo lo que uno necesita saber es que la mayoría de trastornos del ritmo cardíaco junto con los latidos normales (incluidos en los latidos tipo N) están divididos en cinco grupos, tal y como se muestra en la figura 5 y que son los que contiene la base de datos utilizada. Cada tipo de latido está subdividido en otros que no se muestran aquí. La tarea que se ha llevado a cabo en este trabajo ha consistido en separar latidos tipo N y S por una parte y V y F por otra.

Tipos de latidos (AAMI)				
N	**S**	**V**	**F**	**Q**
Latidos que no son de tipo S, V, F o Q	Latidos supraventriculares	Latidos ventriculares	Latidos de unión	Latidos desconocidos

Tabla 5: Se muestran las cinco clases AAMI de latidos, incluidas en la base de datos MIT-BIH Arrhythmia.

Se ha utilizado la base de datos MIT-BIH Arrhythmia, bien conocida en cardiología y disponible en línea [25], para el entrenamiento y evaluación del clasificador. Ésta consta de 48 registros de 47 pacientes de diferentes edades y sexos. La base de datos consta de dos derivaciones, con una duración aproximada de 30 minutos muestreados a 360 Hz. Teniendo en cuenta que el corazón de una persona adulta late entre 70 y 80 veces por minuto, cada ECG contiene unos 2250 latidos y en la base de datos cada uno está caracterizado por 170 muestras ($D = 170$). Estas muestras fueron tomadas alrededor del punto R (máximo), con 70 muestras antes de dicho punto y 100 después con el fin de incluir las ondas P y T de cada latido. Para mayor claridad en la figura 20 se muestra la señal típica de un latido y en la figura 21 la distribución de los datos en entrenamiento y test.

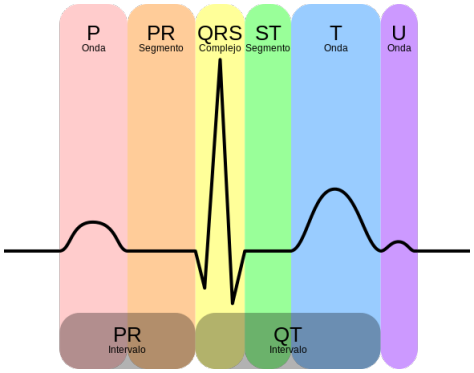


Figura 20: Señal típica de un latido, voltaje en función del tiempo. Imagen bajo licencia *CC BY-SA 3.0* via *Wikipedia*.

Figura 21: Se muestra la cantidad de latidos utilizados en las fases de aprendizaje y evaluación. DS1 y DS2 son las dos derivaciones con las etiquetas modificadas respecto a la base de datos original.

6.1. Motivación

A continuación se exponen algunos datos relevantes sobre el tema para motivar la tarea de clasificación:

- Según la Sociedad Española de Cardiología (SEC) [26], las enfermedades cardiovasculares son la primera causa de defunción en España, por encima del cáncer y de las enfermedades respiratorias.

- La técnica de obtención de los latidos codificados que contienen los ECGs no requiere instrumental caro y es no invasiva.

- El ECG es el método de evaluación de la actividad cardíaca más usado a nivel mundial y es la primera prueba que se le realiza al paciente en este ámbito.

- En la mayoría de los casos, tratar las arritmias a tiempo puede prevenir fallos y paros cardíacos.

6.2. Procedimiento y evaluación

Primero se dividen los latidos correspondientes a los diferentes pacientes de acuerdo con los estándares AAMI. Después, con el propósito de separar los latidos tipo N/S de los tipo V/F, la base de datos ha sido modificada. Originalmente las dos derivaciones (DS1 y DS2) contenían las clases de latidos que se mencionan en la tabla 5, primero se han eliminado los latidos tipo Q y posteriormente se ha asignado la categoría '1' a los de tipo V/F y '0' a los de tipo N/S.

Se sabe que añadir el tiempo entre latidos como característica de entrada adicional mejora los resultados, aunque aquí no se ha tenido en cuenta. Se mostrarán las matrices de confusión correspondientes a la regresión multilineal y a la logística, para posteriormente comparar el mejor resultado con algún otro método publicado recientemente, algo que en ocasiones no es muy riguroso porque diferentes autores siguen diferentes criterios de evaluación y se debe buscar alguno que use la misma base de datos y se adapte razonablemente a nuestro modo de exponer los resultados.

6.3. Resultados

En la tabla 6 se muestran las matrices de confusión correspondientes a las regresiones mencionadas. La regresión logística resulta funcionar mejor, lo que es una ventaja porque la respuesta proporciona directamente las probabilidades de que se trate de un tipo de latido u otro.

P \ O	N/S		V/F	
N/S	43807	29916	2286	16177
V/F	770	499	2838	3109

Tabla 6: Matriz de confusión de la tarea correspondiente al apartado 6. Se han empleado los siguientes parámetros: $N = 150$, $\theta = 0.2$, $OS = 2$, $\beta = 0.9$, $\gamma = 0.015$ y $\phi = 1.7$. Se muestran en azul y naranja los resultados correspondientes a las regresiones logística y multilineal. El umbral del probabilidad escogido para la regresión logística es 0.2. El reetiquetado de las clases en la regresión multilineal se corresponde con N/S $\rightarrow -\frac{46809+4202}{46809}$ y V/F $\rightarrow \frac{46809+4202}{4202}$.

Los resultados son comparables con el estado del arte, ver [27] por ejemplo, aunque la comparación no es directa.

7. Conclusión

Los resultados expuestos a lo largo del trabajo muestran que el *reservoir computer* que se discute en este trabajo es capaz de realizar tanto tareas de regresión como de clasificación (apartados

4, 5 y 6). Además, hemos visto que los dos métodos de integración sugeridos en el apartado 2.2.1 funcionan como un interruptor que simplemente se activa o se desactiva en función de la tarea que se quiera llevar a cabo, como ya se ha mencionado anteriormente. En tareas en las que no influye el orden de inyección de los estímulos hemos empleado una ELM y en las que sí es relevante dicho orden hemos empleado una ESN, aunque como ya se ha comentado, en el caso de los ECGs los resultados podrían mejorar con una ESN que tenga en cuenta efectos de memoria para los latidos de cada paciente en el contexto de sus latidos anteriores. Lo interesante aquí es que el programa se puede ver como un marco de trabajo unificado para ambas arquitecturas de red [2]. Como contribución propia al programa original, destacar la incorporación de la regresión logística mediante modelos GLM y los métodos de clasificación múltiple en general.

Vale destacar que los resultados se podrían mejorar escogiendo la máscara con algún método determinista, como la propagación hacia atrás o el descenso del gradiente. Además dicha máscara puede construirse con más de dos bits si la tarea lo requiere. Otra posible mejora consiste en añadir capas ocultas, es decir, tener más de un nodo físico sujeto a retraso temporal, equivalente a una *deep neural network* [28].

Referencias

[1] M. C. Soriano, S. Ortin, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, and G. Van der Sande, "Delay-based reservoir computing: noise effects in a combined analog and digital implementation," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 388–393, 2015.

[2] S. Ortín, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martín, I. Fischer, C. Mirasso, and J. Gutiérrez, "A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron," *Scientific reports*, vol. 5, 2015.

[3] M. L. Alomar, M. C. Soriano, M. Escalona-Morán, V. Canals, I. Fischer, C. R. Mirasso, and J. L. Rosselló, "Digital implementation of a single dynamical node reservoir computer," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 62, no. 10, pp. 977–981, 2015.

[4] A. Teichman and S. Thrun, "Practical object recognition in autonomous driving and beyond," in *Advanced Robotics and its Social Impacts (ARSO), 2011 IEEE Workshop on*, pp. 35–38, IEEE, 2011.

[5] J. Zhao, Z. Xie, L. Jiang, H. Cai, H. Liu, and G. Hirzinger, "Levenberg-marquardt based neural network control for a five-fingered prosthetic hand," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 4482–4487, IEEE, 2005.

[6] M. A. Escalona-Moran, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Electrocardiogram classification using reservoir computing with logistic regression," *Biomedical and Health Informatics, IEEE Journal of*, vol. 19, no. 3, pp. 892–898, 2015.

[7] J. Daniel and J. H. Martin, "Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics," *Englewood Cliffs, N: Prentice-Hall*, 2009.

[8] L. Appeltant, "Reservoir computing based on delay-dynamical systems," *These de Doctorat, Vrije Universiteit Brussel/Universitat de les Illes Balears*, 2012.

[9] I. Wolfram Research, "Highly automated machine learning," *Champaign, Illinois*, 2015.

[10] URL: http://gallery.cortanaintelligence.com/Experiment/a635502fc98b402a890efe21cec65b92. Accedido 30/05/2016.

[11] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural Networks*, vol. 20, no. 3, pp. 323–334, 2007.

[12] J. L. Rossello, V. Canals, A. Oliver, and A. Morro, "Studying the role of synchronized and chaotic spiking neural ensembles in neural information processing," *International journal of neural systems*, vol. 24, no. 05, p. 1430003, 2014.

[13] S. J. Xilinx, "Ca, usa," *Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators, Application Note*, 1996.

[14] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, vol. 2, p. 468, 2011.

[15] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[16] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, 1995.

[17] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[18] J. C. A. Barata and M. S. Hussein, "The moore–penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, 2012.

[19] M. Collins, S. Dasgupta, and R. E. Schapire, "A generalization of principal components analysis to the exponential family," in *Advances in neural information processing systems*, pp. 617–624, 2001.

[20] M. Aitkin, "A general maximum likelihood analysis of variance components in generalized linear models," *Biometrics*, vol. 55, no. 1, pp. 117–128, 1999.

[21] URL: http://es.mathworks.com/help/stats/index.html. Accedido 03/04/2016.

[22] H. Jaeger, *Short term memory in echo state networks*. GMD-Forschungszentrum Informationstechnik, 2001.

[23] A. S. Weigend, *Time series prediction: forescasting the future and understanding the past*. No. 04; QA280, T5., 1994.

[24] U. Hübner, N. Abraham, and C. Weiss, "Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared nh 3 laser," *Physical Review A*, vol. 40, no. 11, p. 6354, 1989.

[25] URL: http://www.physionet.org/physiobank/database/mitdb/. Accedido 01/05/2016.

[26] URL: http://secardiologia.es. Accedido 01/05/2016.

[27] C. L. Herry, M. Frasch, and H.-t. Wu, "Characterization of ecg patterns with the synchrosqueezing transform," *arXiv preprint arXiv:1510.02541*, 2015.

[28] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.