Universidad Nacional de Cuyo

Facultad de Ciencias Exactas y Naturales

Aprendizaje autónomo de redes neuronales artificiales

por

Agustín M. Bilen

Director: Dr. Pablo F. Kaluza

Informe final del Seminario de Investigación,
presentado como parte de los requerimientos para la obtención del grado de
Licenciado en Ciencias Básicas con Orientación en Física.
Marzo de 2016

Resumen

Estudiamos numéricamente redes neuronales artificiales multicapa con procesamiento unidireccional (*feed-forward*) y su optimización con respecto a dos propiedades claves y de especial importancia en el campo de las redes complejas y los modelos computacionales de sistemas biológicos: el cumplimiento de una función compleja con generalización de lo aprendido y la robustez estructural. En primer lugar, nos interesa optimizar las redes para cumplir cierta función: el reconocimiento de las vocales en una matriz de píxeles. Con ello, esperamos no sólo que cada red sepa clasificar los patrones aprendidos, sino que pueda generalizar a casos novedosos lo que se le enseñó en casos particulares, clasificando correctamente las vocales aún cuando las señales que se le muestren presenten ruido o sean defectuosas. En segundo lugar, buscamos que las redes creadas sean estructuralmente robustas, esto es, conserven su buena operatividad luego de sufrir daños en su topología. Usualmente, para la optimización de redes neuronales artificiales, los algoritmos de aprendizaje que se emplean dependen de un agente externo implícito en su formulación que durante el proceso guía a la red en la modificación de sus parámetros hasta que ésta alcanza un desempeño satisfactorio u óptimo. La línea central de este trabajo es la implementación de un método estocástico de aprendizaje por refuerzo, denominado *aprendizaje autónomo*, según el cual el propio estado de la red define la magnitud y la dirección de los cambios para que ésta logre optimizarse. Encontramos que, para la función planteada y el diseño de red elegido, si bien la convergencia hacia el desempeño óptimo de las redes no es tan rápida como en el método analítico clásico de retropropagación (*backpropagation*), las redes optimizadas con el aprendizaje autónomo resultan exitosamente funcionales y estructuralmente robustas. Demostramos, como punto fuerte de nuestro trabajo, que el aprendizaje autónomo permite de manera natural una implementación multiparámetrica según la cual las redes pueden evolucionar autónomamente optimizando más de un parámetro a la vez. En nuestro caso, los parámetros son *dos*: funcionalidad y robustez estructural.

Agradecimientos

En primer lugar, agradezco a mi madre Cecilia E. E. Bricco y a mi padre J. Miguel Bilen, por su ejemplo de vida, por todo su cariño y su trabajo. Toda gran posibilidad que me ha traído hasta aquí, no es más que el fruto de su infatigable dedicación y esfuerzo. Y por supuesto a mis hermanas, dos hermosas cómplices a quienes también pertenece cualquier mérito que yo pueda cosechar. Espero que vean en éste y en todo futuro trabajo, la fe emocional y material que depositaron en mí.

A mis amigos, en orden alfabético: Alejandro M., Andrés H., Anthony P., Daniela N., Franco P., Franco S., Luis "Capulina", María "Pope" N., Paula M., Rodrigo B., Rodrigo Z., Santiago F., Xorge V. y Xosé V. Por todo su acompañamiento, su paciencia, y aquella gran amistad que ha hecho de mí alguien infinitamente afortunado.

A mi director Pablo F. Kaluza, por su gran calidad y solidaridad intelectual, por su constantemente buena predisposición a instruir y guiarme, y por su enorme paciencia. No sólo ha sido un placer trabajar bajo su dirección, sino que su ejemplo a lo largo de estos años ha ayudado en gran medida a definir la vara de excelencia a la cual espero honrar en mi futura carrera como investigador.

A mi facultad. En términos generales, a todo el personal docente, administrativo, y de mantenimiento, porque su arduo trabajo hoy me permite llamarle hogar a mi primera casa de estudios. En términos precisos, a Eduardo Bringa y Enrique Miranda, por poner incesantemente todos sus consejos, recursos y voluntad a disposición de sus estudiantes, gracias a lo cual, en particular, fue posible que me recibiera en tiempo y forma para comenzar este año mis estudios de posgrado. A este respecto, agradezco fuertemente también a Luis Marone. A Andrés Aceña, Enrique Miranda, y Sebastián Simondi, por valiosas charlas 'extraoficiales' de importancia fundamental para la conservación de mi entusiasmo y motivación como estudiante de física. A Cecilia Fernández Gauna, por la gran oportunidad de crecimiento que significó para mí poder realizar mi trabajo final de Física Experimental II como pasante en el Observatorio Pierre Auger; agradezco de la misma forma a Ricardo Sato por todo lo aprendido durante dicha pasantía. Agradezco finalmente a Roberto Isoardi y Enrique Miranda, por aceptar ser parte del jurado de este trabajo.

A Recreo y en especial a Lilia M. Dubini. No alcanzan las palabras para agradecer todo el crecimiento que fue posible durante estos años gracias a su ejemplo de pasión, convicción y trabajo, así como a la inmensa fe depositada, y renovada años tras año, en el equipo del cual formé parte.

Finalmente, a mi país. Espero en épocas futuras estar a la altura de lo que ha invertido en mí.

Índice general

Dedicado a mi madre, a mi padre, y a mis dos hermanas.

"Biological neural networks are just one of many possible solutions to the problem of processing information."

-Raúl Rojas, *Neural Networks: A Systematic Introduction*

Capítulo 1

Introducción

1.1. Motivación

Las redes neuronales artificiales tienen su origen a mediados del siglo XX [1] y surgen rápidamente como una alternativa de modelo computacional frente a los existentes en esa época (p. ej. máquina de Turing, red de automatas celulares) [2] tomando como fuente de inspiración a los sistemas nerviosos de los seres vivos y su gran potencial de cómputo. Estos últimos, se caracterizan marcadamente por su sofisticada capacidad de procesamiento en paralelo de la información, por la variada especialización funcional de sus componentes, y por un alto nivel de jerarquización, comunicación y coordinación entre sus partes. Estas características les permiten a dichos sistemas, entre otras cosas, el reconocimiento de patrones complejos, la capacidad de aprender y la capacidad de generalizar lo aprendido para responder apropiadamente ante estímulos novedosos. Al mismo tiempo, los sistemas biológicos que motivan los diseños computacionales poseen una gran resistencia a las fallas que puedan sufrir las partes que los constituyen. Esta robustez estructural les permite seguir cumpliendo sus funciones aún luego de que alguno de sus componentes se encuentre dañado.

Los modelos de redes neuronales artificiales, también llamados *coneccionistas* o de *procesamiento paralelo distributivo*, apuntan a alcanzar un alto rendimiento de cómputo para cierta tarea mediante la densa interconexión de sus elementos o unidades de procesamiento [3]. El diseño de redes neuronales artificiales se puede dividir en dos etapas generales: el diseño o *configuración estructural* de la red y la *optimización de sus parámetros* para hacerla funcional. Considerando las propiedades de los modelos biológicos mencionadas anteriormente, como estructura genérica y común a todas las redes se adopta una configuración compuesta por unidades que procesan la información y conexiones dirigidas que transmiten la información procesada entre los nodos de la red (Figura 1.1-*superior*). Desde una

perspectiva pragmática, el funcionamiento de una red genérica se puede entender como el accionar de una caja negra que recibe una señal de entrada y produce una de salida (Fig. 1.1-*inferior*). Ahora, el objetivo es que la red produzca la respuesta correcta o apropiada ante cierta señal entrante, con lo cual la pregunta es: dada una función para la cual se desea construir una red ¿cómo se la diseña para que cumpla dicha función exitosamente? O bien, una vez elegida la estructura de la red ¿cómo se le asignan los valores a los parámetros de la red para que ésta sea funcional?

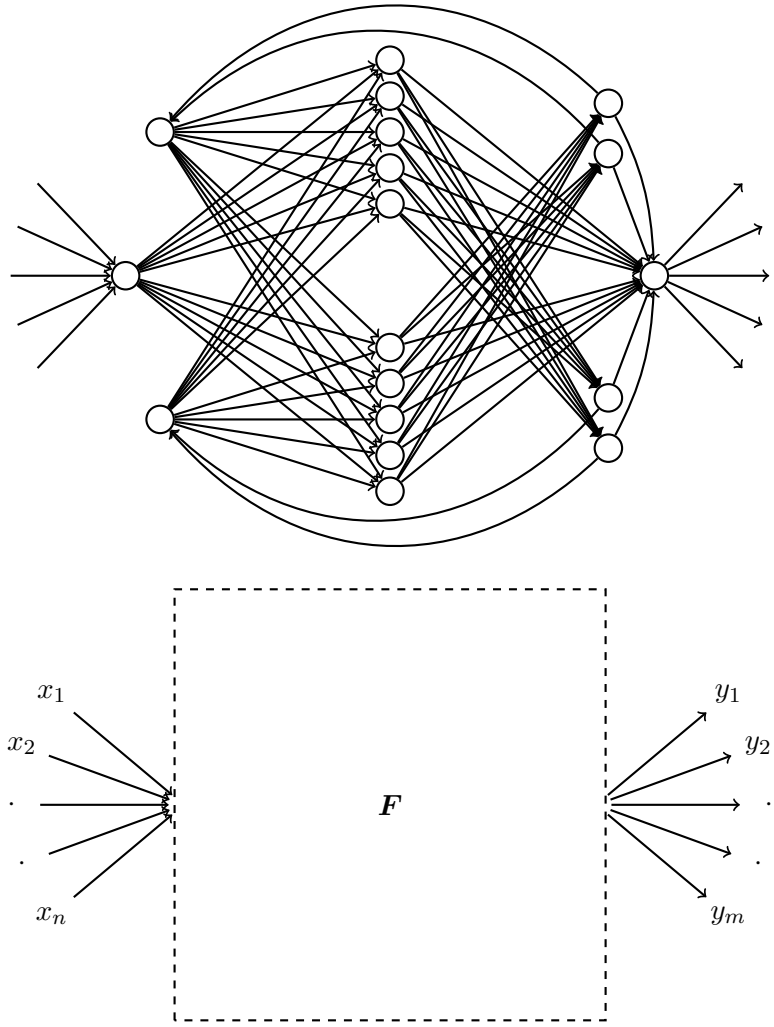

Figura 1.1. (*Arriba*) Ejemplo de estructura ilustrando los elementos básicos de las redes neuronales artificiales: nodos de procesamiento y conexiones dirigidas que los intercomunican. (*Abajo*) Concepción operativa de una red neuronal artificial: el sistema recibe una señal de entrada $(x_1, x_2, \ldots, x_n)$, la procesa, y produce una señal de salida $(y_1, y_2, \ldots, y_m) = \boldsymbol{F}(x_1, x_2, \ldots, x_n)$.

La respuesta está en la segunda etapa en el diseño de las redes: su optimización por medio del llamado *algoritmo de aprendizaje* utilizado para entrenar a la red. Un algoritmo de aprendizaje es esencialmente una metodología de búsqueda de los parámetros estructurales óptimos a la cual se somete la red. Si el aprendizaje es exitoso, se habrán encontrado aquellos valores de los parámetros de

la red con los cuales los errores cometidos por la misma al cumplir la función asignada serán mínimos, es decir, la red se habrá acercado satisfactoriamente a su *función objetivo*. Es importante destacar que, si bien el algoritmo de aprendizaje prescribe una *forma* de acercamiento a los valores óptimos de la red para cierto problema, no es una regla explícita o programable para encontrarlos sino un método de búsqueda a partir de ejemplos representativos [4], según el cual el sujeto del aprendizaje muta bajo cierta regla con el fin de encontrar dichos parámetros al interactuar con su entorno. Esto tiene como consecuencia el hecho de que no siempre resulta claro qué clase de algoritmo utilizar para cierto problema, o bien cuál de todos los que se puedan usar será el que mejor entrene a la red. Entre los numerosos métodos de aprendizaje que hay, algunos de los más conocidos son el aprendizaje del Perceptrón [5], el aprendizaje por el método *simulated annealing* [6, 7, 8], y el aprendizaje por retropropagación (*backpropagation* [9]).

En este trabajo, nos enfocamos en una estructura particular de red llamada *feed-forward* o multicapa con procesamiento unidireccional, y nos proponemos implementar un algoritmo de *aprendizaje autónomo* [10] para entrenarla. Dicho algoritmo se destaca entre entre otros métodos utilizados de aprendizaje en que la prescripción para la optimización de los parámetros no depende de la supervisión constante de un agente externo a la red, como es el caso del método *simulated annealing* o el de retropropagación, sino que se da internamente como parte de su dinámica (*self-supervised* [11]) de manera que los parámetros (en nuestro caso, los pesos de las conexiones) se modifican de acuerdo al estado en el que se encuentra la red con respecto a su objetivo, o bien, a su desviación del comportamiento ideal. De esta forma, la red evoluciona de manera autónoma hacia la configuración estructural óptima para la tarea asignada.

1.2. Objetivos

Como primer objetivo, buscamos aplicar el aprendizaje autónomo a una red multicapa para resolver el problema planteado: el reconocimiento de las vocales en una matriz de píxeles. Además de que la red optimizada pueda reconocer los patrones mediante los cuales se la entrenó para el reconocimiento, esperamos que la red clasifique correctamente patrones que presenten ruido, es decir, que sean versiones defectuosas de los patrones originales. Esto nos permitirá concluir que las redes construidas son capaces no sólo de aprender sino de *generalizar* lo aprendido a instancias novedosas.

Como segundo objetivo, nos interesa que las mismas sean robustas ante la eliminación de uno de sus nodos en la capa de procesamiento, es decir, sigan pudiendo reconocer las vocales satisfactoriamente aún estando dañadas. Estudiamos no sólo la posibilidad de implementar el mismo algoritmo

de aprendizaje para lograr esto, sino que buscamos la optimización *simultánea* de ambas propiedades (funcionalidad y robustez) extendiendo la formulación del aprendizaje autónomo para que éste nos permita su aplicación *multiparamétrica*.

En pocas palabras, investigamos la posibilidad de lograr que las redes *evolucionen autónomamente para ser funcionales y robustas*.

1.3. Organización del trabajo

El trabajo se organiza de la siguiente manera:

En el Capítulo 2, se presenta el modelo general de las redes neuronales artificiales. En primer lugar, en la Sección 2.1 se discuten aquellos elementos básicos de los sistemas neuronales biológicos que motivan el diseño de los modelos artificiales. Posteriormente, en la Sección 2.2 se expone en detalle la estructura general de las redes neuronales artificiales y sus propiedades, y se da a conocer la configuración particular con la que trabajaremos en nuestros estudios: las redes *feed-forward*.

Como mencionamos anteriormente, el aprendizaje de las redes para su optimización hacia cierta función es parte esencial de su diseño. En el Capítulo 3, damos a conocer qué se entiende por aprendizaje en redes neuronales artificiales. En la Sección 3.1 planteamos en términos generales el problema del aprendizaje, exponiendo la terminología adecuada para describirlo y mencionando algunos de los escenarios más comunes asociados a su tratamiento. Luego de detallar en la Sección 3.2 el método clásico de aprendizaje por retropropagación, presentamos en la Sección 3.3 aquel que constituye el objeto de nuestro estudio: *el aprendizaje autónomo*. Concluimos el capítulo aplicando ambos métodos a un caso artificial a modo de ejemplo.

Nuestro estudio abarca los Capítulos 4 y 5. En el primero de ellos, aplicamos el aprendizaje autónomo a 100 redes *feed-forward* con el fin de optimizarlas para que puedan cumplir con la función asignada, y evaluamos el rendimiento de las redes en el cumplimiento de dicha función, para determinar qué tan efectivo es el aprendizaje autónomo en la optimización de las redes con respecto a su *funcionalidad*. En el Capítulo 5 reformulamos el aprendizaje autónomo para poder optimizar las redes con respecto a dos parámetros de forma paralela, la funcionalidad y la robustez estructural, y evaluamos los resultados comparando estas redes con las redes construidas en el primer estudio, introduciendo daños en ambos casos y analizando las distribuciones de las redes dañadas. Finalmente, en el Capítulo 6, presentamos las conclusiones generales de nuestro trabajo.

Capítulo 2

Redes Neuronales Artificiales

En este capítulo, presentamos el modelo general de las redes neuronales artificiales. Estas intentan imitar las capacidadedes computacionales que poseen las redes neuronales biológicas [2], por lo cual para entender mejor los modelos artificiales y su diseño consideramos en primer lugar cuáles son las propiedades fundamentales de los sistemas nerviosos de los animales desde el punto de vista de las tareas de procesamiento de la información que estos llevan a cabo. Una vez extraídos de los modelos biológicos aquellos aspectos esenciales a nuestro fin, procedemos a definir el modelo de las redes neuronales artificiales, describiendo su estructura, sus propiedades, y el modo de operar que las convierte en valiosas herramientas computacionales. Para concluir, en la última sección describimos la arquitectura particular de red que usaremos en nuestros estudios.

2.1. Sistemas neuronales biológicos

La función del sistema nervioso de un ser vivo es procesar la información proveniente de su ambiente (estímulos tanto internos como externos) y emitir acorde a ello las señales necesarias para responder adecuadamente [12]. Una de las principales claves de estos sistemas para realizar tareas extremadamente sofisticadas de proceso y control radica en su particular arquitectura de red [3], en la cual distintos patrones de interconexión comunican a las neuronas entre sí. Esta interconexión entre las neuronas, junto con un alto grado de especialización y jerarquización, según lo cual distintas subredes del sistema se encargan de distintos tipos de procesamiento, no sólo facilita la paralelización y la comunicación entre partes remotas para el procesamiento de señales complejas y las respuestas adecuadas, sino que permite que los tiempos característicos asociados a estas tareas lleguen a ser extremadamente veloces (p. ej. del orden de los 100 ms para el reconocimiento de patrones visuales complejos en el cerebro humano [13]).

2.1.1. La neurona

Las unidades fundamentales de procesamiento del sistema nervioso son las células neuronales o *neuronas*. En la Figura 2.1 se ilustra la fisiología típica de estas células[1]. Las *dendritas* son las vías de transmisión de la información *entrante* proveniente de otras células, mientras que el axón se encarga de portar las señales de salida, es decir, las señales procesadas por la neurona. La información que

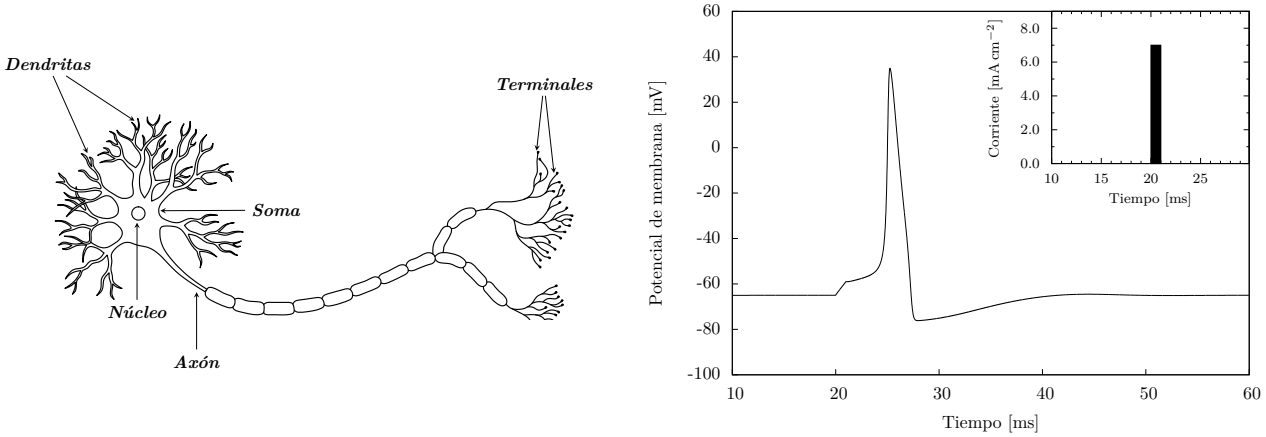

Figura 2.1. (*Izquierda*) Fisilogía básica de una neurona. (*Derecha*) Potencial de acción producido en una neurona debido a una corriente externa de $7\,\mathrm{mA\,cm^{-2}}$ inyectada a los 20 ms durante 1 ms (calculado numéricamente con el modelo de Hodgkin-Huxley [14]).

procesa una neurona está codificada en señales eléctricas. En el estado de inactivación o reposo de la neurona, su membrana celular mantiene una diferencia de potencial eléctrico, llamado *potencial de reposo*, entre el interior (plasma intracelular) y el exterior (fluido intersticial) de la célula debido a la permeabilidad de dicha membrana a ciertas especies iónicas y al accionar de un transporte activo de iones a través de ella. Los iones involucrados son, principalmente, Na^+, K^+, y Cl^-, y el valor aproximado del potencial de reposo es de $-70\,\mathrm{mV}$. Cuando la neurona recibe a través de las dendritas señales eléctricas que en conjunto logran reducir la diferencia de potencial a través de la membrana celular por encima de cierto potencial umbral, se genera un aumento agudo en el potencial de la neurona que se propaga a lo largo del axón. A la generación de este *potencial de acción* (Fig. 2.1-*derecha*), le sigue un *periodo refractario* (aproximadamente de 10 ms) durante el cual la membrana de la neurona está hiperpolarizada, lo que aumenta la diferencia entre el potencial umbral y el potencial de membrana, e imposibilita la producción de otro disparo. Además de limitar la frecuencia a la cual la neurona puede disparar, el periodo refractario tiene como consecuencia fundamental definir una única dirección de propagación del potencial de acción, o bien, de sus señales [15].

[1]Otros seres vivos como los insectos tienen una morfología distinta pero el funcionamiento básico es similar al que describimos aquí [2].

2.1.2. Comunicación y almacenamiento: Sinapsis

La comunicación entre neuronas ocurre en regiones llamadas *sinapsis* [16]. Cuando el potencial de acción proveniente de una neurona (*neurona presináptica*, pintada de *blanco* en la Figura 2.2) llega al lugar de la sinapsis propagándose por el axón, la terminales de éste liberan moléculas llamadas *neurotransmisores*, que viajan hacia la membrana de la neurona receptora (*neurona postsináptica*, gris en la figura), sobre sus dendritas o directamente al cuerpo celular. El efecto producido es un cambio en la permeabilidad de la membrana de la neurona postsináptica, lo que favorece una de dos consecuencias: la *despolarización* o la *hiperpolarización* de la misma, por medio de corrientes iónicas que se propagan por difusión por el cuerpo principal de la neurona. La despolarización de la membrana favorece la generación de potenciales de acción, mientras que la hiperpolarización la inhibe. En el primer caso, se dice que la sinapsis es excitatoria, mientras que en el segundo, inhibitoria. A su vez, el tipo de sinapsis está determinado por la naturaleza del neurotransmisor asociado a la neurona presináptica, que en general es único para cada una.

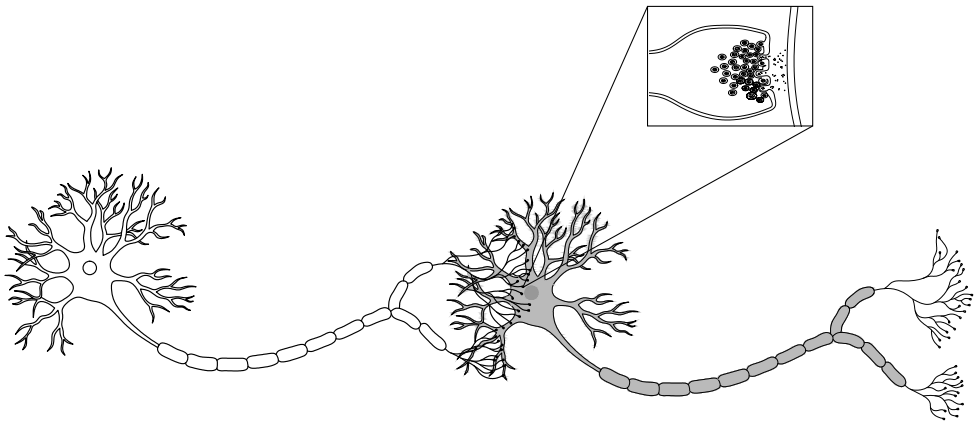

Figura 2.2. Conexión entre dos neuronas. El recuadro magnifica la región de una sinapsis, donde una de las terminales del axón de la neurona presináptica (blanca) libera sus neurotransmisores y estos derivan hasta la membrana de la neurona postsináptica (gris).

Si bien la existencia de un potencial umbral en cada neurona implica que su activación es del tipo todo-o-nada, las sinapsis asociadas a una misma neurona pueden tener distintos niveles de eficiencia para iniciar un potencial de acción, lo que dependerá de la cantidad de canales iónicos que se estimulen con la llegada de una señal [2]. Dicho de otra forma, la neurona postsináptica *pesa* de distinta forma las señales entrantes según las reciba por una u otra sinapsis. Actualmente, se cuenta con una numerosa cantidad de resultados [17, 18, 19] que revelan que uno de los principales métodos de almacenamiento de la información en el sistema nervioso se basa precisamente en esta propiedad de las sinapsis, y que durante los distintos procesos de aprendizaje lo que cambia, o bien el factor de *plasticidad* que permite dicho aprendizaje, es la eficiencia de cada sinapsis para estimular a cierta neurona.

2.2. Redes neuronales artificiales

Habiendo discutido los elementos básicos que conforman los sistemas neuronales biológicos, establecemos en esta sección la transición hacia los modelos de redes neuronales artificiales. Si bien es cierto que estos últimos no pretenden establecer una rigurosa correspondencia con sus contrapartes biológicas en toda su complejidad, se extraen de ellos ciertas caracerísticas para su diseño. Estas son [20]:

- Unidades o nodos que procesan la información (neuronas).
- Conexiones que comunican a los nodos entre sí (sinapsis), y que dan al sistema una estructura general de red. Dos tipos de conexiones: inhibitorias y excitatorias.
- Distintas fuerzas o eficiencias de las conexiones para comunicar las señales entre distintos pares de nodos. Durante los procesos de aprendizaje, estas fuerzas pueden variar (*plasticidad sináptica* [21]), y de esa forma almacenan efectivamente la información necesaria para aprender.
- En cada nodo: integración de todas las señales que llegan a ella (dentro de cierta ventana de tiempo) resultando en una señal entrante *neta*.
- Estado de activación asociado a cada nodo, dependiente de su umbral de activación, de la señal entrante neta y de su estado previo, y que transmite como su señal de salida hacia otros nodos.
- Subredes dentro del sistema que cumplen distintas funciones en la secuencia general de recibir señales, procesarlas, y generar las respuestas adecuadas.
- Aprendizaje adaptativo mediante el cual se alcanza la óptima correspondencia *estímulo-respuesta* a través de la repetición y el condicionamiento.

El objetivo de esta sección es dar precisión a estos aspectos para definir las bases de los modelos de redes neuronales artificiales. El último de los ítems en la lista (el aprendizaje y la optimización del sistema), se tratará en detalle en el próximo capítulo.

2.2.1. Las unidades de procesamiento

Introducimos el modelo enfocándonos en la actividad de una unidad genérica de procesamiento, también conocido como unidad de cómputo o simplemente *nodo* de la red, como se ilustra en la Fig. 2.3. Este nodo (representado por un círculo en la figura), recibe señales de intensidad y_j provenientes

de otras partes de la red, a través de las conexiones dirigidas correspondientes (indicadas con flechas). Cada una de estas conexiones tiene asociada una fuerza o *peso* dado por w_j que regula la fracción de la señal y_j (es decir, la señal que viaja por dicha conexión) que ingresa al nodo en cuestión. Así, la *entrada o excitación neta* x está dada por

$$x = \sum_{j=1}^{k} w_j y_j = w_1 y_1 + w_2 y_2 + \cdots + w_k y_k \quad . \tag{2.1}$$

La regla o ley de integración (2.1) expresa que las señales que llegan por distintas conexiones están moduladas de acuerdo al peso de cada una[2], lo cual no es más que una forma simple y elegante de corresponder a lo mencionado anteriormente sobre las distintas eficiencias de las sinapsis para transmitir señales de una neurona a otra. Luego de la integración de las señales entrantes, el próximo

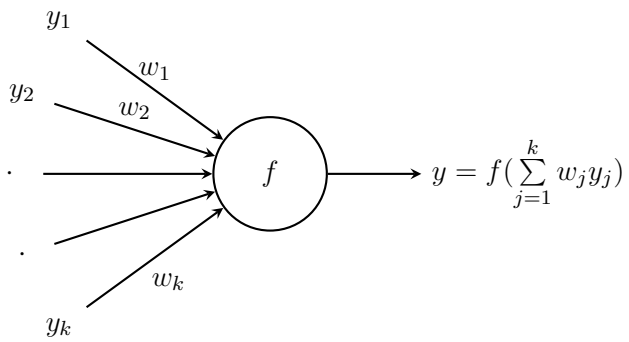

Figura 2.3. Nodo de procesamiento de las redes. El nodo (círculo blanco), recibe las señales y_j a través de las conexiones con pesos w_j, las integra en una señal neta según la Ec. 2.1 y computa su señal de salida a través de f.

paso en la tarea de cómputo consiste en procesar el resultado a través de cierta función f asociada a dicho nodo, denominada *función de transferencia*[3]. De esta forma, la señal de salida x del nodo está dada por

$$y = f(x) = f(\sum_{j=1}^{k} w_j y_j) \quad . \tag{2.2}$$

En pocas palabras, un nodo no es más que un elemento que integra las señales númericas que recibe por distintos canales y calcula con la señal integrada el valor de su función característica, valor que a su vez constituye la señal de salida.

[2]Se pueden considerar otras formas de integrar las señales [22], pero esta es la más ampliamente usada.

[3]Es posible introducir un paso intermedio entre la integración de las señales entrantes y la generación de la señal de salida teniendo en cuenta el instante de arribo de las señales y el estado previo del nodo [4]. En nuestro trabajo, la dinámica de las redes prescindirá de este paso intermedio, así como de la variable temporal y los estados almacenados.

2.2.2. Sistemas de nodos: Redes

Desde el punto de vista teórico, la base de cualquier modelo computacional consiste en definir sus funciones primitivas y las reglas de composición entre ellas [2], lo cual determina aquellas funciones que podrán computarse dentro del modelo. Dicho de otra forma, las funciones *computables* por el modelo son todas aquellas funciones que pueden expresarse en términos de las funciones primitivas y las composiciones permitidas.

Para introducir las redes a partir de los nodos vistos en la sección anterior, empezamos por establecer la conexión entre esta perspectiva general y las redes neuronales artificiales, considerando en la Figura 2.4 una red relativamente simple, compuesta por tres nodos n_j, cada uno de los cuales tiene su función de transferencia $f_j : A_j \subseteq \mathbb{R} \to \mathbb{R}$. El sistema recibe, a través de los nodos n_1 y n_2, una señal $(x_1,x_2) \in A \subseteq \mathbb{R}^2$. Cada uno de estos nodos produce una señal de salida $f_j(x_j)$ $(j = 1,2)$ que transmite al nodo n_3 a través de las conexiones con pesos w_{j3}. Por último, este nodo integra las señales entrantes (de acuerdo con la Ec. 2.1) y produce su salida y dada por

$$y = f_3\left[w_{13}f_1(x_1) + w_{23}f_2(x_2)\right] \quad , \tag{2.3}$$

la cual a su vez constituye la respuesta de la red a la señal entrante (x_1,x_2). Esto significa que la red es equivalente a un mapeo $F : A \subseteq \mathbb{R}^2 \to \mathbb{R}$ dado por

$$F = f_3 \circ (w_{13}f_1 + w_{23}f_2) \quad . \tag{2.4}$$

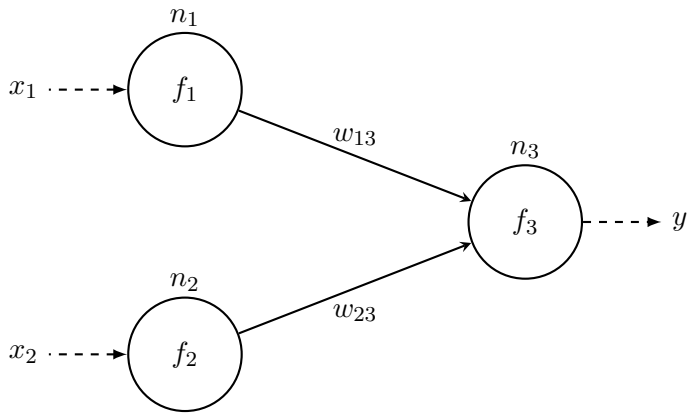

Figura 2.4. Red *feed-forward*. La red procesa las señales en una sola dirección.

El mapeo F se suele denominar la *función de red*, y se puede ver que su forma depende de las funciones de transferencia f_j, de los pesos de las conexiones w_{i3} y de la topología o el diseño estructural

de la red. El procesamiento de la información en este tipo particular de red es unidireccional y es la base para la construcción de redes multicapa que se verá en la próxima sección.

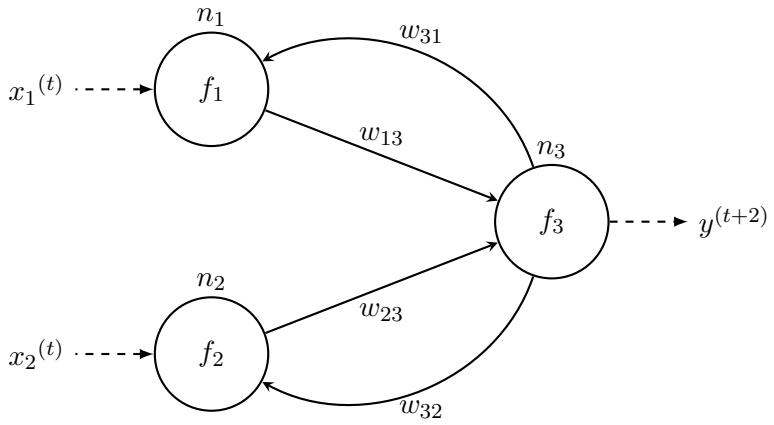

Figura 2.5. Red recurrente. El nodo n_3 retroalimenta a la red a través de las conexiones con pesos w_{31} y w_{32}.

En la Figura 2.5 se muestra otro tipo de configuración en donde existen ciclos que retroalimentan a determinados nodos (red con *feedback*). Si bien la función de red sigue dependiendo de su estructura, del tipo de nodo y de los pesos, en estos sistemas es necesario introducir una variable pseudotemporal o de control. De manera que si, por ejemplo, en el instante t el nodo n_1 recibe las señales $x_1{}^{(t)}$ y $y^{(t)}$, su señal de salida correspondiente $f_1(x_1{}^{(t)} + w_{31}y^{(t)})$ se emitirá y llegará al nodo n_3 en el instante $t+1$. De forma similar, en el instante $t+1$ llegará a n_3 la señal $f_2(x_2{}^{(t)} + w_{32}y^{(t)})$ proveniente de n_2, con lo cual la salida del sistema en $t+2$ será

$$y^{(t+2)} = f_3[w_{13}f_1(x_1{}^{(t)} + w_{31}y^{(t)}) + w_{23}f_2(x_2{}^{(t)} + w_{32}y^{(t)})] \quad . \tag{2.5}$$

Estos dos casos simples, ejemplifican dos grandes clasificaciones de las redes neuronales artificiales: las redes con procesamiento unidireccional (p. ej. el modelo del Perceptrón [5]) y las redes recurrentes (p. ej. el modelo de Hopfield [23]). No obstante, lo esencial y común a ellas desde la perspectiva teórico computacional que mencionamos anteriormente, es que computan una función de red, que queda determinada por (a) un conjunto de funciones primitivas, que no son más que las funciones de transferencia de cada nodo; y (b) las reglas de composición entre ellas, las cuales están implícitas en la topología de la red, y el modo de sincronización de procesamiento en el caso de redes recurrentes.

Para resumir, el mapeo o la correspondencia entrada-salida de la red depende de tres elementos fundamentales:

- La naturaleza de los nodos; en rigor, la función de transferencia de cada uno.

- La topología de la red.
- Los pesos de las conexiones.
- El tipo de sincronización de procesamiento.

2.2.3. Funciones de transferencia

En el caso general, cada nodo de la red puede tener su propia función de transferencia, con lo cual el tipo de señal de salida dependerá de cada uno. Comúnmente y a nuestros fines, la convención más simple es que f sea la misma para todos los nodos y la intensidad tanto de las señales entrantes a la red como de las señales que emiten los nodos, esté en el intervalo $[0,1]$ (notar que, según la Fig. 2.3, esto incluye tanto a y_j como a y). A su vez, los pesos de las conexiones pueden tomar cualquier valor real[4], y por ende también la excitación neta x (ver Ec. 2.1). Esto significa que la función de salida f debe ser $f : A \subseteq \mathbb{R} \to [0,1]$. El carácter todo-o-nada de activación en la neurona sugiere que la forma más adecuada de la función de transferencia f es la función escalón $H_\theta : \mathbb{R} \to \{0,1\}$ dada por

$$H_\theta(x) = \begin{cases} 1 & \text{si } x \geq \theta \\ 0 & \text{si } x < \theta \end{cases} , \tag{2.6}$$

de forma tal que

$$f(x) = H_\theta(x) = \begin{cases} 1 & \text{si } x \geq \theta \\ 0 & \text{si } x < \theta \end{cases} , \tag{2.7}$$

con lo cual se tiene que un nodo no activará ($= 0$) a menos que la señal neta de entrada x supere el valor de θ, lo que queda definido entonces como el umbral de activación del nodo. Una forma equivalente y más conveniente de introducir este umbral es incluyéndolo como un sesgo (*offset*) en el argumento de la función de transferencia, de manera que la señal de salida esté dada por

$$f(x-\theta) = H_0(x-\theta) = \begin{cases} 1 & \text{si } x-\theta \geq 0 \\ 0 & \text{si } x-\theta < 0 \end{cases} . \tag{2.8}$$

Como se puede ver en la Fig. 2.6, esto equivale a conectar cada nodo con una señal constante unitaria mediante una conexión de peso $-\theta$. Esta será la forma en la que tendremos en cuenta los umbrales en nuestro trabajo.

[4]La inclusión de pesos en las conexiones no es estrictamente necesaria, como en el caso de las unidades de McCulloch-Pitts [1].

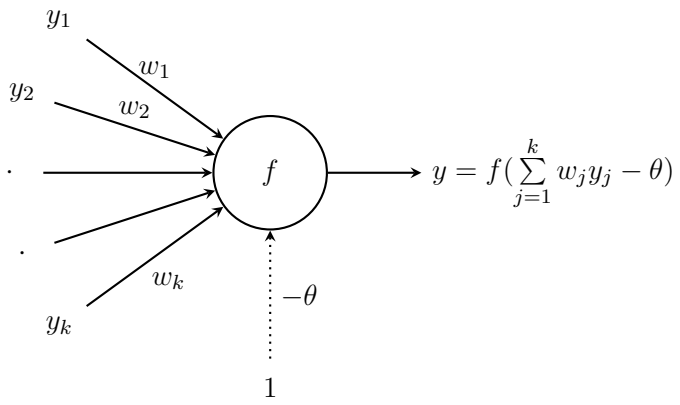

Figura 2.6. Nodo con señal constante 1 conectado con fuerza $-\theta$ al nodo bajo interés, como forma conveniente de implementar el umbral θ.

Una consecuencia de la introducción de un umbral y de la forma de x (Ec. 2.1) es que, al permitir que los pesos de las conexiones puedan tomar valores tanto positivos como negativos, se admite el aspecto análogo al carácter excitatorio e inhibitorio (resp.) de las sinapsis [11], ya que aquellos términos en (2.1) en los cuales las señales estén multiplicadas por pesos negativos se opondrán a que x supere el umbral θ, favoreciendo el caso $f(x-\theta) = 0$. De forma similar, los pesos positivos favorecerán $f(x-\theta) = 1$.

Si bien hay modelos que implementan funciones escalón como funciones de transferencia, como las unidades de McCulloch-Pitts [1] y el perceptrón [5], en otros casos resulta conveniente (como se verá más adelante) tomar una versión diferenciable de la función escalón. Una de las más comunes es la sigmoidea $s_\beta : \mathbb{R} \to (0,1)$ dada por

$$s_\beta(x) = \frac{1}{1+\exp(-\beta x)} \quad . \tag{2.9}$$

En la Figura 2.7 se muestran estos dos tipos de funciones. Se puede ver que, asintóticamente, s_β se comporta como H_0, y que en torno al origen la semejanza aumenta con mayor β.

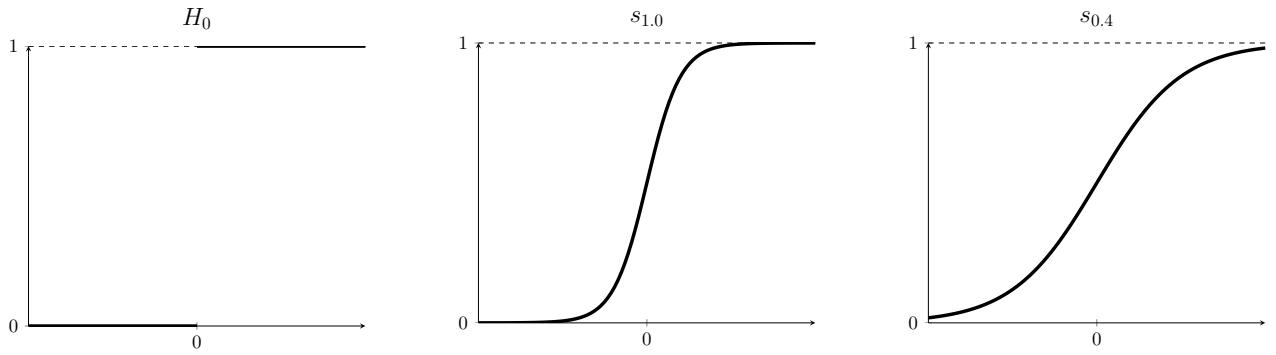

Figura 2.7. Funciones H_0 y s_β utilizadas comúnmente como funciones de transferencia en los nodos.

2.2.4. Redes multicapa

En esta sección, presentamos el tipo de redes con el que trabajaremos en nuestro estudio, llamadas redes multicapa con procesamiento unidireccional. La Fig. 2.8 ilustra la estructura general de estas redes. La configuración consta de subconjuntos o capas de nodos (arreglos verticales de nodos en la figura) que procesan las señales en simultáneo, y conexiones que comunican a cada nodo de cierta capa con todos los nodos de la capa siguiente.

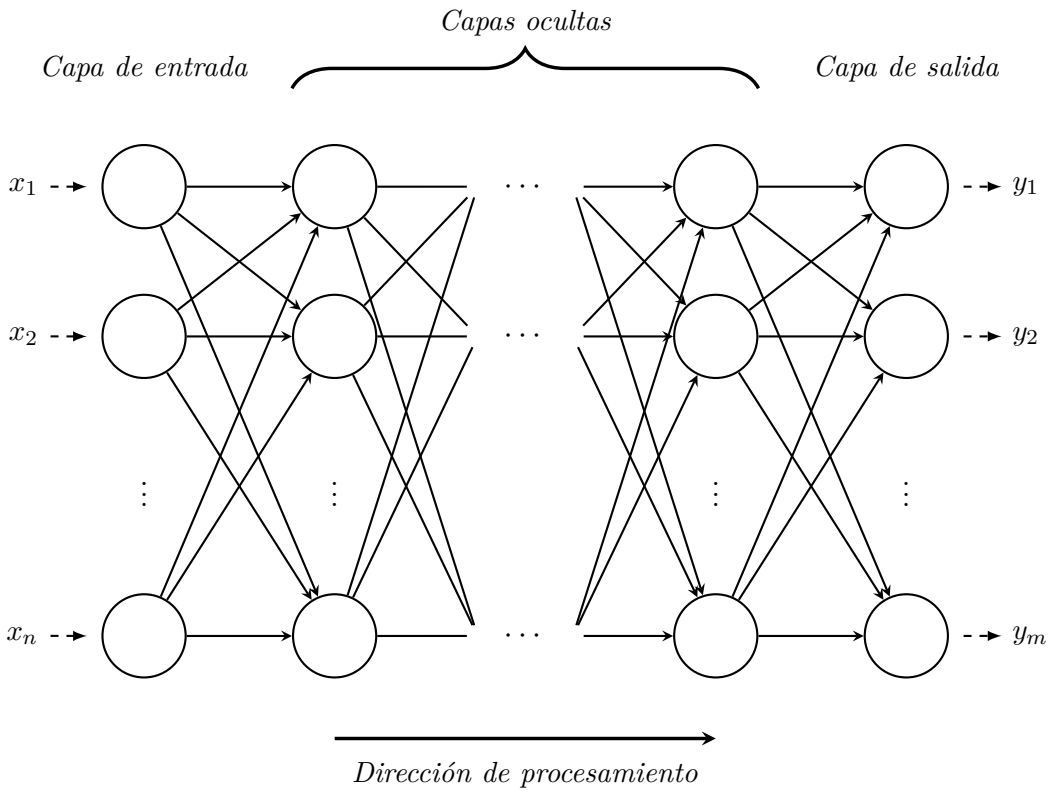

Figura 2.8. Red multicapa con procesamiento unidireccional, conocida como *feed-forward*, con señal de entrada y de salida n y m-dimensional, respectivamente. Las capas ocultas se encargan del procesamiento.

Hay tres tipos de capas. La *capa de entrada* es donde se deposita la señal de entrada de la red $(x_1, x_2, \ldots, x_n)$. Para el procesamiento en pasos discretos de tiempo, el modo de sincronización es el siguiente. Cada uno de los n nodos de entrada procesa simultáneamente, digamos, en el instante t_0, una de las componentes de esta señal, y la transmite a la primera capa de procesamiento. En el próximo instante $t_0 + 1$ todos los nodos de dicha capa procesan la señal que recibieron de la capa de entrada. En general, en el instante $t_0 + k$ todos los nodos de la k-ésima capa procesan las señales que recibieron de la $(k-1)$-ésima capa en el instante $t_0 + k - 1$. Así, las señales resultantes se propagan unidireccionalmente a través de todas las capas de procesamiento de la red, también llamadas *capas ocultas*, cada una de las cuales puede tener una cantidad arbitraria de nodos. Finalmente, los m nodos

de la *capa de salida* reciben la señal resultante, la procesan una última vez y producen la señal de salida de la red $(y_1, y_2, \ldots, y_m)$.

Generalmente, la cantidad de capas ocultas y de nodos que cada una posea dependerá de la función para la cual se desee implementar la red. Especificada dicha función, una cantidad relativamente demasiado baja de capas y nodos ocultos dificultará que la red aprenda o se optimice para cumplirla. En el otro extremo, si se exceden estas cantidades la red se sobreajustará a los datos de la muestra utilizados para optimizarla (ver Secc. 3.1), y tendrá un rendimiento indeseado al procesar elementos ajenos a ella.

Con respecto a las capacidades computacionales de las redes *feed-forward*, existen resultados [24, 25] que demuestran que las redes multicapa con funciones del tipo sigmoideas[5] en los nodos, pueden aproximar cualquier función continua con dominio $[0,1]^n$. Adicionalmente, es posiblde dar estimaciones de la cantidad de nodos ocultos necesarios para la aproximación en función del nivel de precisión que se desee [25].

[5]Esto es, $\sigma : \mathbb{R} \to [0,1] / \lim_{t\to\infty} \sigma(t) = 1 \wedge \lim_{t\to-\infty} \sigma(t) = 0$

Capítulo 3

Aprendizaje y optimización

En el capítulo anterior, presentamos los fundamentos estructurales y de procesamiento de las redes neuronales artificiales, abstrayéndonos de su uso para una aplicación en particular. Se vio que una red neuronal artificial determina una función de red, la cual, en el caso de redes que no poseen ciclos de retroalimentación y que nos permiten prescindir de la variable temporal, depende de la forma en la que cada nodo procesa la información (es decir, la función de transferencia de cada nodo), del patrón de conexión entre los nodos de la red, y de los pesos asociados a las conexiones.

Ahora bien, el fin último en el diseño de las redes está en lograr que estas puedan realizar determinada tarea de procesamiento, en implementaciones tan diversas como pueden ser el reconocimiento de patrones gráficos [26] y sonoros [27] complejos, la detección de procesos fundamentales en la física de partículas [28], y la clasificación diagnóstica entre distintos tipos de cáncer [29, 30]. Cualquiera que sea el caso, existe una correspondencia ideal *entrada-salida* asociada a cada tarea y algunas pocos ejemplos o casos particulares de esta correspondencia, a partir de los cuales la red debe no solo reproducir dichas instancias, sino que debe poder interpolar o *generalizar* satisfactoriamente a otros casos. Para ser precisos, si Φ es esta correspondencia ideal o *función objetivo* determinada por cierta tarea, entonces la función de red F debe aproximar a Φ lo mejor posible siendo que sólo conoce los valores de Φ en una cantidad relativamente baja de puntos.

El presente capítulo trata sobre aquella parte fundamental en el diseño de las redes encargada de esta *optimización* funcional: el *aprendizaje* de la red. En la primera sección delineamos los bases generales de cualquier proceso de aprendizaje, y en la Secc. 3.2 presentamos un método clásico para el entrenamiento de las redes: el aprendizaje por el método de retropropagación. En la Secc. 3.3 damos a conocer el método de aprendizaje con el cual trabajaremos en nuestro estudio: el aprendizaje autónomo. Concluimos el capítulo aplicando a modo ilustrativo los dos tipos de aprendizaje en un ejemplo artificial: una red cuya tarea es aproximar $\Phi(x,y) = \cos(2\pi y)\sin(2\pi x)$.

3.1. Aspectos generales

3.1.1. El problema y los tipos de aprendizaje

El aprendizaje de las redes es un proceso adaptativo en el cual las propiedades de una red, ya sea su topología o los pesos de las conexiones, se someten a cambios o mutaciones de manera que la función de red resultante vaya aproximando cada vez mejor a la función objetivo. Lo más común es tomar como parámetros libres del aprendizaje a los pesos de las conexiones, en sintonía con el paradigma Hebbiano de aprendizaje [21] según el cual los cambios adaptativos y el almacenamiento de información involucrado en al aprendizaje están asociados a la plasticidad de las fuerzas sinápticas (ver Secc. 2.1).

Para ser claros en la discusión que sigue, exponemos el problema del aprendizaje de las redes en términos precisos, para un tipo concreto de situación:

- Se tiene una tarea para la cual se desea implementar una red. Esta tarea define una *función objetivo*, que notamos con Φ. Si la entrada es n-dimensional y la salida m-dimensional, esta función es $A \subseteq \mathbb{R}^n \to \mathbb{R}^m$. En consecuencia, la única condición que debe satisfacer la red, *a priori*, es tener n nodos de entrada y m nodos de salida, es decir, su función de red F debe ser $A \subseteq \mathbb{R}^n \to \mathbb{R}^m$.
- Φ no se conoce explícitamente, sólo su aplicación en un conjunto reducido de puntos $\Omega_0 = \{\chi_1, \ldots, \chi_N\} \subset \mathbb{R}^n$. Es decir, sólo se cuenta con $\{(\chi_1, \Phi(\chi_1)), \ldots, (\chi_N, \Phi(\chi_N))\} \subset \mathbb{R}^n \times \mathbb{R}^m$. No obstante, y como hecho clave, se tiene algún tipo de noción sobre cómo debe ser Φ en otros puntos que no pertenecen a este conjunto.
- El objetivo del aprendizaje es lograr que $F \to \Phi$ usando el conocimiento de Φ en Ω_0, pero que a su vez la aproximación sea satisfactoria para puntos $x \in \mathbb{R}^n$ que no estén en Ω_0, es decir, que la red interpole satisfactoriamente. A esta interpolación la llamamos *generalización* de lo aprendido.

Con esto en consideración, el tipo de aprendizaje se puede clasificar de distintas formas [31], entre las cuales mencionamos aquellas más estrechamente relacionadas con nuestros propósitos:

Aprendizaje supervisado. Este tipo de aprendizaje corresponde a la situación de la lista anterior, donde se cuenta con los valores de la función Φ en los puntos de Ω_0, y este conocimiento se usa para entrenar a la red.

Aprendizaje no supervisado. En este caso, se desconoce Φ completamente. Sólo se cuenta con un conjunto de datos y la dimensión n con la que están dados, y el objetivo del aprendizaje es que la red agrupe los elementos de este conjunto en distintas categorías (o *clusters*) extrayendo determinados rasgos característicos de los datos. Este tipo de problemática surge, por ejemplo, cuando se desea reducir la dimensionalidad de la muestra de datos descartando aquella información presente en ellos que sea irrelevante.

Aprendizaje por refuerzo. Esta clase de aprendizaje se distingue de las demás en que la red no recibe de un agente externo la prescripción (*policy* [31]) para la modificación de sus parámetros en búsqueda de su optimización, sino que ésta es parte de la red y se da como una ley de evolución de dichos parámetros que los modifica internamente en base a la respuesta de la red a las señales que recibe y que depende fuertemente de la interacción entre la red y su entorno (ver Secc. 3.3).

Adicionalmente, se pueden adoptar algoritmos de aprendizaje que sean combinación de estos tres. Por ejemplo, el aprendizaje autónomo con el que trabajaremos, es una combinación de aprendizaje por refuerzo y supervisado.

En el caso del aprendizaje supervisado, el conjunto $\{(\chi_1,\Phi(\chi_1)),\ldots,(\chi_N,\Phi(\chi_N))\}$, se denomina *conjunto de aprendizaje* (*training set*). Para evaluar qué tan lejos está una red de su rendimiento óptimo con respecto a este conjunto y así poder dar una dirección efectiva a las modificaciones de la red durante el proceso de aprendizaje, se introduce el llamado *error de aprendizaje*, que denotamos ε, y está dado por

$$\varepsilon = \eta \sum_{p=1}^{N} |F(\chi_p) - \Phi(\chi_p)|^2 \quad , \tag{3.1}$$

donde η es una constante de normalización y $|\cdot|$ denota la norma euclideana, en nuestro caso, $|\cdot| : \mathbb{R}^m \to \mathbb{R}$. El error de aprendizaje mide qué tan lejos está la función de red F de su objetivo Φ, pero sólo para aquellos puntos del conjunto de aprendizaje. En general, ε es la magnitud que los algoritmos de aprendizaje tratan de *minimizar* introduciendo modificaciones en la red. Por ejemplo, el método *simulated annealing* [7] consiste en (a) ejecutar cierta modificación o mutación (por ejemplo, cambiar el valor de cierto peso en una conexión), evaluar ε y compararlo con su valor antes de que la red se haya modificado; (b) si ε disminuyó como resultado de la mutación, se acepta el cambio y se repite (a) con una nueva modificación; si ε aumentó, el cambio no se acepta, o se acepta con cierta frecuencia de probabilidad; (c) el proceso se repite hasta que ε haya disminuido a determinado valor.

Una vez que el algoritmo de aprendizaje concluye, para medir la efectividad del entrenamiento de la red es necesario evaluar su desempeño sobre puntos que no hayan sido usados para entrenarla, es

decir, sobre un conjunto $\Omega = \{\xi_1, \ldots, \xi_M\} \subseteq A - \Omega_0 \subseteq \mathbb{R}^n$. El conjunto Ω se denomina *conjunto de prueba*. Es necesario tener algún nivel de conocimiento sobre el comportamiento que se espera de la red al evaluar estos puntos. En el caso ideal, se conoce la función Φ en dicho conjunto, es decir, se cuenta con $\{(\xi_1, \Phi(\xi_1)), \ldots, (\xi_M, \Phi(\xi_M))\}$ y se puede calcular el llamado *error de prueba* E [11], dado por

$$E = \eta' \sum_{q=1}^{M} [F(\xi_q) - \Phi(\xi_q)]^2 \quad . \tag{3.2}$$

La distinción entre el error de aprendizaje y el error de prueba de una red, es análoga a la diferencia entre la mera memorización de los datos y el aprendizaje con capacidad de generalización [31]. Cuando el algoritmo de aprendizaje condujo a un error de aprendizaje muy pequeño pero el error de prueba es inaceptable, se dice que la red se *sobreajustó* a los datos del conjunto de aprendizaje [2].

Como se verá más adelante, en nuestro caso los puntos χ_i serán patrones en una matriz de píxeles, y ξ_j los patrones originales pero con defectos introducidos (píxeles invertidos), y se esperará que la red pueda seguir clasificándolos correctamente.

3.2. Aprendizaje por retropropagación

3.2.1. Formulación general

El aprendizaje por el método de retropropagación [9] es un método analítico aplicado a redes multicapa para minimizar el error ε, tomando como parámetros libres a los pesos de la red[1]. El principio general detrás del algoritmo es simple:

1. Se inician los pesos de la red $\boldsymbol{w} = (w_1, \ldots, w_k)$ (generalmente con una distribución pequeña en torno a cero [9]). Se procesan los elementos del conjunto de aprendizaje, y se calcula el error ε.
2. Se calcula $\nabla_{\boldsymbol{w}}\varepsilon = (\frac{\partial\varepsilon}{\partial w_1}, \ldots, \frac{\partial\varepsilon}{\partial w_k})$.
3. Se modifican los pesos según

   $$\Delta\boldsymbol{w} = -\alpha\nabla_{\boldsymbol{w}}\varepsilon \quad , \tag{3.3}$$

   donde α es una constante positiva denominada *tasa de aprendizaje*.
4. Se repite el proceso con los pesos de la red actualizados $\boldsymbol{w}' = \boldsymbol{w} + \Delta\boldsymbol{w}$.

[1] Evidentemente, ε depende de los pesos de la red ya que depende de la función de red F.

De esta manera, el aprendizaje por retropropagación no hace más que buscar el mínimo del error en el espacio de los pesos de la red. Así, una combinación de pesos que minimice ε es una solución al problema de aprendizaje [2].

El primer punto a notar es que se debe exigir la diferenciabilidad de la función error. Como el producto por un escalar y la composición de funciones conservan la diferenciabilidad, basta con tomar una función de transferencia diferenciable para los nodos de la red. Las sigmoideas s_β vistas en la Secc. 2.2.3, por ejemplo, garantizan esto, y es directo verificar que s'_β está dada por

$$s'_\beta(x) = \beta s_\beta(x)[1 - s_\beta(x)] \quad . \tag{3.4}$$

El segundo punto a notar es que si los pesos iniciales son tales que el error se encuentra en el entorno de algún mínimo local, el algoritmo convergerá hacia dicho mínimo, y no sé alcanzará el mínimo global. Una manera de disminuir las posibilidades de que esto suceda es repetir el algoritmo con distintas distribuciones iniciales de los pesos. Otra alternativa es agregar un término pseudoinercial a la corrección de los pesos, de forma tal que la corrección de los pesos en cierta iteración t del algoritmo sea proporcional a la corrección anterior, es decir

$$\Delta \boldsymbol{w}(t) = -\alpha_1 \nabla_{\boldsymbol{w}} \varepsilon + \alpha_2 \Delta \boldsymbol{w}(t-1) \quad . \tag{3.5}$$

En lo que sigue, se mostrará cómo encontrar las derivadas $\dfrac{\partial \varepsilon}{\partial w_i}$. Como punto preliminar, se introducen algunos cambios convenientes en la notación. El error ε se puede escribir como

$$\varepsilon = \sum_{p=1}^{N} \varepsilon_p \quad , \tag{3.6}$$

donde se define

$$\varepsilon_p = \eta |F(\chi_p) - \Phi(\chi_p)|^2 \quad , \tag{3.7}$$

y se suele tomar $\eta = 1/2$ [2] para cancelar el factor cuadrático de las derivaciones. De esta forma, la corrección de los pesos está dada por

$$\Delta \boldsymbol{w} = -\alpha \sum_{p=1}^{N} \nabla_{\boldsymbol{w}} \varepsilon_p \quad , \tag{3.8}$$

lo cual permite trabajar con cada ε_p individualmente en la tarea de encontrar las derivadas parciales. Adicionalmente, se denota $F(\chi_p) = y = (y_1, \ldots, y_m)$ y $\Phi(\chi_p) = \gamma = (\gamma_1, \ldots, \gamma_m)$ (es decir, prescindiendo del subíndice p, ya que se trabajará con cada ε_p por separado de forma genérica), con lo

cual

$$\varepsilon_p = \frac{1}{2}|y-\gamma|^2 = \frac{1}{2}\sum_{i=1}^{m}(y_i-\gamma_i)^2 \quad . \tag{3.9}$$

3.2.2. Aplicación a redes *feed-forward*

Aplicamos el algoritmo de retropropagación a una red *feed-forward*, con la sigmoidea s_1 (denotada simplemente s) como función de transferencia en los nodos. Definimos las características de la red y la notación correspondiente a continuación (ver Fig. 3.1):

- La red tiene $k+2$ capas, que indexamos con el arreglo ordenado $(0,1,\ldots,k,k+1)$, siendo 0 la capa de entrada, 1 a k las capas ocultas (en el orden de procesamiento) y $k+1$ la de salida. La cantidad de nodos de la m-ésima capa es n_m.
- Al j-ésimo nodo de la m-ésima capa lo indicamos con $\mathrm{N}(j,m)$.
- El peso que conecta al nodo $\mathrm{N}(i,m-1)$ con el nodo $\mathrm{N}(j,m)$ lo denotamos $w_{ij}^{(m)}$.
- La entrada neta y la salida neta del nodo $\mathrm{N}(j,m)$ son $x_j^{(m)}$ y $y_j^{(m)}$ respectivamente, y están dadas por (ver Ec. 2.1 y 2.2):

$$x_j^{(m)} = \sum_{i=1}^{n_m} w_{ij}^{(m)} y_i^{(m-1)} \tag{3.10}$$

$$y_j^{(m)} = s[x_j^{(m)}] \quad . \tag{3.11}$$

El punto de partida es el error de la red al procesar cierto patrón, el cual podemos reescribir en la nueva notación:

$$\varepsilon_p = \sum_{\ell=1}^{n_{k+1}} \left(y_\ell^{(k+1)} - \gamma_\ell\right)^2 \quad . \tag{3.12}$$

La función error ε_p depende de los pesos de la red a través de la señal de salida $y_\ell^{(k+1)}$ de los nodos en la última capa. Para encontrar la derivada de ε_p respecto a cierto peso $w_{ij}^{(m)}$, la idea es aplicar sucesivamente la regla de la cadena, empezando por la expresión 3.12 asociada a los nodos de salida, y moviéndose de capa en capa *hacia atrás* hasta llegar a la conexión en cuestión.

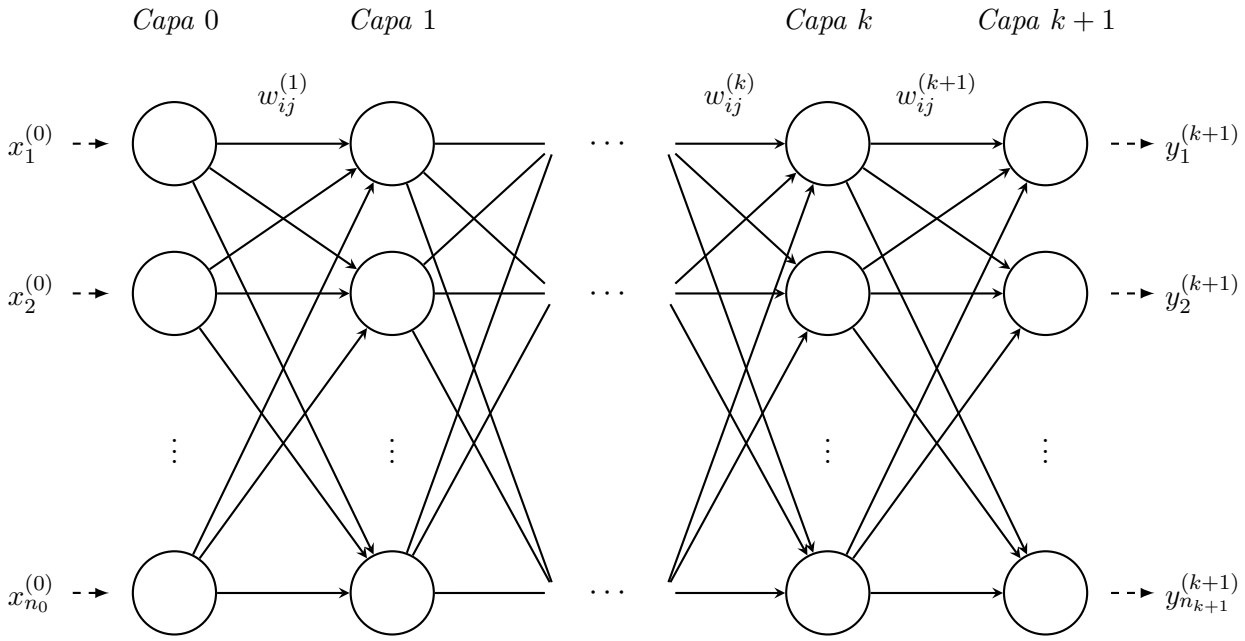

Figura 3.1. Red *feed-forward* con la notación empleada para desarrollar el algoritmo de aprendizaje por retropropagación. La red tiene $k+1$ capas, indexándose como 0 y $k+1$ las capas de entrada y de salida, resp. Los pesos asociados a un par de capas $m-1$ y m se denotan $w_{ij}^{(m)}$, mientras que la entrada neta y la salida del nodo j en la capa m se denotan $x_j^{(m)}$ y $y_j^{(m)}$.

Consideremos cierto peso $w_{ij}^{(k+1)}$. Partiendo de (3.12) y aplicando la regla de la cadena:

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k+1)}} = \sum_{\ell=1}^{n_{k+1}} \frac{\partial \varepsilon_p}{\partial y_\ell^{(k+1)}} \frac{\partial y_\ell^{(k+1)}}{\partial x_\ell^{(k+1)}} \frac{\partial x_\ell^{(k+1)}}{\partial w_{ij}^{(k+1)}} \quad . \tag{3.13}$$

Observando (3.10), el tercer factor de cada término resulta

$$\frac{\partial x_\ell^{(k+1)}}{\partial w_{ij}^{(k+1)}} = \begin{cases} 0 & , \quad \ell \neq j \\ y_i^{(k)} & , \quad \ell = j \end{cases} \quad , \tag{3.14}$$

de manera que el único término que contribuye en la suma sobre ℓ es aquel con $\ell = j$. Así,

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k+1)}} = \frac{\partial \varepsilon_p}{\partial y_j^{(k+1)}} \frac{\partial y_j^{(k+1)}}{\partial x_j^{(k+1)}} y_i^{(k)} \quad . \tag{3.15}$$

Los dos primeros factores en el lado derecho de (3.15) se agrupan en una magnitud $\delta_j^{(k+1)}$ llamada *error retropropagado hasta el nodo* $\mathrm{N}(j,k+1)$, dada por

$$\delta_j^{(k+1)} = \frac{\partial \varepsilon_p}{\partial y_j^{(k+1)}} \frac{\partial y_j^{(k+1)}}{\partial x_j^{(k+1)}} \quad , \tag{3.16}$$

de manera que

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k+1)}} = y_i^{(k)} \delta_j^{(k+1)} \quad . \tag{3.17}$$

Consideramos ahora cierto peso $w_{ij}^{(k)}$ entre un par anterior de capas. De manera análoga, pero con un paso más de derivación, se tiene

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k)}} = \sum_{\ell=1}^{n_{k+1}} \delta_\ell^{(k+1)} \frac{\partial x_\ell^{(k+1)}}{\partial w_{ij}^{(k)}} \tag{3.18}$$

$$\frac{\partial x_\ell^{(k+1)}}{\partial w_{ij}^{(k)}} = \sum_{\ell'=1}^{n_k} w_{\ell'\ell}^{(k+1)} \frac{\partial y_{\ell'}^{(k)}}{\partial x_{\ell'}^{(k)}} \frac{\partial x_{\ell'}^{(k)}}{\partial w_{ij}^{(k)}} \tag{3.19}$$

$$\frac{\partial x_{\ell'}^{(k)}}{\partial w_{ij}^{(k)}} = \begin{cases} 0 & , \quad \ell' \neq j \\ y_i^{(k-1)} & , \quad \ell' = j \end{cases} \quad , \tag{3.20}$$

donde hemos usado (3.16). Reemplazando (3.20) en (3.19),

$$\frac{\partial x_\ell^{(k+1)}}{\partial w_{ij}^{(k)}} = w_{j\ell}^{(k+1)} \frac{\partial y_j^{(k)}}{\partial x_j^{(k)}} y_i^{(k-1)} \quad , \tag{3.21}$$

y esto último en (3.18),

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \frac{\partial y_j^{(k)}}{\partial x_j^{(k)}} \sum_{\ell=1}^{n_{k+1}} w_{j\ell}^{(k+1)} \delta_j^{(k+1)} \quad . \tag{3.22}$$

En este caso, se define el error retropropagado hasta el nodo N(j,k) como

$$\delta_j^{(k)} = \frac{\partial y_j^{(k)}}{\partial x_j^{(k)}} \sum_{\ell=1}^{n_{k+1}} w_{j\ell}^{(k+1)} \delta_\ell^{(k+1)} \quad , \tag{3.23}$$

con lo cual se tiene finalmente

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(k)}} = y_i^{(k-1)} \delta_j^{(k)} \quad . \tag{3.24}$$

Se puede ver entonces que, definiendo recursivamente el error retropropagado, es posible expresar la derivada de ε_p con respecto a cualquier peso $w_{ij}^{(m)}$:

$$\frac{\partial \varepsilon_p}{\partial w_{ij}^{(m)}} = y_i^{(m-1)} \delta_j^{(m)} \tag{3.25a}$$

$$\delta_j^{(m)} = \begin{cases} s'[x_j^{(m)}](y_j^{(m)} - \gamma_j) & , \, m = k+1 \\ s'[x_j^{(m)}] \sum\limits_{\ell=1}^{n_{m+1}} w_{j\ell}^{(m+1)} \delta_\ell^{(m+1)} & , \, m < k+1 \end{cases} \quad , \tag{3.25b}$$

donde hemos usado (3.11) y (3.12) para calcular las derivadas correspondientes en (3.25b). Podemos entonces dar una formulación precisa del algoritmo de aprendizaje por retropropagación:

Algoritmo de aprendizaje por retropropagación:

0. Iniciar los pesos de la red con valores cercanos a cero.

1.1. (*Paso hacia adelante*) Procesar el p-ésimo elemento del conjunto de aprendizaje. Computar y almacenar en cada nodo el valor de $s'[x_j^{(m)}]$ y $y_j^{(m)}$.

1.2. (*Paso hacia atrás*) Computar el error retropropagado para todos los nodos de la red según la definición 3.25b, empezando por los de la capa de salida ($m = k + 1$) y siguiendo hacia capas anteriores ($m < k + 1$).

1.3. (*Corrección respecto de p*) Computar $\dfrac{\partial \varepsilon_p}{\partial w_{ij}^{(m)}}$ según la ecuación 3.25a, para todo $w_{ij}^{(m)}$, y almacenar el valor

$$\Delta_p w_{ij}^{(m)} = -\alpha \frac{\partial \varepsilon_p}{\partial w_{ij}^{(m)}} \quad .$$

2. Repetir los pasos ***1.1*** a ***1.3*** para los demás elementos del conjunto de aprendizaje y corregir los pesos de la red según

$$\Delta w_{ij}^{(m)} = \sum_{p=1}^{N} \Delta_p w_{ij}^{(m)} \quad .$$

3. Repetir los pasos ***1*** a ***2*** con los pesos de la red corregidos.

□

3.3. Aprendizaje autónomo

El carácter *adaptativo* de los sistemas biológicos es producto de modificaciones estructurales internas [32] ejecutadas autónomamente en base a la experiencia y la experimentación con el fin de alcanzar cierto *comportamiento* (o *correspondencia estímulo-respuesta*) *ideal* en la interacción con su entorno. En los modelos de redes neuronales artificiales, la forma más común de incorporar esta plasticidad estructural en pos de la adaptación es a través de los pesos sinápticos $\boldsymbol{w}$, en el marco de lo que se denomina aprendizaje Hebbiano [21]. A su vez, los elementos que conforman la interacción de la red con su entorno son: (a) la recepción de una señal de entrada (*estímulo*); (b) la emisión de una señal de salida (*respuesta*) correspondiente; y (c) un tercer tipo de señal de retroalimentación que informa

a la red sobre qué tan apropiada fue su respuesta a la señal que recibió (como, por ejemplo, el error ε que hemos definido en la Ec. 3.1). Este último elemento, si bien a primera vista artificioso, es lo que guía todo proceso de aprendizaje por refuerzo en los seres vivos [33].

El método de aprendizaje por retropropagación es, desde el punto de vista matemático, la solución analítica más natural al problema de aprendizaje de las redes entendido meramente como un problema de optimización y aplicado a la desviación del comportamiento ideal (el error ε) como función de los parámetros de ajuste (los pesos $\boldsymbol{w}$):

$$\Delta \boldsymbol{w} \propto -\nabla_{\boldsymbol{w}} \varepsilon \ . \tag{3.26}$$

Sin embargo, en lo que respecta a la relación entre los modelos de inteligencia artificial y sus contrapartes en los sistemas biológicos, la aparentemente inocua ley de cambio expresada en (3.26) tiene al menos dos características indeseadas.

La primera de ellas es su excesiva (de hecho, infinita) localidad, implícita en el carácter *diferencial* de la ley. Según (3.26), el cambio Δw de los pesos de la red entre un instante y el posterior es en la dirección en la que un cambio *infinitesimal* de los pesos disminuiría máximamente el error. De cierta forma, interpretado desde el punto de vista de un sistema modificándose para adaptarse, esto implica que la red sabe cuál es la mejor modificación antes de ejecutarla. Lo único que salva la consistencia de esto en el marco de la interpretación adaptativa, es considerar en la dinámica de la red a un agente externo o supervisor que, además de conocer por completo su estructura, la *guía* con las herramientas del cálculo infinitesimal hasta su óptima configuración.

La segunda característica indeseada es el excesivo nivel de detalle con el que la red recibe información sobre su desempeño. Es decir, no hay una señal que informe sobre su desempeño *globalmente*, sino que al calcularse el gradiente con respecto a *todos* los pesos de la red, se tiene información sobre el efecto individual que tiene *cada uno de ellos* en dicho desempeño, y esta información se explota con todo su potencial en (3.26) para prescribir las modificaciones de un instante a otro.

Lo que estudiamos en este trabajo es la optimización de las redes por medio de un tipo de aprendizaje estocástico por refuerzo llamado *aprendizaje autónomo* [10]. En él, los parámetros de un sistema en cierto instante evolucionan dependiendo del resultado que tuvieron las modificaciones anteriores en el desempeño de la red con respecto a su comportamiento objetivo. A su vez, la señal que informa sobre la desviación del comportamiento objetivo modula dicha evolución *de la misma forma* para todos ellos, es decir, globalmente. Además, la presencia de una señal ruidosa en la ley de evolución implica que la red se ve obligada a incurrir en modificaciones *exploratorias* en el espacio de parámetros. Si bien en la situación de aprendizaje que planteamos a las redes las respuestas a las señales entrantes están

bien definidas, con lo cual el aprendizaje también califica como supervisado, este conocimiento no se comunica explícitamente a la red sino indirectamente a través de la señal que sirve como *refuerzo* (o, inversa pero equivalentemente, *penalización*), en nuestro caso, el error ε.

En lo que sigue, presentamos el aprendizaje autónomo para sistemas con dinámica continua [10]. Posteriormente, lo formulamos para sistemas con dinámica discreta, tal como lo usaremos para el aprendizaje de nuestras redes.

3.3.1. Sistemas continuos

Consideramos un sistema dinámico con variables de estado $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ [34], cuya evolución está regida por

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{w}) \quad , \tag{3.27}$$

donde $\boldsymbol{f} : E \subseteq \mathbb{R}^n \to \mathbb{R}^n$ depende de los parámetros $\boldsymbol{w} = (w_1, w_2, \ldots, w_k)$. Supóngase que el sistema debe cumplir cierta tarea o función $\boldsymbol{F} : E \subseteq \mathbb{R}^n \to \mathbb{R}^m$ dada por $\boldsymbol{F}(\boldsymbol{x})$, respecto de la cual existe cierto desempeño ideal u objetivo $R_0 \in \mathbb{R}^m$. Para que $\boldsymbol{F}(\boldsymbol{x}) \to R_0$, el sistema debe optimizarse a través de la evolución de los parámetros $\boldsymbol{w}$.

Definiendo la desviación del comportamiento ideal como $\epsilon = |\boldsymbol{F}(\boldsymbol{x}) - R_0|$, el aprendizaje autónomo prescribe dicha evolución mediante el siguiente sistema de ecuaciones diferenciales estocásticas:

$$\tau \frac{d\boldsymbol{w}}{dt} = -[\epsilon(t) - \epsilon(t - \Delta)][\boldsymbol{w}(t) - \boldsymbol{w}(t - \Delta)] + \epsilon S \boldsymbol{\xi}(t) \quad , \tag{3.28}$$

donde $\boldsymbol{\xi}(t) = (\xi_1(t), \xi_2(t), \ldots, \xi_k(t))$ es un arreglo de ruidos gaussianos blancos [35] y correlación nula par a par, es decir, $< \xi_\alpha(t) >= 0$ y $< \xi_\beta(t)\xi_\gamma(t') >= 2\delta_{\beta\gamma}\delta(t - t')$, y S es la intensidad de este ruido. La constante τ define el tiempo característico en la evolución de w y debe satisfacer $\tau >> 1$ en relación al tiempo característico del sistema (3.27). A su vez, la constante de referencia temporal pasada Δ debe satisfacer $\tau >> \Delta >> 1$. Las ecuaciones (3.27) y (3.28) definen un sistema dinámico extendido en las variables $\boldsymbol{x}$ y $\boldsymbol{w}$: mientras que el estado $\boldsymbol{x}$ evoluciona según (3.27), los parámetros $\boldsymbol{w}$ guían lentamente al sistema, a través de (3.28), a minimizar la desviación ϵ con respecto al comportamiento objetivo.

Los elementos esenciales de (3.28) son los siguientes. En primer lugar, la evolución de $\boldsymbol{w}$ depende del rendimiento previo del sistema. Si, como resultado de un cambio $\delta\boldsymbol{w} = \boldsymbol{w}(t) - \boldsymbol{w}(t - \Delta)$, la desviación del objetivo disminuyó, es decir $\delta\epsilon = \epsilon(t) - \epsilon(t - \Delta) < 0$, los próximos cambios $\delta'\boldsymbol{w}$ tenderán a mantener la dirección previa, esto es, $\delta'\boldsymbol{w} \cdot \delta\boldsymbol{w} > 0$. De forma similar, si el rendimiento del sistema empeoró ($\delta\epsilon > 0$) la nueva variación de $\boldsymbol{w}$ tenderá a ser en la dirección opuesta, es decir, $\delta'\boldsymbol{w} \cdot \delta\boldsymbol{w} < 0$.

En segundo lugar, el término difusivo, además de reducir las posibilidades de que el sistema converja en mínimos locales intermedios de ϵ, sirve como un término que *explora* [31] nuevas modificaciones en el espacio de $\boldsymbol{w}$, con lo cual, además de la evolución dirigida aportada por el primer término, el sistema prueba localmente y de forma no causal otras modificaciones en $\boldsymbol{w}$. Por último, como este término está modulado por la desviación ϵ, en el límite del sistema alcanzando la máxima optimización ($\epsilon \to 0$) su contribución es nula.

3.3.2. Sistemas discretos

En nuestros estudios, el aprendizaje de las redes será en pasos discretos de tiempo[2]. Para ello, formulamos el aprendizaje autónomo para sistemas con dinámica discreta.

Algoritmo de aprendizaje autónomo

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = -\gamma(\epsilon_n - \epsilon_{n-1})(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}) + \epsilon_n S\boldsymbol{\xi}_n \tag{3.29}$$

□

Aquí, z_k es el valor de la variable z (w, ϵ, o ξ) en la k-ésima iteración del algoritmo. De la misma forma que en el caso continuo, el arreglo ordenado $\boldsymbol{w}$ contiene todos los parámetros del sistema, y ϵ es la desviación con respecto al comportamiento ideal. La constante γ es una constante positiva que regula la magnitud de la evolución de $\boldsymbol{w}$, mientras que S es la intensidad del ruido gaussiano $\boldsymbol{\xi}$.

Naturalmente, los parámetros $\boldsymbol{w}$ en nuestro estudio de las redes serán los pesos de las conexiones. A su vez, la desviación ϵ se definirá en términos de aquello que queramos optimizar. En el caso del error de aprendizaje ε, como el objetivo es minimizarlo, la desviación asociada a éste será $\epsilon_\varepsilon = |\varepsilon - 0| = \varepsilon$. Como se verá en el Cap. 5, en el caso de la robustez ρ, al ser su valor óptimo la unidad, la desviación correspondiente estará dada por $\epsilon_\rho = |1 - \rho|$.

[2]Es importante aclarar que cada paso discreto de tiempo en el aprendizaje, lo que aquí llamamos *iteración*, no es lo mismo que un paso de tiempo en una red *feed-forward*, sino que corresponde al procesamiento *completo* de una señal por parte de ella, es decir, desde que se deposita la señal en los nodos de entrada, hasta que los nodos de salida emiten la señal de salida correspondiente (ver Secc. 2.2.4).

3.4. Ejemplo

Ejemplificamos la aplicación numérica de los dos tipos de aprendizaje discutidos, tomando como función objetivo $\Phi : [0,1]^2 \to [0,1]$ dada por

$$\Phi(x,y) = 0.5 \times [1 + \cos(2\pi y)\sin(2\pi x)] \tag{3.30}$$

Si bien la parte interesante es únicamente $\cos(2\pi y)\sin(2\pi x)$ debemos ajustarla para que el dominio y la imagen de Φ coincidan con el dominio y la imagen de la función de red F. Como trabajamos con funciones sigmoideas en los nodos, una forma adecuada de hacer esto es (3.30). La red que usamos tiene, por supuesto, *dos* nodos de entrada y *uno* de salida, mientras que implementamos una sola capa oculta con siete nodos para el procesamiento. Los nodos de entrada reciben señales continuamente en el intervalo $[0,1]$, y las señales de salida se encuentran en $[0,1]$.

Como conjunto de aprendizaje, tomamos 100 puntos elegidos aleatoriamente en $[0,1]^2$, junto con sus imágenes en Φ. En el caso del aprendizaje por retropropagación, el valor de la tasa de aprendizaje α con el que optimizamos la red es 0.10. Para el aprendizaje autónomo, tomamos $\gamma = 350.0$ y $S = 0.08$. En ambos casos, iniciamos los pesos de la red con distribución uniforme centrada en el origen y con dispersión 10.0.

La Figura 3.2 muestra los resultados del aprendizaje con 1×10^5 iteraciones del algoritmo en el caso del aprendizaje por retropropagación (columna izquierda) y 1×10^7 con el aprendizaje autónomo (columna derecha). En la última fila se muestra la evolución del error de aprendizaje ε. En las dos primeras filas se grafica la función objetivo Φ (malla de líneas sólidas), la función de red F luego del aprendizaje (malla de líneas discontinuas), y los puntos del conjunto de aprendizaje (círculos blancos). Esto se hace en dos instancias: luego de 100 iteraciones (*primera fila*), y al finalizar la aplicación del algoritmo (*segunda fila*). Sobre el plano xy de estos gráficos se proyecta la diferencia $\Delta \equiv |F(x) - \Phi(x)|$ (mapeo en escala de grises) para todo $x \in [0,1]^2$.

Se puede ver que ambos algoritmos convergen satisfactoriamente a la función objetivo. Adicionalmente, observando la evolución de ε en ambos casos, se puede notar claramente la naturaleza estocástica del algoritmo autónomo versus el carácter analítico del método de retropropagación.

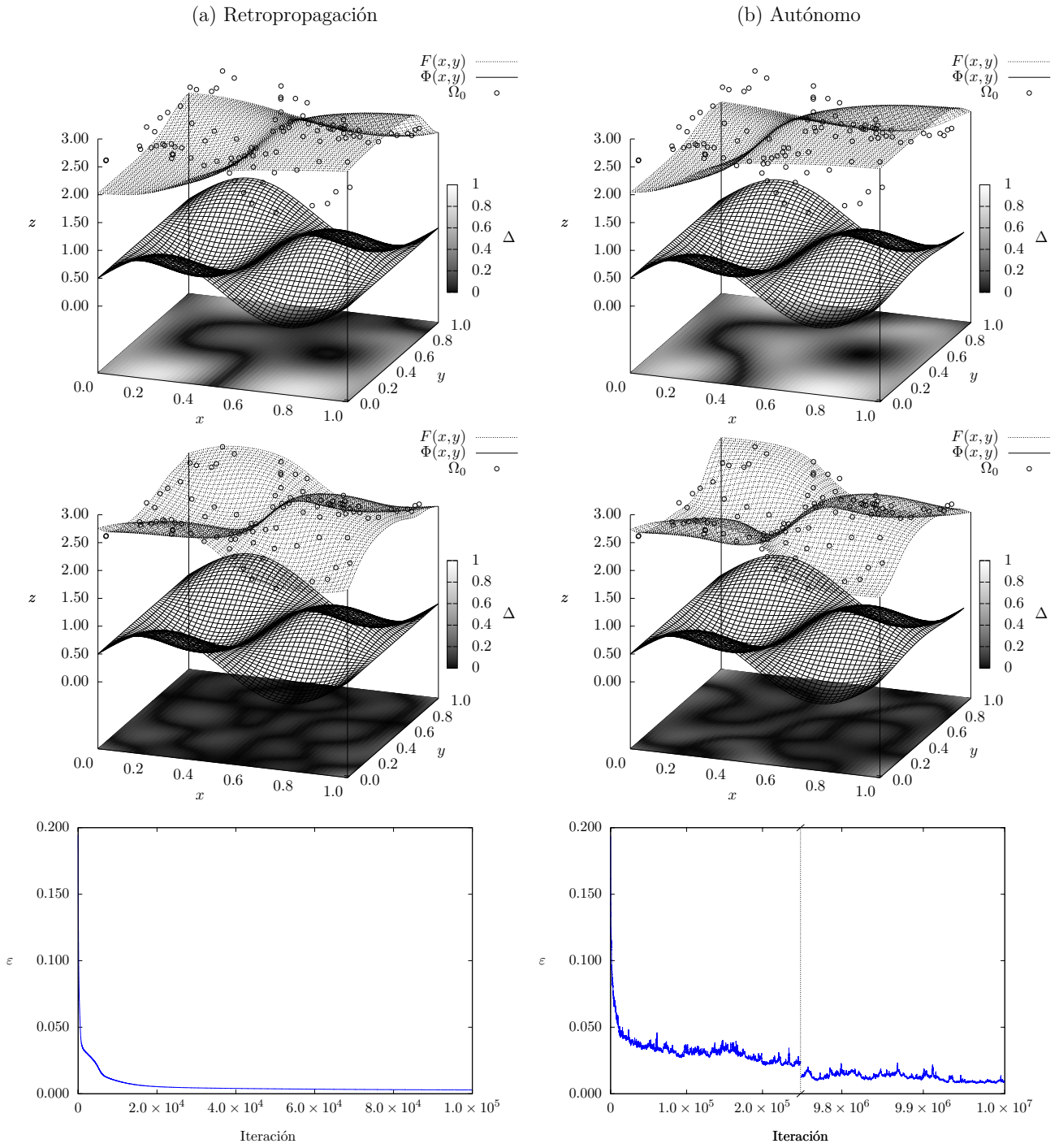

Figura 3.2. Aproximación de $\Phi(x,y) = 0.5 \times [1 + \cos(2\pi y)\sin(2\pi x)]$ con una red neuronal artificial usando el aprendizaje por retropropagación (columna izquierda) y el aprendizaje autónomo (columna derecha). Las dos primeras filas muestran la función objetivo Φ (malla de líneas sólidas), conjunto de aprendizaje (círculos blancos), función de red F (malla de líneas discontinuas, *z-offset* $= 2.00$), y la diferencia $\Delta \equiv |F - \Phi|$ en todo $[0,1]^2$ proyectada sobre el plano xy (mapeada en escala de grises); la primera fila corresponde a 100 iteraciones de los algoritmos, mientras que la segunda fila al resultado final. A su vez, en la última fila se grafica la evolución de ε durante el aprendizaje.

Capítulo 4

Estudio I: Funcionalidad

El primer estudio que realizamos tiene como objetivo evaluar el desempeño del aprendizaje autónomo (3.29) aplicado a una red neuronal artificial multicapa con el fin de entrenarla para cumplir determinada tarea, es decir, optimizarla para ser *funcional*. El escenario de aprendizaje que planteamos es del tipo *supervisado* y de clasificación, ya que entrenaremos a la red mediante pares *entrada-salida* bien definidos. Para ello, el parámetro que nos permite definir el comportamiento ideal de la red es el error de aprendizaje ε (3.1), el cual explicitamos nuevamente para ser claros:

$$\varepsilon = \eta \sum_{p=1}^{N} |F(\chi_p) - \Phi(\chi_p)|^2 \quad . \tag{4.1}$$

Aquí, cada χ_p es una señal de entrada procesada por la red, $F(\chi_p)$ es la respuesta correspondiente de la red, y $\Phi(\chi_p)$ es la respuesta *objetivo*, siendo el conjunto $\Omega_0 = \{(\chi_1, \Phi(\chi_1)), \ldots, (\chi_N, \Phi(\chi_N))\}$ el conjunto de aprendizaje utilizado para el entrenamiento. La constante de normalización con la que trabajeremos es $\eta = 1/Nm$, donde m es la cantidad de nodos de salida de la red[1].

Como el objetivo es $\varepsilon \to 0$, la desviación ϵ del comportamiento ideal de la red con respecto a dicho parámetro es

$$\epsilon = |\varepsilon - 0| = \varepsilon \quad , \tag{4.2}$$

y el aprendizaje autónomo toma entonces la forma

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = -\gamma(\varepsilon_n - \varepsilon_{n-1})(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}) + \varepsilon_n S \boldsymbol{\xi}_n \quad . \tag{4.3}$$

La ley de evolución autónoma (4.3) será la que guíe a la red hasta su óptima funcionalidad.

[1] Como se verá, $N = 5$ y $m = 5$, con lo cual $\eta = 1/25$.

Nuestro primer estudio se organiza de la siguiente forma. En la Secc. 4.1 exponemos el problema planteado a la red: el reconocimiento gráfico de las cinco vocales dadas en una matriz de píxeles. Presentamos la relación entre dicha matriz y los nodos de entrada de la red, así como el conjunto de aprendizaje que emplearemos para entrenarla. Luego de esto, en la Secc. 4.2 buscamos los valores óptimos de los parámetros γ y S para la aplicación del aprendizaje autónomo (4.3).

Finalmente, exponemos nuestros resultados en la Secc. 4.3. En primer lugar, evaluamos el aprendizaje de la red, es decir, su desempeño sobre el conjunto de aprendizaje. Posterior a esto, analizamos su rendimiento por fuera de este conjunto para evaluar su capacidad de *generalizar* lo aprendido, para lo cual definimos un conjunto de prueba compuesto por las señales originales de las vocales pero con defectos introducidos aleatoriamente.

4.1. El problema

La tarea asignada a la red es el reconocimiento gráfico de las cinco vocales (A, E, I, O, U) dadas sobre una matriz de 7×5 píxeles. La correspondencia entre la matriz de píxeles y la red es la siguiente (ver Figura 4.1):

La red se construye con 35 nodos de entrada y 5 de salida. Cada nodo de entrada está asociado a un píxel de la matriz, y reproduce analógicamente el brillo de dicho píxel en escala de grises (invertida), con una señal numérica en el intervalo $[0,1]$ (con *blanco* $\equiv 0$ y *negro* $\equiv 1$). Una señal de entrada $x \in [0,1]^{35}$ correspondiente a determinado patrón en la matriz de píxeles, es procesado por la red y produce cierta señal de salida $F(x) \in [0,1]^5$ (representada por el arreglo vertical 5×1 de píxeles en la figura). Los 5 nodos de salida funcionan como clasificadores, cada uno asociado a una de las vocales. En el caso de que la red esté perfectamente optimizada para el reconocimiento, el resultado de procesar el patrón de determinada vocal será que todos los nodos de salida emitan una señal nula con excepción del nodo asociado a dicha vocal, el cual emitirá una señal unitaria.

4.1.1. Conjunto de aprendizaje

En la Fig. 4.2 se muestra el conjunto de aprendizaje Ω_0, dado por:

$$\Omega_0 = \{(\mathbb{A},\Phi(\mathbb{A})),(\mathbb{E},\Phi(\mathbb{E})),(\mathbb{I},\Phi(\mathbb{I})),(\mathbb{O},\Phi(\mathbb{O})),(\mathbb{U},\Phi(\mathbb{U}))\} \quad . \tag{4.4}$$

Como mencionamos anteriormente, asociamos cada uno de los 5 nodos de salida a uno de los patrones, denotando $n_{\mathbb{X}}$ al nodo que corresponde al patrón $\mathbb{X}$.

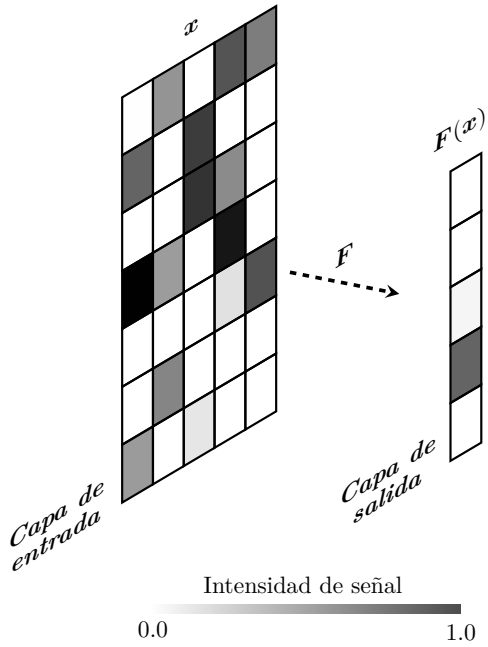

Figura 4.1. Ilustración representativa de la correspondencia entre la matriz 7×5 de píxeles y los 35 nodos de entrada. El brillo en escala de grises de cada píxel se transmite analógicamente (señal numérica entre 0 y 1) a un nodo de entrada, y comienza el procesamiento de la señal x por parte de la red. La señal de salida de la red $F(x) \in [0,1]^5$ se representa con el arreglo 5×1.

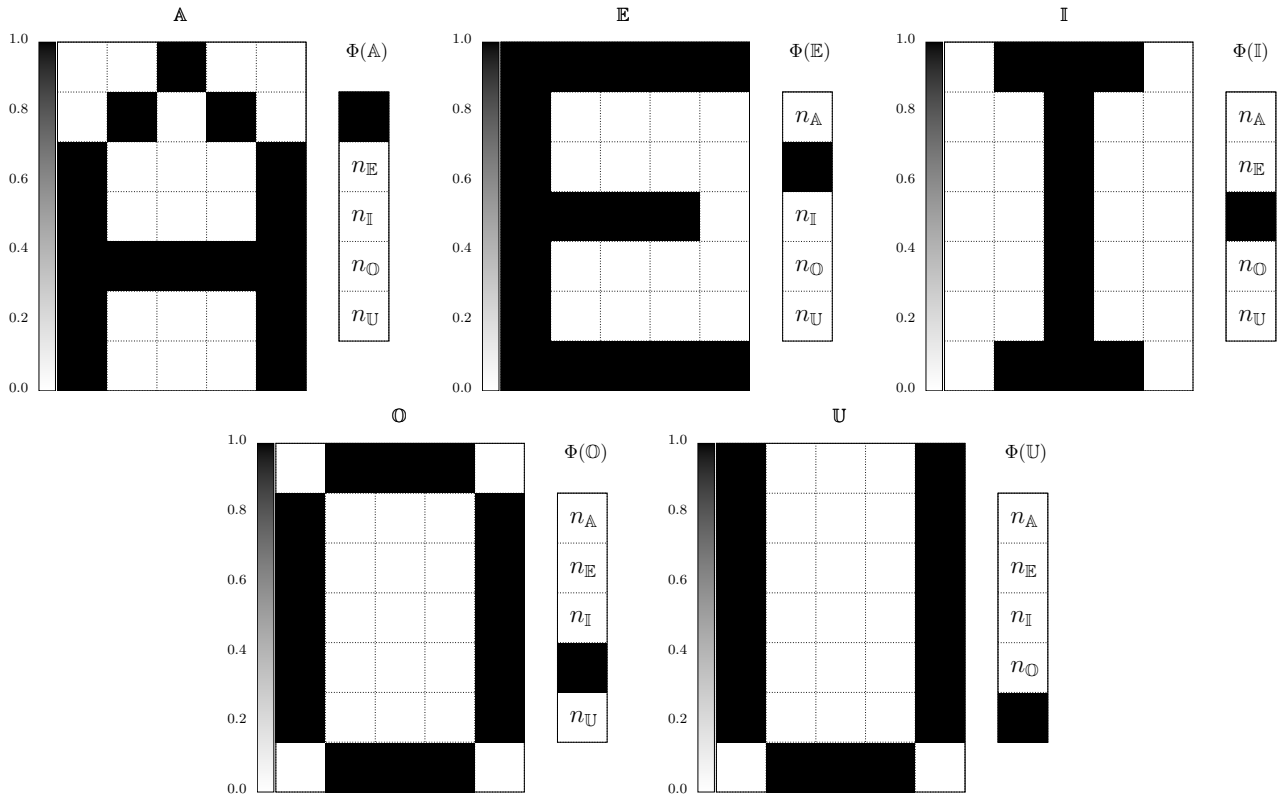

Figura 4.2. Elementos del conjunto de aprendizaje para el reconocimiento de las vocales. En cada par $(\chi, \Phi(\chi))$, cada píxel está asociado a un nodo de entrada (*izquierda*) o de salida (*derecha*) de la red. Cada uno de los 5 nodos de salida se asocia a una vocal: cuando a la red se le muestra el patrón $\mathbb{X}$, la señal de salida del nodo $n_{\mathbb{X}}$ debe ser 1, y la de los demás 0.

La escala de grises en la figura, a la izquierda de cada patrón, indica la intensidad de la señal asociada a cada unidad. Como se puede notar, las señales de entrada y de salida del conjunto de aprendizaje son, en el lenguaje de la red, secuencias de 0's y 1's. En el caso de las señales de entrada, los componentes de estas secuencias son las señales entrantes netas de los respectivos nodos de entrada. En el caso de los nodos de salida, las secuencias contienen la señal de salida de cada uno. Por ejemplo,para el par $(\mathbb{E},\Phi(\mathbb{E}))$, la señal de entrada es

$$(1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1) \quad , \tag{4.5}$$

y la señal de salida

$$(0,1,0,0,0) \quad . \tag{4.6}$$

Por supuesto, las señales de salida $F(\mathbb{X})$ de las redes optimizadas generalmente no serán estrictamente binarias. En la Figura 4.3 se ilustra un ejemplo, con una red optimizada ($\varepsilon = 0.019$) que procesa el elemento $\mathbb{E}$.

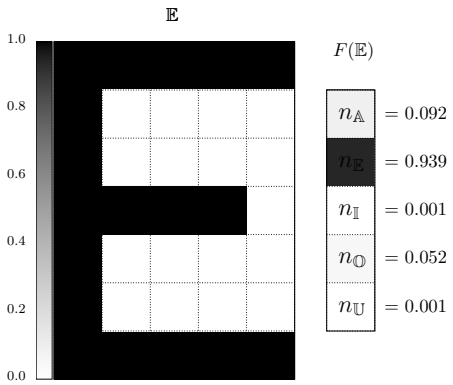

Figura 4.3. Ejemplo de respuesta de una red optimizada ($\varepsilon = 0.019$) que procesa el patrón $\mathbb{E}$.

4.1.2. La red

La red con la que trabajamos es del tipo *feed-forward*, con 35 nodos de entrada y 5 de salida ($F : [0,1]^{35} \to [0,1]^5$) y una capa oculta con 15 nodos de procesamiento, como se muestra en la Fig. 4.4. La función de transferencia de los nodos es la sigmoidea s dada por

$$s(x) = \frac{1}{1 + \exp(-x)} \quad . \tag{4.7}$$

Conectamos a todos los nodos un nodo de sesgo que emite una señal constante unitaria (ver Secc. 2.2.3). El peso de la conexión de este nodo con cualquier otro (equivalente al umbral de este último)

será un parámetro más a optimizar durante el aprendizaje, con la excepción de aquellos asociados a la capa de entrada (todos los umbrales son constantes e iguales a 0).

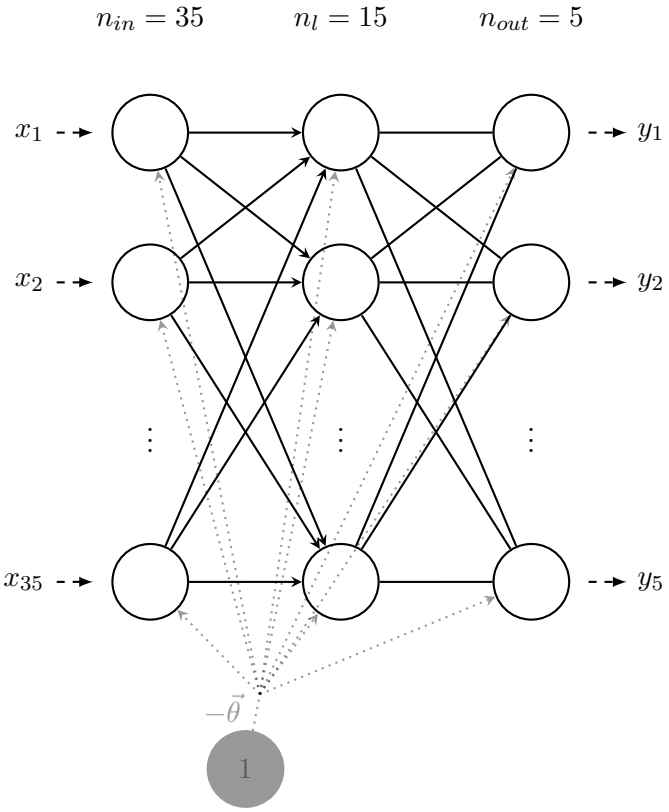

Figura 4.4. Propiedades de la red utilizada para el reconocimiento de las vocales. El nodo gris y sus conexiones proveen los umbrales $\vec{\theta}$ de la red (*cero* para los nodos de entrada). La capa de entrada recibe la señal de la matriz 7×5, y la capa de salida con 5 nodos señaliza a cuál de las cinco vocales corresponde la entrada. La capa oculta se encarga del procesamiento y provee los parámetros libres (los pesos de sus conexiones) a modificarse durante el aprendizaje.

4.2. Parámetros del aprendizaje

Para aplicar el aprendizaje autónomo (4.3) al entrenamiento de las redes, buscamos preliminarmente los valores óptimos de sus parámetros γ y S para el problema en cuestión. Para ello, elegimos un punto (γ', S') en el espacio $\gamma \times S$ y optimizamos 100 redes con 15×10^3 iteraciones del aprendizaje tomando $S = S'$ y $\gamma = \gamma'$. Luego, promediamos el error de las 100 redes, elegimos otro punto y repetimos el proceso para otros puntos en cierta región de $\gamma \times S$. La región en la que buscamos el mínimo de $< \varepsilon >$ es $\log_{10}\gamma \in [-3.00, 3.50] \wedge \log_{10} S \in [-3.00, 0.50]$, con apreciación $\delta \log_{10}\gamma = \delta \log_{10} S = 0.25$. La Figura 4.5 muestra el resultado de esta búsqueda. Una vez que encontramos el mínimo, reducimos la apreciación a $\delta \log_{10}\gamma = \delta \log_{10} S = 0.05$ y buscamos en una pequeña región en torno a él. En el Cuadro **??** se reúnen los resultados de las dos búsquedas.

$\delta \log_{10}$	$\log_{10} \gamma$	$\log_{10} S$	$< \varepsilon >_{min}$
$\pm$ 0.25	1.75	-0.75	0.081
$\pm$ 0.05	1.90	-0.90	0.079

Cuadro 4.1. Resultados de la búsqueda del mínimo de $< \varepsilon >$ en el espacio de los parámetros del aprendizaje.

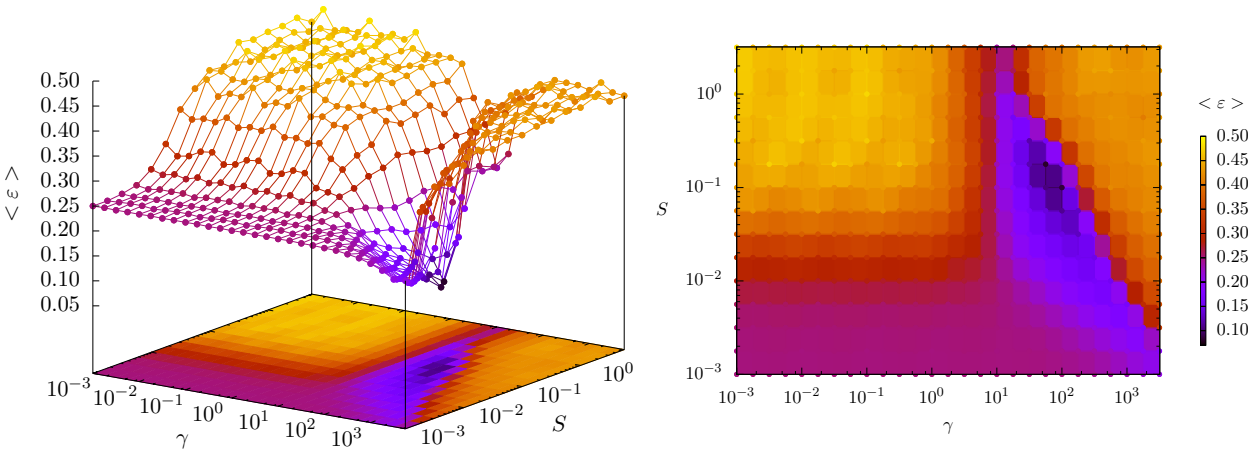

Figura 4.5. Búsqueda de los valores óptimos de γ y S para el aprendizaje ($\delta \log_{10} = 0.25$ en ambas direcciones).

4.3. Resultados

Armados con los parámetros óptimos del aprendizaje autónomo, optimizamos 100 redes con respecto ε en 2×10^5 iteraciones del aprendizaje, iniciando los pesos en cero. En la Figura 4.6 se muestra la distribución final de ε (*fila superior*), así como la evolución del error promedio $< \varepsilon >$ durante el aprendizaje (*fila inferior*). Además, comparamos los resultados con aquellos obtenidos aplicando el método de aprendizaje por retropropagación[2]. Como se puede ver, si bien la convergencia a cero es al menos *cuatro* órdenes de magnitud más rápida en el caso del aprendizaje por retropropagación, el aprendizaje autónomo logra minimizar el error de aprendizaje de las redes desde un valor inicial 0.250, a un valor medio final $< \varepsilon >= 0.035$.

En la Figura 4.7 mostramos la evolución de ε durante el aprendizaje autónomo de dos redes para dos casos representativos. En la primera de ellas (Fig. 4.7a) la red logra evolucionar satisfactoriamente minimizando el error ε. En la segunda, el error desciende abruptamente al comenzar el aprendizaje e inmediatamente se estanca en un valor elevado de ε (0.32, incluso mayor al error inicial 0.25). De las 100 redes que optimizamos, sólo hubo un caso como este. Para el resto de las redes, el aprendizaje se mantuvo activo hasta el final y el error logró disminuirse.

[2]En este caso, los pesos se inician con valores no nulos muy pequeños.

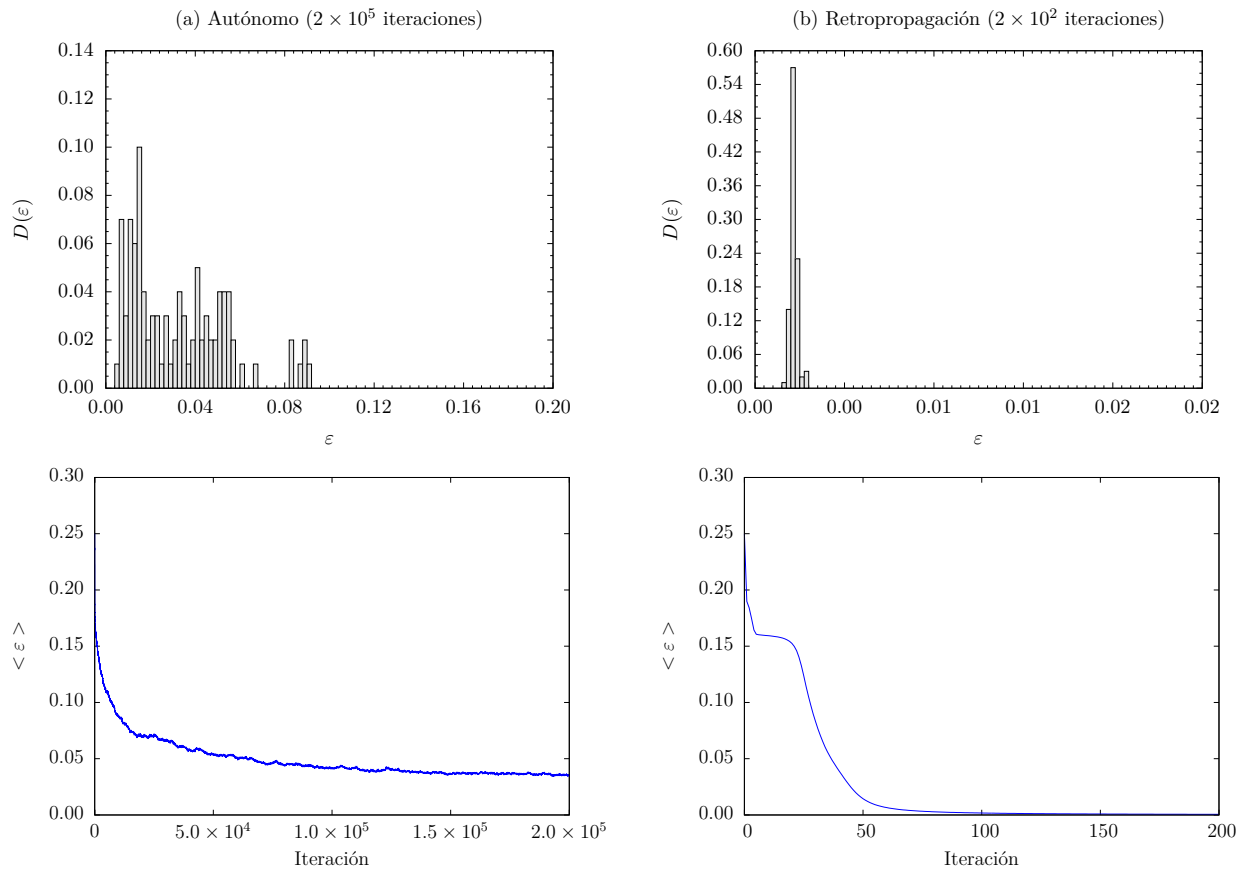

Figura 4.6. Distribución de ε sobre las 100 redes optimizadas mediante el aprendizaje autónomo (a) y por retropropagación (b). En la fila inferior se muestra la evolución promedio del error sobre el conjunto de 100 redes.

En lo que sigue, estudiamos estadísticamente el rendimiento de las 99 redes que evolucionaron exitosamente para saber si el aprendizaje logrado es suficiente para el reconocimiento de las vocales. Para ello, analizamos la funcionalidad de las redes optimizadas en dos etapas: *aprendizaje* (Sección 4.3.1) y *prueba* (Sección 4.3.2). En la primera, evaluamos qué tan bien las redes reconocen los elementos de entrada del conjunto de aprendizaje. En la segunda, evaluamos su desempeño sobre elementos ajenos a este conjunto para saber si son capaces de generalizar lo aprendido. En cada caso, realizamos dos tipos de análisis:

El primero de ellos consiste en evaluar directamente la intensidad de activación de cada nodo de salida $n_{\mathbb{X}}$ luego de procesar los patrones bajo interés.

En el segundo, digitalizamos las señales de salida de dichos nodos introduciendo determinado *umbral* de activación u para todos ellos, y analizamos la frecuencia de activación de cada $n_{\mathbb{X}}$ (tomada sobre las 99 redes). En particular, nos interesa saber si existen umbrales que podemos imponer de manera que dicha frecuencia sea máxima para el nodo correspondiente al patrón mostrado en cada caso. A estos umbrales los llamamos umbrales *óptimos*.

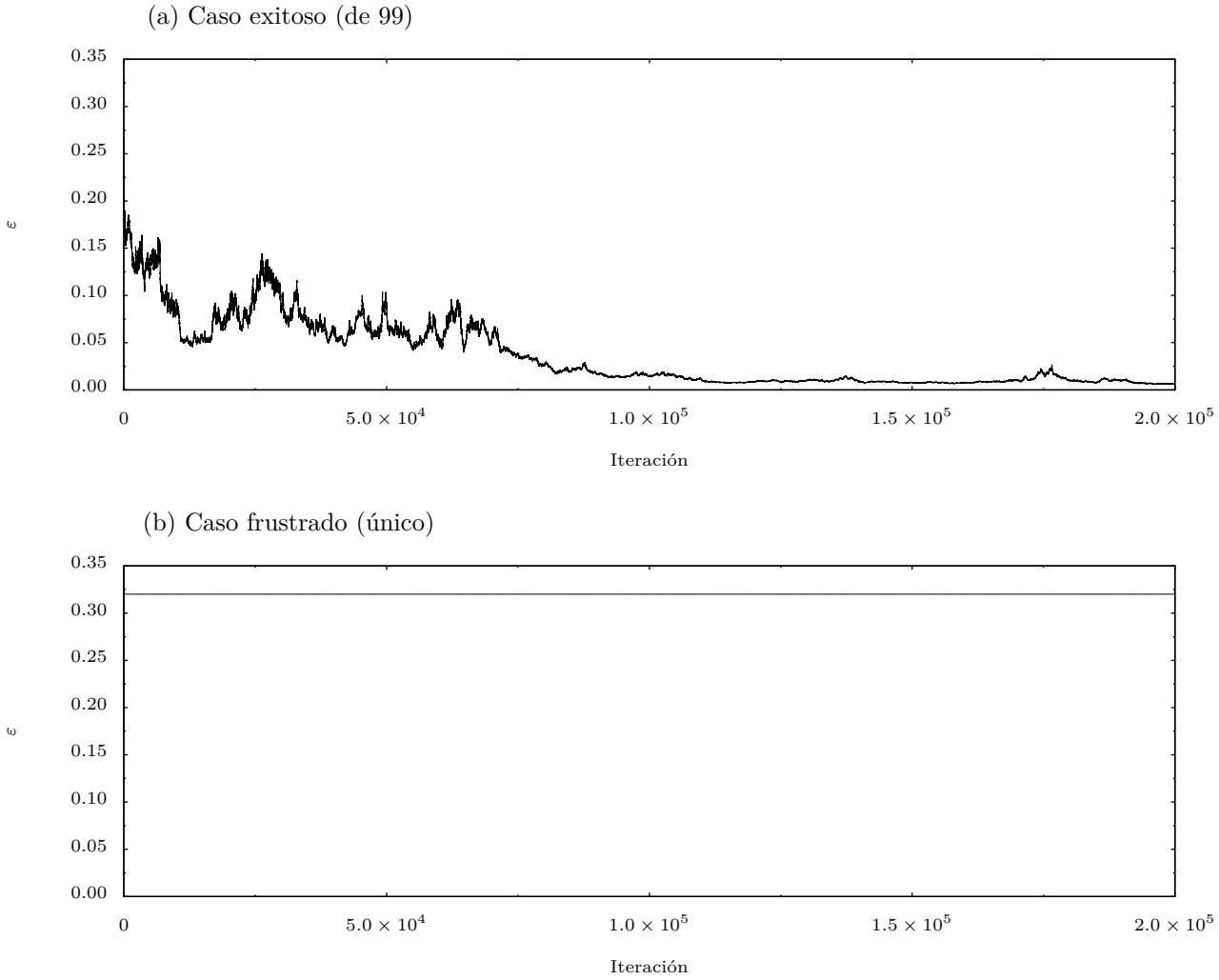

Figura 4.7. Dos ejemplos de la evolución de ε durante el aprendizaje autónomo. (a) Caso en el que el aprendizaje es exitoso. (b) El error de aprendizaje fluctúa violentamente poco después del inicio, salta hacia un valor elevado, y permance allí hasta el final.

4.3.1. Aprendizaje

Analizamos el desempeño de las 99 redes sobre las entradas del conjunto de aprendizaje: $\omega_0 = \{\mathbb{A},\mathbb{E},\mathbb{I},\mathbb{O},\mathbb{U}\}$.

4.3.1a. Intensidad promedio de activación

El primer análisis consiste en presentarle a las redes cierto patrón $\mathbb{X} \in \omega_0$, y promediar la respuesta de cada nodo de salida sobre las 99 redes. Esto lo hacemos en tres instancias durante el aprendizaje: luego de (a) 4×10^3, (b) 5×10^3, y (c) 2×10^5 iteraciones. En la Figura 4.8 se grafican los resultados.

Cada cuadro muestra la activación promedio de los nodos de salida luego del procesamiento de cierto patrón (indicado en la esquina superior derecha de cada uno). Se puede ver que, a medida que

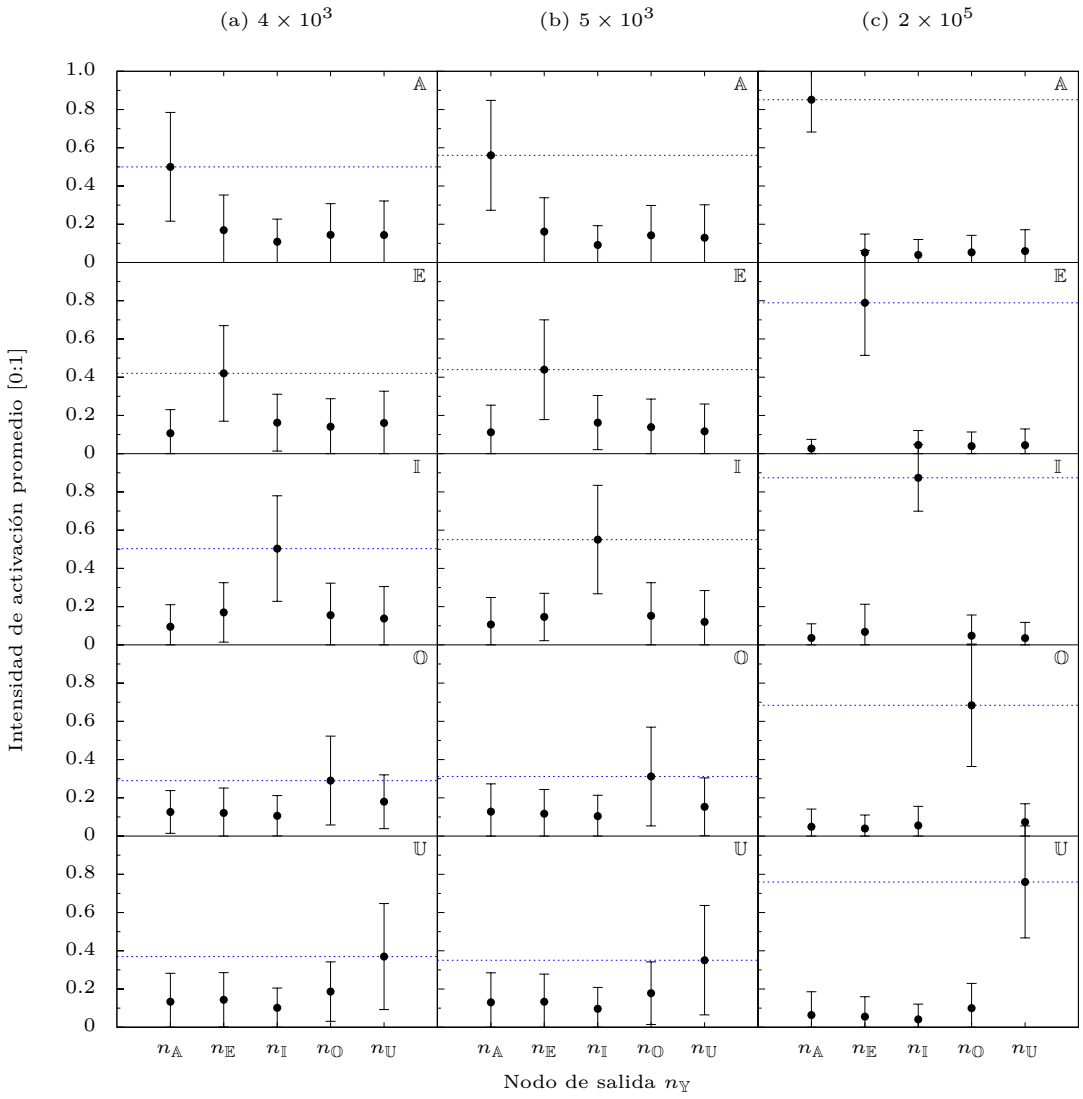

Figura 4.8. Intensidad de activación promedio de cada nodo de salida $n_{\mathbb{Y}}$ para las 99 redes optimizadas con specto a ε, luego de procesar cada entrada del conjunto de aprendizaje (indicado en la esquina superior derecha de cada ventana). La línea discontinua interseca el punto de máxima abcisa, siendo roja si el nodo en cuestión no coincide con la vocal mostrada y azul en el caso contrario.

progresa el aprendizaje, el reconocimiento mejora, aumentando la diferencia entre el nodo asociado al patrón y los demás nodos. La situación mejora notablemente al concluir con 2×10^5 iteraciones el aprendizaje.

4.3.1b. Búsqueda de umbrales óptimos de activación.

El segundo análisis consiste en buscar los umbrales óptimos para la digitalización de las señales de salida de los nodos $n_{\mathbb{Y}}$. El proceso es el siguiente:

a. Se fija un valor umbral u.

b. Se les presenta a las redes cierto patrón $\mathbb{X} \in \omega_0$.

c. Para cada nodo de salida $n_{\mathbb{Y}}$, se cuenta la cantidad de veces (de 99) que su respuesta a $\mathbb{X}$ superó o igualó el valor umbral u fijado (frecuencia de activación).

d. Finalmente, se evalúa para cuál de los nodos de salida dicha frecuencia es máxima y, si este nodo es el que corresponde al patrón mostrado, es decir, $n_{\mathbb{X}}$, el umbral se considera óptimo para $\mathbb{X}$.

e. Se repite esto para todo elemento de ω_0. Si u resulta óptimo para todos ellos, el umbral se considera *globalmente* óptimo.

Graficamos los resultados de este análisis construyendo una matriz $(u,\mathbb{X})$ (Figura 4.9), de la siguiente forma: Sea $F_{n_{\mathbb{Y}}}(u,\mathbb{X})$ la cantidad de redes para las cuales la intensidad de activación del nodo $n_{\mathbb{Y}}$ superó o igualó el umbral u. Si se tiene $F_{n_{\mathbb{X}}}(u,\mathbb{X}) = \text{máx}\left\{F_{n_{\mathbb{Y}}}(u,\mathbb{X})\right\}_{\mathbb{Y}\in\omega_0}$ y dicho $n_{\mathbb{X}}$ es el *único* nodo para el que se cumple esto, entonces se le asigna un 1 al lugar $(u,\mathbb{X})$ de la matriz (rectángulo gris en la figura). De lo contrario, se le asigna un 0 (rectángulo blanco). Nuevamente, hacemos esto para tres instancias durante el aprendizaje.

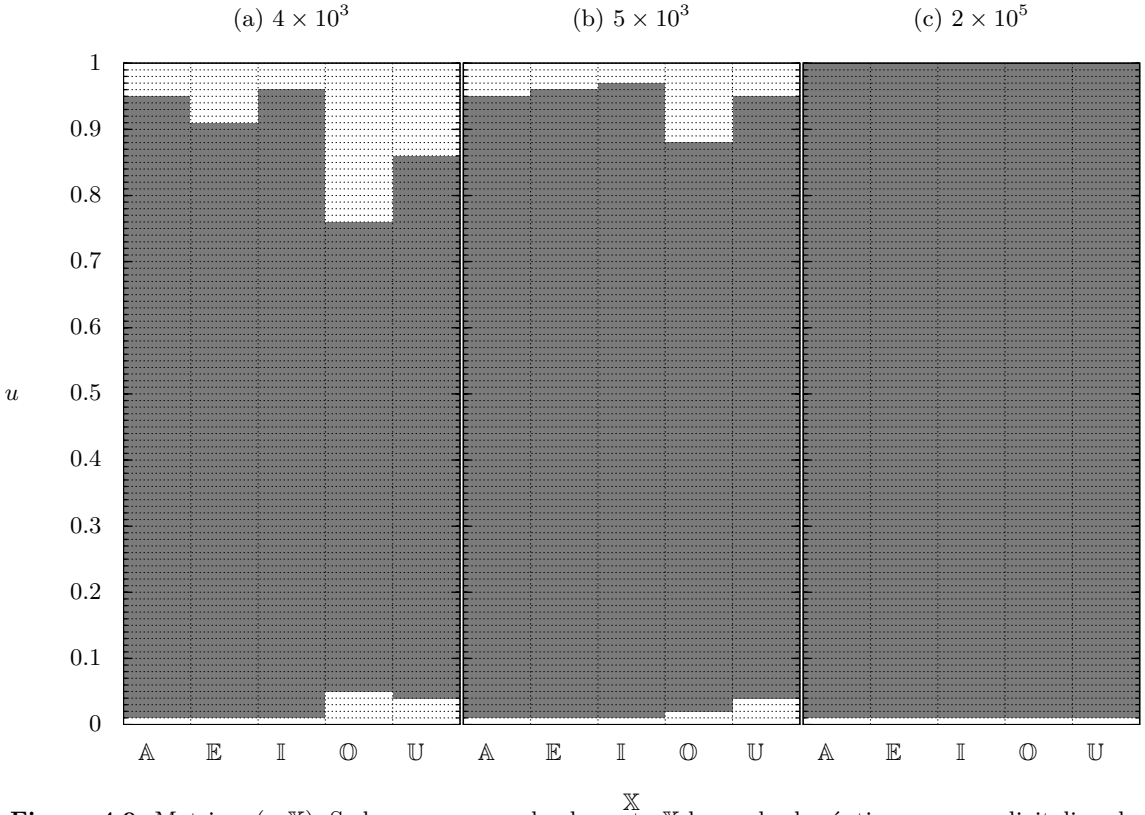

Figura 4.9. Matrices $(u,\mathbb{X})$. Se buscan para cada elemento $\mathbb{X}$ los umbrales óptimos u para digitalizar la señales de salida y saber si desde esa perspectiva las redes califican como funcionales.

En la Figura 4.9 se puede ver que la matriz $(u,\mathbb{X})$ se va llenando a medida que avanza el aprendizaje. En (a), por ejemplo, el umbral $u = 0.80$ es óptimo para todos los patrones menos el patrón $\mathbb{O}$. En (b) esto mismo sucede pero para $u = 0.90$. Finalmente, la matriz (c) indica que, como resultado del

aprendizaje, las redes admiten como umbrales óptimos a todo $u \in [0,1]$ ($\delta u = 0.01$), lo cual demuestra que, desde esta perspectiva, el aprendizaje fue absolutamente satisfactorio.

4.3.1c. Frecuencia media de activación con $u = 0.80$.

Para terminar, elegimos un umbral globalmente óptimo de la matriz (c), y graficamos la frecuencia media de activación $f_{n_\mathbb{Y}}(u,\mathbb{X}) = F_{n_\mathbb{Y}}(u,\mathbb{X})/99$ de cada nodo $n_\mathbb{Y}$. El valor que elegimos es $u = 0.80$. Los resultados se ilustran en la Figura 4.10.

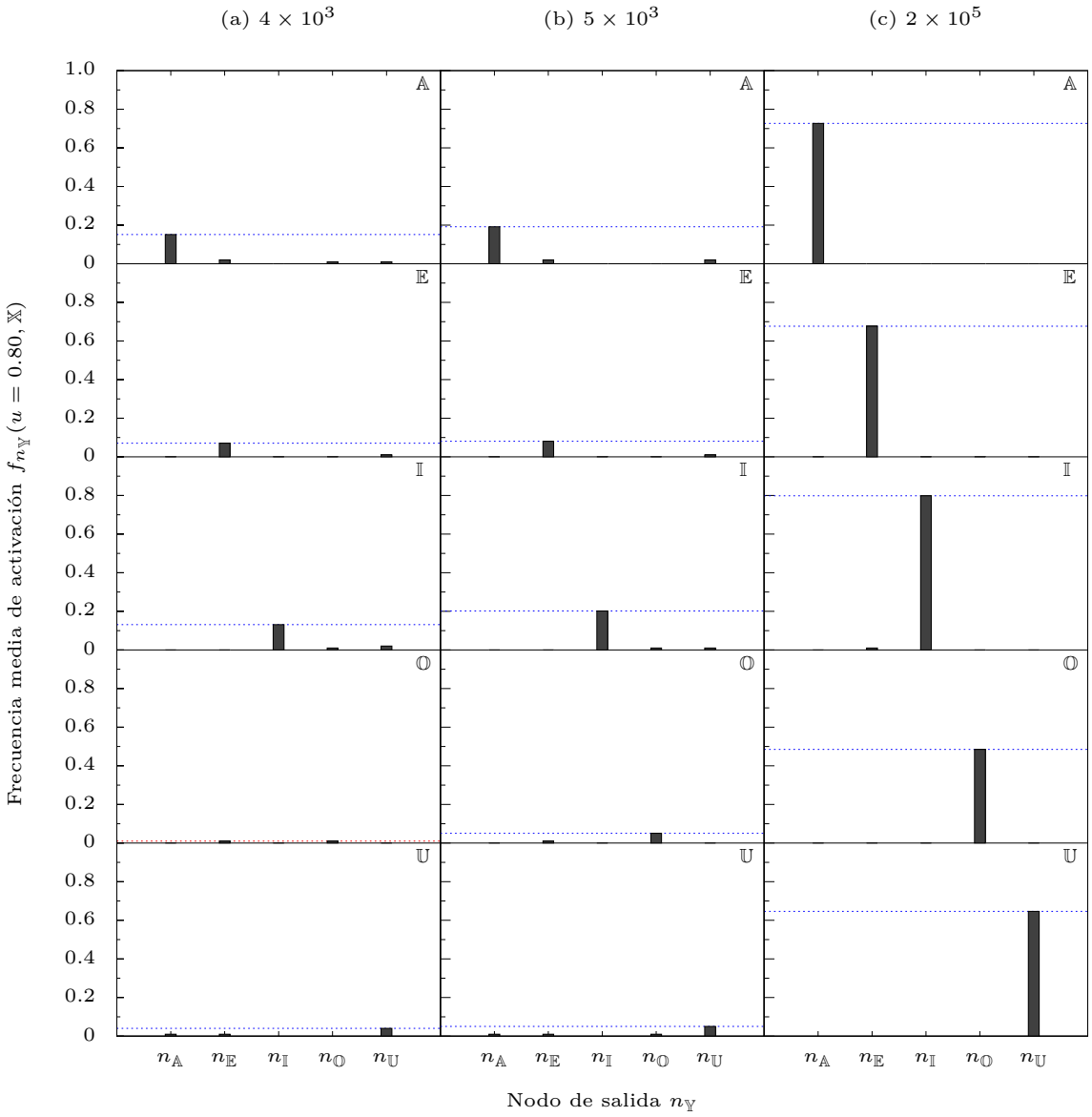

Figura 4.10. Frecuencia de activación promedio de cada nodo de salida, tomando $u = 0.80$ como umbral de activación.

Se puede notar la mejora de las respuestas de las redes a medida que avanza el aprendizaje. En 4.10a el patrón $\mathbb{O}$ aún no es reconocible por el conjunto de redes con el umbral utilizado 0.80, lo cual corresponde a lo que se ve en la matriz $(u,\mathbb{X})$ respectiva (Fig. 4.9a).

4.3.2. Prueba

Nos enfocamos ahora en la robustez *funcional* de las redes[3], es decir, su capacidad para reconocer elementos que no se usaron para entrenarla. El conjunto de prueba que usamos a tal fin, lo construimos invirtiendo aleatoriamente cierta cantidad de píxeles en los patrones originales de entrada. Precisamente, tomamos un elemento de entrada $\mathbb{X}$ de ω_0. Elegimos un lugar en la matriz de píxeles y, si el lugar contenía un 1 (0) lo hacemos 0 (1). Si hacemos esto $\jmath$ veces (por supuesto, siempre en distintos lugares), obtenemos un patrón $\mathbb{X}_\jmath$ con $\jmath$ píxeles invertidos con respecto al patrón original $\mathbb{X}$. A su vez, las salidas objetivo correspondientes a estos nuevos patrones son iguales a las de los patrones originales, es decir $\Phi(\mathbb{X}_\jmath) = \Phi(\mathbb{X})$. La Figura 4.11 (próxima página) muestra el conjunto de prueba que construimos. El mismo está dado por:

$$\Omega = \{(\mathbb{A}_\jmath, \Phi(\mathbb{A})), (\mathbb{E}_\jmath, \Phi(\mathbb{E})), (\mathbb{I}_\jmath, \Phi(\mathbb{I})), (\mathbb{O}_\jmath, \Phi(\mathbb{O})), (\mathbb{U}_\jmath, \Phi(\mathbb{U}))\}_{\jmath \leq 4} \tag{4.8}$$

Es decir, las entradas son todos aquellos patrones en los que se han invertido *hasta* 4 píxeles.

El análisis que se hace con Ω es análogo al de la sección anterior: estudiamos la intensidad media de activación de los nodos, la matriz $(u, X_\jmath)$, y la frecuencia media de activación para cierto umbral óptimo u.

[3]No debe confundirse con la robustez *estructural* a tratarse en el próximo capítulo.

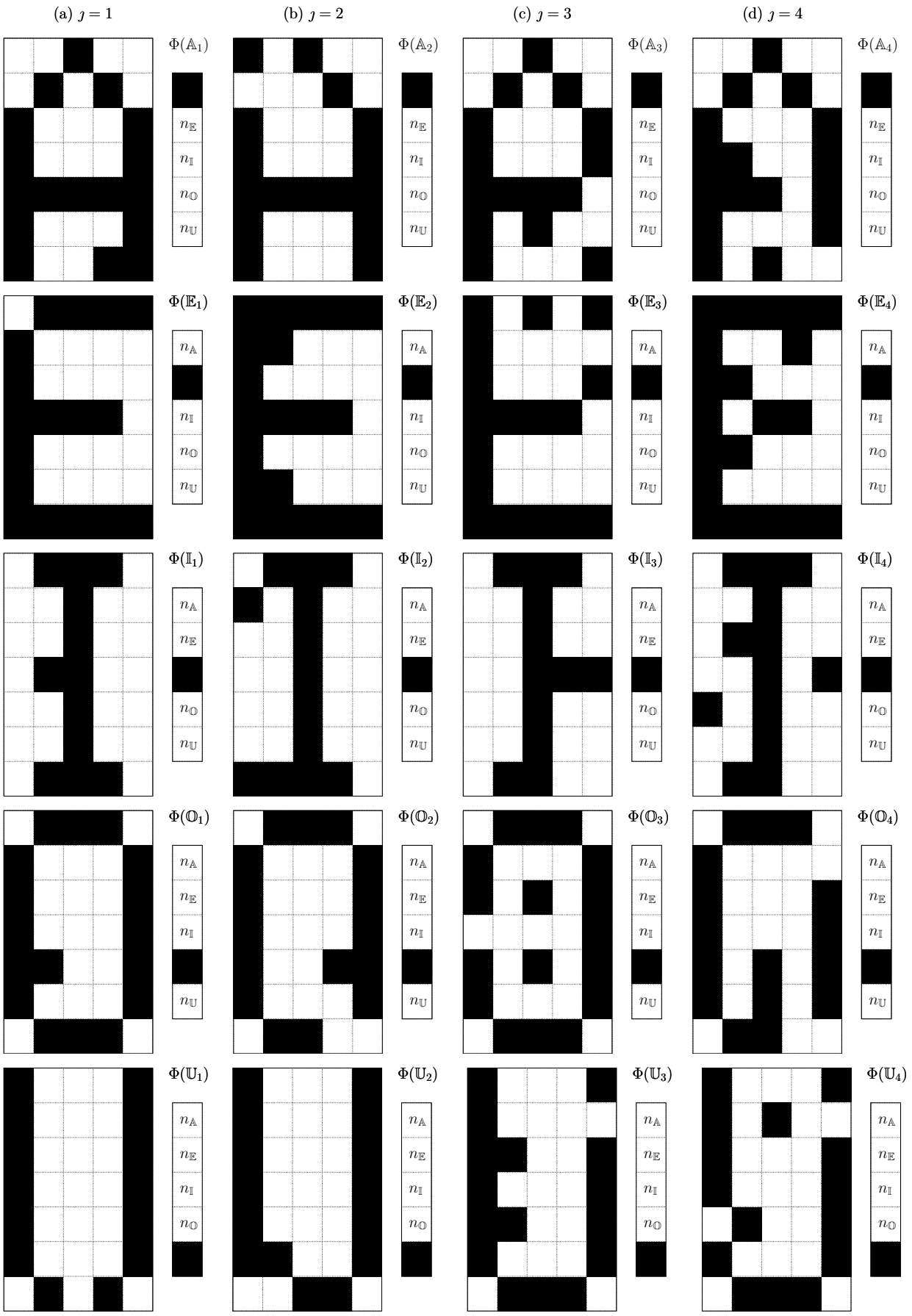

Figura 4.11. Conjunto de prueba. La entrada $\mathbb{X}_\jmath$ tiene $\jmath$ píxeles invertidos ($\mathbb{X}_\jmath = \mathbb{A}_\jmath, \mathbb{E}_\jmath, \mathbb{I}_\jmath, \mathbb{O}_\jmath, \mathbb{U}_\jmath$).

4.3.2a. Intensidad promedio de activación.

La Figura 4.12 muestra la activación promedio de cada nodo de salida luego de procesar los patrones defectuosos $\mathbb{X}_\jmath$. Si bien a medida que aumentan los defectos sobre los patrones originales, disminuye la activación máxima de los nodos correspondientes (lo cual es aceptable), en todos los casos el nodo de máxima activación corresponde al patrón original. Es decir, las redes son capaces de generalizar lo aprendido.

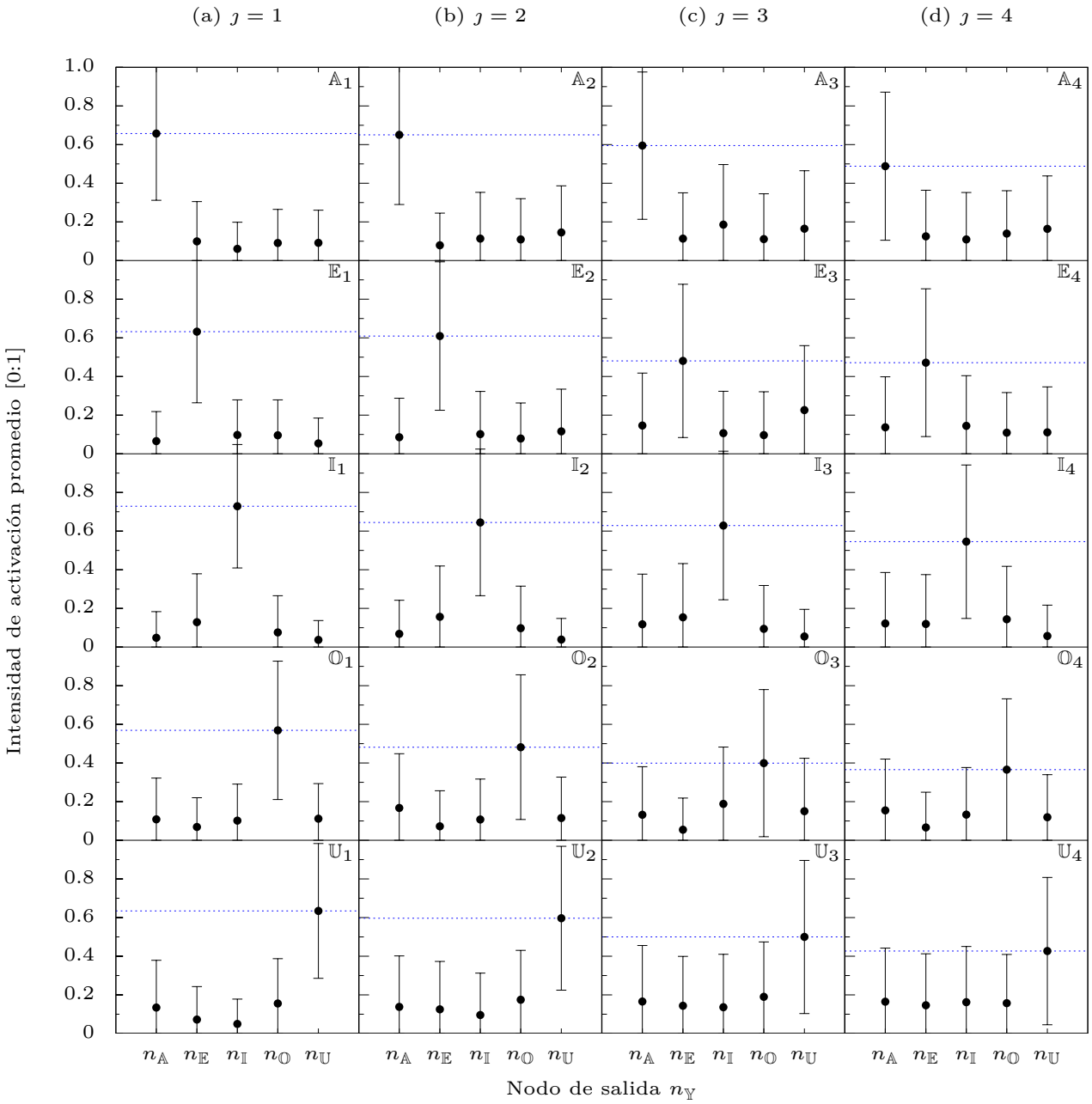

Figura 4.12. Intensidad de activación promedio de los nodos de salida, luego de que la red procesa los patrones defectuosos. En todos los casos el nodo de máxima activación corresponde al patrón mostrado a la red.

4.3.2b. Búsqueda de umbrales óptimos.

Las matrices (u, X_j) para $\jmath \leq 4$ se muestran en la Figura 4.13. En este análisis también se manifiesta el rendimiento satisfactorio de las redes, ya que para todo $\jmath \leq 4$, todos los umbrales resultan globalmente óptimos. En particular, todos los umbrales óptimos en el caso de los patrones originales (Fig. 4.9) son también óptimos para *todos* los patrones defectuosos.

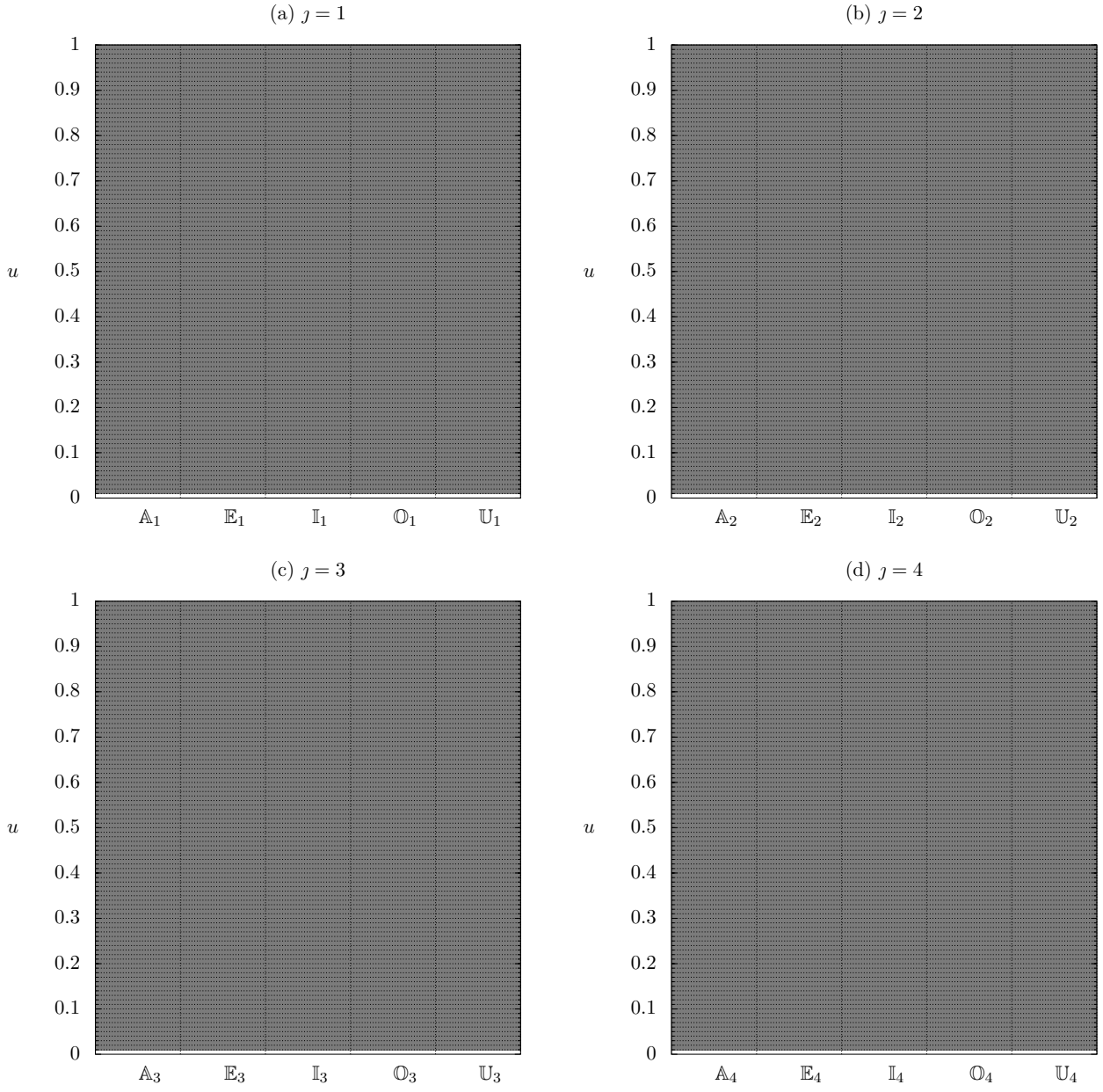

Figura 4.13. Matrices $(u, \mathbb{X})$.

4.3.2c. Frecuencia media de activación con $u = 0.80$.

Tomando $u = 0.80$ como en la sección anterior, se grafica en la Fig. 4.14 la frecuencia media de activación $f_{n_{\mathbb{Y}}}(u,\mathbb{X}_\jmath)$ de los nodos de salida. En acuerdo con los resultados que se observan en las matrices $(u,\mathbb{X}_\jmath)$, las redes presentan un excelente reconocimiento de todos los patrones defectuosos con el valor umbral 0.80.

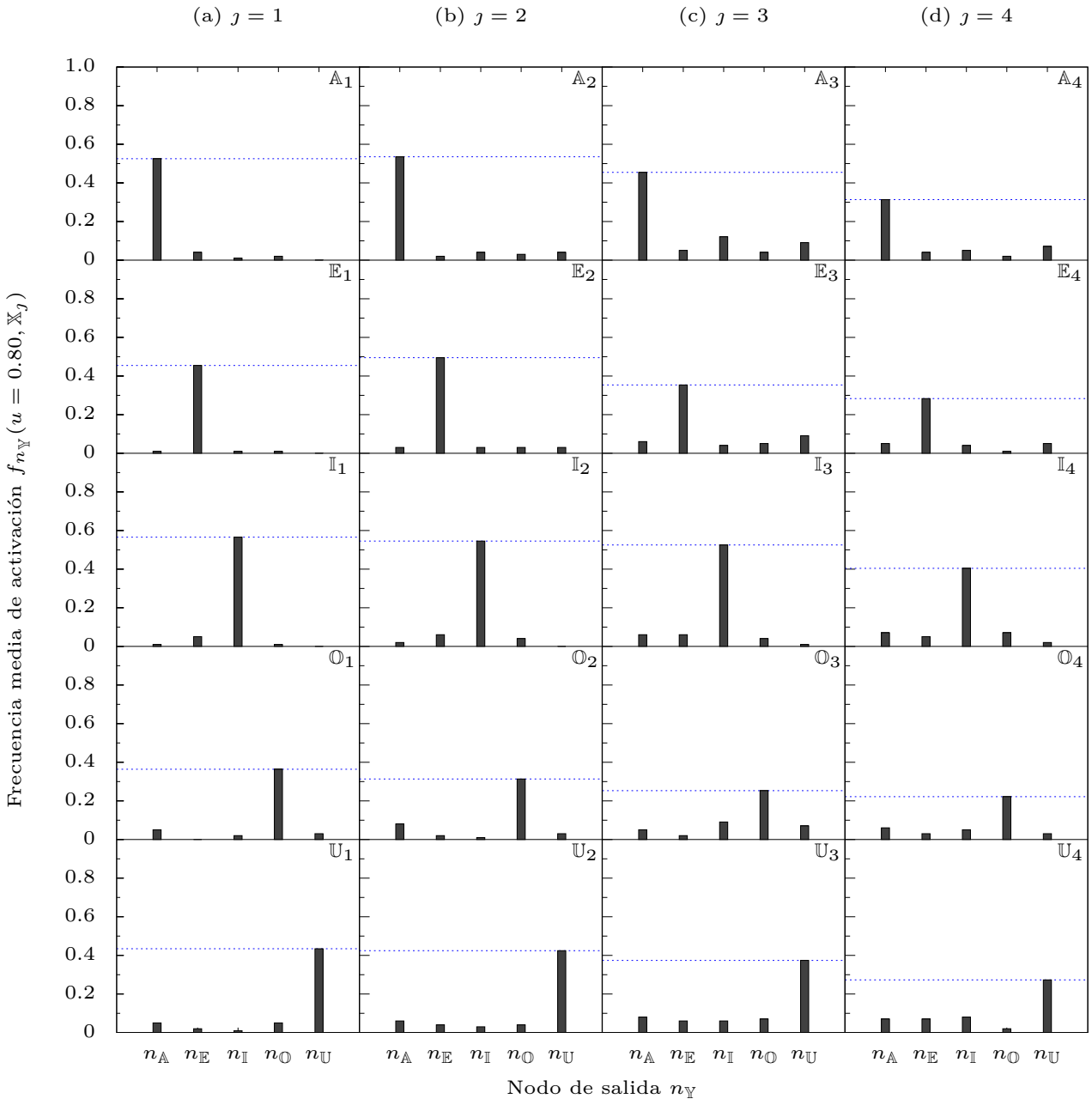

Figura 4.14. Frecuencia de activación promedio, tomando $u = 0.80$ como umbral de activación.

4.4. Conclusiones

Los resultados de nuestro primer estudio demuestran que el aprendizaje autónomo fue exitoso para entrenar las redes a reconocer los patrones de las vocales.

Si bien en comparación con el método de retropropagación la convergencia hacia los valores óptimos de los parámetros fue considerablemente más lenta, de las 100 redes que optimizamos con el aprendizaje autónomo, 99 de ellas evolucionaron favorablemente disminuyendo su error de aprendizaje, lo cual quedó demostrado por la evolución del error promedio $< \varepsilon >$ del conjunto, así como por la distribución final de ε sobre las redes optimizadas.

Para demostrar que el nivel de aprendizaje alcanzado fue bueno, analizamos estadísticamente el rendimiento de las redes al procesar elementos del conjunto de aprendizaje y elementos de un conjunto de prueba que construimos introduciendo defectos sobre los patrones originales, con el fin de saber hasta qué punto las redes eran capaces de *generalizar* lo aprendido. El análisis se realizó desde dos perspectivas, ambas de las cuales pusieron en evidencia que el aprendizaje fue más que satisfactorio. En la primera, analizamos directamente la intensidad de activación de las señales de salida al procesar los patrones bajo interés. En la segunda, buscamos umbrales para digitalizar dichas señales, que resultaran óptimos para los patrones a reconocer por la red, de manera que las frecuencias promedio de activación de los nodos fueran máximas para aquellos nodos correspondientes a los patrones mostrados. En ambos casos las redes supieron reconocer no sólo los patrones mostrados durante el aprendizaje, sino versiones defectuosas de estos últimos. Es decir, como resultado del aprendizaje autónomo, las redes resultaron exitosamente *funcionales* para el reconocimiento robusto de las vocales.

Capítulo 5

Estudio II: Robustez

La capacidad de un sistema para conservar su buen funcionamiento colectivo luego de sufrir perturbaciones, daños o fallas locales, es una propiedad común a la inmensa mayoría de los sistemas complejos naturales [36, 37]. Tanto el funcionamiento de una célula como la supervivencia de un organismo [38] dependen críticamente de dicha *robustez*, para regular su estabilidad funcional y así poder tolerar las perturbaciones inherentes a ambientes cuyas condiciones fluctuan incesantemente. Asimismo, sistemas de redes complejas creados por el humano, como las redes de comunicación (p. ej. el internet [39], la WWW [40]) y de transporte [41], deben ser lo suficientemente robustos como para evitar que el mal funcionamiento de sus partes por diversos motivos resulte en pérdidas sustanciales de la información o los bienes que fluyen por ellas [42].

En los sistemas biológicos, la resiliencia para soportar perturbaciones o fallas es producto de procesos evolutivos naturales, al mismo tiempo que dichos procesos son favorecidos por ella [37]. En el caso ideal, el diseño de sistemas complejos artificiales para fines tecnológicos o logísticos debe lograr imitar los altos grados de robustez de los sistemas biológicos, sin incurrir en la mera redundancia de sus elementos. La pregunta crucial es entonces la siguiente [43]: ¿es posible construir sistemas artificiales que, además de ser adecuados para cumplir cierta función, sean robustos para tolerar fallas? El diseño de modelos abstractos que pueden evolucionar para ser funcionales y robustos [43, 44, 45, 46] tiene como objetivo entender las propiedades generales que caracterizan a los sistemas dotados de robustez, como paso previo a la construcción de sistemas reales.

Nuestro segundo estudio se propone aplicar el aprendizaje autónomo como una forma sucinta y efectiva de lograr esto, para el caso de las redes neuronales artificiales en el contexto del problema de aprendizaje que estudiamos en el capítulo anterior (ver Secc. 4.1). El objetivo de este estudio, es

optimizar las redes para que, además de aprender a reconocer los patrones de las vocales, mantengan su buen funcionamiento luego de sufrir la eliminación de cualquiera de sus nodos intermedios.

Para ello, partimos definiendo la medida de robustez que aplicamos a las redes. Posteriormente, reformulamos el aprendizaje autónomo para que su aplicación admita el aprendizaje de las redes y la optimización de su robustez de forma *simultánea*. Esto introduce dos nuevos parámetros del aprendizaje asociados a la optimización de la robustez, cuyos valores óptimos buscamos de manera análoga a la búsqueda que realizamos en el capítulo anterior. Comparando las distribuciones resultantes con las de las redes optimizadas únicamente para ser funcionales, demostramos la efectividad del aprendizaje para construir redes robustas y funcionales. Finalmente, comparamos estructuralmente ambos conjuntos de redes para saber si existen diferencias entre aquellas que son robustas y aquellas que no lo son.

5.1. Definición

Comenzamos dando una definición precisa [44] para la robustez de una red con respecto a la eliminación de uno de sus nodos intermedios. Con *eliminar* cierto nodo, nos referimos a aquello que es equivalente a igualar a cero los pesos de todas las conexiones asociadas a él. Sea M la cantidad de nodos intermedios que posee cierta red. Sea Γ el conjunto de redes ζ que se obtienen a partir de la red original mediante la eliminación de *uno* de sus nodos intermedios (con lo cual $|\Gamma| = M$). La robustez ρ de la red respecto a la eliminación de uno de sus nodos intermedios queda definida por

$$\rho = \frac{1}{M} \sum_{\zeta \in \Gamma} H[h - \varepsilon(\zeta)] \quad , \tag{5.1}$$

siendo H la función escalón centrada en el origen y $\varepsilon(\zeta)$ el error de la red ζ. La constante h, es un umbral por encima del cual consideramos el funcionamiento de las redes como *no* satisfactorio. Precisamente, si el error ε de una red se encuentra por encima de este umbral, su funcionalidad se considera inaceptable.

Se puede notar que $\rho : 0 \leq \rho \leq 1$ y que su valor óptimo es la unidad, es decir, cuando la eliminación de cualquiera de los nodos intermedios de la red en cuestión no es suficiente para que su error aumente por encima de h.

Para fijar el valor de h consideramos lo siguiente [43]. Tomamos las 100 redes optimizadas autónomamente con respecto a ε (ver Secc. 4.3) y eliminamos uno de los nodos intermedios de cada una. Haciendo esto para todos los nodos intermedios ($M = 15$), obtenemos un conjunto de 1500 redes dañadas (en la Fig. 5.1 se muestran las distribuciones de ε sobre las redes sanas y las dañadas, ambas

normalizadas a 1). El valor de h es aquel para el cual el 50 % de las redes dañadas tiene un error menor a h (línea discontinua en la Fig. 5.1b). En nuestro caso, $h = 0.060$.

Con respecto a dicho valor, es importante destacar lo siguiente. Considerando únicamente a las 99 redes cuyo aprendizaje fue exitoso, el error medio de las redes sanas es $\mu_\varepsilon = 0.032$, y la desviación estándar correspondiente $\sigma_\varepsilon = 0.021$, teniéndose así

$$h = \mu_\varepsilon + 1.33 \cdot \sigma_\varepsilon \quad . \tag{5.2}$$

Es decir, el valor de h se encuentra a menos de dos desviaciones estándar del error medio de las 99 redes originales que califican como funcionales. Teniendo en cuenta los resultados del capítulo anterior, en los cuales se demostró que el funcionamiento de dichas redes fue estadísticamente más que satisfactorio, podemos aseverar entonces que 0.060 es un buen valor umbral para usar en (5.1).

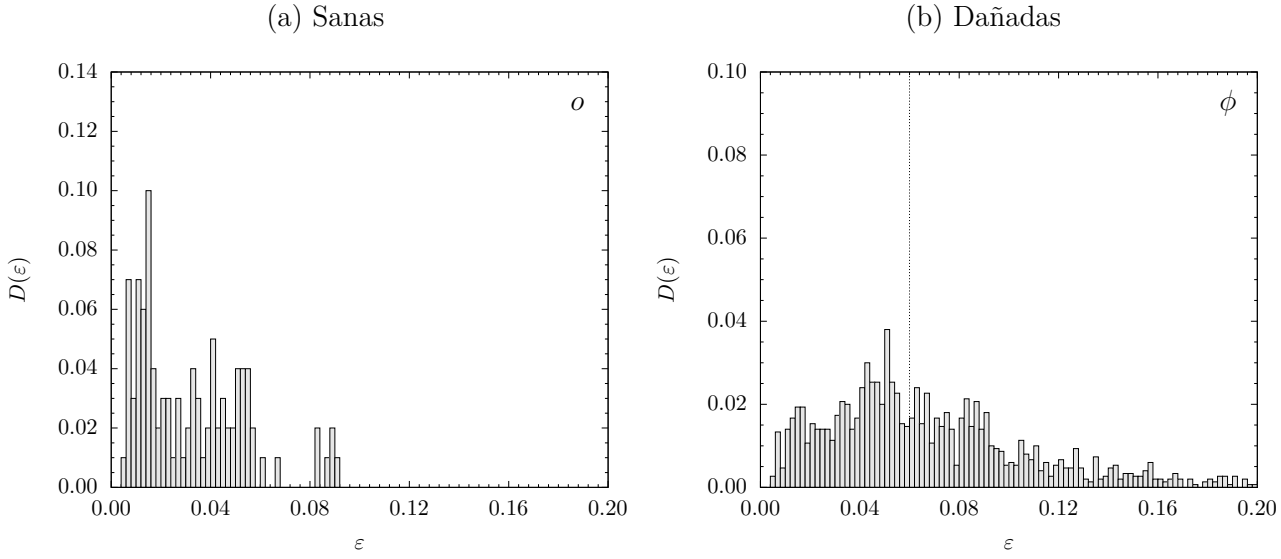

Figura 5.1. Distribución de ε de las redes optimizadas para ser solamente funcionales. El gráfico en (a) corresponde a las redes sanas y en (b) al conjunto de redes que se obtiene elimiando un nodo intermedio en las redes sanas.

5.2. Aprendizaje autónomo multiparamétrico

Presentamos la formulación multiparamétrica del aprendizaje autónomo, la cual permitirá entrenar a las redes para el reconocimiento de las vocales (minimizando ε) y al mismo tiempo optimizarlas con respecto a la robustez ρ para que conserven su funcionalidad luego de la eliminación de un nodo intermedio. Las desviaciones de ϵ_ε y ϵ_ρ del comportamiento ideal con respecto a ε y ρ, están dadas

por

$$\epsilon_\varepsilon = \varepsilon \tag{5.3}$$

$$\epsilon_\rho = 1 - \rho \quad . \tag{5.4}$$

Con esto, el aprendizaje autónomo para la optimización simultánea de ε y ρ se formula de la siguiente manera:

$$\Delta_{n+1}\boldsymbol{w} = -[\gamma_\varepsilon \Delta_n \varepsilon + \gamma_\rho \Delta_n (1-\rho)]\Delta_n \boldsymbol{w} + \varepsilon_n S_\varepsilon \boldsymbol{\xi}_n + (1-\rho_n) S_\rho \boldsymbol{\xi'}_n \quad , \tag{5.5}$$

donde se ha introducido la notación Δ_k : $\Delta_k z = z_k - z_{k-1}$ ($z = w$, ε ó ρ). La idea detrás de la optimización de ε y ρ en (5.5) es la misma que en la formulación original del aprendizaje 3.3.2, salvo que ahora hay *dos* tipos de desviaciones que conducen la evolución de $\boldsymbol{w}$: ϵ_ε y ϵ_ρ. En concomitancia con esto, los parámetros que regulan las modificaciones respectivas ahora son cuatro: (γ_ε,S_ε) y (γ_ρ,S_ρ).

Nuestra hipótesis es que, con los valores adecuados de los parámetros γ_ε, S_ε, γ_ρ y S_ρ, esta nueva forma del aprendizaje puede guiar la evolución de los pesos de la red no sólo para que ésta resulte funcional ($\varepsilon \to 0$) sino que además resulte estructuralmente robusta con respecto a la eliminación de un nodo intermedio ($\rho \to 1$).

5.3. Parámetros óptimos del aprendizaje

Nuestro primer objetivo es encontrar los parámetros óptimos para aplicar el aprendizaje (5.5) a las redes. Habiendo encontrado los valores óptimos de γ_ε y S_ε para el caso de la optimización exclusivamente con respecto a ε (Secc. 4.2), adoptamos dichos valores y buscamos la combinación óptima (γ_ε,S_ε,γ_ρ,S_ρ) explorando únicamente en $\gamma_\rho \times S_\rho$. La motivación de esto es que la robustez ρ de las redes está, por definición, supeditada al error ε y, de cierto modo, el punto de partida para su optimización debe contar con valores de ε lo suficientemente aceptables. Por lo tanto, buscamos γ_ρ y S_ρ con los valores de γ_ε y S_ε que ya sabemos que nos garantizan esto.

Como en el caso anterior, optimizamos 100 redes mediante (5.5) durante 15×10^3 iteraciones y promediamos sobre ellas el valor de $1-\rho$. La región que exploramos es $\log_{10}\gamma_\rho \in [-1.00, 1.50] \times \log_{10} S_\rho \in [-5.00, -0.75]$, con apreciación $\delta \log_{10} = 0.25$ en ambas direcciones. En la Figura 5.2 ilustramos los resultados de esta primera búsqueda. Nuevamente, una vez que encontramos el mínimo, reducimos la apreciación a $\delta \log_{10} = 0.05$ y buscamos en una pequeña región en torno a él. En el Cuadro 5.1 reunimos los resultados.

$\log_{10}\gamma_\varepsilon$	$\log_{10} S_\varepsilon$	$\delta \log_{10}$	$\log_{10}\gamma_\rho$	$\log_{10} S_\rho$	$< 1-\rho >_{min}$
1.90	-0.90	$\pm$ 0.25	0.75	-1.75	0.370
1.90	-0.90	$\pm$ 0.05	0.65	-1.65	0.322

Cuadro 5.1. Resultados de la búsqueda del mínimo de $< 1-\rho >$ en el espacio $\gamma_\rho \times S_\rho$. Los valores de γ_ε y γ_ε son los óptimos para la optimización con respceto a ε (ver Cuadro **??**).

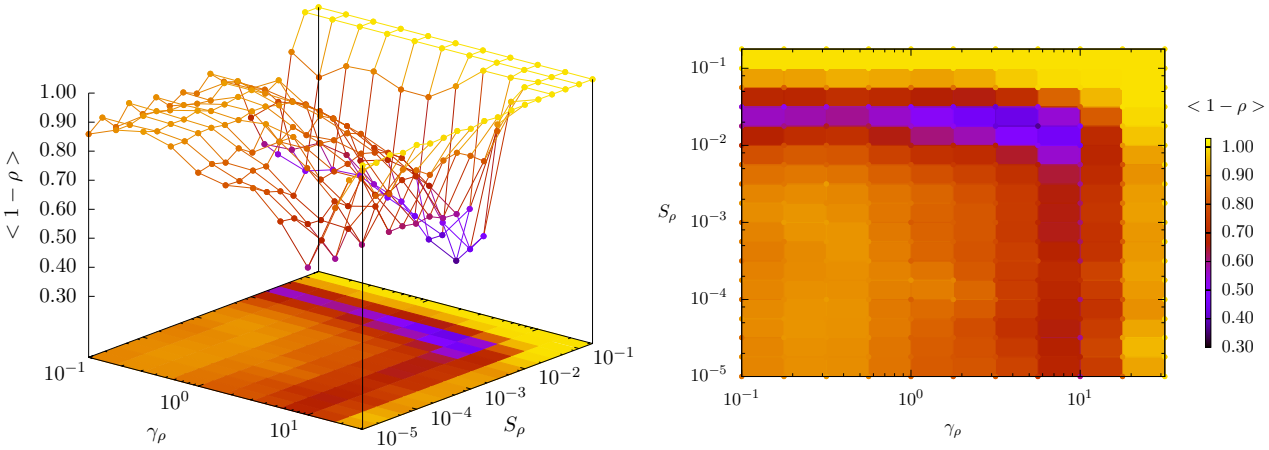

Figura 5.2. Búsqueda de los valores óptimos de los parámetros γ_ρ y S_ρ del aprendizaje autónomo para la optimización de ε y ρ.

Si bien con nuestra búsqueda encontramos mínimos bien definidos en $\gamma_\varepsilon = 10^{0.65}$ y $S_\varepsilon = 10^{-1.65}$, los valores que tomaremos para nuestro estudio no coincidirán puntualmente con ellos, sino que tomaremos otros valores cercanos a este punto (aún en la región *azul* de la Fig. 5.2): $\gamma_\varepsilon = 10^{0.90}$ y $S_\varepsilon = 10^{-2.00}$. La razón de esto es que con estos últimos valores encontramos que el porcentaje de evoluciones *frustradas* (ver Fig. 5.3) fue un 10 % menor que usando los otros valores de los parámetros. La conclusión que se extrae de este hecho es que, si bien una búsqueda inicial sistemática de los parámetros del aprendizaje es en cualquier caso necesaria, la sensibilidad del aprendizaje con respecto a sus parámetros es en última instancia desconocida y no queda descartada la posibilidad de obtener resultados más favorables con otros valores dentro de la región de los mínimos encontrados.

5.4. Resultados

Aplicamos el aprendizaje multiparamétrico (5.5) a 100 redes durante 2×10^5 iteraciones. En la Figura 5.3a se grafica la evolución de $< \varepsilon >$ y $< \rho >$ durante el aprendizaje (eje *izquierdo* y *derecho*, respectivamente). En el caso del error de aprendizaje (*curva negra*), el valor medio inicial de las redes es 0.250, y al finalizar el algoritmo su valor es 0.035, es decir, logra reducirse por un factor mayor que 7. A su vez, la robustez (*curva azul*) comienza con un valor nulo y termina en 0.823, lo cual es en gran medida satisfactorio. El Cuadro 5.2 reúne estos resultados.

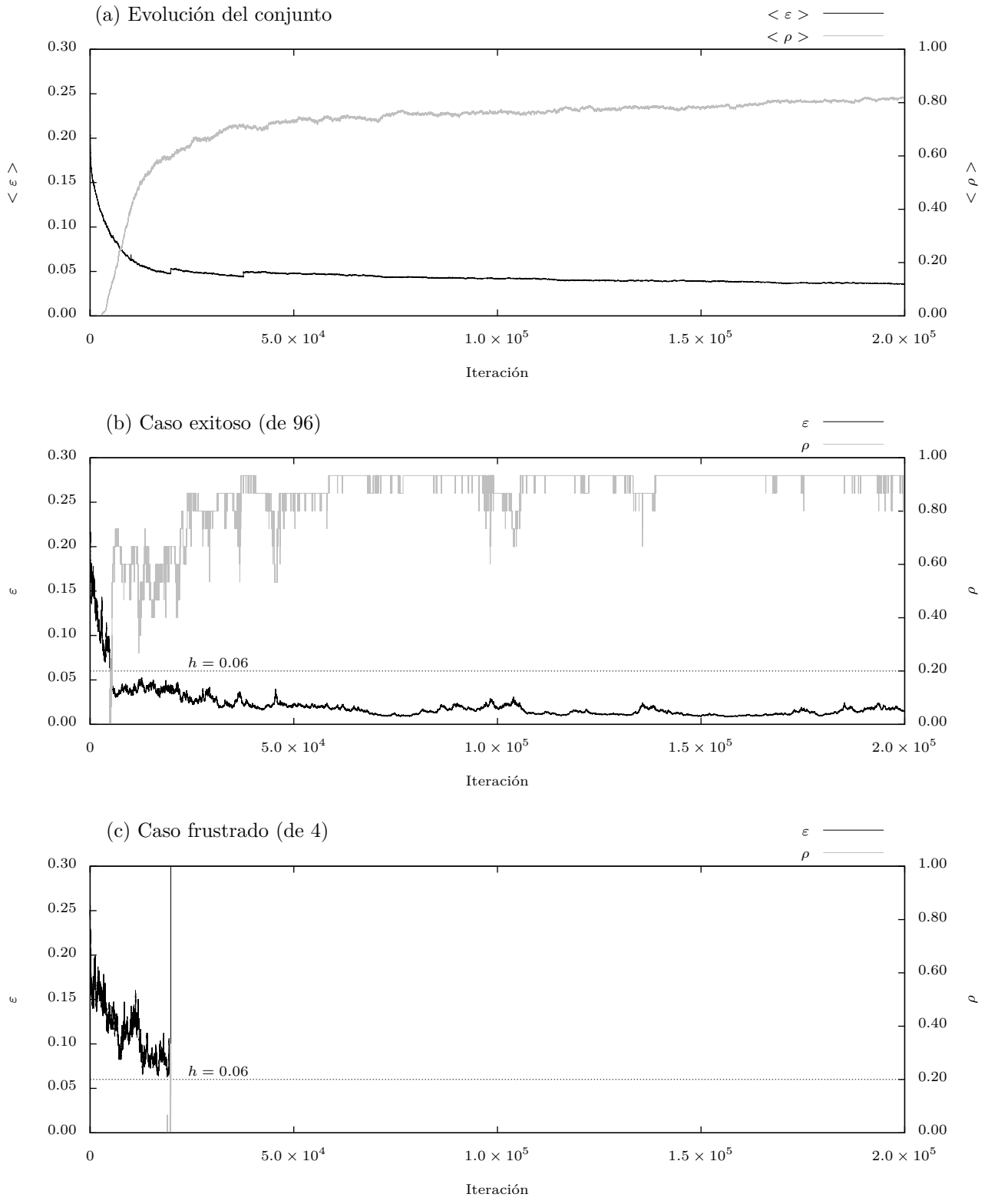

Figura 5.3. (a) Evolución de $< \varepsilon >$ y $< \rho >$ para el ensamble de 100 redes durante el aprendizaje. (b) Ejemplo de evolución exitosa para una de las 100 redes (de 96 casos similares). (c) Ejemplo de aprendizaje frustrado (de 4).

Las Figuras 5.3b y 5.3c ejemplifican la evolución de dos redes para dos casos representativos: una evolución en la cual se logra la optimización exitosa de ε y ρ, y otra en la cual el valor de ε se dispara

a un valor elevado y se estanca allí, lo cual a su vez impide la optimización de ρ y el aprendizaje queda completamente frustrado. De las 100 redes a las cuales aplicamos el aprendizaje, hubo 96 casos exitosos y 4 frustrados.

En la próxima sección, analizamos la distribución de ε sobre estas redes luego de eliminar uno de sus nodos intermedios y la comparamos con la correspondiente a las redes únicamente funcionales que construimos en el capítulo anterior, con el fin de evaluar qué tan bueno es el grado de robustez alcanzado. Concluímos el capítulo comparando las distribuciones de los pesos de las conexiones para saber si existe diferencia entre las redes robustas y las redes solamente funcionales, desde el punto de vista estructural.

Iteración	$< \varepsilon >$	$dev(\varepsilon)$	$< \rho >$	$dev(\rho)$
0	0.250	0.000	0.000	0.000
2×10^5	0.036	0.089	0.823	0.202

Cuadro 5.2. Resultados del aprendizaje autónomo multiparamétrico para la optimización simultánea de ε y ρ. Notar el elevado valor de $dev(\varepsilon)$ debido a los 4 casos abortivos del aprendizaje.

5.4.1. Distribuciones

Analizamos la distribución de ε y ρ para los dos conjuntos que hemos construído hasta ahora: (a) las redes optimizadas con respecto a ε y ρ (el presente capítulo); (b) las redes optimizadas únicamente con respecto a ε (el capítulo anterior). En las Figuras 5.4a y 5.4b (respectivamente) se ilustran estas distribuciones.

La primera y segunda fila muestran para ambos casos las distribuciones de ε y ρ para las redes sanas (o), es decir, las redes originales intactas[1]. Si bien las distribuciones de ε en (a) y (b) son ambas comparablemente satisfactorias, la diferencia es notable en lo que respecta a la robustez, pudiéndose observar un aumento considerable en la población de redes robustas ($\rho \geq 0.80$) cuando las redes se optimizaron también con respecto a ρ.

La diferencia se hace más evidente aún, considerando para ambos casos la distribución de ε sobre las 1500 redes dañadas (ϕ) que se obtienen eliminando uno de los 15 nodos intermedios en cada una de las 100 redes originales. Cuando las redes que *no* se han optimizado con respecto a ρ se dañan, la fracción de redes dañadas por debajo del umbral h a redes sanas por debajo de h es 0.542, es decir, casi el 46 % de las redes que eran funcionales (con criterio h) deja de serlo. En el caso de las redes que

[1]El ancho de muestreo para los histogramas es de 0.002 para ε y 0.02 para ρ.

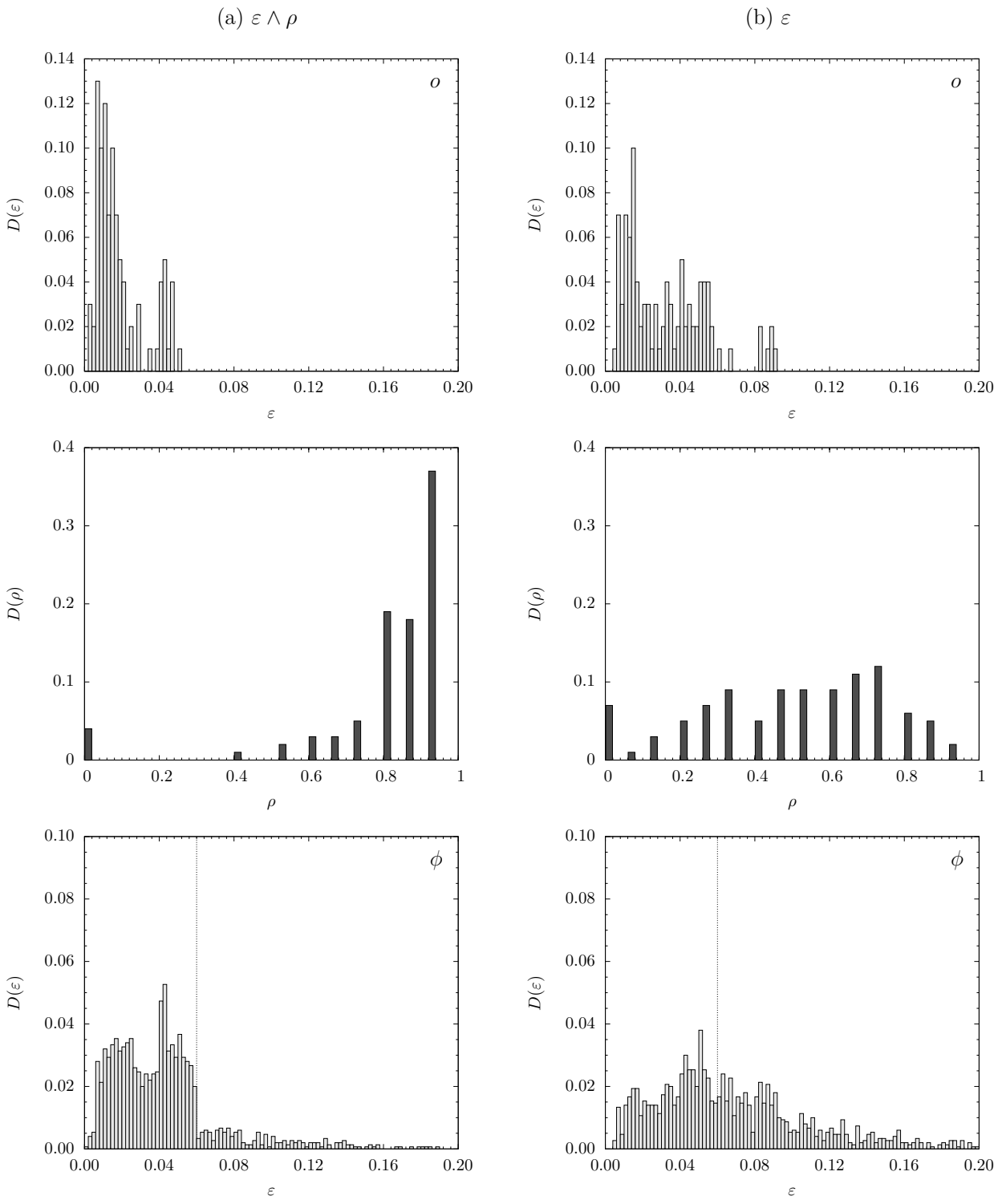

Figura 5.4. Distribución de ε (redes sanas o y dañadas ϕ mediante la eliminación de uno de sus nodos intermedios) y la robustez ρ. (a) 100 redes optimizadas con el aprendizaje autónomo multiparamétrico con respecto a ε y ρ. (b) 100 redes optimizadas autónomamente con respecto a ε (Estudio I del capítulo anterior). En las dos últimas filas es donde se advierte pronunciadamente la diferencia entre ambos conjuntos de redes, quedando demostrado que el aprendizaje autónomo multiparamétrico fue éxitoso en la construcción de redes *robustas* además de funcionales.

sí fueron optimizadas con respecto a ρ, esta fracción aumenta considerablemente a 0.857, con lo cual los daños sólo afectan a poco menos del 15 % de las redes funcionales. Esta diferencia se manifiesta claramente en los gráficos (a)ϕ y (b)ϕ, en los cuales se puede notar el mayor nivel de población en la

ventana $\varepsilon < h$ para el primer caso.

En el Cuadro 5.3 se compara entre ambos conjuntos de redes el efecto de eliminar un nodo intermedio. La magnitud $\nu^{(h)}$ es el cociente entre las frecuencias acumuladas $F(h)$ del error ε para las distribuciones ϕ y o, es decir:

$$\nu(h) = \frac{F_\phi(h)}{F_o(h)} \tag{5.6}$$

	$aut(\varepsilon \wedge \rho)$	$aut(\varepsilon)$
$F_o(h)$	0.960	0.920
$F_\phi(h)$	0.823	0.499
$\nu(h)$	0.857	0.542

Cuadro 5.3. Comparación del efecto de la eliminación de un nodo intermedio entre las redes funcionales ($aut(\varepsilon)$) y las redes robustas y funcionales ($aut(\varepsilon \wedge \rho)$).

5.4.2. Diferencias estructurales

Habiendo demostrado que el aprendizaje autónomo fue efectivo para optimizar las redes con respecto a su robustez, nos preguntamos si existen diferencias a nivel estructural entre las redes que son robustas y aquellas que son únicamente funcionales[2].

Empezamos considerando la distribución de los pesos de todas las conexiones de todas las redes, para cada conjunto de 96 redes optimizadas exitosamente. La Figura 5.5 ilustra estas distribuciones, donde se han normalizado los histogramas a 1. Se puede ver que, si bien la diferencia no es muy grande, los pesos de las redes robustas (Fig. 5.5a) están más concentradas en torno al origen. Podemos intentar entender la razón de esto con algunas consideraciones.

Calculamos en primer lugar la cantidad de conexiones ℓ^* que al ser eliminadas no cambian el error ε de su red, es decir, las conexiones inútiles. En el Cuadro 5.4 mostramos la fracción promedio de conexiones inútiles sobre conexiones totales $\ell_{tot} = 620$. Se puede ver que prácticamente no hay diferencia entre las redes robustas y las redes sólo funcionales, siendo en promedio el 10 % de las conexiones totales las que tienen efecto *nulo* sobre el error de la red.

El segundo análisis que realizamos resulta un poco más revelador, y se basa en generalizar esto último de la siguiente forma. Queremos saber cuál es la cantidad ℓ de conexiones que, al eliminarse,

[2]Consideramos sólo aquellas redes de las 100 cuya evolución no se frustró. En el caso de las redes optimizadas con respecto a ε y ρ estas son 96 de 100. En el caso de las redes optimizadas solamente con respecto a ε, son 99 de 100. Para simplificar el análisis comparativo, trabajamos sólo con 96 de estas 99.

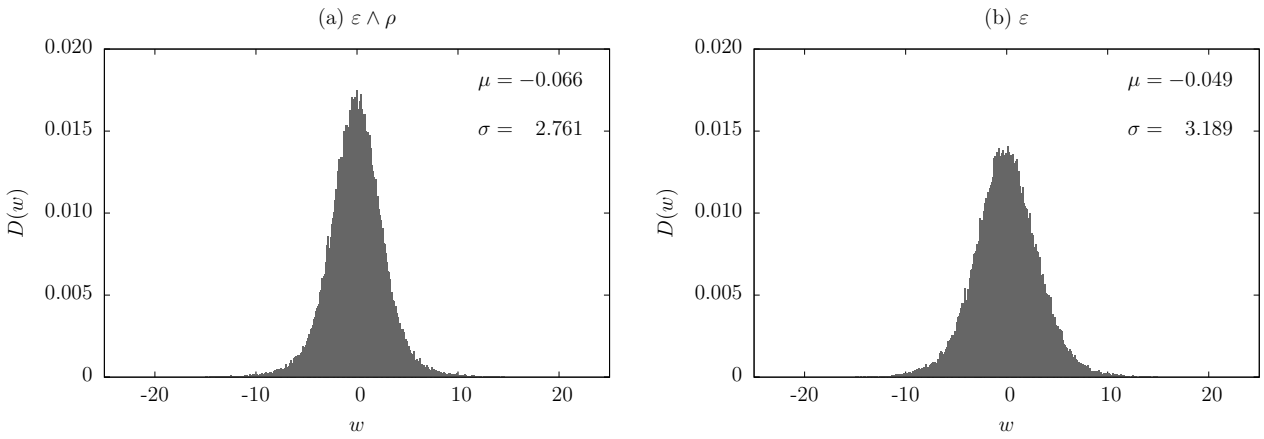

Figura 5.5. Distribución de los pesos w del conjunto de 96 redes que fueron optimizadas exitosamente con respecto a ε y ρ (a) y de 96 de las 99 que fueron exitosamente optimizadas con respecto a ε (b), para el estudio del capítulo anterior. Se puede notar que la distribución en (a) es más aguda en torno al origen.

	$aut(\varepsilon \wedge \rho)$	$aut(\varepsilon)$
$< \ell^*/\ell_{tot} >$	0.100	0.101
$dev < \ell^*/\ell_{tot} >$	0.012	0.011

Cuadro 5.4. Fracción promedio de conexiones inútiles a conexiones totales, en el caso de las redes funcionales ($aut(\varepsilon)$) y las redes robustas y funcionales ($aut(\varepsilon \wedge \rho)$).

producen un cambio *mayor o igual* que $\delta\varepsilon$ sobre el error ε. En la Figura 5.6 se muestra el resultado (la curva *azul* corresponde a las redes robustas y la *roja* a las redes sólo funcionales). Calculamos el promedio de $< \ell/\ell_{tot} >$ sobre todas las redes, para cada valor fijado de $\delta\varepsilon$. Hacemos esto para $\delta\varepsilon \in [0.00, 0.10]$ con incremento 1.00×10^{-4}.

Teniendo en mente que para ambos conjuntos de redes el error promedio $< \varepsilon >$ es de orden ~ 0.01 (0.018 y 0.032, para $< \varepsilon >_{\varepsilon\wedge\rho}$ y $< \varepsilon >_{\varepsilon}$, respectivamente), notamos lo siguiente acerca de los resultados de la Fig. 5.6. Para valores relativamente altos de $\delta\varepsilon$, el promedio de conexiones relevantes es mayor en el caso de las redes sólo funcionales. Esta diferencia (curva *negra* en el gráfico inferior) tiende asintóticamente a cero a medida que aumenta $\delta\varepsilon$, y es más notable en la región intermedia $(0.005, 0.01)$, donde $\delta\varepsilon$ aún es comparable con el error promedio de las redes, siendo máxima en 0.008. Para $\delta\varepsilon$ tendiendo a cero, la diferencia desciende abruptamente y se vuelve prácticamente nula en el origen. Como ya se notó, la fracción promedio de conexiones inútiles ($\delta\varepsilon = 0$) difiere muy poco entre los dos conjuntos ($\Delta \sim 0.001$). Sin embargo, el hecho de que la diferencia decaiga abruptamente cerca del origen significa que la fracción promedio de conexiones que al eliminarse provocan cambios *iguales* a $\delta\varepsilon$ en intervalos pequeños de esta región, es mayor para el caso de las redes robustas.

El análisis de la Fig. 5.6 revela que la fracción promedio de conexiones por red que causan incrementos considerables $\delta\varepsilon$ en el error, es mayor para las redes que no han sido optimizadas con respecto a la robustez y, a su vez, para $\delta\varepsilon \sim 0$ ($\neq 0$), la fracción de conexiones correspondientes es mayor para

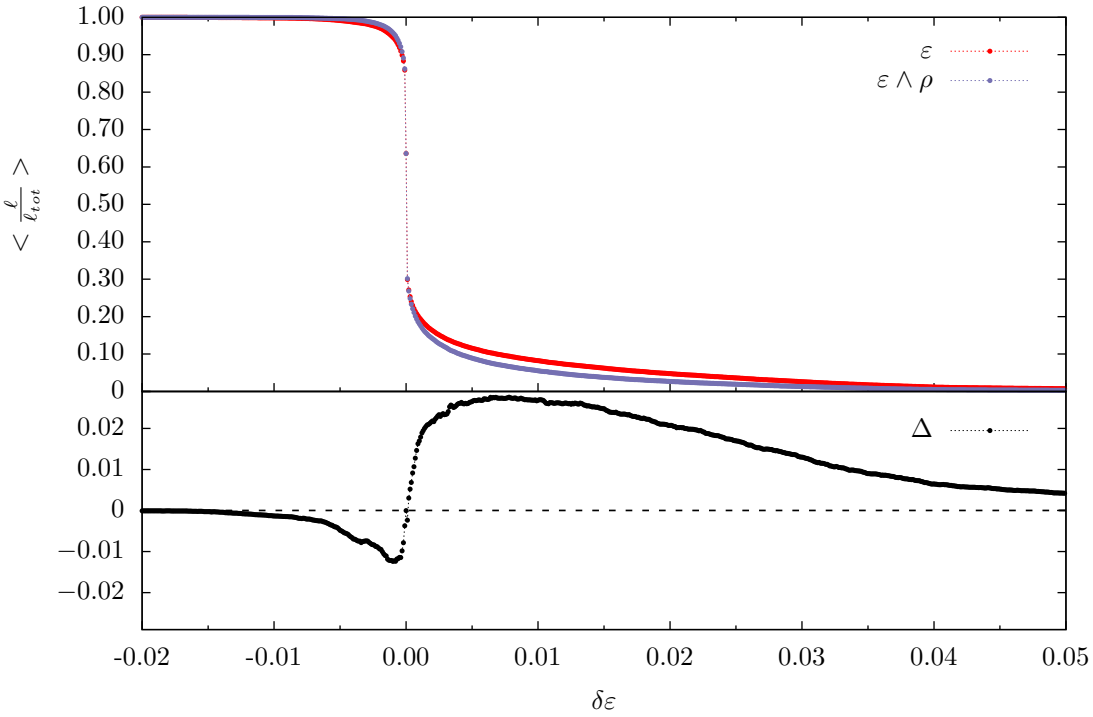

Figura 5.6. Fracción promedio $< \ell/\ell_{tot} >$ de conexiones que al eliminarse producen un cambio *mayor o igual* a $\delta\varepsilon$ en el error de una red. La curva *azul* corresponde a las redes 96 robustas, mientras que la *roja* a las redes sólo funcionales. En la ventana inferior de grafica la diferencia entre ambas. La diferencia es más notable en regiones de cambios considerables $\delta\varepsilon$ y decae abruptamente cerca del origen.

el conjunto de redes robustas. Esto parece estar de acuerdo con las distribuciones de la Fig. 5.5 en las cuales se puede ver que los pesos para el conjunto de las redes robustas tienden a ser menores en valor absoluto. Es decir, pareciera ser que la optimización de la robustez tiende a 'diluir' el efecto que la eliminación de las conexiones tiene sobre el error de las redes, con mayor cantidad de conexiones provocando menores cambios en él y menor cantidad de ellas provocando cambios considerables.

5.5. Conclusiones

La implementación del aprendizaje autónomo multiparamétrico para la optimización simultánea de la funcionalidad y la robustez de las redes resultó exitosa.

De 100 redes a las cuales aplicamos el aprendizaje, 96 evolucionaron optimizando ε y ρ a valores aceptables, mientras que sólo 4 casos fracasaron. La evolución de $< \varepsilon >$ y $< \rho >$ durante el aprendizaje demostró la convergencia hacia valores cercanos al valor objetivo de cada uno. A su vez, la comparación entre las distribuciones de redes dañadas para el caso de las redes optimizadas únicamente con respecto a ε y para las redes optimizadas con respecto a ε y ρ, demostró la notable diferencia entre ambos casos.

A nivel estructural, encontramos leves diferencias entre las distribuciones de los pesos para cada conjunto de redes, a decir: mayor cantidad de pesos con valores cercanos a cero en el caso de las redes robustas. Esto, luego de analizar el efecto que la eliminación de las conexiones tiene sobre el error de la red en cada caso, nos sugiere que uno de los resultados de la optimización con respecto a la robustez ρ es que hay *menos* conexiones cuyos efectos sobre el error ε al ser eliminadas son significativos. Sin embargo, las diferencias encontradas son lo suficientemente exiguas como para impedirnos ser conclusivos en esta dirección.

Capítulo 6

Conclusiones generales

Diseñamos redes neuronales artificiales funcionales y estructuralmente robustas, aplicando, como ley de evolución de los pesos sinápticos, el aprendizaje autónomo. Mediante una reformulación simple y directa del aprendizaje, logramos implementarlo para la optimización de ambos parámetros *en paralelo*. Los resultados que obtuvimos demuestran que las redes optimizadas de esta forma tienen un excelente rendimiento al cumplir con la función asignada, pudiendo generalizar lo aprendido durante el entrenamiento, y que además conservan su buen funcionamiento luego de sufrir daños estructurales.

El primer estudio consistió en optimizar autónomamente las redes únicamente para que fuesen funcionales, siendo su tarea asignada el reconocimiento de las vocales. Estudiando el rendimiento de las redes optimizadas sobre el conjunto de aprendizaje y sobre un conjunto de prueba compuesto por señales defectuosas, demostramos que las redes no sólo lograron el reconocimiento exitoso de los patrones que se usaron para entrenarlas, sino que demostraron tener un excelente rendimiento para el reconocimiento de los patrones originales con defectos aleatorios introducidos en ellos. Con ello, concluímos que al aprendizaje autónomo fue altamente efectivo para la construcción de redes *funcionalmente robustas*.

En lo que respecta a la comparación entre el aprendizaje autónomo y el aprendizaje por el método de retropropagación, encontramos que este último convergió mucho más rápido al valor mínimo del error. Esto lo atribuimos, por un lado, al nivel de exactitud garantizado por el método de retropropagación, según lo cual la dirección a seguir en el espacio de los pesos para minimizar el error se determina *analíticamente* iteración a iteración. Con esto, la regla de evolución es, *en cada iteración*, localmente la *óptima*. Por otro lado, en el caso del aprendizaje autónomo, la parte causal que dirige la evolución sólo depende de las consecuencias que tuvieron las modificaciones previas sobre la desviación de la red con respecto a su objetivo, es decir que la efectividad y la rapidez de convergencia del aprendizaje

dependen inherentemente de cómo sea sea la exploración en el espacio de parámetros, exploración que a su vez es de naturaleza *estocástica*.

Como resultado de nuestro segundo estudio demostramos que, aplicando el aprendizaje autónomo multiparamétrico a la optimización de las redes con respecto a su error de aprendizaje y a su robustez estructural (en particular, su robustez con respecto a la eliminación de un nodo de procesamiento) las redes resultaron ser marcadamente más resistentes a los daños estructurales en comparación con las que sólo fueron optimizadas para ser funcionales. Mientras que el 46 % de aquellas redes optimizadas solamente para ser funcionales perdieron su funcionalidad por encima de cierto umbral luego de ser dañadas, en el caso de las redes optimizadas para ser funcionales y robustas, este porcentaje fue del 15 %. A raíz de esto, la conclusión *central* de nuestro trabajo es que el aprendizaje autónomo multiparamétrico demostró ser un método efectivo para el diseño de redes *funcionales y estructuralmente robustas*, en el marco de la tarea que le asignamos y el diseño de red con el que trabajamos.

Por último, mencionamos una línea a seguir en futuras investigaciones. Como ya se mencionó, una de las desventajas del aprendizaje autónomo que advertimos durante nuestro trabajo es su baja rapidez de convergencia, hecho que resultó agravarse si intentabamos aumentar la dimensión del conjunto de aprendizaje (por ejemplo, el reconocimiento de las vocales *y* los números). Es posible que modulando el término estocástico ξ con otras potencias de la desviación ϵ esta convergencia se acelere. Con potencias mayores a uno, la convergencia tenderá a ser más estable cuando la desviación haya disminuido lo suficiente. Con potencias menores a uno, sucederá lo opuesto, y puede que el nivel de ruido para valores bajos de ϵ siga siendo lo suficientemente alto como para permitir que esta siga descendiendo.

Bibliografía

1. W. S. McCulloch y W. Pitts, *The bulletin of mathematical biophysics* **5**, 115-133 (1943).
2. R. Rojas, *Neural networks: a systematic introduction* (Springer, 1996).
3. R. P. Lippmann, *ASSP Magazine, IEEE* **4**, 4-22 (1987).
4. J. Freeman y D. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques* (Addison-Wesley, 1991).
5. F. Rosenblatt, *Psychological review* **65**, 386 (1958).
6. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller y E. Teller, *The journal of chemical physics* **21**, 1087-1092 (1953).
7. S. Kirkpatrick, *Journal of statistical physics* **34**, 975-986 (1984).
8. V. Černỳ, *Journal of optimization theory and applications* **45**, 41-51 (1985).
9. D. E. Rumelhart, G. E. Hinton y R. J. Williams, *Cognitive modeling* **5**, 1 (1988).
10. P. Kaluza y A. S. Mikhailov, *Physical Review E* **90**, 030901 (2014).
11. P. P. V. D. Smagt y B. J. A. Kröse, *An introduction to neural networks* (The University of Amsterdam, 1996).
12. S. C. Manrubia, A. S. Mikhailov y D. Zanette, *Emergence of dynamical order: synchronization phenomena in complex systems* (World Scientific, 2004), vol. 2.
13. W. Gerstner, en *Plausible neural networks for biological modelling* (Springer, 2001), págs. 23-48.
14. A. L. Hodgkin y A. F. Huxley, *The Journal of physiology* **117**, 500 (1952).
15. P. F. Kaluza, *Dinámica de sistemas neuronales con interacciones sinápticas.* Tesis de Maestría. Universidad Nacional de Cuyo. Insituto Balseiro, (2003).
16. J. Hansen y B. Koeppen, *Atlas of Neuroanatomy and Neurophysiology:* (Aubrey Durkin, 2002), `https://books.google.com.ar/books?id=ukQQBgAAQBAJ`.
17. M. Gilson, C. Savin y F. Zenke, *Frontiers in computational neuroscience* **9** (2015).

18. Y. Cohen, D. A. Wilson y E. Barkai, *Cerebral cortex* **25**, 180-191 (2015).

19. T. J. Hausrat *y col.*, *Nature communications* **6** (2015).

20. D. E. Rumelhart, J. L. McClelland, P. R. Group *y col.*, *Parallel distributed processing* (IEEE, 1988), vol. 1.

21. D. O. Hebb, *The organization of behavior: A neuropsychological theory* (John Wiley, New York, 1949).

22. J. A. Feldman y D. H. Ballard, *Cognitive science* **6**, 205-254 (1982).

23. J. J. Hopfield, *Proceedings of the national academy of sciences* **79**, 2554-2558 (1982).

24. G. Cybenko, *Mathematics of control, signals and systems* **2**, 303-314 (1989).

25. V. Kůrková, *Neural networks* **5**, 501-506 (1992).

26. E. Tapia y R. Rojas, en *Graphics recognition. Recent advances and perspectives* (Springer, 2003), págs. 329-340.

27. G. Hinton *y col.*, *Signal Processing Magazine, IEEE* **29**, 82-97 (2012).

28. P. Baldi, P. Sadowski y D. Whiteson, *Phys. Rev. Lett.* **114**, 111801 (11 2015).

29. J. Khan *y col.*, *Nature medicine* **7**, 673-679 (2001).

30. X. Hu *y col.*, *Nature Reviews Urology* **10**, 174-182 (2013).

31. M. Mohri, A. Rostamizadeh y A. Talwalkar, *Foundations of machine learning* (MIT press, 2012).

32. A. H. Klopf, "Brain function and adaptive systems: a heterostatic theory", inf. téc. (DTIC Document, 1972).

33. R. S. Sutton y A. G. Barto, *Introduction to reinforcement learning* (MIT Press Cambridge, 1998), vol. 135.

34. L. Perko, *Differential equations and dynamical systems* (Springer-Verlag, 1991), ISBN: 9780387974439, `https://books.google.com.ar/books?id=xftQAAAAMAAJ`.

35. M. O. Cáceres, *Elementos de estadistica de no equilibrio y sus aplicaciones al transporte en medios desordenados* (Reverté, 2003).

36. A Schuster, *International Journal of Computational Intelligence* **4**, 88-94 (2008).

37. H. Kitano, *Nature Reviews Genetics* **5**, 826-837 (2004).

38. L. H. Hartwell, J. J. Hopfield, S. Leibler y A. W. Murray, *Nature* **402**, C47-C52 (1999).

39. R. Cohen, K. Erez, D. Ben-Avraham y S. Havlin, *Physical Review Letters* **85**, 4626 (2000).

40. R. Albert, H. Jeong y A.-L. Barabási, *Nature* **401**, 130-131 (1999).

41. P. Kaluza, A. Kölzsch, M. T. Gastner y B. Blasius, *Journal of the Royal Society Interface* **7**, 1093-1103 (2010).

42. R. Albert, H. Jeong y A.-L. Barabási, *Nature* **406**, 378-382 (2000).

43. P Kaluza y A. S. Mikhailov, *The European Physical Journal B* **85**, 1-16 (2012).

44. P. Kaluza, M. Ipsen, M. Vingron y A. S. Mikhailov, *Physical Review E* **75**, 015101 (2007).

45. P. Kaluza, M. Vingron y A. S. Mikhailov, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **18**, 026113 (2008).

46. P. Kaluza y A. S. Mikhailov, *EPL (Europhysics Letters)* **79**, 48001 (2007).