Universidad de La Laguna

SECCIÓN DE MATEMÁTICAS | FACULTAD DE CIENCIAS

Claudio García Vargas

Redes neuronales de funciones base radiales

Radial Basis Functions Neural Networks

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, junio de 2017

Dirigido por

María Isabel Marrero Rodríguez

María Isabel Marrero Rodríguez
Dpto. de Análisis Matemático
Universidad de La Laguna
Aptdo. de Correos 456
38200 La Laguna, Tenerife

Agradecimientos

Quisiera aprovechar estas líneas para agradecer a todos los profesores que han sido partícipes de mi formación a lo largo de estos años. Mención especial a mi tutora del Trabajo Fin de Grado, la profesora María Isabel Marrero Rodríguez, no sólo por guiarme a través del mismo sino también por su apoyo en todo momento.

Resumen · Abstract

Resumen

En teoría de la aproximación, las funciones radiales condicionalmente definidas positivas, o funciones base radiales (RBF, por sus siglas en inglés), se usan para resolver problemas de interpolación en datos dispersos del espacio euclídeo. Entre los muchos subproductos de la interpolación RBF resultan particularmente interesantes las redes neuronales RBF, cuyo estudio ha evolucionado hasta constituir una disciplina de interés independiente en el campo de las ciencias de la computación y la inteligencia artificial, con numerosas aplicaciones en áreas tan diversas como finanzas, medicina, biología, geología, ingeniería o física. El presente trabajo tiene por objeto estudiar, mediante técnicas de análisis funcional, las propiedades de interpolación y aproximación por redes neuronales RBF en espacios de funciones continuas e integrables, ilustrando el corpus teórico con experimentos numéricos y alguna aplicación práctica.

Palabras clave: *Red neuronal RBF – Función base radial – Interpolación exacta – Aproximación universal – Problemas de clasificación.*

Abstract

In approximation theory, conditionally positive definite radial functions, or radial basis functions (RBF), are used to solve problems of interpolation of sparse data in Euclidean space. Among the many byproducts of RBF interpolation, the RBF neural networks are particularly interesting, as their study has evolved into an independent subject in the fields of computer science and artificial intelligence, with many applications in areas so diverse as finance, medicine, biology, geology, engineering or physics. The purpose of this report is to study, by means of functional-analytic techniques, the properties of interpolation and approximation by RBF neural networks in spaces of continuous and of integrable functions, illustrating the theory with some numerical experiments and practical applications.

Keywords: *RBF neural network – Radial Basis Function – Exact interpolation – Universal approximation – Classification problems.*

Contenido

Introducción

Una red neuronal artificial es un sistema procesador de información cuyo funcionamiento se inspira en el de las redes neuronales biológicas. Originariamente, aquéllas pretendían modelizar el funcionamiento de éstas. Con el transcurso del tiempo fueron surgiendo modelos artificiales carentes de significación biológica, pero que resultaron idóneos para resolver problemas de procesamiento de información.

Las principales características compartidas por ambos tipos de redes son:

- El procesamiento de la información ocurre en elementos llamados neuronas.
- Una red neuronal está formada por un conjunto de neuronas conectadas entre sí y con el exterior por medio de enlaces.
- Si bien entre las neuronas reales existen conexiones bidireccionales, se puede asumir que a través de los enlaces se transmiten señales en un único sentido sin más que considerar dos enlaces unidireccionales en sentidos opuestos. Esto motiva que una neurona tenga entradas y salidas (en realidad, cada neurona tiene una única salida que puede ser, a su vez, entrada de muchas otras).
- El resultado del procesamiento que ocurre en una neurona es una función no lineal de las entradas y de un conjunto de parámetros.

El último punto constituye la base del funcionamiento de las redes neuronales artificiales, ya que el conjunto de parámetros de los que dependen dichas funciones se va ajustando de acuerdo a lo que van aprendiendo. Para ser un poco más específicos, pensemos en un ejemplo dentro de uno de los campos en los que las redes neuronales artificiales tienen mayor auge: el reconocimiento de patrones. Supongamos que se quiere tener una red neuronal artificial capaz de reconocer letras manuscritas a partir de imágenes digitales (es decir, cada imagen digital corresponde a una letra manuscrita). Nuestra red artificial tendrá como entrada la matriz de puntos de la imagen digital y 28 salidas, una para cada letra; deseamos poner un 1 en la salida de la letra correspondiente a la imagen de

entrada y 0 en las demás. Cada persona traza las letras de forma diferente (más aún, nadie traza dos iguales), pero supongamos que disponemos de un conjunto de imágenes que sabemos a qué letra corresponden, escritos por un conjunto de varias personas, llamado muestra o base de datos. El primer paso es entrenar la red. Para ello se procede como sigue:

1. Se toma una valor inicial para los parámetros de la misma.
2. Se observa la salida que se obtiene para una imagen y se compara con la salida deseada (error).
3. Con un algoritmo adecuado se modifican los parámetros en función del error que se obtuvo en el paso 2.

Los pasos 2 y 3 se repiten con todas las imágenes de la muestra.

Luego de que la red ha sido entrenada, si el algoritmo es bueno y la muestra suficientemente heterogénea, la red es capaz de responder con un porcentaje de aciertos muy alto a las imágenes que en adelante le pongamos.

El aprendizaje de las redes neuronales puede ocurrir de dos modos: supervisado o no supervisado. En el modo supervisado el aprendizaje se logra comparando directamente la salida de la red con la respuesta correcta ya conocida. En el modo no supervisado, la información disponible sólo está en correlación con los datos de entrada o señales. Se espera que la red forme categorías a partir de estas correlaciones, y produzca una señal para cada categoría de la entrada. Claramente, el ejemplo antes expuesto correspondería al modo supervisado.

Las redes neuronales artificiales también pueden ser clasificadas, según su arquitectura, en redes progresivas y redes regresivas o recursivas. En las progresivas se tiene una estructura de capas, donde la salida de una cierta neurona sólo puede servir de entrada a neuronas de la capa siguiente salvo que se trate de una neurona de la última capa, en cuyo caso su salida ya será una salida de la red. Las redes recursivas permiten retroalimentación entre capas y presentan una dinámica de mayor complejidad. El esquema anteriormente comentado como ejemplo corresponde a una red progresiva.

El objetivo de este trabajo es dar una visión general de las redes neuronales de funciones base radiales (RBF por sus siglas en inglés), un subtipo de redes neuronales progresivas de la forma

$$h(\mathbf{x}) = \sum_n w_n \phi(\|\mathbf{x} - \mathbf{x}^n\|),$$

donde los vectores $\mathbf{x}^n$ se denominan centros y los escalares w_n, pesos. Aunque incialmente fueron introducidas para resolver el problema de la interpolación exacta, el estudio de las redes RBF ha evolucionado vertiginosamente hasta constituir una disciplina de interés independiente en el campo de las ciencias de la computación y la inteligencia artificial, con numerosas aplicaciones en áreas tan dispares como finanzas, medicina, biología, geología, ingeniería o física. Nuestro

interés se focaliza particularmente en caracterizar las funciones base que generan redes con la llamada propiedad de aproximación universal, es decir, que son densas en espacios de funciones continuas e integrables. Esto se hará en el capítulo 2 de la memoria, donde se probará, siguiendo [17] pero introduciendo también algún argumento original [5], que la condición de que la función de activación ϕ no sea un polinomio es necesaria y, junto con ciertas condiciones adicionales poco restrictivas, también suficiente para que la correspondiente red neuronal sea un aproximante universal. Ilustramos este resultado con algunos experimentos numéricos, cuyo código Python se recoge en el apéndice.

Para el resto del trabajo nos hemos apoyado fundamentalmente en la monografía [1]. El capítulo 1 motiva la introducción de las redes neuronales RBF desde varias perspectivas distintas, proporcionando así un marco teórico común para los diferentes enfoques. En el capítulo 3 nos ocupamos de una de las múltiples aplicaciones de las redes neuronales RBF, a saber, la resolución de problemas de clasificación, y discutimos como ejemplo la transmisión de símbolos en sistemas de comunicación.

La memoria se completa con la bibliografía y el preceptivo póster en lengua inglesa que resume sus contenidos.

1

Redes neuronales RBF

1.1. El problema de la interpolación exacta

Los métodos que utilizan funciones base radiales (RBF, por sus siglas en inglés) se originaron para dar solución al problema de interpolación exacta en varias variables.

Consideremos una aplicación definida en un conjunto de entrada $\mathbf{x}$, contenido en un espacio de dimensión d, con valores en un conjunto objetivo unidimensional t. El conjunto de datos se compone de N vectores de entrada $\mathbf{x}^n$, junto con los correspondientes t^n. El objetivo es encontrar una función $h(\mathbf{x})$ tal que

$$h(\mathbf{x}^n) = t^n, \quad n = 1, 2, \ldots, N. \tag{1.1}$$

La técnica de funciones base radiales presenta un conjunto de N *funciones base*, una para cada dato, de la forma $\phi(\|\mathbf{x} - \mathbf{x}^n\|)$, donde ϕ es una función no lineal cuyas características se discutirán más adelante. Así, la n-ésima función base dependerá de la distancia $\|\mathbf{x} - \mathbf{x}^n\|$, usualmente la euclídea, entre $\mathbf{x}$ y $\mathbf{x}^n$. La salida de la aplicación será una combinación lineal de las funciones base

$$h(\mathbf{x}) = \sum_n w_n \phi(\|\mathbf{x} - \mathbf{x}^n\|). \tag{1.2}$$

Las condiciones de interpolación (1.1) se pueden escribir matricialmente en la forma

$$\mathbf{\Phi}\mathbf{w} = \mathbf{t}, \tag{1.3}$$

donde $\mathbf{t} = (t^n)$, $\mathbf{w} = (w_n)$, y la matriz cuadrada $\mathbf{\Phi}$ tiene elementos $\mathbf{\Phi}_{nn'} = \phi(\|\mathbf{x}^n - \mathbf{x}^{n'}\|)$. Si existe la matriz inversa $\mathbf{\Phi}^{-1}$, es posible resolver (1.3):

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{t}. \tag{1.4}$$

Se demuestra que para una amplia colección de funciones ϕ la matriz $\mathbf{\Phi}$ es, en efecto, no singular, siempre que los datos sean distintos. Cuando los pesos

en la ecuación (1.2) se toman como el conjunto de valores dados por (1.4), la función $h(\mathbf{x})$ representa una superficie continua y diferenciable que pasa por todos y cada uno de los puntos dados.

Tanto los estudios teóricos como los empíricos han mostrado que, a efectos del problema de interpolación, muchas propiedades de la función interpolante son independendientes de la forma precisa de la función no lineal ϕ. En la literatura se han ensayado varias posibilidades para ϕ, si bien, gracias a la cantidad de propiedades analíticas que atesora, la elección más frecuente es la función gaussiana:

$$\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{1.5}$$

donde σ es un parámetro cuyo valor controla las propiedades de regularidad de la función interpolante. Otras elecciones posibles son las siguientes:

- La función
$$\phi(x) = (x^2 + \sigma^2)^{\alpha}, \quad 0 < \alpha < 1,$$
que para $\alpha = 1/2$ se conoce como *multicuádrica*.
- Multicuádricas inversas generalizadas:
$$\phi(x) = (x^2 + \sigma^2)^{-\beta}, \quad \beta > 0.$$
- La función *thin-plate spline*:
$$\phi(x) = x^2 \ln x.$$
- La función cúbica:
$$\phi(x) = x^3.$$
- La función lineal:
$$\phi(x) = x.$$
Nótese que esta función es lineal en x, pero $\phi(\|\mathbf{x} - \mathbf{x}^n\|)$ es no lineal en las componentes de $\mathbf{x}$.

El uso de métodos de funciones base radiales en problemas de interpolación exacta se ilustra en la figura 1.1, considerando una aplicación de una entrada y una salida.

La generalización a varias variables de salida es inmediata. Cada vector de entrada $\mathbf{x}^n$ debe ser aplicado exactamente en un único vector de salida $\mathbf{t}^n$, de componentes t_k^n. Así, (1.1) se convierte en

$$h_k(\mathbf{x}^n) = t_k^n, \quad n = 1, 2, \ldots, N,$$

donde los $h_k(\mathbf{x})$ se obtienen por superposición lineal de las mismas N funciones base que se usaron en el caso de una sola variable de salida:

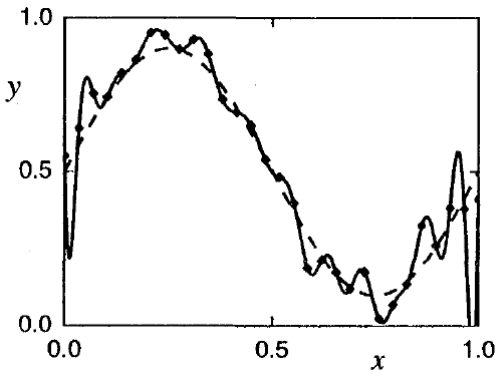

Figura 1.1. Ejemplo sencillo de interpolación exacta usando RBF. Se ha generado un conjunto de 30 datos muestreando la función $y = 0{,}5 + 0{,}4 \operatorname{sen} 2\pi x$, representada por la curva de trazo discontinuo, y añadiendo ruido gaussiano con desviación típica 0,05. La curva de trazo sólido muestra el interpolante que resulta de usar funciones base gaussianas de la forma (1.5) con amplitud $\sigma = 0{,}067$, lo que se corresponde aproximadamente con el doble de la distancia entre los datos. Los valores de los pesos de la capa de salida se han determinado usando técnicas de inversión matricial.

$$h_k(\mathbf{x}) = \sum_n w_{kn} \phi(\|\mathbf{x} - \mathbf{x}^n\|). \tag{1.6}$$

Por analogía con (1.4), los pesos se obtienen en la forma

$$w_{kn} = \sum_{n'} (\mathbf{\Phi}^{-1})_{nn'} t_k^{n'}. \tag{1.7}$$

Obsérvese que en (1.7) se usa la misma matriz $\mathbf{\Phi}^{-1}$ para todas las funciones de salida.

1.2. Redes neuronales de funciones base radiales

Las combinaciones lineales de funciones base radiales tratadas hasta el momento proporcionan un interpolante que pasa exactamente por cada uno de los puntos del conjunto de datos. Como ilustra el ejemplo de la figura 1.1, si los datos presentan distorsión entonces el interpolante exacto es, típicamente, una función muy oscilante, lo cual no resulta deseable, en general; en tal caso, se deberá elegir como interpolante otra función que sea más suave y promedie el ruido de los datos. Una limitación adicional del procedimiento de interpolación exacta descrito anteriormente es que el número de funciones base es igual al

número de puntos del conjunto de datos, de manera que si éste es grande la evaluación de la función interpolante puede resultar computacionalmente muy costosa.

Introduciendo una serie de modificaciones al procedimiento de interpolación exacta, se obtiene el modelo de redes neuronales de funciones base radiales. Tal modificación proporciona una función que interpola de forma suave a los datos, donde el número de funciones base está determinado por la complejidad de la función a representar, más que por el tamaño del conjunto de datos. Las modificaciones requeridas son las siguientes:

1. El número de funciones base usadas, M, no es necesariamente igual al total de datos, N, siendo usualmente una cantidad mucho menor que N.
2. No se exige que los centros de las funciones base sean los vectores de entrada. Por el contrario, la selección de centros adecuados forma parte del proceso de entrenamiento de la red.
3. Cada función base tiene sus propios parámetros ajustables (por ejemplo, la amplitud en el caso de la gaussiana), cuyos valores también se determinan durante el entrenamiento.
4. Los parámetros de sesgo están incluidos en la suma lineal. Se encargan de compensar la diferencia entre el valor promedio de las funciones base de activación sobre el conjunto de datos y los correspondientes promedios de los datos objetivo.

Una vez efectuados estos cambios en la fórmula de interpolación exacta (1.6), llegamos a la siguiente expresión de la aplicación que da la red neuronal de funciones base radiales:

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0}. \tag{1.8}$$

Si se desea, los sesgos w_{k0} pueden ser incluidos dentro del sumatorio añadiendo una función base extra, que denotaremos por ϕ_0, cuya activación se establece en 1. En el caso de funciones base gaussianas se tiene

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2}\right), \tag{1.9}$$

siendo $\mathbf{x}$ el vector de entrada d-dimensional con componentes x_i, y μ_j el vector que determina el centro de cada función base ϕ_j, con componentes μ_{ji}. Esta aplicación puede ser representada mediante el diagrama de la red neuronal que se muestra en la figura 1.2. Se pueden considerar topologías de redes RBF más generales, por ejemplo con más de una capa oculta, si bien este tipo de arquitecturas no son muy utilizadas en la práctica.

Apelando a la intuición, cabría conjeturar que las superposiciones lineales de funciones localizadas, como (1.8) y (1.9), pueden servir de aproximantes

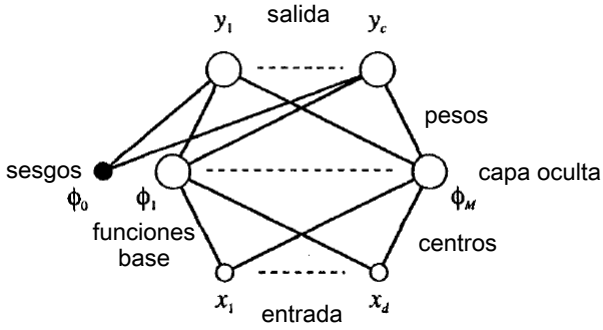

Figura 1.2. Arquitectura de una red neuronal de funciones base radiales, correspondiente a (1.8). Cada función base actúa como una unidad oculta. Los segmentos que conectan las funciones base ϕ_j con las entradas representan las componentes μ_{ji} del vector μ_j Los pesos w_{kj} se muestran como segmentos que conectan las funciones base con las unidades de salida, y los sesgos como pesos de una función base extra ϕ_0 cuya salida se establece en 1.

universales, es decir, conformar un conjunto denso en espacios de funciones continuas e integrables. Park y Sandberg [20, 21] probaron que, bajo condiciones poco restrictivas sobre las funciones base, se tiene efectivamente la propiedad de aproximación universal. Los resultados obtenidos por estos autores consisten en demostraciones de existencia que presuponen la disponibilidad de un número arbitrariamente grande de unidades en la capa oculta, pero sin facilitar procedimientos prácticos para la construcción de tales redes. No obstante, esos resultados son de vital importancia, por cuanto proporcionan la base teórica necesaria para desarrollar las aplicaciones de forma matemáticamente segura. El capítulo 2 de esta memoria expone un artículo de Liao, Fang y Nuttle [17] que rebaja las hipótesis impuestas por Park y Sandberg en su investigación. Concretamente, Liao *et al.* prueban que la hipótesis de integrabilidad no es necesaria para que la función base radial dé lugar a una red neuronal que sea un aproximante universal, sino que basta con que la base sea continua en casi todo punto, localmente esencialmente acotada y no sea un polinomio para que la correspondiente red RBF aproxime a cualquier función continua en un compacto con respecto a la norma uniforme. Es más, las redes RBF pueden aproximar a cualquier función de $L^p(\mu)$, donde $1 \leq p < \infty$ y μ es una medida finita, siempre que se use en la capa oculta una función de activación base radial esencialmente acotada que no sea un polinomio.

Girosi y Poggio [9] han probado que las redes neuronales RBF también poseen la propiedad de *mejor aproximación*. Un esquema de aproximación tiene esta propiedad si el conjunto de los aproximantes (esto es, el conjunto de funcio-

nes correspondientes a todas las elecciones posibles de los parámetros ajustables) contiene una función cuyo error de aproximación es mínimo para cualquiera de las funciones a aproximar.

1.3. Entrenamiento de la red

Un aspecto clave de las redes neuronales de funciones base radiales se encuentra en la distinción entre los roles de los pesos de la primera y segunda capa. Como se verá, las funciones base pueden ser interpretadas de forma que se permita a los pesos de la primera capa (es decir, a los parámetros que rigen las funciones base) ser establecidos mediante técnicas de aprendizaje no supervisado. Esto conlleva a un procedimiento de entrenamiento de la red que se desarrolla en dos etapas. En la primera etapa, se usa exclusivamente el conjunto de datos de entrada $\{\mathbf{x}^n\}$ para determinar los parámetros de las funciones base (μ_j y σ_j en el caso de las gaussianas consideradas anteriormente). Tras esto, las funciones base se mantienen fijas mientras que en la segunda etapa del entrenamiento se buscan los pesos de la segunda capa. Más adelante (sección 1.5) se discutirán algunas técnicas para la optimización de las funciones base; aquí asumiremos que los parámetros de estas funciones ya han sido elegidos y nos centraremos en discutir la cuestión de cómo optimizar los pesos de la segunda capa. Adviértase que si hubiera menos funciones base que datos entonces, en general, no será posible encontrar un conjunto de pesos que hagan que la función interpolante se ajuste exactamente al conjunto de datos.

Para empezar, se considera la red neuronal RBF definida en (1.8), absorbiendo los sesgos en los pesos para obtener

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}),$$

donde ϕ_0 es una función base extra con un valor de activación fijado en $\phi_0 = 1$. Esta red puede ser representada matricialmente escribiendo

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}\phi,$$

con $\mathbf{W} = (w_{kj})$ y $\phi = (\phi_j)$. Puesto que las funciones base se consideran fijas, la red es equivalente a una red de dos capas, cuyos pesos pueden ser optimizados minimizando una función de error adecuada. A este propósito, como se verá, resulta particularmente conveniente considerar la suma del error cuadrático medio, que se expresa en la forma siguiente:

$$E = \frac{1}{2}\sum_{n}\sum_{k}\{y_k(\mathbf{x}^n) - t_k^n\}^2, \tag{1.10}$$

donde t_k^n es el valor objetivo de la unidad de salida k-ésima cuando la red tiene a $\mathbf{x}^n$ como vector de entrada. Como la función de error es una función cuadrática de los pesos, es posible encontrar su mínimo en términos de la solución de un sistema de ecuaciones lineales, a saber,

$$\mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{W}^\top = \mathbf{\Phi}^\top \mathbf{T}, \tag{1.11}$$

siendo $(\mathbf{T})_{nk} = t_k^n$ y $(\mathbf{\Phi})_{nj} = \phi_j(\mathbf{x}^n)$. La solución formal para los pesos viene dada por

$$\mathbf{W}^\top = \mathbf{\Phi}^\dagger \mathbf{T},$$

donde

$$\mathbf{\Phi}^\dagger = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top$$

denota la matriz pseudoinversa de $\mathbf{\Phi}$.

Si la matriz $\mathbf{\Phi}$ no es cuadrada, evidentemente no tendrá una verdadera inversa; sin embargo, la pseudoinversa posee la propiedad de que $\mathbf{\Phi}^\dagger \mathbf{\Phi} = \mathbf{I}$, siendo $\mathbf{I}$ la matriz identidad. Obsérvese que, en general, $\mathbf{\Phi}\mathbf{\Phi}^\dagger \neq \mathbf{I}$. En caso de que la matriz $\mathbf{\Phi}^\top \mathbf{\Phi}$ sea singular, el sistema (1.11) no tendrá solución única. No obstante, si se define la matriz pseudoinversa como

$$\mathbf{\Phi}^\dagger = \lim_{\varepsilon \to 0} (\mathbf{\Phi}^\top \mathbf{\Phi} + \varepsilon \mathbf{I})^{-1} \mathbf{\Phi}^\top,$$

se demuestra que este límite siempre existe y su valor minimiza la función de error (1.10).

En la práctica, las ecuaciones (1.11) se resuelven usando métodos de descomposición en valores singulares, para evitar los problemas debidos al mal condicionamiento que pueda presentar la matriz $\mathbf{\Phi}$. De esta forma, se comprueba que los pesos de la segunda capa pueden ser encontrados rápidamente usando técnicas lineales de inversión matricial.

En lo que sigue, se considerarán redes neuronales de funciones base radiales en las que la dependencia de los pesos de la segunda capa sea lineal, y donde el error cometido venga controlado por la función de error cuadrático medio. Es posible considerar otras funciones de activación de tipo no lineal, y también otras elecciones de la función de error; pero entonces el problema de la determinación de los pesos de la segunda capa ya no es lineal, sino que se hace necesaria una optimización no lineal de los pesos, mientras que una de las ventajas principales del uso de redes neuronales RBF es justamente la posibilidad de evitar este tipo de optimización durante el entrenamiento.

Como ilustración del uso de redes RBF, podemos considerar el conjunto de datos del ejemplo de la figura 1.1 y la aplicación obtenida usando una red RBF en la que el número de funciones base es menor que el número de datos, como se muestra en la figura 1.3. El parámetro σ de esta última fue elegido aproximadamente igual al doble del espaciado medio entre los centros de las funciones base. En la sección 1.5 se discutirán algunas técnicas para determinar

los parámetros de las funciones base, incluyendo σ_j. Por el momento, nos limitaremos a hacer notar el efecto que producen elecciones inadecuadas de σ: las figuras 1.4 y 1.5 muestran el resultado de elegir valores de σ muy pequeños y muy grandes, respectivamente.

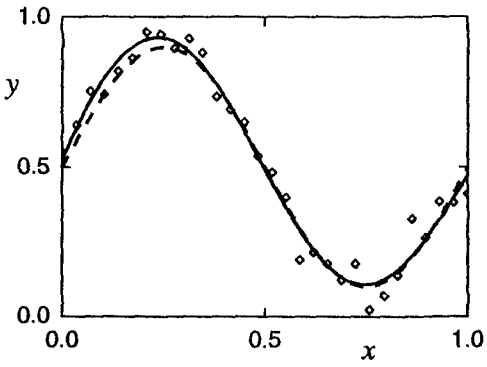

Figura 1.3. La gráfica muestra el mismo conjunto de 30 puntos de la figura 1.1, junto con una aplicación de la red (curva sólida) en la que el número de funciones base se ha fijado en 5, cantidad significativamente inferior al número de datos. Como centros de las funciones base se ha elegido aleatoriamente un subconjunto de los datos de entrada. Como parámetro de amplitud para todas las funciones base se ha fijado $\sigma = 0{,}4$, que de nuevo viene a ser el doble de la distancia media entre los centros. Los pesos de la segunda capa se han determinado minimizando una función de error cuadrático medio mediante descomposición en valores singulares.

1.4. Teoría de la regularización

La teoría de la regularización proporciona un método para controlar las propiedades de regularidad de una función. Consiste en añadir a la función de error un término diseñado para penalizar las aplicaciones que no son suaves. Por simplicidad en la notación consideraremos redes con una única salida y, de manera que con un error cuadrático medio la función de error total a minimizar adopta la forma

$$E = \frac{1}{2}\sum_n \{y(\mathbf{x}^n) - t^n\}^2 + \frac{\nu}{2}\int |Py|^2 \, d\mathbf{x}, \tag{1.12}$$

donde P es un cierto operador diferencial y ν es denominado parámetro de regularización. Las funciones y con una curvatura grande típicamente darán

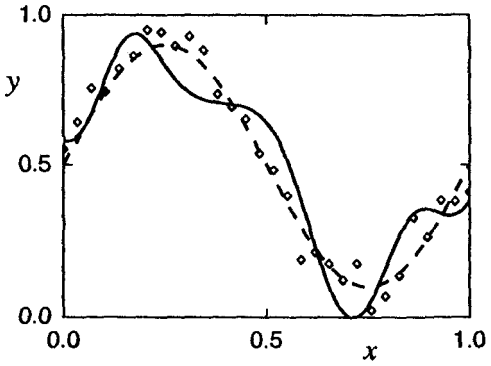

Figura 1.4. La gráfica muestra la misma situación que la figura 1.3, pero tomando como amplitud $\sigma = 0{,}08$. La función resultante de la red no es suficientemente suave y proporciona una representación muy pobre de la función que generó los datos.

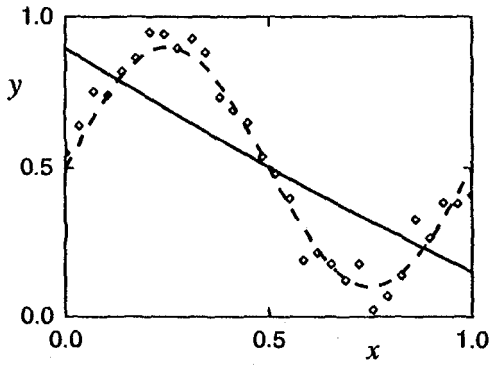

Figura 1.5. La gráfica muestra la misma situación que la figura 1.3, pero tomando como amplitud $\sigma = 10{,}0$. La función resultante de la red es excesivamente suave y, de nuevo, proporciona una representación muy pobre de la función que generó los datos.

lugar a valores grandes de $|Py|^2$, y por tanto aumentarán el error total. Así, el parámetro ν controla la importancia relativa del término de regularización, y con ello el grado de suavidad de la función y.

Podemos resolver el problema de mínimos cuadrados regularizado (1.12) mediante cálculo variacional. Si imponemos que la derivada funcional de (1.12) con respecto a y sea nula, resulta

$$\sum_n \{y(\mathbf{x}^n) - t^n\}\delta(\mathbf{x} - \mathbf{x}^n) + \nu \widehat{P} P y(\mathbf{x}) = 0, \tag{1.13}$$

donde $\widehat{P}$ es el operador diferencial adjunto de P y δ es la función delta de Dirac. Las ecuaciones (1.13) son las *ecuaciones de Euler-Lagrange* correspondientes a (1.12). Se puede escribir una solución formal de dichas ecuaciones en términos de las *funciones de Green* del operador $\widehat{P}P$, que son las funciones $G(\mathbf{x}, \mathbf{x}')$ para las cuales

$$\widehat{P} P G(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}'). \tag{1.14}$$

Si el operador P es invariante por traslaciones y rotaciones, las funciones de Green dependen solamente de la distancia $\|\mathbf{x} - \mathbf{x}'\|$, y por lo tanto son radiales. La solución formal de (1.13) es entonces

$$y(\mathbf{x}) = \sum_n w_n G(\|\mathbf{x} - \mathbf{x}^n\|), \tag{1.15}$$

expresión que tiene la forma de un desarrollo lineal en funciones base radiales. Sustituyendo (1.15) en (1.13) y usando (1.14) obtenemos

$$\sum_n \{y(\mathbf{x}^n) - t^n\}\delta(\mathbf{x} - \mathbf{x}^n) + \nu \sum_n w_n \delta(\mathbf{x} - \mathbf{x}^n) = 0.$$

Integrando ahora sobre un entorno de $\mathbf{x}^n$ encontramos que los coeficientes w_n satisfacen

$$y(\mathbf{x}^n) - t^n + \nu w_n = 0. \tag{1.16}$$

Para determinar los valores de w_n, evaluamos (1.15) en los datos de entrenamiento $\mathbf{x}^n$ y sustituimos en (1.16), lo que proporciona los valores buscados como soluciones del sistema lineal

$$(\mathbf{G} + \nu \mathbf{I})\mathbf{w} = t. \tag{1.17}$$

Aquí, $(\mathbf{G})_{nn'} = G(\|\mathbf{x}^{n'} - \mathbf{x}^n\|)$, $(\mathbf{w})_n = w_n$, $(\mathbf{t})_n = t_n$, e $\mathbf{I}$ denota la matriz identidad.

Si, en particular, se elige el operador P tal que

$$\int |Py|^2 \, d\mathbf{x} = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l! 2^l} \int |D^l y(\mathbf{x})|^2 \, d\mathbf{x},$$

donde $D^{2l} = (\nabla^2)^l$ y $D^{2l+1} = \nabla(\nabla^2)^l$, siendo ∇ y ∇^2 los operadores gradiente y laplaciano, respectivamente, entonces las funciones de Green son gaussianas de amplitud σ.

Vemos así que existe un paralelismo entre esta forma de desarrollo en funciones base y la discutida en la sección 1.1 en el contexto de interpolación exacta.

Ahora, las funciones de Green $G(\|\mathbf{x} - \mathbf{x}^n\|)$ se corresponden con las funciones base $\phi(\|\mathbf{x} - \mathbf{x}^n\|)$, y cada una de ellas está centrada en un dato del conjunto de entrenamiento. Nótese que cuando $\nu = 0$, (1.17) se reduce al resultado de interpolación exacta (1.3). El efecto del término de regularización se ilustra en la figura 1.6.

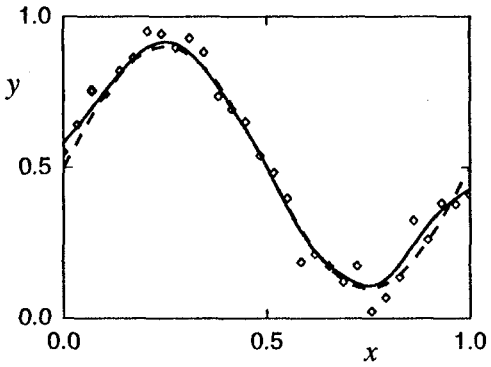

Figura 1.6. La gráfica muestra la misma situación que la figura 1.1, con igual amplitud $\sigma = 0{,}067$, pero añadiendo un término de regularización de coeficiente $\nu = 40$. Aunque la función resultante (representada por la curva de trazo continuo) no se ajusta a los datos, es más suave y proporciona una aproximación mucho mejor a la función objetivo (representada por la curva de trazo discontinuo).

En la práctica, la regularización también se aplica a redes neuronales de funciones base radiales en las que las funciones base no están necesariamente centradas en los datos, y en las que el número de funciones base no coincide con el número de datos. Además, se puede considerar términos de regularización cuyas funciones base no son necesariamente las funciones de Green. Mientras el término de regularización sea una función cuadrática de la aplicación proporcionada por la red, los pesos de la segunda capa pueden ser calculados, de nuevo, como solución de un sistema de ecuaciones lineales que minimiza un error cuadrático. Por ejemplo, el regularizador

$$\frac{\nu}{2} \sum_n \sum_k \sum_i \left(\frac{\partial^2 y_k^n}{\partial x_i^2} \right)^2$$

penaliza las aplicaciones que tienen curvatura grande y conduce a pesos de la segunda capa que son solución del sistema

$$\mathbf{MW} = \mathbf{\Phi}^\top \mathbf{T}, \tag{1.18}$$

donde

$$(\mathbf{M})_{jj'} = \sum_n \left\{ \phi_j^n \phi_{j'}^n + \nu \sum_i \left(\frac{\partial^2 \phi_j^n}{\partial x_i^2} \frac{\partial^2 \phi_{j'}^n}{\partial x_i^2} \right) \right\}$$

y $\mathbf{\Phi} = (\phi_j^n)$ es como anteriormente. Cuando $\nu = 0$, (1.18) se reduce a (1.11). La inclusión del término de regularización no incrementa el tiempo de computación, que se invierte fundamentalmente en resolver (1.18).

Para un desarrollo detallado de los contenidos de esta sección remitimos a [10, Section 5.5].

1.5. Optimización de las funciones base

Una de las principales ventajas con que cuentan las redes neuronales de funciones base radiales es la posibilidad de elegir los parámetros de las unidades ocultas sin necesidad de aplicar un proceso de optimización no lineal a toda la red. En esta sección se abordarán algunas de las diferentes estrategias a tener en cuenta para la selección de dichos parámetros.

Las redes neuronales de funciones base radiales comparecen en multitud de aplicaciones, además de las ya mencionadas de interpolación exacta y regularización y de la estimación de probabilidades *a posteriori* en problemas de clasificación que se verá en el capítulo 3. Todos estos puntos de vista sugieren que los parámetros de las funciones base deben conformar una representación de la densidad de probabilidad de los datos de entrada. Ello conduce a un procedimiento de entrenamiento no supervisado para la optimización de los parámetros de las funciones base que sólo dependerá de los datos de entrada, ignorando la información objetivo. Los centros μ_j de las funciones base pueden ser considerados como *prototipos* de los vectores de entrada. Algunas de las estrategias que se discutirán en la presente sección están motivadas por estas consideraciones.

Existen muchas aplicaciones posibles de las redes neuronales donde abundan los datos de entrada sin etiquetar, mientras que los datos etiquetados escasean. Por ejemplo, puede resultar sencillo reunir ejemplos de datos de entrada para la red sin procesar, pero el etiquetarlos con variables objetivo seguramente requiera de un experto humano, lo que limita severamente la cantidad de datos que pueden ser etiquetados en un tiempo razonable. El proceso de entrenamiento en dos etapas de las redes neuronales RBF es particularmente ventajoso para este tipo de aplicaciones, por cuanto la determinación de la representación no lineal dada por la segunda capa de la red se puede efectuar mediante el uso de una cantidad grande de datos no etiquetados, dejando un número relativamente reducido de los parámetros conducentes a la tercera capa por estimar usando los datos etiquetados. En cada una de las dos etapas podemos asegurar que la cantidad de parámetros a estimar es muy inferior a la cantidad de datos, como sería deseable para obtener una generalización adecuada.

Una de las principales dificultades potenciales con las redes neuronales RBF emana del carácter local de la representación de las unidades ocultas. Si el subespacio de datos tiene dimensión intrínseca d y los centros de las funciones base llenan este subespacio, entonces el número de centros crece exponencialmente con d. Además de incrementar el tiempo de cómputo, un número elevado de funciones base requiere un número elevado de patrones de entrenamiento para garantizar que los parámetros de la red son determinados correctamente.

El problema se vuelve particularmente severo si se consideran variables de entrada que presentan una variación significativa pero tienen poca importancia a la hora de determinar las variables de salida adecuadas. Estas entradas irrelevantes son frecuentes en las aplicaciones. Cuando los centros se eligen solamente a partir de los datos de entrada, no hay forma de distinguir las entradas relevantes de las que no lo son. Ilustramos esta idea con la gráfica de la figura 1.7. En ella se observa una variable y que es una función no lineal de la variable de entrada x_1. Nos gustaría poder aproximar dicha función usando una red RBF. Seleccionamos las funciones base de forma que cubran la región del eje x_1 donde hay datos presentes. Supongamos ahora que se introduce otra variable x_2 que no está correlacionada con x_1. Entonces el número de funciones base necesarias para cubrir la región requerida del espacio de entrada crece drásticamente, como se muestra en la figura 1.8. Sin embargo, si la variable y no depende de x_2, estas funciones base adicionales carecen de utilidad para ajustar el valor de y.

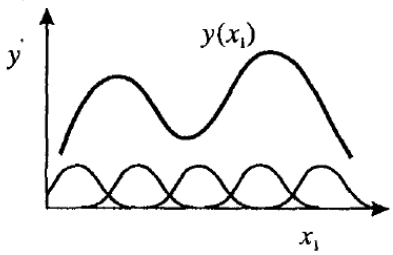

Figura 1.7. Función $y(x_1)$ modelizada mediante una red de funciones base radiales.

Por tanto, existen razones de cierto calado que aconsejan el uso de métodos no supervisados para determinar los parámetros de la segunda capa de una red RBF mediante la modelización de la densidad de los datos de entrada. Este método también ha probado su eficacia en la práctica. Sin embargo, es necesario advertir que una elección óptima de los parámetros de las funciones base para estimar la densidad no siempre conduce a un ajuste óptimo de la curva. Tal situación se ilustra en la figura 1.9.

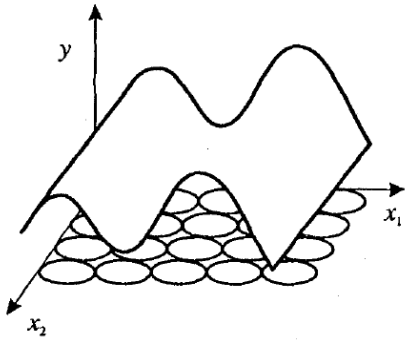

Figura 1.8. Función de la figura 1.7 tras la introducción de la variable irrelevante x_2. El número de funciones base cuyas localizaciones están determinadas únicamente por los datos de entrada crece drásticamente, a pesar de que x_2 no aporta información relevante para hallar la variable de salida.

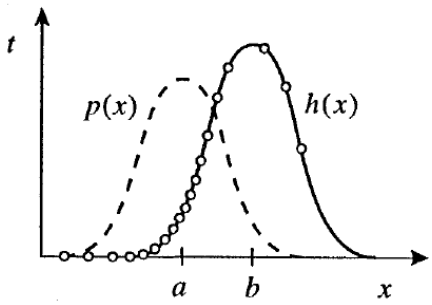

Figura 1.9. Ilustración del hecho de que el uso de métodos no supervisados que se apoyan en una estimación de la densidad para determinar los parámetros de las funciones base no es necesariamente optimal a la hora de aproximar la función objetivo. El conjunto de datos está representado por los círculos y se genera a partir de una distribución gaussiana p, que se corresponde con el trazo discontinuo. El aprendizaje no supervisado de una función base gaussiana la centraría en el punto a, dando una buena aproximación a p. Los valores objetivo para los datos de entrada están generados a partir de una gaussiana centrada en b y representada mediante el trazo sólido. La función base centrada en a no proporciona un buen ajuste a la curva h, mientras que si la función base estuviera centrada en b representaría h de manera exacta.

A continuación se muestran algunas técnicas para elegir apropiadamente los centros μ_j.

1.5.1. Subconjuntos de datos

Un procedimiento sencillo para seleccionar los centros μ_j de las funciones base consiste en elegirlos aleatoriamente de entre los datos de entrada. Claramente, este proceso no es óptimo por cuanto puede requerir un elevado número de funciones base para alcanzar un funcionamiento adecuado, pero suele usarse como paso previo a otras técnicas adaptativas iterativas.

Un segundo procedimiento consiste en tomar el conjunto completo de datos como centros de las funciones base y removerlos selectivamente de manera que se produzca la mínima distorsión en el funcionamiento del sistema.

Estos procedimientos sólo están orientados a la selección de los centros; los parámetros de amplitud σ_j deben ser elegidos por otros métodos. Un enfoque heurístico consistiría en tomar todos los σ_j iguales entre sí y a algún múltiplo de la distancia promedio entre los centros de las funciones base. De esta forma se podría garantizar que las funciones base se superponen hasta un cierto grado y obtener rápidamente una representación bastante suave de la distribución de los datos de entrenamiento, si bien es cierto que tal representación probablemente diste mucho de ser óptima.

1.5.2. Algoritmos de *clustering*

Aunque por razones de espacio no las desarrollaremos aquí, cabe citar que la simple selección de un subconjunto de datos como centros de las funciones base se puede mejorar usando técnicas de *clustering* para encontrar un conjunto de centros que refleje con mayor exactitud la distribución de todos los datos. Entre estas técnicas se encuentra el *algoritmo clustering de K-medias* [19] o el *mapa de características autoorganizado* [13].

1.5.3. Mínimos cuadrados ortogonales

Este método se fundamenta en la idea de selección de funciones base que se desarrolla a continuación. Supongamos que se comienza considerando una red de una sola función base. De forma sucesiva, para cada dato se ajusta el centro de la función base al vector de entrada correspondiente a dicho punto y luego se establecen los pesos de la segunda capa mediante la técnica de la matriz pseudoinversa, usando el conjunto de N datos al completo. Finalmente, se retiene el valor del centro que minimice el error residual. De esta manera, en cada paso del algoritmo se incrementa el número de funciones base. Si en un determinado momento se han elegido l datos como centros de las funciones base, se entrenan $N - l$ redes donde cada uno de los $N - l$ puntos restantes se toman como centros de funciones base adicionales. De éstas se retiene la que minimice el error cuadrático medio residual, y el algoritmo avanza a la siguiente etapa.

Un enfoque de este tipo sería muy costoso computacionalmente, pues obligaría a calcular en cada paso una solución pseudoinversa completa para cada posible elección de funciones base. Un procedimiento mucho más eficiente para conseguir el mismo resultado es el de mínimos cuadrados ortogonales propuesto por Chen *et al.* [4]. En resumen, el algoritmo involucra la adición secuencial de nuevas funciones base, cada una centrada en uno de los datos, tal como se acaba de exponer. Esto se consigue construyendo un conjunto de vectores ortogonales en el espacio S generados por los vectores de las unidades de activación de la capa oculta para cada patrón del conjunto de datos. Así, será posible calcular de forma directa qué punto debe ser elegido como centro de la siguiente función base a fin de minimizar el error cuadrático medio residual. Los valores de los pesos de la tercera capa se determinan simultáneamente. Si se deja actuar el algoritmo durante el tiempo necesario se seleccionarán todos los puntos y el error residual acabaría siendo nulo, por lo que debe ser interrumpido antes de que esto suceda.

1.5.4. Modelos gaussianos mixtos

Hasta el momento se han discutido procedimientos heurísticos para seleccionar los parámetros de las funciones base de manera que éstas aproximen la distribución de los datos de entrada. Un enfoque más certero, sin embargo, sería el que se detalla a continuación. Las funciones base de la red neuronal pueden ser consideradas como las componentes de un modelo de densidad mixta, cuyos parámetros son optimizados por máxima verosimilitud. Representamos la densidad de los datos de entrada mediante un modelo mixto de la forma

$$p(\mathbf{x}) = \sum_{j=1}^{M} P(j)\phi_j(\mathbf{x}),$$

donde los parámetros $P(j)$ son los coeficientes mixtos y $\phi_j(\mathbf{x})$ las funciones base de la red. Obsérvese que los coeficientes mixtos pueden ser considerados como las probabilidades *a priori* de los datos que han sido generados por la componente mixta j-ésima. La función de verosimilitud viene dada por

$$\mathcal{L} = \prod_{n} p(\mathbf{x}^n)$$

y es maximizada tanto en relación a los coeficientes mixtos $P(j)$ como a los parámetros de las funciones base. Esta maximización se puede lograr, por ejemplo, calculando las derivadas de $\mathcal{L}$ respecto de los parámetros y usándolas en algoritmos estándar de optimización no lineal.

Una vez optimizado el modelo mixto, los coeficientes $P(j)$ pueden ser descartados para usar las funciones base en la red RBF, donde los pesos de la segunda capa se hallan mediante entrenamiento supervisado.

1.6. Entrenamiento supervisado

El establecimiento de las funciones base mediante la estimación de la densidad de los datos de entrada no tiene en cuenta las etiquetas objetivo asociadas a esos datos. Con el fin de encontrar los parámetros de las funciones base que permitan optimizar el tiempo de computación de las salidas de la red, debemos tener en cuenta los datos objetivo durante la fase de entrenamiento; es decir, se debe ejecutar un entrenamiento supervisado.

Los parámetros de las funciones base para la regresión se pueden hallar tratando los centros y amplitudes, junto con los pesos de la segunda capa, como parámetros adaptativos a determinar minimizando una función de error. La elección de los mencionados parámetros mediante entrenamiento supervisado supone un problema de optimización no lineal que usualmente requiere de un cierto esfuerzo computacional, aunque se puede rebajar el coste si se identifican las funciones base relevantes frente a las que no lo son, evitando así cálculos innecesarios. Se han descrito en la literatura técnicas que permiten encontrar las unidades relevantes de forma eficiente. También se puede utilizar cualquiera de las técnicas no supervisadas descritas anteriormente para inicializar los parámetros de las funciones base y proceder luego a efectuar un ajuste más fino mediante aprendizaje supervisado. Sin embargo, hay que tener en cuenta que el entrenamiento supervisado de las funciones base no garantiza que éstas permanezcan localizadas y que con este tipo de entrenamiento se pierden algunas de las principales ventajas de las redes RBF, a saber, el entrenamiento rápido en dos etapas y la posibilidad de interpretar la representación de la capa oculta.

2

Aproximación universal por redes neuronales RBF

2.1. Introducción

La posibilidad de aproximación universal mediante redes neuronales progresivas (densidad del conjunto de estas funciones en espacios de funciones continuas o integrables) ha sido estudiada por muchos investigadores, entre los que podemos citar los siguientes: Chen y Chen [3]; Cybenko [6]; Hornik [11, 12]; Leshno, Lin, Pinkus y Schocken [16]; Mhaskar y Micchelli [18]; Park y Sandberg [20, 21]. Bajo un conjunto de restricciones muy leves sobre las funciones de activación que se encuentran en la capa oculta, estos autores han demostrado que una red neuronal progresiva de tres capas es capaz de aproximar una extensa clase de funciones, incluyendo tanto a las funciones continuas como a las integrables.

Los resultados conocidos en la literatura se han construido principalmente sobre redes neuronales progresivas de tres capas con un único nodo de salida lineal. En este capítulo se mantendrá este diseño estándar. Las funciones que se pueden obtener mediante redes neuronales progresivas de tres capas son de la forma

$$\sum_{i=1}^{N} c_i g(x, \theta_i, b_i),$$

donde N representa el número de nodos ocultos, $x \in \mathbb{R}^n$ es una variable y, para $i = 1, 2, \ldots, N$, $b_i, c_i \in \mathbb{R}$, $\theta_i \in \mathbb{R}^n$ son parámetros y $g(x, \theta_i, b_i)$ es la función de activación que se usa en la capa oculta.

Cabe destacar que la mayoría de estas funciones de activación se pueden categorizar en dos clases: funciones ridge y funciones base radiales. Las funciones ridge son de la forma

$$g(x, \theta, b) = \sigma(\theta^\top x + b),$$

donde σ es una función de $\mathbb{R}$ en $\mathbb{R}$, $x \in \mathbb{R}^n$ es una variable, $\theta \in \mathbb{R}^n$ es un vector director y $b \in \mathbb{R}$ es un sesgo. La sigmoide, de uso común, es un ejemplo de

función ridge, siendo su expresión

$$g(x) = \frac{1}{1 + e^{-(\theta^\top x + b)}}.$$

Las funciones base radiales adoptan la forma

$$g(x, \theta, b) = \phi\left(\frac{x - \theta}{b}\right), \tag{2.1}$$

donde ϕ aplica $\mathbb{R}^n$ en $\mathbb{R}$, $x \in \mathbb{R}^n$ es una variable, $\theta \in \mathbb{R}^n$ es un vector central y $b \in \mathbb{R}$ es un parámetro de extensión. El ejemplo paradigmático de esta clase de funciones es la gaussiana

$$g(x) = \exp\left(-\frac{\|x - \theta\|^2}{b}\right).$$

Los trabajos de investigación que usan funciones ridge como función de activación son más comunes (Cybenko [6]; Hornik [11, 12]; cf. de León [15]). El resultado obtenido por Leshno, Lin, Pinkus y Schocken [16] se considera uno de los más generales. Estos autores probaron que si se usa una función de activación ridge en la capa oculta, continua en casi todo punto, localmente esencialmente acotada y que no sea un polinomio, entonces una red neuronal de tres capas puede aproximar a cualquier función continua con respecto a la norma uniforme.

En comparación, la literatura que usa funciones base radiales como función de activación es menos extensa (Chen y Chen [3]; Mhaskar y Micchelli [18]). El estudio más conocido es probablemente el de Park y Sandberg [20, 21], quienes probaron que si se usa como función de activación en la capa oculta una función base radial, continua en casi todo punto, acotada e integrable en $\mathbb{R}^n$ y con integral no nula, entonces una red neuronal de tres capas puede aproximar a cualquier función de $L^p(\mathbb{R}^n)$ con respecto a la norma L^p, para $1 \leq p < \infty$.

El artículo de Liao, Fang y Nuttle [17], en el que se fundamenta este capítulo, se centra en ampliar las conclusiones a las que llegaron Park y Sandberg en su investigación, probando que la hipótesis de integrabilidad no es necesaria para que la función base radial dé lugar a un aproximante universal válido. Por el contrario, bastan condiciones más débiles, similares a las impuestas por Leshno y colaboradores en su trabajo sobre las funciones ridge. En concreto, demostraremos que si se usa una función base radial como función de activación en la capa oculta, continua en casi todo punto, localmente esencialmente acotada y que no sea un polinomio, entonces las redes neuronales RBF de tres capas pueden aproximar a cualquier función continua con respecto a la norma uniforme. Es más, las redes RBF pueden aproximar a cualquier función de $L^p(\mu)$, donde $1 \leq p < \infty$ y μ es una medida finita, siempre que se use en la capa oculta una función de activación base radial esencialmente acotada que no sea un polinomio.

La organización de este capítulo es como sigue. En la sección 2.2 introduciremos algunas definiciones y notaciones básicas, reservando los resultados principales para la sección 2.3. En la sección 2.4 se recogen algunos experimentos numéricos que apoyan dichos resultados.

2.2. Preliminares

A lo largo de este capítulo se empleará la siguiente notación: $\mathbb{R}^n$ denotará el espacio euclídeo real n-dimensional y K un subconjunto compacto de $\mathbb{R}^n$. El conjunto de todas las funciones continuas definidas en K, con la norma del máximo

$$\|f\|_{C(K)} = \max_{x\in K} |f(x)|,$$

será denotado por $C(K)$. Además, $C^\infty(\mathbb{R}^n)$ hará referencia al conjunto de todas las funciones infinitamente diferenciables definidas en $\mathbb{R}^n$, y $C_c^\infty(\mathbb{R}^n)$ al conjunto de todas las funciones infinitamente diferenciables con soporte compacto en $\mathbb{R}^n$.

Recordemos que, dada una medida finita μ, la norma del supremo esencial de una función f se define como

$$\|f\|_{L^\infty(\mathbb{R}^n)} = \operatorname*{ess\,sup}_{x\in\mathbb{R}^n} |f(x)| = \text{ínf}\left\{\lambda \in \mathbb{R} \,|\, \mu\left\{x : |f(x)| \geq \lambda\right\} = 0\right\}.$$

Además, para $1 \leq p < \infty$,

$$\|f\|_{L^p(\mu)} = \left(\int_{\mathbb{R}^n} |f(x)|^p d\mu(x)\right)^{1/p}.$$

Denotamos por $L^\infty(\mathbb{R}^n)$ el conjunto de todas las funciones f para las cuales $\|f\|_{L^\infty(\mathbb{R}^n)} < \infty$, y por $L^p(\mu)$ el conjunto de todas las funciones f tales que $\|f\|_{L^p(\mu)} < \infty$.

Análogamente, dado un conjunto compacto $K \subset \mathbb{R}^n$ se consideran la norma $\|\cdot\|_{L^\infty(K)}$ para el espacio $L^\infty(K)$ y la norma $\|\cdot\|_{L^p(K)}$ para el espacio $L^p(K)$. El conjunto de todas las funciones f tales que $\|f\|_{L^\infty(K)} < \infty$ (respectivamente, $\|f\|_{L^p(K)} < \infty$) para cada conjunto compacto $K \subset \mathbb{R}^n$ se denotará por $L^\infty_{loc}(\mathbb{R}^d)$ (respectivamente, $L^p_{loc}(\mathbb{R}^d)$).

Diremos que una función es continua en casi todo punto con respecto a una medida μ, si el conjunto de sus puntos de discontinuidad tiene μ-medida nula. Un conjunto de funciones S es denso en $C(K)$ (respectivamente, en $L^p(\mu)$), si para cualquier $\varepsilon > 0$ y $f \in C(K)$ (respectivamente, $f \in L^p(\mu)$), existe $g \in S$ tal que $\|g - f\|_{L^\infty(\mu)} \leq \varepsilon$ (respectivamente, $\|g - f\|_{L^p(\mu)} \leq \varepsilon$).

La convolución de dos funciones f y g se define como

$$(f * g)(x) = \int_{\mathbb{R}^n} f(x-t)g(t)\,dt \quad (x \in \mathbb{R}^n).$$

La transformada de Fourier de una función f se denotará por $\widehat{f}$. El soporte de una función f se denotará por supp f. Una n-upla $\alpha = (\alpha_1, \ldots, \alpha_n)$ de enteros no negativos se denominará multi-índice. Definimos su orden por $|\alpha| = \alpha_1 + \cdots + \alpha_n$ y su factorial por $\alpha! = \alpha_1! \cdots \alpha_n!$ El operador diferencial D^α se define como

$$D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_n}\right)^{\alpha_n}.$$

Dada una función $\phi(x)$ $(x \in \mathbb{R}^n)$,

$$\text{span}\,\{\phi(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$$

denota el conjunto de todas las funciones definidas sobre $\mathbb{R}^n$ de la forma

$$x \mapsto \sum_{i=1}^{N} \beta_i \phi(a_i x + \theta_i),$$

donde $\beta_i \in \mathbb{R}$ $(i = 1, 2, \ldots, N)$ y N es un entero positivo. Para más terminología y propiedades de análisis funcional relacionadas con lo anterior se puede consultar el texto de Rudin [22].

2.3. Resultados principales

Recordemos que las funciones base radiales son de la forma (2.1). De ahora en adelante, por simplicidad, las escribiremos como $\phi(ax+\theta)$, con $a \in \mathbb{R}$ y $x, \theta \in \mathbb{R}^n$.

Para empezar, tenemos el siguiente resultado.

Teorema 2.1 *Sea ϕ una función de $\mathbb{R}^n$ en $\mathbb{R}$. Si $\phi \in C^\infty(\mathbb{R}^n)$ y no es un polinomio entonces, para cualquier conjunto compacto $K \subset \mathbb{R}^n$,*

$$\Phi = \text{span}\,\{\phi\,(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$$

es denso en $C(K)$ con respecto a la norma del máximo, es decir, dada una función $f \in C(K)$ y cualquier $\varepsilon > 0$, existe $g \in \Phi$ tal que $|f(x) - g(x)| \leq \varepsilon$ para todo $x \in K$.

Demostración. Supongamos que Φ no es denso en $C(K)$. Por el teorema de Hahn-Banach (cf. [6]), existe una medida signada finita $\lambda \neq 0$ en K, tal que

$$\int_K \phi(ax+\theta)\, d\lambda(x) = 0 \quad (a \in \mathbb{R},\ \theta \in \mathbb{R}^n).$$

Como $\phi \in C^\infty(\mathbb{R}^n)$, usando el desarrollo de Taylor en varias variables podemos escribir

$$\phi(ax+\theta) = \sum_{|\alpha|=0}^{\infty} \frac{1}{\alpha!}(D^\alpha\phi)(\theta)(ax)^\alpha = \sum_{|\alpha|=0}^{\infty} \frac{a^{|\alpha|}}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha$$
$$= \phi(\theta) + a\sum_{|\alpha|=1} \frac{1}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha + a^2\sum_{|\alpha|=2} \frac{1}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha + \cdots .$$

Sea

$$H(a) = \int_K \phi(ax+\theta)\, d\lambda(x).$$

Puesto que $H(a) = 0$ para todo $a \in \mathbb{R}$ y todo $\theta \in \mathbb{R}^n$, la derivada k-ésima de H con respecto a a se expresa de la siguiente forma:

$$\frac{d^k H}{da^k}$$
$$= \int_K \left[\sum_{|\alpha|=k} \frac{k!}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha + a \sum_{|\alpha|=k+1} \frac{(k+1)!}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha + \cdots \right] d\lambda(x) = 0,$$

para todo $\theta \in \mathbb{R}^n$. Haciendo $a = 0$ encontramos que

$$\left.\frac{d^k H}{da^k}\right|_{a=0}$$
$$= \int_K \left[\sum_{|\alpha|=k} \frac{k!}{\alpha!}(D^\alpha\phi)(\theta)x^\alpha \right] d\lambda(x) = \sum_{|\alpha|=k} \left[\frac{k!}{\alpha!}(D^\alpha\phi)(\theta) \int_{\mathbb{R}^n} x^\alpha\, d\lambda(x) \right] = 0$$

para todo $\theta \in \mathbb{R}^n$. Equivalentemente,

$$\sum_{i=1}^{r(k)} c_i(\theta)t_i = 0$$

cualquiera que sea $\theta \in \mathbb{R}^n$, donde $r(k)$ es el número de multi-índices α de orden $|\alpha| = k$ y, para cada $i = 1, 2, \ldots, r(k)$, $c_i(\theta) = k!(D^\alpha\phi)(\theta)$ y

$$t_i = \frac{1}{\alpha!}\int_K x^\alpha\, d\lambda(x).$$

Ya que $\phi \in C^\infty(\mathbb{R}^n)$ y no es un polinomio, cada función $c_i(\theta)$ es continua y no constante. Por tanto, $c_i(\theta)$ puede tomar una infinidad de valores para diferentes valores de θ. Luego, deben existir, al menos, $r(k)+1$ valores de θ que hacen que el anterior sistema lineal sea sobredeterminado. Así, la única solución del dicho sistema lineal es la trivial: $t_i = 0$ para cada $i = 1, 2, \ldots, r(k)$. Es decir,

$$\int_K x^\alpha\, d\lambda(x) = 0$$

para todo multi-índice α. De aquí se deduce para la transformada de Fourier de λ que

$$\widehat{\lambda}(t) = \int_K e^{-it^\top x}\, d\lambda(x) = 0 \quad (t \in \mathbb{R}^n).$$

En virtud de [23, Example 7.12b y Theorem 7.15a], inferimos que $\lambda = 0$. Esta contradicción completa la prueba. □

El Teorema 2.1 nos dice que si la función de activación usada en la capa oculta es infinitamente diferenciable y no es un polinomio, entonces la correspondiente red neuronal RBF de tres capas es un aproximante universal. La condición de que esta función sea infinitamente diferenciable es muy restrictiva en teoría, pero puesto que las redes neuronales suelen ser entrenadas mediante algoritmos recursivos, los cuales asumen frecuentemente que la función de activación en la capa oculta es diferenciable, tal requisito no causa demasiados problemas en la práctica. Afortunadamente, podemos debilitarlo gracias al siguiente Lema 2.2.

Antes de proceder a enunciar y demostrar dicho lema, es preciso hacer un comentario. Wu *et al.* [25] observaron que la prueba del Lema 2.2 dada en [17, Lemma 1] (trabajo en el que, como ya quedó dicho, se basa el presente capítulo) no es matemáticamente rigurosa, y obtuvieron una nueva demostración de este resultado. De hecho, tal como Wu *et al.* señalan:

> *In Liao* et al. *(2003), the key points of the proof of this lemma are the statements that «then for any multi-index α such that $|\alpha| = \infty$, we have $D^\alpha \sigma * \omega(x) = 0$» and «according to Friedman (1963, pp. 57–59), σ is a polynomial of degree $< |\alpha| = \infty$». However, $D^\alpha \sigma * \omega(x)$ is certainly not mathematically well defined for $|\alpha| = \infty$. And the Corollary in Friedman (1963, pp. 57–59), which requires $|\alpha|$ to be fixed and finite, can not be directly applied.*

En su nueva demostración, Wu *et al.* distinguen dos casos. Mediante argumentos de análisis funcional, hemos conseguido probar que el segundo caso queda descartado y que la demostración del primero sigue directamente sin más que utilizar identidades aproximadas. Nuestra demostración, que recogemos debajo, ha sido publicada en [5] y presentada en formato de póster en la V Escuela-Taller de la Red de Análisis Funcional y Aplicaciones (ICMAT, Madrid, 2015).

Lema 2.2 *Sea σ una aplicación de $\mathbb{R}^n$ en $\mathbb{R}$. Si $\sigma \in L^\infty_{loc}(\mathbb{R}^n)$ y σ no es un polinomio, entonces existe al menos una $\omega \in C^\infty_c(\mathbb{R}^n)$ tal que*

$$(\sigma * \omega)(x) = \int_{\mathbb{R}^n} \sigma(x - t)\omega(t)\, dt$$

no es un polinomio.

Demostración. Por simplicidad en la notación, escribimos $\mathcal{D} = C_c^\infty(\mathbb{R}^n)$ y proveemos a este espacio de su topología usual [23, Definition 6.3 y Theorem 6.4], de manera que su dual $\mathcal{D}'$ es el espacio de las distribuciones sobre $\mathbb{R}^n$ [23, Definition 6.7], al que dotamos de su topología débil* [23, Section 6.16]. Sobre el espacio $C^\infty(\mathbb{R}^n)$ consideramos igualmente su topología usual [23, Section 1.46].

Supongamos que $\sigma * \omega$ es un polinomio para toda $\omega \in \mathcal{D}$, y sea B la bola unidad cerrada de $\mathbb{R}^n$. Denotamos por $\mathcal{D}_B$ la familia de todas las funciones $u \in \mathcal{D}$ soportadas en B. Entonces, en particular, $\sigma * u$ es un polinomio siempre que $u \in \mathcal{D}_B$.

Afirmamos la existencia de un entero positivo m tal que grado $(\sigma * u) \leq m$ para toda $u \in \mathcal{D}_B$. En efecto, recordemos que $\mathcal{D}_B$ tiene estructura de espacio de Fréchet [23, Section 1.46]. Por otra parte, es evidente que

$$\mathcal{D}_B = \bigcup_{k=1}^{\infty} Y_k,$$

donde

$$Y_k = \{u \in \mathcal{D}_B : \text{grado}\,(\sigma * u) \leq k\} \quad (k \in \mathbb{N})$$

es un subespacio de $\mathcal{D}_B$. Además, cada Y_k $(k \in \mathbb{N})$ es cerrado en $\mathcal{D}_B$. Para verlo, fijemos $k \in \mathbb{N}$ y supongamos que $\{u_j\}_{j=1}^\infty \subset Y_k$ es tal que $\lim_{j\to\infty} u_j = u$ en la topología de $\mathcal{D}_B$. Entonces $\lim_{j\to\infty} u_j = u$ también en la topología de $\mathcal{D}$ [23, Theorem 6.5]. Como la aplicación $\sigma * \cdot$ es continua de $\mathcal{D}$ en $C^\infty(\mathbb{R}^n)$ [23, Theorem 6.33], sigue que $\lim_{j\to\infty} \sigma * u_j = \sigma * u$ en la topología de $C^\infty(\mathbb{R}^n)$. Pero la sucesión $\{\sigma * u_j\}_{j=1}^\infty$ está en efl espacio de todos los polinomios de grado no superior a k, el cual es un subespacio de $C^\infty(\mathbb{R}^n)$ de dimensión finita, y por lo tanto cerrado [23, Theorem 1.21]. Esto conlleva a que $\sigma * u$ sea un polinomio de grado no mayor que k, así que $u \in Y_k$.

Llegados a este punto, se puede aplicar el teorema de categoría de Baire [23, Section 2.2] para inferir que algún Y_m tiene interior no vacío. Como Y_m es un subespacio de $\mathcal{D}_B$, esto fuerza a que $Y_m = \mathcal{D}_B$ y establece nuestra afirmación sobre la existencia de m.

Para completar la prueba, se considera una identidad aproximada $\{h_j\}_{j=1}^\infty \subset \mathcal{D}_B$ [23, Definition 6.31]. Entonces, como se acaba de demostrar, cada $\sigma * h_j$ $(j \in \mathbb{N})$ es un polinomio de grado no mayor que m, y $\lim_{j\to\infty} \sigma * h_j = \sigma$ en la topología de $\mathcal{D}'$ [23, Theorem 6.32]. El operador D^α $(\alpha \in \mathbb{N},\ |\alpha| = m + 1)$ es secuencialmente continuo de $\mathcal{D}'$ en $\mathcal{D}'$ [23, Theorem 6.17], de manera que

$$0 = \lim_{j\to\infty} D^\alpha(\sigma * h_j) = D^\alpha \sigma \quad (\alpha \in \mathbb{N},\ |\alpha| = m + 1)$$

en $\mathcal{D}'$. El corolario en [8, pp. 57-59] ya permite concluir que σ es un polinomio y que grado $(\sigma * u) \leq m$, como se pretendía. □

Lema 2.3 *Sea σ una aplicación de $\mathbb{R}^n$ en $\mathbb{R}$. Si $\sigma \in L^\infty_{loc}(\mathbb{R}^n)$ y σ es continua en casi todo punto entonces, para cada $\omega \in C^\infty_c(\mathbb{R}^n)$, $\sigma * \omega$ puede ser aproximada uniformemente desde $\Sigma = \operatorname{span}\{\sigma(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$.*

Demostración. Recordemos que

$$(\sigma * \omega)(x) = \int_{\mathbb{R}^n} \sigma(x-t)\omega(t)\,dt \quad (x \in \mathbb{R}^n)$$

está bien definida.

Supongamos que supp $\omega \subseteq [-T,T]^n$. Demostraremos que $\sigma * \omega$ puede ser aproximada uniformemente en $[-T,T]^n$ por

$$\sum_{i=1}^{m^n} \sigma(x-t_i)\,\omega(t_i)\left(\frac{2T}{m}\right)^n,$$

donde $\{t_i \in \mathbb{R}^n : i = 1,2,\ldots,m^n\}$ es el conjunto formado por todos los puntos en $[-T,T]^n$ de la forma

$$\left[-T+\frac{2i_1T}{m},\ldots,-T+\frac{2i_nT}{m}\right]^n, \quad i_1,i_2,\ldots,i_n = 1,2,\ldots,m.$$

Pongamos $\Delta_i = [t_{i-1},t_i]$. Sumando y restando en el primer miembro la expresión

$$\sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,\omega(t)\,dt$$

y aplicando la desigualdad triangular, encontramos que

$$\begin{aligned}
&\left|\int_{\mathbb{R}^n} \sigma(x-t)\,\omega(t)\,dt - \sum_{i=1}^{m^n} \sigma(x-t_i)\,\omega(t_i)\left(\frac{2T}{m}\right)^n\right| \\
&\le \left|\int_{\mathbb{R}^n} \sigma(x-t)\,\omega(t)\,dt - \sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,\omega(t)\,dt\right| \\
&\quad + \left|\sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,\omega(t)\,dt - \sum_{i=1}^{m^n} \sigma(x-t_i)\,\omega(t_i)\left(\frac{2T}{m}\right)^n\right|. \qquad (2.2)
\end{aligned}$$

Si en el segundo término del segundo miembro de (2.2) utilizamos que

$$\int_{\Delta_i} dt = \left(\frac{2T}{m}\right)^n,$$

podemos escribir:

$$\left| \sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,\omega(t)\,dt \;-\; \sum_{i=1}^{m^n} \sigma(x-t_i)\,\omega(t_i) \left(\frac{2T}{m}\right)^n \right|$$
$$= \left| \sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,[\omega(t)-\omega(t_i)]\,dt \right|$$
$$\leq \sum_{i=1}^{m^n} \int_{\Delta_i} |\sigma(x-t_i)|\,|\omega(t)-\omega(t_i)|\,dt.$$

Como ω es continua, es uniformemente continua en el compacto $[-T,T]^n$. Por consiguiente, cabe elegir m lo suficientemente grande como para que

$$\sum_{i=1}^{m^n} \int_{\Delta_i} |\sigma(x-t_i)|\,|\omega(t)-\omega(t_i)|\,dt \leq \varepsilon.$$

En el primer término del segundo miembro de (2.2), tenemos:

$$\left| \int_{\mathbb{R}^n} \sigma(x-t)\,\omega(t)\,dt \;-\; \sum_{i=1}^{m^n} \int_{\Delta_i} \sigma(x-t_i)\,\omega(t)\,dt \right|$$
$$= \left| \sum_{i=1}^{m^n} \int_{\Delta_i} [\sigma(x-t)-\sigma(x-t_i)]\,\omega(t)\,dt \right|$$
$$\leq \sum_{i=1}^{m^n} \int_{\Delta_i} |\sigma(x-t)-\sigma(x-t_i)|\,|\omega(t)|\,dt.$$

Ya que σ es continua en casi todo punto, el conjunto de sus puntos de discontinuidad tiene medida cero. Por lo tanto, dado un $\delta > 0$ se puede encontrar un número contable de intervalos abiertos, cuya unión U es de medida δ, tal que σ es uniformemente continua en $[-T,T]^n \backslash U$. Puesto que

$$\Delta_i = (\Delta_i \backslash U) \cup (\Delta_i \cap U) \quad (i = 1, 2, \ldots, m^n)$$

y esta unión es disjunta, la estimación anterior se puede expresar como

$$\sum_{i=1}^{m^n} \int_{\Delta_i} |\sigma(x-t)-\sigma(x-t_i)|\,|\omega(t)|\,dt$$
$$= \sum_{i=1}^{m^n} \int_{\Delta_i \backslash U} |\sigma(x-t)-\sigma(x-t_i)|\,|\omega(t)|\,dt$$
$$+ \sum_{i=1}^{m^n} \int_{\Delta_i \cap U} |\sigma(x-t)-\sigma(x-t_i)|\,|\omega(t)|\,dt. \tag{2.3}$$

Ahora, ya que σ es uniformemente continua en $[-T,T]^n \backslash U$, en el primer término del segundo miembro de (2.3) se puede elegir m suficientemente grande como para que

$$\sum_{i=1}^{m^n} \int_{\Delta_i \backslash U} |\sigma(x-t) - \sigma(x-t_i)|\,|\omega(t)|\,dt \leq \varepsilon.$$

En el segundo término del segundo miembro de (2.3), al tenerse que

$$\int_U dt = \delta,$$

se puede elegir δ lo suficientemente pequeño como para que

$$\sum_{i=1}^{m^n} \int_{\Delta_i \cap U} |\sigma(x-t) - \sigma(x-t_i)|\,|\omega(t)|\,dt \leq 2\|\sigma\|_{L^\infty([-T,T]^n)} \|\omega\|_{L^\infty(\mathbb{R}^n)} \delta \leq \varepsilon.$$

De ambas acotaciones resulta

$$\sum_{i=1}^{m^n} \int_{\Delta_i} |\sigma(x-t) - \sigma(x-t_i)|\,|\omega(t)|\,dt \leq 2\varepsilon.$$

Se concluye que

$$\left| \int_{\mathbb{R}^n} \sigma(x-t)\,\omega(t)\,dt - \sum_{i=1}^{m^n} \sigma(x-t_i)\,\omega(t_i) \left(\frac{2T}{m}\right)^n \right| \leq 3\varepsilon$$

para todo $x \in [-T,T]^n$. La prueba queda completada. □

Combinando los Lemas 2.2 y 2.3 con el Teorema 2.1, ya se consigue el resultado principal.

Teorema 2.4 *Sea σ una aplicación de $\mathbb{R}^n$ en $\mathbb{R}$. Si σ es continua en casi todo punto, localmente esencialmente acotada y no es un polinomio entonces, para cualquier conjunto compacto $K \subset \mathbb{R}^n$, $\Sigma = \operatorname{span}\{\sigma(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$ es denso en $C(K)$ con respecto a la norma del máximo, es decir, dada $f \in C(K)$ y cualquier $\varepsilon > 0$, existe $g \in \Sigma$ tal que $\|f-g\|_{L^\infty(K)} \leq \varepsilon$ para todo $x \in K$.*

Demostración. Por el Lema 2.2, sabemos que existe algún $\omega \in C_c^\infty(\mathbb{R}^n)$ tal que $\sigma * \omega$ no es un polinomio. Como $\sigma * \omega \in C^\infty(\mathbb{R}^n)$, por el Teorema 2.1 se tiene que $\operatorname{span}\{(\sigma*\omega)(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$ es denso en $C(K)$. Por el Lema 2.3, $\sigma * \omega$ puede ser aproximado uniformemente desde Σ, de donde se infiere que $\operatorname{span}\{(\sigma*\omega)(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$ puede ser aproximado uniformemente desde Σ. Así pues, Σ es denso en $C(K)$. □

El resultado anterior nos dice que si la función de activación usada en la capa oculta es continua en casi todo punto, localmente esencialmente acotada y no es un polinomio, entonces una red neuronal de tres capas con función de activación base radial es un aproximante universal. Este hecho extiende significativamente los resultados obtenidos por Park y Sandberg [20, 21]. En particular, para que la red tenga la propiedad de aproximación universal, la función de activación no tiene que ser necesariamente integrable.

Cabe observar que si la función de activación usada en la capa oculta es un polinomio, la red neuronal sólo puede producir polinomios de un cierto grado. Por tanto, la condición de que la función de activación no sea un polinomio siempre es necesaria para conseguir la aproximación universal. No sabemos si los requisitos de que dicha función de activación sea continua en casi todo punto y localmente esencialmente acotada también son necesarios.

Aparte de la aproximación universal en $C(K)$, tenemos el siguiente resultado sobre aproximación universal en el espacio $L^p(\mu)$, donde $1 \leq p < \infty$ y μ es una medida finita.

Teorema 2.5 *Sean μ una medida finita en $\mathbb{R}^n$ y σ una aplicación de $\mathbb{R}^n$ en $\mathbb{R}$. Si $\sigma \in L^\infty(\mu)$ y no es un polinomio, entonces $\Sigma = \operatorname{span}\{\sigma(ax+\theta) : a \in \mathbb{R},\ \theta \in \mathbb{R}^n\}$ es denso en $L^p(\mu)$, para $1 \leq p < \infty$.*

Demostración. Fijemos $1 \leq p < \infty$ y supongamos que Σ no es denso en $L^p(\mu)$. Como $\sigma \in L^p(\mu)$, Σ es un subespacio de $L^p(\mu)$. Por el teorema de Hahn-Banach (cf. [6]), existe una función $g \in L^q(\mu) \setminus \{0\}$, donde $q = p/(p-1)$, tal que

$$\int_{\mathbb{R}^n} \sigma(ax+\theta)g(x)\,d\mu(x) = 0$$

para todo $a \in \mathbb{R}$ y $\theta \in \mathbb{R}^n$. Por otra parte, el Lema 2.2 proporciona $\omega \in C_c^\infty(\mathbb{R}^n)$ tal que $\sigma * \omega$ no es un polinomio.

Consideremos la integral

$$\int_{\mathbb{R}^n} (\sigma * \omega)(ax+\theta)g(x)\,d\mu(x) = \int_{\mathbb{R}^n} \left[\int_{\mathbb{R}^n} \sigma(ax+\theta-t)\omega(t)\,dt \right] g(x)\,d\mu(x).$$

Como

$$\int_{\mathbb{R}^n} \left[\int_{\mathbb{R}^n} |\sigma(ax+\theta-t)\omega(t)g(x)|\,dt \right] d\mu(x)$$

$$\leq \|\sigma\|_{L^\infty(\mathbb{R}^n)} \|\omega\|_{L^1(\mathbb{R}^n)} \mu(\mathbb{R}^n)^{1/p} \|g\|_{L^q(\mathbb{R}^n)} < \infty,$$

por el teorema de Fubini se tiene

$$\begin{aligned}
&\int_{\mathbb{R}^n} (\sigma * \omega)(ax+\theta) g(x)\, d\mu(x) \\
&= \int_{\mathbb{R}^n} \left[\int_{\mathbb{R}^n} \sigma(ax+\theta-t)\omega(t)\, dt \right] g(x)\, d\mu(x) \\
&= \int_{\mathbb{R}^n} \left[\int_{\mathbb{R}^n} \sigma(ax+\theta-t) g(x)\, d\mu(x) \right] \omega(t)\, dt = 0.
\end{aligned}$$

Ya que $\sigma * \omega \in C^\infty(\mathbb{R}^n)$ y no es un polinomio, el resto de la prueba es idéntica a la del Teorema 2.1, con $d\lambda(x) = g(x)\, dx$. □

2.4. Experimentos numéricos

En esta sección se presentan varios experimentos numéricos que ilustrarán la validez de los resultados alcanzados en este capítulo. Mostraremos gráficamente que, aun cuando la función de activación que comparece en la capa oculta no sea integrable, una red neuronal RBF tiene la propiedad de aproximación universal siempre y cuando dicha función de activación satisfaga las condiciones (menos restrictivas) de ser continua en casi todo punto y localmente esencialmente acotada, pero no un polinomio. En concreto, se hará una comparativa usando, esencialmente, dos funciones de activación diferentes en la capa oculta para aproximar otras dos funciones de una variable, evaluando también la diferencia entre los errores de aproximación. Por un lado consideraremos la función base radial

$$F(x) = \exp\left(\frac{\|x-\theta\|^2}{b} \right),$$

que llamaremos P-gaussiana, y por otro lado la gaussiana tradicional o N-gaussiana

$$G(x) = \exp\left(-\frac{\|x-\theta\|^2}{b} \right).$$

Aquí, $x \in \mathbb{R}^n$ es la variable, $\theta \in \mathbb{R}^n$ es un vector de centros, y $b \in \mathbb{R}$ es un parámetro de amplitud. Mientras que la N-gaussiana G es integrable, la P-gaussiana F no lo es, si bien satisface las condiciones más débiles propuestas en este capítulo.

En primer lugar consideramos la función utilizada como ejemplo en las gráficas del capítulo 1, a saber, $f(x) = 0{,}5 + 0{,}4 \operatorname{sen} 2\pi x$ en el intervalo $[0, 1]$. En las situaciones descritas en las figuras 2.1, 2.2 y 2.3, la recíproca de la gaussiana (P-gaussiana) proporciona una aproximación similar o mejor que la gaussiana (N-gaussiana). Como segundo ejemplo consideramos la función $f(x) = \operatorname{sen}(3(x + 0{,}5)^3 - 1)$ en el intervalo $[-1, 1]$. La figura 2.4 muestra la aproximación de la función f obtenida usando las funciones de activación mencionadas, con los parámetros que se indican. Se observa que la salida de la P-gaussiana proporciona un ajuste mejor que la correspondiente a la N-gaussiana.

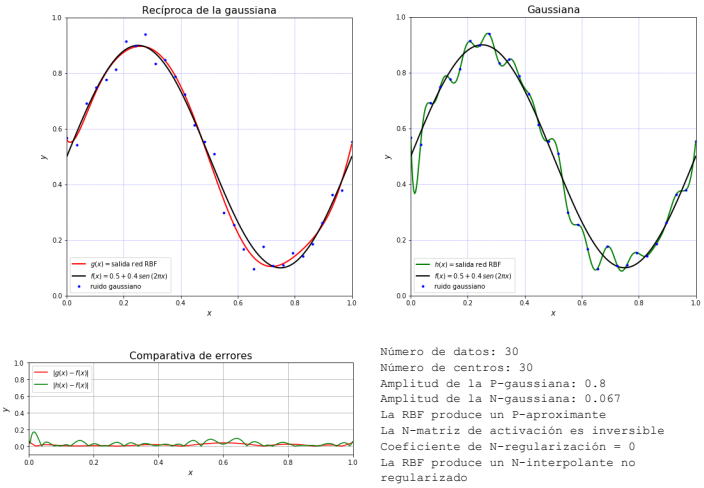

Figura 2.1. Con 30 datos, 30 centros, y amplitudes 0.8 para la P-gaussiana y 0.067 para la N-gaussiana, la P-RBF produce un aproximante y la N-RBF, un interpolante. Compárese con la figura 1.1.

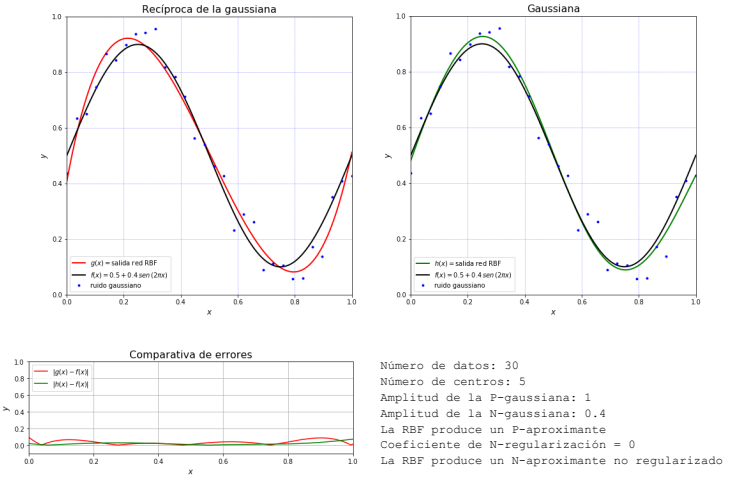

Figura 2.2. Con 30 datos, 5 centros, y amplitudes 1 para la P-gaussiana y 0.4 para la N-gaussiana, ambas RBFs producen una buena aproximación. Compárese con la figura 1.3.

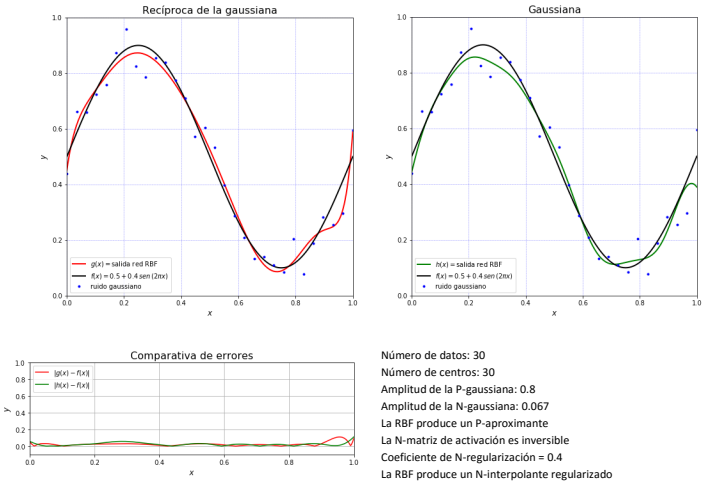

Figura 2.3. Con 30 datos, 30 centros, y amplitudes 0.8 para la P-gaussiana y 0.067 para la N-gaussiana, la N-RBF necesita una regularización con coeficiente 0.4 para conseguir un ajuste similar al de la P-RBF. Compárese con las figuras 1.6 y 2.1.

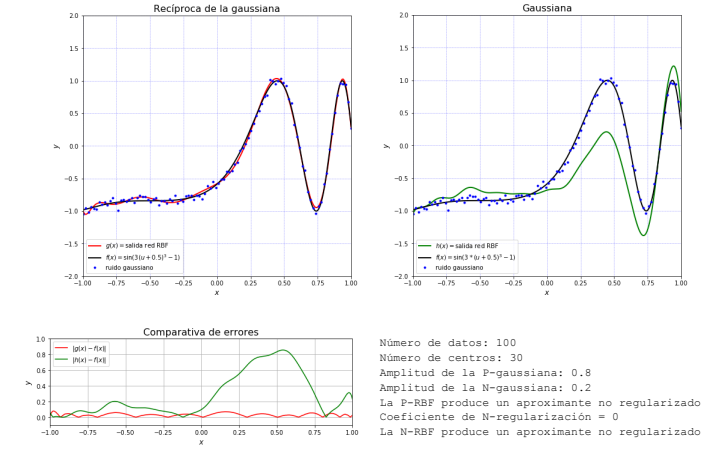

Figura 2.4. Aproximación con 100 datos de entrenamiento y 30 centros elegidos aleatoriamente, tomando como amplitudes 0.2 para la gaussiana y 0.8 para su recíproca. Se observa que esta última proporciona una aproximación mucho mejor que aquélla.

3

Aplicación a problemas de clasificación

3.1. Resolución de problemas de clasificación usando RBF

Se obtiene una visión más profunda de la naturaleza de las redes neuronales de funciones base radiales al considerar su uso en la resolución de diversos problemas de clasificación que surgen de forma natural. Supongamos que disponemos de un conjunto de datos, los cuales se pueden agrupar en tres clases diferentes (figura 3.1). Con otro tipo de redes multicapa sería posible separar las diferentes clases usando unidades ocultas que forman hiperplanos en el espacio de entrada, como se muestra en la figura 3.1(a). Un enfoque alternativo consiste en modelizar las clases separadas mediante funciones núcleo locales, tal como se aprecia en la figura 3.1(b). Este último tipo de representación es el asociado con las redes RBF.

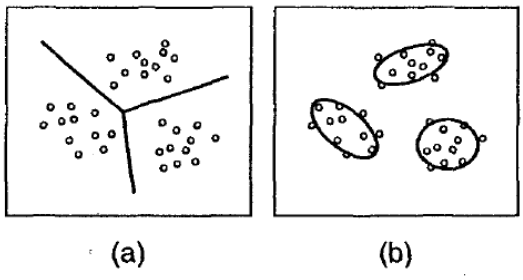

Figura 3.1. Esquema de datos bidimensionales separados en tres clases. En (a) se muestra la separación de las clases mediante hiperplanos. En (b) se observa un enfoque alternativo consistente en ajustar cada clase con una función núcleo, que se corresponde con el tipo de representación obtenido mediante redes RBF.

Supongamos que se modelizan los datos de cada clase $\mathcal{C}_k$ usando una única función núcleo, la cual escribimos como $p(\mathbf{x}|\mathcal{C}_k)$. El objetivo en un problema de clasificación es modelizar las probabilidades *a posteriori* $p(\mathcal{C}_k|\mathbf{x})$ para cada una de las clases. Tales probabilidades se pueden obtener a través del teorema de Bayes mediante las probabilidades *a priori* $p(\mathcal{C}_k)$, como sigue:

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})p(\mathcal{C}_{k'})}. \end{aligned} \tag{3.1}$$

Y se podrían interpretar como una forma simple de red RBF con funciones base normalizadas, dadas por

$$\phi_k(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})p(\mathcal{C}_{k'})},$$

donde las conexiones de la segunda capa consistirían en un peso desde cada unidad oculta a la correspondiente unidad de salida, con valor $p(\mathcal{C}_k)$. Las salidas de esta red representan, por tanto, aproximaciones a las probabilidades *a posteriori*.

En la práctica, en lugar de una única función núcleo (que no proporciona una buena representación de las distribuciones $p(\mathbf{x}|\mathcal{C}_k)$) se toma una cantidad M de ellas, etiquetadas por un índice j, para representar cada una de las densidades condicionadas a las clases. Así, escribimos

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_{j=1}^{M} p(\mathbf{x}|j)p(j|\mathcal{C}_k). \tag{3.2}$$

A partir de (3.2) es posible obtener una expresión para la densidad no condicionada $p(\mathbf{x})$ sin más que tomar la suma sobre todas las clases:

$$\begin{aligned} p(\mathbf{x}) &= \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) \\ &= \sum_{j=1}^{M} p(\mathbf{x}|j)p(j), \end{aligned} \tag{3.3}$$

siendo

$$p(j) = \sum_k p(j|\mathcal{C}_k)p(\mathcal{C}_k).$$

De nuevo, estamos interesados en las probabilidades *a posteriori* de pertenencia a cada clase. Éstas pueden ser calculadas sustituyendo las expresiones (3.2) y (3.3) en el teorema de Bayes (3.1) para obtener

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\sum_{j=1}^{M} p(j|\mathcal{C}_k)p(\mathbf{x}|j)p(\mathcal{C}_k)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')p(j')} \frac{p(j)}{p(j)} \tag{3.4}$$

$$= \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}), \tag{3.5}$$

donde en (3.4) se ha insertado un factor extra $1 = p(j)/p(j)$. La expresión (3.5) representa una red de funciones base radiales, en la cual las funciones base normalizadas vienen dadas por

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')p(j')} = p(j|\mathbf{x})$$

y los pesos de la segunda capa, por

$$w_{kj} = \frac{p(j|\mathcal{C}_k)p(\mathcal{C}_k)}{p(j)} = p(\mathcal{C}_k|j).$$

Así, las activaciones de las funciones base pueden ser interpretadas como las probabilidades *a posteriori* de la presencia de las correspondientes características en los datos de entrada, y los pesos pueden ser similarmente contemplados como las probabilidades *a posteriori* de pertenencia a una clase determinada, dada la presencia de esas características.

3.2. Ejemplo: decisión de símbolos enviados en un sistema de comunicación

Un sistema de comunicación está formado básicamente por tres elementos: emisor, canal y receptor. Si el sistema es de tiempo continuo, la transmisión consiste en muestrear la señal a transmitir, cuantificarla y codificarla. La señal codificada se envía por el canal, y al llegar al receptor se decodifica y se debería obtener el símbolo transmitido. En la práctica esto no sucede, ya que el canal introduce ruido y la decodificación nunca es perfecta, lo que obliga a situar a la salida del decodificador un nuevo elemento, que llamaremos selector, encargado de decidir qué símbolo fue realmente enviado.

Se pretende en este ejemplo crear un selector que sea una red RBF.

3.2.1. Planteamiento del problema en términos de una red RBF

Con el fin de que tenga una sola salida, consideraremos un alfabeto binario. Supongamos que los símbolos susceptibles de ser introducidos en el codificador son dos vectores $\mu_1, \mu_2 \in \mathbb{R}^2$, es decir, que nuestro alfabeto está formado por el

conjunto $\mathcal{A} = \{\mu_1, \mu_2\} \subset \mathbb{R}^2$. La salida del decodificador será un símbolo $\alpha(i)$ contaminado por un ruido $\eta(i)$, que supondremos aditivo. Esta salida será una entrada $\mathbf{x}(i)$ de la red RBF, de manera que $\mathbf{x}(i) = \alpha(i) + \eta(i)$. La salida y que se desea obtener en la red es:

$$y = \begin{cases} 1, & \text{si } \alpha(i) = \mu_1 \\ 0, & \text{si } \alpha(i) = \mu_2. \end{cases}$$

Denotaremos por H_1 al suceso $\{\alpha = \mu_1\}$ y H_2 al suceso $\{\alpha = \mu_2\}$ y llamaremos p_1, p_2 a las probabilidades de que sucedan H_1 y H_2, respectivamente. Puesto que el alfabeto es binario, necesariamente $p_1 = 1 - p_2$. Asumiremos una distribución gaussiana para el ruido, de media nula y amplitudes σ_1 y σ_2, respectivamente. La distribución de $\mathbf{x}$ condicionada al símbolo enviado se puede expresar entonces en la forma

$$f_{\mathbf{x}}(\mathbf{x}|H_1) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{\|\mathbf{x} - \mu_1\|^2}{2\sigma_1^2}\right), \tag{3.6}$$

$$f_{\mathbf{x}}(\mathbf{x}|H_2) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{\|\mathbf{x} - \mu_2\|^2}{2\sigma_2^2}\right). \tag{3.7}$$

Se pretende minimizar la probabilidad de error P_e del selector. Para ello, establecemos la siguiente regla de decisión:

$$\begin{cases} \text{decidir } H_1 \text{ si } \mathbf{x} \in \mathcal{Z}_1 \\ \text{decidir } H_2 \text{ si } \mathbf{x} \in \mathcal{Z}_2, \end{cases}$$

siendo $\mathcal{Z}_1$ y $\mathcal{Z}_2$ dos regiones planas disjuntas cuya unión es todo $\mathbb{R}^2$. La probabilidad de error valdrá:

$$\begin{aligned} P_e &= p_1 \int_{\mathcal{Z}_2} f_{\mathbf{x}}(\mathbf{x}|H_1) d\mathbf{x} + p_2 \int_{\mathcal{Z}_1} f_{\mathbf{x}}(\mathbf{x}|H_2) d\mathbf{x} \\ &= p_1 \Big(1 - \int_{\mathcal{Z}_1} f_{\mathbf{x}}(\mathbf{x}|H_1) d\mathbf{x}\Big) + p_2 \int_{\mathcal{Z}_1} f_{\mathbf{x}}(\mathbf{x}|H_2) d\mathbf{x} \\ &= p_1 + \int_{\mathcal{Z}_1} \left[p_2 f_{\mathbf{x}}(\mathbf{x}|H_2) - p_1 f_{\mathbf{x}}(\mathbf{x}|H_1)\right] d\mathbf{x}. \end{aligned} \tag{3.8}$$

La forma óptima de tomar la decisión será la que minimice P_e, es decir, que $\mathcal{Z}_1$ esté formada por todos aquellos puntos $\mathbf{x}$ tales que

$$p_2 f_{\mathbf{x}}(\mathbf{x}|H_2) - p_1 f_{\mathbf{x}}(\mathbf{x}|H_1) < 0.$$

Dicho de otra forma, la decisión óptima será:

$$\begin{cases} \text{decidir } H_1 \text{ si } \dfrac{f_{\mathbf{x}}(\mathbf{x}|H_2)}{f_{\mathbf{x}}(\mathbf{x}|H_1)} < \dfrac{p_1}{p_2}, \\ \text{decidir } H_2 \text{ si } \dfrac{f_{\mathbf{x}}(\mathbf{x}|H_2)}{f_{\mathbf{x}}(\mathbf{x}|H_1)} > \dfrac{p_1}{p_2}. \end{cases}$$

Utilizamos las ecuaciones (3.6) y (3.7) para encontrar la frontera de decisión, que estará formada por los puntos $\mathbf{x}$ del plano tales que

$$\exp\left(-\frac{\|\mathbf{x}-\mu_1\|^2}{2\sigma_1^2}+\frac{\|\mathbf{x}-\mu_2\|^2}{2\sigma_2^2}\right)=\frac{p_1}{p_2}\frac{\sigma_1^2}{\sigma_2^2}.$$

Tras un cálculo sencillo llegamos a la expresión

$$\left\|\mathbf{x}-\frac{\sigma_1^2\mu_2-\sigma_2^2\mu_1}{\sigma_1^2-\sigma_2^2}\right\|^2=\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2-\sigma_2^2}\left[\frac{\|\mu_1-\mu_2\|^2}{\sigma_1^2-\sigma_2^2}+2\ln\left(\frac{p_1}{p_2}\frac{\sigma_1^2}{\sigma_2^2}\right)\right],$$

indicativa de que la frontera de decisión es una circunferencia $\mathcal{C}$ de centro $\mathbf{x}_c$ y radio r, donde

$$\mathbf{x}_c=\frac{\sigma_1^2\mu_2-\sigma_2^2\mu_1}{\sigma_1^2-\sigma_2^2},\quad r^2=\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2-\sigma_2^2}\left[\frac{\|\mu_1-\mu_2\|^2}{\sigma_1^2-\sigma_2^2}+2\ln\left(\frac{p_1}{p_2}\frac{\sigma_1^2}{\sigma_2^2}\right)\right].$$

Así pues, hemos hallado de forma teórica la región de decisión óptima. La diferencia entre este análisis teórico y el que se realiza con redes RBF reside en que para poder decidir de forma óptima ha sido necesario conocer todo sobre la distribución de los símbolos y su ruido, lo que, a su vez, obliga a conocer a la perfección el canal y la forma de codificación, mientras que trabajando con redes RBF no es necesario conocer nada más que un conjunto de entradas y salidas.

3.2.2. Elección de valores para la simulación

Notación	Valor	Descripción
μ_1	$(0,0)^\top$	símbolo del alfabeto $\mathcal{A}$
μ_2	$(2,0)^\top$	símbolo del alfabeto $\mathcal{A}$
σ_1^2	1	varianza del ruido que afecta a μ_1
σ_2^2	4	varianza del ruido que afecta a μ_2
p_1	0,5	probabilidad de aparición del símbolo μ_1
p_2	0,5	probabilidad de aparición del símbolo μ_2

Tabla 3.1. Valores considerados en el problema.

Usamos los valores de la tabla 3.1 para calcular el centro $\mathbf{x}_c$ y el radio r de la frontera de decisión:

$$\mathbf{x}_c = \Big(-\frac{2}{3}, 0 \Big)^\top, \quad r \simeq 2{,}34.$$

También se puede calcular teóricamente la probabilidad de error que se obtendría en caso de que el selector decida de forma óptima. Resolviendo numéricamente la integral (3.8) resulta que $P_e = 0{,}1849$; en consecuencia, la probabilidad de que decida correctamente es $P_c = 0{,}8151$.

3.2.3. Simulación numérica

Para finalizar, presentaremos el resultado de una simulación correspondiente a la situación descrita. Según lo que acabamos de obtener, una cota superior para el porcentaje de aciertos es el 81,51 %.

Dada la sencillez del planteamiento y las restricciones impuestas por la normativa académica sobre la extensión de la memoria, se ha omitido el código utilizado para implementar la simulación. La red se ha entrenado a partir de una muestra de $N = 500$ datos generada aleatoriamente, tomando como centros un subconjunto aleatorio (cf. sección 1.5.1) de $M < N$ de esos datos y calculando los pesos mediante la matriz pseudoinversa. En su arquitectura se han usado como funciones de activación gaussianas de igual amplitud σ, obtenida según la fórmula

$$\sigma = \frac{d_{max}}{\sqrt{2M}},$$

donde d_{max} es el diámetro del conjunto de los centros. En la figura 3.2 se puede ver el resultado de la clasificación de un conjunto de 1000 puntos en el cuadrado $[-1, 3] \times [-2, 2]$.

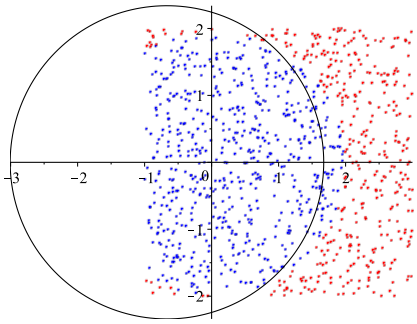

Figura 3.2. Resultado de la clasificación. Se aprecia cómo los puntos correspondientes a las dos asignaciones quedan separados por la circunferencia, que se corresponde con la frontera de decisión.

Bibliografía

[1] C.M. Bishop: *Neural networks for pattern recognition*. Clarendon Press, 1995.

[2] M.D. Buhmann: *Radial Basis Functions: Theory and implementations*. Cambridge University Press, 2003.

[3] T. Chen, H. Chen: Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks* **6** (1995), no. 4, 911–917.

[4] S. Chen, C.F.N. Cowan, P.M. Grant: Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks* **2** (1991), no. 2, 302–309.

[5] C.A. Cruz-Rodríguez, C. García-Vargas, I. Marrero: Further comments on «Relaxed conditions for radial-basis function networks to be universal approximators». *Revista de la Academia Canaria de Ciencias* **27** (2015/2016), 29–31.

[6] G. Cybenko: Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems* **3** (1989), 303–314.

[7] G. Fasshauer: *Meshfree approximation methods with MATLAB.* World Scientific, 2007.

[8] A. Friedman: *Generalized functions and partial differential equations*. Prentice-Hall, 1963.

[9] F. Girosi, T. Poggio: Networks and the best approximation property. *Biological Cybernetics* **63** (1990), 169–176.

[10] S. Haykin: *Neural networks*, 2nd ed. Pearson Prentice Hall, 1999.

[11] K. Hornik: Approximation capabilities of multilayer feedforward neural networks. *Neural Networks* **4** (1990), 251–257.

[12] K. Hornik: Some new results on neural network approximation. *Neural Networks* **6** (1993), 1069–1072.
[13] T. Kohonen: Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **43** (1982), 59–69.
[14] S. Lafon: *Redes neuronales*. Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Uruguay, 2002. Disponible en http://iie.fing.edu.uy/ense/asign/tes/materiales/monografias/RedesNeuronales.pdf.
[15] S. de León Pérez: *Aproximación universal por modelos computacionales ridge y redes neuronales*. Trabajo Fin de Grado en Matemáticas, Universidad de La Laguna, septiembre 2014.
[16] M. Leshno, V. Lin, A. Pinkus, S. Schocken: Multilayer feedforward networks with a polynomial activation function can approximate any function. *Neural Networks* **6** (1993), 861–867.
[17] Y. Liao, S.-c. Fang, H.L.W. Nuttle: Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks* **16** (2003), 1019–1028.
[18] H. Mhaskar, C. Micchelli: Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics* **13** (1992), 350–373.
[19] J. Moody, C.J. Darken: Fast learning in networks of locally-tuned processing units. *Neural Computation* **1** (1989), no. 2, 281–294.
[20] J. Park, I.W. Sandberg: Universal approximation using radial-basis-function networks. *Neural Computation* **3** (1991), no. 2, 246–257.
[21] J. Park, I.W. Sandberg: Approximation and radial-basis-function networks. *Neural Computation* **5** (1993), 305–316.
[22] W. Rudin: *Real and complex analysis*, 3rd ed. McGraw-Hill, 1987.
[23] W. Rudin: *Functional analysis*, 2nd ed. McGraw-Hill, 1991.
[24] H. Wendland: *Scattered data approximation*. Cambridge University Press, 2005.
[25] W. Wu, D. Nan, J.-l. Long, Y.-m. Ma: A comment on «Relaxed conditions for radial-basis function networks to be universal approximators». *Neural Networks* **21** (2008), 1464–1465.

Apéndice

Se reproduce a continuación el código Python utilizado en los capítulos 1 y 2 para aproximar e interpolar mediante redes neuronales RBF con funciones base gaussianas y recíprocas de gaussianas. Está basado en un *snippet* de T. Rueckstiess [*]. El entrenamiento se hace eligiendo aleatoriamente los centros entre los datos de entrada y calculando los pesos mediante la matriz pseudoinversa. En el caso de funciones de activación gaussianas, se contempla la posibilidad de aplicar una regularización.

```
#! /usr/bin/python

from scipy import *
from scipy.linalg import norm, pinv, inv, solve
from numpy.linalg import matrix_rank

from matplotlib import pylab as plt

class RBF:

    def __init__(self, indim, numData, numCenters,
        outdim):
         self.indim = indim
         self.outdim = outdim
         self.numData = numData
         self.numCenters = numCenters
         self.betap = 0.5/float(input('Amplitud de la P-
             gaussiana: '))**2
```

[*] http://www.rueckstiess.net/research/snippets/show/72d2363e.

```
        self.betan = 0.5/float(input('Amplitud de la N-
            gaussiana: '))**2
        self.centers = [random.uniform(0, 1, indim) for
             i in range(numCenters)]
        self.W = random.random((self.numCenters, self.
            outdim))

    def _basisfuncGp(self, c, d):
        assert len(d) == self.indim
        return exp(self.betap * norm(c-d)**2)

    def _basisfuncGn(self, c, d):
        assert len(d) == self.indim
        return exp(-self.betan * norm(c-d)**2)

    def _calcAct(self, X, vc):
        # cálculo de las RBFs de activación
        # 'vc' determina la elección de la función base
             gaussiana con exponente (P)ositivo o (N)
            egativo
        G = zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                if vc == "P":
                    G[xi,ci] = self._basisfuncGp(c, x)
                else:
                    G[xi,ci] = self._basisfuncGn(c, x)
        # print(vc + '-matriz de activación: ', G)
        return G

    def train(self, X, Y, numData, numCenters, vc):
        """ X: matriz de dimensiones n * indim
            y: vector columna de dimensiones n * 1 """
        # entrenamiento de la red

        # print(vc + '-datos de entrada: ', X)

        # centros: (subconjunto aleatorio de los) datos
             de entrada
        if numData == numCenters:
            self.centers = X
        else:
```

```
            rnd_idx = random.permutation(X.shape[0])[:
                self.numCenters]
            self.centers = [X[i,:] for i in rnd_idx]
        # print(vc + '-centros: ', self.centers)

        # pesos, sin/con regularización'
        G = self._calcAct(X, vc)
        tipo = 'La ' + vc + '-RBF produce un'
        if matrix_rank(G) == G.shape[0]:
            print ('La ' + vc + '-matriz de activación
                es inversible')
            tipo = tipo + ' interpolante'
        else:
            tipo = tipo + ' aproximante'
        if vc == "P":
            self.W = dot(pinv(G), Y)
            tipo = tipo + ' no'
        else:
            FR = float(input('Coeficiente de ' + vc +'-
                regularización = '))
            if FR == 0:
                self.W = dot(pinv(G), Y)
                tipo = tipo + ' no'
            GR = dot(transpose(G), G)
            n_col = GR.shape[0]
            self.W = dot(dot(inv(GR + FR*eye(n_col)),
                transpose(G)), Y)
        tipo = tipo + ' regularizado'
        print(tipo)
        # print(vc + '-pesos: ', self.W)
        return self.W

    def test(self, X, vc):
        """ X: matriz de dimensiones n * indim """
        # salida de la red RBF
        G = self._calcAct(X, vc)
        Y = dot(G, self.W)
        # print(vc + '-salida: ', Y)
        return Y

if __name__ == '__main__':

    # ##### Ejemplo 1D #####
```

```
# d = int(input('Densidad gráfica: '))
d = 300
n = int(input('Número de datos: '))
m = int(input('Número de centros: '))
# epsilon = float(input('Amplitud del ruido
   gaussiano: '))
epsilon = 0.05

# se crea una matriz d*1 de d puntos equiespaciados
    entre 0 y 1, ambos inclusive, para representar
    gráficamente f
u = mgrid[0:1:complex(0,d)].reshape(d, 1)
x = mgrid[0:1:complex(0,n)].reshape(n, 1) # idem de
    n puntos, que serán los datos de entrada
# w = sin (3*(x+0.5)**3 - 1)
v = 0.5 + 0.4 * sin(2*pi*u) # ordenadas de la
   gráfica de f
w = 0.5 + 0.4 * sin(2*pi*x) # datos objetivo
y = w + random.normal(0, epsilon, w.shape) # se
   añade ruido gaussiano a los datos objetivo

# regresión RBF
rbf = RBF(1, n, m, 1) # entrada y salida
   unidimensionales, con n datos y m centros
rbf.train(x, y, n, m, "P") # entrenamiento de la P-
   RBF con los datos de entrada y los datos
   objetivo con ruido
zP = rbf.test(u, "P") # salida de la P-RBF
rbf.train(x, y, n, m, "N") # entrenamiento de la N-
   RBF con los datos de entrada y los datos
   objetivo con ruido
zN = rbf.test(u, "N") # salida de la N-RBF

# gráficas -----
plt.figure(figsize = (8, 8))
plt.plot(u, zP, 'r', linewidth = 2, label = r'$g(x)
    = $' + 'salida red RBF')
plt.plot(u, v, 'k-', linewidth = 2, label = r'$f(x)
    = 0.5 + 0.4\,sen\,(2\pi x)$')
plt.plot(x, y, 'b.', label = 'ruido gaussiano')
plt.legend(loc = 3)
plt.xlabel(r'$x$', fontsize = 12)
```

```
plt.ylabel(r'$y$', fontsize = 12)
plt.title('Recíproca de la gaussiana', fontsize=16)
plt.grid(color = 'b', alpha = 0.5, linestyle = '
   dashed', linewidth = 0.5)
plt.xlim(0.0, 1.0)
plt.ylim(0.0, 1.0)

plt.figure(figsize=(8, 8))
plt.plot(u, zN, 'g', linewidth = 2, label = r'$h(x)
    = $' + 'salida red RBF')
plt.plot(u, v, 'k-', linewidth = 2, label = r'$f(x)
    = 0.5 + 0.4\,sen\,(2\pi x)$')
plt.plot(x, y, 'b.', label = 'ruido gaussiano')
plt.legend(loc = 3)
plt.xlabel(r'$x$', fontsize = 12)
plt.ylabel(r'$y$', fontsize = 12)
plt.title('Gaussiana', fontsize=16)
plt.grid(color = 'b', alpha = 0.5, linestyle = '
   dashed', linewidth = 0.5)
plt.xlim(0.0, 1.0)
plt.ylim(0.0, 1.0)

plt.figure(figsize = (8, 3))
plt.plot(u, abs(v-zP), 'r-', label = r'$|g(x)-f(x)|
   $')
plt.plot(u, abs(v-zN), 'g-', label = r'$|h(x)-f(x)|
   $')
plt.legend(loc = 2)
plt.xlabel(r'$x$', fontsize = 12)
plt.ylabel(r'$y$', fontsize = 12)
plt.title('Comparativa de errores', fontsize = 16)
plt.grid(True)
plt.xlim(0.0, 1.0)
plt.ylim(-0.1, 1.0)

plt.tight_layout()
plt.show()
```

Radial basis functions neural networks

Claudio García Vargas
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100305241@ull.edu.es

Abstract

In approximation theory, conditionally positive definite radial functions, or radial basis functions (RBF), are used to solve problems of interpolation of sparse data in Euclidean space. Among the many byproducts of RBF interpolation, the RBF neural networks are particularly interesting. The purpose of this report is to study, by means of functional-analytic techniques, the properties of interpolation and approximation by RBF neural networks in spaces of continuous and of integrable functions, illustrating the theory with some numerical experiments and practical applications.

1. Introduction

An artificial neural network is an information processing system whose performance is inspired by that of biological neural networks. Originally, artificial neural networks intended to model the operation of these. With the course of time, artificial models have emerged that lack any biological signification but have proved themselves to be useful for solving information processing problems.
The result of the processing occurring in a neuron is a non-linear function of the inputs and of a set of parameters. This point constitutes the basis of the operation of neural networks, because the set of parameters on which such functions depend are adjusted according to what they are learning. After training, with a good algorithm and a sufficiently good sample, a neural network is capable to perform the task it has been trained for with a high degree of accuracy.
The learning of neural networks can occur in two ways: supervised or unsupervised. In supervised mode, learning is achieved directly by comparing the output of the network with the correct answer already known. In unsupervised mode, the available information is only in correlation with the input or signal data. Artificial neural networks can also be classified by their architecture in regressive or progressive networks, according to whether or not they allow feedback among layers.

2. RBF neural networks

The purpose of this work is to give a general overview of radial basis functions (RBF) neural networks and study their interpolation and approximation properties. These are a kind of progressive neural networks of the form

$$h(\mathbf{x}) = \sum_n w_n \phi(\|\mathbf{x} - \mathbf{x}^n\|),$$

where ϕ is a basis function, the vectors $\mathbf{x}^n$ are called centers, and the scalars w_n are weights. Initially they were introduced to solve exact interpolation, but nowadays they constitute a field of independent interest within computer science and artificial intelligence, with manifold applications to areas as diverse as finance, medicine, biology, geology, engineering or physics. Besides exact interpolation, some of the topics treated are network training, regularization theory and optimization of basis functions (including data subsets, clustering algorithms, orthogonal least squares and gaussian mixed models), as well as supervised training.

3. Universal approximation by RBF neural networks

In this chapter, following [5] (cf. also [6] and [2]) we study the universal approximation property of three-layered radial basis function (RBF) networks. We show that the integrability condition usually imposed on the activation function ϕ can be dropped. Instead, the following holds.

Theorem 3.1 *Let σ be a function from $\mathbb{R}^n$ to $\mathbb{R}$. If σ is continuous almost everywhere, locally essentially bounded, and not a polynomial then, for any compact set $K \subset \mathbb{R}^n$, $\Sigma = \operatorname{span}\{\sigma(ax+\theta) : a \in \mathbb{R}, \theta \in \mathbb{R}^n\}$ is uniformly dense in $C(K)$, that is, given any $f \in C(K)$ and any $\varepsilon > 0$, there exists $g \in \Sigma$ such that $\|f - g\|_{L^\infty(K)} \leq \varepsilon$ for all $x \in K$.*

Theorem 3.2 *Let μ be a finite measure on $\mathbb{R}^n$ and σ a function from $\mathbb{R}^n$ to $\mathbb{R}$. If $\sigma \in L^\infty(\mu)$ is not a polynomial, then $\Sigma = \operatorname{span}\{\sigma(ax+\theta) : a \in \mathbb{R}, \theta \in \mathbb{R}^n\}$ is dense en $L^p(\mu)$, for all $1 \leq p < \infty$.*

These results are illustrated by the so-called P-gaussian and N-gaussian functions, respectively given by

$$F(x) = \exp\left(\frac{\|x-\theta\|^2}{b}\right),$$

$$G(x) = \exp\left(-\frac{\|x-\theta\|^2}{b}\right).$$

Here, $x \in \mathbb{R}^n$ is the variable, $\theta \in \mathbb{R}^n$ is a center vector, and $b \in \mathbb{R}$ is a width parameter. Although the N-gaussian G is integrable while the P-gaussian F is not, F satisfies the hypotheses of the above theorems. In fact, figures 1 and 2 show that a RBF with P-gaussians as activation functions can perform a better approximation than a N-gaussian based RBF.

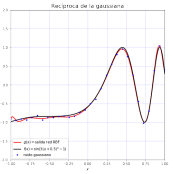

Figure 1: *Approximation by a P-gaussian RBF.*

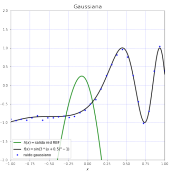

Figure 2: *Approximation by a N-gaussian RBF.*

4. Application to classification problems

In this section, a probabilistic approach to RBF networks is given, and an application of this approach to symbol classification in a communication channel is discussed.

References

[1] C.M. Bishop: *Neural networks for pattern recognition*. Clarendon Press, 1995.

[2] C.A. Cruz-Rodríguez, C. García-Vargas, I. Marrero: Further comments on «Relaxed conditions for radial-basis function networks to be universal approximators». *Revista de la Academia Canaria de Ciencias* **27** (2015/2016), 29–31.

[3] G. Fasshauer: *Meshfree approximation methods with MATLAB*. World Scientific, 2007.

[4] S. de León Pérez: *Aproximación universal por modelos computacionales ridge y redes neuronales*. Trabajo Fin de Grado en Matemáticas, Universidad de La Laguna, septiembre 2014.

[5] Y. Liao, S.-c. Fang, H.L.W. Nuttle: Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks* **16** (2003), 1019–1028.

[6] W. Wu, D. Nan, J.-l. Long, Y.-m. Ma: A comment on «Relaxed conditions for radial-basis function networks to be universal approximators». *Neural Networks* **21** (2008), 1464–1465.