Universidad Nacional de Ingeniería

Facultad de Ingeniería Mecánica

TESIS

Desarrollo de una Red Neuronal Artificial para identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo de la calidad del agua

Para obtener el Título profesional de:

Ingeniero Mecatrónico

Elaborado por:

David Benjamín Fernández Villanueva

0009-0004-4378-4586

Asesor

Dr. Ricardo Raúl Rodríguez Bustinza

0000-0002-6411-7123

Lima – Perú

2024

Dedicatoria

Dedicado con mucho aprecio y cariño a mis padres y hermanos por haberme forjado en el camino de la rectitud y apoyado en todo momento.

Agradecimientos

Gracias a mis familiares, amigos y todas las personas que me rodean por compartir sus experiencias y momentos de la vida.

Gracias al Ing. Rodríguez Bustinza Ricardo por el apoyo durante mi etapa universitaria y el apoyo a la elaboración de este trabajo. Gracias a todos mis maestros de la escuela, colegio y universidad por haberme transmitido su conocimiento intelectual y haber compartido sus experiencias.

RESUMEN

En este trabajo se identifica de manera satisfactoria la curva característica de un sensor de pH mediante el entrenamiento y validación de una Red Neuronal Artificial.

Esta afirmación resulta de la aceptación de la hipótesis general "El Desarrollo de una Red Neuronal Artificial permite identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo de la calidad del agua". Esto se logró utilizando una metodología experimental, en la que se utilizó una arquitectura perceptron con 6 entradas, 35 neuronas en la capa oculta, y 1 neurona en la capa de salida, se inicializaron los pesos utilizando el método Widrow y Nguyen y para el entrenamiento se utilizó el algoritmo general de regularización Bayesiana. Se aceptó cada una de las hipótesis específicas debido a los resultados óptimos del Capítulo V "Análisis de resultados" con respecto a la suma de errores cuadráticos, número efectivo de parámetros, regresión lineal de los objetivos y salidas de la RNA, coeficiente de correlación, histograma de errores, la comparación de las medidas del grado de pH de tres sustancias diferentes utilizando el módulo electrónico desarrollado con la RNA entrenada y un dispositivo electrónico de medición (de fábrica) fueron valores similares con un error máximo de 0.057pH.

Finalizamos este trabajo realizando las recomendaciones como un punto de partida para ampliar este trabajo.

ABSTRACT

In this work, the characteristic curve of a pH sensor is satisfactorily identified through the training and validation of an Artificial Neural Network.

This statement results from the acceptance of the general hypothesis "The Development of an Artificial Neural Network allows the identification of the Characteristic Curve of a Hydrogen PH Potential Sensor in a water quality monitoring company." This was achieved using an experimental methodology, in which a perceptron architecture was used with 6 inputs, 35 neurons in the hidden layer, and 1 neuron in the output layer, the weights were initialized using the Widrow and Nguyen method and for training The general Bayesian regularization algorithm was used. Each of the specific hypotheses was accepted due to the optimal results of Chapter V "Analysis of results" with respect to the sum of squared errors, effective number of parameters, linear regression of the objectives and outputs of the ANN, correlation coefficient, histogram of errors, the comparison of the measurements of the pH degree of three different substances using the electronic module developed with the trained ANN and an electronic measuring device (factory) were similar values with a maximum error of 0.057pH.

We conclude this work by making recommendations as a starting point to expand this work.

ÍNDICE DE CONTENIDO

Índice de Figuras

Índice de tablas

Definición de términos

Potencial de hidrógeno, pH	:	La concentración de hidrogeniones se expresa por el valor de pH, que.
Vector	:	Hace referencia a una columna de números.
Vector fila	:	Hace referencia a una fila de una matriz, usado como vector (columna).
Algoritmo de aprendizaje	:	Forma en la que pueda aprender una red neuronal artificial, y puede ser supervisados (entrada, salida, objetivo), no supervisado (entrada, salida) y por reforzamiento (recompensa/castigo).
Calidad del agua	:	El término calidad del agua se usa para describir la condición del agua, incluidas sus características químicas, físicas y biológicas, generalmente con respecto a su idoneidad para un propósito particular (es decir, beber, nadar o pescar), [1].
Análisis de componentes principales - PCA	:	El método de componentes principales tiene por objeto transformar un conjunto de variables, a las que se denomina *originales*, en un nuevo conjunto de variables denominadas *componentes principales*. Estas últimas se caracterizan por estar incorrelacionadas entre sí y, además, pueden ordenarse de acuerdo con la información que llevan incorporada. [2]
Kernel PCA	:	Método para realizar una forma no lineal de análisis de componentes principales. [3]
Optimizar	:	Minimización de la complejidad de la red neuronal artificial y de los cálculos que pueda realizar el computador.
Vehículos conectados	:	Se refiere a aplicaciones, servicios y tecnologías que conectan un vehículo a su entorno. Un vehículo conectado incluye los diferentes dispositivos de comunicación (integrados o portátiles) presentes en el vehículo, que permiten la conectividad en el automóvil con otros dispositivos presentes en el vehículo y / o permiten la conexión del vehículo a dispositivos, redes, aplicaciones y servicios externos, [4].
Occam's razor	:	Principio de parsimonia, lo más simple es lo mejor.

Notación, Nomenclatura y Siglas

Notación

En este trabajo se utiliza una notación estándar con el fin de realizar una lectura clara y sencilla sin menoscabar el rigor científico.

En la igualdades y ecuaciones matemáticas en texto y las figuras se usa la siguiente notación:

Escalares	—	letras pequeñas en cursivas: a,b,c
Vectores	—	letras pequeñas en negro, no cursivas: **a**, **b**, **c**
Matrices	—	letras mayúsculas en negro, no cursivas: **A**, **B**, **C**

Nomenclatura

$\mathbf{A}^{\mathrm{T}}$	—	Matriz transpuesta
$\mathbf{W}^{k}$	—	Matriz de pesos
$w_i^{k,j}(t)$	—	Elemento escalar de la matriz de pesos
$\mathbf{w}_i^{k}(t)$	—	Vector columna de la matriz de pesos
${}_i\mathbf{w}^{k}(t)$	—	Vector fila de la matriz de pesos
$\mathbf{b}^{k}(t)$	—	Vector bias
$b_i^{k}(t)$	—	Elemento escalar del vector bias
$\mathbf{p}(t)$	—	Vector de entrada
$p_i(t)$	—	Elemento del vector de entrada
$\mathbf{p}_q$	—	Conjunto de datos de vectores de entrada
$\mathbf{a}^{k}(t)\ o\ \mathbf{a}_q^{k}$	—	Vector de salida
$a_i^{k}(t)\ o\ a_{i,q}^{k}$	—	Elemento del vector de salida
$\mathbf{f}^{k}(\mathbf{n}^{k})$	—	Vector de la función de transferencia

$f^k(n_i^k)$	—	Elemento del vector de la función de transferencia
$\mathbf{t}(t)\ o\ \mathbf{t}_q$	—	Vector targets u objetivos
$t_i(t)\ o\ t_{i,q}$	—	Elemento del vector de objetivos
$\mathbf{e}(t)\ o\ \mathbf{e}_q$	—	Vector de errores
$e_i(t)\ o\ e_{i,q}$	—	Elemento escalar del vector de errores

Siglas y abreviaturas

SI	—	Sistema Internacional de Unidades
RNA	—	Red neuronal artificial.
pH	—	Potencial de hidrogeno (actividad del ion hidrógeno)
DAQ	—	Adquisición de datos.
SDK	—	Kit de desarrollo de software.
GNBR	—	Regularización Bayesiana con aproximación de Gauss-Newton.
VAC	—	Voltaje corriente directa.
EKG	—	Electrocardiograma.
GND	—	Punto de referencia en un circuito eléctrico desde.

INTRUDUCCIÓN

El cliente es el verdadero protagonista de las empresas. Adaptarse a sus necesidades se ha convertido, por lo tanto, en algo fundamental. WAPOSAT, es una empresa que se dedica al monitoreo de la calidad del agua y para ello hace uso de sensores y hardware predefinido de terceros. Esta dependencia de terceros con respecto al hardware limita las soluciones a las necesidades de sus clientes, por lo que les es crítico encontrar una solución a esta limitación de hardware. El primer paso para desarrollar su propio hardware es identificar la curva característica de los diferentes sensores que utiliza.

En este trabajo, se pretende identificar la curva característica de uno sensor de pH mediante redes neuronales artificiales debido a que el valor del pH depende de 6 variables de entrada lo que lo difícilmente se puede realizar con otros métodos.

Para este fin se hace uso de la capacidad de aprendizaje, generalización, tolerancia a fallos y facilidad de implementación de las redes neuronales artificiales en microcontroladores.

En orden de desarrollo de este trabajo es:

El Capítulo I presenta los antecedentes; motivación/justificación, metodología de trabajo, alcances, limitaciones. Planteamiento del problema; enunciado y formulación del problema, situación actual. Objetivos; objetivo general y específicos. Hipótesis y operacionalización de variables; hipótesis general, hipótesis específicas, variables y operacionalización de variables.

El Capítulo II presenta el marco teórico, en ella se presentan diferentes conceptos y teorías que servirán como referencia en el desarrollo del problema de este trabajo. Las principales teorías son Redes Neuronales artificiales, Modelo neuronal, Arquitectura de Redes Neuronales Artificiales, Percetron multicapa, Preprocesamientos de datos, Generalización de entrenamiento de Redes Neuronales Artificiales.

En capítulo III presenta la Metodología de la investigación, tipo y diseño de la investigación, alcances y limitaciones.

El capítulo IV presenta el desarrollo del problema, en ella se presenta la solución al problema, el plan de acción y el desarrollo detallado del problema en donde se realiza el pre-entrenamiento, el entrenamiento de la red neuronal artificial.

En el capítulo IV se presenta el análisis de resultados, post-entrenamiento de la red neuronal artificial. Aquí se hace un análisis detallado de los resultados, para poder obtener las conclusiones de este trabajo.

Luego de analizar detalladamente los resultados se presenta las Conclusiones. En ella se hace una conclusión exhaustiva y crítica de las hipótesis planteadas en el Capítulo I. Adicionalmente se hace las recomendaciones pertinentes, y se presenta la bibliografía consultada en este trabajo.

Finalmente se presenta los diferentes anexos. Estos anexos nos permiten entender mejor este trabajo, sin embargo, para mantener el orden y buena lectura de este trabajo se ha creído pertinente presentarlo de manera aparte, en los anexos.

Se agradece a la empresa WAPOSAT S.A.C. por haber facilitado el uso del hardware actual con el que trabaja.

CAPITULO I

PARTE INTRUDUCTORIA DEL TRABAJO

1.1 Antecedentes

WAPOSAT S.A.C.[1], al dedicarse al monitoreo de la calidad del agua, hace uso necesariamente de sensores y sus respectivas tarjetas electrónicas para adquirir, acondicionar, tratar e interpretar correctamente las señales de los parámetros que se desean medir, estos sensores son adquiridos a la empresa LIBELIUM[2] y otros terceros. Sin embargo, la empresa LIBELIUM, quien suministra estos sensores, no proporciona las curvas características de cada sensor de manera explícita, sino ya programadas dentro de microcontrolador asociado a dicho sensor.

WAPOSAT desarrolla software e integra hardware [3] de terceros para el monitoreo de la calidad del agua. Esta dependencia de terceros con respecto al hardware limita las soluciones a las necesidades de sus clientes, por lo que les es crítico encontrar una solución a esta limitación de hardware.

1.1.1 Motivación/justificación

La gerencia técnica comercial de WAPOSAT llegó a la conclusión de que necesitan desarrollar su propio hardware para monitoreo de la calidad del agua. Sin embargo, los

[1] WAPOSAT: Water Pollution Satellite.
[2] LIBELIUM es una multinacional tecnológica española, fundada en 2006. Diseña y fabrica hardware y un kit completo de desarrollo de software (SDK) para redes de sensores inalámbricos para que integradores de sistemas ingeniería y consultorías puedan ofrecer soluciones confiables de Internet de Cosas (IoT), M2M y Smart Cities con un tiempo mínimo de comercialización.
[3] Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

sensores lo seguirán importando de LIBELIUM y otros terceros. No obstante, para desarrollar su propio hardware necesitan primeramente las curvas características de cada sensor, al este respecto la empresa LIBELIUM, quien suministra estos sensores, no proporciona las curvas características de sus sensores, ya que su negocio es vender sus sensores con la tarjeta electrónica de control incluida.

Por lo tanto, WAPOSAT para brindar soluciones flexibles que se adapten a las necesidades de sus clientes, necesita su propio hardware, lo que implica identificar la curva característica de cada sensor que desea usar y adicionalmente diseñar y construir su hardware adecuado (tarjeta electrónica).

Para tal objetivo, el área técnica comercial de WAPOSAT deberá demostrar que es posible identificar la curva característica y a la vez construir su respectiva tarjeta electrónica para uno de sus sensores más complejos y difíciles de calibrar (sensor de Potencial de Hidrógeno, pH) antes de aprobar un presupuesto para la migración a su propio hardware.

El diseño, entrenamiento y validación de una red neuronal artificial que identifique la curva característica de un sensor de Potencial de Hidrógeno, pH (uno de los sensores más complejos que utiliza, ya que sus mediciones dependen también del sensor de temperatura y diversos puntos de referencia), les ayudará en la primera etapa. Una vez identificada plenamente la curva característica de uno de los sensores, el sensor de pH, se procederá con el diseño y fabricación de su respectivo hardware para adquirir, acondicionar, tratar e interpretar los datos.

1.1.2 Metodología de trabajo

Para cumplir los objetivos, descritos más adelante, se iniciará con la adquisición de datos mediante una DAQ, posteriormente se tratará dichos datos, luego seleccionaremos una arquitectura apropiada, así como un método de entrenamiento y optimización, finalmente se va a testear la RNA entrenada mediante un análisis minucioso de ella. Una

vez la RNA pase con éxito el testeo minucioso, ésta podrá utilizarse en el rango de entrada para el cual fue entrenada.

1.1.3 Alcances

El alcance está limitado a obtener la curva característica del sensor de Potencial de Hidrógeno, pH, de la empresa Libelium y la construcción de una tarjeta electrónica que ejecute la RNA entrenada para la medición de temperatura y pH. El sensor de pH depende de 6 variables (temperatura, lectura del sensor de pH, y 4 puntos de calibración) para indicar el nivel de pH de un líquido.

Aquí no se detalla las herramientas y procedimientos de adquisición de datos, así como también no se detalla el hardware utilizado para la lectura, tratamiento e interpretación de datos del sensor utilizado.

Se adjunta el código de programación de la tarjeta electrónica desarrollada.

No se tomará en cuenta la posible histéresis[4] que pueda sufrir el sensor de pH y el sensor PT1000 (de temperatura).

1.1.4 Limitaciones

El rango de medición de un sensor común de pH abarca el rango de 0 a 14, sin embargo, en este trabajo se va a determinar la curva característica par un rango entre 0.6 a 13.9 de pH a una temperatura entre 0.2°C y 95.5°C. Para temperaturas fuera de estos límites no se garantiza el correcto desempeño de la RNA.

Los resultados de este trabajo están orientados a un sensor pH de tipo electrodo combinado, que es suministrado por Libelium. Esta limitación se debe a que cada fabricante construye los sensores de pH con sus propias especificaciones, incluso

[4] Es un fenómeno que intrínseco de los materiales, en la que la relación de salida/entrada a veces revelará resultados diferentes a medida que las señales varían en la dirección del movimiento [137].

dos sensores similares fabricados por una misma empresa pueden comportarse de manera diferente, ya que su comportamiento depende del estado de los elementos químicos que utiliza para realizar su tarea.

1.2 Planteamiento del problema

1.2.1 Enunciado del problema

El problema es que la empresa WAPOSAT no cuenta con la curva característica del sensor de Potencial de Hidrógeno, PH[5]. La causa de esto es que la empresa Libelium, de quien adquiere este sensor, no suministra la curva característica de dicho sensor, lo que trae como consecuencia no ofrecer una solución flexible en hardware y software que se adapten a las necesidades específicas de cada cliente, con lo que pierde competitividad en el mercado.

WAPOSAT de no considerar la identificación de la curva característica de su sensor de pH (sensor más complejo que utiliza), se verá obligado a siempre comprar sensores con su respectiva tarjeta electrónica de adquisición, acondicionamiento, tratamiento e interpretación de datos de un fabricante tercero, y ofrecer soluciones hasta donde lo permitan los sensores y las tarjetas de control actualmente utilizadas.

1.2.2 Formulación del problema

¿De qué manera se identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo de la calidad del agua?

1.2.2.1 Problemas específicos

1. ¿Qué se requiere para el entrenamiento de la Red Neuronal Artificial?

[5] Este sensor de pH es uno de los varios tipos de sensores que hace uso la empresa Waposat

2. ¿Qué arquitectura de Red Neuronal Artificial utilizar?
3. ¿Qué algoritmo utilizar para entrenar la Red Neuronal Artificial?
4. ¿De qué manera ejecutar la red neuronal artificial entrenada?

1.2.3 Situación actual

1.2.3.1 Situación actual de medición

Actualmente WAPOSAT realiza la medición del Potencial de Hidrogeno, pH, con sensores y su correspondiente tarjeta electrónica de adquisición, acondicionamiento, e interpretación de datos suministrados por Libelium. La tarjeta electrónica es una caja negra, pues el fabricante no proporciona la curva característica del sensor de pH. Todo ello debido a que no cuentan con hardware propio.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar una Red Neuronal Artificial para identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo en tiempo real de la calidad del agua.

1.3.2 Objetivos Específicos

1. Adquirir datos de pH y temperatura, y sus voltajes correspondientes del módulo electrónico Smart Water y Waspmote para entrenar la Red Neuronal Artificial.

2. Determinar y parametrizar una arquitectura de Red Neuronal Artificial para entrenar la misma mediante un algoritmo de entrenamiento.
3. Elegir y ejecutar un algoritmo de entrenamiento para entrenar la Red Neuronal Artificial.
4. Diseñar y construir un equipo electrónico para ejecutar la Red Neuronal Artificial entrenada y validada en una empresa de monitoreo de la calidad del agua.

1.4 Hipótesis y operacionalización de variables

1.4.1 Hipótesis General

El Desarrollo de una Red Neuronal Artificial permite identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo de la calidad del agua.

1.4.2 Hipótesis específicas

1.4.2.1 Hipótesis específica 1

La adquisición de datos de pH y temperatura, y sus voltajes correspondientes del módulo electrónico Smart Water y Waspmote permite entrenar la Red Neuronal Artificial.

1.4.2.2 Hipótesis especifica 2

La determinación y parametrización de una arquitectura de Red Neuronal Artificial permite entrenar la misma mediante un algoritmo de entrenamiento.

1.4.2.3 Hipótesis específica 3:

La elección y ejecución de un algoritmo de entrenamiento nos permite entrenar la Red Neuronal Artificial.

1.4.2.4 Hipótesis específica 4:

El Diseño y construcción de un equipo electrónico permite ejecutar la red neuronal artificial entrenada en una empresa de monitoreo de la calidad del agua.

1.4.3 Operacionalización de Variables

Tabla 1. Variable dependiente y variables independientes, (fuente: elaborado por el autor).

Variable dependiente	Definición	Dimensiones	Indicadores
Identificación de la curva característica de un Sensor de Potencial de Hidrógeno	Relación salida/entrada que muestra la respuesta ante una determinada entrada en un ciclo cerrado.	- Entrada de temperatura. - Entrada de pH. - Puntos de calibración	- Rango de temperatura de 0.2°C a 80°C. - Rango de pH de 0.6 a 13.9 - 4 puntos de calibración (pH 4, pH 7, pH 10, T °C)
Variables Independientes	**Definición**	**Dimensiones**	**Indicadores**
Desarrollo de una Red Neuronal Artificial		- Numero de capas ocultas - Número de neuronas por capas	- Cantidad de capas ocultas. - Cantidad de neuronas por capa.

CAPITULO II

MARCO TEÓRICO

2.1 Potencial de hidrogeno - pH

2.1.1 Introducción

El concepto de pH es único entre las cantidades fisicoquímicas que se encuentran comúnmente en el Libro Verde de la IUPAC[6] en que, en términos de su definición,

$$\mathrm{pH} = -\log\left(\mathrm{a_H}\right) \tag{2.1}$$

involucra una sola cantidad de iones, la actividad del ión hidrógeno, que no se puede medir por ningún método termodinámicamente válido y requiere una convención para su evaluación, [5].

2.1.2 Actividad de iones de hidrógeno

El pH fue originalmente definido por Sørensen en 1909 en términos de la concentración de iones de hidrógeno (en la nomenclatura moderna) como $\mathrm{pH} = -\log(c_\mathrm{H}/c°)$ donde c_H es la concentración de iones de hidrógeno en mol dm^{-3}, y $c° = 1\ \mathrm{mol\ dm}^{-3}$ es la concentración de cantidad estándar. Posteriormente, se ha aceptado que es más satisfactorio definir el pH en términos de la actividad relativa de los iones de hidrógeno en solución.

$$\mathrm{pH} = -\log(\mathrm{a_H}) = -\log\left(m_\mathrm{H}\gamma_\mathrm{H}/m°\right) \tag{2.2}$$

donde $\mathrm{a_H}$ es la actividad relativa (base de la molalidad) y γ_H es el coeficiente de actividad molal del ion hidrógeno H^+ en la molalidad m_H, y $m°$ es la molalidad estándar. *La*

[6] La Unión Internacional de Química Pura y Aplicada (IUPAC) es la autoridad mundial en nomenclatura y terminología química, incluida la denominación de nuevos elementos en la tabla periódica; en métodos estandarizados para la medición; y sobre pesos atómicos, y muchos otros datos evaluados críticamente, [146].

cantidad de pH pretende ser una medida de la actividad de los iones de hidrógeno en solución. Sin embargo, dado que se define en términos de una cantidad que no puede medirse mediante un método termodinámicamente válido, la ecuación (2.2) puede ser solo una *definición nocional* de pH, [6].

2.1.3 Trazabilidad y Métodos Primarios de Medición

2.1.3.1 Relación con el SI

Como el pH, una sola cantidad de iones, no es determinable en términos de una unidad fundamental (o base) de cualquier sistema de medición, anteriormente existía cierta dificultad para proporcionar una base adecuada para la trazabilidad de las mediciones de pH. Ahora se dispone de un enfoque satisfactorio en el sentido de que las determinaciones de pH pueden incorporarse al SI si pueden trazarse hasta mediciones realizadas utilizando un método que cumple con la definición de un "método primario de medición", [7].

2.1.3.2 Método primario de medida

La característica esencial de dicho método es que debe funcionar de acuerdo con una ecuación de medición bien definida en la que todas las variables se pueden determinar experimentalmente en términos de unidades SI. Cualquier limitación en la determinación de las variables experimentales, o en la teoría, debe incluirse dentro de la incertidumbre estimada del método si se debe establecer la trazabilidad al SI. Si se usa una convención sin una estimación de su incertidumbre, no se establecería la trazabilidad real al SI. En la siguiente sección, se muestra que la célula Harned cumple la definición de un método primario para la medición de la función de acidez, $\mathrm{p}(a_{\mathrm{H}}\gamma_{\mathrm{Cl}})$, [8].

2.1.4 Célula Harned como Método Primario para la Medición Absoluta del Ph

2.1.4.1 Celda Harned

La celda sin transferencia es definida por:

$$\mathrm{Pt} \mid \mathrm{H_2} \mid \text{beffer S}, \mathrm{Cl^-} \mid \mathrm{AgCl} \mid \mathrm{Ag} \qquad (2.3)$$

conocida como la célula Harned y que contiene tampón estándar, S, e iones de cloruro, en forma de cloruro de potasio o sodio, que se agregan para usar el electrodo de cloruro de plata-plata. La aplicación de la ecuación de Nernst a la reacción celular espontánea:

$$^1/_2\,\mathrm{H_2} + AgCl \longrightarrow Ag(s) + H^+ + Cl^- \qquad (2.4)$$

produce la diferencia de potencial E_1 de la celda [corregida a 1 atm (101.325 kPa), la presión parcial del gas de hidrógeno I utilizado en electroquímica de preferencia a 100 kPa] como

$$E_\mathrm{I} = E° - [(RT/F)\mathrm{lg}10]\,\mathrm{lg}[(m_\mathrm{H}\gamma_\mathrm{H}/m°)(m_\mathrm{Cl}\gamma_\mathrm{Cl}/m°)] \qquad (2.5)$$

que se puede reorganizar, ya que $a_\mathrm{H} = m_\mathrm{H}\gamma_\mathrm{H}/m°$, para dar la función de acidez

$$\mathrm{p}(a_\mathrm{H}\gamma_\mathrm{Cl}) = -\log(a_\mathrm{H}\gamma_\mathrm{Cl})$$
$$= (E_\mathrm{I} - E°)/[(RT/F)\mathrm{ln}10] + \mathrm{lg}(m_{Cl}/m°) \qquad (2.6)$$

donde $E°$ es la diferencia de potencial estándar de la célula y, por lo tanto, del electrodo de cloruro de plata-plata, y γ_Cl es el coeficiente de actividad del ion cloruro.

Nota 1: El signo del potencial de electrodo estándar de una reacción electroquímica es el que se muestra en un voltímetro de alta impedancia cuando el cable conectado al electrodo de hidrógeno estándar está conectado al polo negativo del voltímetro.

Los pasos en el uso de la celda se resumen en la Figura *2.1* y se describe en los siguientes párrafos.

La diferencia de potencial estándar del electrodo de cloruro de plata-plata, $E°$, se determina a partir de una celda Harned en la que solo HCl está presente en una molalidad fija (por ejemplo, $m = 0.01\ \text{mol kg}^{-1}$). La aplicación de la ecuación de Nernst a la célula HCl

$$\text{Pt} \mid \text{H}_2 \mid \text{HCl}(m) \mid \text{AgCl} \mid \text{Ag} \qquad (2.7)$$

da

$$E_{\text{Ia}} = E° - [(2RT/F)\ln 10]\,\lg[(m_{\text{HCl}}/m°)(\gamma_{\pm\text{Cl}})\,] \qquad (2.8)$$

donde E_{Ia} se ha corregido a 1 presión parcial de atmósfera de gas hidrógeno (101.325 kPa) y $\gamma_{\pm\text{HCl}}$ es el coeficiente medio de actividad iónica del HCl, [9].

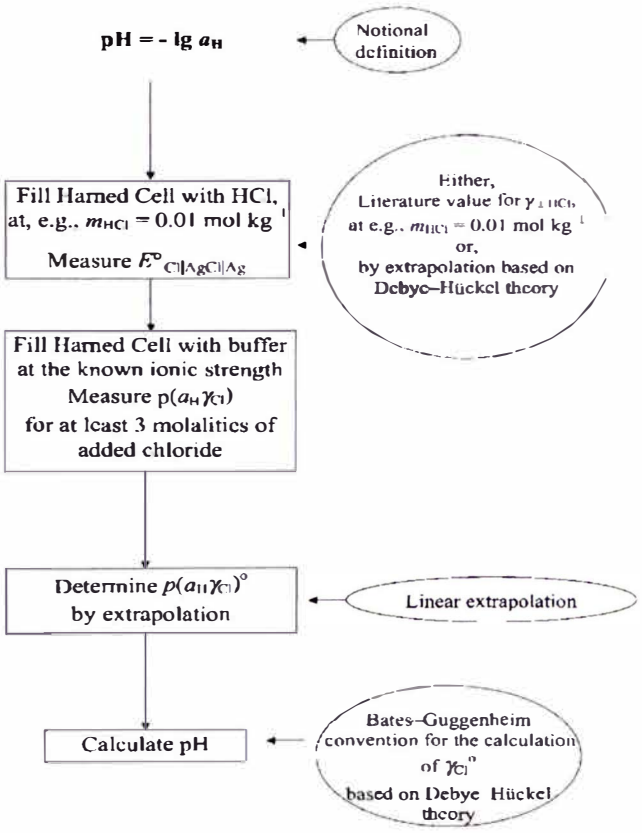


Figura 2.1. Operación de la célula Harned como método principal para la medición del pH absoluto, *[9]*.

2.1.5 Calibración de un sensor de pH

2.1.5.1 Soluciones tampón

Los tampones son soluciones que tienen valores de pH constantes y la capacidad de resistir cambios en ese nivel de pH. Se utilizan para calibrar el sistema de medición de pH (electrodo y medidor). Puede haber pequeñas diferencias entre la salida de un electrodo y otro, así como cambios en la salida de los electrodos con el tiempo. Por lo tanto, el sistema debe calibrarse periódicamente. Los tampones están disponibles con una amplia gama de valores de pH, y vienen en forma líquida premezclada o como cápsulas de polvo seco convenientes. La mayoría de los medidores de pH requieren calibración a varios valores de pH específicos. Por lo general, una calibración se realiza cerca del punto isopotencial (la señal producida por un electrodo a pH 7 es de 0 mV a 25 ° C), y una segunda se realiza típicamente a pH 4 o pH 10. Es mejor seleccionar un tampón como lo más cerca posible del valor real de pH de la muestra a medir, [10].

Normalmente cada fabricante tienes su propio procedimiento de calibrar el sensor de pH que fabrican. A continuación, se describe la calibración del sensor de pH de Libelium utilizado en este trabajo.

2.1.5.2 Calibración del sensor de pH Libelium

Se recomienda una calibración periódica para los sensores de pH si se desea una medición precisa. Si el sensor se desplegará en un entorno con una temperatura cambiante o la calibración se realizará a una temperatura diferente de las condiciones de operación, también se requerirá una compensación de temperatura para actualizar la sensibilidad del sensor a las condiciones actuales

El material requerido para la calibración del sensor de pH consiste en una placa de sensor Waspmote y Smart Water, el sensor de pH a calibrar (más un sensor Pt-1000 si se va a aplicar compensación de temperatura) y tres soluciones tampón de pH, una de 7.0 pH y dos de valores más altos y más bajos (4.0 pH y 10.0 pH). Tenga en cuenta que, para una calibración adecuada, todos los tampones deben estar a la misma temperatura, siendo una temperatura lo más cercana posible a la de la operación o, si no se conoce, de aproximadamente 25 °C. La siguiente lista incluye el proceso completo de calibración, [11]:

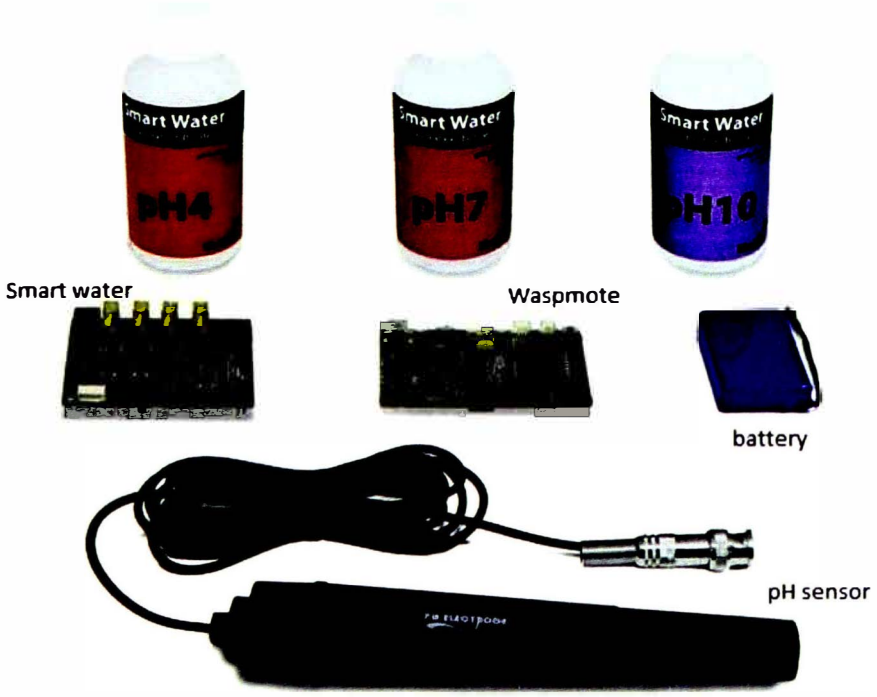


Figura 2.2. Imagen del material necesario para el proceso de calibración de pH, *[11]*.

1. Encienda el Waspmote con la placa del sensor Smart Water y el sensor de pH y el sensor Pt-1000 conectados.
2. Cargue el código "pH Sensor Reading", descrito en el Anexo J, y asegúrese de que los datos se reciban correctamente a través del USB u otro módulo de comunicación.
3. Vierta las soluciones en tres vasos de precipitados. La solución de pH 4.0 es roja, la solución de pH 7.0 es amarilla y la solución de pH 10.0 es azul.

Se recomienda que las soluciones estén a la temperatura que se encontrará en el entorno de instalación.

4. Introduzca el sensor de pH y el Pt-1000 en la solución tampón de pH 7.0 y espere una medición estable, que puede demorar unos minutos. Asegúrese de que los sensores estén completamente inmersos en la solución. Cuando haya una salida estable para los sensores, anote el valor en voltios obtenido. Es realmente importante dar tiempo a que la salida se estabilice, especialmente la primera vez que usamos un sensor. Esto llevará varios minutos.

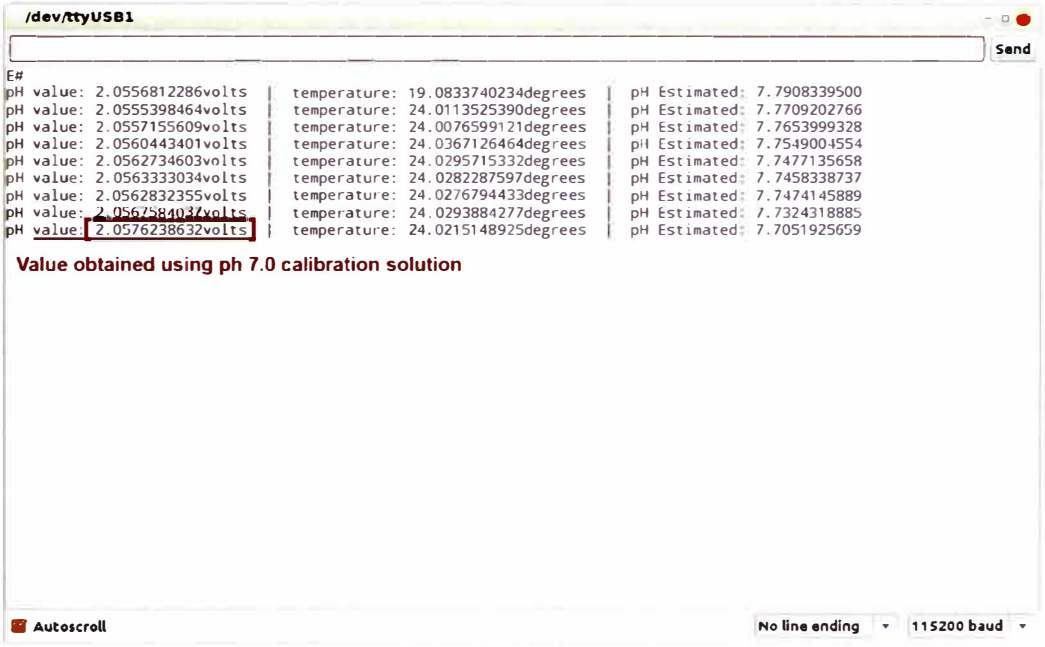


Figura 2.3. Este valor en voltios se debe anotar para cada solución de calibración, *[12]*.

5. Saque el sensor de la solución y enjuáguelo suavemente, preferiblemente con agua destilada o desionizada, e introdúzcalo en la solución de pH 4.0, lo que provocará un aumento en el voltaje de salida, junto con el sensor Pt1000 para verificar que Todas las soluciones están a la misma temperatura. Nuevamente, espere la estabilización de los valores de salida y anótelos.

6. Repita el paso 3 con la solución de pH 10.0, lo que debería hacer que el voltaje de salida del sensor sea inferior al de la solución de pH 7.0. Por debajo de 25 ºC, las salidas esperadas para estas soluciones son 2.048 V para 7 pH, 2.227 V para 4.0 pH y 1.868 V para 10.0 pH, con la posibilidad de encontrar una diferencia de algunas décimas de milivoltios para cada valor y un cambio en la sensibilidad debido a la diferencia de temperatura.
7. Los valores significativamente diferentes después de la exposición del sensor a las soluciones pueden ser causados por una burbuja en el bulbo sensible, especialmente si es la primera calibración después del envío. Sacudir el sensor hacia abajo como un termómetro clínico los eliminará, resolviendo el problema.
8. Introduzca los valores de calibración en el código de medición como se muestra en las imágenes a continuación.

Figura 2.4.En esta definición, debe escribir el valor en voltios obtenido con las soluciones de pH, *[13]*.

9. Cargue el código nuevamente con los nuevos valores de calibración obtenidos del proceso de calibración.

10. Para saber más sobre los kits de calibración proporcionados por Libelium, vaya a la sección "Soluciones de calibración", [14].

2.1.6 Selección de sensores de pH

La selección de un electrodo de pH se basa principalmente en los comportamientos químicos y físicos del medio del proceso. Las combinaciones con los requisitos específicos del proceso o de la industria, como "higiénico", reducirán la elección de electrodos de pH adecuados para ciertas aplicaciones. Sin embargo, los criterios clave se basan en la vida útil máxima esperada y los esfuerzos de mantenimiento, como la calibración o el rellenado de KCL, [15].

Entornos de aplicación, [15]:

- Estándar.
- Con elevado valor de materia orgánica.
- Con conductividad pequeña.
- Sanitarias.
- En condiciones de proceso extremas – productos abrasivos.
- Con riesgo elevado de formación de adherencias.
- Químicamente exigente.

2.1.7 Aplicaciones

Se espera que el mercado de medidores de pH crezca a una tasa compuesta anual del 3% durante el período de pronóstico de 2016 a 2024. Los medidores de pH se utilizan para medir la oxidación/reducción y el potencial de acidez/alcalinidad en el laboratorio, agua y aguas residuales, productos químicos y petroquímicos, farmacéuticos y varias otras aplicaciones.

Los medidores de pH se utilizan principalmente en plantas de tratamiento de agua, plantas de energía, sector minero, industria de petróleo y gas, industria de procesamiento químico, industria de alimentos y bebidas e industrias farmacéuticas que juntas constituyen el mercado. La disponibilidad de bajo costo, fácil de operar, junto con las crecientes preocupaciones alimentarias y ambientales y el aumento de los gastos de atención médica son factores importantes que impulsan el crecimiento del mercado. También se anticipa que el medidor de pH altamente confiable y preciso con un nivel reducido de mantenimiento será un factor de crecimiento importante.

Los investigadores utilizan medidores de pH en una amplia gama de campos de investigación, incluidos biológicos y químicos, agrícolas y ambientales, y prácticamente todo tipo de fabricación debido al amplio uso de características ampliadas relacionadas con la tecnología de medidores de pH. La creciente preocupación por la seguridad debido a los efectos nocivos sobre la calidad de los bienes producidos en instalaciones industriales está impulsando aún más el mercado. Se espera que el segmento de tecnología de análisis estereoscópico crezca a la tasa más alta debido a la amplia gama de aplicaciones que requieren alta precisión y exactitud, como pruebas de medicina, pruebas de alimentos y bebidas, pruebas ambientales, académicas e investigación.

La industria de la ciencia de los alimentos contribuyó con un 40% de ingresos y también tuvo la mayoría de la participación de mercado de medidores de pH. La gran área de aplicación en industrias como el procesamiento de alimentos y jugos de frutas, vino, agricultura, lácteos y otros son impulsores importantes del mercado.

Un número cada vez mayor de laboratorios de investigación y análisis, como laboratorios de análisis de agua, análisis de alimentos y análisis de suelos, también son uno de los factores que se espera aumenten la demanda en el mercado. Sin embargo, la dificultad para medir el pH de soluciones con baja concentración de iones, el daño debido a ciertas soluciones y el crecimiento de bacterias y la contaminación son algunos factores que limitan el crecimiento del mercado de medidores de pH.

El segmento de productos de medidor de pH incluye equipos portátiles y en línea (Inline u online), junto con un segmento de tipo de sobremesa. Los dispositivos portátiles están especializados para medir el pH con precisión, son fáciles de transportar y operar con un rendimiento confiable durante largas horas. El tamaño pequeño y la alta movilidad de estos medidores es uno de los factores clave que se atribuyen a su rápido crecimiento. Los medidores en línea (Inline u online) son adecuados para unidades de fabricación de medicamentos, plantas de tratamiento de agua, plantas de energía, para fines de laboratorio y más adecuados para su utilización en entornos hostiles.

Sobre la base del uso final, el mercado se segmenta aún más en biotecnología y compañías farmacéuticas, pruebas ambientales, alimentos y bebidas, instituciones académicas y gubernamentales. La industria de alimentos y bebidas es el mayor consumidor del mercado debido a las crecientes regulaciones para la seguridad del consumidor. Se espera que la mejora tecnológica, como el análisis de múltiples parámetros y la incorporación de sensores inteligentes en medidores de pH, sean los factores que impulsen el mercado durante el período de pronóstico.

Se espera que el análisis del nivel de pH del agua para evitar la corrosión en las tuberías, la creciente conciencia sobre la calidad del agua potable y los equipos como el condensador, las calderas, aumenten el uso industrial de los medidores de pH.

América del Norte es el mayor consumidor de medidores de pH debido a la presencia de los principales actores clave que aumentan la cuota de mercado. La cuota de mercado de EE. UU. Es importante debido a la popularidad en varias áreas de aplicación con pautas gubernamentales sobre industrias intensivas en agua. La Agencia de Protección Ambiental de los Estados Unidos (EPA) impuso regulaciones estrictas sobre las plantas de tratamiento de agua para reducir los contaminantes. Los productos de mesa debido a la alta demanda en investigación industrial, laboratorios e institutos de investigación dominaron la industria regional.

Se espera que las crecientes preocupaciones ambientales, de seguridad alimentaria y de medicinas aumenten la demanda en la región de Asia Pacífico. Las aplicaciones de dispositivos de medición novedosos entre científicos agrícolas, investigadores industriales, expertos y una creciente conciencia sobre sensores tecnológicamente avanzados son factores probables que permiten el crecimiento del mercado. Se espera que la conciencia sobre la contaminación y el tratamiento de aguas residuales aumente la demanda en la región.

La presencia de infraestructura industrial, atención médica avanzada sofisticada y un alto grado de actividades de I + D, problemas de salud y conciencia sobre las enfermedades transmitidas por el agua contribuyeron a la demanda de medidores de pH en Europa, la región que genera mayores ingresos para medidores de pH.

Se espera que el producto de sobremesa pierda participación en los próximos años debido a la creciente adopción de instrumentos de análisis de parámetros múltiples. Se espera que la importancia del monitoreo continuo del valor del pH en los procesos industriales impulse la demanda en el mercado del medidor en linea de pH.

Los principales fabricantes mundiales de medidores de pH incluyen Hanna Instruments, Danaher Corporation, Agilent Technologies, Thermo Fisher Scientific Corporation, Metrohm USA, Mettler Toledo, Horiba y PerkinElmer, [16].

Resumen de las principales aplicaciones, [16]:

- Tratamiento de agua potable y agua residuales
- Plantas de energía
- Sector minero (para evitar la corrosión de tuberías)
- Industria petroquímica (petróleo y gas)
- Industria química
- Industrias farmacéuticas

- Industrias biotecnológicas
- Industria de alimentos y bebidas (procesamiento de alimentos y jugos de frutas, vino, agricultura, lácteos)
- Industria agrícolas y ambientales
- Entornos de alta precisión y exactitud, como pruebas de medicina, pruebas de alimentos y bebidas, pruebas ambientales, académicas e investigación.

2.2 Curva característica de un sensor

La observación cuidadosa de la relación salida/entrada de un bloque a veces revelará resultados diferentes a medida que las señales varían en la dirección del movimiento. Los sistemas mecánicos a menudo muestran una pequeña diferencia de longitud a medida que se invierte la dirección de la fuerza aplicada. El mismo efecto surge cuando un campo magnético se invierte en un material magnético. Esta característica se llama histéresis. La Figura 2.5 es un diagrama generalizado de la relación salida/entrada que muestra lo que ocurre un ciclo cerrado. El efecto generalmente se reduce a medida que se reduce la amplitud de las excursiones sucesivas, siendo esta una forma de tolerar el efecto. Está presente en la mayoría de los materiales. Se han desarrollado materiales especiales que exhiben baja histéresis para su aplicación. [17]

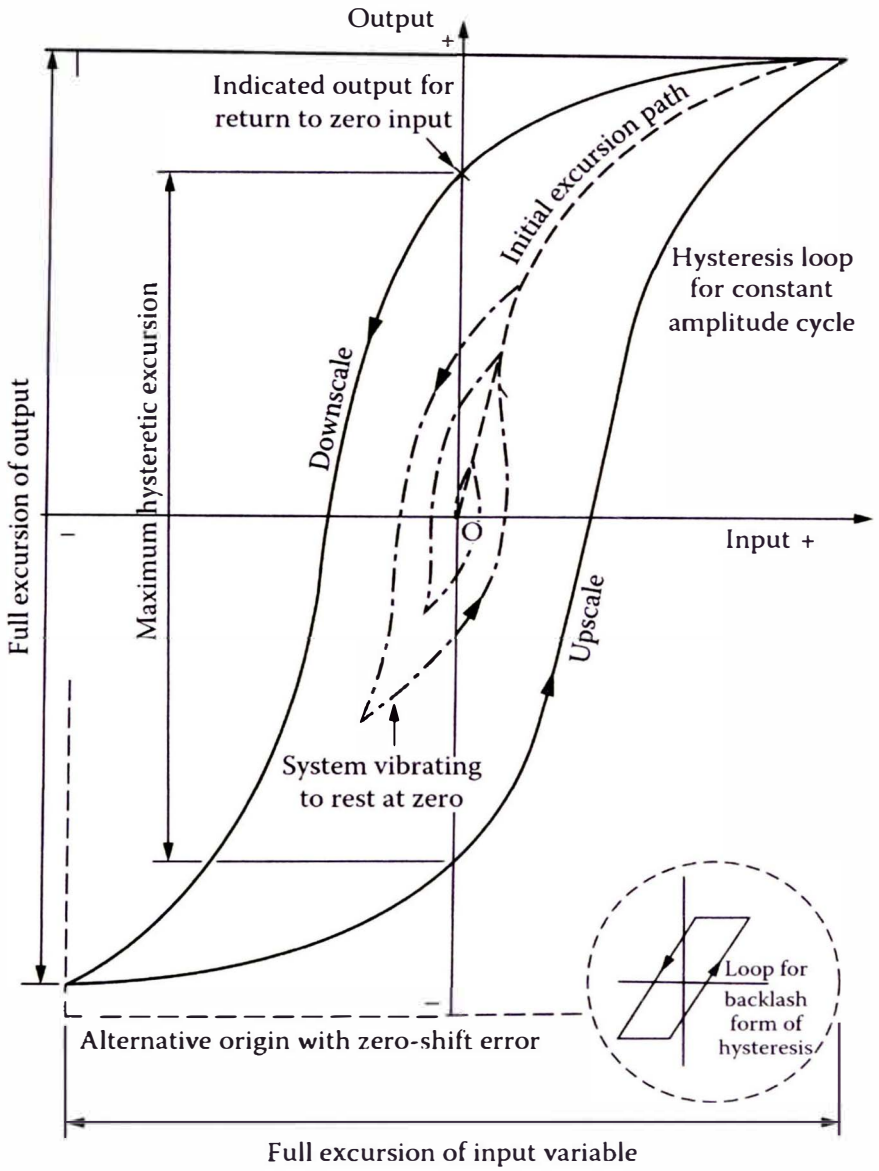


Figura 2.5. Gráfico generalizado de la relación salida/entrada donde está presente la histéresis *[18]*.

2.3 Series de Taylor

2.3.1 Para el caso de vectores

Por supuesto, el índice de desempeño de la RNA no será función de un escalar. Será una función de todos los parámetros de RNA (pesos y bias), de los cuales puede

haber un número muy grande. Por lo tanto, necesitamos extender la expansión de la serie Taylor a funciones de muchas variables. Considere la siguiente función de variables:

$$F(\mathbf{x}) = F(x_1, x_2, \ldots, x_n). \tag{2.9}$$

La expansión de la serie de Taylor para esta función, cerca del punto x^*, será:

$$\begin{aligned} F(\mathbf{x}) = F(\mathbf{x}^*) &+ \frac{\partial}{\partial x_1} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*) \\ &+ \frac{\partial}{\partial x_2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_2 - x_2^*) + \cdots \\ &+ \frac{\partial}{\partial x_n} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_n - x_n^*) \\ &+ \frac{1}{2}\frac{\partial^2}{\partial x_1^2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)^2 \\ &+ \frac{1}{2}\frac{\partial^2}{\partial x_1 x_2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)^2 (x_2 - x_2^*) + \cdots. \end{aligned} \tag{2.10}$$

Esta notación es un poco engorrosa. Es más conveniente escribirlo en forma matricial, como en:

$$\begin{aligned} F(\mathbf{x}) = F(\mathbf{x}^*) &+ \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) \\ &+ \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \cdots \end{aligned} \tag{2.11}$$

donde $\nabla F(\mathbf{x})$ es la *gradiente*, y es definida como

$$\nabla F(\mathbf{x}) = \left[\frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \quad \cdots \quad \frac{\partial}{\partial x_n} F(\mathbf{x})\right]^T, \tag{2.12}$$

y $\nabla^2 F(\mathbf{x})$ es el Hessiano, y es definido como:

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 x_2} F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1 x_n} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_1 x_2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1 x_2} F(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_n x_2} F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix} \quad (2.13)$$

El gradiente y el Hessian son muy importantes para nuestra comprensión de las superficies de desempeño. [19]

2.4 Redes neuronales artificiales

2.4.1 Introducción

Mientras se lee estas palabras se estás utilizando una compleja red neuronal biológica. Tiene un conjunto altamente interconectado de unas 10^{11} neuronas para facilitar la lectura, respiración, movimiento y pensamiento. Cada una de sus neuronas biológicas, un rico conjunto de tejidos y química, tiene la complejidad, si no la velocidad, de un microprocesador. Parte de tu estructura neural estaba con nosotros al nacer. Otras partes han sido establecidas por la experiencia. Los científicos apenas han empezado a comprender cómo funcionan las redes neuronales biológicas. En general, se entiende que todas las funciones neuronales biológicas, incluida la memoria, se almacenan en las neuronas y en las conexiones entre ellas. El aprendizaje se ve como el establecimiento de nuevas conexiones entre neuronas o la modificación de las conexiones existentes. Esto nos lleva a la siguiente pregunta: aunque solo tenemos una comprensión rudimentaria de las redes neuronales biológicas, ¿es posible construir un pequeño conjunto de simples "neuronas" artificiales y quizás capacitarlas para que cumplan una función útil? La respuesta es "sí".

Las neuronas que consideramos aquí no son biológicas. Son extremadamente simples abstracciones de neuronas biológicas, realizadas como elementos en un programa o tal vez como circuitos hechos de silicio. Las redes de estas neuronas artificiales no tienen una fracción del poder del cerebro humano, pero pueden ser entrenadas para realizar funciones útiles. [20]

2.5 Breve historia sobre las redes neuronales artificiales

Aunque el estudio del cerebro humano y el interés de imitarlo mediante la lógica y otras teorías tiene miles de años y se extiende a nuestra actualidad, resumiremos brevemente los aportes más relevantes, aclarando previamente que todo aporte por más pequeño que sea tiene gran importancia, ya que investigadores más notables tomaron esos pequeños aportes y lo ampliaron, mejoraron o lo abordaron desde otro punto de vista en menor o gran medida.

1315: Ramon Llull publica el **Ars Magna,** en esta obra Llull planteó una de las primeras tesis sobre razonamiento automático; es decir, el desarrollo de una máquina lógica capaz de validar o invalidar argumentos. Llull diseñó una máquina llamada ***Ars generalis ultima*** era un autómata muy rudimentario, concretamente un autómata finito; aun así se considera el primer intento de utilización de medios lógicos para producir conocimiento y, aunque era una teoría que no podía ir más allá de "los estados que se habían programado", fue una curiosa primera implementación de sistemas relacionados con la inteligencia artificial y, por tanto, de la búsqueda que una máquina fuese capaz de aprender y pensar (o demostrar dogmas de fe). [21] [22]

Pensadores y matemáticos relevantes se sintieron atraídos por el trabajo de LLull, lo analizaron y lo estudiaron. Giordano Bruno, Gottfried Leibniz y Pierre-Simón Laplace son tres de esos hombres, todos filósofos, matemáticos, astrónomos y físicos. [23]

1943: Warren S. McCulloch (neurophysiologist) y Walter Pitts (mathematician) publican "**A Logical Calculus of the Ideas Immanent in Nervous Activity**". Este artículo sostienen que "Debido al carácter "todo o nada" de la actividad nerviosa, los eventos neuronales y las relaciones entre ellos pueden tratarse mediante la lógica proposicional." [24] [25], en el que tratan sobre el funcionamiento de la neurona biológica y la demostración matemática de que la actividad neuronal y sus relaciones puede tratarse mediante la lógica proposicional. Este articulo sentó las bases para un amplio estudio de las Redes Neuronales Artificiales tal y como lo conocemos hoy en día.

1949: Donald Hebb hace un estudio minucioso del proceso de aprendizaje neuronal en el que sostienen que "las percepciones *simples* son en realidad complejas, que son aditivas, que dependen en parte de la actividad motora, y que su aparente simplicidad es solo el resultado final de un largo proceso de aprendizaje" [26] así como "Los experimentos con animales y los datos clínicos humanos indican que la percepción de diagramas simples como conjuntos distintivos no se da de inmediato, sino que se adquiere lentamente a través del aprendizaje." [27]. De lo que se deduce que el aprendizaje neuronal se fortalece cada vez que se utiliza, con lo que posteriormente se desarrollará algoritmos de entrenamiento para las Redes Neuronales Artificiales.

1954: No fue hasta 1954 (el año de la muerte de Turing) que Belmont Farley y Wesley Clark, trabajando en el MIT, lograron ejecutar las primeras simulaciones por computadora de pequeñas redes neuronales. Farley y Clark pudieron entrenar redes neuronales artificiales que contenían como máximo 128 neuronas para reconocer patrones simples. Además, descubrieron que la destrucción aleatoria de hasta el 10% de las neuronas en una RNA entrenada no afecta el rendimiento de la RNA en su tarea, una

característica que recuerda la capacidad del cerebro para tolerar el daño limitado infligido por una cirugía, un accidente, o enfermedad. [28]

1958: Frank Rosenblatt trabajó con perceptrones en el Laboratorio Aeronáutico de Cornell (1957-1959) [29] que culminó en el desarrollo y la construcción del hardware del Mark I [30], un clasificador de patrones visuales, tenía una capa de entrada (sensorial) de 400 unidades fotosensibles en una cuadrícula de 20x20, una pequeña retina, una capa de asociación de 512 unidades (motores paso a paso), cada una de las cuales podría tomar varias entradas excitadoras e inhibitorias, y una capa de salida (respuesta) de 8 unidades. [31]

1961: K. Steinbuch publica "Die Lernmatrix" (la matriz de aprendizaje) en el que trata sobre memorias asociativas matriciales. Las aplicaciones importantes de la matriz de aprendizaje serán: reconocimiento automático de caracteres, reconocimiento automático de voz, decodificación general de caracteres aprendidos y posiblemente alterados, recuperación de información. [32]

1960: Bernard Widrow comenzó a trabajar en redes neuronales artificiales a fines de la década de 1950, casi al mismo tiempo que Frank Rosenblatt desarrolló la regla de aprendizaje de perceptrón. En 1960, Widrow, y su estudiante graduado Marcian Hoff, introdujeron la RNA ADALINE (ADAptive LInear NEuron) y una regla de aprendizaje que llamaron algoritmo LMS (Least Mean Square) [33].

1975: CMAC es un sistema adaptativo mediante el cual las funciones de control para muchos grados de libertad que operan simultáneamente se pueden calcular haciendo referencia a una tabla en lugar de mediante una solución matemática de ecuaciones simultáneas, [34].

1980: Kunihiko Fukushima presenta Neocognitron en el que propone un modelo de red neuronal artificial para un mecanismo de reconocimiento de patrones visuales. La RNA se autoorganiza al "aprender sin un maestro" y adquiere la capacidad de reconocer patrones de estímulo basados en la similitud geométrica (Gestalt) de sus formas sin verse afectados por sus posiciones. Esta RNA recibe el nombre de "neocognitron". Después de completar la autoorganización, la RNA tiene una estructura similar al modelo de jerarquía del sistema nervioso visual propuesto por Hubel y Wiesel. La RNA consta de una capa de entrada (matriz de fotorreceptores) seguida de una conexión en cascada de varias estructuras modulares, cada una de las cuales está compuesta por dos capas de células conectadas en cascada. La primera capa de cada módulo consiste en "S-células", que muestran características similares a las células simples o células hipercomplejas de orden inferior, y la segunda capa consiste en "C-células" similares a las células complejas o células hipercomplejas de orden superior. Las sinapsis aferentes de cada célula S tienen plasticidad y son modificables, [35].

1987: Stephen Grossberg presenta su teoría de resonancia adaptativa, o ART, es una teoría cognitiva y neural de cómo el cerebro aprende rápidamente a clasificar, reconocer y predecir objetos y eventos en un mundo cambiante, y un conjunto de algoritmos que incorporan computacionalmente los principios de ART y que se utilizan en Aplicaciones tecnológicas y de ingeniería a gran escala en las que se necesita un aprendizaje rápido, estable e incremental sobre entornos cambiantes complejos. El ART aclara los procesos cerebrales de los cuales emergen las experiencias conscientes. Predice un vínculo funcional entre los procesos de conciencia, aprendizaje, expectativa, atención, resonancia y sincronía (CLEARS), incluida la predicción de que "todos los estados conscientes son estados resonantes". Esta conexión aclara cómo la dinámica del cerebro permite que un individuo se adapte de forma autónoma en tiempo real a un mundo

que cambia rápidamente. ART predice cómo funciona la atención de arriba hacia abajo y regula el aprendizaje rápido y estable de las categorías de reconocimiento, [36], [37].

1982: Teuvo Kohonen realiza un estudio teórico y simulaciones por computadora de un nuevo proceso de autoorganización. El descubrimiento principal es que en una RNA simple de elementos físicos adaptativos que recibe señales de un espacio de eventos primario, las representaciones de señales se asignan automáticamente a un conjunto de respuestas de salida de tal manera que las respuestas adquieren el mismo orden topológico que el del evento primario, [38].

1982: J. A. Feldman & D. H. Ballard, presentan un modelo conexionista general basado en las conexiones complejas y paralelas de los animales. [39]

1984: E. Hinton y T. Sejnowski en su teoría Boltzmann Machines describen un método general de búsqueda paralela, basado en mecanismos estadísticos, y muestran cómo conduce a una regla de aprendizaje general para modificar las fortalezas de conexión para incorporar el conocimiento sobre un dominio de tareas de manera eficiente, [40].

1986: E. Rumelhart, E. Hinton & J. Williams, describen un nuevo procedimiento de aprendizaje, retropropagación, para redes neuronales artificiales de unidades similares a neuronas. El procedimiento ajusta repetidamente los pesos de las conexiones en la RNA para minimizar una medida de la diferencia entre el vector de salida real de la red y el vector de salida deseado, [41].

1997: Schmidhuber & Hochreiter propusieron un marco de red neuronal artificial recurrente, la memoria a largo plazo (LSTM), [42].

A continuación, en la **¡Error! No se encuentra el origen de la referencia.** se presenta un breve resumen de la evolución histórica de las redes neuronales artificiales.

Tabla 2. Resumen de la evolución de las redes neuronales artificiales (fuente: resumen de la sección Breve historia sobre las redes neuronales elaborado por el autor, tomando como referencia la Figura 2-7 del libro DARPA Neural Network Study - Final Report [43].

EARLY WORK 1940-1960 FUNDAMENTAL CONCEPTS	MCCULLOCH & PITTS	BOOLEAN LOGIC	Los eventos neuronales y las relaciones entre ellos pueden tratarse mediante la lógica proposicional [24] [25]
	HEBB	SYNAPTIC LEARNING RULE	"Las percepciones *simples* son en realidad complejas, que son aditivas, que dependen en parte de la actividad motora, y que su aparente simplicidad es solo el resultado final de un largo proceso de aprendizaje" [26].
	FARLEY & CLARK	FIRST SIMULATION	Pudieron entrenar redes neuronales que contenian como máximo 128 neuronas para reconocer patrones simples, [28].
	RONSENBLATT	PERCEPTRON	Trabajó con perceptrones en el Laboratorio Aeronáutico de Cornell (1957-1959) [29], que culminó en el desarrollo y la construcción del hardware del Mark I [30], un clasificador de patrones visuales.
	STEINBUCH, TAYLOR	ASSOCIATIVE MEMORIES	Publica "Die Lernmatrix" (la matriz de aprendizaje) en el que trata sobre memorias asociativas matriciales
TRANSITION 1960-1980 THEORICAL FOUNDATIONS	Widrow & Hoff	LMS Algorithm	Introdujeron la red ADALINE (ADAptive LInear NEuron) y una regla de aprendizaje que llamaron algoritmo LMS (Least Mean Square) [33].
	Albus	Cerebellum Model (CMAC)	CMAC es un sistema adaptativo mediante el cual las funciones de control para muchos grados de libertad que operan simultáneamente se pueden calcular haciendo referencia a una tabla en lugar de mediante una solución matemática de ecuaciones simultáneas, [34].
	Anderson	Linear Associative memory	
	Von Der Malsburg	Competitive Learning	
	Fukushima	Neocognitron	Propone un modelo de red neuronal artificial para un mecanismo de reconocimiento de patrones visuales, [35].
	Grossberg	ART, BCS	Teoria cognitiva y neural de cómo el cerebro aprende rápidamente a

			clasificar, reconocer y predecir objetos y eventos en un mundo cambiante, [36], [37].
RESURGENCE 1980- THEORY BIOLOGY COMPUTERS	Kohonen	Feature Map	Crea un principio que facilita la formación automática de mapas topológicamente correctos de características de eventos observables, [38].
	Feldman & Ballard	Connectionist Models	J. A. Feldman & D. H. Ballard, presentan un modelo conexionista general basado en las conexiones complejas y paralelas de los animales. [39]
	Hopfield	Associative Memory Theory	
	Reilly, Cooper et al.	RCE	
	Hinton & Sejnowski	Boltzmann Machines	Describen un método general de búsqueda paralela, basado en mecanismos estadísticos, y muestran cómo conduce a una regla de aprendizaje general para modificar las fortalezas de conexión para incorporar el conocimiento sobre un dominio de tareas de manera eficiente, [40].
	Rumelhart	Backpropagation	Describen un nuevo procedimiento de aprendizaje, retropropagación. El procedimiento ajusta repetidamente los pesos de las conexiones en la red, [41].
	Schmidhuber & Hochreiter	LSTM	Propusieron un marco de red neuronal recurrente, la memoria a largo plazo (LSTM), [42].

2.6 Aplicaciones de las redes neuronales

Un estudio DARPA de las rede neuronales de 1988 [44], enumera varias aplicaciones de redes neuronales artificiales, comenzando con el ecualizador de canal adaptativo en aproximadamente 1984. Este dispositivo, que es un gran éxito comercial, es una red única que se utiliza en sistemas telefónicos de larga distancia para estabilizar la señal de voz. El informe DARPA continúa para enumerar otras aplicaciones comerciales, incluido un reconocedor de palabras pequeñas, un monitor de procesos, un clasificador de sonar y un sistema de análisis de riesgos.

Miles de redes neuronales se han aplicado en cientos de campos desde que se redactó el informe DARPA. Una lista de algunas de esas aplicaciones sigue.

Aeroespacial

Pilotos automáticos de aeronaves de alto rendimiento, simulaciones de trayectoria de vuelo, sistemas de control de aeronaves, mejoras de piloto automático, simulaciones de componentes de aeronaves, detectores de fallas de componentes de aeronaves, [45].

Automoción

Sistemas automáticos de guía para automóviles, control de inyectores de combustible, sistemas automáticos de frenado, detección de fallas, sensores de emisiones virtuales, analizadores de actividad de garantía, [46].

Bancario

Lectores de cheques y otros documentos, evaluadores de aplicaciones de crédito, previsión de efectivo, clasificación de empresas, previsión de tasa de cambio, predicción de tasas de recuperación de préstamos, medición del riesgo de crédito, [46].

Defensa

Dirección de armas, seguimiento de objetivos, discriminación de objetos, reconocimiento facial, nuevos tipos de sensores, sonar, radar y procesamiento de señales de imagen, incluyendo compresión de datos, extracción de características y supresión de ruido, identificación de señales/imágenes, [46].

Electrónica

Código de predicción de secuencia, diseño de chip de circuito integrado, control de procesos, análisis de fallos de chip, visión de máquina, síntesis de voz, modelado no lineal, [46].

Entretenimiento

Animación, efectos especiales, previsión de mercado, [46].

Financiero

Evaluación de bienes raíces, asesor de préstamos, evaluación de hipotecas, calificación de bonos corporativos, análisis de uso de líneas de crédito, programa de negociación de cartera, análisis financiero corporativo, predicción del precio de la moneda, [46].

Seguros

Evaluación de la aplicación de políticas, optimización de productos, [46].

Fabricación

Control de procesos de fabricación, diseño y análisis de productos, diagnóstico de procesos y máquinas, identificación de partículas en tiempo real, sistemas de inspección de calidad visual, pruebas de cerveza, análisis de calidad de soldadura, predicción de calidad de papel, análisis de calidad de chip de computadora, análisis de operaciones de molienda, producto químico Análisis de diseño, análisis de mantenimiento de máquinas, licitación de proyectos, planificación y gestión, modelado dinámico de sistemas de procesos químicos, [46].

Médico

Análisis de células de cáncer de mama, análisis de EEG y ECG, diseño de prótesis, optimización de los tiempos de trasplante, reducción de gastos hospitalarios, mejora de la calidad del hospital, aviso de pruebas en salas de emergencias, [47].

Petróleo y gas

Exploración, sensores inteligentes, modelado de yacimientos, decisiones de tratamiento de pozos, interpretación sísmica, [47].

Robótica

Control de trayectoria, robot de montacargas, controladores de manipuladores, sistemas de visión, vehículos autónomos, [47].

Habla

Reconocimiento de voz, compilación de voz, clasificación de vocales, síntesis de texto a voz, [47].

Valores

Análisis de mercado, calificación automática de bonos, sistemas de asesoramiento bursátil, [47].

Telecomunicaciones

Compresión de imágenes y datos, servicios de información automatizados, traducción en tiempo real del lenguaje hablado, sistemas de procesamiento de pagos de los clientes, [47].

Transporte

Sistemas de diagnóstico de frenos de camiones, programación de vehículos, sistemas de enrutamiento, [47].

Conclusión

La cantidad de aplicaciones de redes neuronales, el dinero que se ha invertido en software y hardware de redes neuronales y la profundidad y amplitud del interés en estos dispositivos es enorme, [47].

Otras aplicaciones: CIMON - Compañero Móvil Interactivo de la Tripulación de la agencia espacial alemana [48], Agentes remotos [49] [50], asistentes de conductores [51] [52] [53] [54] [55] [56] [57] [56] [58] [59] [60] [61], mantenimiento [62], seguro automotriz [63], pólizas de seguros [64], automovilismo [65] [66].

2.7 Modelo Neuronal

2.7.1 Red Neuronal de una sola entrada

En la Figura 2.6 se muestra una neurona de una entrada. La entrada escalar p se multiplica por el peso escalar w para formar wp, y el resultado de los términos se envía al sumador. La otra entrada, 1, se multiplica por un bias b y luego se pasa a la sumador. La salida n del sumador, a menudo denominada entrada de red neuronal artificial, la cual ingresa a la función de transferencia f, que produce a la salida de la neurona el escalar a. (Algunos autores usan el término "función de activación" en lugar de la función de transferencia y "offset" en lugar del bias.)

Si relacionamos este sencillo modelo con la neurona biológica, el peso w corresponde a la fuerza de una sinapsis, el cuerpo celular está representado por la suma y la función de transferencia, y la salida a de la neurona representa la señal en el axón.

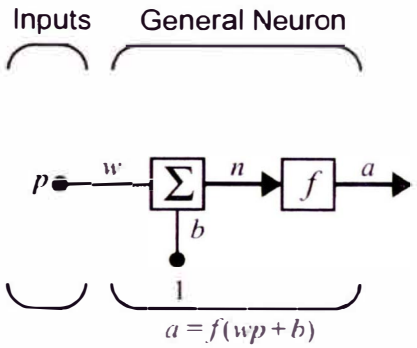


Figura 2.6. Neurona de una sola entrada, *[67]*.

La salida de la neurona es calculada como:

$$a = f(wp + b). \tag{2.14}$$

Si, por ejemplo, $w = 3, p = 2$ y $b = -1.5$, luego

$$A = f(3(2) - 1.5) = f(1.5). \tag{2.15}$$

La salida actual depende de la función de transferencia particular que se elija.

El bias es muy parecido a un peso, excepto que tiene una entrada constante de 1. Sin embargo, si no desea tener un bias en una neurona en particular, puede omitirse.

Tenga en cuenta que w y b son dos parámetros escalares ajustables de la neurona. Normalmente, la función de transferencia es elegida por el diseñador y luego los parámetros w y b serán ajustados según alguna regla de aprendizaje, de modo que la relación de entrada/salida de la neurona cumpla con algún objetivo específico. [68, pp. 2-3]

2.7.2 Funciones de transferencia

La función de transferencia en la Figura 2.6 puede ser una función lineal o no lineal de n. Se elige una función de transferencia particular para satisfacer alguna especificación del problema que la neurona está intentando resolver.

Existe una gran variedad de funciones de transferencia. Tres de las funciones más utilizadas se discuten a continuación.

2.7.2.1 Función de transferencia de límite rígido (Hard Limit Transfer Function)

La función de transferencia de límite rígido (Hard Limit), que se muestra en el lado izquierdo de la Figura 2.7, establece la salida de la neurona en 0 si el argumento de la función es menor que 0, o 1 si su argumento es mayor o igual que 0. Esta función se utiliza para crear neuronas que clasifican entradas en dos categorías distintas.

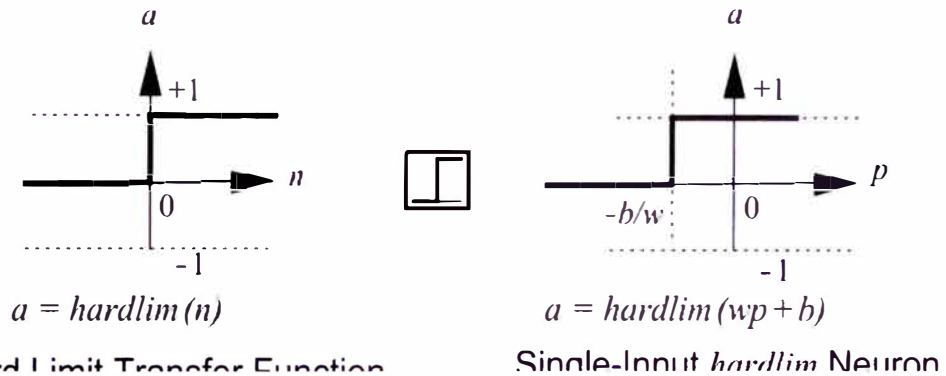


Figura 2.7. Función de Transferencia Rígido, [69].

El gráfico en el lado derecho de la Figura 2.7 ilustra la característica de entrada/salida de una neurona de entrada única que utiliza una función de transferencia de límite duro (hard limit). Aquí podemos ver el efecto del peso y el bias. Tenga en cuenta que se muestra un icono para la función de transferencia de límite rígido entre las dos figuras. Dichos íconos reemplazarán la f general en los diagramas de red para mostrar la función de transferencia particular que se está utilizando.

2.7.2.2 Función de transferencia lineal (Linear Transfer Function)

La salida de una *función de transferencia lineal* es igual a su entrada:

$$a = n, \tag{2.16}$$

esto se ilustra en la Figura 2.8.

Neuronas con esta función de transferencia son usado en las RNA ADALINE.

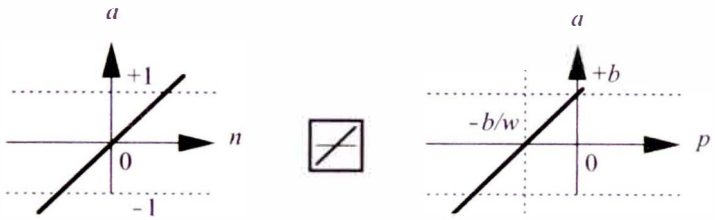


Figura 2.8. Función de Transferencia Lineal, *[69]*.

La salida (a) frente a la característica de entrada (p) de una neurona lineal de entrada única con un bias b se muestra a la derecha de la Figura 3.

2.7.2.3 Función de transferencia logarítmica sigmoidal

La función de transferencia sigmoidal-logarítmica se muestra en la Figura 4.

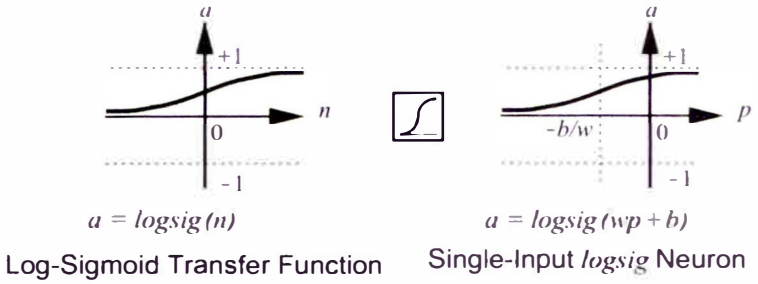


Figura 2.9. Función de Transferencia Logarítmica Sigmoidal, *[70]*.

Esta función de transferencia toma la entrada (que puede tener cualquier valor entre más y menos infinito) y reduce la salida en el rango de 0 a 1, de acuerdo con la expresión:

$$a = \frac{1}{1 + e^{-n}}. \tag{2.17}$$

La función de transferencia logarítmica-sigmoidal se usa comúnmente en redes neuronales multicapa que se entrenan con el algoritmo de propagación hacia atrás (backpropagation), en parte porque esta función es diferenciable.

La mayoría de las funciones de transferencia se resumen en la Tabla 3. Por supuesto, si lo desea puede definir otras funciones de transferencia además de las que se muestran en la Tabla 3.

Tabla 3. Funciones de transferencia, [71]

Nombre	Relación de entrada/salida	Icono	Función en MATLAB
Limite rígido (hard limit)	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Limite rígido simétrico (Symmetrical Hard Limit)	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Lineal (Linear)	$a = n$		purelin
Lineal Saturado (Saturating Linear)	$a = 1 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Lineal Saturado Simétrico (Symmetric Saturating Linear)	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Sigmoidal Logaritmico (Log-Sigmoid)	$a = \frac{1}{1 + e^{-n}}$		logsig
Sigmoidal Tangente Hiperbólico (Hyperbolic Tangent Sigmoid)	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$		tansig
Lineal Positivo (Positive Linear)	$a = 0 \quad n < 0$ $a = n \quad n \leq 0$		poslin
Competitivo (competitive)	$a = 1$ *neurona con n máximo* $a = n$ *todas las otras neuronas*		compet

2.7.3 Neurona de múltiples entradas

Normalmente, una neurona tiene más de una entrada. En la Figura 2.10 se muestra una neurona con R entradas. Las entradas individuales $p_1, p_2, \ldots p_R$ están ponderadas cada una por los elementos correspondientes $w_{1,1}, w_{1,2}, \ldots w_{1,R}$ de la matriz de pesos $\mathbf{W}$.

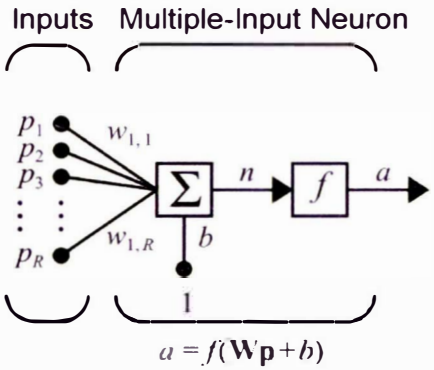


Figura 2.10. Neurona de Múltiples entradas, *[72]*.

La neurona tiene un bias b, que se suma con las entradas ponderadas para formar la entrada neta n.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b. \tag{2.18}$$

Esta expresión puede ser escrita matricialmente en forma matricial:

$$n = \mathbf{W}\mathbf{p} + b, \tag{2.19}$$

donde la matriz $\mathbf{W}$ para el caso de una neurona tiene solo una fila.

Ahora la salida de la neurona se puede escribir como:

$$a = f(\mathbf{W}\mathbf{p} + b). \tag{2.20}$$

Afortunadamente, las redes neuronales a menudo se pueden describir con matrices.

Hemos adoptado una convención particular al asignar los índices de los elementos de la matriz de ponderación. El primero índice indica el destino neuronal particular para ese peso. El segundo índice indica la fuente de la señal alimentada a la neurona. Por lo tanto, los índices en $w_{1,2}$ dicen que este peso representa la conexión a la primera (y única)

neurona de la segunda fuente. Por supuesto, esta convención es más útil si hay más de una neurona, como será el caso más adelante.

Nos gustaría dibujar redes neuronales con varias neuronas, cada una con varias entradas. Además, nos gustaría tener más de una capa de neuronas. Se puede imaginar cuán compleja puede aparecer una red neuronal de este tipo si se dibujaran todas las líneas. Tomaría mucha tinta, difícilmente podría leerse, y la gran cantidad de detalles podría ocultar las características principales. Por lo tanto, vamos a utilizar una notación ab abreviada. Una neurona de entrada múltiple que usa esta notación se muestra en la Figura 2.11.

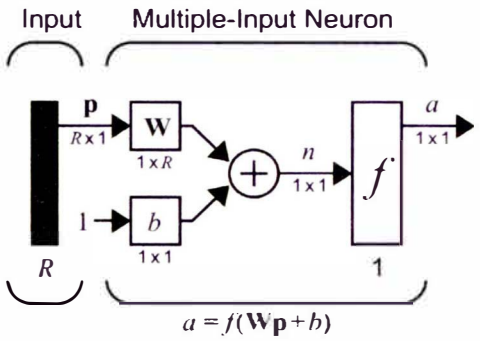


Figura 2.11. Neurona con R entradas, Notación Abreviada, *[73]*.

Como se muestra en la Figura 2.11, el vector de entrada **p** está representado por la barra vertical a la izquierda. Las dimensiones de **p** se muestran debajo de la variable como $R \times 1$, lo que indica que la entrada es un solo vector de R elementos. Estas entradas van a la matriz de peso W, que tiene R columnas, pero solo una fila en este caso de una sola neurona. Una constante 1 ingresa a la neurona como entrada y se multiplica por un bias (o sesgo) escalar b. La entrada neta a la función de transferencia es n, que es la suma del bias b y el producto $\mathbf{Wp}$. La salida de la neurona a es un escalar en este caso. Si tuviéramos más de una neurona, la salida de la RNA sería un vector.

Las dimensiones de las variables en estas notaciones abreviadas siempre se incluirán., de modo que pueda saber inmediatamente si estamos hablando de un escalar, un vector o una matriz. No se tendrá que adivinar el tipo de variable o sus dimensiones.

Tenga en cuenta que el número de entradas a una RNA se establece según las especificaciones externas del problema. Si, por ejemplo, desea diseñar una RNA para predecir las condiciones de vuelo de cometa y las entradas son la temperatura del aire, la velocidad del viento y la humedad, entonces habría tres entradas a la RNA. [68]

2.8 Arquitectura de redes neuronales artificales

En general, una neurona, incluso con muchas entradas, puede no ser suficiente. Podríamos necesitar cinco o diez, operando en paralelo, en lo que llamaremos una "capa". Este concepto de una capa se describe a continuación.

2.8.1 Neuronas de una capa

En la Figura 2.12 se muestra una RNA de una sola capa de S neuronas. Tenga en cuenta que cada una de las R entradas está conectada a cada uno de las neuronas y que la matriz de pesos ahora tiene S filas.

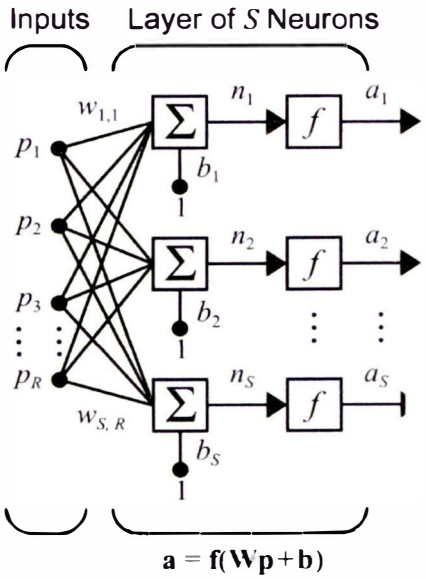


Figura 2.12. Capa de S Neuronas, *[74]*.

La capa incluye la matriz de ponderación, los sumadores, el vector bias **b**, los cuadros de función de transferencia y el vector de salida **a**. Algunos autores se refieren a las entradas como otra capa, pero no lo haremos aquí.

Cada elemento del vector de entrada **p** está conectado a cada neurona a través de la matriz de ponderaciones **W**. Cada neurona tiene un bias b_i, un sumador, una función de transferencia f y una salida a_i. En conjunto, las salidas forman el vector de salida **a**.

Es común que el número de entradas a una capa sea diferente del número de neuronas (es decir, $R \neq S$).

Podría preguntarse si todas las neuronas en una capa deben tener la misma función de transferencia. La respuesta es no; puede definir una sola capa (compuesta) de neuronas que tienen diferentes funciones de transferencia combinando dos de las RNA que se muestran arriba en paralelo. Ambas RNA tendrían las mismas entradas, y cada RNA crearía algunas de las salidas.

Los elementos del vector de entrada entran en la RNA a través de la matriz de ponderación **W**:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \vdots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}. \qquad (2.21)$$

Como se señaló anteriormente, los índices de fila de los elementos de la matriz **W** indican la neurona de destino asociada con ese peso, mientras que los índices de la columna indican la fuente de la entrada para ese peso. Por lo tanto, los índices en $w_{3,2}$ dicen que este peso representa la conexión a la tercera neurona desde la segunda fuente.

Afortunadamente, las S neuronas y R entradas de una RNA de una capa también se puede dibujar en notación abreviada, como se muestra en la Figura 2.13.

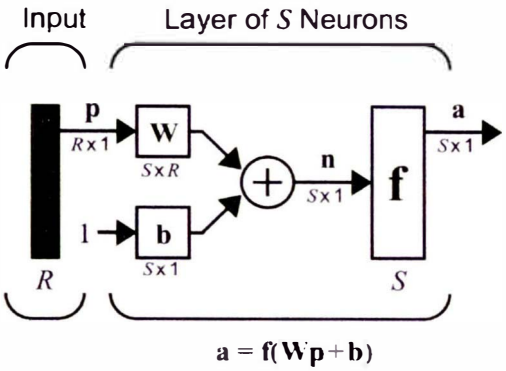
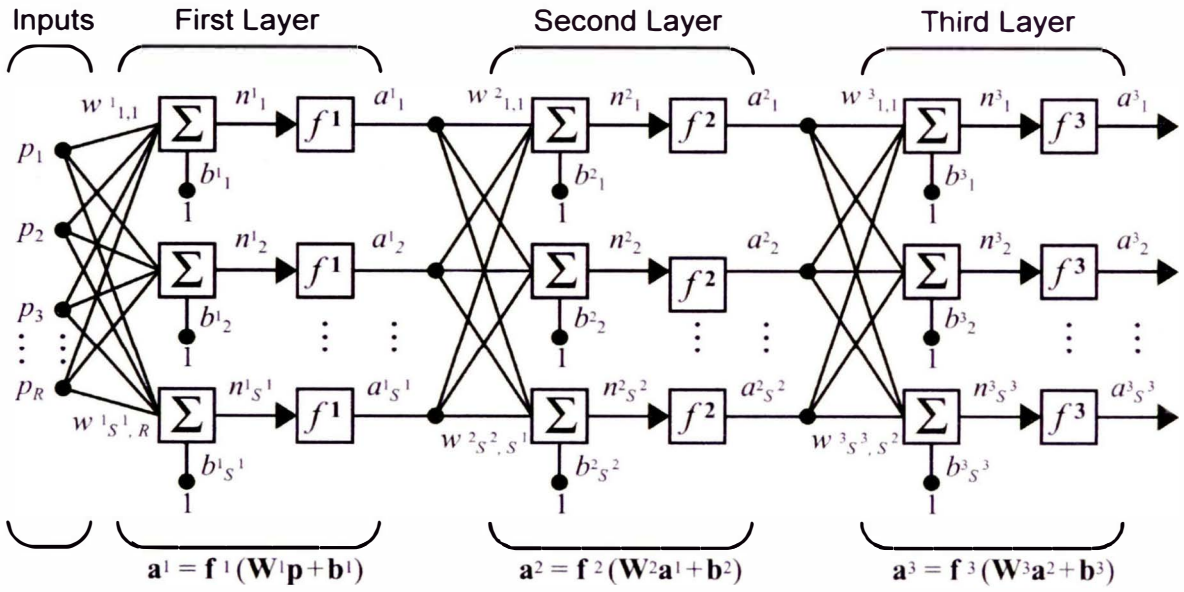


Figura 2.13. Capa de S Neuronas, Notación Abreviada, *[75]*.

Aquí nuevamente, los símbolos debajo de los variables dicen que, para esta capa, **p** es un vector de longitud R, **W** es una matriz $S \times R$, y **a** y **b** son vectores de longitud S. Como se definió anteriormente, la capa incluye la matriz de ponderación, las operaciones de suma y multiplicación, el vector de bias **b**, los cuadros de función de transferencia y el vector de salida.

2.8.2 Neuronas de múltiples capas

Ahora consideremos una RNA con varias capas. Cada capa tiene su propia matriz de pesos **W**, su propio vector bias **b**, un vector de entrada de RNA **n** y un vector de salida **a**. Necesitamos introducir alguna notación adicional para distinguir entre estas capas. Usaremos superíndices para identificar las capas. Específicamente, agregamos el número de la capa como superíndice a los nombres de cada una de estas variables. Por lo tanto, la matriz de pesos para la primera capa se escribe como $\mathbf{W}^1$, y la matriz de pesos para la segunda capa se escribe como $\mathbf{W}^2$. Esta notación se utiliza en la RNA de tres capas que se muestra en la Figura 2.14.



$$\boldsymbol{a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3)}$$

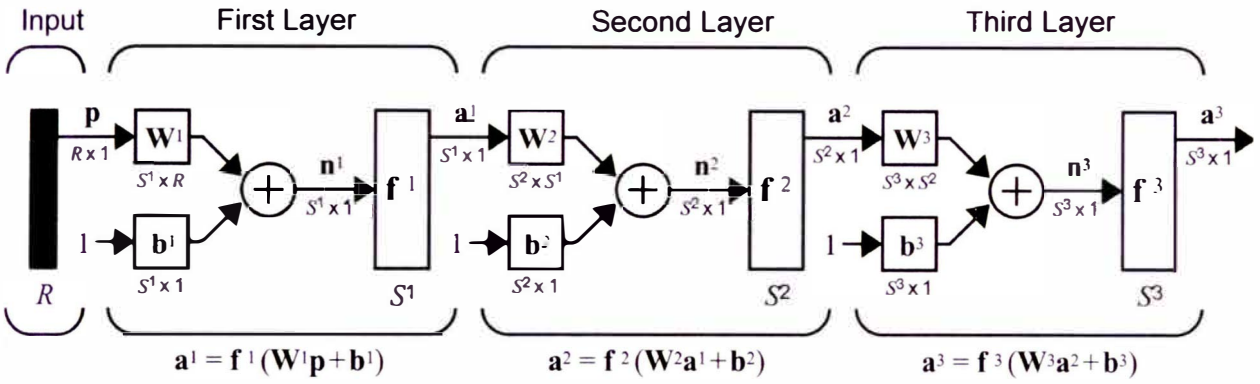
Figura 2.14. Red neuronal artificial de tres capas, [76].

Como se muestra, hay R entradas, S^1 neuronas en la primera capa, S^2 neuronas en la segunda capa, etc. Como se señaló, las diferentes capas pueden tener diferentes números de neuronas.

Las salidas de las capas uno y dos son las entradas para las capas dos y tres. Por lo tanto, la capa 2 se puede ver como una RNA de una capa con $R = S^1$ entradas, $S = S^2$ neuronas y una matriz de pesos $\mathbf{W}^2$ de dimension $S^2 \times S^1$. La entrada a la capa 2 es $\mathbf{a}^1$, y la salida es $\mathbf{a}^2$.

Una capa cuya salida es la salida de RNA se denomina *capa de salida*. Las otras capas se llaman *capas ocultas*. La RNA que se muestra arriba tiene una capa de salida (capa 3) y dos capas ocultas (capas 1 y 2).

La misma RNA de tres capas analizada anteriormente también se puede dibujar usando nuestra notación abreviada, como se muestra en la Figura 2.15.



$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3)$$

Figura 2.15. Red neuronal artificial de Tres Capas, Notación abreviada, *[77]*.

Las RNA multicapa son más poderosas que las RNA de una sola capa. Por ejemplo, una RNA de dos capas que tiene una primera capa sigmoidal y una segunda capa lineal puede entrenarse para aproximar la mayoría de las funciones arbitrariamente bien. Las RNA de una sola capa no pueden hacer esto.

En este punto, la cantidad de opciones que se deben hacer al especificar una RNA puede parecer abrumadora, por lo que consideremos este tema. El problema no es tan malo como parece. Primero, recuerde que la cantidad de entradas a la RNA y la cantidad de salidas de la RNA están definidas por especificaciones de problemas externos. Entonces, si hay cuatro variables externas para ser utilizadas como entradas, hay cuatro entradas a la RNA. De manera similar, si va a haber siete salidas de la RNA, debe haber siete neuronas en la capa de salida. Finalmente, las características deseadas de la señal de salida también ayudan a seleccionar la función de transferencia para la capa de salida. Si una salida debe ser -1 o 1, entonces se debe usar una función de transferencia de límite rígido simétrica. Por lo tanto, la arquitectura de una RNA de una sola capa está casi completamente determinada por las especificaciones del problema, incluido el número específico de entradas, salidas y la característica particular de la señal de salida.

Ahora, ¿y si tenemos más de dos capas? Aquí el problema externo no le indica directamente la cantidad de neuronas requeridas en las capas ocultas. De hecho, hay

pocos problemas para los cuales se puede predecir el número óptimo de neuronas necesarias en una capa oculta. Este problema es un área activa de investigación.

En cuanto al número de capas, la mayoría de las RNA neuronales prácticas tienen solo dos o tres capas. Cuatro o más capas se utilizan raramente.

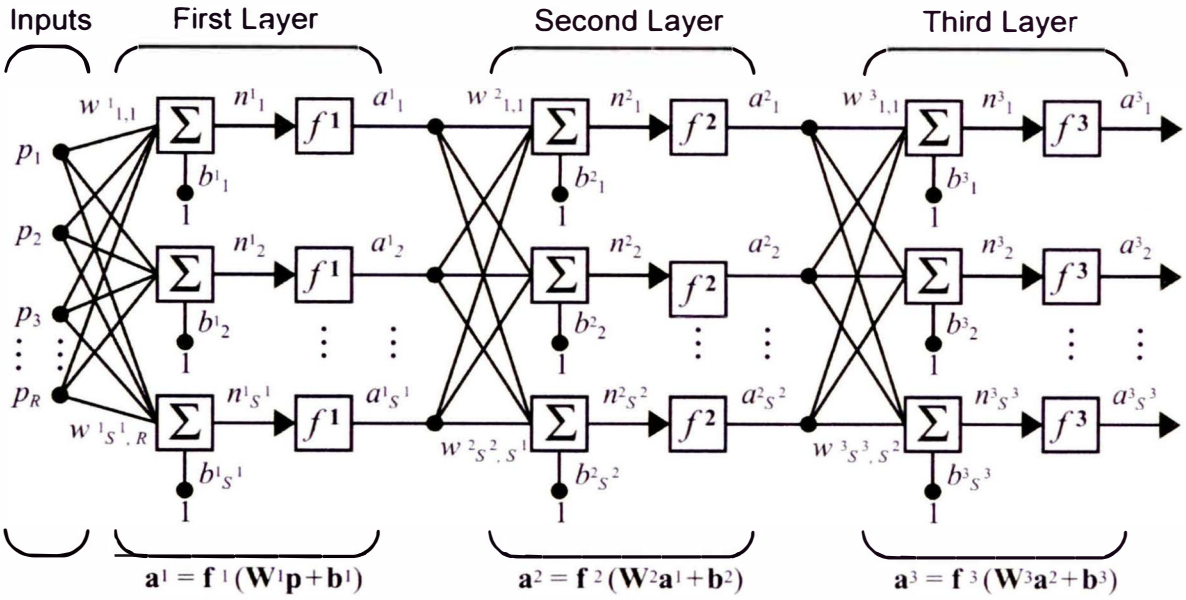
Debemos decir algo sobre el uso de bias. Uno puede elegir neuronas con o sin bias. El bias le da a la RNA una variable adicional, por lo que es de esperar que las RNA con bias sean más poderosas que las que no lo tienen, y eso es cierto. Tenga en cuenta, por ejemplo, que una neurona sin bias siempre tendrá una entrada de RNA n de cero cuando las entradas de RNA sean cero. Esto puede no ser deseable y puede evitarse mediante el uso de un bias.

2.9 Perceptron multicapa

La Figura 2.16 muestra un diagrama de perceptrón de tres capas. Tenga en cuenta que simplemente hemos puesto en cascada tres RNA de perceptrón. La salida de la primera red neuronal es la entrada a la segunda red neuronal, y la salida de la segunda red neuronal es la entrada a la tercera red neuronal. Cada capa puede tener un número diferente de neuronas, e incluso una función de transferencia diferente. Se está usando superíndices para identificar el número de capa. Por lo tanto, la matriz de pesos para la primera capa se escribe como $\mathbf{W^1}$ y la matriz de ponderación para la segunda capa se escribe $\mathbf{W^2}$.

Para identificar la estructura de una RNA de múltiples capas, a veces usaremos la siguiente notación abreviada, donde el número de entradas va seguido del número de neuronas en cada capa, [78]:

$$R - S^1 - S^2 - S^3 \tag{2.22}$$



$$\boldsymbol{a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3)}$$

Figura 2.16. RNA de tres capas [79]

2.9.1 Aproximación de funciones

En los sistemas de control, por ejemplo, el objetivo es encontrar una función de retroalimentación adecuada que se asigne desde las salidas medidas a las entradas de control. En los filtros adaptativos, el objetivo es encontrar una función que asigne los valores retardados de una señal de entrada a una señal de salida apropiada. El siguiente ejemplo ilustrará la flexibilidad del perceptrón de múltiples capas para implementar funciones.

Considere la RNA 1-2-1 de dos capas que se muestra en la Figura 2.17. Para este ejemplo, la función de transferencia, o función de activación, para la primera capa es log-sigmoidal y la función de transferencia para la segunda capa es lineal. En otras palabras,

$$f^1(n) = \frac{1}{1 + e^{-n}}, \text{y } f^2(n) = n \tag{2.23}$$

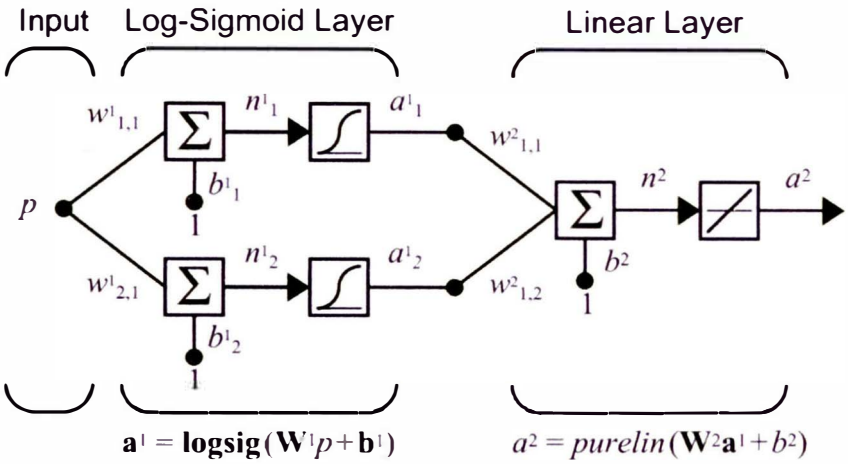


Figura 2.17. Ejemplo de red de aproximación de funciones, *[80]*.

2.10 Entrenamiento de una Red Neuronal Artificial multicapa

2.10.1 Algoritmo de propagación hacia atrás (Backpropagation Algorithm)

De lo establecido anteriormente en la Figura 2.15, para las RNA multicapa, la salida de una capa se convierte en la entrada a la siguiente capa. Las ecuaciones que describen esta operación son:

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}), \tag{2.24}$$

para $m = 0, 1, \ldots, M-1$, donde M es el número de capas de la RNA. El numero de neuronas en la primera capa recibe las entradas externas:

$$\mathbf{a}^0 = \mathbf{p}. \tag{2.25}$$

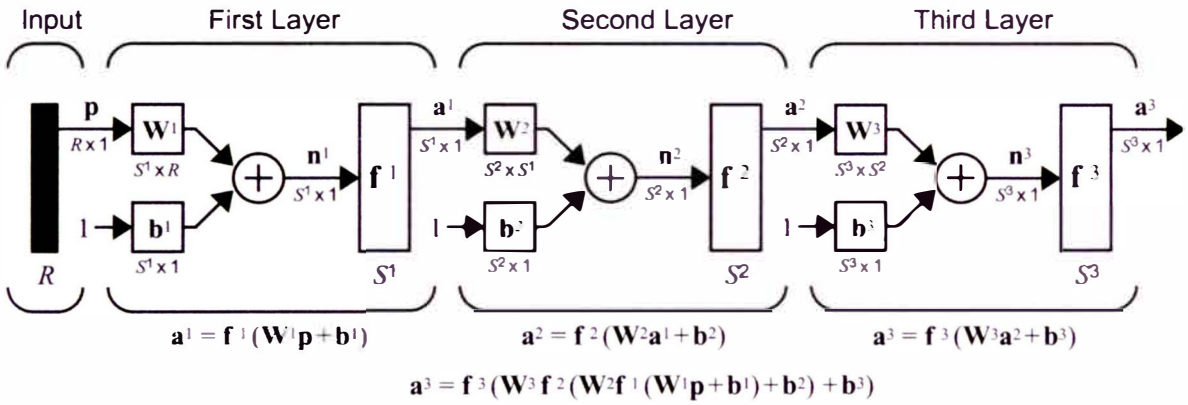


Figura 2.18. RNA de tres capas, notación abreviada, *[81]*.

Índice de desempeño o performance:

$$F(x) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})], \tag{2.26}$$

Aproximación de índice de performance:

$$\hat{F}(\mathbf{x}) = \mathbf{e}^T(k)\mathbf{e}(k) = \big(\mathbf{t}(k)-\mathbf{a}(k)\big)^T\big(\mathbf{t}(k)-\mathbf{a}(k)\big). \tag{2.27}$$

Sensibilidad

$$\mathbf{S}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_1^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_1^m} \end{bmatrix}, \tag{2.28}$$

Propagación hacia adelante,

$$\mathbf{a}^0 = \mathbf{p} \tag{2.29}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1});\ m = 0,1,2,\dots,M-1 \tag{2.30}$$

$$\mathbf{a} = \mathbf{a}^M \tag{2.31}$$

Propagación hacia atrás: Backpropagation,

$$\mathbf{S}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}) \tag{2.32}$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}\ ;\ m = M-1,\dots,2,1 \tag{2.33}$$

donde

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots 0 & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \tag{2.34}$$

$$\dot{f}^m(n_j^m) = \frac{\delta f^m(n_j^m)}{\delta n_j^m} \tag{2.35}$$

Actualización de pesos:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{2.36}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m \tag{2.37}$$

2.10.2 Algoritmo Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt es una variación del método de Newton que fue diseñado para minimizar las funciones que son sumas cuadráticas de otras funciones no lineales. Esto es adecuado para el entrenamiento de redes neuronales artificiales donde el índice de rendimiento es el error cuadrático medio.

Comencemos considerando la forma del método de Newton donde el índice de rendimiento es una suma de cuadrados:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k, \tag{2.38}$$

donde $\mathbf{A}_k \equiv \nabla^2\big(F(\mathbf{x})\big)\big|_{\mathbf{x}=\mathbf{x}_k}$ y $\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$.

Si asumimos que F(**x**) es una función de suma de cuadrados:

$$F(\mathbf{x}) = \sum_{i=1}^{N} v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x})\mathbf{v}(\mathbf{x}), \tag{2.39}$$

entonces el elemento j de la gradiente pude ser:

$$[\nabla F(\mathbf{x})]_j = \frac{\partial F(\mathbf{x})}{\partial x_j} = 2\sum_{i=1}^{N} v_i(\mathbf{x})\frac{\partial v_i(\mathbf{x})}{\partial x_j}. \tag{2.40}$$

La gradiente puede por lo tanto ser escrita en forma matricial:

$$\nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}), \tag{2.41}$$

donde

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1(\mathbf{x})}{\partial x_1} & \frac{\partial v_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial v_2(\mathbf{x})}{\partial x_1} & \frac{\partial v_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_N(\mathbf{x})}{\partial x_1} & \frac{\partial v_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_N(\mathbf{x})}{\partial x_n} \end{bmatrix}, \tag{2.42}$$

es la matriz Jacobiana.

Seguidamente se desea encontrar la matriz de Hesse. El elemento k, j de la matriz de Hesse sería:

$$[\nabla^2 F(\mathbf{x})]_{k,j} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_j} = 2\sum_{i=1}^{N}\left\{\frac{\partial v_i(\mathbf{x})}{\partial x_k}\frac{\partial v_i(\mathbf{x})}{\partial x_j} + v_i(\mathbf{x})\frac{\partial^2 v_i(\mathbf{x})}{\partial x_k \partial x_j}\right\}. \tag{2.43}$$

La matriz de Hesse puede expresarse en forma matricial:

$$\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x}), \tag{2.44}$$

donde

$$\mathbf{S}(\mathbf{x}) = \sum_{i=1}^{N} v_i(\mathbf{x})\nabla^2 v_i(\mathbf{x}). \tag{2.45}$$

Y si asumimos que $\mathbf{S}(\mathbf{x})$ es pequeño, podemos aproximar la matriz de Hesse como

$$\nabla^2 F(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}). \tag{2.46}$$

Si sustituimos la ecuación (2.46) y la ecuación (2.41) en la ecuación (2.38), obtenemos el método Gauss-Newton:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k, \tag{2.47}$$

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k - [2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}2\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \\ &= \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k).\end{aligned} \tag{2.48}$$

No te que la ventaja de Gauss-Newton sobre el método estándar de Newton es que no requiere el cálculo de las segundas derivadas.

Un problema con el método Gauss-Newton es que la matriz $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ puede ser no invertible. Esto se puede superar mediante el uso de la siguiente modificación a la matriz aproximada de Hesse:

$$\mathbf{G} = \mathbf{H} + \mu\mathbf{I}. \tag{2.49}$$

Para ver cómo esta matriz se puede hacer invertible, suponga que los valores propios y los vectores propios de $\mathbf{H}$ son $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ y $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\}$. Luego,

$$\mathbf{G}\mathbf{z}_i = [\mathbf{H} + \mu\mathbf{I}]\mathbf{z}_i = \mathbf{H}\mathbf{z}_i + \mu\mathbf{z}_i = \lambda_i\mathbf{z}_i + \mu\mathbf{z}_i = (\lambda_i + \mu)\mathbf{z}_i. \tag{2.50}$$

Por lo tanto, los vectores propios de G son similares que los valores propios de H, y los valores propios de $\mathbf{G}$ es $(\lambda_i + \mu)$. $\mathbf{G}$ puede hacerse definido positivo al aumentar μ hasta $(\lambda_i + \mu) > 0$ para todo i, y por lo tanto la matriz será invertible.

Esto nos lleva al algoritmo de *Levenberg-Marquardt*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k). \tag{2.51}$$

o

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k). \tag{2.52}$$

Este algoritmo tiene la característica muy útil de que a medida que aumenta μ_k se acerca al algoritmo de descenso más pronunciado (steepest descent) con una tasa de aprendizaje pequeña:

$$\mathbf{x}_{k+1} \cong \mathbf{x}_k - \frac{1}{\mu_k}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - \frac{1}{2\mu_k}\nabla F(\mathbf{x}), \qquad (2.53)$$

para valores grandes de μ_k, mientras que si μ_k decrece a cero el algoritmo se asemeja al algoritmo Gauss-Newton.

El algoritmo inicia con el ajuste de μ_k en un valor pequeño (por ejemplo $\mu_k = 0.01$). Si un paso no produce un valor menor para $F(\mathbf{x})$, entonces el paso se repite con μ_k multiplicado por algún factor $\vartheta > 1$ (por ejemplo $\vartheta = 10$). Eventualmente $F(\mathbf{x})$ debería disminuir, ya que estaríamos dando un pequeño paso en la dirección del descenso más empinado. Si un paso produce un valor menor para $F(\mathbf{x})$, entonces μ_k se divide por ϑ para el siguiente paso, de modo que el algoritmo se acerque a Gauss-Newton, lo que debería proporcionar una convergencia más rápida. El algoritmo proporciona un buen equilibrio entre la rapidez del método de Newton y la convergencia garantizada del descenso más empinado.

Ahora vamos a ver cómo podemos aplicar el algoritmo Levenberg-Marquardt al problema de entrenamiento de una RNA multicapa. El índice de desempeño de entrenamiento de una RNA multicapa es el error cuadrático medio. Si cada objetivo (target) ocurre con igual probabilidad, el error cuadrático medio es proporcional a la suma de errores cuadráticos sobre los $\mathbf{Q}$ objetivos (targets) del conjunto de datos de entrenamiento:

$$F(\mathbf{x}) = E_D = \sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q)^T(\mathbf{t}_q - \mathbf{a}_q) = F(\mathbf{x}) = E_D = \sum_{q=1}^{Q}\mathbf{e}_q^T\mathbf{e}_q$$

$$= \sum_{q=1}^{Q}\sum_{j=1}^{S^M}(e_{j,q})^2 = \sum_{q=1}^{N}(v_i)^2, \qquad (2.54)$$

donde $e_{j,q}$ es el elemento j del error q del par entrada/objetivo (input/target).

La ecuación (2.54) es equivalente al índice de desempeño, ecuación (2.39), para la cual fue diseñado Levenberg-Marquardt. Por lo tanto, debería ser sencillo adaptar el algoritmo para el entrenamiento de la RNA. Resulta que esto es cierto en concepto, pero requiere un poco de cuidado en la elaboración de los detalles.

Cálculo de la matriz Jacobiana

El paso clave en el algoritmo de Levenberg-Marquardt es el cálculo de la matriz jacobiana. Para realizar este cálculo usaremos una variación del algoritmo de retro propagación. Recordar que en el procedimiento estándar de retro propagación se calcula las derivadas de los errores al cuadrado, con respecto a los pesos y bias de la RNA. Para crear la matriz jacobiana necesitamos calcular las derivadas de los errores, en lugar de las derivadas de los errores al cuadrado.

Antes de presentar el procedimiento para calcular el Jacobiano, echemos un vistazo más de cerca a su forma (Ecuación (2.42)). Tenga en cuenta que el vector de error es

$$\mathbf{v}^T = [\, v_1 \quad v_2 \quad \cdots \quad v_N] = [\, e_{1,1} \quad e_{2,1} \quad \cdots \quad e_{S^M,1} \quad e_{1,2} \quad \cdots \quad e_{S^M,Q}], \quad (2.55)$$

el vector de parámetros (pesos y bias) es:

$$\mathbf{x}^T = [\, x_1 \quad x_2 \quad \cdots \quad x_n]$$

$$= \left[\, w^1_{1,1} \quad w^1_{1,2} \quad \cdots \quad w^1_{S^1,R} \quad b^1_1 \quad \cdots \quad b^1_{S^1} \quad w^2_{1,1} \quad \cdots \quad b^M_{S^M}\right], \quad (2.56)$$

$$N = Q \times S^M, \quad (2.57)$$

$$n = S^1(R+1) + S^2(S^1+1) + \cdots + S^M(S^{M-1}+1), \quad (2.58)$$

N : candad de errores de entrenamiento de la RNA.

S^M : cantidad de neuronas en la capa de salida.

Q : cantidad de pares de vectores de entrada/salida.

n : cantidad de parámetros de la RNA.

Por lo tanto, si hacemos la sustitución en la ecuación (2.42), la matriz Jacobiana para el entrenamiento de redes neuronales multicapa puede escribirse [82]

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w^1_{1,1}} & \frac{\partial e_{2,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{2,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{2,1}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w^1_{1,1}} & \frac{\partial e_{S^M,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{S^M,1}}{\partial w^1_{S^1,1}} & \frac{\partial e_{S^M,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}. \quad (2.59)$$

Los términos en esta matriz jacobiana se pueden calcular mediante una simple modificación al algoritmo de retro propagación.

La retro propagación estándar calcula los términos como [82]

$$\frac{\partial \hat{F}(s)}{\partial x_l} = \frac{\partial \mathbf{e}_q^T \mathbf{e}_q}{\partial x_l}. \quad (2.60)$$

2.11 Preprocesamiento de datos

2.11.1 Normalización

El objetivo principal de la etapa de preprocesamiento de datos es facilitar el entrenamiento de la RNA. El preprocesamiento de datos consta de pasos tales como normalización, transformaciones no lineales, extracción de características, codificación de entradas/objetivos discretos, manejo de datos faltantes, etc. La idea es realizar un

procesamiento preliminar de los datos para facilitar el entrenamiento de la RNA para extraer la información relevante.

Por ejemplo, en las RNA multicapa, las funciones de transferencia sigmoidales se usan a menudo en las capas ocultas. Estas funciones se saturan esencialmente cuando la entrada neta es mayor que tres $(\exp(-3) \cong 0.05)$. No queremos que esto suceda al comienzo del proceso de entrenamiento, porque el gradiente será muy pequeño. En la primera capa, la entrada neta es un producto de la entrada multiplicada por el peso más el sesgo. Si la entrada es muy grande, entonces el peso debe ser pequeño para evitar que la función de transferencia se sature. Es una práctica estándar normalizar las entradas antes de aplicarlas a la RNA. De esta manera, la inicialización de los pesos de la RNA a pequeños valores aleatorios garantiza que el producto de peso-entrada será pequeño. Además, cuando los valores de entrada están normalizados, las magnitudes de los pesos tienen un significado consistente. Esto es especialmente importante cuando se usa la regularización Bayesiana. La regularización requiere que los valores de peso sean pequeños. Sin embargo, "pequeño" es un término relativo; Si los valores de entrada son muy pequeños, necesitamos grandes pesos para producir una entrada neta significativa. La normalización de las entradas aclara el significado de los pesos "pequeños".

Hay dos métodos estándar para la normalización. El primer método normaliza los datos para que se encuentren en un rango estándar, generalmente de -1 a 1. Esto se puede hacer con

$$\mathbf{p}^n = 2(\mathbf{p} - \mathbf{p}^{min})./(\mathbf{p}^{max} - \mathbf{p}^{min}) - 1 \qquad (2.61)$$

donde $\mathbf{p}^{min}$ es el vector que contiene los valores mínimos de cada elemento de los vectores de entrada en el conjunto de datos, $\mathbf{p}^{max}$ contiene los valores máximos, ./ representa una división elemento por elemento de los dos vectores, y $\mathbf{p}^n$ es el vector de entrada normalizado resultante.

Un procedimiento alternativo de normalización es ajustar los datos para que tengan una media y varianza especificadas, generalmente 0 y 1. Esto se puede hacer con la transformación

$$\mathbf{p}^n = (\mathbf{p} - \mathbf{p}^{mean})./(\mathbf{p}^{std}) \tag{2.62}$$

donde $\mathbf{p}^{mean}$ es el promedio de los vectores de entrada en el conjunto de datos, y $\mathbf{p}^{std}$ es el vector que contiene las desviaciones estándar de cada elemento de los vectores de entrada.

En general, el paso de normalización se aplica tanto a los vectores de entradas como a los vectores de objetivos en el conjunto de datos.

Además de la normalización, que implica una transformación lineal, las *transformaciones no lineales* a veces también se realizan como parte de la etapa de preprocesamiento. A diferencia de la normalización, que es un proceso estándar que puede aplicarse a cualquier conjunto de datos, estas transformaciones no lineales son casos específicos. Por ejemplo, muchas variables económicas muestran una dependencia logarítmica. En ese caso, podría ser apropiado tomar el logaritmo de los valores de entrada antes de aplicarlos a la RNA. Otro ejemplo es la simulación de dinámica molecular, en la que las fuerzas atómicas se calculan como funciones de las distancias entre los átomos. Como se sabe que las fuerzas están inversamente relacionadas con las distancias, podríamos realizar la transformación recíproca en las entradas, antes de aplicarlas a la RNA. Esto representa una forma de incorporar el conocimiento previo en el entrenamiento de las RNA. Si la transformación no lineal se elige inteligentemente, puede hacer que el entrenamiento de la RNA sea más eficiente. El preprocesamiento descargará parte del trabajo requerido de la RNA para encontrar la transformación subyacente entre las entradas y los objetivos.

Otro paso de preprocesamiento de datos se llama *extracción de características*. Esto generalmente se aplica a situaciones en las que la dimensión de los vectores de entrada sin procesar es muy grande y los componentes del vector de entrada son redundantes. La idea de la extracción de características es reducir la dimensión del espacio de entrada calculando un pequeño conjunto de características de cada vector de entrada y utilizando las características como la entrada a la RNA. Por ejemplo, las RNA se pueden utilizar para analizar señales de electrocardiograma (EKG) para identificar problemas cardíacos. El EKG puede involucrar 12 o 15 señales (derivaciones) medidas durante varios minutos a una alta tasa de muestreo. Esto es demasiada información para aplicar directamente a la RNA. En su lugar, extraeríamos ciertas características de la señal de EKG, como los intervalos de tiempo promedio entre ciertas formas de onda, amplitudes promedio de ciertas ondas, etc.

También hay ciertos métodos de extracción de características de propósito general. Uno de ellos es el método de *análisis de componentes principales* (PCA). Este método transforma los vectores de entrada originales para que los componentes de los vectores transformados no estén correlacionados. Además, los componentes del vector transformado están ordenados de tal manera que el primer componente tiene la mayor varianza, el segundo componente tiene la siguiente mayor variación, etc. Generalmente, mantenemos solo los primeros componentes del vector transformado, que representan la mayor parte de la varianza en el vector original. Esto da como resultado una gran reducción en la dimensión del vector de entrada, si los componentes originales están altamente correlacionados. El inconveniente de usar PCA es que solo considera las relaciones lineales entre los componentes del vector de entrada. Al reducir la dimensión mediante una transformación lineal, es posible que perdamos información no lineal. Dado que el propósito principal del uso de RNA es obtener el poder de sus capacidades de mapeo no lineal, debemos tener cuidado al usar *componentes principales* para reducir la dimensión

de entrada antes de aplicar las entradas a la RNA. Existe una versión no lineal de PCA, llamada kernel PCA[7], [83], [3], [84].

2.12 Inicialización de parámetros

Tradicionalmente, los pesos de una RNA se establecían en pequeños números aleatorios (por ejemplo, entre -1 y $+1$).

La inicialización de los pesos de las RNA es un campo de estudio ya que la cuidadosa inicialización de la RNA puede acelerar el proceso de aprendizaje.

El software de Redes Neuronales Keras[8], al momento de escribir este documento posee las siguientes maneras de inicializar los pesos, [85]:

Zeros	:	Genera tensores inicializados a 0
Ones	:	Genera tensores inicializados a 1
Constant	:	Genera tensores inicializados a un valor constante
RandomNormal	:	Genera tensores con una distribución normal
RandomUniform	:	Genera tensores con una distribución uniforme
TruncatedNormal:	:	Genera una distribución normal truncada
VarianceScaling	:	Capaz de adaptar su escala a la forma de los pesos
Orthogonal	:	Genera una matriz ortogonal aleatoria
Identity	:	Genera la matriz identidad
lecun_uniform	:	Inicializador uniforme LeCun.

[7] En ella Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Mülller describen un método para realizar una forma no lineal de análisis de componentes principales [83].

[8] Keras es una API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow, CNTK o Theano. Fue desarrollado con un enfoque en permitir la experimentación rápida. Poder pasar de la idea al resultado con el menor retraso posible es clave para hacer una buena investigación.

glorot_normal	:	Inicializador normal de Glorot, también llamado inicializador normal de Xavier.
glorot_uniform	:	Inicializador uniforme Glorot, también llamado inicializador uniforme Xavier
he_normal	:	Inicializador normal He
lecun_normal	:	Inicializador normal LeCun.
he_uniform	:	Inicializador de escala de varianza uniforme.

2.12.1 Widrow y Nguyen – inicialización de pesos y bias

Hay otro enfoque para establecer los pesos y bias iniciales para una RNA de dos capas. Fue introducido por Widrow y Nguyen. La idea es establecer la magnitud de los pesos en la primera capa de tal manera que la región lineal de cada función sigmoidea cubra $1/S^1$ del rango de la entrada. Los bias se establecen aleatoriamente, de modo que el centro de cada función sigmoidea cae aleatoriamente en el espacio de entrada. Los detalles del método se muestran a continuación (suponiendo que las entradas a la RNA se hayan normalizado a valores entre -1 y 1), [86].

Se establece la fila i de $\mathbf{W}^1$, ${}_i\mathbf{W}^1$, para que tenga una dirección aleatoria y una magnitud de

$$\|{}_i\mathbf{W}^1\| = 0.7(S^1)^{1/R}. \tag{2.63}$$

El bias b_i se establece en un valor aleatorio uniforme entre $-\|{}_i\mathbf{W}^1\|$ y $\|{}_i\mathbf{W}^1\|$. [87]

$$-\|{}_i\mathbf{W}^1\| \leq b_i \leq \|{}_i\mathbf{W}^1\| \tag{2.64}$$

2.13 Generalización

Una de las cuestiones clave en el diseño de una RNA multicapa es determinar la cantidad de neuronas que se deben utilizar.

En efecto, se sabe que, si el número de neuronas es demasiado grande, la RNA se adaptará a los datos de entrenamiento. Esto significa que el error en los datos de entrenamiento será muy pequeño, pero la RNA no funcionará tan bien cuando se presenten datos nuevos. Una RNA que generalice bien se desempeñará tan bien en los datos nuevos como en los datos de entrenamiento.

La complejidad de una RNA neuronal está determinada por el número de parámetros libres que tiene (pesos y sesgos), que a su vez está determinado por el número de neuronas. Si una RNA es demasiado compleja para un conjunto de datos dado, es probable que se ajuste en exceso y tenga una generalización deficiente.

En la siguiente sección veremos que podemos ajustar la complejidad de una RNA para que se ajuste a la complejidad de los datos. Además, esto se puede hacer sin cambiar el número de neuronas. Podemos ajustar el número efectivo de parámetros libres sin cambiar el número real de parámetros libres.

2.13.1 Planteamiento del problema de Generalización

Comencemos nuestra discusión de la generalización definiendo el problema. Comenzamos con un ejemplo de conjunto de entrenamiento de entradas de RNA y sus correspondientes salidas objetivo:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots \{\mathbf{p}_Q, \mathbf{t}_Q\}. \tag{2.65}$$

Para nuestro desarrollo del concepto de generalización, asumiremos que las salidas de destino se generan por:

$$\mathbf{t}_q = \mathbf{g}(\mathbf{p}_q) + \varepsilon_q, \tag{2.66}$$

donde $\mathbf{g}(.)$ es alguna función desconocida, y ε_q es una fuente de ruido aleatoria, independiente y de media cero. Nuestro objetivo de entrenamiento será producir una RNA que se aproxime $\mathbf{g}(.)$, ignorando el ruido.

El índice de rendimiento estándar para el entrenamiento de la RNA es la suma del error cuadrático en el conjunto de entrenamiento:

$$F(\mathbf{x}) = E_D = \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q), \tag{2.67}$$

donde $\mathbf{a}_q$ es la salida de la RNA para la entrada $\mathbf{p}_q$. Estamos utilizando la variable E_D para representar la suma del error cuadrático en los datos de entrenamiento, porque más adelante modificaremos el índice de rendimiento para incluir un término adicional.

2.13.2 Regularización

En este método, modificamos el índice de rendimiento de error de suma al cuadrado de la ecuación (2.67) para incluir un término que penaliza la complejidad de la RNA. Este concepto fue introducido por Tikhonov, quien añadió un término de pena o regularización, que incluía las derivadas de la función de aproximación (RNA en nuestro caso), lo que obligó a que la función resultante fuera suave. Bajo ciertas condiciones, este término de regularización se puede escribir como la suma de los cuadrados de los pesos de la RNA, como en

$$F(\mathbf{x}) = \beta E_D + \alpha E_w = \beta \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (t_q - \mathbf{a}_q) + \alpha \sum_{i=1}^{n} x_i^2, \tag{2.68}$$

donde la proporción β/α controla la complejidad efectiva de la solución de RNA. Cuanto mayor sea esta relación, más suave será la respuesta de la RNA. (Tenga en cuenta que podríamos haber usado un solo parámetro aquí, pero los desarrollos en secciones posteriores requerirán dos parámetros).

¿Por qué queremos penalizar la suma cuadrática de los pesos, y en qué es similar esto a reducir el número de neuronas?

Considere la Figura 2.19,

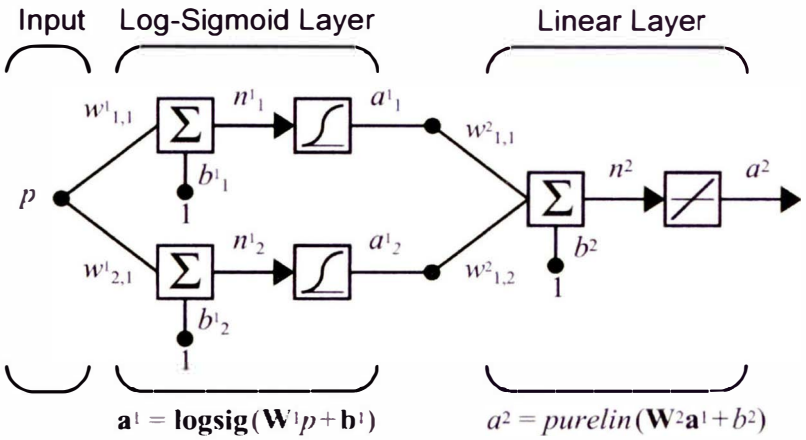
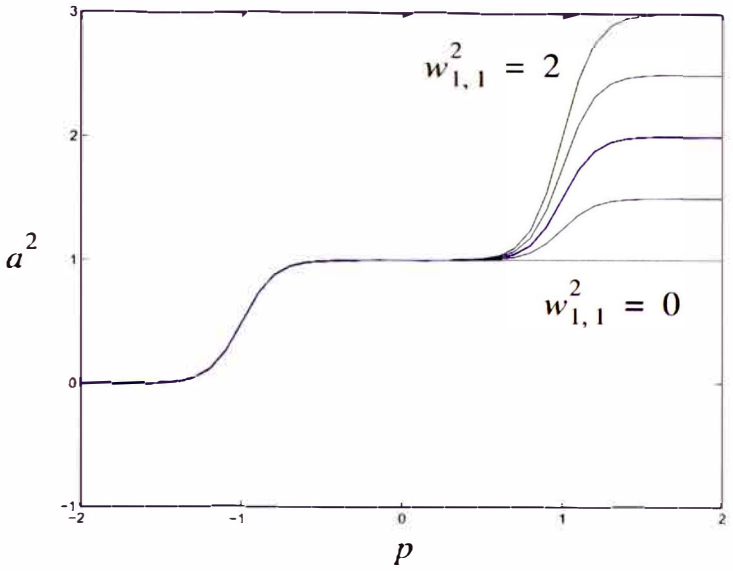


Figura 2.19. Example Function Approximation Network, *[88]*.

Al aumentar un peso aumenta la pendiente de la función de RNA. Puede ver este efecto en la **¡Error! No se encuentra el origen de la referencia.**, donde hemos cambiado el peso $w_{1,1}^2$ de 0 a 2. Cuando los pesos son grandes, la función creada por la RNA puede tener grandes pendientes y, por lo tanto, es más probable que se sobre ajuste a los datos de entrenamiento. Si restringimos los pesos para que sean pequeños, entonces la función de RNA creará una interpolación suave a través de los datos de entrenamiento, como si la RNA tuviera una pequeña cantidad de neuronas.



La clave para el éxito del método de regularización para producir una RNA que generalice bien es la elección correcta de la relación de regularización α/β. La Figura 2.20 ilustra el efecto de cambiar esta relación. Aquí se ha entrenado una RNA 1-20-1 en 21 muestras con ruido de una onda sinusoidal.

En la Figura 2.20, la línea azul representa la verdadera función, y los círculos grandes sin re rellenar representan los datos ruidosos. La curva negra representa la respuesta de RNA entrenada, y los círculos más pequeños rellenos con cruces representan la respuesta de RNA en los puntos de entrenamiento. En la Figura 2.20, podemos ver que la relación α/β produce el mejor ajuste para la función verdadera. Para relaciones más grandes que esto, la respuesta de la RNA es demasiado suave, y para relaciones más pequeñas que esto, la RNA se sobre ajusta.

Existen varias técnicas para configurar el parámetro de regularización. Un enfoque es utilizar un conjunto de validación; el parámetro de regularización se establece para minimizar el error cuadrático en el conjunto de validación. En las siguientes dos secciones describiremos una técnica diferente para configurar automáticamente el parámetro de regularización. Se llama regularización Bayesiana.

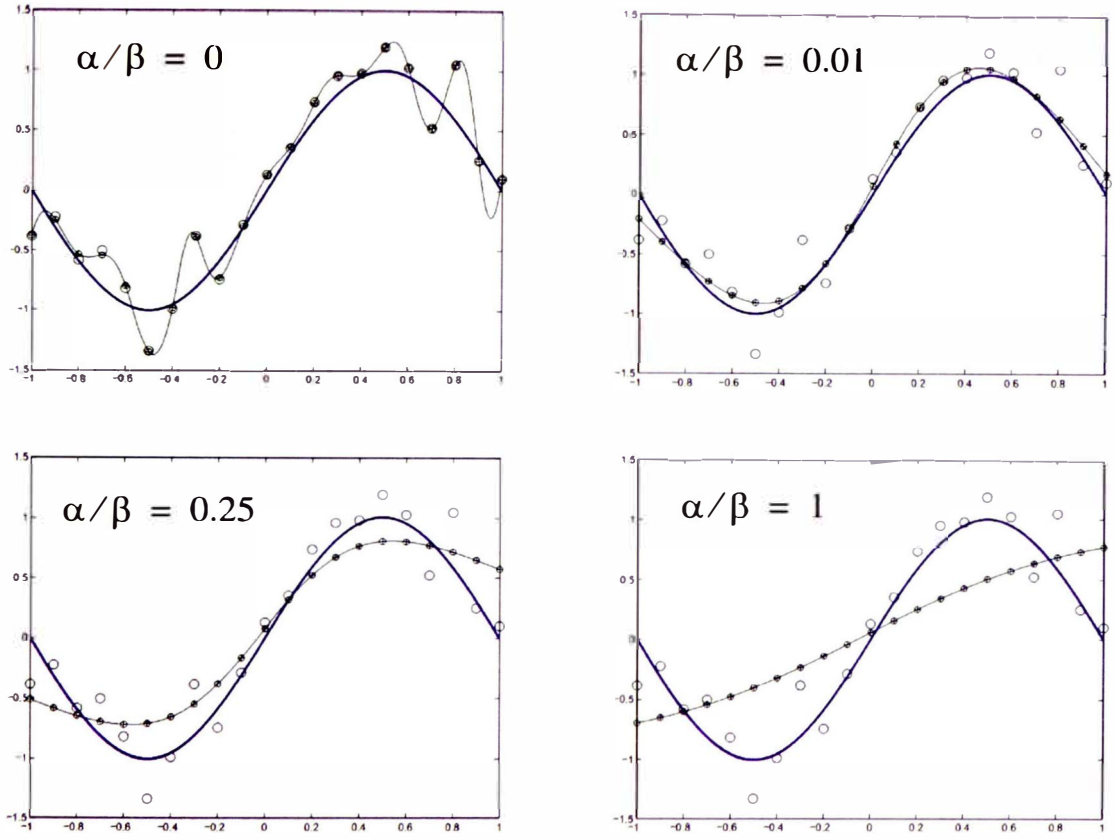


Figura 2.20. Efecto de la relación de regularización, *[90]*.

2.13.3 Análisis Bayesiano

Thomas Bayes fue un ministro presbiteriano que vivió en Inglaterra en el siglo XVIII. También fue un matemático aficionado. Su obra más importante fue publicada después de su muerte. En él, presentó lo que ahora se conoce como el teorema de Bayes. El teorema establece que si se tiene dos eventos aleatorios A y B, entonces, la probabilidad condicional de la ocurrencia de A, dada la ocurrencia de B, se puede calcular como:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.69}$$

La ecuación (2.69) se denomina regla de Bayes. Cada uno de los términos en esta expresión tiene un nombre por el cual es comúnmente referido. $P(A)$ se llama probabilidad *previa*. Nos dice lo que sabemos sobre A antes de saber el resultado de B. $P(A|B)$ se llama probabilidad *posterior*. Esto nos dice lo que sabemos acerca de A después de que

sabemos acerca de B. $P(B|A)$ es la probabilidad condicional de B dado A. Normalmente este término viene dado por nuestro conocimiento del sistema que describe la relación entre B y A. $P(B)$ es la probabilidad marginal del evento B, y actúa como un factor de normalización en la regla de Bayes.

Para ilustrar cómo se puede usar la regla de Bayes, considere la siguiente situación médica. Suponga que el 1% de la población tiene una enfermedad determinada. Hay una prueba que se puede realizar para detectar la presencia de esta enfermedad. La prueba es 80% precisa para detectar la enfermedad en personas que la tienen. Sin embargo, el 10% de las veces, alguien sin la enfermedad registrará una prueba positiva. Si toma la prueba y se registra positivo, su pregunta sería: ¿Cuál es la probabilidad de que realmente tenga la enfermedad? La mayoría de nosotros (incluida la mayoría de los médicos, como se ha demostrado en muchos estudios), supondría que la probabilidad es muy alta, teniendo en cuenta que la prueba es 80% precisa para detectar la enfermedad en una persona enferma. Sin embargo, este no es el caso, y la regla de Bayes puede ayudarnos a superar esta falta de intuición, cuando se trata de probabilidad.

Deje que A represente el evento de que *tiene la enfermedad*. Deje que B represente el evento de que *tiene un resultado positivo en la prueba*. Entonces podemos usar la regla de Bayes para encontrar $P(A|B)$, que es la probabilidad de que tenga la enfermedad, dado que tiene una prueba positiva. Sabemos que la probabilidad previa $P(A)$ sería 0.01, porque el 1% de la población tiene la enfermedad. $P(B|A)$ es 0.8, porque la prueba es 80% precisa para detectar la enfermedad en personas que la tienen. (Tenga en cuenta que esta probabilidad condicional se basa en nuestro conocimiento del procedimiento de prueba y su precisión). Para usar la regla de Bayes, necesitamos un término más, que es P (B). Esta es la probabilidad de obtener una prueba positiva, ya sea que tenga o no la enfermedad. Esto se puede obtener agregando la probabilidad de tener una prueba positiva cuando tiene la enfermedad a la probabilidad de tener una prueba positiva cuando no tiene la enfermedad:

$$P(B) = P(A \cap B) + P(\bar{A} \cap B) = P(B|A)P(A) + P(B|\bar{A})P(\bar{A}) \quad (2.70)$$

donde hemos usado la definición de probabilidad condicional:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, o \;\; P(A \cap B) = P(B|A)P(A). \quad (2.71)$$

Si conectamos nuestras probabilidades conocidas en la ecuación (2.70), encontramos

$$P(B|A) = 0.8 \times 0.01 + 0.1 \times 0.99 = 0.107 \quad (2.72)$$

donde $P(B|\bar{A})$ es 0.1, porque el 10% de las personas sanas registran una prueba positiva.

Ahora podemos usar la regla de Bayes para encontrar la probabilidad posterior $P(A|B)$:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{0.8 \times 0.01}{0.107} = 0.0748 \quad (2.73)$$

Esto nos dice que incluso si obtiene una prueba positiva, solo tiene un 7,5% de posibilidades de tener la enfermedad. Para la mayoría de nosotros, este resultado no es intuitivo.

La clave de la regla de Bayes es la probabilidad previa P (A). En este caso, las probabilidades previas de tener la enfermedad eran solo de 1 en 100. Si este número hubiera sido mucho mayor, entonces nuestra probabilidad posterior $P(A|B)$ también habría aumentado significativamente. Cuando se utiliza la regla de Bayes, es importante que la probabilidad previa $P(A)$ refleje con precisión nuestro conocimiento previo.

En la siguiente sección, aplicaremos el análisis bayesiano al entrenamiento de RNA multicapa. La ventaja de los métodos Bayesianos es que podemos insertar conocimiento previo a través de la selección de la probabilidad previa. Para el entrenamiento de las RNA, asumiremos previamente que la función que estamos aproximando es fluida. Esto significa que los pesos no pueden ser demasiado grandes, como se demostró en la **¡Error! No se**

encuentra el origen de la referencia.. La clave consistirá en incorporar este conocimiento previo en una elección apropiada para la probabilidad previa.

En el apartado 2.13.4 se detalla la teoría original de interpolación o regularización bayesiana desarrollado por David J.C. MacKay. En el apartado 2.13.5 se detalla la teoría de Regularización Bayesiana (específica a RNA) tomando como base la teoría original de Interpolación Bayesiana desarrollado por David J.C. MacKay

2.13.4 Interpolación Bayesiana general (David J.C. MacKay)

Resumen

Aunque el análisis bayesiano ha estado en uso desde Laplace, el método bayesiano de comparación de modelos solo se ha desarrollado recientemente en profundidad. En este capítulo, el enfoque bayesiano para la regularización y la comparación de modelos se demuestra al estudiar el problema de inferencia de interpolar datos ruidosos. Los conceptos y métodos descritos son bastante generales y pueden aplicarse a muchos otros problemas de modelado de datos.

Las constantes de regularización se establecen examinando su distribución de probabilidad posterior (*posterior probability*). Los regularizadores alternativos (previos (*priors)*) y los conjuntos de bases alternativas se comparan objetivamente evaluando la evidencia (*evidence*) para ellos. La "navaja de Occam" se incorpora automáticamente en este proceso.

La forma en que Bayes infiere los valores de regularización de constantes y niveles de ruido tiene una interpretación elegante en términos del número efectivo de parámetros determinados por el conjunto de datos. Este marco se debe a Gull and Skilling, [91].

2.13.4.1 Modelamiento de datos y la Navaja de Occam (Occam's razor)

En ciencia, una tarea central es desarrollar y comparar modelos para tener en cuenta los datos que se recopilan. En particular, esto es cierto en los problemas de aprendizaje, clasificación de patrones, interpolación y agrupamiento. En la tarea de modelado de datos intervienen dos niveles de inferencia (Figura 2.21). En el primer nivel de inferencia, asumimos que uno de los modelos que inventamos es verdadero, y ajustamos ese modelo a los datos. Típicamente un modelo incluye algunos parámetros libres; ajustar el modelo a los datos implica inferir qué valores deberían tomar esos parámetros, dados los datos. Los resultados de esta inferencia a menudo se resumen por los valores de parámetros más probables y las barras de error en esos parámetros. Esto se repite para cada modelo. El segundo nivel de inferencia es la tarea de comparación de modelos. Aquí, deseamos comparar los modelos a la luz de los datos y asignar algún tipo de preferencia o clasificación a las alternativas,[9] [92].

[9] Tenga en cuenta que ambos niveles de *inferencia* (*inference*) son distintos de la *teoria de la decisión* (*decision theory*). El objetivo de la inferencia es, dado un espacio de hipótesis definido y un conjunto de datos particular, asignar probabilidades a las hipótesis. La teoría de la decisión suele elegir entre acciones alternativas en función de estas probabilidades para minimizar la expectativa de una "función de pérdida". Este capítulo se refiere solo a la inferencia y no hay funciones de pérdida o utilidades involucradas.

Otro concepto erróneo se refiere a la relación entre la comparación del modelo y la elección del modelo. Al enfatizar el método bayesiano de comparación de modelos, no quiero decir que la acción correcta sea elegir el modelo más probable. La "manera correcta" de hacer predicciones bayesianas es integrarse en nuestro espacio modelo.

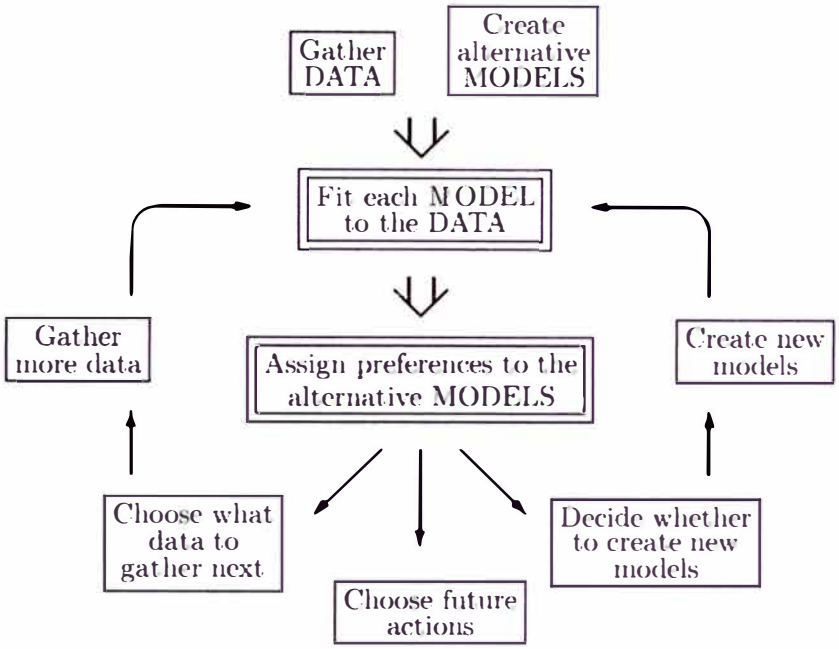


Figura 2.21. Donde la inferencia bayesiana encaja en el proceso de modelado de datos, *[93]*.

La Figura 2.21 ilustra una abstracción de la parte del proceso científico en el que se recopilan y modelan los datos. En particular, esta figura se aplica a la clasificación de patrones, el aprendizaje, la interpolación, etc. Los dos cuadros de doble marco denotan los dos pasos que implican *inferencia*. Solo en esos dos pasos se puede usar la regla de Bayes. Bayes no te dice cómo inventar modelos, por ejemplo.

El primer cuadro, "ajustar cada modelo a los datos" (fitting each model to the data), es la tarea de inferir a qué parámetros del modelo se les puede dar el modelo y los datos. Bayes puede usarse para encontrar los valores de parámetro más probables y barras de error en esos parámetros. El resultado de aplicar Bayes a este problema es a menudo poco diferente de las respuestas dadas por las estadísticas ortodoxas.

La segunda tarea de inferencia, la comparación de modelos a la luz de los datos, es donde Bayes se encuentra en una clase propia. Este segundo problema de inferencia requiere una navaja de Occam cuantitativa para penalizar los modelos demasiado complejos. Bayes puede asignar preferencias objetivas a los modelos alternativos de una manera que incorpore automáticamente la navaja de Occam.

Por ejemplo, considere la tarea de interpolar un conjunto de datos ruidoso. El conjunto de datos podría interpolarse utilizando un modelo de splines, utilizando funciones de base radial, utilizando polinomios o utilizando RNA feedforward. En el primer nivel de inferencia, tomamos cada modelo individualmente y encontramos el mejor ajuste interpolante para ese modelo. En el segundo nivel de inferencia, queremos clasificar los modelos alternativos y establecer para nuestro conjunto de datos en particular que, por ejemplo, "las splines son probablemente el mejor modelo de interpolación" o "si el interpolante se modela como un polinomio, probablemente debería ser un cúbico"

Los métodos bayesianos son capaces de resolver de manera consistente y cuantitativa ambas tareas de inferencia. Existe un mito popular que establece que los métodos bayesianos solo difieren de los métodos estadísticos ortodoxos (también conocidos como 'frecuentistas' o 'teoría de muestreo') mediante la inclusión de antecedentes subjetivos que son arbitrarios y difíciles de asignar, y generalmente no hacen mucha diferencia a las conclusiones. Es cierto que, en el primer nivel de inferencia, los resultados de Bayesianos a menudo diferirán poco del resultado de un ataque ortodoxo. Lo que no se aprecia ampliamente es cómo Bayes realiza el segundo nivel de inferencia. Es aquí donde los métodos bayesianos son totalmente diferentes de los métodos de la teoría del muestreo ortodoxo. De hecho, cuando la regresión y la estimación de densidad se discuten en la mayoría de los textos estadísticos la tarea de comparación de modelos se ignora virtualmente; no existe un método ortodoxo general para resolver este problema, [92].

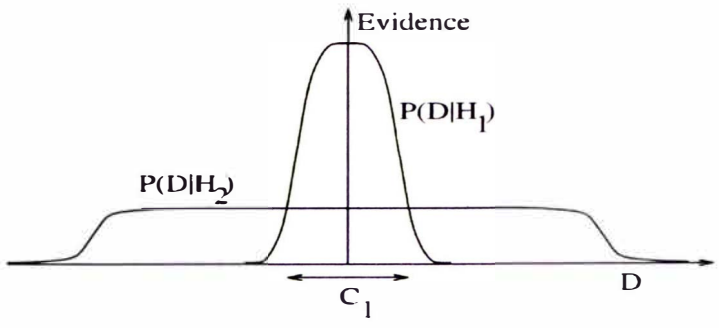


Figura 2.22. Por qué Bayes encarna la navaja de Occam, *[94]*.

Esta Figura 2.22 proporciona la intuición básica de por qué se penalizan los modelos complejos. El eje horizontal representa el espacio de posibles conjuntos de datos D. Los modelos de recompensas de la regla de Bayes en proporción a cuánto predijeron (predicted) los datos que ocurrieron. Estas predicciones se cuantifican por una distribución de probabilidad normalizada en D. En este trabajo, esta probabilidad de los datos dados el modelo $Hi, P(D|H_i)$, se llama evidencia para H_i.

Un modelo simple H_1 solo realiza un rango limitado de predicciones, que se muestra por $P(D|H_1)$; Un modelo H_2 más potente, que tiene, por ejemplo, más parámetros libres que H_1, puede predecir una mayor variedad de conjuntos de datos. Sin embargo, esto significa que H_2 no predice los conjuntos de datos en la región C_1 con tanta fuerza como H_1. Suponga que se han asignado iguales probabilidades previas a los dos modelos. Entonces, si el conjunto de datos cae en la región C_1, el modelo *menos potente* H_1 será el *modelo más probable*, [92].

La comparación de modelos es una tarea difícil porque no es posible elegir simplemente el modelo que mejor se adapte a los datos: los modelos más complejos siempre pueden ajustarse mejor a los datos, por lo que la elección del modelo de máxima probabilidad nos llevaría inevitablemente a modelos inverosímiles sobre parametrizados que generalizan mal. La "navaja de Occam" es el principio que establece que los modelos innecesariamente complejos no deberían preferirse a los más simples. Los métodos bayesianos incorporan automática y cuantitativamente la navaja de Occam, sin la introducción de términos de penalización ad hoc[10]. Los modelos complejos se auto penalizan automáticamente bajo la regla de Bayes. La Figura 2.22 da la intuición básica

[10] Ad hoc: que es apropiado, adecuado o especialmente dispuesto para un determinado fin.

de por qué esto debería esperarse; el resto de este capítulo explorará esta propiedad en profundidad.

Los métodos bayesianos, concebidos simultáneamente por Bayes y Laplace, fueron presentados primero en profundidad por el geofísico de Cambridge Sir Harold Jeffreys[11]. La base lógica para el uso bayesiano de las probabilidades como medidas de plausibilidad fue establecida posteriormente por Cox[12], quien demostró que la inferencia consistente en un espacio de hipótesis cerrada puede mapearse en las probabilidades. Para una revisión general de la filosofía bayesiana, se alienta al lector a leer los excelentes documentos de Jaynes y Loredo, y el texto recientemente reimpreso de Box y Tiao. Desde Jeffreys, el énfasis de la mayoría de la teoría de probabilidad bayesiana ha sido 'utilizar formalmente información previa', es decir, realizar inferencia de una manera que explique el conocimiento previo y la ignorancia que tenemos, que los métodos ortodoxos omiten. Sin embargo, el trabajo de Jeffreys también sentó las bases para la comparación del modelo bayesiano, que no implica un énfasis en la información previa, sino que hace hincapié en obtener la máxima información de los datos. Jeffreys aplicó esta teoría a problemas simples de comparación de modelos en geofísica, por ejemplo, probando si un solo parámetro adicional está justificado por los datos. Desde la década de 1960, los métodos de comparación de modelos de Jeffreys se han aplicado y ampliado en la literatura económica y por un pequeño número de estadísticos. Solo recientemente se ha desarrollado y aplicado este aspecto del análisis bayesiano a problemas más complejos en otros campos, [92].

Este capítulo revisará la comparación del modelo bayesiano, la 'regularización' y la estimación del ruido, estudiando el problema de interpolar datos ruidosos. El marco bayesiano que describiré para estas tareas se debe a Gull and Skilling, que han utilizado

---

[11] Sir Harold Jeffreys, (22 de abril de 1891 - 18 de marzo de 1989, Cambridge), astrónomo y geofísico británico conocido por su amplia variedad de contribuciones científicas.
[12] Richard Threlkeld Cox (5 de agosto de 1898 - 2 de mayo de 1991) fue profesor de física en la Universidad Johns Hopkins, conocido por el teorema de Cox relacionado con los fundamentos de la probabilidad.

métodos bayesianos para lograr el estado del arte en la reconstrucción de imágenes. El mismo enfoque para la regularización también ha sido desarrollado en parte por Szeliski. La comparación del modelo bayesiano también es discutida por Smith y Spiegelhalter y por Bretthorst, que ha utilizado métodos bayesianos para hacer retroceder los límites de la detección de señales de RMN[13]. La misma teoría bayesiana subyace al sistema de clasificación no supervisado, AutoClass. Una de las primeras aplicaciones de estos sofisticados métodos bayesianos de comparación de modelos con datos reales es la de Patrick y Wallace.

A medida que las cantidades de datos recopilados a lo largo de la ciencia y la ingeniería continúan aumentando, y el poder computacional y las técnicas disponibles para modelar esos datos también se multiplican, creo que los métodos bayesianos serán una herramienta cada vez más importante para refinar nuestras habilidades de modelado. Espero que esta revisión ayude a introducir estas técnicas a la comunidad de modelos 'neuronales', [92].

2.13.4.2 La evidencia y el factor Occam

Anotemos la regla de Bayes para los dos niveles de inferencia descritos anteriormente, para ver explícitamente cómo funciona la comparación de modelos bayesianos. Se supone que cada modelo H_i (H significa "hipótesis") tiene un vector de parámetros $\mathbf{w}$. Un modelo se define por su forma funcional y dos distribuciones de probabilidad: una distribución 'anterior' (*prior*) $P(\mathbf{w}|\, H_i)$ que establece qué valores pueden tomar los parámetros del modelo; y las predicciones $P(D|\mathbf{w}, H_i)$ que el modelo hace sobre los datos D cuando sus parámetros tienen un valor particular $\mathbf{w}$. Tenga en cuenta que los modelos con la misma parametrización, pero diferentes antecedentes sobre los parámetros, se definen por lo tanto como modelos diferentes, [95].

[13] Resonancia Magnética Nuclear (RMN).

1. Ajuste de modelo.

En el primer nivel de inferencia, asumimos que un modelo H_i es verdadero, e inferimos cuáles son los parámetros $\mathbf{w}$ del modelo que podrían recibir los datos D. Usando la regla de Bayes, la **probabilidad posterior (posterior probability)** de los parámetros $\mathbf{w}$ es:

$$P(\mathbf{w}|D, H_i) = \frac{P(D|\mathbf{w}, H_i)P(\mathbf{w}|H_i)}{P(D|H_i)}, \tag{2.74}$$

en otras palabras:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}.$$

$$\text{Posterior} = \frac{\text{Probabilidad} \times \text{Previo}}{\text{Evidencia}}.$$

La constante de normalización $P(D|H_i)$ se ignora comúnmente, ya que es irrelevante para el primer nivel de inferencia, es decir, la elección de $\mathbf{w}$; pero será importante en el segundo nivel de inferencia, y lo llamamos la **evidencia (evidence)** de H_i. Es común usar métodos basados en gradientes para encontrar el máximo de la parte posterior (*posterior*), que define el valor más probable para los parámetros, $\mathbf{w}_{\text{MP}}$; entonces es común resumir la distribución posterior por el valor de $\mathbf{w}_{\text{MP}}$ y las barras de error en estos mejores parámetros de ajuste. Las barras de error se obtienen de la curvatura de la parte posterior; escribiendo el Hessian $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, H_i)$ y la expansión de Taylor del logaritmo del posterior (*posterior*) con $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_{\text{MP}}$

$$P(\mathbf{w}|D, H_i) \cong P(\mathbf{w}_{\text{MP}}|D, H_i) \exp\left(-\frac{1}{2}\Delta\mathbf{w}^{\text{T}}\mathbf{A}\Delta\mathbf{w}\right), \tag{2.75}$$

vemos que el posterior puede aproximarse localmente como un Gaussiano con matriz de covarianza (barras de errores) $\mathbf{A}^{-1}$, [14] [96].

2. Comparación de modelo.

En el segundo nivel de inferencia, deseamos inferir qué modelo es más plausible dados los datos. La probabilidad posterior de cada modelo es:

$$P(H_i|D) \propto P(D|H_i)P(H_i), \tag{2.76}$$

Observe que el término dependiente de datos $P(D|H_i)$ es la evidencia de H_i, que apareció como la constante de normalización en (2.74). El segundo término, $P(H_i)$, es un previo (*prior*) 'subjetivo' sobre nuestro espacio de hipótesis que expresa cuán plausible pensamos que eran los modelos alternativos antes de que llegaran los datos. Más adelante veremos que esta parte subjetiva de la inferencia generalmente se verá abrumada por el término objetivo, la evidencia. Suponiendo que no tenemos ninguna razón para asignar previos (*priors*) $P(H_i)$ muy diferentes a los modelos alternativos, **los modelos H_i se clasifican evaluando la evidencia**. La ecuación (2.76) no se ha normalizado porque en el proceso de modelado de datos podemos desarrollar nuevos modelos después de que los datos hayan llegado (Figura 2.21), cuando se detecta una insuficiencia de los primeros modelos, por ejemplo. Por lo tanto, no comenzamos con un espacio de hipótesis completamente definido. La inferencia es abierta: buscamos continuamente modelos más probables para dar cuenta de los datos que recopilamos. Los nuevos modelos se comparan con modelos anteriores al evaluar la evidencia de ellos, [97].

[14] Si esta aproximación es buena o no dependerá del problema que estemos resolviendo. Para los modelos de interpolación discutidos en este capítulo, solo hay un máximo en la distribución posterior, y la aproximación gaussiana es exacta. Para modelos estadísticos más generales, todavía esperamos que la parte posterior esté dominada por picos locales de Gauss debido al teorema del límite central. Múltiples máximos que surgen en modelos más complejos complican el análisis, pero los métodos bayesianos aún pueden aplicarse con éxito.

El concepto clave de este capítulo es el siguiente: para asignar una preferencia a modelos alternativos H_i, un Bayesiano evalúa la evidencia $P(D|H_i)$. Este concepto es muy general: la evidencia se puede evaluar para modelos paramétricos y 'no paramétricos' por igual; si nuestra tarea de modelado de datos es un problema de regresión, un problema de clasificación o un problema de estimación de densidad, la evidencia es la cantidad transportable de Bayesian para comparar modelos alternativos. En todos estos casos, la evidencia incorpora naturalmente la navaja de Occam; Examinaremos cómo funciona esto en breve.

Por supuesto, la evidencia no es la historia completa si tenemos buenas razones para asignar antecedentes desiguales a los modelos alternativos H. (Usar solo la evidencia para la comparación del modelo es equivalente a usar la máxima probabilidad para la estimación de parámetros). El ejemplo clásico es la hipótesis 'Sure Thing', E.T Jaynes, que es la hipótesis de que el conjunto de datos será D, el conjunto de datos preciso que realmente ocurrió; la evidencia de la hipótesis de 'Sure Thing' es enorme. Pero 'Sure Thing' pertenece a una inmensa clase de hipótesis similares a las que se les debería asignar probabilidades previas (*prior probabilities*) correspondientemente pequeñas; entonces la probabilidad posterior (*posterior probability*) de 'Sure Thing' es insignificante junto con cualquier modelo sensible. Los modelos como 'Sure Thing' rara vez se proponen seriamente en la vida real, pero si tales modelos se desarrollan, entonces claramente debemos pensar con precisión qué previos (*priors*) son apropiados. Patrick y Wallace, estudiando la geometría de los antiguos círculos de piedra (¡sobre los cuales algunas personas han propuesto teorías extremadamente elaboradas!), discuten un método práctico de asignar probabilidades previas (*prior probabilities*) relativas a modelos alternativos evaluando las longitudes de los programas de computadora que decodifican datos previamente codificados bajo cada modelo. Este procedimiento introduce un segundo tipo de navaja de Occam (*Occam's razor*) en la inferencia, es decir, un bias previo

(*prior* bias) contra modelos complejos. Sin embargo, no incluiremos tales sesgos anteriores (*prior* bias) aquí; abordaremos solo la preferencia de los datos por los modelos alternativos, es decir, la evidencia (*evidence*) y la navaja de Occam (*Occam's razor*) que incorpora. En el límite de grandes cantidades de datos, este objetivo de la navaja de Occam (*Occam's razor*) siempre será el más importante de los dos, [95].

Un enfoque bayesiano moderno de los previos (priors)

Cabe señalar que el énfasis de este enfoque bayesiano moderno no está en la inclusión de previos (*priors*) en la inferencia. No hay un 'previo subjetivo' (*'subjective prior'*) significativo en todo este capítulo. El énfasis está en la idea de que los grados consistentes de preferencia por hipótesis alternativas están representados por probabilidades, y las preferencias relativas por modelos se asignan evaluando esas probabilidades. Históricamente, el análisis bayesiano ha sido acompañado por métodos para resolver el previo $P(w|H)$ (*prior* $P(w|H)$) "correcto" para un problema, por ejemplo, los principios de razón insuficiente y máxima entropía. Sin embargo, el Bayesiano moderno no adopta una actitud fundamentalista para asignar los previos (*priors*) "correctos": se pueden probar muchas prioridades diferentes, lo que permite que los datos nos informen cuál es la más adecuada. Cada previo (*prior*) particular corresponde a una hipótesis diferente sobre cómo es el mundo. Podemos comparar estas hipótesis alternativas a la luz de los datos evaluando la evidencia. Esta es la forma en que se comparan los regularizadores alternativos, por ejemplo. Si probamos un modelo y obtenemos predicciones terribles, hemos aprendido algo. "Una falla en la predicción bayesiana es una oportunidad para aprender", y podemos volver al mismo conjunto de datos con nuevos modelos, utilizando nuevos previos (*priors*), por ejemplo, [98].

Evaluando la evidencia

Ahora estudiemos explícitamente la evidencia para obtener una idea de cómo funciona la navaja de Occam Bayesiana. La evidencia es la constante de normalización para la ecuación (2.74):

$$P(D|H_i) = \int P(D|\mathbf{w}, H_i)P(\mathbf{w}|H_i)d\mathbf{w}. \qquad (2.77)$$

Para muchos problemas, incluida la interpolación, es común que la posterior (*posterior*) $P(\mathbf{w}|D, H_i) \propto P(D|\mathbf{w}, H_i)P(\mathbf{w}|H_i)$ tenga un pico fuerte en los parámetros más probables $\mathbf{w}_{\mathrm{MP}}$ (Figura 2.23) Entonces la evidencia se puede aproximar por la altura del pico del integrando $P(D|\mathbf{w}, H_i)P(\mathbf{w}|H_i)$ multiplicado por su ancho, $\Delta\mathbf{w}$:

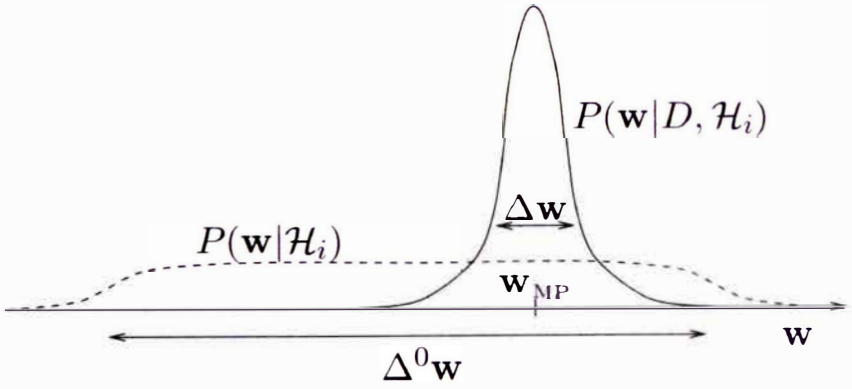


Figura 2.23. El factor Occam, *[99]*.

La Figura 2.23 muestra las cantidades que determinan el factor Occam para una hipótesis H_i que tiene un único parámetro $\mathbf{w}$. La distribución previa (línea de puntos) para el parámetro tiene un ancho $\Delta^0\mathbf{w}$. La distribución posterior (línea continua) tiene un pico único en $\mathbf{w}_{\mathrm{MP}}$ con un ancho característico $\Delta\mathbf{w}$. El factor Occam es $\frac{\Delta\mathbf{w}}{\Delta^0\mathbf{w}}$.

$$P(D|H_i) \cong P(D|\mathbf{w}_{\mathrm{MP}}, H_i)P(\mathbf{w}_{\mathrm{MP}}|H_i)\Delta\mathbf{w}. \qquad (2.78)$$

$$Evidence \cong Best\ \text{fit}\ likelihood \times Occam\ factor$$

Por lo tanto, la evidencia se encuentra tomando la mejor probabilidad de ajuste que el modelo pueda lograrlo y multiplicándolo por un 'factor Occam', que es un término con una magnitud menor que uno que penaliza a H_i por tener el parámetro $\mathbf{w}$, [100].

Interpretación del factor Occam

La cantidad $\Delta\mathbf{w}$ es la incertidumbre posterior en $\mathbf{w}$. Imagine por simplicidad que el previo $P(\mathbf{w}|H_i)$ es uniforme en algún intervalo grande $\Delta^0\mathbf{w}$, que representa el rango de valores de $\mathbf{w}$ que H_i creía posible antes de que llegaran los datos (figura 2.3). Luego $P(\mathbf{w}_{\mathrm{MP}}|H_i) = \frac{1}{\Delta^0\mathbf{w}}$, y

$$Factor\ Occam = \frac{\Delta\mathbf{w}}{\Delta^0\mathbf{w}}$$

es decir, **la proporción del volumen accesible posterior (*posterior*) del espacio de parámetros de H_i al volumen accesible previo (*prior*)**, o el factor por el cual el espacio de hipótesis de H_i se colapsa cuando llegan los datos. El modelo H_i puede verse como compuesto por un cierto número de submodelos equivalentes, de los cuales solo uno sobrevive cuando llegan los datos. El factor Occam es el inverso de ese número. El logaritmo del factor Occam se puede interpretar como la cantidad de información que obtenemos sobre el modelo cuando llegan los datos.

Típicamente, un modelo complejo con muchos parámetros, cada uno de los cuales puede variar libremente en un amplio rango $\Delta^0\mathbf{w}$, será penalizado con un factor Occam mayor que un modelo más simple. El factor Occam también proporciona una penalización para los modelos que deben ajustarse finamente para ajustar los datos; El factor Occam promueve modelos para los cuales la precisión requerida de los parámetros $\Delta\mathbf{w}$ es burda. El factor Occam es, por lo tanto, una medida de la complejidad del modelo, pero a diferencia de la dimensión V–C o la complejidad algorítmica, se relaciona con la complejidad de las predicciones que el modelo realiza en el espacio de datos; por lo tanto,

depende del número de puntos de datos y otras propiedades del conjunto de datos. El modelo que logra la mayor evidencia está determinado por un intercambio entre minimizar esta medida de complejidad natural y minimizar el mal ajuste de datos, [101].

Factor Occam para varios parámetros.

Si **w** es k-dimensional, y si el posterior está bien aproximado por un gaussiano, el factor Occam se obtiene del determinante de la matriz de covarianza del gaussiano:

$$P(D|H_i) \cong P(D|\mathbf{w}_{\mathrm{MP}}, H_i)\, P(\mathbf{w}_{\mathrm{MP}}|H_i)(2\pi)^{\frac{k}{2}} \det{}^{-\frac{1}{2}}\mathbf{A}. \qquad (2.79)$$

$$Evidence \cong Best\ \text{fit}\ likelihood \times Occam\ factor$$

donde $P(D|H_i)$ es la evidencia ($evidence$), $P(D|\mathbf{w}_{\mathrm{MP}}, H_i)$ es la mejor probabilidad de ajuste ($best\ \text{fit}\ likelihood$), y $P(\mathbf{w}_{\mathrm{MP}}|H_i)(2\pi)^{\frac{k}{2}} \det{}^{-\frac{1}{2}}\mathbf{A}$ es el factor Occam ($Occam\ factor$).

Donde $\mathbf{A} = -\nabla\nabla \log P(\mathbf{w}|D, H_i)$, el Hessian que ya evaluamos cuando calculamos las barras de error en $\mathbf{w}_{\mathrm{MP}}$. A medida que aumenta la cantidad de datos recopilados, N, se espera que esta aproximación gaussiana sea cada vez más precisa debido al teorema del límite central. Para los modelos de interpolación lineal discutidos en este capítulo, esta expresión gaussiana es exacta para cualquier N, [102].

Comentarios

- La selección del modelo bayesiano es una extensión simple de la selección del modelo de máxima probabilidad: **la evidencia se obtiene multiplicando la mejor probabilidad de ajuste por el factor Occam**.

  Para evaluar el factor Occam todo lo que necesitamos es el Hessian de $\mathbf{A}$, si la aproximación gaussiana es buena. Por lo tanto, el método bayesiano de comparación de modelos al evaluar la evidencia no es más exigente

computacionalmente que la tarea de encontrar para cada modelo los mejores parámetros de ajuste y sus barras de error.

- Es común que haya degeneraciones en modelos con muchos parámetros, es decir, varios parámetros equivalentes podrían ser reetiquetados sin afectar la probabilidad. En estos casos, el lado derecho de la ecuación (2.79) debe multiplicarse por la degeneración de $\mathbf{w}_{\mathbf{MP}}$ para obtener la estimación correcta de la evidencia.
- Los métodos de 'longitud mínima de descripción' (MDL) están estrechamente relacionados con este marco bayesiano. El logaritmo de evidencia $\log_2 P(D|H_i)$ es el número de bits en el mensaje más corto ideal que codifica los datos D usando el modelo H_i. El criterio de Akaike, originalmente derivado como un predictor de error de generalización, puede verse, como el 'B.I.C.' de Schwartz, como una aproximación a MDL y Bayes. Cualquier implementación de MDL requiere aproximaciones para evaluar la longitud del mensaje más corto ideal. Aunque algunos de los primeros trabajos sobre comparación de modelos complejos involucraron el marco MDL, MDL no tiene ventajas aparentes, y en mi trabajo yo aproximo la evidencia directamente.
- Debe enfatizarse que el factor Occam no tiene nada que ver con lo complejo que es computacionalmente usar un modelo. La evidencia es una medida de plausibilidad de un modelo. La cantidad de tiempo de CPU que se necesita para usar cada modelo es ciertamente un tema interesante que puede sesgar nuestras decisiones hacia modelos más simples, pero la regla de Bayes no aborda ese problema. Elegir entre modelos sobre la base de cuántas llamadas de función necesitan es un ejercicio de teoría de la decisión, que no se aborda

en este capítulo[15]. Una vez que se han inferido las probabilidades descritas anteriormente, se pueden elegir acciones óptimas utilizando la teoría de decisión estándar con una función de utilidad adecuada, [102].

2.13.4.3 El problema de interpolación ruidosa

La interpolación bayesiana a través de datos sin ruido ha sido estudiada por Skilling y Sibisi. En este capítulo estudio el problema de interpolar a través de datos donde se supone que las variables dependientes son ruidosas (una tarea también conocida como 'regresión', 'ajuste de curva', 'estimación de señal' o en la comunidad de RNA, 'aprendizaje'). No estoy examinando el caso donde las variables independientes también son ruidosas. Este problema diferente y más difícil ha sido estudiado para el caso de la adaptación de línea recta por Gull, [103].

Supongamos que el conjunto de datos a interpolar es un conjunto de pares $D = \{x_m, t_m\}$, donde $m = 1 \ldots N$ es una etiqueta que corre sobre los pares. Por simplicidad, trataré x y t como escalares, pero el método se generaliza al caso multidimensional. Para definir un modelo de interpolación lineal, se elige un conjunto de funciones básicas k fijas $\mathcal{A} = \{\varphi_h(x)\}$, y se supone que la función interpolada tiene la forma:

$$y(x) = \sum_{h=1}^{k} w_h \varphi_h(x), \tag{2.80}$$

donde los parámetros w_h deben inferirse de los datos. El conjunto de datos se modela como que se desvía de esta asignación bajo algún proceso de ruido aditivo $\mathcal{N}$:

$$t_m = y(x_m) + v_m. \tag{2.81}$$

[15] El capítulo al que hace referencia el autor no se aborda en este trabajo.

Si v se modela como ruido gaussiano de media cero con desviación estándar σ_v, entonces la probabilidad de los datos[16] dados los parámetros **w** es:

$$P(D|\mathbf{w},\beta,\mathcal{A},\mathcal{N}) = \frac{\exp\left(-\beta E_D(D|\mathbf{w},\mathcal{A})\right)}{Z_D(\beta)}, \tag{2.82}$$

donde $\beta = 1/\sigma_v^2$, $E_D = \sum_m \frac{1}{2}(y(x_m) - t_m)^2$, y $Z_D = (2\pi/\beta)^{N/2}$.

$P(D|\mathbf{w},H_i,\beta,\mathcal{A},\mathcal{N})$ es llamado probabilidad (*likelihood*). Es bien sabido que encontrar los parámetros de probabilidad máxima $\mathbf{w}_{\mathrm{ML}}$ puede ser un problema "mal planteado". Es decir, la **w** que minimiza la E_D está subdeterminada y/o depende sensiblemente de los detalles del ruido en los datos; La máxima probabilidad de interpolación en tales casos oscila enormemente para adaptarse al ruido. Por lo tanto, está claro que para completar un modelo de interpolación necesitamos un previo $\mathcal{R}$ que exprese el tipo de suavidad que esperamos que tenga el interpolante $y(x)$. Un modelo puede tener un previo (*prior*) de la forma

$$P(y|\mathcal{R},\alpha) = \frac{\exp\left(-\alpha E_y(y|\mathcal{R})\right)}{Z_y(\alpha)}, \tag{2.83}$$

donde E_y podría ser, por ejemplo, la funcional $E_y = \int y''(x)^2 dx$ (que es el regularizador para la interpolación spline cúbica[17]). El parámetro α es una medida de cuán suave se espera que sea $f(x)$. Tal previo (*prior*) también se puede escribir como un previo (*prior*) en los parámetros **w**:

$$P(\mathbf{w}|\alpha,\mathcal{A},\mathcal{R}) = \frac{\exp\left(-\alpha E_W(\mathbf{w}|\mathcal{A},\mathcal{R})\right)}{Z_W(\alpha)}, \tag{2.84}$$

---

[16] Estrictamente, esta probabilidad debería escribirse $P(\{t_m\}|\{x_m\},\mathbf{w},\beta,\mathcal{A},\mathcal{N})$, ya que estos modelos de interpolación no predicen la distribución de las variables de entrada $\{x_m\}$; esta libertad de notación se tomará a lo largo de esta tesis.

[17] Estrictamente, este previo (*prior*) particular puede ser incorrecto porque una $y(x)$ de la forma $w_1 x + w_0$ no está limitada por este previo (*prior*).

donde $Z_W = \int d^k \mathbf{w} \exp(-\alpha E_W)$. E_W (o E_y) se conoce comúnmente como una función de regularización.

El modelo de interpolación H ahora está completo, y consiste en una selección de funciones básicas $\mathcal{A}$, un modelo de ruido $\mathcal{N}$ con parámetro β y un previo (*prior*) $\mathcal{R}$ (regularizador), con regularización constante α. Los ajustes particulares de los hiperparámetros α y β se verán como submodelos de H, [103].

El primer nivel de inferencia

Si se conocen α y β, entonces la probabilidad posterior de los parámetros **w** es:[18]

$$P(\mathbf{w}|D,\alpha,\beta,\mathcal{A},\mathcal{R},\mathcal{N}) = \frac{P(D|\mathbf{w},\beta,\mathcal{A},\mathcal{N})P(\mathbf{w}|\alpha,\mathcal{A},\mathcal{R})}{P(D|\alpha,\beta,\mathcal{A},\mathcal{R},\mathcal{N})}, \tag{2.85}$$

Escribiendo[19]

$$M(\mathbf{w}) = \alpha E_W + \beta E_D \tag{2.86}$$

el posterior (posterior) es

$$P(\mathbf{w}|D,\alpha,\beta,\mathcal{A},\mathcal{R},\mathcal{N}) = \frac{\exp\big(-M(\mathbf{w})\big)}{Z_M(\alpha,\beta)}, \tag{2.87}$$

donde $Z_M(\alpha,\beta) = \int d^k \mathrm{w} \exp(-M)$. Vemos que minimizar la función objetivo combinada M corresponde a encontrar el *interpolante más probable*, $\mathbf{w}_{MP}$. Las barras de error en el mejor ajuste interpolante[20] se pueden obtener del Hessian de M, $\mathbf{A} = \nabla\nabla M$, evaluado en $\mathbf{w}_{MP}$.

---

[18] El regularizador α, R se ha omitido de las variables de condicionamiento en la probabilidad porque la distribución de datos no depende del previo (*prior*) una vez que se conoce **w**. Del mismo modo, lo anterior no depende de β, N.

[19] El nombre M significa 'mal ajustado' (*misfit*); más tarde se demostrará que M es la medida natural de mal ajuste (*misfit*), en lugar de $\chi_D^2 = 2\beta E_D$.

[20] Estas barras de error representan la incertidumbre del interpolante y no deben confundirse con la dispersión típica de puntos de datos ruidosos en relación con el interpolante.

Esta es la conocida visión bayesiana de la regularización, también conocida como "máxima probabilidad penalizada" o "regresión de cresta".

Los métodos bayesianos proporcionan mucho más que una simple interpretación para la regularización. Lo que hemos descrito hasta ahora es solo el primero de tres niveles de inferencia. (El segundo nivel descrito en las secciones 1 y 2, 'comparación de modelos', se divide en un segundo y tercer nivel para este problema, porque cada modelo de interpolación está formado por un continuo de submodelos con diferentes valores de α y β) En el segundo nivel, Bayes nos permite asignar objetivamente valores a α y β, que comúnmente se desconocen *a priori*. En el tercero, Bayes nos permite clasificar cuantitativamente conjuntos de bases alternativos $\mathcal{A}$, regularizadores alternativos (*priors*) $\mathcal{R}$, y en principio, modelos de ruido alternativos $\mathcal{N}$[21]. Además, podemos comparar cuantitativamente la interpolación bajo cualquier modelo $H = \{\mathcal{A}, \mathcal{N}, \mathcal{R}\}$ con otros modelos de interpolación y aprendizaje, como las RNA, si se les aplica un enfoque bayesiano similar. Ni el segundo ni el tercer nivel de inferencia se pueden ejecutar con éxito sin la navaja de Occam.

La teoría bayesiana del segundo y tercer nivel de inferencia se ha desarrollado recientemente; El objetivo de este capítulo es revisar ese marco. La sección siguiente describirá el método bayesiano de inferir α y β; la sección subsiguiente describirá la comparación del modelo bayesiano para el problema de interpolación. Ambos problemas de inferencia se resuelven mediante la evaluación de la evidencia (*evidence*) apropiada, [104].

[21] La inferencia bayesiana de una distribución ligeramente no gaussiana se realiza en Box y Tiao.

2.13.4.3 Selección de parámetros α y β

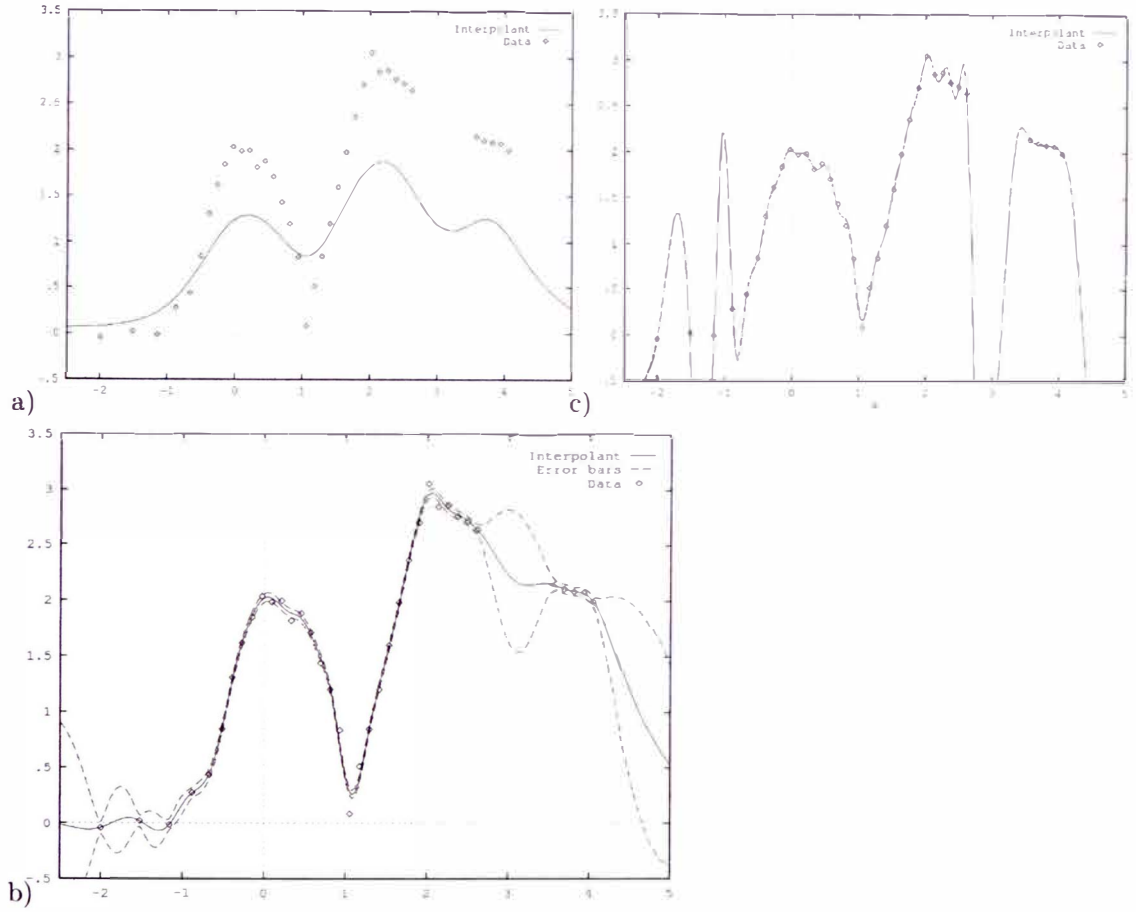


Figura 2.24. Cómo el mejor interpolante depende de α, *[105]*.

Estas figuras introducen un conjunto de datos, 'X', que se interpola con una variedad de modelos en este capítulo. Observe que la densidad de los puntos de datos no es uniforme en el eje x. En las tres figuras, el conjunto de datos se interpola utilizando un modelo de función de base radial con una base de 60 funciones Cauchy igualmente espaciadas, todas con un radio de 0.2975. El regularizador es $E_W = \frac{1}{2}\sum w^2$, donde w son los coeficientes de las funciones básicas. Cada figura muestra el interpolante más probable para un valor diferente de α: a) 6000; b) 2.5; c) 10^{-7}. Observe en los valores extremos cómo los datos están sobre alisado y sobre ajustados respectivamente. Suponiendo un prior (*prior*) plano, $\alpha = 2.5$ es el valor *más probable* de α. En b), el interpolante más probable

se muestra con sus barras de error 1σ, que representan cuán inciertos somos sobre el interpolante en cada punto, bajo el supuesto de que el modelo de interpolación y el valor de α son correctos. Observe cómo las barras de error aumentan en magnitud cuando los datos son escasos. Las barras de error no se hacen más grandes cerca del punto de datos cercano a (1,0), porque el modelo de función de base radial no espera discontinuidades agudas; las barras de error se obtienen suponiendo que el modelo es correcto, de modo que ese punto se interpreta como un valor atípico improbable.

Típicamente, α no se conoce a priori y, a menudo, β también se desconoce. A medida que α varía, las propiedades del mejor interpolante (más probable) varían. Suponga que estamos utilizando un previo (*prior*) que fomenta la suavidad, e imagine que interpolamos en un valor muy grande de α; entonces esto restringirá que el interpolante sea muy suave y plano, y no se ajustará bien a los datos (Figura 2.24.a). A medida que disminuye α, el interpolante comienza a ajustarse mejor a los datos (Figura 2.24.b). Si α se hace aún más pequeño, el interpolante oscila enormemente para sobre ajustarse al ruido en los datos (Figura 2.24.c). La elección del 'mejor' valor de α es nuestro primer problema de 'navaja de Occam': los valores grandes de α corresponden a modelos simples que hacen predicciones restringidas y precisas, diciendo que 'el interpolante no tendrá curvatura extrema en ningún lado'; un pequeño valor de α corresponde al modelo más potente y flexible que dice 'el interpolante podría ser cualquier cosa, nuestra creencia previa en la suavidad es muy débil'. La tarea es encontrar un valor de α que sea lo suficientemente pequeño como para que los datos se ajusten, pero no tan pequeños que se sobre ajusten. Para varios problemas mal planteados, como la deconvolución, el valor preciso del parámetro de regularización es cada vez más importante. Las estadísticas ortodoxas tienen formas de asignar valores a dichos parámetros, basándose por ejemplo en criterios de ajuste incorrecto, el uso de datos de prueba, y validación cruzada. Gull ha

demostrado por qué el uso popular de los criterios de ajuste incorrecto es incorrecto y cómo Bayes establece estos parámetros. El uso de datos de prueba puede ser una técnica poco confiable a menos que haya grandes cantidades de datos disponibles, [106].

Ajuste incorrecto, χ^2, y el efecto de las mediciones de parámetros

Para N variables gaussianas independientes con media μ y desviación estándar σ, la estadística $\chi^2 = \sum (x - \mu)/\sigma^2$ es una medida de desajuste. Si μ se conoce *a priori*, χ^2 tiene una expectativa $N \pm \sqrt{N}$. Sin embargo, si μ se ajusta a partir de los datos estableciendo $\mu = \bar{x}$, 'usamos un grado de libertad', y χ^2 tiene la expectativa $N - 1$. En el segundo caso, μ es un "parámetro bien medido". Cuando los datos determinan un parámetro de esta manera, es inevitable que el parámetro también ajuste parte del ruido en los datos. Es por eso que la expectativa de χ^2 se reduce en uno. Esta es la base de la distinción entre los botones σ_N y σ_{N-1} en su calculadora. Es común que se ignore esta distinción, pero en casos como la interpolación donde el número de parámetros libres es similar al número de puntos de datos, es esencial encontrar y hacer una distinción análoga. Se demostrará que las elecciones bayesianas de α y β se expresan más simplemente en términos del número efectivo de parámetros bien medidos, γ, que se derivan a continuación.

Los criterios de ajuste incorrecto son "principios" que establecen parámetros como α y β al exigir que χ^2 tenga un valor particular. El principio de discrepancia requiere $\chi^2 = N$. Otro principio requiere $\chi^2 = N - k$, donde k es el número de parámetros libres. Encontraremos que surge un criterio de ajuste intuitivo para el valor más probable de β; por otro lado, la elección bayesiana de α no estará relacionada con el valor del ajuste incorrecto, [107].

Elección Bayesiana de α y β

Para inferir de los datos qué valor deben tener α y β, la evaluación bayesiana de la distribución de probabilidad posterior:

$$P(\alpha,\beta|D,H) = \frac{P(D|\alpha,\beta,H)P(\alpha,\beta|H)}{P(D|H)}, \qquad (2.88)$$

El término dependiente de datos $P(D|\alpha,\beta,H)$ ya apareció antes como la constante de normalización en la ecuación (2.85), y se llama evidencia de α y β. De manera similar, la constante de normalización de (2.88) se denomina evidencia de H, y aparecerá más adelante cuando comparemos modelos alternativos $H = \{\mathcal{A}, \mathcal{N}, \mathcal{R}\}$ a la luz de los datos.

Si $P(\alpha,\beta|H)$ es un previo (*prior*) plano[22] (que corresponde a la afirmación de que no sabemos qué valor deberían tener α y β), la evidencia es la función que utilizamos para asignar una preferencia a valores alternativos de α y β. Se da en términos de las constantes de normalización definidas anteriormente por

$$P(D|\alpha,\beta,H) = \frac{Z_M(\alpha,\beta)}{Z_W(\alpha)Z_D(\beta)}. \qquad (2.89)$$

La navaja de Occam está implícita en esta fórmula: si α es pequeña, la gran libertad en el rango previo (*prior*) de posibles valores de **w** se penaliza automáticamente por el gran valor consiguiente de Z_W; Los modelos que se ajustan bien a los datos logran un valor grande de Z_M. El valor óptimo de α logra un compromiso entre ajustar bien los datos y ser un modelo simple.

Ahora para asignar una preferencia a (α, β), nuestra tarea computacional es evaluar las tres integrales Z_M, Z_W y Z_D. Volveremos a esta tarea en un momento.

[22] Dado que α y β son parámetros de escala, este previo (*prior*) debe entenderse como un previo (*prior*) plano sobre $\log\alpha$ y $\log\beta$.

¡Pero eso suena como determinar tu previo después de que hayan llegado los datos!

Cuando escuché por primera vez la explicación anterior de la regularización bayesiana, me sentí descontento porque parecía que lo anterior se elegía de un conjunto de posibles antecedentes después de que llegaran los datos. Para ser precisos, como se describió anteriormente, se selecciona el valor más probable de α; entonces el previo correspondiente a ese valor de α solo se usa para inferir cuál podría ser el interpolante. Así no es como Bayes nos haría inferir el interpolante. Es el conjunto combinado de previos (priors) lo que define nuestro previo (prior), y debemos integrarnos sobre este conjunto cuando hacemos inferencia [23]. Analicemos qué sucede si seguimos este enfoque adecuado. El método anterior de usar solo el previo (*prior*) más probable surgirá como una buena aproximación.

El verdadero posterior (*posterior*) $P(\mathbf{w}|D,H)$ se obtiene integrando sobre α y β:

$$P(\mathbf{w}|D,H) = \int P(\mathbf{w}|D,\alpha,\beta,H)P(\alpha,\beta|D,H)d\alpha d\beta. \qquad (2.90)$$

En palabras, la probabilidad posterior (*posterior probability*) sobre **w** puede escribirse como una combinación lineal de las posteriores (*posteriors*) para todos los valores de α, β. Cada densidad posterior (*posterior*) está ponderada por la probabilidad de α, β dados los datos, que aparecieron en (2.88). Esto significa que si $P(\alpha,\beta|D,H)$ tiene un pico dominante en $\hat{\alpha}$, $\hat{\beta}$, entonces el verdadero posterior (*posterior*) $P(\mathbf{w}|D,H)$ estará dominado por la densidad $P(\mathbf{w}|D,\alpha,\beta,H)$. Mientras las propiedades posteriores $P(\mathbf{w}|D,\alpha,\beta,H)$ no cambien rápidamente con α, β cerca de $\hat{\alpha}$, $\hat{\beta}$ y el pico en $P(\alpha,\beta|D,H)$ sea fuerte, estamos justificados al usar la aproximación:

[23] Es notable que Laplace casi acertó en 1774; al inferir la media de una distribución laplaciana, infirió la probabilidad posterior de un parámetro molesto como β en (2.88), y luego intentó integrar el parámetro molesto como en la ecuación (2.91).

$$P(\mathbf{w}|D,H) \cong P\left(\mathbf{w}\middle|D,\hat{\alpha},\hat{\beta},H\right). \qquad (2.91)$$

Esta aproximación es válida si cuando $\gamma \gg 1$, y, en el espectro de valores propios de $\beta\mathbf{B}$, el número de valores propios dentro de e-fold de $\hat{\alpha}$ es $\ll \gamma$. Es una investigación en curso desarrollar métodos computacionales para casos en los que esta aproximación no sea válida (Sibisi y Skilling, comunicación personal, Neal, comunicación personal). En algunos casos, incluidos los modelos lineales de este capítulo, la integral (2.91) se puede realizar analíticamente. Elegí usar las aproximaciones independientemente, porque 1) las aproximaciones dan una intuición más clara de cómo los métodos bayesianos resuelven los problemas de regularización; 2) las aproximaciones son aplicables a casos en los que no existe una solución analítica; y 3) las aproximaciones se relacionan más estrechamente con los métodos de regularización alternativos, que buscan encontrar valores 'óptimos' de α, β, [108].

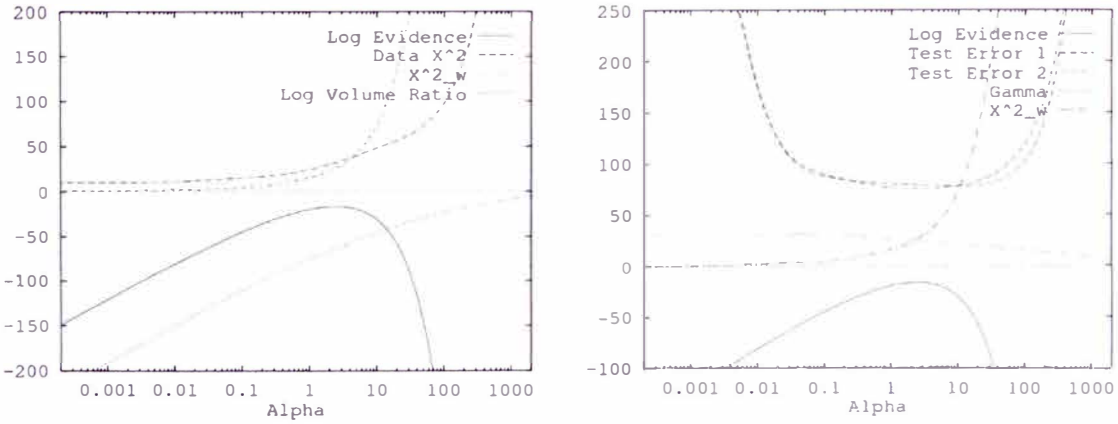


Figura 2.25. Elección de α, *[109].*

a) **La evidencia como una función de** α: Usando el mismo modelo de función de base radial que en la Figura *2.24.* Cómo el mejor interpolante depende de αFigura 2.24, este gráfico muestra el logaritmo de la evidencia (*evidence*) como una función de α, y muestra las funciones que componen el logaritmo de la evidencia (*evidence*), es decir, el desajuste de datos $\mathcal{X}_D^2 = 2\beta E_D$, el término de penalización de peso $\mathcal{X}_W^2 = 2\alpha E_w$ y el logaritmo de la relación de volumen $(2\pi)^{k/2}\det{}^{-1/2}\mathbf{A}/Z_W(\alpha)$.

b) **Criterios para optimizar** α: este gráfico muestra la evidencia logarítmica en función de α, y las funciones cuya intersección ubica la evidencia máxima: el número de buenas mediciones de parámetros γ y χ_W^2. También se muestra el error de prueba (reescalado) en dos conjuntos de prueba; encontrar el mínimo de error de prueba es un criterio alternativo para establecer α. Ambos conjuntos de prueba tenían más del doble de tamaño que el conjunto de datos interpolados. Observe cómo el punto en el que $\chi_W^2 = \gamma$ es claro e inequívoco, lo que no se puede decir de los mínimos de las energías de prueba. La evidencia da a α un intervalo de confianza de $1 - \sigma$ de [1.3, 5.0]. Los mínimos de error de prueba se distribuyen más ampliamente debido al ruido de muestra finito.

¿Por qué no encontrar el punto óptimo en w, α, β?

No es satisfactorio simplemente maximizar la probabilidad o la probabilidad posterior simultáneamente sobre **w**, α y β; tanto la posterior como la probabilidad tienen picos sesgados, de modo que el valor de máxima probabilidad para los parámetros no está en el mismo lugar que la mayoría de la probabilidad posterior. Para tener una idea de esto, aquí hay un problema más familiar: examine la probabilidad posterior de los parámetros de un Gaussiano (μ, σ) dadas N muestras: el valor de probabilidad máximo para σ es σ_N, pero el valor más probable para σ (encontrado integrando sobre μ) es σ_{N-1}. Debe enfatizarse que esta distinción no tiene nada que ver con el previo sobre los parámetros α y β, que es plano aquí. Es el proceso de marginación que corrige el sesgo que afecta tanto la máxima probabilidad como la máxima a posteriori, [110].

Evaluando la evidencia (evidence)

Volvamos a nuestro hilo de pensamiento en la ecuación (2.89). Para evaluar la evidencia de α, β, queremos encontrar las integrales Z_M, Z_W y Z_D. Por lo general, la integral más difícil de evaluar es Z_M.

$$Z_M = \int d^k \mathbf{w} \exp\big(-M(\mathbf{w}, \alpha, \beta)\big). \tag{2.92}$$

Si el regularizador $\mathcal{R}$ es una función cuadrática (y los favoritos son), entonces E_D y E_W son funciones cuadráticas de $\mathbf{w}$, y podemos evaluar Z_M exactamente. Haciendo $\nabla\nabla E_W = \mathbf{C}$ y $\nabla\nabla E_D = \mathbf{B}$ y luego usando $\mathbf{A} = \alpha\mathbf{C} + \beta\mathbf{B}$, tenemos:

$$M = M(\mathbf{w}_{\mathrm{MP}}) + \frac{1}{2}(\mathbf{w} + \mathbf{w}_{\mathrm{MP}})^{\mathrm{T}}\mathbf{A}(\mathbf{w} + \mathbf{w}_{\mathrm{MP}}). \tag{2.93}$$

donde $\mathbf{w}_{\mathrm{MP}} = \beta\mathbf{A}^{-1}\mathbf{B}\mathbf{w}_{\mathrm{ML}}$. Esto significa que Z_M es la integral Gaussiana:

$$Z_M = e^{-M_{MP}}(2\pi)^{\frac{k}{2}} \det{}^{-\frac{1}{2}}\mathbf{A}. \tag{2.94}$$

En muchos casos donde el regularizador no es cuadrático (por ejemplo, basado en la entropía), esta aproximación gaussiana todavía es útil. Por lo tanto, podemos escribir la evidencia logarítmica de α y β como:

$$\begin{aligned} log P(D|\alpha, \beta, H) \\ &= -\alpha E_W^{MP} - \beta E_D^{MP} - \frac{1}{2}\log\det\mathbf{A} - \log Z_W(\alpha) \\ &\quad - \log Z_D(\beta) + \frac{k}{2}\log 2\pi\,. \end{aligned} \tag{2.95}$$

El término βE_D^{MP} representa el mal ajuste del interpolante a los datos. Los tres términos $-\beta E_D^{MP} - \frac{1}{2}\log\det\mathbf{A} - \log Z_W(\alpha)$ constituyen el logaritmo del 'factor Occam' (Occam factor) que penaliza los pequeños valores de α: la relación $(2\pi)^{k/2}\det{}^{-1/2}\mathbf{A}/Z_W(\alpha)$ es la relación del volumen accesible posterior en el espacio de parámetros al volumen accesible previo (*prior*), y el término αE_W^{MP} mide

qué tan lejos está $\mathbf{w}_{MP}$ de su valor nulo. La Figura 2.25a ilustra el comportamiento de estos diversos términos en función de α para el mismo modelo de función de base radial como se ilustra en la Figura 2.24.

Ahora podríamos proceder a evaluar la evidencia numéricamente en función de α y β, pero es posible una comprensión más profunda y fructífera de este problema, [111].

Propiedades de la evidencia máxima

El máximo sobre α, β de $P(D|\alpha, \beta, H) = Z_M(\alpha, \beta)/(Z_W(\alpha)Z_D(\beta))$ tiene algunas propiedades notables que dan una visión más profunda de este enfoque bayesiano. Los resultados de esta sección son útiles tanto numérica como intuitivamente.

Siguiendo a Gull, transformamos en la base en la cual el Hessian de E_W es la identidad, $\nabla\nabla E_W = \mathbf{I}$. Esta transformación es simple en el caso de E_W cuadrática: rotar en la base del vector propio de $\mathbf{C}$ y estire los ejes para que la forma cuadrática E_W se vuelva homogénea. Esta es la base natural para el previo (*prior*). Continuaré refiriéndome al vector de parámetros en esta base como $\mathbf{w}$, así que de aquí en adelante $E_W = \frac{1}{2}\sum w_i^2$. Usando $\nabla\nabla M = \mathbf{A}$ y $\nabla\nabla E_D = \mathbf{B}$ como anteriormente, diferenciamos la evidencia logarítmica con respecto a α y β para encontrar la condición que se satisface al máximo. La evidencia logarítmica, de (2.95), es:

$$\log P(D|\alpha,\beta,H) = -\alpha E_W^{MP} - \beta E_D^{MP} - \frac{1}{2}\log\det\mathbf{A} + \frac{k}{2}\log\alpha + \frac{k}{2}\log\beta + \frac{N}{2}\log 2\pi . \qquad (2.96)$$

Primero, diferenciando con respecto a α, necesitamos evaluar $\frac{d}{d\alpha}\log\det\mathbf{A}$. Usando $\mathbf{A} = \alpha\mathbf{I} + \beta\mathbf{B}$,

$$\frac{d}{d\alpha}\log\det\mathbf{A} = Trace\left(\mathbf{A}^{-1}\frac{d\mathbf{A}}{d\alpha}\right)$$

$$= \mathrm{Trace}(\mathbf{A}^{-1}\mathbf{I}) = \mathrm{Trace}\mathbf{A}^{-1}.$$

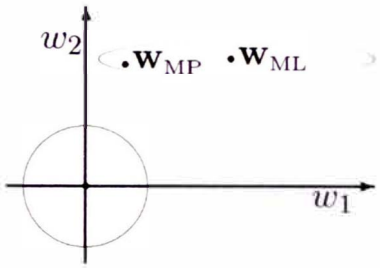


Figura 2.26. Mediciones de parámetros buenos y malos, *[112]*.

Sea w_1 y w_2 los componentes en el espacio de parámetros en dos direcciones paralelas a los vectores propios de la matriz de datos $\mathbf{B}$. El círculo representa la distribución previa característica para $\mathbf{w}$. La elipse representa un contorno característico de la probabilidad, centrado en la solución de probabilidad máxima $\mathbf{w}_{\mathrm{ML}}$. $\mathbf{w}_{\mathrm{MP}}$ representa el vector de parámetro más probable. w_1 es una dirección en la que λ_1 es pequeño en comparación con α, es decir, los datos no tienen una fuerte preferencia sobre el valor de w_1; w_1 es un parámetro mal medido, y el término $\frac{\lambda_1}{\lambda_1+\alpha}$ está cerca de cero. w2 es una dirección en la que λ_1 es grande; w_2 está bien determinado por los datos, y el término $\frac{\lambda_2}{\lambda_2+\alpha}$ está cerca de uno.

Este resultado es exacto si E_W y E_D son cuadráticos. De lo contrario, este resultado es una aproximación, omitiendo términos en $\partial\mathbf{B}/\partial\alpha$. Ahora, al diferenciar (2.96) y establecer la derivada a cero, obtenemos la siguiente condición para el valor más probable de α:

$$2\alpha E_D^{MP} = k - \alpha\mathrm{Trace}\mathbf{A}^{-1}. \qquad (2.97)$$

La cantidad a la izquierda es la medida adimensional de la cantidad de estructura introducida en los parámetros por los datos, es decir, cuánto difieren los parámetros ajustados de su valor nulo. Se puede interpretar como la $\mathcal{X}^2$, de los parámetros, ya que es igual a $\mathcal{X}_w^2 = \sum w_i^2/\sigma_w^2$, con $\alpha = 1/\sigma_w^2$.

La cantidad a la derecha de (2.97) se llama el número de buenas mediciones de parámetros, γ, y tiene un valor entre 0 y k. Se puede escribir en términos de los valores propios de $\beta\mathbf{B}$, λ_a, donde el subíndice a corre sobre los k vectores propios. Los valores propios de $\mathbf{A}$ son $\boldsymbol{\lambda}_a + \alpha$, por lo que tenemos:

$$\gamma = k - \alpha \mathrm{Trace}\mathbf{A}^{-1} = k - \sum_{a=1}^{k} \frac{\alpha}{\lambda_a + \alpha} = \sum_{a=1}^{k} \frac{\lambda_a}{\lambda_a + \alpha}. \qquad (2.98)$$

Cada valor propio $\boldsymbol{\lambda}_a$ mide cuan fuertemente los datos determinan un parámetro. La constante α mide cuán fuertemente los parámetros están determinados por el previo. El término $a°\ \gamma_a = \lambda_a/(\lambda_a + \alpha)$ es un número entre 0 y 1 que mide la fuerza de los datos en relación con el previo en la dirección a (Figura 2.26): los componentes de $\mathbf{w}_{\mathrm{MP}}$son dados por $\mathbf{w}_{\mathrm{MPa}} = \gamma_a \mathbf{w}_{\mathrm{ML}a}$.

Una dirección en el espacio de parámetros para la cual λ_a es pequeña en comparación con α no contribuye al número de buenas mediciones de parámetros. Por lo tanto, γ es una medida del número efectivo de parámetros que están bien determinados por los datos. Como $\alpha/\beta \to 0$, γ aumenta de 0 a k. Por lo tanto, la condición (2.97) para el valor más probable de α puede interpretarse como una estimación de la varianza σ_W^2 de la distribución gaussiana de la que se extraen los pesos, en base a muestras efectivas γ de esa distribución: $\sigma_W^2 = \sum w_i^2/\gamma$.

Este concepto no solo es importante para localizar el valor óptimo de α: solo se espera que las mediciones de parámetros γ buenos contribuyan a la reducción del desajuste de datos que ocurre cuando un modelo se ajusta a datos ruidosos. En el proceso de ajustar $\mathbf{w}$ a los datos, es inevitable que ocurra alguna adaptación del modelo al ruido, porque algunos componentes del ruido son indistinguibles de los datos reales. Típicamente, una unidad ($\mathcal{X}^2$) de ruido se ajustará para cada parámetro bien determinado. Los parámetros mal determinados están determinados únicamente por el regularizador, por lo que no reducen $\mathcal{X}_D^2$ de esta manera. Ahora examinaremos cómo este concepto entra en la elección bayesiana de β.

Recuerde que la expectativa del ajuste incorrecto $\mathcal{X}^2$ entre el interpolante verdadero y los datos es N. Sin embargo, no conocemos el interpolante verdadero, y la única medida de ajuste incorrecto a la que tenemos acceso es el $\mathcal{X}^2$ entre el interpolante inferido y los datos, $\mathcal{X}_D^2 = 2\beta E_D$. El 'principio de discrepancia' de las estadísticas ortodoxas establece que los parámetros del modelo deben ajustarse para que $\mathcal{X}_D^2 = N$. El trabajo en la regresión de mínimos cuadrados no regularizada sugiere que debemos estimar el nivel de ruido para establecer $\mathcal{X}_D^2 = N - k$, donde k es el número de parámetros libres. Descubramos la opinión de la regla de Bayes sobre este asunto.

Diferenciamos la evidencia logarítmica (2.96) con respecto a β y obtenemos, estableciendo la derivada a cero:

$$2\beta E_D = N - \gamma. \qquad (2.99)$$

Por lo tanto, la estimación de ruido más probable, $\hat{\beta}$, no satisface $\mathcal{X}_D^2 = N$ o $\mathcal{X}_D^2 = N - k$; más bien, $\mathcal{X}_D^2 = N - \gamma$. Esta estimación bayesiana del nivel de ruido naturalmente tiene en cuenta el hecho de que los parámetros que han sido determinados por los datos inevitablemente suprimen parte del ruido en los datos, mientras que los parámetros mal medidos no lo hacen. La cantidad $N - \gamma$ se puede llamar el número efectivo de grados de libertad. Tenga en cuenta que el valor de $\mathcal{X}_D^2$ solo entra en la determinación de β: los criterios de ajuste incorrecto (misfit) no tienen ningún papel en la elección bayesiana de α.

En resumen, en el valor óptimo de α y β, $\mathcal{X}_W^2 = \gamma$, $\mathcal{X}_D^2 = N - \gamma$. Observe que esto implica que el ajuste incorrecto total $M = \alpha E_W + \beta E_D$ satisface la ecuación simple $2M = N$.

El interpolante resultante de la elección bayesiana de α se ilustra en la Figura 2.24b. La Figura 2.25b ilustra las funciones involucradas con la elección bayesiana de α, y las compara con el enfoque de "error de prueba". Se omite la demostración de la elección bayesiana de β, ya que es sencilla; β se fija a su verdadero valor para las demostraciones de este capítulo.

Estos resultados se generalizan al caso en que hay dos o más regularizadores separados con constantes de regularización independientes $\{\alpha_c\}$. En este caso, cada regularizador tiene varias buenas mediciones de parámetros γ_c asociadas a él.

Encontrar la evidencia máxima con un enfoque frontal implicaría evaluar $\det \mathbf{A}$ mientras se busca sobre α, β; los resultados anteriores ((2.97),(2.99)) nos permiten acelerar esta búsqueda (por ejemplo, mediante el uso de fórmulas de reestimación como $\alpha := \gamma/2E_W$) y reemplazar la evaluación de $\det \mathbf{A}$ por la evaluación de $\mathrm{Trace}\mathbf{A}^{-1}$. Para problemas de grandes dimensiones donde esta tarea es exigente, Skilling ha desarrollado métodos para estimar $\mathrm{Trace}\mathbf{A}^{-1}$ estadísticamente en el tiempo k^2, [113].

2.13.5 Regularización Bayesiana (específica a las RNA)

Aunque hay muchos enfoques para la selección automática del parámetro de regularización, nos concentraremos en uno desarrollado por David MacKay[24], [114]. Este enfoque coloca el entrenamiento de RNA en un marco estadístico bayesiano. Este marco es útil para muchos aspectos del entrenamiento, además de la selección del parámetro de regularización, por lo que es un concepto importante para familiarizarse. Hay dos niveles en este análisis bayesiano. Comenzaremos con el Nivel I.

2.13.5.1 Marco Bayesiano de Nivel I

El marco Bayesiano comienza con el supuesto de que los pesos de la RNA son variables aleatorias. Luego elegimos los pesos que maximizan la probabilidad condicional de los pesos dados los datos. La regla de Bayes se usa para encontrar esta función de probabilidad:

[24] David MacKay fue un verdadero polímato que hizo contribuciones pioneras a la teoría de la información, la inferencia y los algoritmos de aprendizaje. Fue uno de los fundadores del enfoque moderno de la teoría de la información, combinando la inferencia bayesiana con algoritmos de redes neuronales artificiales para permitir la toma racional de decisiones por computadora. [138]

$$P(\mathbf{x}|D,\alpha,\beta,M) = \frac{P(D|\mathbf{x},\beta,M)P(\mathbf{x}|\alpha,M)}{P(D|\alpha,\beta,M)}, \tag{2.100}$$

Donde **x** es el vector que contiene todos los pesos y bias en la RNA, D representa el conjunto de datos de entrenamiento, α y β son parámetros asociados con las funciones de densidad $P(D|\mathbf{x},\beta,M)$ y $P(\mathbf{x}|\alpha,M)$, y M es el modelo seleccionado – la arquitectura de la RNA que hemos elegido (es decir, que cantidad de capas y que cantidad de neuronas en cada capa).

Vale la pena tomarse un tiempo para investigar cada uno de los términos en la ecuación (2.100). Primero, $P(D|\mathbf{x},\beta,M)$ es la densidad de probabilidad para los datos, dado un cierto conjunto de pesos **x**, el parámetro β (que explicaremos en breve) y la elección del modelo M. Si asumimos que los términos de ruido ε_q en la ecuación (2.66), que lo volvemos a repetir aquí:

$$\mathbf{t}_q = \mathbf{g}(\mathbf{p}_q) + \varepsilon_q,$$

son independientes y tienen una distribución gaussiana, entonces la densidad de probabilidad para los datos (llamada *función de probabilidad*),

$$P(D|\mathbf{x},\beta,M) = \frac{1}{Z_D(\beta)}\exp(-\beta E_D), \tag{2.101}$$

donde $\beta = 1/(2\sigma_\varepsilon^2)$, σ_ε^2 es la varianza de cada elemento de ε_q, E_D es el error cuadrático (como se definió en la ecuación (2.67)) que lo volvemos a repetir aquí,

$$F(\mathbf{x}) = E_D = \sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q)^T(\mathbf{t}_q - \mathbf{a}_q),$$

y

$$Z_D(\beta) = (2\pi\sigma_\varepsilon^2)^{N/2} = (\pi/\beta)^{N/2}, \tag{2.102}$$

donde $N = Q \times S^M$, como en la siguiente ecuación (que pertenece al algoritmo básico de Levenberg-Marquardt)

$$F(\mathbf{x}) = \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q) =$$

$$= \sum_{q=1}^{Q} \mathbf{e}_q^T \mathbf{e}_q = \sum_{q=1}^{Q} \sum_{j=1}^{S^M} (e_{j,q})^2 = \sum_{i=1}^{N} (v_i)^2,$$

donde Q es la cantidad (entrada/objetivo) de pares de datos de entrenamiento, como en la ecuación (2.65)

La ecuación (2.101) se llama la *función de probabilidad*. Es una función de los pesos $\mathbf{x}$ de la RNA, y describe la probabilidad de que se produzca un conjunto de datos dado, dado un conjunto específico de pesos. El método de *máxima probabilidad* selecciona los pesos para maximizar la función de probabilidad, que en este caso gaussiano es lo mismo que minimizar el error de cuadrado E_D. Por lo tanto, nuestro índice de rendimiento de la suma del error cuadrático estándar se puede derivar estadísticamente al asumir el ruido gaussiano en el conjunto de entrenamiento, y nuestra elección estándar para los pesos es la estimación de máxima probabilidad.

Ahora considere el segundo término en el lado derecho de la ecuación (2.100): $P(\mathbf{x}|\alpha, M)$. Esto se llama la *densidad previa*. Incorpora nuestro conocimiento sobre los pesos de la RNA antes de recopilar datos. Las estadísticas bayesianas nos permiten incorporar el conocimiento previo a través de la densidad previa. Por ejemplo, si asumimos que los pesos son valores pequeños centrados alrededor de cero, podríamos seleccionar una densidad previa gaussiana de media cero (zero-mean):

$$P(\mathbf{x}|\alpha, M) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W), \qquad (2.103)$$

donde $\alpha = 1/(2\sigma_w^2)$, σ_w^2 es la varianza de cada uno de los pesos, E_W es la suma cuadrática de los pesos (como fue definido en la ecuación (2.68)), y

$$Z_W(\alpha) = (2\pi\sigma_w^2)^{n/2} = (\pi/\alpha)^{n/2}, \tag{2.104}$$

donde n es el número de pesos y bias en la RNA

$$n = S^1(R+1) + S^2(S^1+1) + \cdots + S^M(S^{M-1}+1). \tag{2.105}$$

El término final del lado derecho de la ecuación (2.100) es $P(D|\alpha, \beta, M)$. Esto se denomina *evidencia* y es un término normalizador que no es una función de $\mathbf{x}$. Si nuestro objetivo es encontrar los pesos $\mathbf{x}$ que maximizan la *densidad posterior* $P(\mathbf{x}|D, \alpha, \beta, M)$, entonces no debemos preocuparnos por $P(D|\alpha, \beta, M)$. (Sin embargo, será importante más adelante para estimar α y β).

Con las suposiciones gaussianas que hicimos anteriormente, podemos reescribir la densidad posterior, utilizando la ecuación. (13.10), en la siguiente forma:

$$P(\mathbf{x}|D, \alpha, \beta, M) = \frac{\frac{1}{Z_W(\alpha)}\frac{1}{Z_D(\beta)}\exp\left(-(\beta E_D + \alpha E_W)\right)}{\text{Factor de normalización}} = \frac{1}{Z_F(\alpha, \beta)}\exp(-F(\mathbf{x})) \tag{2.106}$$

donde $Z_F(\alpha, \beta)$ es una función de α y β (pero no una función de $\mathbf{x}$), y $F(\mathbf{x})$ es nuestro índice de desempeño de regularizado, que definimos en la ecuación (2.68). Para encontrar el valor más probable para los pesos, debemos maximizar la densidad posterior $P(\mathbf{x}|D, \alpha, \beta, M)$. Esto es equivalente a minimizar el índice de desempeño de regularizado $F(\mathbf{x}) = \beta E_D + \alpha E_W$.

Por lo tanto, nuestro índice de desempeño de regularizado se puede derivar utilizando estadísticas bayesianas, con el supuesto de ruido gaussiano en el conjunto de entrenamiento y una densidad previa gaussiana para los pesos de la RNA. Identificaremos los pesos que maximicen la densidad posterior como $\mathbf{x}^{\mathrm{MP}}$, o *más probable*. Esto debe contrastarse con los pesos que maximizan la función de probabilidad: $\mathbf{x}^{\mathrm{ML}}$.

Observe cómo este marco estadístico proporciona un significado físico para los parámetros α y β. El parámetro β es inversamente proporcional a la varianza en el ruido de medición ε_q. Por lo tanto, si la varianza de ruido es grande, β será pequeña y la relación de regularización α/β será grande. Esto obligará a que los pesos resultantes sean pequeños y la función de RNA sea suave (como se ve en la Figura 2.20). Cuanto mayor sea el ruido de medición, más suavizaremos la función de RNA, para promediar los efectos del ruido.

El parámetro α es inversamente proporcional a la varianza en la distribución anterior para los pesos de la RNA. Si esta variación es grande, significa que tenemos muy poca certeza sobre los valores de los pesos de la RNA y, por lo tanto, podrían ser muy grandes. El parámetro α será pequeño y la relación de regularización α/β también será pequeña. Esto permitirá que los pesos de la RNA sean grandes, y se permitirá que la función de la RNA tenga más variación (como se ve en la Figura 2.20). Cuanto mayor sea la variación en la densidad anterior para los pesos de la RNA, mayor será la variación que la función de RNA podrá tener.

2.13.5.2 Marco bayesiano de nivel II

Hasta ahora tenemos una interesante derivación estadística del índice de desempeño de regularizado y una nueva visión de los significados de los parámetros α y β, pero lo que realmente queremos encontrar es una forma de estimar estos parámetros a partir de los datos. Para hacer esto, necesitamos llevar el análisis Bayesiano a otro nivel. Si queremos estimar α y β usando el análisis bayesiano, necesitamos la densidad de probabilidad $P(\alpha, \beta|D, M)$. Usando la regla de Bayes, esto puede escribirse

$$P(\alpha, \beta|D, M) = \frac{P(D|\alpha, \beta, M)P(\alpha, \beta|M)}{P(D|M)}. \qquad (2.107)$$

Este tiene el mismo formato que la ecuación (2.100), con la función de probabilidad y la densidad previa en el numerador del lado derecho. Si asumimos una densidad previa uniforme (constante) $P(\alpha, \beta|M)$ para los parámetros de regularización α y β, entonces la maximización posterior se logra maximizando la función de probabilidad $P(D|\alpha, \beta, M)$. Sin embargo, tenga en cuenta que esta función de probabilidad es el factor de normalización (evidencia) de la ecuación (2.100). Dado que hemos asumido que todas las probabilidades tienen una forma gaussiana, conocemos la forma de la densidad posterior de la ecuación (2.100). Se muestra en la ecuación (2.106). Ahora podemos resolver la ecuación (2.100) para el factor de normalización (evidencia).

$$P(D|\alpha, \beta, M) = \frac{P(D|\mathbf{x}, \beta, M)P(\mathbf{x}|\alpha, M)}{P(\mathbf{x}|D, \alpha, \beta, M)}$$

$$= \frac{\left[\frac{1}{Z_D(\beta)}\exp(-\beta E_D)\right]\left[\frac{1}{Z_W(\alpha)}\exp(-\alpha E_W)\right]}{\frac{1}{Z_F(\alpha, \beta)}\exp\big(-F(\mathbf{x})\big)} \tag{2.108}$$

$$= \frac{Z_F(\alpha, \beta)}{Z_D(\beta)Z_W(\alpha)} \cdot \frac{\exp(-\beta E_D - \alpha E_W)}{\exp\big(-F(\mathbf{x})\big)} = \frac{Z_F(\alpha, \beta)}{Z_D(\beta)Z_W(\alpha)}$$

Tenga en cuenta que conocemos las constantes $Z_D(\beta)$ y $Z_W(\alpha)$ de la ecuación (2.102) y la ecuación (*2.104)*. La única parte que no sabemos es $Z_F(\alpha, \beta)$. Sin embargo, podemos estimarla utilizando una expansión de la serie de Taylor.

Como la función objetivo tiene la forma cuadrática en un área pequeña que rodea un punto mínimo, podemos expandir $F(\mathbf{x})$ en una serie de Taylor de segundo orden (consulte la ecuación (2.11)) alrededor de su punto mínimo, $\mathbf{x}^{MP}$, donde el gradiente es cero:

$$F(\mathbf{x}) \approx F(\mathbf{x}^{MP}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{MP})^T \mathbf{H}^{MP}(\mathbf{x} - \mathbf{x}^{MP}), \tag{2.109}$$

donde $\mathbf{H} = \beta\nabla^2 E_D + \alpha\nabla^2 E_W$ es la matriz Hessian de $F(\mathbf{x})$, y $\mathbf{H}^{MP}$ es la Hessian evaluado en $\mathbf{x}^{MP}$. Podemos ahora substituir esta aproximación en la expresión para la densidad posterior, ecuación (2.106):

$$P(\mathbf{x}|D,\alpha,\beta,M) = \frac{1}{Z_F}\exp\left[-F(\mathbf{x}^{MP}) - \frac{1}{2}(\mathbf{x}-\mathbf{x}^{MP})^T\mathbf{H}^{MP}(\mathbf{x}-\mathbf{x}^{MP})\right], \quad (2.110)$$

que puede ser reescrita como

$$P(\mathbf{x}|D,\alpha,\beta,M) \approx \left\{\frac{1}{Z_F}\exp\left(-F(\mathbf{x}^{MP})\right)\right\}\exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}^{MP})^T\mathbf{H}^{MP}(\mathbf{x}-\mathbf{x}^{MP})\right]. \quad (2.111)$$

La forma estándar de la densidad Gaussiana es

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n|(\mathbf{H}^{MP})^{-1}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}^{MP})^T\mathbf{H}^{MP}(\mathbf{x}-\mathbf{x}^{MP})\right). \quad (2.112)$$

por lo tanto, la ecuación (2.112) con la ecuación (2.106), podemos resolverlo para $Z_F(\alpha,\beta)$:

$$Z_F(\alpha,\beta) \approx (2\pi)^{\frac{n}{2}}(\det((\mathbf{H}^{MP})^{-1}))^{\frac{1}{2}}\exp\left(-F(\mathbf{x}^{MP})\right). \quad (2.113)$$

Colocando este resultado en la ecuación (2.108), podemos resolver los valores óptimos para α y β en el punto mínimo. Hacemos esto tomando la derivada con respecto a cada uno de los registros de la ecuación (2.108) y ponerlos a cero.

$$\alpha^{MP} = \frac{\gamma}{2E_W(\mathbf{x}^{MP})} \text{ and } \beta^{MP} = \frac{N-\gamma}{2E_D(\mathbf{x}^{MP})}, \quad (2.114)$$

donde $\gamma = n - 2\alpha^{MP}\mathrm{tr}(\mathbf{H}^{MP})^{-1}$ es llamado el *número efectivo de parámetros, y n* es el número total de parámetros en la RNA. El término γ es una medida de cuántos parámetros (pesos y bias) en la RNA se usan efectivamente para reducir la función de error. Puede ir de cero a n. (Vea el ejemplo en la página 13-23 para más análisis de.)

2.13.5.3 Algoritmo general de regularización Bayesiana

La optimización bayesiana de los parámetros de regularización requiere el cálculo de la matriz de Hesse de $F(\mathbf{x})$ en el punto mínimo $\mathbf{x}^{MP}$. Proponemos utilizar la aproximación de Gauss-Newton a la matriz de Hesse, que está disponible fácilmente si el algoritmo de optimización Levenberg-Marquardt se utiliza para localizar el punto mínimo (consulte la ecuación (2.52)). El cómputo adicional requerido para la optimización de la regularización es mínimo.

Estos son los pasos necesarios para la optimización bayesiana de los parámetros de regularización, con la aproximación de Gauss-Newton a la matriz de Hesse:

0. Inicializar, α y β y los pesos. Los pesos se inicializan aleatoriamente, y luego se calculan E_D y E_W. Establezca $\gamma = n$, y calcule α y β usando la ecuación (2.114).
1. Tome un paso del algoritmo de Levenberg-Marquardt para minimizar la función objetivo $F(x) = \beta E_D + \alpha E_W$.
2. Calcule el número efectivo de parámetros $\gamma = n - 2\alpha \mathrm{tr}(\mathbf{H})^{-1}$, haciendo uso de la aproximación de Gauss-Newton al Hessiano, disponible en el algoritmo de entrenamiento de Levenberg-Marquardt:
   $\mathbf{H} = \nabla^2 F(\mathbf{x}) \approx 2\beta \mathbf{J}^{\mathrm{T}}\mathbf{J} + 2\alpha \mathrm{I}_{\mathrm{n}}$, donde $\mathbf{J}$ es la matriz jacobiana de los errores del conjunto de entrenamiento (consulte la ecuación (2.59)).
3. Calcular nuevas estimaciones para los parámetros de regularización $\alpha^{MP} = \frac{\gamma}{2E_W(\mathbf{x})}$ and $\beta^{MP} = \frac{N-\gamma}{2E_D(\mathbf{x})}$.
4. Ahora itere los pasos del 1 al 3 hasta la convergencia.

Tenga en cuenta que con cada reestimación de los parámetros de regularización α y β, la función objetivo $F(\mathbf{x})$ cambia; por lo tanto, el punto mínimo se está moviendo. Si

atravesara la superficie de rendimiento generalmente se mueve hacia el siguiente punto mínimo, entonces las nuevas estimaciones para los parámetros de regularización serán más precisas. Eventualmente, la precisión será lo suficientemente buena como para que la función objetivo no cambie significativamente en las iteraciones posteriores. Así, obtendremos convergencia.

Cuando se utiliza este algoritmo de aproximación de Gauss-Newton a la regularización bayesiana (GNBR), se obtienen los mejores resultados si los datos de entrenamiento se asignan primero en el rango $[-1,1]$ (o alguna región similar).

En la Figura 2.27 puede ver los resultados del entrenamiento de una RNA 1-20-1 con GNBR en el mismo conjunto de datos representado en Figura 2.20. La RNA se ha adaptado a la función subyacente, sin adaptarse excesivamente al ruido. El ajuste parece similar al obtenido en la Figura 2.20, con la relación de regularización establecida en $\alpha/\beta = 0.01$. De hecho, al finalizar el entrenamiento con GNBR, la proporción de regularización final para este ejemplo fue $\alpha/\beta = 0.0137$.

El proceso de entrenamiento para este ejemplo se ilustra en la Figura 2.28. En la parte superior izquierda de esta Figura 2.28, se ve el error cuadrático en el conjunto de entrenamiento. Tenga en cuenta que no necesariamente disminuye en cada iteración. En la parte superior derecha de la Figura 2.28, se ve el error cuadrático de testeo. Esto se obtuvo comparando la función de RNA con la función real en una cantidad de puntos entre -1 y 1. Es una medida de la capacidad de generalización de la RNA. (Esto no sería posible en un caso práctico, donde la verdadera función era desconocida). Tenga en cuenta que el error de prueba es mínimo al finalizar el entrenamiento.

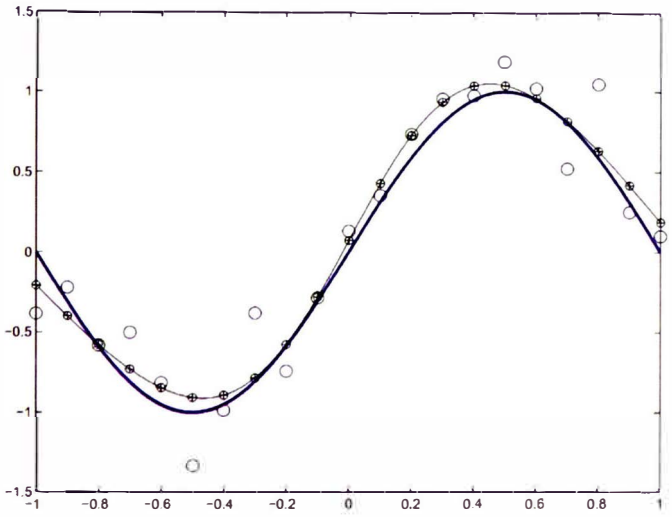


Figura 2.27. Ajuste por Regularización Bayesiana, *[115]*.

La Figura 2.28 también muestra la relación de regularización α/β y el número efectivo de parámetros γ durante el entrenamiento. Estos parámetros no tienen un significado particular durante el proceso de entrenamiento, pero al finalizar el entrenamiento son significativos. Como mencionamos anteriormente, el índice de regularización final fue $\alpha/\beta = 0.0137$, lo cual es consistente con nuestra investigación anterior de regularización, ilustrada en la Figura 2.20. El número efectivo final de parámetros fue $\gamma = 5.2$. Esto es de un total de 61 pesos y bias totales en la RNA.

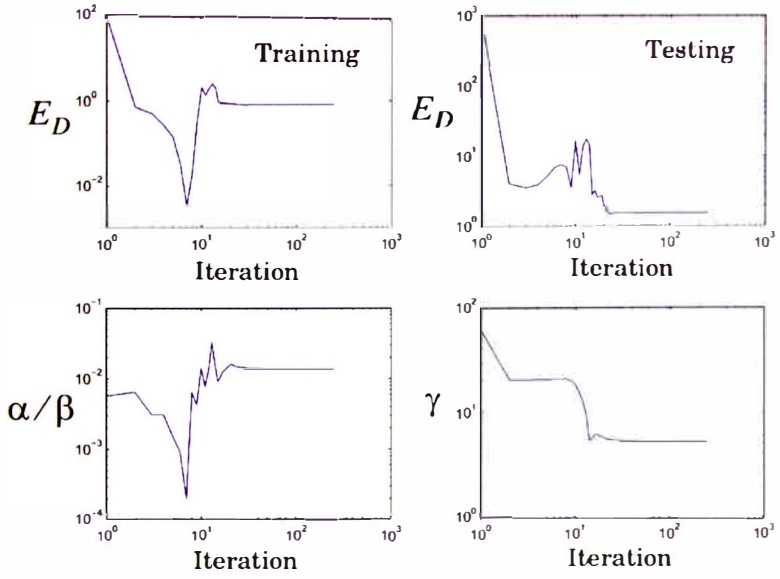


Figura 2.28. Proceso de entrenamiento usando Regularización Bayesiana, *[115]*.

El hecho de que en este ejemplo el número efectivo de parámetros sea mucho menor que el número total de parámetros (6 en comparación con 61) significa que bien podríamos haber utilizado una RNA más pequeña para ajustar estos datos. Hay dos desventajas de una RNA grande: 1) puede sobre ajustar los datos, y 2) requiere más cálculos para calcular la salida de la RNA. Hemos superado la primera desventaja entrenando con GNBR; Aunque la RNA tiene 61 parámetros, es equivalente a una RNA con solo 6 parámetros. La segunda desventaja solo es importante si el tiempo de cálculo para la respuesta de la RNA es crítico para la aplicación. Este no suele ser el caso, ya que el tiempo para calcular una respuesta de RNA a una entrada particular se mide en milisegundos. En aquellos casos en que el tiempo de cálculo es significativo, puede entrenar a una RNA más pequeña en el conjunto de datos.

Por otro lado, cuando el número efectivo de parámetros está cerca del número total de parámetros, esto puede significar que la RNA no es lo suficientemente grande como para ajustar los datos. En este caso, debe aumentar el tamaño de la RN a entrenar en el conjunto de datos, [116].

CAPITULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1 Tipo y diseño de la investigación

La investigación es experimental. Para cumplir los objetivos, descritos en el Capítulo I, se iniciará con la adquisición de datos mediante una DAQ, seguidamente se tratará dichos datos, estos datos serán separados en dos grupos tanto para el entrenamiento y el testeo del entrenamiento de la RNA. Posteriormente seleccionaremos una arquitectura de RNA apropiada, así como un método de entrenamiento y optimización, finalmente se va a validar la RNA entrenada mediante una comparación del resultado esperado y resultado obtenido (los datos para la validación no se utilizarán en el entrenamiento de la RNA). Una vez la RNA pase con éxito la validación minuciosa, ésta podrá utilizarse en el rango de entrada para el cual fue entrenada. Finalmente, la RNA será ejecutada en un circuito electrónico, y será comparado con otros equipos de medición de PH.

A = Conjunto de datos para entrenamiento

B= Conjunto de datos para el testeo (no es utilizada para el entrenamiento)

El conjunto de datos "A" será utilizado para el entrenamiento, el conjunto de datos "B" será utilizado para la validación de la RNA entrenada.

3.2 Alcances

El alcance está limitado a obtener la curva característica del sensor de Potencial de Hidrógeno, pH, de la empresa Libelium y la construcción de un equipo electrónico que ejecute la RNA entrenada para la medición de temperatura y pH. El sensor de pH depende

de 6 variables (temperatura, lectura del sensor de pH, y 4 puntos de calibración) para indicar el nivel de pH de un líquido.

La adquisición de datos se realiza mediante los softwares LabVIEW, MATLAB y el dispositivo NI myDAQ.

Se adjunta el código de programación del microcontrolador utilizado en la tarjeta desarrollada y los script MATLAB para tratamiento de datos, entrenamiento y validación de la RNA.

No se tomará en cuenta la posible histéresis[25] que pueda sufrir el sensor de pH y el sensor PT1000 te temperatura (ambos sensores vienen de fábrica junto con la tarjeta electrónica Smart Water).

3.3 Limitaciones

El rango de medición de un sensor común de pH abarca el rango de 0 a 14, sin embargo, en este trabajo se va a determina la curva característica par un rango entre 0.6 a 13.9 de pH a una temperatura entre 0.2°C y 95.5°C. Para temperaturas fuera de estos límites no se garantiza el correcto desempeño de la RNA.

Los resultados de este trabajo están orientados a un sensor pH de tipo electrodo combinado, que es suministrado por Libelium. Esta limitación se debe a que cada fabricante construye los sensores de pH con sus propias especificaciones, incluso dos sensores similares fabricados por una misma empresa pueden comportarse de manera diferente, ya que su comportamiento depende del estado de los elementos químicos que utiliza para realizar su trabajo.

[25] Es un fenómeno que intrínseco de los materiales, en la que la relación de salida/entrada a veces revelará resultados diferentes a medida que las señales varían en la dirección del movimiento [137].

CAPITULO IV

DESARROLLO DEL PROBLEMA

4.1 Presentación de la solución del problema

Desarrollo de una red neuronal artificial que identifique la curva característica de un sensor de potencial de hidrógeno pH, e implementación de una tarjeta electrónica que ejecute la red neuronal artificial entrenada.

4.2 Descripción detallada del plan de acción

El plan de acción se divide en cuatro etapas principales, las cuales son el preentrenamiento, entrenamiento y análisis post entrenamiento, y diseño y construcción de una tarjeta electrónica que ejecute el algoritmo de la red neuronal entrenada, a continuación, se describe cada uno de ellos:

4.2.1 Preentrenamiento de la red neuronal artificial

- Se tomará datos de entrada que consiste en variaciones de voltajes en los sockets de entrada de la tarjeta electrónica correspondientes a los sensores de pH y temperatura (estas variaciones de voltaje simulan las variaciones del entorno al que son expuesto estos sensores), además de los cuatro puntos de calibración[26] del sensor de pH. Adicionalmente se tomarán datos de salida de la tarjeta electrónica correspondientes a los datos de entrada (variaciones de voltaje). Esto quiere decir que se toma datos de salida por su correspondiente conjunto de datos de entrada.
- Se tomará aproximadamente 20 datos equidistantes de pH para cada variación de temperatura de 5°C, así como para la variación 0.04 para los parámetros de calibración pH4, pH7, pH10, y una variación de 5°C para el parámetro de

[26] Estos puntos de calibración se establecen en el software de programación de la tarjeta electrónica Waspmote.

variación de temperatura. En caso durante la validación y testeo de la RNA se identifique que esta cantidad de datos es insuficiente, se volverá a tomar más datos hasta que sean lo suficientemente como para que la RNA pueda generalizar satisfactoriamente.

- Se eliminará los datos atípicos, ya que a menudo estos se generan por el ingreso de ruido durante la toma de datos.
- Se normalizará los datos, tanto de entrada como de salida (objetivos) con el fin de evitar posibles saturaciones de las funciones de activación.
- Se apartará conjuntos de datos para el entrenamiento, validación y testeo. El conjunto de datos para testeo se utilizará para determinar si la Red Neuronal Artificial generaliza bien en el rango de entrenamiento.
- Se utilizará una arquitectura de RNA multicapa para el aprendizaje de la RNA, tomando en cuenta que tenemos 6 entradas y una sola salida. Adicionalmente, no podemos predecir si las entradas y la correspondiente salida son lineal o no lineal. Sabremos si hemos elegido bien o mal la arquitectura de la RNA durante el testeo. Análisis estadísticos para tener una referencia si hemos elegido bien o mal la cantidad de neuronas en la capa oculta (ya que la cantidad de neuronas en la capa de salida está determinada por el miso problema). En caso de que los análisis nos indiquen una mala elección, volveremos a elegir una nueva arquitectura. Recordemos que el entrenamiento de una red neuronal artificial es cíclico, y se tiene que probar con diferentes arquitecturas e inicialización de parámetros ya que la cantidad de entradas, salidas, tipo de trabajo que va a realizar solo nos da nociones básicas para la elección de la arquitectura, inicializaciones de parámetros y funciones de activación.

4.2.2 Entrenamiento de la red neuronal artificial

- Se inicializará los pesos de la RNA utilizando el método Widrow y Nguyen.
- Se entrenará la RNA utilizando algún algoritmo de entrenamiento apropiado según el tipo de problema que intentamos solucionar.
- Se realizará múltiples ejecuciones del algoritmo para diferentes números de neuronas de la capa, o capas ocultas, según se elija.

4.2.3 Análisis Post-Entrenamiento

- Se analizará los resultados de diferentes ejecuciones para diferentes números de neuronas en la capa oculta. Estos resultados tomarán en cuenta la suma de errores cuadráticos, la suma de parámetros cuadráticos, el error máximo del testeo de la RNA, error medio del testeo de la RNA y número efectivo de parámetros.
- Se realizará un análisis comparativo de los objetivos y las salidas de la RNA entrenada para el conjunto de datos de testeo.
- Se realizará un análisis de la regresión lineal entre los objetivos y la salida de la RNA para el conjunto de datos de testeo.
- Se realizará el análisis del coeficiente de correlación y coeficiente de determinación.
- Se construirá un Histograma de errores de la RNA con el fin de identificar valores atípicos.

4.2.4 Diseño y fabricación de una tarjeta electrónica que ejecute el algoritmo de la red neuronal entrenada.

- Se diseñará los circuitos y PCB footprints de la tarjeta electrónica haciendo uso del software especializado Proteus.

- Se ensamblará, testeará, y se comparará su desempeño con un dispositivo medidor de pH de fábrica.

4.3 Desarrollo detallado del problema

4.3.1 Descripción del sistema a intervenir

El sistema está compuesto por la tarjeta electrónica Smart Water, tarjeta electrónica Waspmote (figura 4.2.), sensor de pH, sensor de temperatura y su correspondiente IDE de usuario.

A la tarjeta electrónica Smart Water se conecta el sensor de pH y temperatura mediante el Socket de pH y Socket de temperatura respectivamente. En la Figura 4.1 la conexión I2 es el GND de la DAQ y la placa Smart Water (0 VDC.), y se toma como referencia para inyección de voltaje al socket de pH y al socket de temperatura.

Por la conexión I1 la DAQ inyecta tención al socket de temperatura, para simular la variación de resistencia del sensor PT1000.

Por la conexión I3 la DAQ inyecta tención al socket de pH, para simular la variación de tensión del sensor de pH.

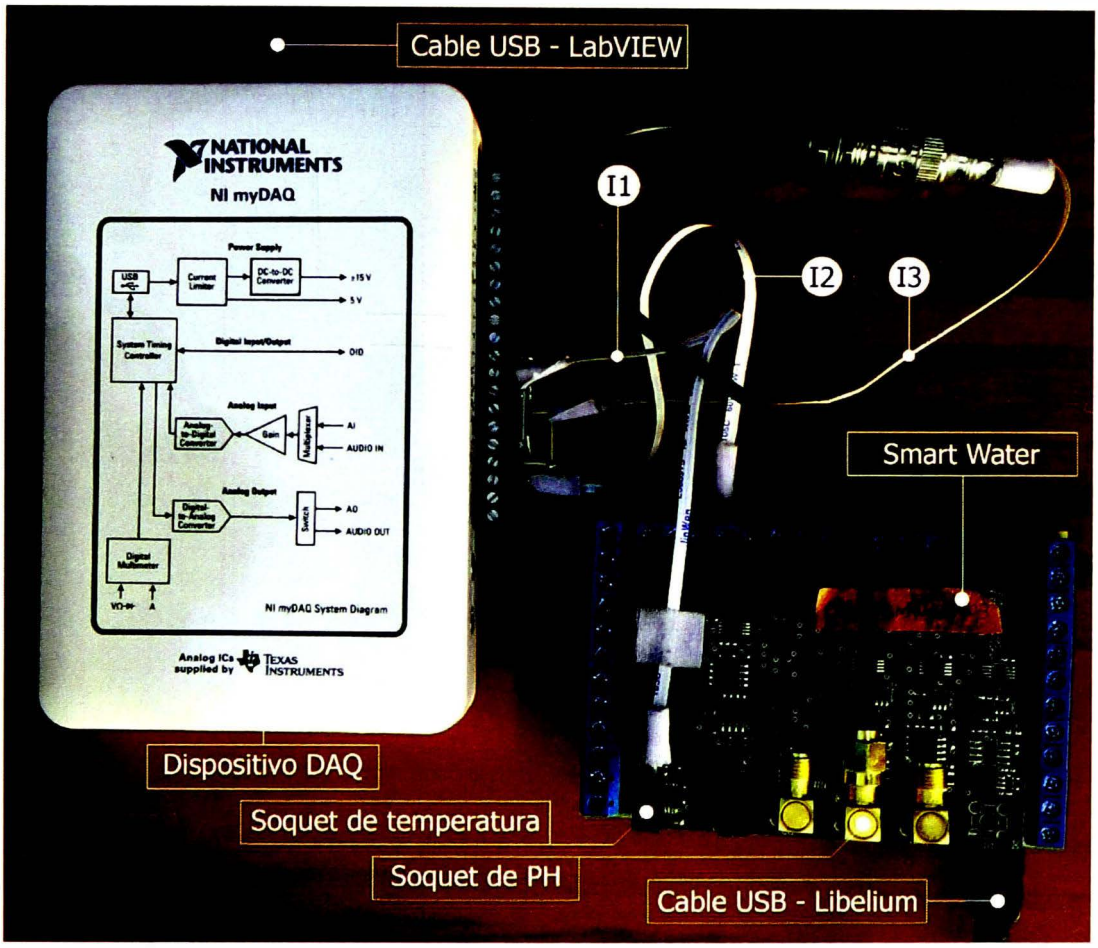


Figura 4.1. Dispositivo DAQ y tarjeta Smart Water[27](fuente: elaboración propia del autor)

Al inyectar diferentes valores de voltaje al socket de pH y de temperatura se obtiene diferentes valores de lectura de pH y de temperatura en el puerto COM, estos datos en el puerto COM son leídos y almacenados por el software LabVIEW.

La tarjeta electrónica Smart Water solo acondiciona las señales entrantes en los sockets, otra tarjeta electrónica llamado Waspmote acoplado en la parte inferior a la tarjeta Smart Water, Figura 4.2, es quien convierte las señales de analógico a digital, las interpreta y los envía al puerto COM.

[27] Imagen de elaboración propia.

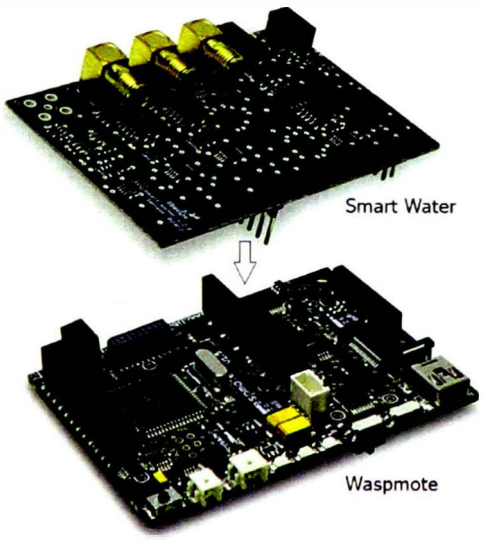


Figura 4.2. Tarjeta electrónica Smart Water y Waspmote (imagen referencial), (fuente: elaboración propia del autor).

La tarjeta Waspmote y Smart Water requiere que el sensor de pH sea calibrado, para ello dispone de 4 valores de calibración (vea la Figura 4.3). Para la adquisición de datos, estos 4 parámetros de calibración deben permanecer constantes y tomar datos de temperatura en el rango de 0°C a 100°C y pH en rango de 0-14 haciendo variar los voltajes en sus correspondientes sockets de entradas en la tarjeta Smart Water. Una vez se haya cubierto los rangos de temperatura y de pH, se cambia ligeramente los 4 parámetros de calibración y se vuelve a tomar datos de temperatura en el rango de 0°C a 100°C y pH en rango de 0-14 haciendo variar los voltajes en sus correspondientes sockets de entradas. Este proceso continúa hasta que se haya adquirido la cantidad necesaria de datos para el entrenamiento de la RNA.

```
//Calibration values
#define cal_point_10  2.025
#define cal_point_7   2.065
#define cal_point_4   2.225
#define cal_temp      30
```

Figura 4.3. Puntos de calibración – Waspmote - Smart-Water (fuente: elaboración propia del autor)

4.3.2 Preentrenamiento de la red neuronal artificial

Para el entrenamiento de la red neuronal artificial seguiremos una serie de pasos ordenados, los cuales se detallan a continuación.

4.3.2.1 Descripción detallada del programa computacional de adquisición de datos

En la Figura 4.4 se muestra una vista general del diagrama de bloques para la adquisición de datos en LabVIEW.

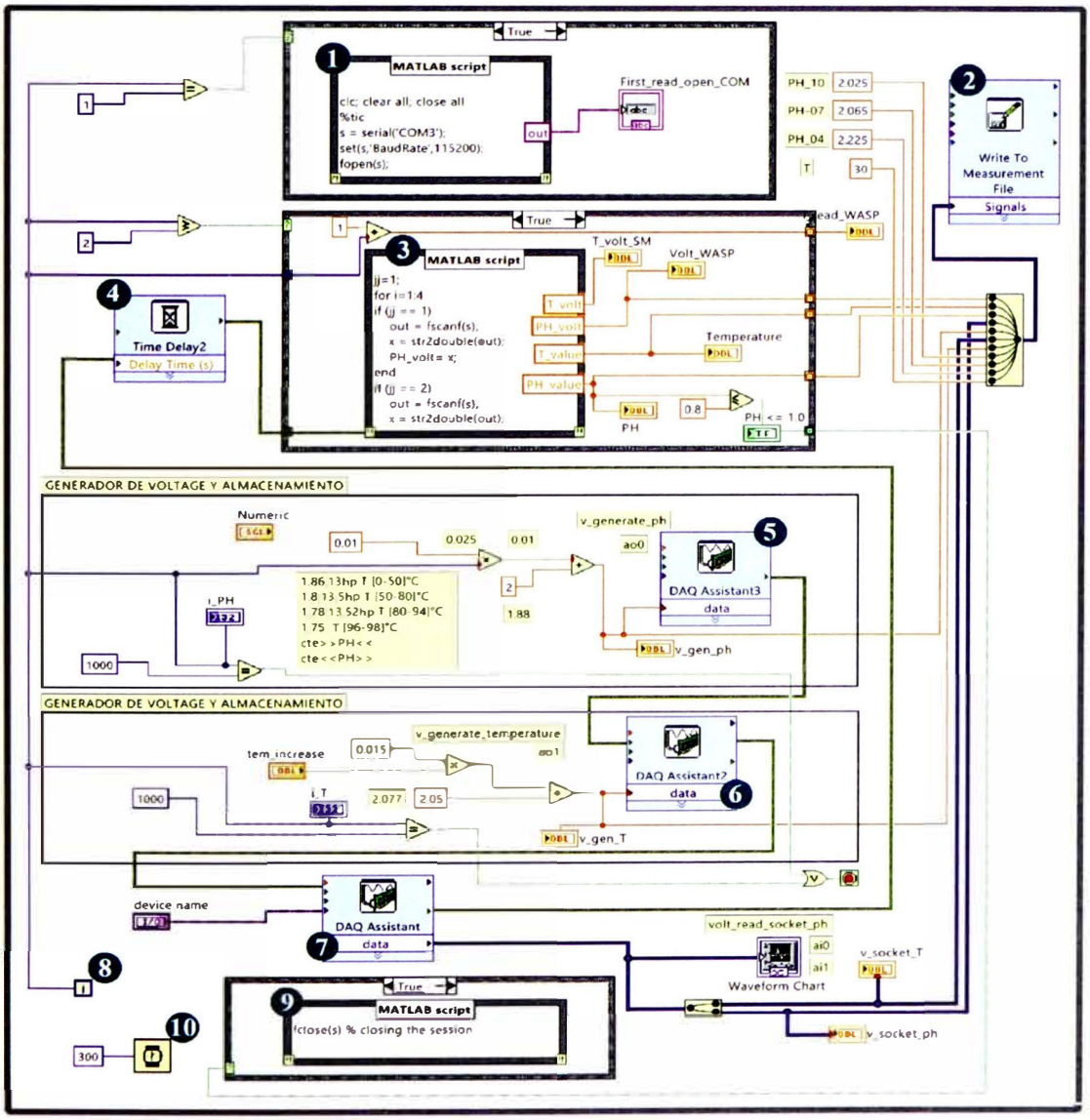


Figura 4.4. Diagrama de bloques de adquisición de datos en LabVIEW (fuente: captura de pantalla, elaborado por el autor)

A continuación, se procederá a describir detalladamente las partes más importantes del diagrama bloques para la adquisición de datos en LabVIEW.

a) Como primer paso se abre el puerto COM y a la vez genera los voltajes que serán inyectados en el socket de pH y el socket de temperatura del Smart Water.

1: Para la apertura del puerto COM se utiliza un *MatLab script node* el cual llama al software MATLAB® para ejecutar el script contenido en su interior.

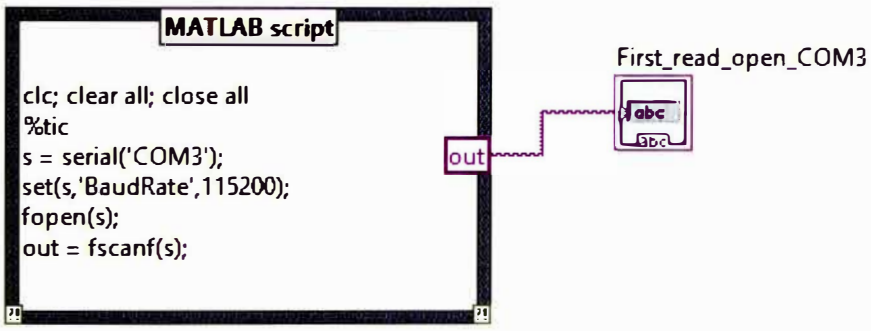


Figura 4.5. MatLab script node que permite abrir el puerto COM (fuente: captura de pantalla, elaborado por el autor)

5: Generación de voltaje en la salida analógica ao0 del dispositivo de adquisición de datos NI myDAQ con la ayuda del *DAQ Assistant* el cual crea, edita y ejecuta tareas utilizando NI-DAQmx. El voltaje de obtenido en la salida analógica ao0 se inyecta al socket de pH del Smart Water.

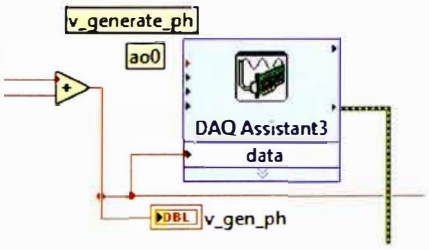


Figura 4.6. DAQ Assistant que permite generar voltaje en la salida analógica ao0. (fuente: captura de pantalla, elaborado por el autor)

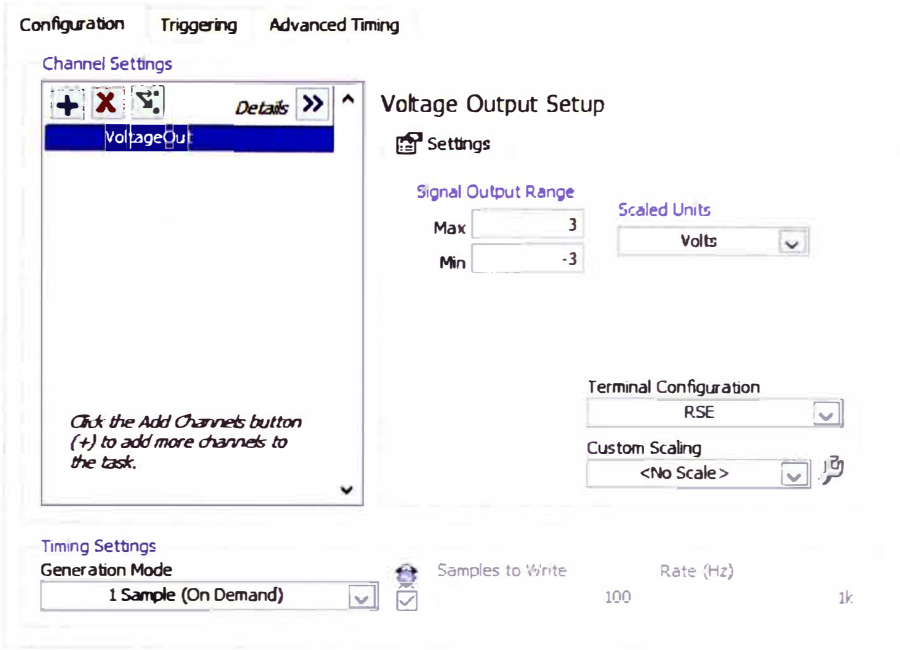


Figura 4.7. Configuración del voltaje de salida del DAQ Assistant que permite generar voltaje en la salida analógica ao0. (fuente: captura de pantalla, elaborado por el autor)

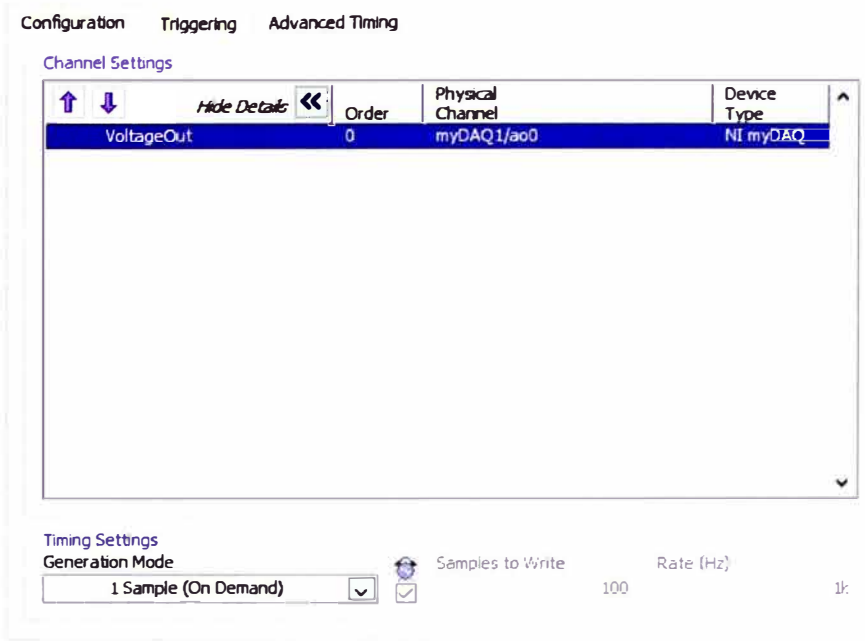


Figura 4.8. Configuración del canal de salida del DAQ Assistant que permite generar voltaje en la salida analógica ao0. (fuente: captura, elaborado por el autor)

6: Generación de voltaje en la salida analógica ao1 del dispositivo de adquisición de datos NI myDAQ con la ayuda del *DAQ Assistant* el cual crea, edita y ejecuta tareas utilizando NI-DAQmx. El voltaje de obtenido en la salida analógica ao1 se inyecta al socket de temperatura, T, del Smart Water.

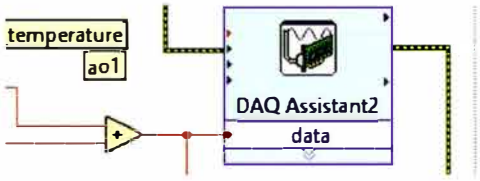


Figura 4.9. DAQ Assistant que permite generar voltaje en la salida analógica ao1. (fuente: captura de pantalla, elaborado por el autor)

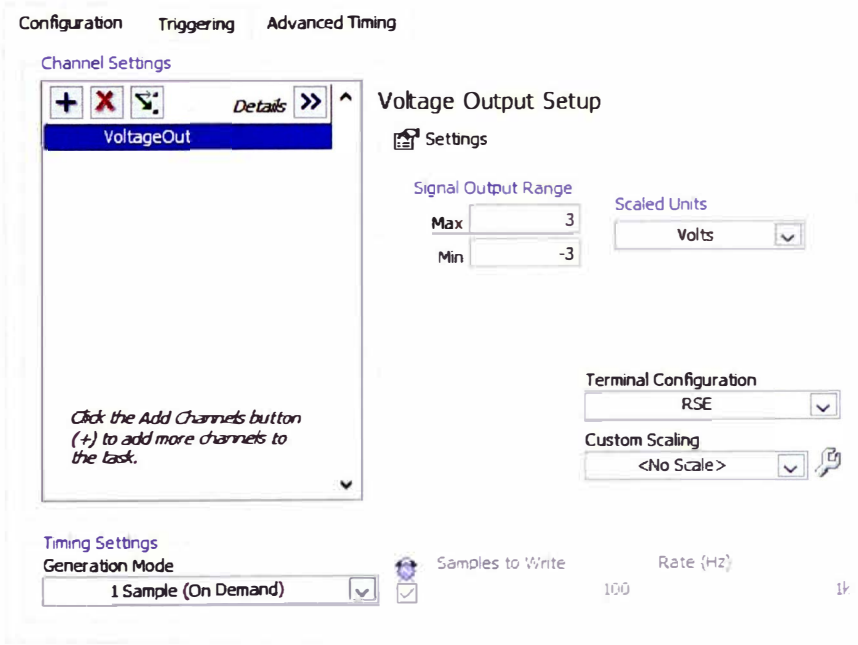


Figura 4.10. Configuración del voltaje de salida del DAQ Assistant que permite generar voltaje en la salida analógica ao1. (fuente: captura de pantalla, elaborado por el autor)

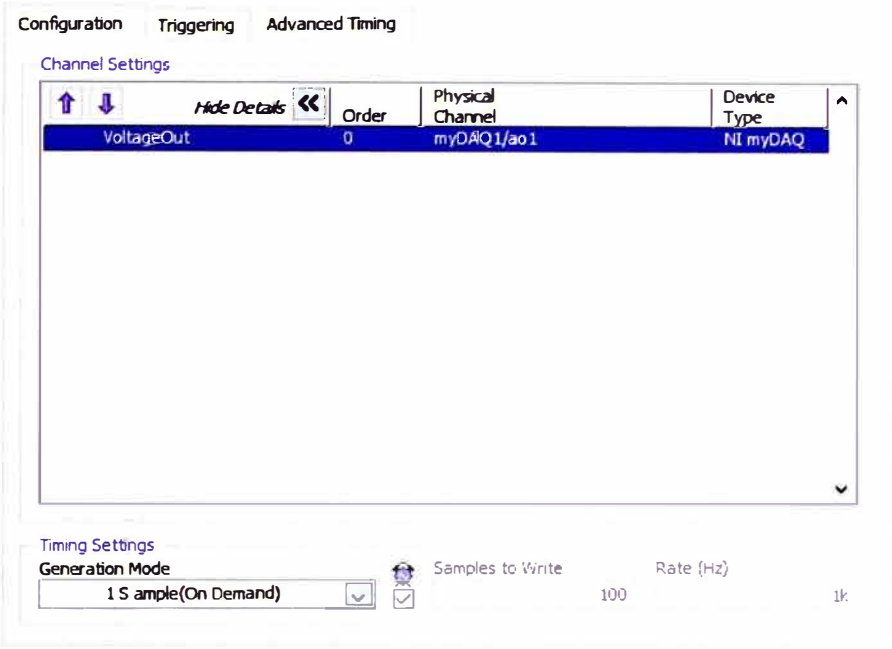


Figura 4.11. Configuración del canal de salida del DAQ Assistant que permite generar voltaje en la salida analógica ao1. (fuente: captura de pantalla, elaborado por el autor)

b) Lectura de voltajes que se ingresan al Smart Water.

7: Lectura de voltajes ingresados al socket del sensor de pH y al socket del sensor de temperatura T del Smart Water, utilizando las entradas analógicas ai0 y ai1 del dispositivo de adquisición de datos NI myDAQ con la ayuda del *DAQ Assistant* el cual crea, edita y ejecuta tareas utilizando NI-DAQmx.

Figura 4.12. DAQ Assistant que permite la lectura de voltajes que son ingresados al socket del sensor de pH y al socket del sensor de temperatura. (fuente: captura de pantalla del diagrama bloques, elaborado por el autor)

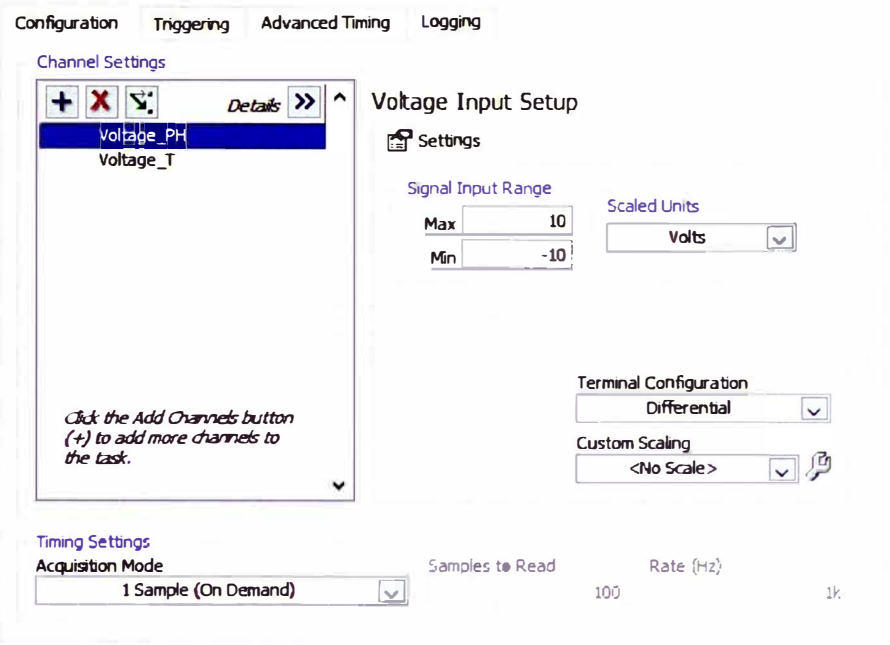


Figura 4.13. Configuración de los voltajes de entrada ai0 y ai1 del DAQ Assistant que permite la lectura de voltajes ingresados al socket del sensor de pH y al socket del sensor de temperatura. (fuente: captura de pantalla, elaborado por el autor)

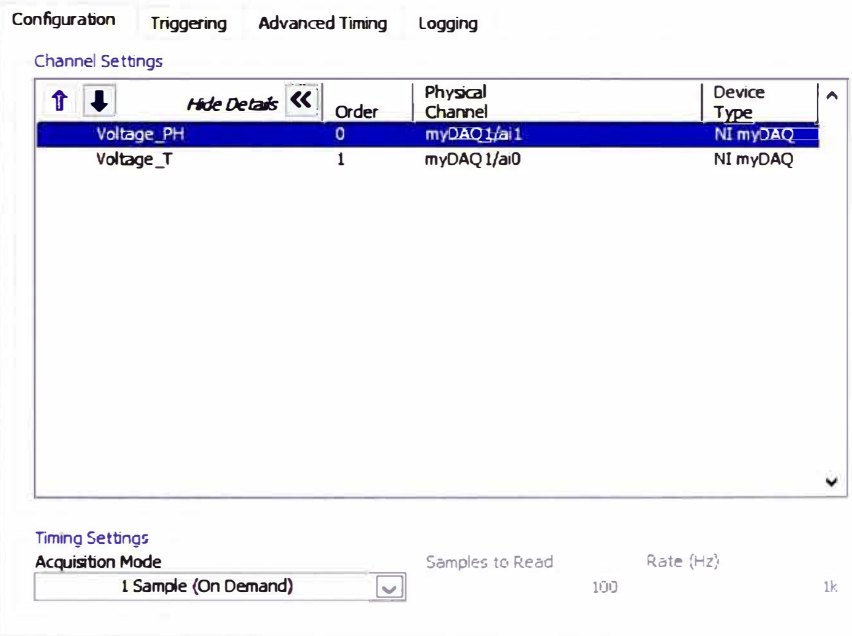


Figura 4.14. Configuración de los canales de entrada ai0 y ai1 del DAQ Assistant que permite la lectura de voltajes ingresados al socket del sensor de pH y al socket del sensor de temperatura. (fuente: captura de pantalla, elaborado por el autor)

c) Lectura del puerto COM

3: Para la lectura del puerto COM se utiliza un *MatLab script node* el cual llama al software MATLAB® para ejecutar el script contenido en su interior. En la Figura 4.16 se muestra detalladamente el script para la lectura de datos presentes en el del puerto COM. Estos datos presentes en el puerto COM son enviados por el programa que controla al Smart Water. Un programa referencial del Smart Water que envía datos al puerto COM se muestra en el Anexo J.

El escript se ejecuta, y con ello la lectura del puerto COM, 1 segundo después de la inyección de tensión en los sockets de pH y temperatura, dado tiempo al Smart Water para que realice los cálculos y envíe los resultados al puerto COM. El retardo de 1 segundo se realiza con un bloque *time delay* Time Delay2, que se observa en la Figura 4.4.

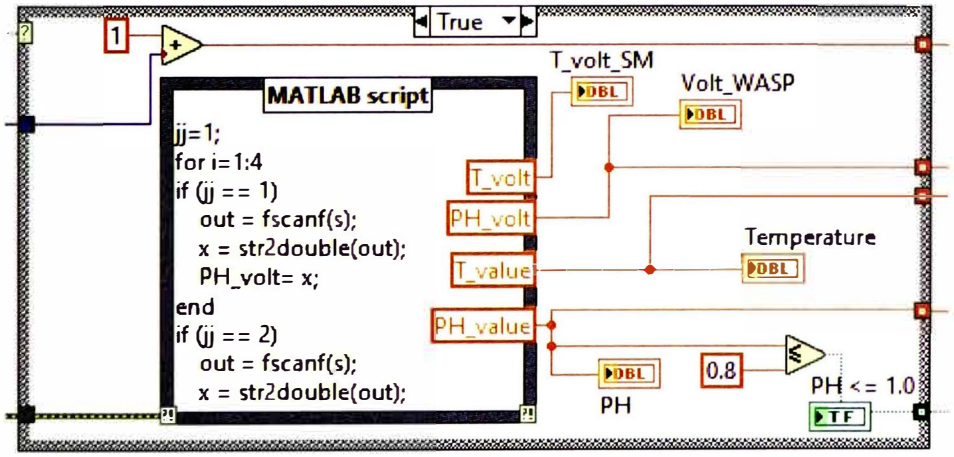


Figura 4.15. MatLab script node que permite leer el puerto COM (fuente: captura de pantalla del diagrama bloques para la adquisición de datos en LabVIEW elaborado por el autor)

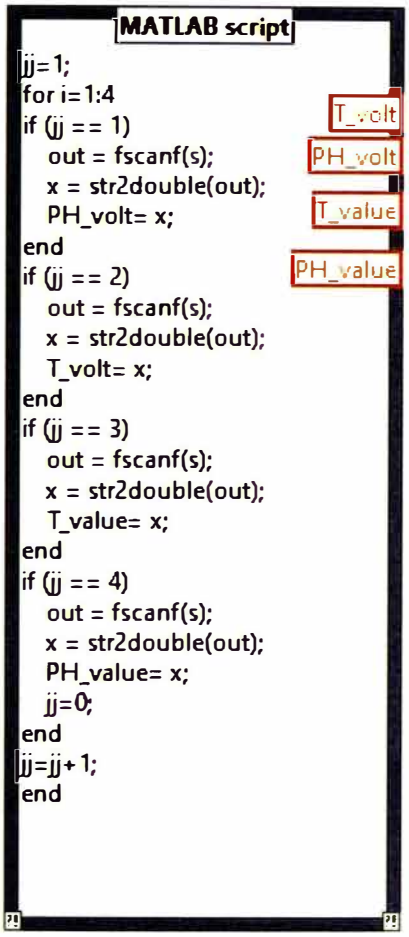


Figura 4.16. MatLab script node con el script detallado que permite leer el puerto COM (fuente: captura de pantalla del diagrama bloques para la adquisición de datos en LabVIEW elaborado por el autor)

d) Almacenamiento de datos

2: El almacenamiento de datos se realiza con el bloque *Write To Measurement File* el cual escribe datos en archivos de medición basados en texto (.lvm), archivos de medición binarios (.tdm o .tdms) o archivos de Microsoft Excel (.xlsx) lo que nos permite almacenar los voltajes generados y los valores leídos del puerto COM. En este caso utilizaremos en archivos de medición basados en texto (.lvm).

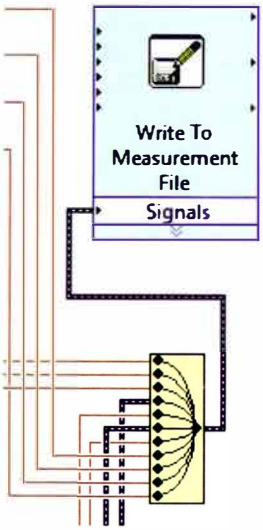


Figura 4.17. Bloque *Write To Measurement File* que nos permite almacenar los valores de voltajes ingresados a los sockets de temperatura y de pH, así como los datos leídos en el puerto COM. (fuente: captura de pantalla, elaborado por el autor)

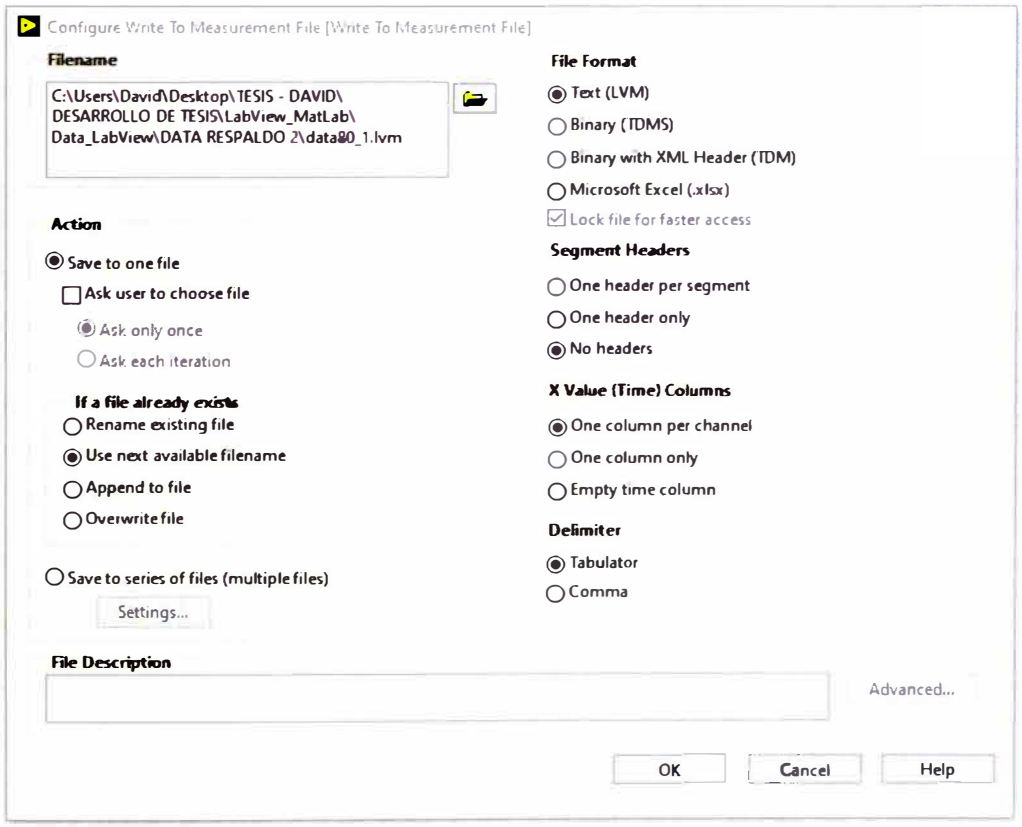


Figura 4.18. Configuración del bloque Write To Measurement File, el cual escribirá datos en archivos de medición basados en texto (.lvm) sin cabeceras y un canal por columna. (fuente: captura de pantalla, elaborado por el autor)

4.3.2.2 Recolección de datos

Normalmente se parte con poco conocimiento previo de la RNA, por lo que es muy importante la calidad de los datos recogidos.

El software de programación de la tarjeta electrónica Waspmote consta de 4 puntos de calibración, como se muestra en la Figura 4.19, con los que se calibra el sensor de pH haciendo uso de líquidos patrones (tampones) con pH 4, pH 7, pH 10 y la temperatura T (temperatura) a la que está siendo calibrado.

La lectura real de pH depende de la variación de tensión del sensor de pH y la variación de resistencia del sensor de temperatura PT1000, aparte los 4 puntos de calibración mencionados previamente.

Los puntos de calibración se tienen que cargar en la tarjeta electrónica Waspmote antes de poner en servicio. La calibración del sensor de pH se detalla en el apartado *Calibración del sensor de pH Libelium* descrito en el marco teórico.

```
//Calibration values
#define cal_point_10  2.025
#define cal_point_7   2.065
#define cal_point_4   2.225
#define cal_temp      30
```

Figura 4.19. Puntos de calibración del sensor de pH, (fuente: captura de pantalla del software de calibración del sensor de pH, elaboración propia del autor).

Al variar la resistencia del sensor de temperatura se traduce en una variación de tensión, (se simulará la variación de la resistencia del sensor de temperatura y pH, mediante la inyección de voltaje directamente en ambos sockets) esta variación de tensión es utilizada para obtener la temperatura a la que estaría sometido el sensor PT1000.

La estrategia para la toma de datos es:

- Hacer variar los 4 puntos de calibración, para cada variación de 4 los puntos de calibración, se hará variar un voltaje en la entrada del socket de temperatura PT100 en el rango correspondiente de 0°C a 95°C, y para cada variación de

temperatura se debe variar el voltaje en la entrada del socket de pH correspondiente de 0 a 14 de pH.

- La **¡Error! No se encuentra el origen de la referencia.** muestra los puntos de calibración (en la **¡Error! No se encuentra el origen de la referencia.**, columnas de color verde), para cada uno de estos puntos de calibración se hizo variar el voltaje correspondiente a pH de 0 a 14, y el "voltaje" del sensor de temperatura de 0°C a 95°C (en intervalos de 5°C) y a la vez se toman datos para cada una de estas variaciones.

Los valores numéricos de la **¡Error! No se encuentra el origen de la referencia.** no son datos adquiridos para entrenamiento de la RNA, sino son puntos de calibración (columnas en color verde). La columna de pH y T (en color plomo) solo indican el rango en que han sido variados ambos parámetros.

Tabla 4. Puntos de calibración (verde) (fuente: elaboración propia del autor).

N°	pH_10	pH_7	pH_4	T	pH (Δ)	T (Δ)		N°	pH_10	pH_7	pH_4	T	pH (Δ)	T (Δ)
1	1.945	2.025	2.185	15	0-14	0-95		49	1.985	2.065	2.185	15	0-14	0-95
2	1.945	2.025	2.185	20	0-14	0-95		50	1.985	2.065	2.185	20	0-14	0-95
3	1.945	2.025	2.185	25	0-14	0-95		51	1.985	2.065	2.185	25	0-14	0-95
4	1.945	2.025	2.185	30	0-14	0-95		52	1.985	2.065	2.185	30	0-14	0-95
5	1.945	2.025	2.225	15	0-14	0-95		53	1.985	2.065	2.225	15	0-14	0-95
6	1.945	2.025	2.225	20	0-14	0-95		54	1.985	2.065	2.225	20	0-14	0-95
7	1.945	2.025	2.225	25	0-14	0-95		55	1.985	2.065	2.225	25	0-14	0-95
8	1.945	2.025	2.225	30	0-14	0-95		56	1.985	2.065	2.225	30	0-14	0-95
9	1.945	2.025	2.265	15	0-14	0-95		57	1.985	2.065	2.265	15	0-14	0-95
10	1.945	2.025	2.265	20	0-14	0-95		58	1.985	2.065	2.265	20	0-14	0-95
11	1.945	2.025	2.265	25	0-14	0-95		59	1.985	2.065	2.265	25	0-14	0-95
12	1.945	2.025	2.265	30	0-14	0-95		60	1.985	2.065	2.265	30	0-14	0-95
13	1.945	2.065	2.185	15	0-14	0-95		61	1.985	2.105	2.185	15	0-14	0-95
14	1.945	2.065	2.185	20	0-14	0-95		62	1.985	2.105	2.185	20	0-14	0-95
15	1.945	2.065	2.185	25	0-14	0-95		63	1.985	2.105	2.185	25	0-14	0-95
16	1.945	2.065	2.185	30	0-14	0-95		64	1.985	2.105	2.185	30	0-14	0-95

17	1.945	2.065	2.225	15	0-14	0-95		65	1.985	2.105	2.225	15	0-14	0-95
18	1.945	2.065	2.225	20	0-14	0-95		66	1.985	2.105	2.225	20	0-14	0-95
19	1.945	2.065	2.225	25	0-14	0-95		67	1.985	2.105	2.225	25	0-14	0-95
20	1.945	2.065	2.225	30	0-14	0-95		68	1.985	2.105	2.225	30	0-14	0-95
21	1.945	2.065	2.265	15	0-14	0-95		69	1.985	2.105	2.265	15	0-14	0-95
22	1.945	2.065	2.265	20	0-14	0-95		70	1.985	2.105	2.265	20	0-14	0-95
23	1.945	2.065	2.265	25	0-14	0-95		71	1.985	2.105	2.265	25	0-14	0-95
24	1.945	2.065	2.265	30	0-14	0-95		72	1.985	2.105	2.265	30	0-14	0-95
25	1.945	2.105	2.185	15	0-14	0-95		73	2.025	2.065	2.185	15	0-14	0-95
26	1.945	2.105	2.185	20	0-14	0-95		74	2.025	2.065	2.185	20	0-14	0-95
27	1.945	2.105	2.185	25	0-14	0-95		75	2.025	2.065	2.185	25	0-14	0-95
28	1.945	2.105	2.185	30	0-14	0-95		76	2.025	2.065	2.185	30	0-14	0-95
29	1.945	2.105	2.225	15	0-14	0-95		77	2.025	2.065	2.225	15	0-14	0-95
30	1.945	2.105	2.225	20	0-14	0-95		78	2.025	2.065	2.225	20	0-14	0-95
31	1.945	2.105	2.225	25	0-14	0-95		79	2.025	2.065	2.225	25	0-14	0-95
32	1.945	2.105	2.225	30	0-14	0-95		80	2.025	2.065	2.225	30	0-14	0-95
33	1.945	2.105	2.265	15	0-14	0-95		81	2.025	2.065	2.265	15	0-14	0-95
34	1.945	2.105	2.265	20	0-14	0-95		82	2.025	2.065	2.265	20	0-14	0-95
35	1.945	2.105	2.265	25	0-14	0-95		83	2.025	2.065	2.265	25	0-14	0-95
36	1.945	2.105	2.265	30	0-14	0-95		84	2.025	2.065	2.265	30	0-14	0-95
37	1.985	2.025	2.185	15	0-14	0-95		85	2.025	2.105	2.185	15	0-14	0-95
38	1.985	2.025	2.185	20	0-14	0-95		86	2.025	2.105	2.185	20	0-14	0-95
39	1.985	2.025	2.185	25	0-14	0-95		87	2.025	2.105	2.185	25	0-14	0-95
40	1.985	2.025	2.185	30	0-14	0-95		88	2.025	2.105	2.185	30	0-14	0-95
41	1.985	2.025	2.225	15	0-14	0-95		89	2.025	2.105	2.225	15	0-14	0-95
42	1.985	2.025	2.225	20	0-14	0-95		90	2.025	2.105	2.225	20	0-14	0-95
43	1.985	2.025	2.225	25	0-14	0-95		91	2.025	2.105	2.225	25	0-14	0-95
44	1.985	2.025	2.225	30	0-14	0-95		92	2.025	2.105	2.225	30	0-14	0-95
45	1.985	2.025	2.265	15	0-14	0-95		93	2.025	2.105	2.265	15	0-14	0-95
46	1.985	2.025	2.265	20	0-14	0-95		94	2.025	2.105	2.265	20	0-14	0-95
47	1.985	2.025	2.265	25	0-14	0-95		95	2.025	2.105	2.265	25	0-14	0-95
48	1.985	2.025	2.265	30	0-14	0-95		96	2.025	2.105	2.265	30	0-14	0-95

El software SDK de Libelium de la tarjeta Waspmote, es capaz de enviar los valores numéricos de voltajes de pH y temperatura leídos en los correspondientes sockets, así como sus correspondientes valores en pH y °C por el puerto COM, como se muestra en Figura 4.20.

```
//USB.print(F("pH value: "));
USB.println(pHVol);
//USB.print(temp);
USB.println(T_volt);
//USB.print(F("degrees  | "));
USB.println(temp);
//USB.print(F(" pH Estimated: "));
USB.println(pHValue);
```

Figura 4.20. Programa de envío de voltajes de temperatura (T) y pH, y sus correspondientes valores en pH y °C a través del puerto COM, (fuente: captura de pantalla, elaboración propia del autor).

Los voltajes ingresados a la tarjeta electrónica Smart Water son controlados por el programa Gen_read_T_V8_final.vi desarrollado en la plataforma LabVIEW, este mismo programa lee sus correspondientes valores en pH y °C (grados Celsius) a través del puerto COM, que previamente fueron enviados por la tarjeta electrónica Waspmote.

En la Figura 4.21 se muestra el panel frontal del programa *Gen_read_T_V8_final.vi*. A través de esta interface se controla la inyección de voltaje a los sockets del sensor de pH y de temperatura.

En la Figura 4.22 se muestra el diagrama de bloque del programa Gen_read_T_V8_final.vi el cual genera voltajes, lee valores de pH y Grados Celsius a través del puerto COM, y almacena dichos valores en archivos de medición basados en texto (.lvm).

En la sección 4.3.2.1 se analiza detalladamente el diagrama de bloques para la adquisición de datos.

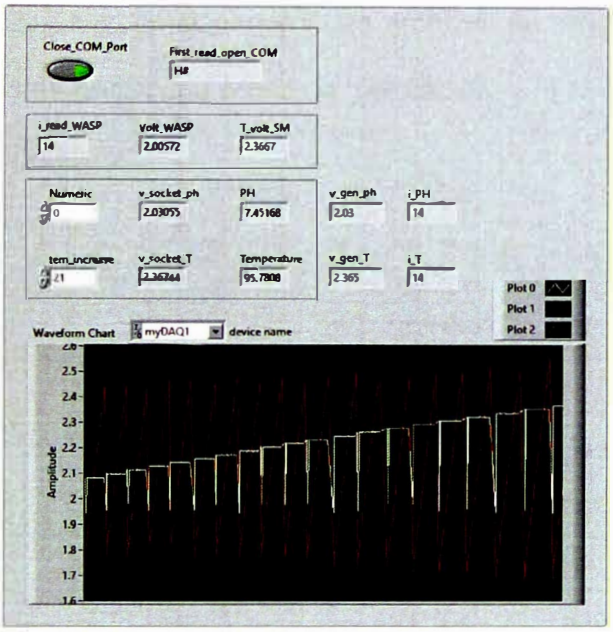


Figura 4.21. Panel frontal del diagrama de bloques para la generación de voltajes, lectura del puerto COM y almacenamiento de los datos leídos – LabView, (fuente: captura de pantalla, elaboración propia del autor).

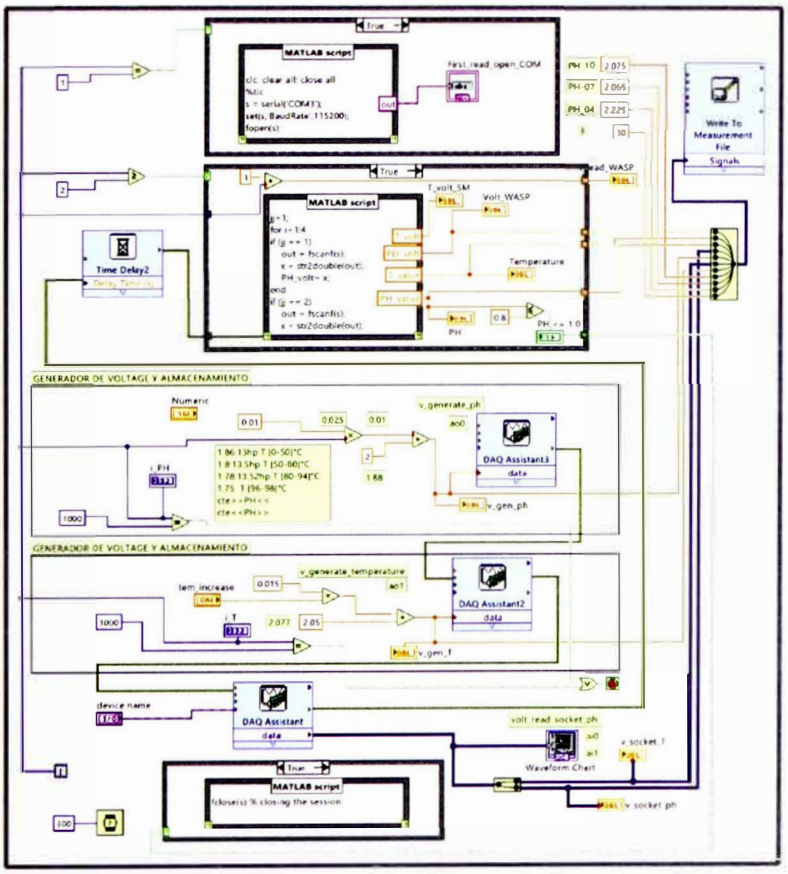


Figura 4.22. Diagrama de bloques general para la adquisición de datos en LabVIEW (fuente: captura de pantalla, elaborado por el autor)

En la Figura 4.23 se muestra parte de los archivos de medición basados en texto (.lvm), conteniendo la data para cada punto de calibración, y la correspondiente variación del voltaje de pH y temperatura.

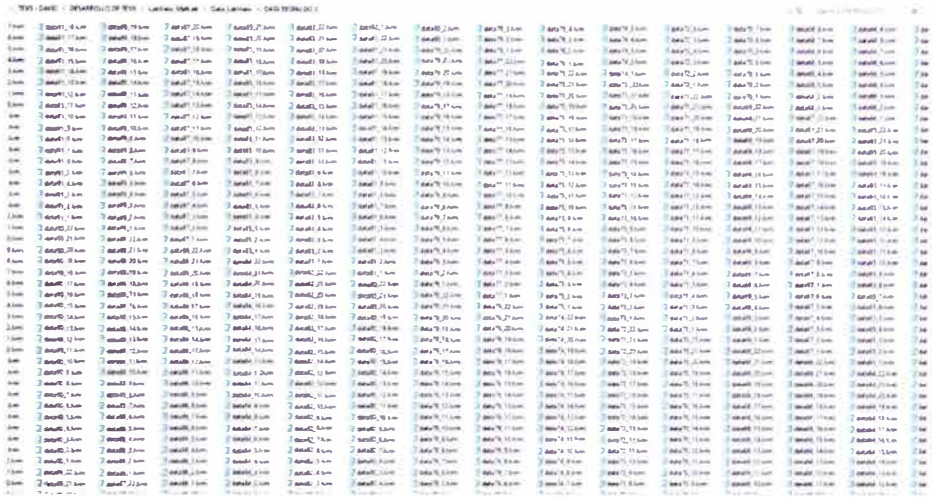


Figura 4.23. Parte de los archivos de medición basado en texto (.lvm) con datos adquiridos para entrenamiento y testeo de la RNA, (fuente: captura de pantalla, elaboración propia del autor).

En la Figura 4.24 se muestra parte del contenido de cada archivo de medición basados en texto (.lvm) de la Figura 4.23 (para cada punto de calibración) capturados por el programa Gen_read_T_V8_final.vi,

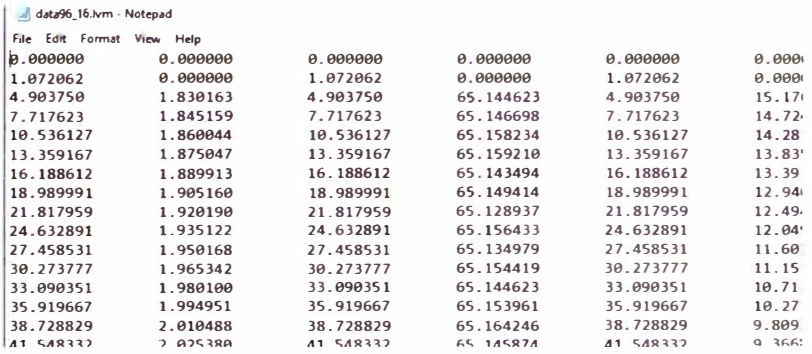


Figura 4.24. Parte del contenido de un archivo de medición basado en texto (.lvm) de recolección de datos, (fuente: captura de pantalla, elaboración propia del autor).

La cantidad de archivos lvm son 2442, cada uno de los cuales tienen en promedio un aproximado de 30 datos cada uno, por lo que la cantidad total de datos son un aproximado de 60000.0 datos. Estos datos aún no están procesados.

4.3.2.3 Preprocesamiento de datos

4.3.2.3.1 Selección de datos en el rango de pH 0.6 y pH 13.9

El propósito principal del preprocesamiento es facilitar el entrenamiento de la RNA.

En la Figura 4.24 podemos observar que las dos primeras filas contienen ceros, estos ceros no son parte de los datos para el entrenamiento de la RNA, por lo que deben ser excluidos del conjunto de datos para el entrenamiento y testeo.

Adicionalmente, en ciertos puntos la cantidad de datos registrados son muy concentrados, por lo que sería innecesario considerarlos todos en el entrenamiento. Considerar una gran cantidad indiscriminada de datos provocaría que la matriz de jacobiana y la matriz Hessiana sean demasiadas grandes lo que necesitaría una gran cantidad de cálculos por parte del computador.

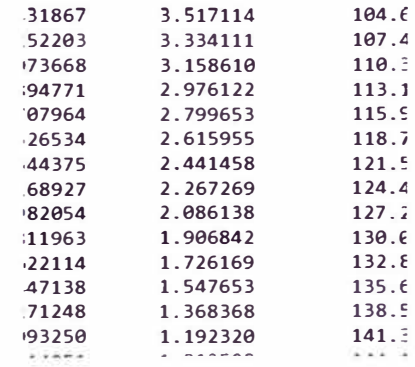


Figura 4.25. Concentración de datos excesivo en cada archivo de medición basado en texto (.lvm), (fuente: captura de pantalla, elaboración propia del autor).

En la Figura 4.25, la columna central corresponde a la variación del pH para cada variación del voltaje en su respectivo socket en la tarjeta Smart Water, vemos que entre un muestreo y otro hay una variación aproximada de 0.2 de pH, por lo que se procede a eliminar ciertos datos, de tal manera que entre una o la siguiente muestra exista una variación de 0.6 de pH.

El script $data\_select.m$ nos ayuda en esa tarea, este script se muestra detalladamente en el

Anexo A.

data_select.m

4.3.2.3.2 Normalización de datos

Para la normalización utilizamos el método descrito en la ecuación 2.61, el cual es implementado en el script *Data_Normalization.m*, el cual se muestra detalladamente en el Anexo B.

Data_Normalization
.m

4.3.2.3.3 Selección de conjuntos de datos

Después de la adquisición de datos, normalmente se divide en tres conjuntos: entrenamiento, validación y testeo. El conjunto de datos de entrenamiento normalmente es el 70% del total de datos, el 15% para validación y el 15% restante para testeo de la RNA. [117]

Posteriormente utilizaremos el algoritmo de entrenamiento por Regularización Bayesiana. Los motivos de esta elección se explicarán más adelante.

El algoritmo de entrenamiento Regularización Bayesiana no requiere validación, por lo que para el entrenamiento se utiliza el 85% del total de la data, y el 15% restante se utiliza para el testeo de la RNA. [118]

La separación de conjunto de datos de entrenamiento y el conjunto de datos de testeo se realiza con el script *Select_Data_Sets.m*, el cual se muestra detalladamente en el Anexo C.

Select_Data_Sets.m

Da ejecución de este script da como resultado:

Cantidad de datos de entrada : size(Dataset_train_input) = [6 28247]

Cantidad de datos de testeo : size(Dataset_testing_input) = [6 4985]

4.3.2.4 Elección de la arquitectura de red neuronal artificial

La arquitectura de la RNA depende en gran medida del problema que estamos abordando.

El problema que abordamos en este trabajo trata de ajuste o *fitting,* que hace referencia una aproximación de una función o regresión.

Las RNA multicapa son más poderosas que las RNA de una sola capa. Por ejemplo, una RNA de dos capas que tiene una primera capa sigmoidal y una segunda capa lineal puede entrenarse para aproximar la mayoría de las funciones arbitrariamente bien. Las RNA de una sola capa no pueden hacer esto [119].

La arquitectura de RNA estándar para problemas de aproximación de funciones o regresiones (fitting) es la arquitectura perceptrón de múltiples capas (multilayer perceptron), con función de activación tangente sigmoidal en las capas ocultas y función de activación lineal en la capa de salida. La función de activación tangente sigmoidal se escoge preferentemente debido a que las entradas son normalizadas. [120]

En cuanto al número de capas, la mayoría de las RNA prácticas tienen solo dos o tres capas. Cuatro o más capas se utilizan raramente, [119].

De los 3 últimos párrafos anteriores podemos asumir una RNA de 2 capas, en caso no resuelva el problema cambiaríamos a una RNA de 3 capas. Adicionalmente sabemos que la lectura de pH depende de 6 factores, por lo que la RNA tendrá 6 entradas, $R = 6$, y una salida correspondiente al pH medido, y por lo tanto una única neurona en la salida.

La cantidad de neuronas en la capa oculta lo se determinará con el número efectivo de parámetros γ del algoritmo de entrenamiento por Regularización Bayesiana.

4.3.3 Entrenamiento de la red neuronal artificial

4.3.3.1 Inicialización de pesos y bias

Antes de iniciar el entrenamiento, debemos inicializar los pesos y los bias de la RNA.

Inicializamos los pesos y los bias de la capa oculta con la ayuda de las ecuaciones (2.63) y (2.64). Método de Nguyen-Widrow.

Para ello hacemos uso del script (función) *Nu_Wid_initial.m*, el cual es detallado en el Anexo E.


Nu_Wid_initial.m

Inicializamos los pesos y bias de la capa de salida aleatoriamente en el rango de -1 y +1:

$$\mathbf{W}^2 = (2 * \text{rand}(1,S1) - 1) * 1; \% \text{ create initial weights.}$$
$$\mathbf{b}^2 = (2 * \text{rand}(1,1) - 1) * 1; \% \text{ create initial bias.}$$

4.3.3.2 Elección y ejecución del algoritmo de entrenamiento

La *regularización bayesiana*, es un algoritmo muy efectivo para entrenar redes neuronales artificiales multicapa para realizar aproximaciones de funciones. Este algoritmo está diseñado para entrenar RNAs para que generalicen bien, sin la necesidad de un conjunto de validación.

Debido a que el conjunto de validación se puede agregar al conjunto de entrenamiento, el rendimiento a menudo es mejor que el obtenido con una detención temprana, [121].

En la sección anterior *selección de la arquitectura* se determinó que la RNA contará con 6 entradas ($R = 6$), varias neuronas en la capa oculta (S^1) y 1 neurona en la capa de salida ($S^2 = 1$). Haciendo referencia a la nomenclatura establecida para la Figura 2.14.

$$R = 6;\ S^1 = S^{1_r},\ S^2 = 1; Q = 28247 \tag{4.1}$$

Para dar inicio al entrenamiento de la RNA ejecutamos el script *training_bayesian_regularization_V7_final.m*, el cual contiene el algoritmo de regularización bayesiana, que a su vez se detalla en el Anexo D.

training_bayesian_r
egularization_V7_fir

Al ejecutar varias veces el escript, con diferente número de neuronas en la capa oculta S^1, se obtuvieron los resultados[28] que se muestran en la **¡Error! No se encuentra el origen de la referencia.**:

Tabla 5. Resultados de entrenamiento de la RNA para diferente número de neuronas en la capa oculta. (fuente: elaboración propia del autor).

S^1	ssE	ssX	gamma γ	error máx.	error med	S^1 (γ)
15	0.5784	16915	113.53	0.3682	0.0226	14.07
20	0.2667	15592	153.94	0.2151	0.0154	19.12
25	0.2003	96166	186.94	0.1574	0.0134	23.24
30	0.1803	29047	224.90	0.1612	0.0128	27.99
35	0.1758	39541	264.69	0.1289	0.0125	32.96
40	0.1855	22701	302.44	0.1750	0.0127	37.68

donde:

S^1 : número de neuronas en la primera capa.

ssE : suma cuadrática de errores.

ssX : suma cuadrática de parámetros (pesos y bias de todas las capas).

γ : número efectivo de parámetros que es usado por la RNA.

error máx.: error máximo (durante el testeo de la RNA, mas no de entrenamiento).

[28] Una tabla más completa de resultados se muestra en el Anexo I. **¡Error! No se encuentra el origen de la referencia.**, en ella también se detalla su contenido.

error med.: Error medio (durante el testeo de la RNA, mas no de entrenamiento).

$S^1(\gamma)$: número de neuronas en la capa oculta recomendado por el algoritmo.

De los resultados, tenemos que para $S^1 = 35$ la suma cuadrática de errores (ssE), el error máximo (error máx.) y el error medio (error med.) son menores al del resto de pruebas. Sin embargo, las neuronas en la capa oculta ($S^1 = 35$) no es la menor, esto implica que se necesitará un mayor poder de computo para realizar los cálculos para $S^1 = 35$ que para valores de S^1 menores a 35. Si embargo actualmente los microcontroladores tienen gran capacidad de cálculo con bajo consumo de energía, por lo que no sería un problema operar una matriz de 35x6 elementos.

Estos resultados lo analizaremos detalladamente en la sección de post entrenamiento.

En la sección 4.3.3.2.1 se presenta el diagrama de flujo del algoritmo del script *training_bayesian_regularization_V7_final.m,* que nos permite entrenar la RNA.

En la sección 4.3.3.2.2 se detalla minuciosamente el procedimiento de cálculo matemático del algoritmo de regularización bayesiana que se utiliza en el script *training_bayesian_regularization_V7_final.m*, el cual se utiliza para el entrenamiento.

4.3.3.2.1 Diagrama de flujo de del algoritmo de entrenamiento

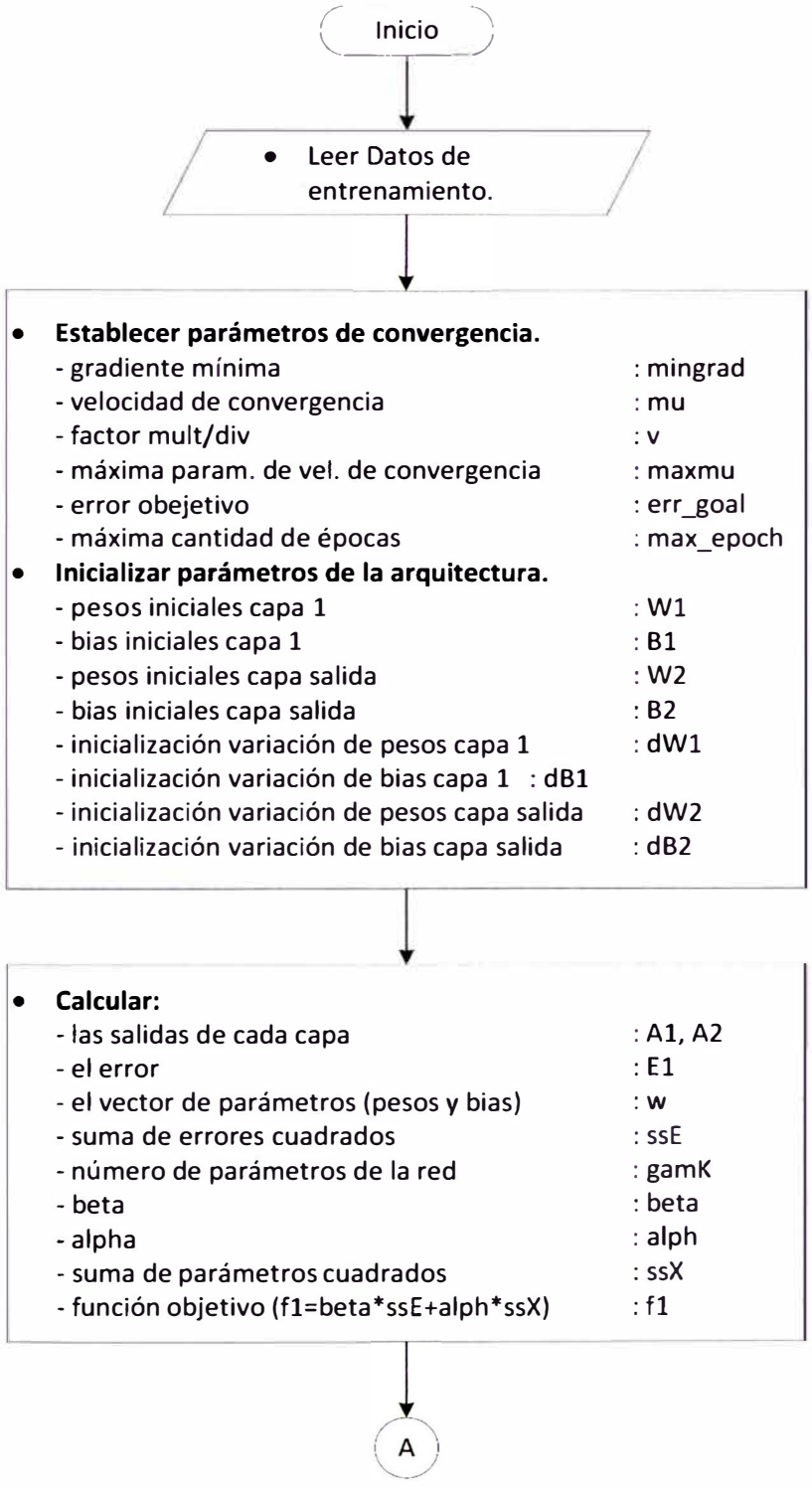


Figura 4.26. Diagrama de flujo – inicialización (fuente: elaboración propia del autor)

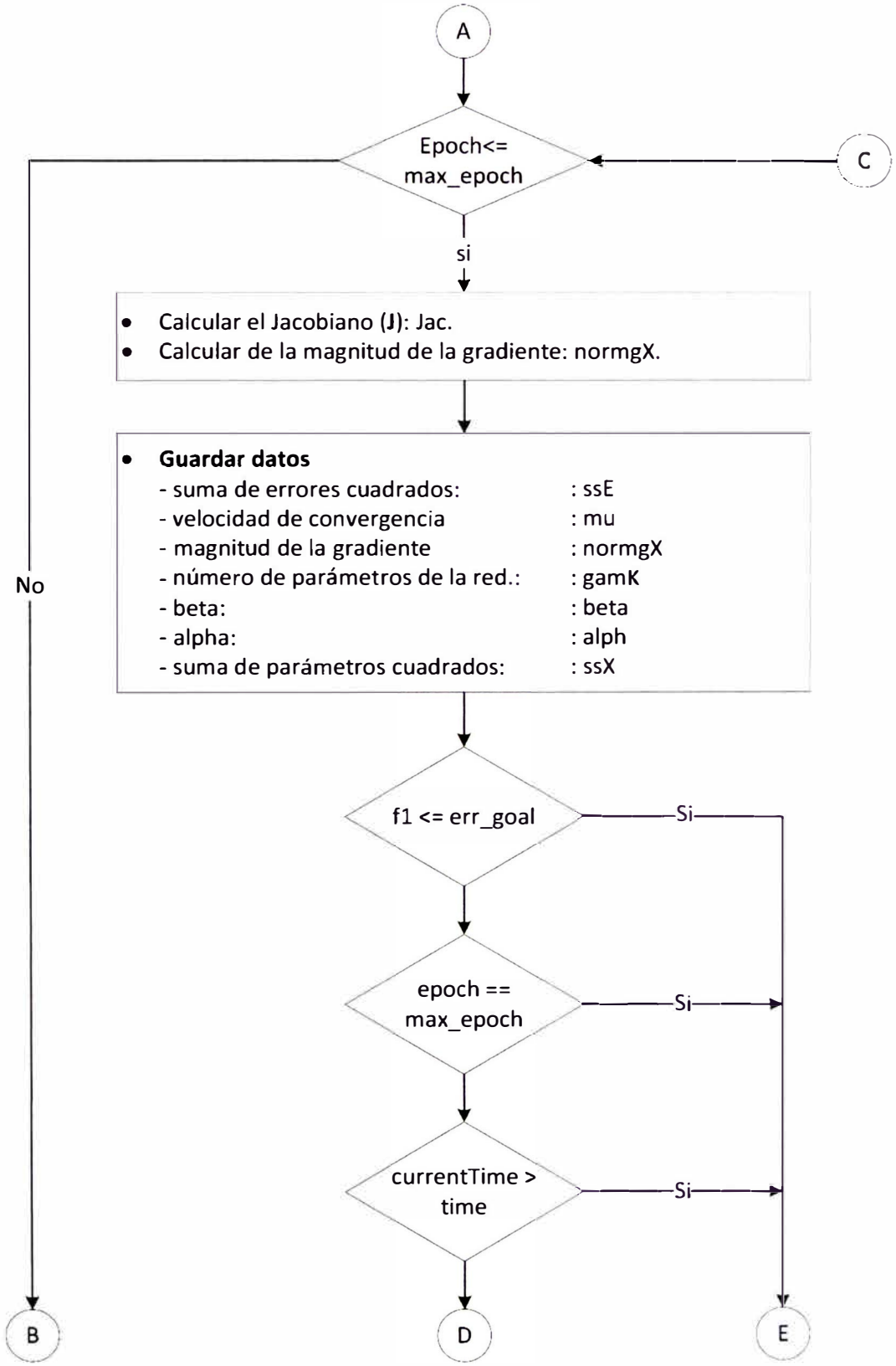


Figura 4.27. Diagrama de flujo – cálculo del Jacobiano, guardado de datos (fuente: elaboración propia del autor).

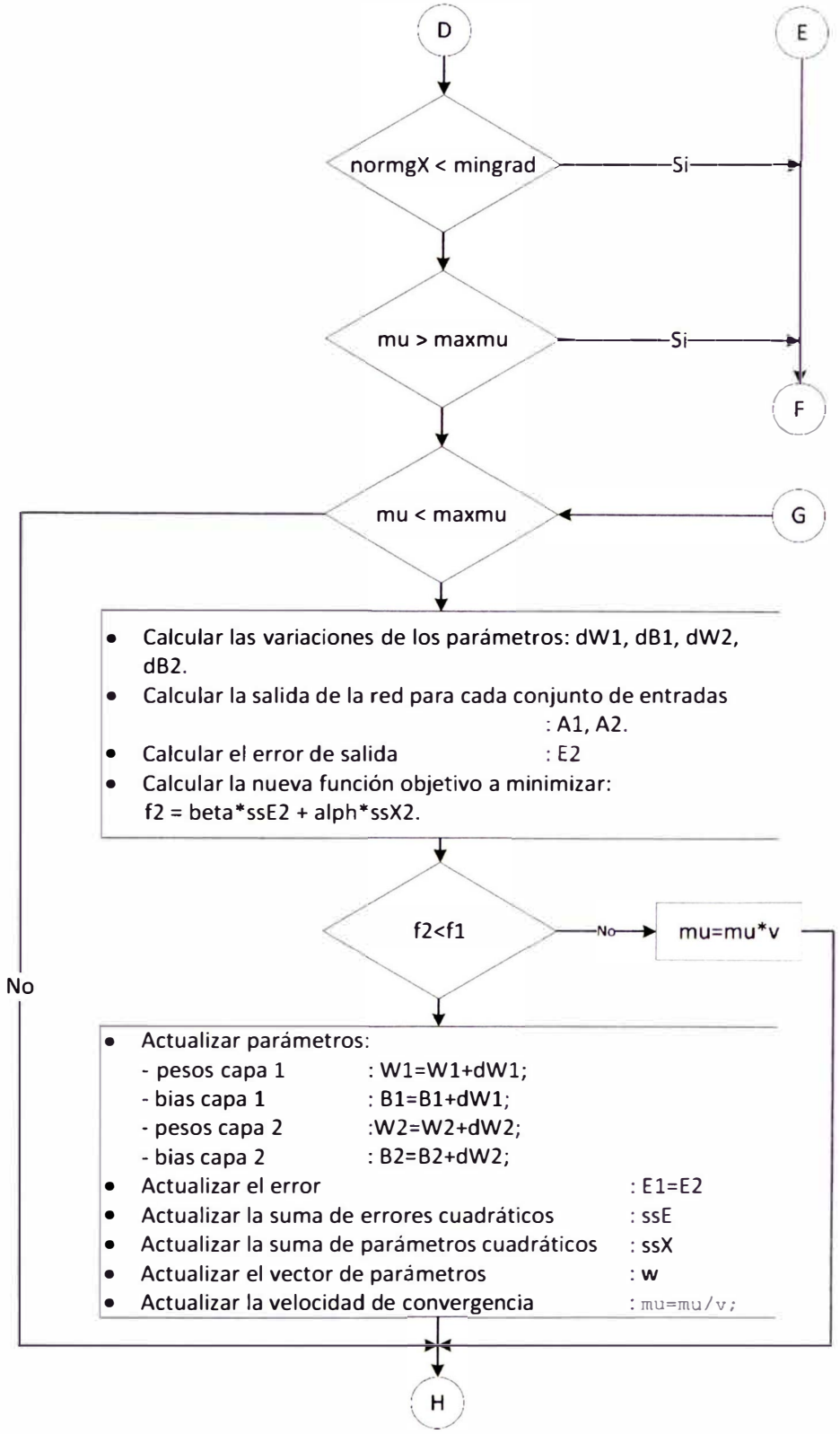


Figura 4.28. Diagrama de flujo – actualización de parámetros (fuente: elaboración propia del autor).

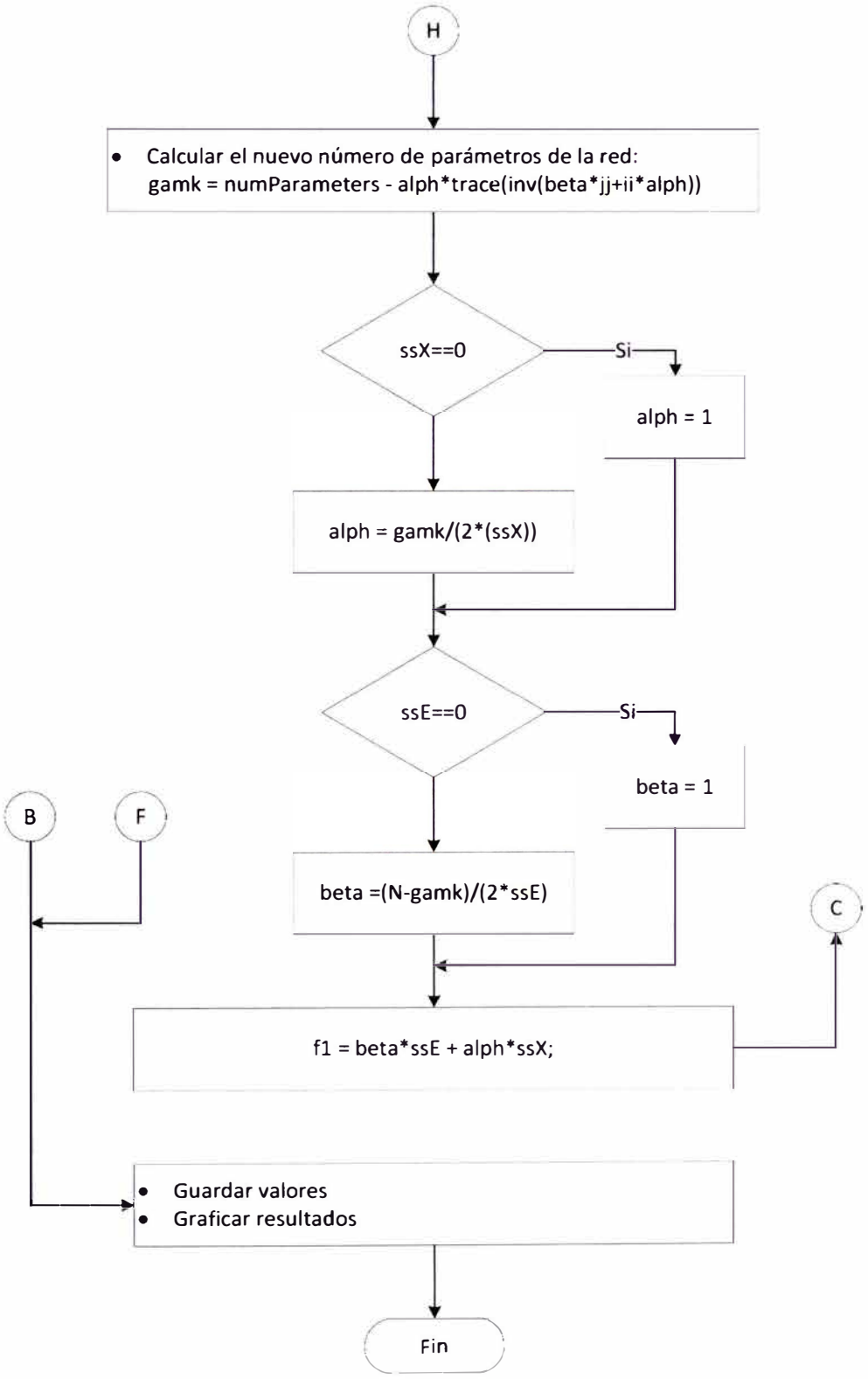


Figura 4.29. Diagrama de flujo – número efectivo de parámetros (fuente: elaboración propia del autor).

4.3.3.2.2 Algoritmo detallado de Regularización Bayesiana

El siguiente algoritmo está diseñando para una RNA de **dos capas**, con $\mathbf{R}$ entradas, $\mathbf{S^1}$ neuronas en la capa oculta, una neurona ($\mathbf{S^2 = 1}$) en la capa de salida, con función de activación tangente sigmoidal (**tansig**) en la capa oculta y función de activación lineal (**pureline**) en capa de salida. La ejecución del algoritmo es en modo batch.

La optimización bayesiana de los parámetros de regularización requiere el cálculo de la matriz de Hesse de $F(\mathbf{x})$ en el punto mínimo $\mathbf{x}^{MP}$. Proponemos utilizar la aproximación de Gauss-Newton mediante la aproximación de la matriz de Hesse, si utilizamos el algoritmo de optimización Levenberg-Marquardt para localizar el punto mínimo (consulte la ecuación (2.51)).

Estos son los pasos necesarios para la optimización bayesiana de los parámetros de regularización, con la aproximación de Gauss-Newton a la matriz de Hesse [122]

1. Inicializar α, β y los pesos. Los pesos se inicializan aleatoriamente, y luego se calculan E_D y E_W. Establezca $\gamma = n$, y calcule α y β usando la ecuación (2.114).

   Entradas y salidas (objetivos):

   Las matrices de entrada y objetivos (target) son $\mathbf{P}$ y $\mathbf{T}$ respectivamente.

$$\mathbf{p} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,Q} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ p_{R,1} & p_{R,2} & \cdots & p_{R,Q} \end{bmatrix}_{\mathrm{R\times Q}}, \tag{4.2}$$

$$\mathbf{T}_{general} = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,Q} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ t_{S^2,1} & t_{S^2,2} & \cdots & t_{S^2,Q} \end{bmatrix}_{\mathrm{S^2\times Q}}. \tag{4.3}$$

   Sabemos que la RNA tiene solo una salida, por lo tanto, $S^2 = 1$

$$\mathbf{T} = [t_{1,1} \quad t_{1,2} \quad \cdots \quad t_{1,Q}]_{1\times \mathrm{Q}}. \tag{4.4}$$

Cantidad de neuronas en la capa S^1:

Se elige con criterio la cantidad de neuronas S^1 para la capa oculta. Se puede elegir, por ejemplo, el doble de la cantidad de entradas de la RNA, posteriormente lo iremos ajustando de acuerdo al valor de resultante de γ. γ es el número efectivo de parámetros que es usado por la RNA.

Inicialización de los pesos y bias de la capa oculta:

Se inicializa los pesos y los bias de la capa oculta con la ayuda de las ecuaciones (2.63) y (2.64), con lo que se obtiene:

$$\mathbf{W}^1 = \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & \cdots & w_{1,R}^1 \\ w_{2,1}^1 & w_{2,2}^1 & \cdots & w_{2,R}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{S^1,1}^1 & w_{S^1,2}^1 & \cdots & w_{S^1,R}^1 \end{bmatrix}_{\mathrm{S^1xR}} \tag{4.5}$$

$$\mathbf{b}^1 = \begin{bmatrix} b_1^1 & b_1^1 & \cdots & b_1^1 \\ b_2^1 & b_2^1 & \cdots & b_2^1 \\ \vdots & \vdots & \ddots & \vdots \\ b_{S^1}^1 & b_{S^1}^1 & \cdots & b_{S^1}^1 \end{bmatrix}_{\mathrm{S^1xQ}} \tag{4.6}$$

Inicialización de los pesos y bias de la capa de salida:

Se inicializa los pesos y bias de la capa de salida aleatoriamente en el rango de -0.5 y 0.5.

$$\mathbf{W}^2 = \begin{bmatrix} w_{1,1}^2 & w_{1,2}^2 & \cdots & w_{1,S^1}^2 \end{bmatrix}_{\mathrm{1xS^1}} \tag{4.7}$$

$$\mathbf{b}^2 = \begin{bmatrix} b_1^2 & b_1^2 & \cdots & b_1^2 \end{bmatrix}_{\mathrm{1xQ}} \tag{4.8}$$

Errores:

Cálculos previos:

$\mathbf{a}^1 = tansig(\mathbf{n}^1)$

$$\mathbf{n}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 \tag{4.9}$$

$$\mathbf{n}^1 = \begin{bmatrix} {}^{1}_{1,1} & w^1_{1,2} & \cdots & w^1_{1,R} \\ w^1_{2,1} & w^1_{2,2} & \cdots & w^1_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w^1_{S^1,1} & w^1_{S^1,2} & \cdots & w^1_{S^1,R} \end{bmatrix}_{S^1 x R}$$

$$\times \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,Q} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ p_{R,1} & p_{R,2} & \cdots & p_{R,Q} \end{bmatrix}_{R \times Q}$$

$$+ \begin{bmatrix} b^1_1 & b^1_1 & \cdots & b^1_1 \\ b^1_2 & b^1_2 & \cdots & b^1_2 \\ \vdots & \vdots & \ddots & \vdots \\ b^1_{S^1} & b^1_{S^1} & \cdots & b^1_{S^1} \end{bmatrix}_{S^1 x Q} \tag{4.10}$$

$$\mathbf{a}^1 = tansig(\mathbf{n}^1)$$

$$= tansig\left(\begin{bmatrix} w^1_{1,1}p_{1,1} + w^1_{1,2}p_{2,1} + w^1_{1,3}p_{3,1} + \cdots + w^1_{1,R}p_{R,1} + b^1_1 \\ w^1_{2,1}p_{1,1} + w^1_{2,2}p_{2,1} + w^1_{2,3}p_{3,1} + \cdots + w^1_{2,R}p_{R,1} + b^1_2 \\ \vdots \\ w^1_{S^1,1}p_{1,1} + w^1_{S^1,2}p_{2,1} + w^1_{S^1,3}p_{3,1} + \cdots + w^1_{S^1,R}p_{R,1} + b^1_{S^1} \end{bmatrix}\right.$$

$$\begin{matrix} w^1_{1,1}p_{1,2} + w^1_{1,2}p_{2,2} + w^1_{1,3}p_{3,2} + \cdots + w^1_{1,R}p_{R,2} + b^1_1 & \cdots \\ w^1_{2,1}p_{1,2} + w^1_{2,2}p_{2,2} + w^1_{2,3}p_{3,2} + \cdots + w^1_{2,R}p_{R,2} + b^1_2 & \cdots \\ \vdots & \ddots \\ w^1_{S^1,1}p_{1,2} + w^1_{S^1,2}p_{2,2} + w^1_{S^1,3}p_{3,2} + \cdots + w^1_{S^1,R}p_{R,2} + b^1_{S^1} & \cdots \end{matrix}$$

$$\left.\begin{matrix} \cdots & w^1_{1,1}p_{1,Q} + w^1_{1,2}p_{2,Q} + w^1_{1,3}p_{3,Q} + \cdots + w^1_{1,R}p_{R,Q} + b^1_1 \\ \cdots & w^1_{2,1}p_{1,Q} + w^1_{2,2}p_{2,Q} + w^1_{2,2}p_{2,Q} + \cdots + w^1_{2,R}p_{R,Q} + b^1_2 \\ \ddots & \vdots \\ \cdots & w^1_{S^1,1}p_{1,Q} + w^1_{S^1,2}p_{2,Q} + w^1_{S^1,3}p_{3,Q} \ldots + w^1_{S^1,R}p_{R,Q} + b^1_{S^1} \end{matrix}\right]_{S^1 x Q}\Bigg) \tag{4.11}$$

$$\mathbf{a}^1 = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & \cdots & a^1_{1,Q} \\ a^1_{2,1} & a^1_{2,2} & \cdots & a^1_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ a^1_{S^1,1} & a^1_{S^1,2} & \cdots & a^1_{S^1,Q} \end{bmatrix}_{S^1 x Q} \tag{4.12}$$

$\mathbf{a}^2 = \text{purelin}(\mathbf{n}^2)$

$$\mathbf{n}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^1 \tag{4.13}$$

$$\mathbf{n}^2 = \begin{bmatrix} w^2_{1,1} & w^2_{1,2} & \cdots & w^2_{1,S^1} \end{bmatrix}_{1xS^1} \times \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & \cdots & a^1_{1,Q} \\ a^1_{2,1} & a^1_{2,2} & \cdots & a^1_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ a^1_{S^1,1} & a^1_{S^1,2} & \cdots & a^1_{S^1,Q} \end{bmatrix}_{S^1xQ}$$

$$+ \begin{bmatrix} b^2_1 & b^2_1 & \cdots & b^2_1 \end{bmatrix}_{1xQ} \tag{4.14}$$

$$\mathbf{n}^2 = \begin{bmatrix} w^2_{1,1}a^1_{1,1} + w^2_{1,2}a^1_{2,1} + w^2_{1,3}a^1_{3,1} \ldots + w^2_{1,S^1}a^1_{S^1,1} + b^2_1 \\ w^2_{1,1}a^1_{1,2} + w^2_{1,2}a^1_{2,2} + w^2_{1,3}a^1_{3,2} \ldots + w^2_{1,S^1}a^1_{S^1,2} + b^2_1 \quad \cdots \\ \cdots \quad w^2_{1,1}a^1_{1,Q} + w^2_{1,2}a^1_{2,Q} + w^2_{1,3}a^1_{3,Q} \ldots + w^2_{1,S^1}a^1_{S^1,Q} + b^2_1 \end{bmatrix}_{1xQ} \tag{4.15}$$

$$\mathbf{a}^2 = \text{purelin}(\mathbf{n}^2)$$

$$\mathbf{a}^2 = \begin{bmatrix} w^2_{1,1}a^1_{1,1} + w^2_{1,2}a^1_{2,1} + w^2_{1,3}a^1_{3,1} \ldots + w^2_{1,S^1}a^1_{S^1,1} + b^2_1 \\ w^2_{1,1}a^1_{1,2} + w^2_{1,2}a^1_{2,2} + w^2_{1,3}a^1_{3,2} \ldots + w^2_{1,S^1}a^1_{S^1,2} + b^2_1 \quad \cdots \\ \cdots \quad w^2_{1,1}a^1_{1,Q} + w^2_{1,2}a^1_{2,Q} + w^2_{1,3}a^1_{3,Q} \ldots + w^2_{1,S^1}a^1_{S^1,Q} + b^2_1 \end{bmatrix}_{1xQ} \tag{4.16}$$

$$\mathbf{a}^2 = \begin{bmatrix} a^2_{1,1} & a^2_{1,2} & \cdots & a^2_{1,Q} \end{bmatrix}_{1xQ} \tag{4.17}$$

Errores: $\mathbf{e}^T = \mathbf{T} - \mathbf{a}^2$,

$$\mathbf{e}^T = \mathbf{T} - \mathbf{a}^2 \tag{4.18}$$

$$\mathbf{e}^T = [t_{1,1} \quad t_{1,2} \quad \cdots \quad t_{1,Q}]_{1\times Q} - [a^2_{1,1} \quad a^2_{1,2} \quad \cdots \quad a^2_{1,Q}]_{1xQ}$$

$$\mathbf{e}^T = [t_{1,1} - a^2_{1,1} \quad t_{1,2} - a^2_{1,2} \quad \cdots \quad t_{1,Q} - a^2_{1,Q}]_{1\times Q} \tag{4.19}$$

$$\mathbf{e}^T = [e_{1,1} \quad e_{1,2} \quad \cdots \quad e_{1,Q}]_{1\times Q} \tag{4.20}$$

$$\mathbf{v}^T = \mathbf{e}^T \tag{4.21}$$

Cantidad de errores *N*

La cantidad de errores lo obtenemos de la ecuación (2.57): $N = Q \times S^M$,

$$N = Q \times S^M = Q \times 1 \tag{4.22}$$

$$N = Q \tag{4.23}$$

N : candad de errores de entrenamiento de la RNA.

S^M : cantidad de neuronas en la capa de salida.

Q : cantidad de pares de vectores de entrada/salida.

Vector de parámetros

Este vector contiene todos los parámetros de la RNA (pesos y bias)

cálculos previos:

$\mathbf{x}_1$

$$= [w^1_{1,1} \quad \cdots \quad w^1_{S^1,1} \quad w^1_{1,2} \quad \cdots \quad w^1_{S^1,2} \quad \cdots \quad w^1_{1,R} \quad \cdots \quad w^1_{S^1,R}]_{1x(S^1xR)} \tag{4.24}$$

$$\mathbf{x}_2 = [b^1_1 \quad b^1_2 \quad \cdots \quad b^1_{S^1}]_{1xS1} \tag{4.25}$$

$\mathbf{x}_3$

$$= [w^2_{1,1} \quad \cdots \quad w^2_{S^2,1} \quad w^2_{1,2} \quad \cdots \quad w^2_{S^2,2} \quad \cdots \quad w^2_{1,S^1} \quad \cdots \quad w^2_{S^2,S^1}]_{1x(S^2xS^1)} \tag{4.26}$$

$$\mathbf{x}_4 = [b^2_1 \quad b^2_2 \quad \cdots \quad b^2_{S^2}]_{1xS^2} \tag{4.27}$$

$$\mathbf{x}^{\mathrm{T}} = \left[[\mathbf{x}_1]_{1x(S^1xR)} \quad [\mathbf{x}_2]_{1xS^1} \quad [\mathbf{x}_3]_{1x(1xS^1)} \quad [\mathbf{x}_4]_{1x1}\right]_{1x\left((S^1xR)+S^1+(1xS^1)+1\right)} \tag{4.28}$$

$$\mathbf{x}^T = [x_{1,1} \quad x_{1,2} \quad \cdots \quad x]_{1\times\left((S^1xR)+S^1+(1xS^1)+1\right)} \tag{4.29}$$

Vector de parámetros:

$$\mathbf{x}^T = [w^1_{1,1} \quad w^1_{2,1} \quad \cdots \quad w^1_{S^1,1} \quad w^1_{1,2} \quad w^1_{2,2} \quad \cdots \quad w^1_{S^1,2} \quad w^1_{1,R} \quad \cdots$$
$$\cdots \quad w^1_{S^1,R} \quad b^1_1 \quad \cdots \quad b^1_{S^1} \quad w^2_{1,1} \quad w^2_{1,2} \quad \cdots \quad w^2_{1,S^1} \quad b^2_1]_{1x\left((S^1xR)+S^1+(1xS^1)+1\right)} \tag{4.30}$$

Inicialización de γ:

Se inicializa γ con la ayuda de la ecuación (2.58):

$$\gamma = n = S^1(R+1) + S^2(S^1+1) = S^1 \times (6+1) + 1 \times (S^1+1)$$

$$\gamma = 8 \times S^1 + 1 \tag{4.31}$$

Cálculo de las funciones cuadráticas $E_D(\mathbf{x})$ y $E_W(\mathbf{x})$:

Con la ayuda de las ecuaciones (2.55), (2.56), (2.67) y (2.68) se calcula $E_D(\mathbf{x})$ y $E_W(\mathbf{x})$.

$$E_D = \sum_{q=1}^{Q} \left(\mathbf{t}_q - \mathbf{a}_q^2\right)^T (t_q - \mathbf{a}_q^2) = \mathbf{v}^T\mathbf{v} \tag{4.32}$$

$$E_w = \sum_{i=1}^{n} x_i^2 = \mathbf{x}^T\mathbf{x} \tag{4.33}$$

Inicializando α y β:

Se inicializa α y β, de acuerdo con las ecuaciones:

$$\alpha = \frac{\gamma}{2E_W(\mathbf{x})} \tag{4.34}$$

$$\beta = \frac{N - \gamma}{2 \quad _D(\mathbf{x})} \tag{4.35}$$

2. Se toma algunos pasos del algoritmo de Levenberg-Marquardt para minimizar la función objetivo $F(x)$.

Minimización de la función objetivo

La función objetivo se calculada en cada época de entrenamiento, con los α, β, E_D y E_W, calculados previamente, y se copara con $F(x)$ el calculado previamente, si el nuevo cálculo es menor al anterior se actualiza los parámetros (pesos y bias).

$$F(x) = \beta E_D + \alpha E_W \tag{4.36}$$

Variación de parámetros

Determinación de la expresión matemática de la variación de parámetros en cada iteración a partir de ecuaciones establecidas en el marco teórico.

Se parte del método de newton para optimizar el índice de desempeño de una función cuadrática, según la ecuación (2.38)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \tag{4.37}$$

donde $\mathbf{A}_k \equiv \nabla^2\big(F(\mathbf{x})\big)\big|_{\mathbf{x}=\mathbf{x}_k}$ y $\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$

$\mathbf{g}_k$: gradiente de la función cuadrática.

$\mathbf{A}_k$: Hessian de la función cuadrática.

De la ecuación (2.68), la función que incluye el término de regularización es:

$$F(\mathbf{x}) = \beta E_D + \alpha E_w = \beta \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (t_q - \mathbf{a}_q) + \alpha \sum_{i=1}^{n} x_i^2 \tag{4.38}$$

$$F(\mathbf{x}) = \beta E_D + \alpha E_w = \beta \sum_{i=1}^{Q} v_i + \alpha \sum_{i=1}^{n} x_i^2 \tag{4.39}$$

donde $\alpha \sum_{i=1}^{n} x_i^2$ es el término de regularización.

Cálculo de la gradiente $\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$

Cambiando el vector $\mathbf{x}$ de la ecuación del método de Newton por el vector $\mathbf{r}$, a fin de no confundirlo con el vector $\mathbf{x}$ del término de regularización ($\alpha \sum_{i=1}^{n} x_i^2$).

Cálculo de la gradiente:

$$[\nabla F(\mathbf{r})]_j = \frac{\partial F(\mathbf{r})}{\partial r_j} = 2\beta \sum_{i=1}^{N} v_i(\mathbf{r}) \frac{\partial v_i(\mathbf{r})}{\partial r_j} + 2\alpha \sum_{i=1}^{N} x_i(\mathbf{r}) \frac{\partial x_i(\mathbf{r})}{\partial r_j} \tag{4.40}$$

$$[\nabla F(\mathbf{r})]_j = \frac{\partial F(\mathbf{r})}{\partial r_j} = 2\beta \sum_{i=1}^{N} v_i(\mathbf{r}) \frac{\partial v_i(\mathbf{r})}{\partial r_j} + 2\alpha \sum_{i=1}^{N} x_i(\mathbf{r})\,(1) \tag{4.41}$$

La gradiente puede por lo tanto ser escrita en forma matricial:

$$\nabla F(\mathbf{r}) = 2\beta \mathbf{J}^T(\mathbf{r}) \mathbf{v}(\mathbf{r}) + 2\alpha \mathbf{x}(\mathbf{r}) \tag{4.42}$$

Cálculo de la matriz de Hesse. El elemento k, j de la matriz de Hesse sería:

$$[\nabla^2 F(\mathbf{r})]_{k,j} = \frac{\partial^2 F(\mathbf{r})}{\partial r_k \partial r_j}$$

$$= 2\beta \sum_{i=1}^{N} \left\{ \frac{\partial v_i(\mathbf{r})}{\partial r_k} \frac{\partial v_i(\mathbf{r})}{\partial r_j} + v_i(\mathbf{r}) \frac{\partial^2 v_i(\mathbf{r})}{\partial r_k \partial r_j} \right\}$$

$$+ 2\alpha \sum_{i=1}^{N} \left\{ \frac{\partial x_i(\mathbf{r})}{\partial r_k}(1) + x_i(\mathbf{r}) \frac{\partial^2 1}{\partial r_k \partial r_j} \right\} \qquad (4.43)$$

$$= 2\beta \sum_{i=1}^{N} \left\{ \frac{\partial v_i(\mathbf{r})}{\partial r_k} \frac{\partial v_i(\mathbf{r})}{\partial r_j} + v_i(\mathbf{r}) \frac{\partial^2 v_i(\mathbf{r})}{\partial r_k \partial r_j} \right\}$$

$$+ 2\alpha \sum_{i=1}^{N} \{(1)(1) + x_i(\mathbf{r})(0)\}$$

$$[\nabla^2 F(\mathbf{r})]_{k,j} = \frac{\partial^2 F(\mathbf{r})}{\partial r_k \partial r_j}$$

$$= 2\beta \sum_{i=1}^{N} \left\{ \frac{\partial v_i(\mathbf{r})}{\partial r_k} \frac{\partial v_i(\mathbf{r})}{\partial r_j} + v_i(\mathbf{r}) \frac{\partial^2 v_i(\mathbf{r})}{\partial r_k \partial r_j} \right\} \qquad (4.44)$$

$$+ 2\alpha \sum_{i=1}^{N} \{(1)\}$$

La matriz de Hesse puede por lo tanto ser escrita en forma matricial:

$$\nabla^2 F(\mathbf{r}) = 2\beta \mathbf{J}^T(\mathbf{r}) \mathbf{J}(\mathbf{r}) + 2\beta \mathbf{S}(\mathbf{r}) + 2\alpha \mathrm{I} \qquad (4.45)$$

Si se asume que $2\beta \mathbf{S}(\mathbf{r})$ es pequeño, como se sugiere en la ecuación (2.46), la matriz de Hesse puede expresarse en forma matricial:

$$\nabla^2 F(\mathbf{r}) = 2\beta \mathbf{J}^T(\mathbf{r}) \mathbf{J}(\mathbf{r}) + 2\alpha \mathrm{I} \qquad (4.46)$$

Se vuelve a cambiar el vector **r** por el vector **x**

$$\nabla^2 F(\mathbf{x}) = 2\beta \mathbf{J}^T(\mathbf{x}) \mathbf{J}(\mathbf{x}) + 2\alpha \mathrm{I} \qquad (4.47)$$

Remplazando la ecuación (4.42) y (4.47) en la ecuación (4.37):

$$\mathbf{x}_{k+1} = \quad_k - \mathbf{A}_k^{-1}\mathbf{g}_k \tag{4.48}$$

donde $\mathbf{A}_k \equiv \nabla^2\big(F(\mathbf{x})\big)\big|_{\mathbf{x}=\mathbf{x}_k}$ y $\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$

$$\mathbf{A}_k \equiv \nabla^2\big(F(\mathbf{x})\big)\big|_{\mathbf{x}=\mathbf{x}_k} = 2\beta\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\alpha\mathrm{I} \tag{4.49}$$

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} = 2\beta\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) + 2\alpha\mathbf{x}(\mathbf{x}) \tag{4.50}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \tag{4.51}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (2\beta\,\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\alpha\mathrm{I})^{-1}\Big(2\beta\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) + 2\alpha\mathbf{x}(\mathbf{x})\Big)$$

$$\mathbf{x}_{k+1} - \mathbf{x}_k = -(\beta\,\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + \alpha\mathrm{I})^{-1}\Big(\beta\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) + \alpha\mathbf{x}(\mathbf{x})\Big)$$

$$\nabla\mathbf{x}_k = -(\beta\,\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + \alpha\mathrm{I})^{-1}\Big(\beta\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) + \alpha\mathbf{x}(\mathbf{x})\Big) \tag{4.52}$$

la matriz de Hesse $\Big(\mathbf{A}_k \equiv \nabla^2\big(F(\mathbf{x})\big)\big|_{\mathbf{x}=\mathbf{x}_k} = \mathbf{H} = \beta\,\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + \alpha\mathrm{I}\Big)$ puede no ser invertible, por lo que se agrega un parámetro según la ecuación (2.49):

$$\mathbf{G} = \mathbf{H} + \mu\mathbf{I} \tag{4.53}$$

remplazamos $\mathbf{G}$ en la en la ecuación (4.52), con lo que obtenemos la variación de los parámetros (pesos y bias):

$$\Delta\mathbf{x}_k = -(\beta\,\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \alpha_k\mathrm{I} + \mu_k\mathbf{I})^{-1}\Big(\beta\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) + \alpha\mathbf{x}(\mathbf{x}_k)\Big) \tag{4.54}$$

donde $\mathbf{x}$ en $\mathbf{x}(\mathbf{x}_k)$ representa el vector de parámetros, $\mathbf{x}_k$ representa el vector de la gran variación de parámetros.

Numero efectivo de parámetros

Se calcula del número efectivo de parámetros $\gamma = n - 2\alpha\mathrm{tr}(\mathbf{H})^{-1}$, haciendo uso de la aproximación de Gauss-Newton al Hessiano del algoritmo de

entrenamiento de Levenberg-Marquardt, $\mathbf{H} = \nabla^2 F(\mathbf{x}) \approx 2\beta \mathbf{J}^T \mathbf{J} + 2\alpha \mathrm{I}_n$, donde $\mathbf{J}$ es la matriz jacobiana de los errores del conjunto de entrenamiento.

Cálculo de la matriz Jacobiana J

Cálculo de la matriz Jacobiana de los errores. De la ecuación (2.42) se tiene:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1(\mathbf{x})}{\partial x_1} & \frac{\partial v_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial v_2(\mathbf{x})}{\partial x_1} & \frac{\partial v_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_N(\mathbf{x})}{\partial x_1} & \frac{\partial v_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{4.55}$$

donde $\mathbf{v}^T = [v_1 \ v_2 \ \ldots\ldots v_N]$ representa el vector de errores del conjunto de entrenamiento, y $\mathbf{x}^T = [x_1 \ x_2 \ \ldots\ldots v_x]$ representa el vector de parámetros de la RNA.

De las ecuaciones (2.55) y (2.56),

$$\begin{aligned} \mathbf{v}^T &= [\, v_1 \quad v_2 \quad \cdots \quad v_N] \\ &= [\, e_{1,1} \quad e_{2,1} \quad \cdots \quad e_{S^M,1} \quad e_{1,2} \quad \cdots \quad e_{S^M,Q}] \end{aligned} \tag{4.56}$$

$$\begin{aligned} \mathbf{x}^T &= [\, x_1 \quad x_2 \quad \cdots \quad x_n] \\ &= \left[\, w^1_{1,1} \quad w^1_{2,1} \quad \cdots \quad w^1_{S^1,R} \quad b^1_1 \quad \cdots \quad b^1_{S^1} \quad w^2_{1,1} \quad \cdots \quad b^M_{S^M}\right] \end{aligned} \tag{4.57}$$

Recordar que los cálculos se están realizando son para una RNA de $\mathbf{R}$ *entradas,* $\mathbf{S^1}$ *neuronas en la capa oculta, y* $\mathbf{S^2} = 1$ *neuronas en la capa de salida, con función de activación tansig en la capa oculta, y función de activación pureline en la capa de salida.*

De las ecuaciones (4.55), (4.56) y (4.57) se obtiene la matriz Jacobiana,

$$\mathbf{J(x)} = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{2,1}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,1}} & \frac{\partial e_{1,1}}{w^1_{1,2}} & \frac{\partial e_{1,1}}{w^1_{2,2}} & \cdots & \frac{\partial e_{1,1}}{w^1_{S^1,2}} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{2,1}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,1}} & \frac{\partial e_{1,2}}{w^1_{1,2}} & \frac{\partial e_{1,1}}{w^1_{2,2}} & \cdots & \frac{\partial e_{1,2}}{w^1_{S^1,2}} & \cdots \\ \frac{\partial e_{1,3}}{\partial w^1_{1,1}} & \frac{\partial e_{1,3}}{\partial w^1_{2,1}} & \cdots & \frac{\partial e_{1,3}}{\partial w^1_{S^1,1}} & \frac{\partial e_{1,3}}{w^1_{1,2}} & \frac{\partial e_{1,3}}{w^1_{2,2}} & \cdots & \frac{\partial e_{1,3}}{w^1_{S^1,2}} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots \\ \frac{\partial e_{1,Q}}{\partial w^1_{1,1}} & \frac{\partial e_{1,Q}}{\partial w^1_{2,1}} & \cdots & \frac{\partial e_{1,Q}}{\partial w^1_{S^1,1}} & \frac{\partial e_{1,Q}}{w^1_{1,2}} & \frac{\partial e_{1,Q}}{w^1_{2,2}} & \cdots & \frac{\partial e_{1,Q}}{w^1_{S^1,2}} & \cdots \end{bmatrix}$$

$$\begin{matrix} \cdots & \frac{\partial e_{1,1}}{\partial w^1_{1,R}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{b^1_1} & \cdots & \frac{\partial e_{1,1}}{b^1_{S^1}} & \frac{\partial e_{1,1}}{w^2_{1,1}} & \frac{\partial e_{1,1}}{w^2_{1,2}} & \cdots \\ \vdots & \frac{\partial e_{1,2}}{\partial w^1_{1,R}} & \vdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{b^1_1} & \vdots & \frac{\partial e_{1,2}}{b^1_{S^1}} & \frac{\partial e_{1,2}}{w^2_{1,1}} & \frac{\partial e_{1,2}}{w^2_{1,2}} & \vdots \\ \vdots & \frac{\partial e_{1,3}}{\partial w^1_{1,R}} & \vdots & \frac{\partial e_{1,3}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,3}}{b^1_1} & \vdots & \frac{\partial e_{1,3}}{b^1_{S^1}} & \frac{\partial e_{1,3}}{w^2_{1,1}} & \frac{\partial e_{1,3}}{w^2_{1,2}} & \vdots \\ \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ \vdots & \frac{\partial e_{1,Q}}{\partial w^1_{1,R}} & \vdots & \frac{\partial e_{1,Q}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,Q}}{b^1_1} & \vdots & \frac{\partial e_{1,Q}}{b^1_{S^1}} & \frac{\partial e_{1,Q}}{w^2_{1,1}} & \frac{\partial e_{1,Q}}{w^2_{1,2}} & \vdots \end{matrix}$$

$$\begin{matrix} \cdots & \frac{\partial e_{1,1}}{\partial w^2_{1,S^1}} & \frac{\partial e_{1,1}}{\partial b^2_1} \\ \vdots & \frac{\partial e_{1,2}}{\partial w^2_{1,S^1}} & \frac{\partial e_{1,2}}{\partial b^2_1} \\ \vdots & \frac{\partial e_{1,3}}{\partial w^2_{1,S^1}} & \frac{\partial e_{1,3}}{\partial b^2_1} \\ \ddots & \cdots & \cdots \\ \vdots & \frac{\partial e_{1,Q}}{\partial w^2_{1,S^1}} & \frac{\partial e_{1,Q}}{\partial b^2_1} \end{matrix}_{Q\text{x}\left((S^1\text{x}R)+S^1+(1\text{x}S^1)+1\right)} \tag{4.58}$$

$$\mathbf{J(x)} = \begin{bmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,n} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ J_{(Q\text{x}S^M),1} & J_{(Q\text{x}S^M),2} & \cdots & J_{(Q\text{x}S^M),n} \end{bmatrix} \tag{4.59}$$

Cálculo de los elementos de la matriz Jacobiana con respecto a los pesos

A continuación, se calcula el primer elemento $J_{1,1}$ de la matriz Jacobiana

$$J_{1,1} = \left[\frac{e_{1,1}}{\partial w^1_{1,1}}\right] \tag{4.60}$$

Se sabe que el error es:

$$e_{1,1} = t_{1,1} - a^2_{1,1}. \tag{4.61}$$

De la ecuación (4.16),

$$a^2_{1,1} = w^2_{1,1}a^1_{1,1} + w^2_{1,2}a^1_{2,1} + w^2_{1,3}a^1_{3,1} \ldots + w^2_{1,S^1}a^1_{S^1,1} + b^2_1, \tag{4.62}$$

de la ecuación (4.11),

$\mathbf{a^1}$

$$= \begin{bmatrix} tansig\left(w^2_{1,1}a^1_{1,1} + w^2_{1,2}a^1_{2,1} + w^2_{1,3}a^1_{3,1} + \cdots + w^2_{1,S^1}a^1_{S^1,1} + b^2_1\right) & \cdots \\ tansig\left(w^1_{2,1}p_{1,1} + w^1_{2,2}p_{2,1} + w^1_{2,3}p_{3,1} + \cdots + w^1_{2,R}p_{R,1} + b^1_2\right) & \cdots \\ \vdots & \ddots \\ tansig\left(w^1_{S^1,1}p_{1,1} + w^1_{S^1,2}p_{2,1} + w^1_{S^1,3}p_{3,1} + \cdots + w^1_{S^1,R}p_{R,1} + b^1_{S^1}\right) & \cdots \end{bmatrix}_{S^1 x Q}$$

$$= \begin{bmatrix} a^1_{1,1} & \cdots \\ a^1_{2,1} & \cdots \\ \vdots & \ddots \\ a^1_{S^1,1} & \cdots \end{bmatrix}_{S^1 x Q} \tag{4.63}$$

remplazando $a^1_{1,1}$, $a^1_{2,1}$, ... en $a^2_{1,1}$,

$$\begin{aligned} a^2_{1,1} = w^2_{1,1} \times tansig\left(w^1_{1,1}p_{1,1} + w^1_{1,2}p_{2,1} + w^1_{1,3}p_{3,1} + \cdots + w^1_{1,R}p_{R,1}\right. \\ \left. + b^1_1\right) \\ + w^2_{1,2} \\ \times tansig\left(w^1_{2,1}p_{1,1} + w^1_{2,2}p_{2,1} + + \cdots + w^1_{2,R}p_{R,1} + b^1_2\right) \\ + \cdots \\ + w^2_{1,S^1} \\ \times tansig\left(w^1_{S^1,1}p_{1,1} + w^1_{S^1,2}p_{2,1} + \cdots + w^1_{S^1,R}p_{R,1} + b^1_{S^1}\right) \end{aligned} \tag{4.64}$$

tomando derivadas parciales,

$$\frac{\partial e_{1,1}}{w_{1,1}^1} = \frac{\partial\left(t_{1,1} - a_{1,1}^2\right)}{\partial w_{1,1}^1} = \frac{-\partial\left(a_{1,1}^2\right)}{\partial w_{1,1}^1}$$

$$= \frac{-\partial\left(w_{1,1}^2 \times tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + w_{1,R}^1 p_{R,1} + b_1^1\right)\right)}{\partial w_{1,1}^1}$$

$$+ \frac{-\partial\left(w_{1,2}^2 \times tansig\left(w_{2,1}^1 p_{1,1} + w_{2,2}^1 p_{2,1} + \cdots + w_{2,R}^1 p_{R,1} + b_2^1\right)\right)}{\partial w_{1,1}^1} + \cdots$$

$$+ \frac{-\partial\left(w_{1,S^1}^2 \times tansig\left(w_{S^1,1}^1 p_{1,1} + w_{S^1,2}^1 p_{2,1} + \cdots + w_{S^1,R}^1 p_{R,1} + b_{S^1}^1\right)\right)}{\partial w_{1,1}^1}$$

$$= \frac{-\partial\left(w_{1,1}^2 \times tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + w_{1,R}^1 p_{R,1} + b_1^1\right)\right)}{\partial w_{1,1}^1} + 0$$

$$+ \cdots + 0$$

$$\frac{\partial e_{1,1}}{\partial w_{1,1}^1} = -w_{1,1}^2 \frac{\partial\left(tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + b_1^1\right)\right)}{\partial w_{1,1}^1} \qquad (4.65)$$

Para facilitar el cálculo, se considera lo siguiente,

$$tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + b_1^1\right) = tansig\left(w_{1,1}^1 p_{1,1} + k\right) \qquad (4.66)$$

donde k, es una función no dependiente de $w_{1,1}^1$, reemplazando

$$\frac{\partial\left(tansg\left(w_{1,1}^{1}\ _{1,1}+k\right)\right)}{\partial w_{1,1}^{1}}=\frac{\partial\left(\frac{2}{1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}-1\right)}{\partial w_{1,1}^{1}}$$

$$=2\frac{\partial\left(\frac{1}{1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}\right)}{\partial w_{1,1}^{1}}-\frac{\partial(1)}{\partial w_{1,1}^{1}}$$

$$=2\frac{\left(\frac{\partial(1)}{\partial w_{1,1}^{1}}\right)\times\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)-1\times\frac{\partial\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)}{\partial w_{1,1}^{1}}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}}$$

$$=2\frac{0-1\times\frac{\partial\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)}{\partial w_{1,1}^{1}}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}}$$

$$=-2\frac{\partial\left(\frac{1}{\partial w_{1,1}^{1}}\right)+\frac{\partial\left(-2\left(w_{1,1}^{1}p_{1,1}\ +k\right)\right)}{\partial w_{1,1}^{1}}\times e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}}$$

$$=-2\frac{-2p_{1,1}\ \times e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}}$$

de donde se obtiene:

$$\frac{\partial\left(tansig\left(w_{1,1}^{1}p_{1,1}\ +k\right)\right)}{\partial w_{1,1}^{1}}=\frac{4p_{1,1}\ \times e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}} \qquad (4.67)$$

La derivada parcial respecto al parámetro $w_{1,1}^{1}$ en la ecuación (4.67) es posible obtenerse sin la necesidad de realizar la derivada parcial de manera explícita, como lo demostraremos a continuación:

Se sabe que:

$$tansig(x)=\frac{2}{1+e^{-2x}}-1 \qquad (4.68)$$

haciendo $x=w_{1,1}^{1}p_{1,1}\ +k$,

$$tansig\left(w_{1,1}^{1}p_{1,1} + k\right) = \frac{2}{1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}} - 1 \qquad (4.69)$$

Dando forma a la ecuación (4.69) para obtener el denominador del lado derecho de la ecuación (4.67),

$$\left(tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1\right)^2 = \left(\frac{2}{1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}}\right)^2$$

$$\left(tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1\right)^2 = \frac{4}{\left(1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}\right)^2} \qquad (4.70)$$

Dando forma a la ecuación (4.69) para obtener el numerador del lado derecho de la ecuación (4.67),

$$tansig\left(w_{1,1}^{1}p_{1,1} + k\right) = \frac{2}{1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}} - 1$$

$$tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1 = \frac{2}{1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}}$$

$$tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1 = \frac{2}{1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}}$$

$$\frac{2}{tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1} - 1 = e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}$$

$$\frac{2 - tansig\left(w_{1,1}^{1}p_{1,1} + k\right) - 1}{tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1} = e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}$$

de donde se obtiene:

$$\frac{1 - tansig\left(w_{1,1}^{1}p_{1,1} + k\right)}{tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1} = e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}. \qquad (4.71)$$

Multiplicando las ecuaciones (4.70) y (4.71) :

$$\left(tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1\right)^2 \left(\frac{1 - tansig\left(w_{1,1}^{1}p_{1,1} + k\right)}{tansig\left(w_{1,1}^{1}p_{1,1} + k\right) + 1}\right)$$

$$= \frac{4}{\left(1 + e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)}\right)^2} e^{-2\left(w_{1,1}^{1}p_{1,1} + k\right)} \qquad (4.72)$$

ordenando,

$$p_{1,1}\left(1-\left(tansig\left(w_{1,1}^{1}p_{1,1}+k\right)\right)^{2}\right)=\frac{4p_{1,1}\times e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}}{\left(1+e^{-2\left(w_{1,1}^{1}p_{1,1}+k\right)}\right)^{2}} \qquad (4.73)$$

Se observa que el lado derecho de las ecuaciones (4.67) y (4.73) son iguales, por lo que,

$$\frac{\partial\left(tansig\left(w_{1,1}^{1}p_{1,1}+k\right)\right)}{\partial w_{1,1}^{1}}=p_{1,1}\left(1-\left(tansig\left(w_{1,1}^{1}p_{1,1}+k\right)\right)^{2}\right) \qquad (4.74)$$

Remplazamos la ecuación (4.74) en la ecuación (4.65):

$$\frac{\partial e_{1,1}}{\partial w_{1,1}^{1}}=-w_{1,1}^{2}\left(p_{1,1}\left(1-\left(tansig\left(w_{1,1}^{1}p_{1,1}+k\right)\right)^{2}\right)\right) \qquad (4.75)$$

De la igualdad (4.63):

$$a_{1,1}^{1}=tansig\left(w_{1,1}^{1}p_{1,1}+k\right)$$

Por lo tanto, el primer elemento de la matriz jacobiana es:

$$J_{1,1}=\frac{\partial e_{1,1}}{\partial w_{1,1}^{1}}=p_{1,1}\left(-w_{1,1}^{2}\left(1-\left(a_{1,1}^{1}\right)^{2}\right)\right) \qquad (4.76)$$

De la misma forma es posible calcular los elementos restantes de la matriz jacobiana, con respecto a los pesos,

$$J_{1,2}=\frac{\partial e_{1,1}}{\partial w_{2,1}^{1}}=p_{1,1}\left(-w_{1,2}^{2}\left(1-\left(a_{2,1}^{1}\right)^{2}\right)\right) \qquad (4.77)$$

$$J_{2,1}=\frac{\partial e_{1,2}}{\partial w_{1,1}^{1}}=p_{1,2}\left(-w_{1,1}^{2}\left(1-\left(a_{1,2}^{1}\right)^{2}\right)\right) \qquad (4.78)$$

$$J_{2,2} = \frac{\partial e_{1,2}}{\partial w_{2,1}^1} = p_{1,2}\left(-w_{1,2}^2\left(1-\left(a_{2,2}^1\right)^2\right)\right)$$

$$J_{1,3} = \cdots$$

Cálculo de los elementos de la matriz Jacobiana con respecto a los bias

Los elementos de la matriz jacobiana con respecto a los bias (derivadas parciales del vector de parámetros, respecto a los bias) se pueden calcular de la siguiente manera:

$$\frac{\partial e_{1,1}}{\partial b_1^1} = \frac{\partial\left(t_{1,1} - a_{1,1}^2\right)}{\partial b_1^1} \tag{4.79}$$

De la ecuación (4.16) y (4.17) se tiene que:

$$a_{1,1}^2 = w_{1,1}^2 a_{1,1}^1 + w_{1,2}^2 a_{2,1}^1 + w_{1,3}^2 a_{3,1}^1 \ldots + w_{1,S^1}^2 a_{S^1,1}^1 + b_1^2 \tag{4.80}$$

en la ecuación (4.80) remplazamos los elementos de la ecuación (4.11) y (4.12)

$$\begin{aligned} a_{1,1}^2 = w_{1,1}^2 \times tansig\big(& w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + w_{1,3}^1 p_{3,1} + \cdots + w_{1,R}^1 p_{R,1} \\ & + b_1^1\big) \\ & + w_{1,2}^2 \\ & \times tansig\big(w_{2,1}^1 p_{1,1} + w_{2,2}^1 p_{2,1} + + \cdots + w_{2,R}^1 p_{R,1} + b_2^1\big) \\ & + \cdots \\ & + w_{1,S^1}^2 \\ & \times tansig\big(w_{S^1,1}^1 p_{1,1} + w_{S^1,2}^1 p_{2,1} + \cdots + w_{S^1,R}^1 p_{R,1} + b_{S^1}^1\big) \end{aligned} \tag{4.81}$$

De la ecuación (*4.79*),

$$\frac{\partial e_{1,1}}{b_1^1} = \frac{\partial\left(t_{1,1} - a_{1,1}^2\right)}{\partial b_1^1} = \frac{-\partial\left(a_{1,1}^2\right)}{\partial b_1^1}$$

$$= \frac{-\partial\left(w_{1,1}^2 \times tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + w_{1,3}^1 p_{3,1} + \cdots + w_{1,R}^1 p_{R,1} + b_1^1\right)\right)}{\partial b_1^1}$$

$$+ \frac{-\partial\left(w_{1,2}^2 \times tansig\left(w_{2,1}^1 p_{1,1} + w_{2,2}^1 p_{2,1} + w_{2,3}^1 p_{3,1} + \cdots + w_{2,R}^1 p_{R,1} + b_2^1\right)\right)}{\partial b_1^1} + \cdots$$

$$+ \frac{-\partial\left(w_{1,S^1}^2 \times tansig\left(w_{S^1,1}^1 p_{1,1} + w_{S^1,2}^1 p_{2,1} + w_{S^1,3}^1 p_{3,1} + \cdots + w_{S^1,R}^1 p_{R,1} + b_{S^1}^1\right)\right)}{\partial b_1^1}$$

$$= \frac{-\partial\left(w_{1,1}^2 \times tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + w_{1,3}^1 p_{3,1} + \cdots + w_{1,R}^1 p_{R,1} + b_1^1\right)\right)}{\partial b_1^1} + 0 + \cdots$$

$$+ 0$$

de donde se obtiene:

$$\frac{\partial e_{1,1}}{\partial b_1^1} = -w_{1,1}^2 \frac{\partial\left(tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + b_1^1\right)\right)}{\partial b_1^1} \qquad (4.82)$$

Para facilitar el cálculo, se considera:

$$tansig\left(w_{1,1}^1 p_{1,1} + w_{1,2}^1 p_{2,1} + \cdots + b_1^1\right) = tansig\left(b_1^1 + k\right)$$

donde k, es función que no depende de b_1^1.

$$\frac{\partial\left(tansg(b_1^1+k)\right)}{\partial b_1^1}=\frac{\partial\left(\frac{2}{1+e^{-2(b_1^1+k)}}-1\right)}{\partial b_1^1}=2\frac{\partial\left(\frac{1}{1+e^{-2(b_1^1+k)}}\right)}{\partial b_1^1}-\frac{\partial(1)}{\partial b_1^1}$$

$$=2\frac{\left(\frac{\partial(1)}{\partial b_1^1}\right)\times\left(1+e^{-2(b_1^1+k)}\right)-1\times\frac{\partial\left(1+e^{-2(b_1^1+k)}\right)}{\partial b_1^1}}{\left(1+e^{-2(b_1^1+k)}\right)^2}$$

$$=2\frac{0-1\times\frac{\partial\left(1+e^{-2(b_1^1+k)}\right)}{\partial b_1^1}}{\left(1+e^{-2(b_1^1+k)}\right)^2}$$

$$=-2\frac{\partial\left(\frac{1}{\partial w_{1,1}^1}\right)+\frac{\partial\left(-2(b_1^1+k)\right)}{\partial b_1^1}\times e^{-2(b_1^1+k)}}{\left(1+e^{-2(b_1^1+k)}\right)^2}$$

$$=-2\frac{-2(1)\times e^{-2(b_1^1+k)}}{\left(1+e^{-2(b_1^1+k)}\right)^2}$$

de donde se obtiene:

$$\frac{\partial\left(tansig(b_1^1+k)\right)}{\partial b_1^1}=\frac{4\times e^{-2(b_1^1+k)}}{\left(1+e^{-2(b_1^1+k)}\right)^2} \qquad (4.83)$$

La derivada parcial en el lado izquierdo de la igualdad en la ecuación (4.83) es posible obtenerse sin la necesidad de realizar la derivada parcial con respecto a los bias, como se demuestra a continuación:

Se sabe que:

$$tansig(b_1^1+k)=\frac{2}{1+e^{-2(b_1^1+k)}}-1 \qquad (4.84)$$

Dando forma a la ecuación (4.84) con el fin de obtener el denominador del lado derecho de la ecuación (4.83),

$$(tansig(b_1^1+k)+1)^2=\left(\frac{2}{1+e^{-2(b_1^1+k)}}\right)^2 \qquad (4.85)$$

$$(tansi(b_1^1 + k) + 1)^2 = \frac{4}{\left(1 + e^{-2(b_1^1+k)}\right)^2}$$

Dando forma a la ecuación (4.84) con el fin de obtener el numerador del lado derecho de la ecuación (4.83),

$$tansig(b_1^1 + k) = \frac{2}{1 + e^{-2(b_1^1+k)}} - 1$$

$$tansig(b_1^1 + k) + 1 = \frac{2}{1 + e^{-2(b_1^1+k)}}$$

$$tansig(b_1^1 + k) + 1 = \frac{2}{1 + e^{-2(b_1^1+k)}}$$

$$\frac{2}{tansig(b_1^1 + k) + 1} - 1 = e^{-2(b_1^1+k)}$$

$$\frac{2 - tansig(b_1^1 + k) - 1}{tansig(b_1^1 + k) + 1} = e^{-2(b_1^1+k)}$$

de donde se obtiene:

$$\frac{1 - tansig(b_1^1 + k)}{tansig(b_1^1 + k) + 1} = e^{-2(b_1^1+k)} \qquad (4.86)$$

Multiplicando las ecuaciones (4.85) y (4.86), tenemos:

$$(tansig(b_1^1 + k) + 1)^2 \left(\frac{1 - tansig(b_1^1 + k)}{tansig(b_1^1 + k) + 1}\right) = \frac{4}{\left(1 + e^{-2(b_1^1+k)}\right)^2} e^{-2(b_1^1+k)} \qquad (4.87)$$

ordenando,

$$\left(1 - \left(tansig(b_1^1 + k)\right)^2\right) = \frac{4 \times e^{-2(b_1^1+k)}}{\left(1 + e^{-2(b_1^1+k)}\right)^2} \qquad (4.88)$$

Observamos que el lado derecho de las ecuaciones (4.83) y (4.88) son iguales, por lo que, el lado izquierdo de ambas ecuaciones es iguales:

$$\frac{\partial\left(tansig(b_1^1+k)\right)}{\partial b_1^1}=\left(1-\left(tansig(b_1^1+k)\right)^2\right) \tag{4.89}$$

$$\frac{\partial e_{1,1}}{\partial b_1^1}=-w_{1,1}^2\frac{\partial\left(tansig\left(w_{1,1}^1p_{1,1}+w_{1,2}^1p_{2,1}+\cdots+b_1^1\right)\right)}{\partial b_1^1}$$

Remplazamos la ecuación (4.89) en la ecuación (4.82):

$$\frac{\partial e_{1,1}}{\partial w_{1,1}^1}=-w_{1,1}^2\left(1-\left(tansig\left(w_{1,1}^1p_{1,1}+w_{1,2}^1p_{2,1}+\cdots+b_1^1\right)\right)^2\right) \tag{4.90}$$

De la igualdad (4.63),

$$a_{1,1}^1=tansig\left(w_{1,1}^1p_{1,1}+w_{1,2}^1p_{2,1}+\cdots+b_1^1\right)$$

Por lo tanto, el primer elemento de la matriz jacobiana con respecto a la derivada parcial respecto a los bias es:

$$J_{1,(RxS^1+1)}=\frac{\partial e_{1,1}}{\partial b_1^1}=\left(-w_{1,1}^2\left(1-\left(a_{1,1}^1\right)^2\right)\right) \tag{4.91}$$

de la misma manera se puede obtener las demás derivadas parciales respecto a los bias, por ejemplo:

$$J_{2,(RxS^1+1)}=\frac{\partial e_{1,1}}{\partial b_1^1}=\left(-w_{1,1}^2\left(1-\left(a_{1,2}^1\right)^2\right)\right) \tag{4.92}$$

De lo anterior se concluye que todos los elementos de la matriz jacobiana de puede componer sin recurrir a las derivadas parciales.

Cálculo completo de los elementos de la matriz jacobiana

Cálculo de todos los elementos de la matriz Jacobiana, mediante operaciones elementales y simples entre vectores y matrices.

Cabe recordar que este algoritmo se está diseñando para una RNA de dos capas, con R entradas, S^1 neuronas en la capa oculta, y una neurona ($S^2 = 1$) en la capa de salida, con función de activación tangente sigmoidal (tansig) en la capa oculta, y función de activación lineal (pureline) en capa de salida)

$$\mathbf{a^1 = A1}$$

$$\mathbf{A1} = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & \cdots & a^1_{1,Q} \\ a^1_{2,1} & a^1_{2,2} & \cdots & a^1_{2,Q} \\ \vdots & \vdots & \ddots & \vdots \\ a^1_{S^1,1} & a^1_{S^1,2} & \cdots & a^1_{S^1,Q} \end{bmatrix}_{S^1 x Q} \tag{4.93}$$

$$\mathrm{D2} = [-1 \quad -1 \quad \ldots \quad -1]_{1xQ} \tag{4.94}$$

Sea $h_1 = (ones - (\mathbf{A1}.\times \mathbf{A1}))$, entonces:

$$h_1 = \begin{bmatrix} 1-(a^1_{1,1})^2 & 1-(a^1_{1,2})^2 & \cdots & 1-(a^1_{1,Q})^2 \\ 1-(a^1_{2,1})^2 & 1-(a^1_{2,2})^2 & \cdots & 1-(a^1_{2,Q})^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1-(a^1_{S^1,1})^2 & 1-(a^1_{S^1,2})^2 & \cdots & 1-(a^1_{S^1,Q})^2 \end{bmatrix}_{S^1 x Q} \tag{4.95}$$

Sea $h_2 = (\mathbf{W^2})^T \times \mathrm{D2}$,

$$\mathbf{W^2} = \begin{bmatrix} w^2_{1,1} & w^2_{1,2} & \cdots & w^2_{1,S^1} \end{bmatrix}_{1xS^1}$$

entonces:

$$h_2 = (\mathbf{W^2})^T \times \mathrm{D2} = \left(\begin{bmatrix} w^2_{1,1} \\ w^2_{1,2} \\ \vdots \\ w^2_{1,S^1} \end{bmatrix}_{1xS^1} \times [-1 \quad -1 \quad \ldots \quad -1]_{1xQ} \right) \tag{4.96}$$

$$h_2 = \begin{bmatrix} -w^2_{1,1} & -w^2_{1,1} & \cdots & -w^2_{1,1} \\ -w^2_{1,2} & -w^2_{1,2} & \cdots & -w^2_{1,2} \\ \vdots & \vdots & \ddots & \vdots \\ -w^2_{1,S^1} & -w^2_{1,S^1} & \cdots & -w^2_{1,S^1} \end{bmatrix}_{S^1 x Q} \tag{4.97}$$

Sea $D1 = h_1 .\times h_2$,

$$D1 = \begin{bmatrix} -w_{1,1}^2\left(1-\left(a_{1,1}^1\right)^2\right) & -w_{1,1}^2\left(1-\left(a_{1,2}^1\right)^2\right) & \cdots \\ -w_{1,2}^2\left(1-\left(a_{2,1}^1\right)^2\right) & -w_{1,2}^2\left(1-\left(a_{2,2}^1\right)^2\right) & \cdots \\ \vdots & \vdots & \ddots \\ -w_{1,S^1}^2\left(1-\left(a_{S^1,1}^1\right)^2\right) & -w_{1,S^1}^2\left(1-\left(a_{S^1,2}^1\right)^2\right) & \cdots \end{bmatrix}$$

$$\begin{matrix} \cdots & -w_{1,1}^2\left(1-\left(a_{1,Q}^1\right)^2\right) \\ \cdots & -w_{1,2}^2\left(1-\left(a_{2,Q}^1\right)^2\right) \\ \ddots & \vdots \\ \cdots & -w_{1,S^1}^2\left(1-\left(a_{S^1,Q}^1\right)^2\right) \end{matrix}\Bigg]_{S^1 x Q} \qquad (4.98)$$

Sea $g_1 = kron\left(\mathbf{p^T}, ones(1, S^1)\right)$,

$$\mathbf{p^T} = \begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{R,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{R,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,Q} & p_{2,Q} & \cdots & p_{R,Q} \end{bmatrix}_{QxR}$$

$$g_1 = \begin{bmatrix} (p_{1,1})_{1,1} & (p_{1,1})_{1,2} & \cdots & (p_{1,1})_{1,S^1} & (p_{2,1})_{1,(1+S^1)} & \cdots \\ (p_{1,2})_{2,1} & (p_{1,2})_{2,2} & \cdots & (p_{1,2})_{2,S^1} & (p_{2,2})_{2,(1+S^1)} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ (p_{1,Q})_{Q,1} & (p_{1,Q})_{Q,2} & \cdots & (p_{1,Q})_{Q,S^1} & (p_{2,Q})_{Q,(1+S^1)} & \cdots \end{bmatrix}$$

$$\begin{matrix} \cdots & (p_{2,1})_{1,2S^1} & \cdots & (p_{R,1})_{1,RS^1} \\ \cdots & (p_{2,2})_{2,2S^1} & \cdots & (p_{R,2})_{2,RS^1} \\ \ddots & \vdots & \ddots & \vdots \\ \cdots & (p_{2,Q})_{Q,2S^1} & \cdots & (p_{R,Q})_{Q,RS^1} \end{matrix}\Bigg]_{Qx(RxS^1)} \qquad (4.99)$$

Sea $g_2 = kron(\, ones(1, R), \mathrm{D1}^{\mathrm{T}})$,

$$g_2 = \begin{bmatrix} \left(-w^2_{1,1}\left(1-\left(a^1_{1,1}\right)^2\right)\right)_{1,1} & \left(-w^2_{1,2}\left(1-\left(a^1_{2,1}\right)^2\right)\right)_{1,2} & \cdots \\ \left(-w^2_{1,1}\left(1-\left(a^1_{1,2}\right)^2\right)\right)_{2,1} & \left(-w^2_{1,2}\left(1-\left(a^1_{2,2}\right)^2\right)\right)_{2,2} & \cdots \\ \vdots & \vdots & \ddots \\ \left(-w^2_{1,1}\left(1-\left(a^1_{1,Q}\right)^2\right)\right)_{Q,1} & \left(-w^2_{1,2}\left(1-\left(a^1_{2,Q}\right)^2\right)\right)_{Q,2} & \cdots \end{bmatrix}$$

$$\begin{matrix} \cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,1}\right)^2\right)\right)_{1,S^1} & \left(-w^2_{1,1}\left(1-\left(a^1_{1,1}\right)^2\right)\right)_{1,(S^1+1)} & \cdots \\ \cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,2}\right)^2\right)\right)_{2,S^1} & \left(-w^2_{1,1}\left(1-\left(a^1_{1,2}\right)^2\right)\right)_{2,(S^1+1)} & \cdots \\ \ddots & \vdots & \vdots & \ddots \\ \cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,Q}\right)^2\right)\right)_{Q,S^1} & \left(-w^2_{1,1}\left(1-\left(a^1_{1,Q}\right)^2\right)\right)_{Q,(S^1+1)} & \cdots \end{matrix}$$

$$\begin{matrix} \cdots\cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,1}\right)^2\right)\right)_{1,(RxS^1)} \\ \cdots\cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,2}\right)^2\right)\right)_{2,(RxS^1)} \\ \ddots\ddots & \vdots \\ \cdots\cdots & \left(-w^2_{1,S^1}\left(1-\left(a^1_{S^1,Q}\right)^2\right)\right)_{Q,(RxS^1)} \end{matrix}_{Qx(RxS^1)} \qquad (4.100)$$

Jacobiano parcial: $J1 = g_1 .* g_2$,

$$J1 = \begin{bmatrix} p_{1,1}\left(-w_{1,1}^2\left(1-\left(a_{1,1}^1\right)^2\right)\right) & p_{1,1}\left(-w_{1,2}^2\left(1-\left(a_{2,1}^1\right)^2\right)\right) & \cdots \\ p_{1,2}\left(-w_{1,1}^2\left(1-\left(a_{1,2}^1\right)^2\right)\right) & p_{1,2}\left(-w_{1,2}^2\left(1-\left(a_{2,2}^1\right)^2\right)\right) & \cdots \\ \vdots & \vdots & \ddots \\ p_{1,Q}\left(-w_{1,1}^2\left(1-\left(a_{1,Q}^1\right)^2\right)\right) & p_{1,Q}\left(-w_{1,2}^2\left(1-\left(a_{2,Q}^1\right)^2\right)\right) & \cdots \end{bmatrix}$$

$$\begin{matrix} p_{1,1}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,1}^1\right)^2\right)\right) & p_{2,1}\left(-w_{1,1}^2\left(1-\left(a_{1,1}^1\right)^2\right)\right) & \cdots \\ p_{1,2}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,2}^1\right)^2\right)\right) & p_{2,2}\left(-w_{1,1}^2\left(1-\left(a_{1,2}^1\right)^2\right)\right) & \cdots \\ \vdots & \vdots & \ddots \\ p_{1,Q}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,Q}^1\right)^2\right)\right) & p_{2,Q}\left(-w_{1,1}^2\left(1-\left(a_{1,Q}^1\right)^2\right)\right) & \cdots \end{matrix}$$

$$\begin{matrix} \cdots & p_{R,1}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,1}^1\right)^2\right)\right) \\ \cdots & p_{R,2}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,2}^1\right)^2\right)\right) \\ \ddots & \vdots \\ \cdots & p_{R,Q}\left(-w_{1,S^1}^2\left(1-\left(a_{S^1,Q}^1\right)^2\right)\right) \end{matrix}\Bigg]_{Q\text{x}(R\text{x}S^1)} \qquad (4.101)$$

Sea $c_1 = kron(\mathrm{A1}^T, one(1, S^1))$

$$\mathrm{A1} = \begin{bmatrix} a_{1,1}^1 & a_{1,2}^1 & \cdots & a_{1,Q}^1 \\ a_{2,1}^1 & a_{2,2}^1 & \cdots & a_{2,Q}^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{S^1,1}^1 & a_{S^1,2}^1 & \cdots & a_{S^1,Q}^1 \end{bmatrix}_{S^1\text{xQ}}$$

$$\mathrm{A1}^T = \begin{bmatrix} a_{1,1}^1 & a_{2,1}^1 & \cdots & a_{S^1,1}^1 \\ a_{1,2}^1 & a_{2,2}^1 & \cdots & a_{S^1,2}^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,Q}^1 & a_{2,Q}^1 & \cdots & a_{S^1,Q}^1 \end{bmatrix}_{Q\text{x}S^1} \qquad (4.102)$$

$$c_1 = \begin{bmatrix} a_{1,1}^1 & a_{2,1}^1 & \cdots & a_{S^1,1}^1 \\ a_{1,2}^1 & a_{2,2}^1 & \cdots & a_{S^1,2}^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,Q}^1 & a_{2,Q}^1 & \cdots & a_{S^1,Q}^1 \end{bmatrix}_{Q\text{x}S^1} \qquad (4.103)$$

Sea $c_2 = kron(one(1, S^1), \mathrm{D2}^T)$

$$one(1, S^1) = [1 \quad 1 \quad \cdots \quad 1]_{1xS^1}$$

$$\mathrm{D2} = [-1 \quad -1 \quad \ldots \quad -1]_{1xQ}$$

$$\mathrm{D2}^{\mathrm{T}} = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}_{Qx1} \tag{4.104}$$

$$c_2 = \begin{bmatrix} -1 & -1 & \cdots & -1 \\ -1 & -1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & -1 \end{bmatrix}_{QxS^1} \tag{4.105}$$

Sea $\mathrm{J2} = [c_1.\times c_2]$ el jacobiano parcial 2,

$$\mathrm{J2} = \begin{bmatrix} -a^1_{1,1} & -a^1_{2,1} & \cdots & -a^1_{S^1,1} \\ -a^1_{1,2} & -a^1_{2,2} & \cdots & -a^1_{S^1,2} \\ \vdots & \vdots & \ddots & \vdots \\ -a^1_{1,Q} & -a^1_{2,Q} & \cdots & -a^1_{S^1,Q} \end{bmatrix}_{QxS^1} \tag{4.106}$$

Matriz Jacobiana completa

$$\mathbf{J} = [\mathrm{J1} \quad \mathrm{D1}^{\mathrm{T}} \quad \mathrm{J2} \quad \mathrm{D2}^{\mathrm{T}}]$$

(4.107)

$$\left[p_{1,1}\left(-w^2_{1,1}\left(1-\left(a^1_{1,1}\right)^2\right)\right) \quad p_{1,1}\left(-w^2_{1,2}\left(1-\left(a^1_{2,1}\right)^2\right)\right) \quad \cdots\right.$$

$$\begin{bmatrix} \cdots & -w^2_{1,S^1}\left(1-\left(a^1_{S^1,1}\right)^2\right) & -a^1_{1,1} & \cdots & -a^1_{S^1,1} & -1 \\ \cdots & -w^2_{1,S^1}\left(1-\left(a^1_{S^1,2}\right)^2\right) & -a^1_{1,2} & \cdots & -a^1_{S^1,2} & -1 \\ \ddots & \cdots & \vdots & \ddots & \vdots & \vdots \\ \cdots & -w^2_{1,S^1}\left(1-\left(a^1_{S^1,Q}\right)^2\right) & -a^1_{1,Q} & \cdots & -a^1_{S^1,Q} & -1 \end{bmatrix}_{Q\text{x}\left((R+2)S^1+1\right)}$$

Con lo que se queda demostrados que es posible obtener la matriz Jacobiana sin realizar las derivadas parciales respecto a los pesos y bias.

Calcular nuevas estimaciones para los parámetros de regularización $\alpha^{MP} = \frac{\gamma}{2E_W(\mathrm{x})}$ and $\beta^{MP} = \frac{N-\gamma}{2E_D(\mathrm{x})}$.

3. Ahora se itera los pasos del 1 al 3 hasta la convergencia.

4.3.4 Diseño y construcción de hardware electrónico

4.3.4.1 Descripción de principales componentes electrónicos de hardware

4.3.4.1.1 Muestreado de variación de voltajes de sensores

La impedancia de salida de un electrodo de pH es extremadamente alta, oscilando entre 10 MΩ y 1000 MΩ [123], por este motivo se seleccionó amplificadores operacionales de precisión de alta impedancia de ingreso, estos son LM2231 [124] y LM2232 [125]:

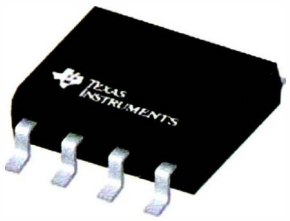


Figura 4.30. Single Micropower, 1.6V, Precision Operational Amplifier with CMOS Inputs

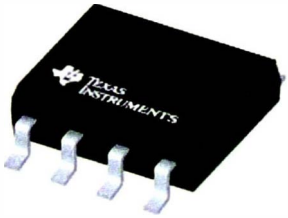


Figura 4.31. Dual, Micropower, 1.6V, Precision, Operational Amplifier with CMOS Input.

Reguladores de tensión de precisión para referenciar al sensor de PH; Regulador de 5.0V de precisión REF4132A50 [126], Regulador de 2.048V de precisión REF3320 [127].

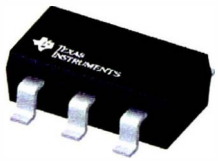


Figura 4.32. 5.00-V, 12-ppm/°C low-noise low-power precision voltage reference.

Figura 4.33. 2.048-V, 30-ppm/°C drift, 3.9-µA, 3-pin SOT-23, 3-pin SC70, 8-pin UQFN voltage reference.

4.3.4.1.2 Conversor analógico digital

El conversor analógico-digital ADS1115 Es un conversor (ADC) de precisión, baja potencia, 16 bits, cuatro entradas analógicas, un multiplexor de cuatro canales, compatibles con I^2C, que se ofrecen en un paquete ultra pequeño. Los dispositivos ADS111x incorporan una referencia de voltaje de referencia y un oscilador [128].

Figura 4.34. 16-bit, 860-SPS, 4-channel, delta-sigma ADC with PGA, oscillator, VREF, comparator and I2C *[128]*.

4.3.4.1.3 Presentación de datos

Para presentar los datos al usuario se seleccionó una pantalla LCD de 2x16 caracteres [129].

Figura 4.35. Módulo LCD de caracteres *[129]*

4.3.4.2 Descripción de diagrama de módulos de hardware

El módulo de lectura de sensores lee la variación de voltaje del sensor de PH y la variación de resistencia del sensor de temperatura PT1000. Para la lectura de PH se utiliza dos amplificadores operacionales (LMP2232) con muy alta impedancia conectados en serie entre el sensor. Para la lectura de temperatura se utilizó un divisor de tensión compuesto por una resistencia de 1000Ohms y un sensor PT1000. El sensor PT1000 se referencia a tierra, el resistor de 1000Ohms se referencia hacia 4.048V, el voltaje intermedio entre ambos dispositivos se ingresa al ADC ADS1115. En la Figura 4.36 se muestra el diagrama de bloques general del hardware electrónico.

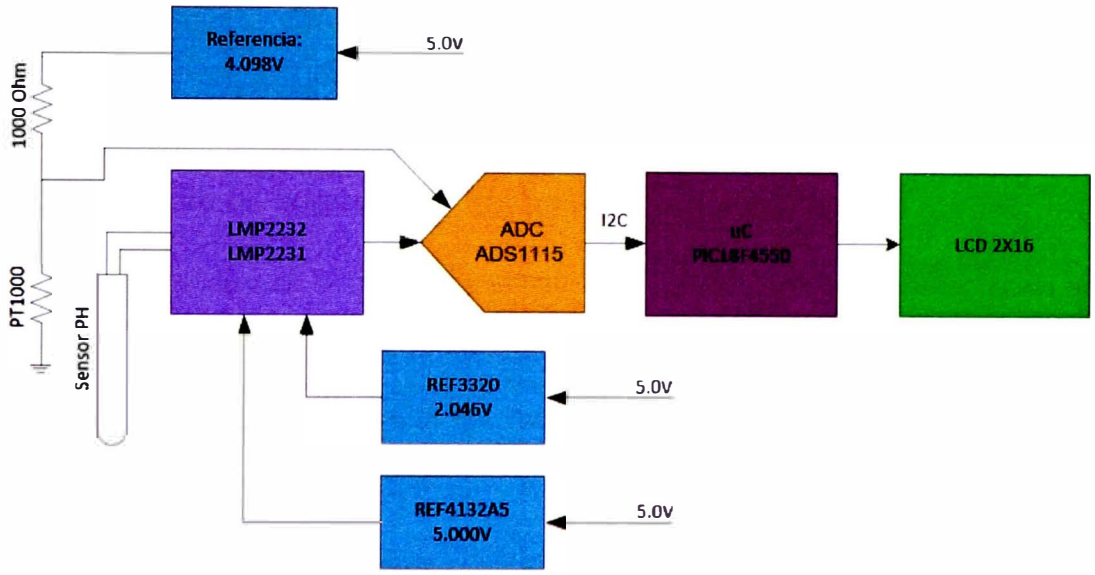


Figura 4.36. Diagrama de bloques del sistema de lectura, procesamiento y presentación de lectura de temperatura y PH.
Fuente: elaboración propia del autor

4.3.4.3 Diseño y simulación del circuito electrónico

En el diseño del hardware electrónico se ha considera su escalabilidad para más sensores de PH y de temperatura, para ello se ha utilizado el protocolo de comunicación I^2C entre el ADC y el microcontrolador, de esta manera la cantidad de sensores a integrar no estará limitado por la cantidad de entras digitales del microcontrolador. El presente trabajo se ha diseñado para 12 sensores, la elección del tipo (temperatura o PH) de sensor es configurables por software. Cabe resaltar que debe utilizarse al menos un sensor de temperatura para dar la referencia de temperatura de medición de la muestra de PH.

En la tarjeta electrónica se han ensamblado componentes electrónicos solo para un sensor de temperatura y un sensor de PH. Teóricamente según el protocolo I^2C y el DAQ utilizado se puede agregar has 512 sensores.

Otro aspecto importe considerado dentro del diseño es su modularidad con la finalidad de adaptarse a los requisitos del cliente. Actualmente solo se ha

construido un módulo con capacidad de 4 sensores, no obstante, si es necesario más sensores solo se debe adicional el hardware para la adquisición de datos.

4.3.4.3.1 Módulo de adquisición, tratamiento y digitalización

En este módulo se adquiere las variaciones de voltajes con una resolución de 16 bits provenientes de los sensores. Los amplificadores operacionales LMP2231 y MLP2232 de alta impedancia reflejan la variación del voltaje del sensor hacia el conversor ADC ADS1115, el cual consta de 4 entradas analógicas, un multiplexor de 4 canales, una referencia de voltaje y una interface I^2C para el envío de los datos hacia el microcontrolador PIC18F4550.

La lectura de PH es un proceso muy crítico y delicado dado la sensibilidad al ruido, pequeñas variaciones voltaje de alrededor de milivoltios altera significativamente el resultado medido de PH. Para referenciar voltajes se ha considerado reguladores de alta precisión REF4132A50 [126], Regulador de 2.048V de precisión REF3320 [127].

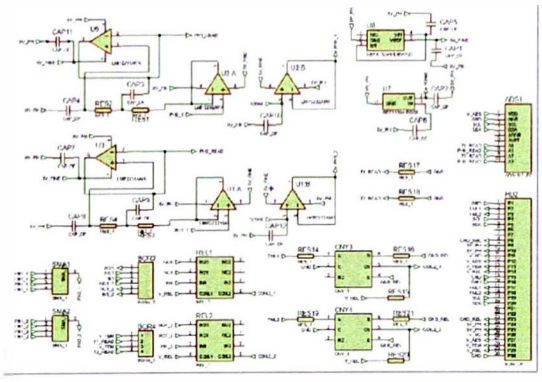


Figura 4.37. Esquema electrónico del Módulo de adquisición, tratamiento y digitalización. Fuente: elaboración propia del autor.

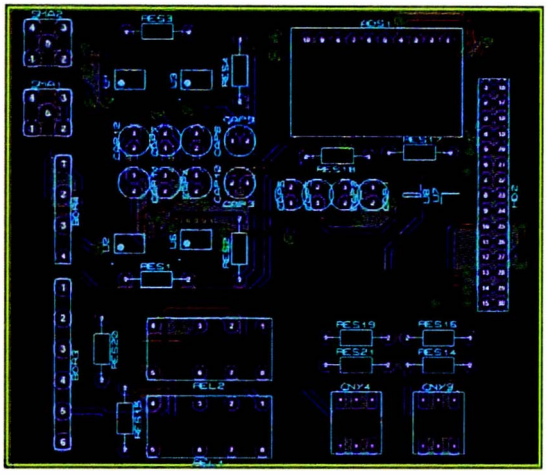


Figura 4.38. PCB Layout del Módulo de adquisición, tratamiento y digitalización.
Fuente: elaboración propia del autor.

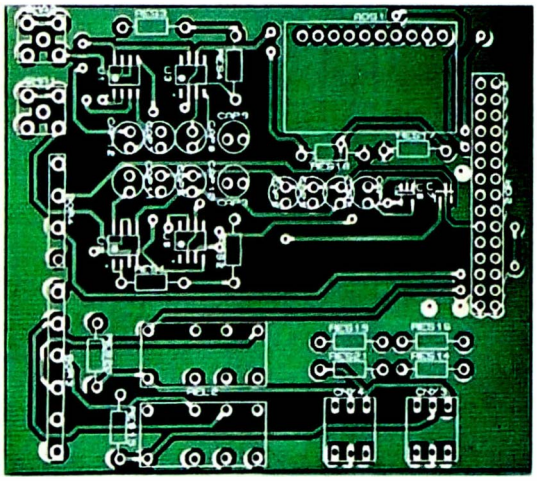


Figura 4.39. PCB físico del Módulo de adquisición, tratamiento y digitalización.
Fuente: elaboración propia del autor.

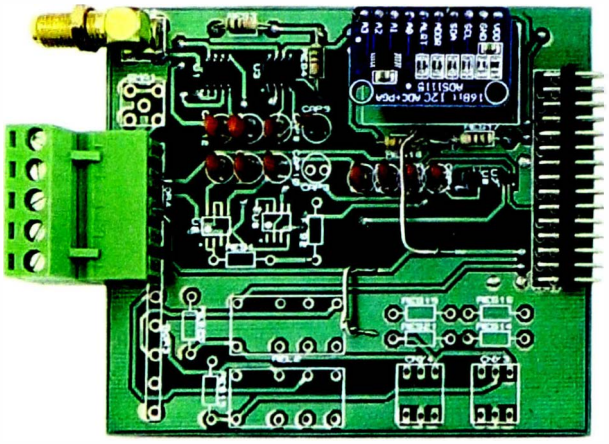


Figura 4.40. Tarjeta electrónica del Módulo de adquisición, tratamiento y digitalización.
Fuente: elaboración propia del autor

4.3.4.3.2 Módulo de procesamiento y presentación de información

El módulo de procesamiento y presentación de la información está gobernado por el microcontrolador PIC18F4550 [130], Velocidad de reloj 18MHz, memoria flash de32K, 35 entradas/salidas, protocolo I^2C. La comunicación de datos entre la DAQ y el microcontrolador se realiza mediante el protocolo I^2C. La ventaja del protocolo I^2C es que se puede adicionar más de un sensor de PH y Temperatura sin la necesidad de utilizar más entradas digitales en el microcontrolador.

La presentación de datos se realiza mediante la interface LCD de 2x16 caracteres, en ella se muestra el voltaje del sensor de temperatura y PH y sus correspondientes valores en °C y PH respectivamente.

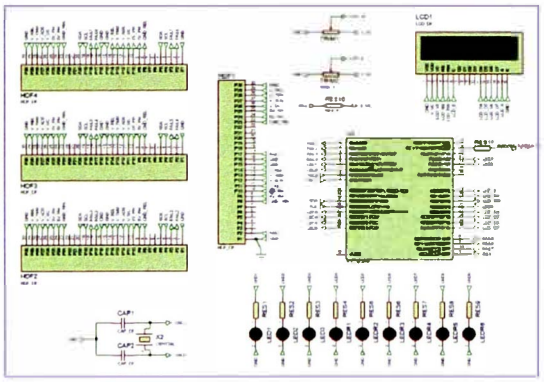


Figura 4.41. Esquema electrónico del módulo de procesamiento y presentación de información.
Fuente: elaboración propia del autor.

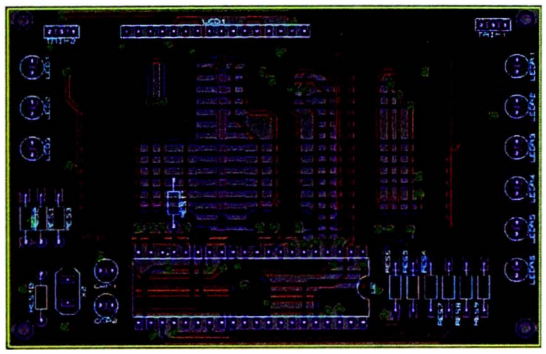


Figura 4.42. PCB Layout del módulo de procesamiento y presentación de información.
Fuente: elaboración propia del autor.

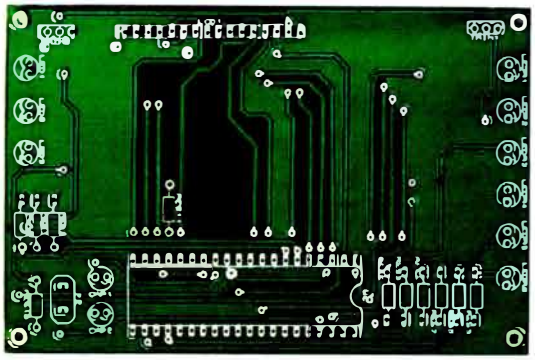


Figura 4.43. PCB físico del módulo de procesamiento y presentación de información.
Fuente: elaboración propia del autor.

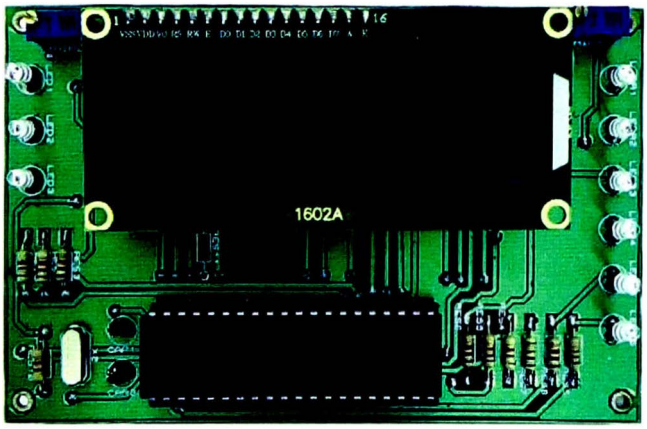


Figura 4.44. Tarjeta electrónica del módulo de procesamiento y presentación de información.
Fuente: elaboración propia del autor.

4.3.4.3.3 Módulo de fuente de alimentación

La fuente de alimentación modular entrega voltajes referenciales.

Para procesos críticos como es la lectura de PH se ha implementado reguladores de voltaje de alta precisión en módulo de adquisición, tratamiento y digitalización: regulador de 5.00V REF4132A50 [126], regulador de 2.048V REF3320 [127] de Texas Instruments.

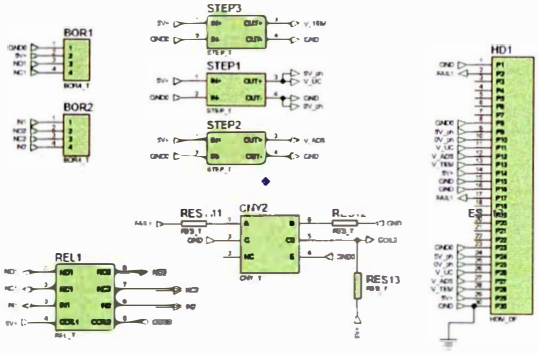


Figura 4.45. Esquema electrónico del módulo de fuente de alimentación.
Fuente: elaboración propia del autor

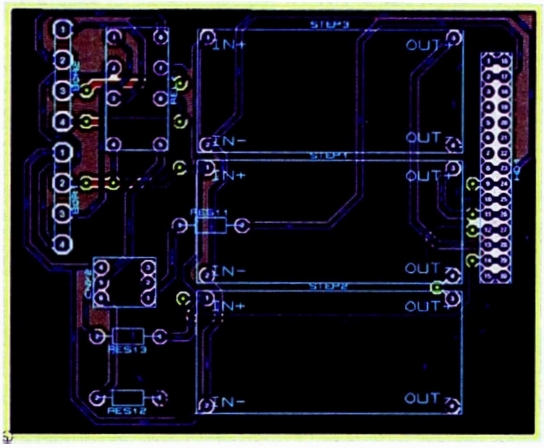


Figura 4.46. PCB Layout del módulo de fuente de alimentación.
Fuente: elaboración propia del autor.

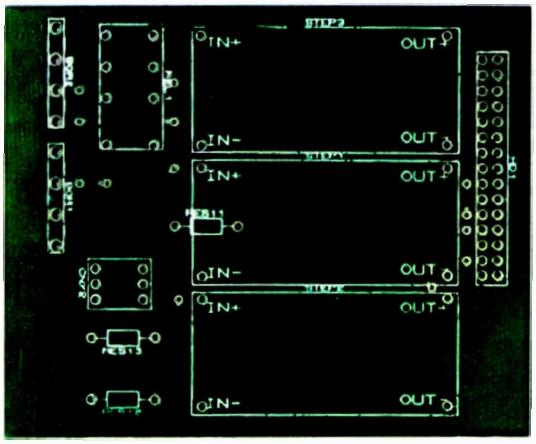


Figura 4.47. PCB física del módulo de fuente de alimentación.
Fuente: elaboración propia del autor.

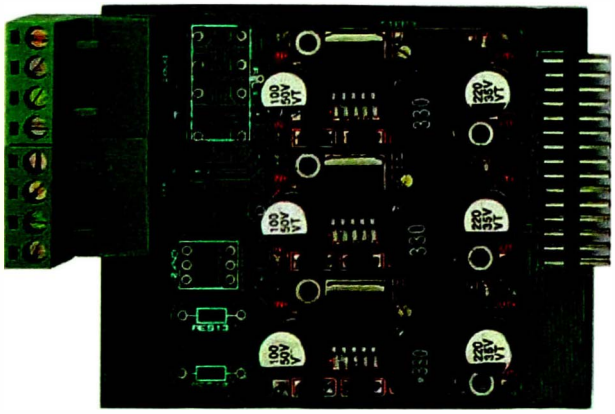


Figura 4.48. Tarjeta electrónica del módulo de fuente de alimentación.
Fuente: elaboración propia del autor.

4.3.4.4 Hardware electrónico ensamblado

En la imagen se muestra el hardware electrónico ensamblado en cual adquiere, trata, procesa y presenta la información. El procesamiento de la información se realiza mediante la RNA entrenada.

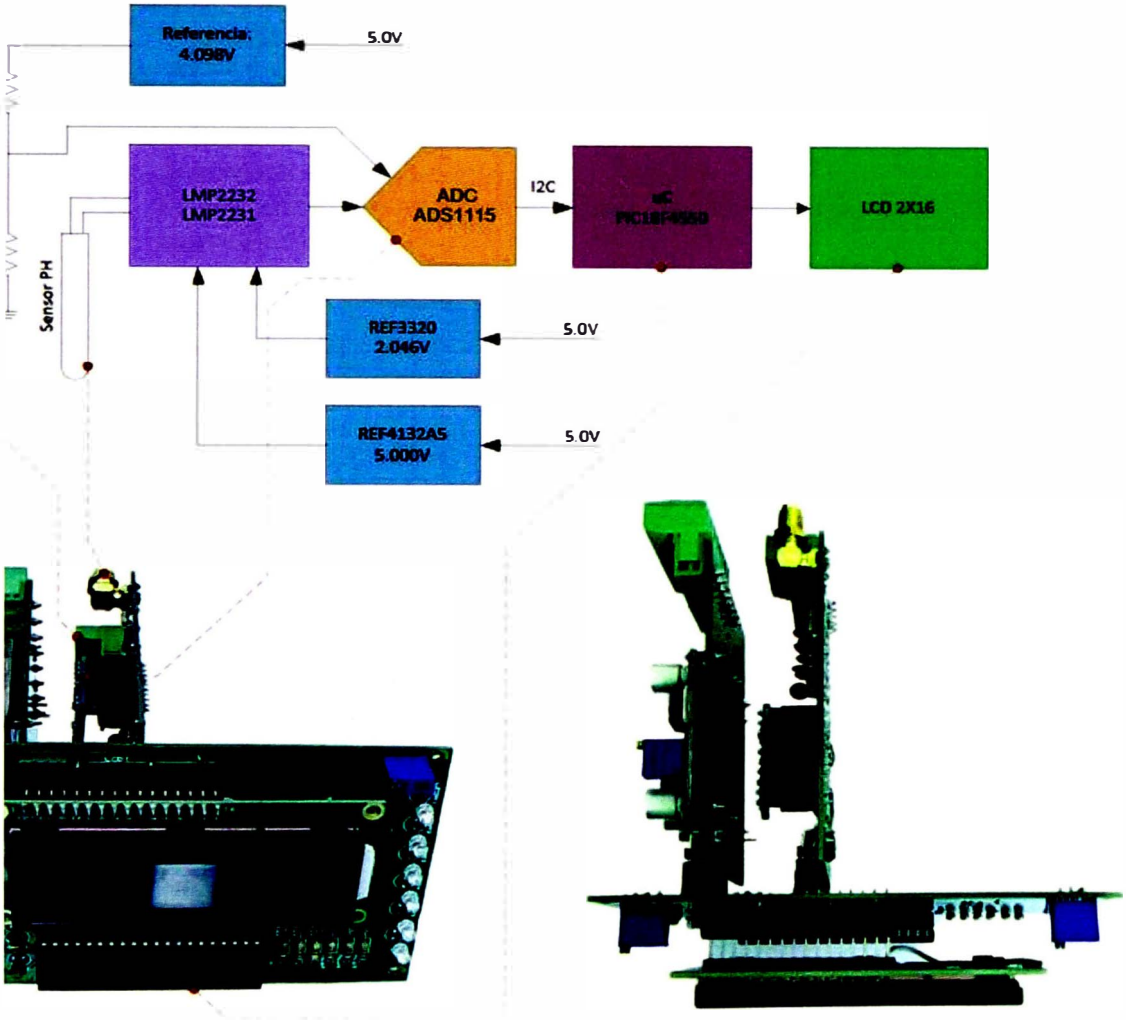


Figura 4.49. Hardware electrónico ensamblado. Vista frontal elevada (izquierda), vista planta superior (derecha).
Fuente: elaboración propia del autor.

4.3.5 Algoritmo para la lectura de PH mediante PIC18F4550

Diagrama de flujo del algoritmo para la lectura de PH programado en el microcontrolador PIC18F4550. El código de instrucciones se muestra en el Anexo K.

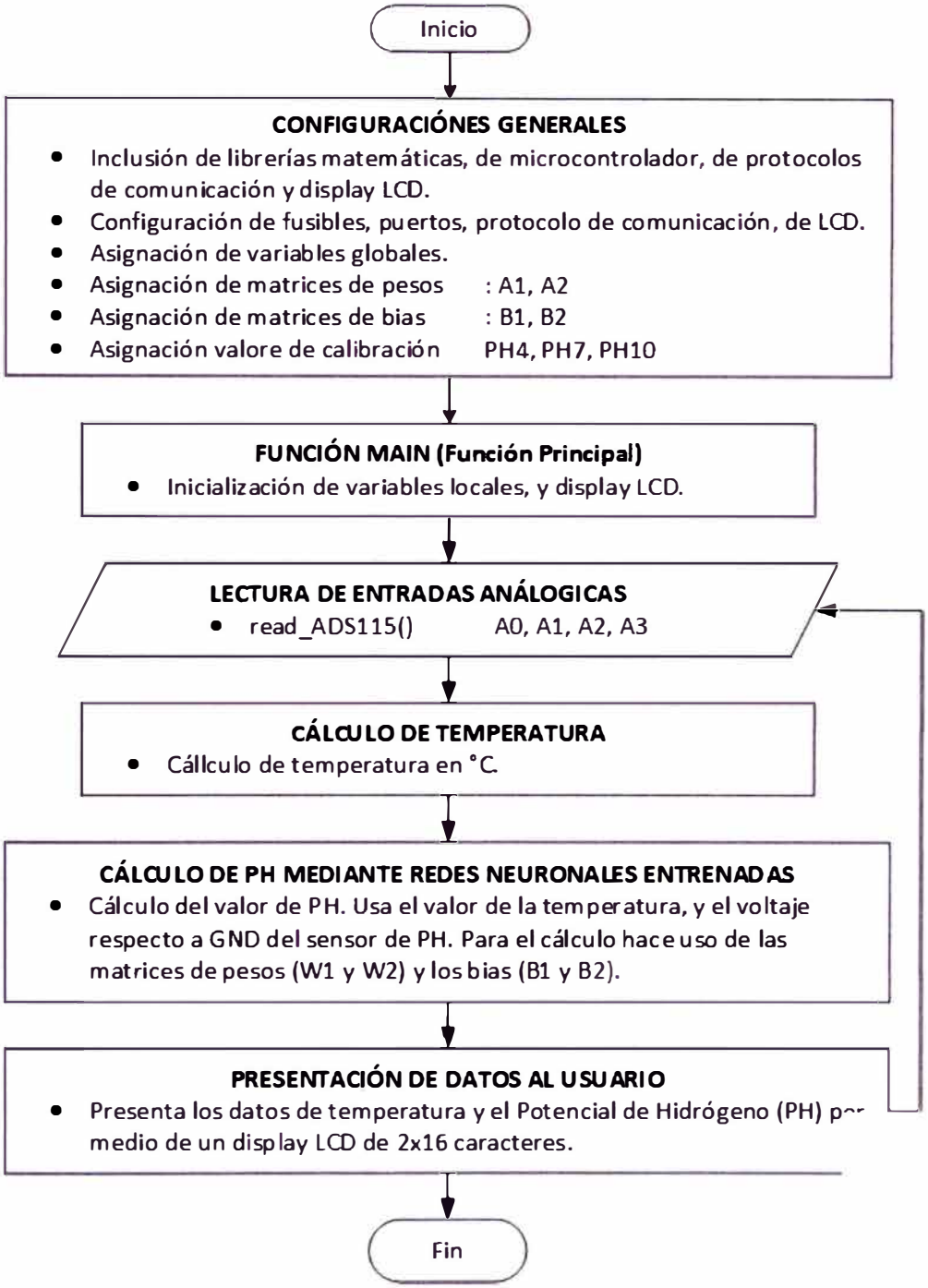


Figura 4.50. Diagrama de flujo del software de adquisición de datos, procesamiento y presentación - PIC18F4550.

CAPITULO V

ANÁLISIS DE RESULADOS

5.1 Análisis Post-Entrenamiento de la red neuronal artificial

En la **¡Error! No se encuentra el origen de la referencia.** se muestra los resultados de entrenamientos para diferente número de neuronas en la capa oculta S^1.

Tabla 6. Resultados de entrenamiento de la red neuronal artificial para diferente número de neuronas en la capa oculta, (fuente: elaboración propia del autor).

S^1	ssE	ssX	gamma γ	error máx.	error med	S^1 (γ)
15	0.5784	16915	113.53	0.3682	0.0226	14.07
20	0.2667	15592	153.94	0.2151	0.0154	19.12
25	0.2003	96166	186.94	0.1574	0.0134	23.24
30	0.1803	29047	224.90	0.1612	0.0128	27.99
35	0.1758	39541	264.69	0.1289	0.0125	32.96
40	0.1855	22701	302.44	0.1750	0.0127	37.68

donde:

S^1 : Número de neuronas en la primera capa.

ssE : Suma cuadrática de errores.

ssX : Suma cuadrática de parámetros (pesos y bias de todas las capas).

γ : Número efectivo de parámetros que es usado por la RNA.

error máx.: Error máximo (durante el testeo de la RNA, mas no de entrenamiento).

error med.: Error medio (durante el testeo de la RNA, mas no de entrenamiento).

$S^1(\gamma)$: número de neuronas en la capa oculta recomendado por el algoritmo.

De los resultados de la **¡Error! No se encuentra el origen de la referencia.**, se tiene que para $S^1 = 35$ la suma cuadrática de errores (ssE), el error máximo (error máx.) y

el error medio (error med.) son menores al del resto de pruebas con diferentes neuronas en capa oculta S^1.

Al analizar el número efectivo de parámetros γ, se tiene:

$$\gamma_{S^1=35} = 32.96, \qquad (5.1)$$

lo que indica que no se está utilizando en gran medida todos los parámetros de la RNA para $S^1 = 35$, y que seria suficiente con $S^1 = 33$. Si embargo la diferencia es solo de 2 neurona y por lo tanto aceptable. Adicionalmente se observa que al utilizar una cantidad de neuronas inferior a 35 la suma de errores cuadráticos aumenta.

Por otro lado, la cantidad de neuronas en la capa oculta ($S^1 = 35$) no es la menor, esto implica que es necesario un mayor poder de computo para realizar los cálculos para S^1 que para valores de S^1 menores que 35. Si embargo actualmente los microcontroladores tienen gran capacidad de cálculo con bajo consumo de energía, por lo que no presentaría dificultad operar una matriz de 35x6 elementos (6, es la cantidad de entradas a la RNA).

Por lo tanto, se elige la RNA con $S^1 = 35$ en la capa oculta.

En la Figura 5.1[29] se muestra la gráfica de la suma de los errores cuadráticos, en ella se observa que es suave, estable y converge a 0.17581. En la Figura 5.2 se observa que la suma de parámetros cuadráticos también logra estabilizarse, lo que indica que la variación del valor numérico de los parámetros (pesos y bias) en cada época es mínima. De la misma manera en la Figura 5.3 se muestra la gráfica del número efectivo de parámetros de la RNA en cada época de entrenamiento, y se puede observar que logra estabilizarse rápidamente.

[29] En esta Figura solo se grafica el tramo de 0 a 300 valores, ya que, si se grafica la totalidad de los datos, no se logra visualizar apropiadamente la curva, ya que los valores máximos (al inicio) y mínimos al final son 82987 y 0.17581 respectivamente.

Figura 5.1. Suma de errores cuadráticos del conjunto de datos de entrenamiento, (fuente: gráfico obtenido al ejecutarse el algoritmo de entrenamiento, elaboración propia del autor).

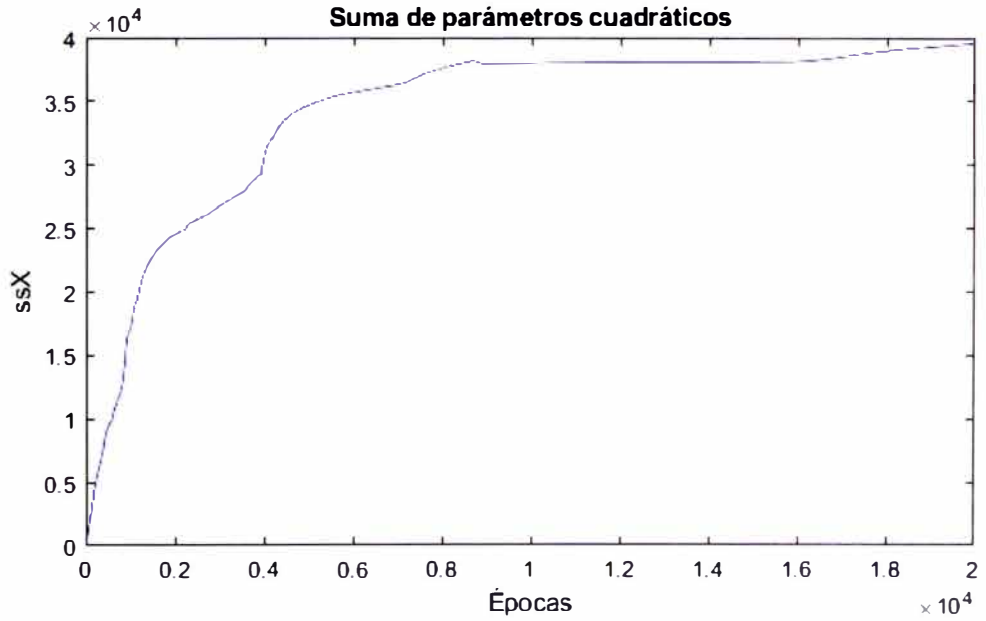


Figura 5.2. Suma de parámetros cuadráticos, (fuente: gráfico obtenido al ejecutarse el algoritmo de entrenamiento, elaboración propia del autor).

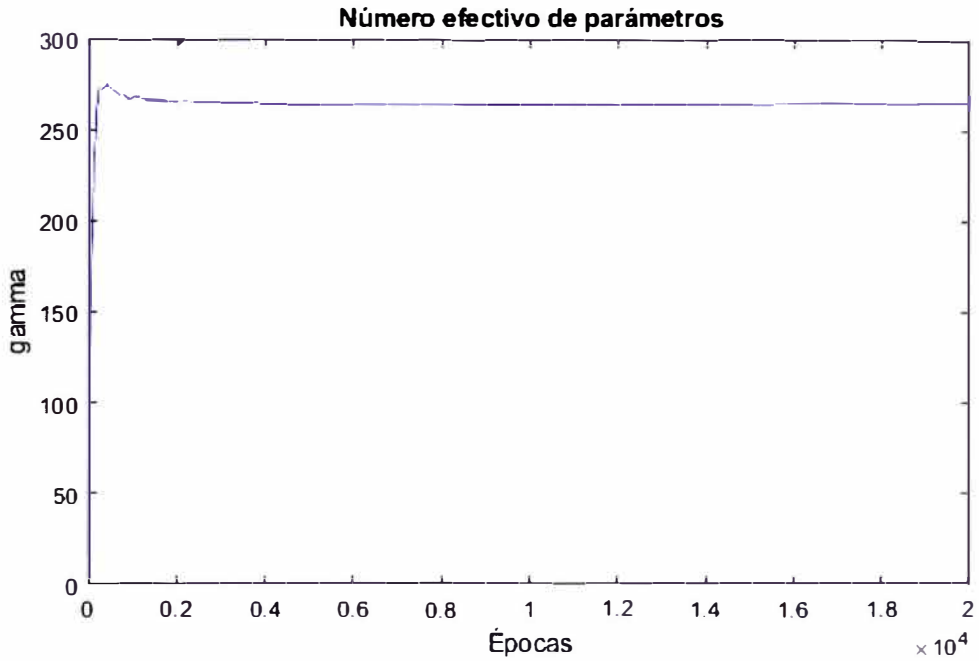


Figura 5.3. Numero efectivo de parámetros, (fuente: gráfico obtenido al ejecutarse el algoritmo de entrenamiento, elaboración propia del autor).

5.1.1 Análisis comparativo de objetivos (targets) y salidas del set de entrenamiento

En la Figura 5.4 se observa la gráfica de los valores objetivos (targets) o reales en color rojo (...+...) del conjunto de datos de testeo, los valores predichos por la RNA entrenada respecto a los valores de las entradas del conjunto de datos de testeo se muestran en color azul (- o -). En la gráfica solo se muestra 20 valores de los 4985 (esto es el 15% del total de muestras tomadas) valores de testeo con el fin de mostrar visualmente la exactitud con la que la RNA puede predecir (es decir generaliza bien, esto será más o menos concluyente con el análisis que se hará en las secciones siguientes). Recordar que el conjunto de datos para el testeo no forma parte del entrenamiento de la RNA. La Figura 5.4 se obtiene al ejecutar el script *targets_vs_network_output.m* el cual se detalla en el Anexo F.

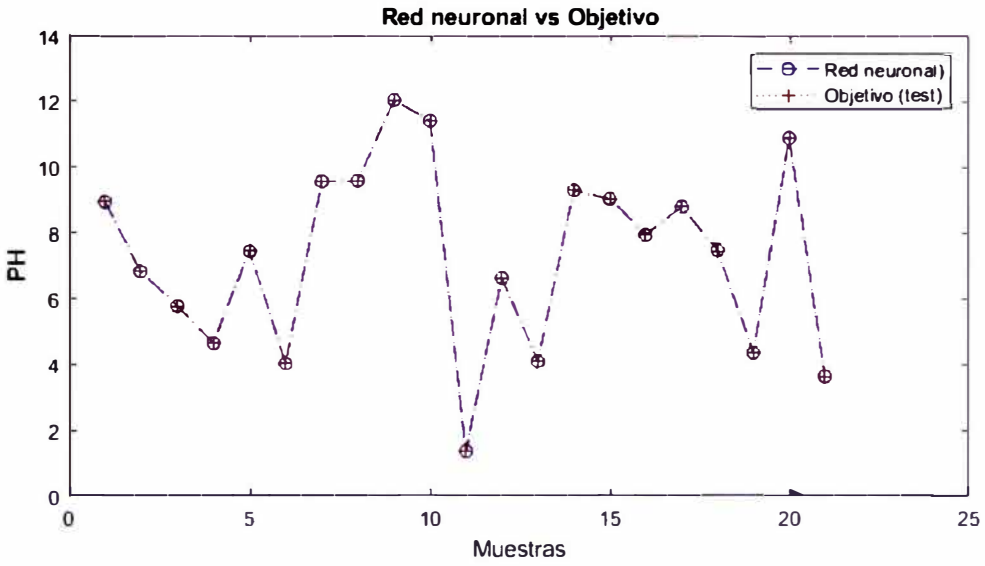


Figura 5.4. Gráfica de los valores objetivo (target) y los valores de salida correspondientes a las entradas del conjunto de testeo, (fuente: gráfico obtenido al ejecutarse el algoritmo de testeo comparativo entre objetivos y salidas predichas por la RNA, elaboración propia del autor).

5.1.2 Análisis de la Regresión lineal

Para el análisis de la regresión lineal se ejecuta el script *regresión_v1.m*, el cual se detalla en el Anexo G.

Análisis:

En la Figura 5.5 se observa la regresión lineal entre los objetivos (targets) y las salidas de la red correspondientes al conjunto de datos de testeo.

La linea azul ( ) representa la regresión lineal, la linea en segmentos de color rojo (--) representa el ajuste perfecto (línea a 45°), y los círculos en color negro (o) representa los datos.

De la gráfica de la Figura 5.5 se puede observar que el ajuste es bastante bueno, por lo tanto, la RNA ha realizado un buen entrenamiento y es capaz de generalizar bastante bien.

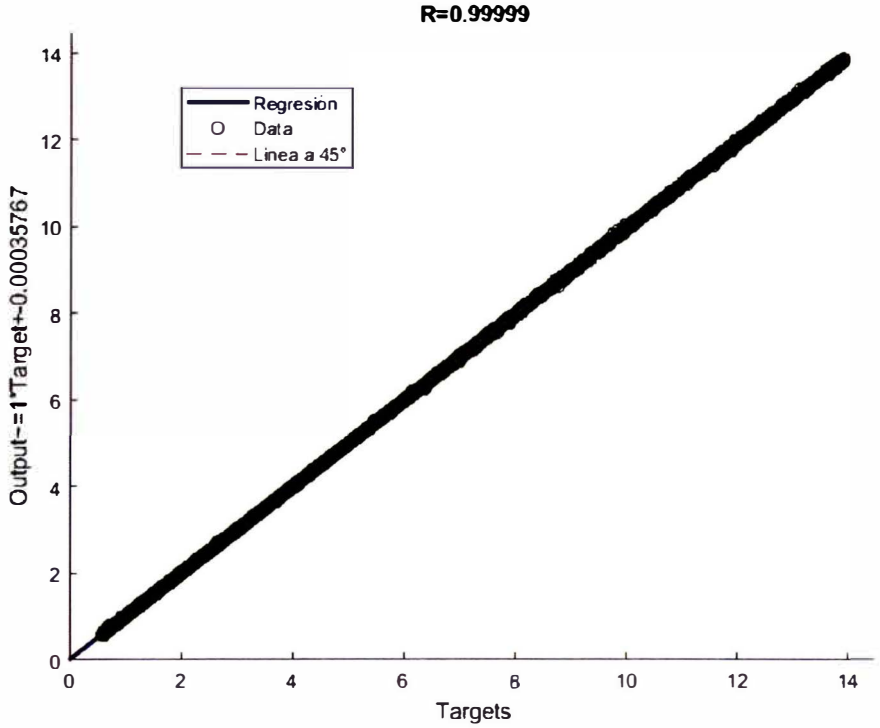


Figura 5.5. Regresión lineal de los objetivos y las salidas de la RNA correspondientes al conjunto de datos de testeo de la RNA, (fuente: gráfico obtenido al ejecutarse el algoritmo de análisis de regresión lineal, elaboración propia del autor).

En la Figura 5.5 debido a que la cantidad de datos de testeo es 4986, al representarlo gráficamente se ve muy condensado. En la Figura 5.6 se ve con más detalle al considerar solo una pequeña parte de los datos de testeo.

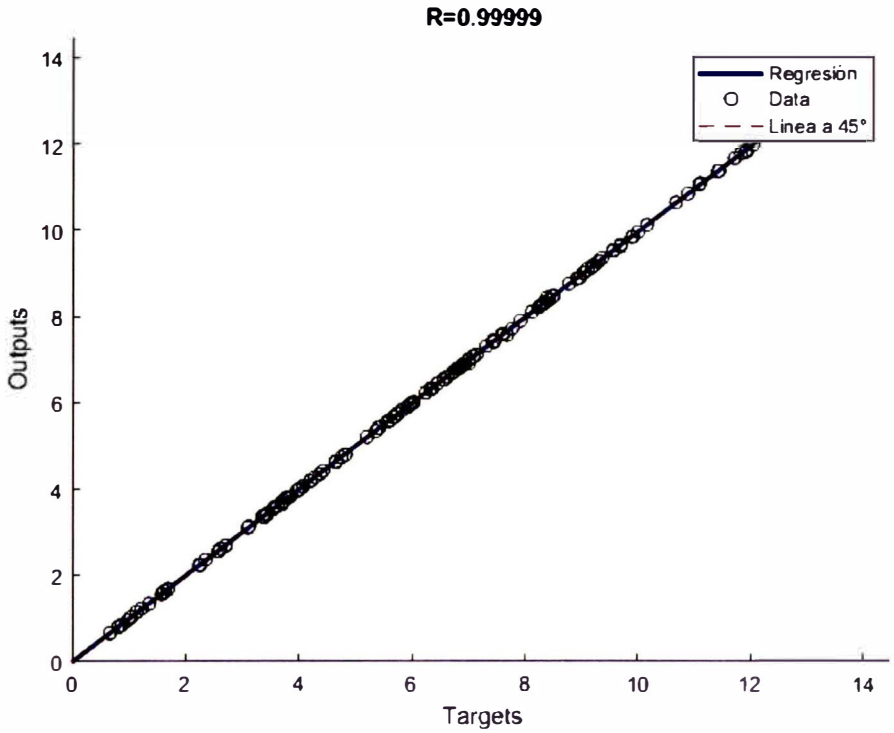


Figura 5.6. Regresión lineal con solo 150 datos del conjunto de datos de testeo de la RNA, (fuente: gráfico obtenido al ejecutarse el algoritmo de análisis de regresión lineal, elaboración propia del autor).

5.1.3 Análisis del coeficiente de correlación y coeficiente de determinación

Al ejecutar el script *regresión_v1.m* tenemos que:

$R = 0.99999$: coeficiente de correlación.

$R^2 = 0.99998$: coeficiente de determinación.

Se debe esperar que el valor de R debe ser cercano a 1, si la RNA es capaz de generalizar bien.

Cuando los valores R o R^2 son significativamente menores que 1, entonces la RNA no ha hecho un buen trabajo al ajustar la función subyacente, [131].

De los resultados tenemos se tiene que $R = 0.99999$ y $R^2 = 0.99998$, son valores muy cercanos a 1, y tomando en cuenta que este análisis es sobre el conjunto de datos de

testeo (datos que no se utilizan durante el entrenamiento de la RNA), se puede afirmar que la RNA generaliza bien (no existe sobre entrenamiento).

5.1.4 Histograma de errores de testeo de la red neuronal artificial

Otra herramienta que puede identificar valores atípicos es un histograma de los errores, como se muestra en la Figura 5.7. El eje y representa el número de errores que se encuentran dentro de cada intervalo en el eje x, [131]. Para obtener la Figura 5.7 ejecutamos el escript *histogram_v1.m* el cual se detalla en el Anexo H.

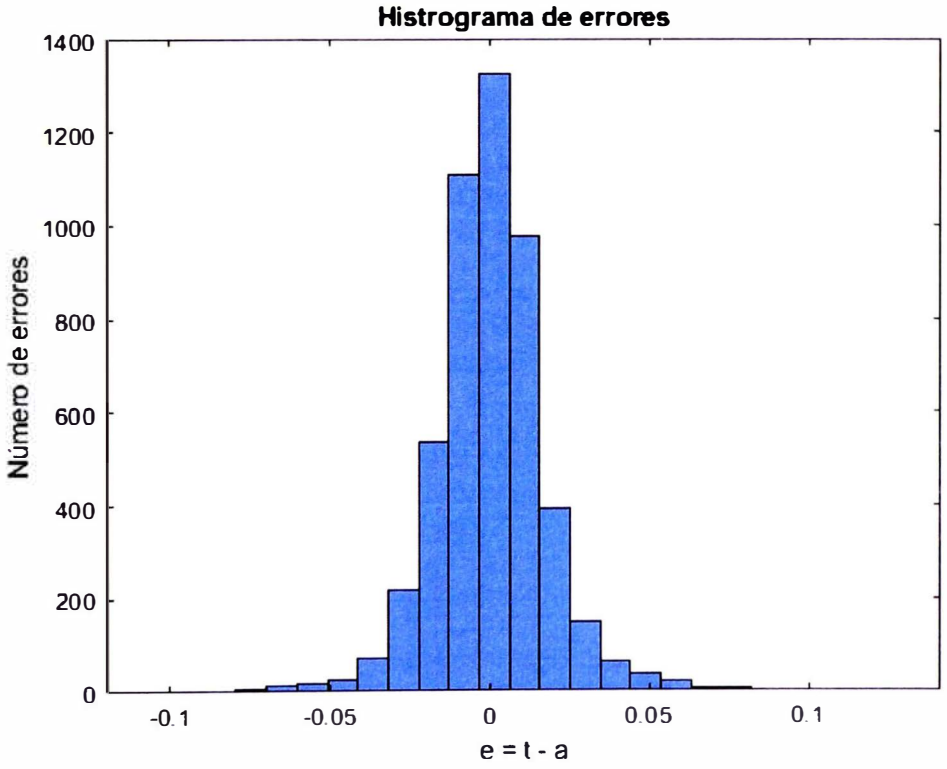


Figura 5.7. Histograma de errores, (fuente: gráfico obtenido al ejecutarse el algoritmo de histograma de errores de testeo de la RNA, elaboración propia del autor).

En la Figura 5.7 se puede ver que no hay valores atípicos, por lo que se puede afirmar que la RNA generaliza bien.

5.1.5 Parámetros de la red neuronal artificial entrenada (pesos y bias)

Los parámetros, pesos y bias, para $S^1 = 35$, de los resultados analizados en las secciones anteriores, son:

$$W^1 = \begin{bmatrix} 0.1278 & 50.5511 & -1.9148 & -18.3453 & -0.0146 & -0.1483 \\ 0.3466 & 0.1352 & 14.9281 & 16.5050 & -27.7630 & 8.6475 \\ -0.0425 & 31.1630 & 0.3771 & -12.9578 & 0.0959 & 0.0455 \\ 0.0176 & -10.5723 & -0.0657 & 4.1926 & 0.0803 & -0.0267 \\ -1.4615 & -13.2632 & 19.9811 & -14.5734 & 0.0219 & 1.4109 \\ -1.4598 & -11.2829 & 20.0665 & -15.4123 & 0.0445 & 1.4086 \\ -0.0377 & 30.2300 & 0.3365 & -12.5550 & 0.1060 & 0.0404 \\ -0.6087 & -14.1800 & 8.3316 & -2.8006 & 0.0530 & 0.6050 \\ 0.0321 & 2.4931 & 0.0351 & -1.5990 & 0.4206 & -0.0230 \\ 0.0971 & 51.3997 & -1.5295 & -19.0565 & -0.0111 & -0.1066 \\ 0.7350 & 0.1619 & 42.0159 & 24.6636 & -51.4476 & 15.0023 \\ -0.5719 & -23.4402 & 7.7899 & 1.9213 & -0.0060 & 0.5762 \\ -0.5552 & -0.0545 & -17.5441 & -21.9076 & 37.6579 & -10.9869 \\ 1.2816 & 13.5257 & -17.5366 & 12.0484 & 0.0018 & -1.2346 \\ 0.8969 & -10.0127 & -0.0382 & -7.7902 & 12.0083 & -0.9232 \\ -0.2595 & 36.5347 & 1.1438 & -13.7672 & -1.7633 & 0.1885 \\ 0.2284 & -9.6622 & -0.0083 & 0.9023 & 3.2756 & -0.2200 \\ -0.0049 & -10.3587 & -0.0368 & 4.3440 & -0.1864 & -0.0024 \\ -0.2622 & -6.3975 & -0.3562 & 6.9329 & -3.3222 & 0.1890 \\ -1.2411 & -13.9400 & 16.9674 & -11.3339 & -0.0029 & 1.1981 \\ 0.5581 & 23.5584 & -7.6146 & -2.1385 & 0.0076 & -0.5636 \\ 0.0096 & -47.1382 & 0.0180 & 19.0437 & -0.1831 & -0.0145 \\ -0.0345 & 31.1563 & 0.2954 & -12.8962 & 0.1134 & 0.0358 \\ 0.1083 & 50.1973 & -1.6612 & -18.4543 & -0.0078 & -0.1223 \\ 1.3718 & 8.1254 & -20.3831 & 16.8605 & -0.0766 & -1.3217 \\ 0.0338 & -10.5981 & -0.1044 & 4.0657 & 0.2541 & -0.0433 \\ 0.4811 & -8.2137 & -0.0175 & -3.1408 & 6.4474 & -0.4725 \\ 1.1578 & 17.8651 & -15.6786 & 8.5982 & -0.0677 & -1.1469 \\ -1.4222 & -9.4134 & 20.0580 & -16.0836 & 0.0638 & 1.3708 \\ -0.5494 & -23.8601 & 7.5059 & 2.3692 & -0.0105 & 0.5561 \\ 0.0394 & 2.7979 & -0.0233 & -1.7149 & 0.4539 & -0.0309 \\ 0.0694 & -11.4671 & -0.1000 & 3.9346 & 0.7123 & -0.0814 \\ -0.2566 & -6.4132 & -0.3492 & 6.8509 & -3.2541 & 0.1851 \\ 0.0886 & -11.1696 & -0.1206 & 3.5939 & 0.9463 & -0.0999 \\ -0.2270 & 9.4612 & 0.0131 & -0.8476 & -3.2371 & 0.2193 \end{bmatrix} \quad (5.2)$$

$$B^1 = \begin{bmatrix} 0.0150 \\ 6.0360 \\ -0.2143 \\ 0.0582 \\ -4.8836 \\ -4.6456 \\ -0.0106 \\ -1.9868 \\ 0.8452 \\ 0.1892 \\ 8.8842 \\ -2.0763 \\ -8.5620 \\ 4.5311 \\ 3.2044 \\ -1.7852 \\ 1.7957 \\ 0.2106 \\ -0.4093 \\ -4.1140 \\ 2.1272 \\ 0.0581 \\ 0.2181 \\ 0.0931 \\ 4.5647 \\ -0.1806 \\ 1.6870 \\ 3.2081 \\ -4.5031 \\ -2.1943 \\ 0.9794 \\ 0.0726 \\ -0.4245 \\ -0.0352 \\ -1.7423 \end{bmatrix} \quad (5.3)$$

$$W^2 = [-8.5040 \quad -0.0120 \quad -14.4388 \quad 30.7739 \quad 25.2449 \quad -26.4490 \quad 29.5620 \quad 0.1264 \quad 2.9620 \quad -8.6957 \quad 0.0026 \quad -4.9377 \quad -0.0085 \quad 15.1675 \quad 0.4662 \quad -0.0349 \quad -2.1302 \quad -11.9549 \quad -1.8770 \quad 4.0064 \quad -9.9959 \quad 2.3842 \quad -13.3509 \quad 18.4209 \quad 6.1574 \quad -12.1440 \quad 0.1784 \quad -0.0787 \quad 17.4323 \quad -4.9572 \quad -2.7288 \quad -14.4982 \quad 1.9453 \quad 8.7814 \quad -2.3425] \quad (5.4)$$

$$B^2 = [-1.6597] \quad (5.5)$$

5.2 Análisis de resultados de hardware y software

El programa de software desarrollado (Anexo K) y programado en el microcontrolador PIC16F4550 calcula el nivel de PH y Temperatura mediante la Red Neuronal Artificial entrenada.

Para la validación del sistema hardware-software (previamente calibrado con sus propias soluciones patrones, Figura 5.8, Figura 5.9, Figura 5.10) desarrollado se utilizó un dispositivo medidor de PH de fábrica (previamente calibrado con sus propias soluciones patrones, Figura 5.11) para comparativo.

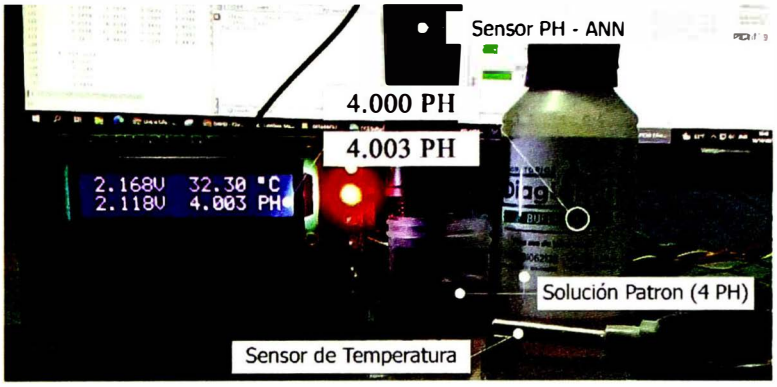


Figura 5.8. Calibración del dispositivo sensor de PH desarrollado. Punto de calibración 4.0 PH.
Fuente: Elaboración propia del autor.

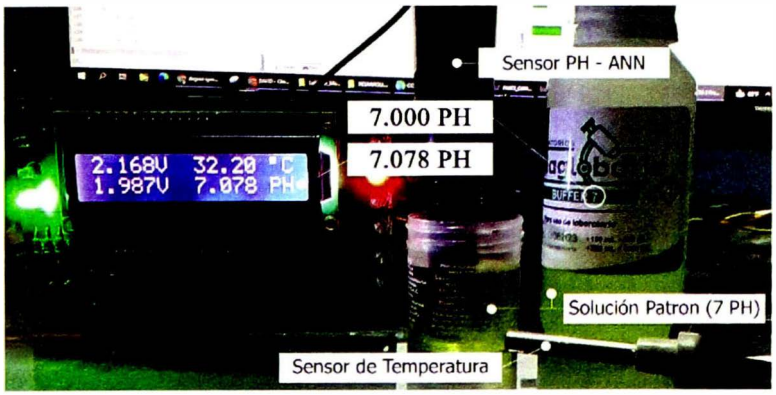


Figura 5.9. Calibración del dispositivo sensor de PH desarrollado. Punto de calibración 7.0 PH.
Fuente: Elaboración propia del autor.

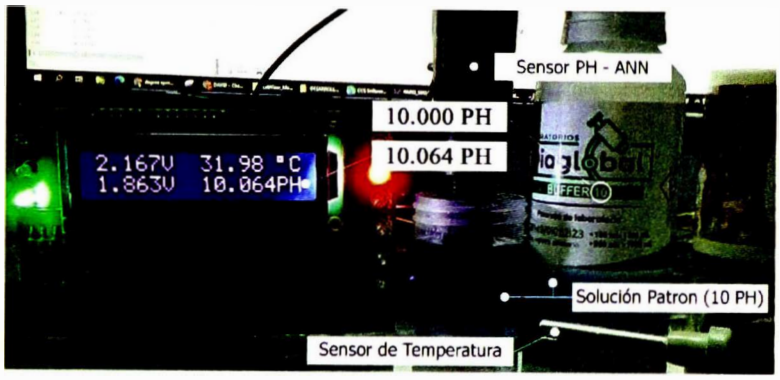


Figura 5.10. Calibración del dispositivo sensor de PH desarrollado. Punto de calibración 10.0 PH.
Fuente: Elaboración propia del autor.

Figura 5.11. Calibración del dispositivo sensor de PH utilizado para validar la lectura del dispositivo sensor desarrollado. Punto de calibración 6.820 PH.
Fuente: Elaboración propia del autor.

5.2.1 Análisis de resultados de medición de grado de PH

Para validar el dispositivo de medición de PH se realizó la medición directa del grado de PH de algunas sustancias y se comparó las mediciones con un segundo sensor de medición de PH (dispositivo fabricado en serie).

Se midió el grado de PH de Yogurt GLORIA. El grado de PH leído por el dispositivo construido fue de 3.452, y el PH leído por el dispositivo de fabrica fue 3.47. El error de PH medido es de 0.018.

Una segunda validación se realizó midiendo el grado de PH de una muestra de gaseosa Inca Kola. El grado de PH leído por el dispositivo construido fue de 2.157, y el PH leído por el dispositivo de fabrica fue de 2.16. El error de PH medido es de 0.003.

Una tercera validación se realizó midiendo el grado de PH de una muestra de Leche GLORIA. El grado de PH leído por el dispositivo construido fue de 6.123, y el PH leído por el dispositivo de fabrica fue de 6.180. El error de PH medido es de 0.057.

El error promedio de PH de las tres mediciones es de 0.026. Este error medio es aceptable para la mayoría de las aplicaciones industriales. En este caso, el error promedio de 0.026 es aceptable para las soluciones que ofrece la empresa solicitante.

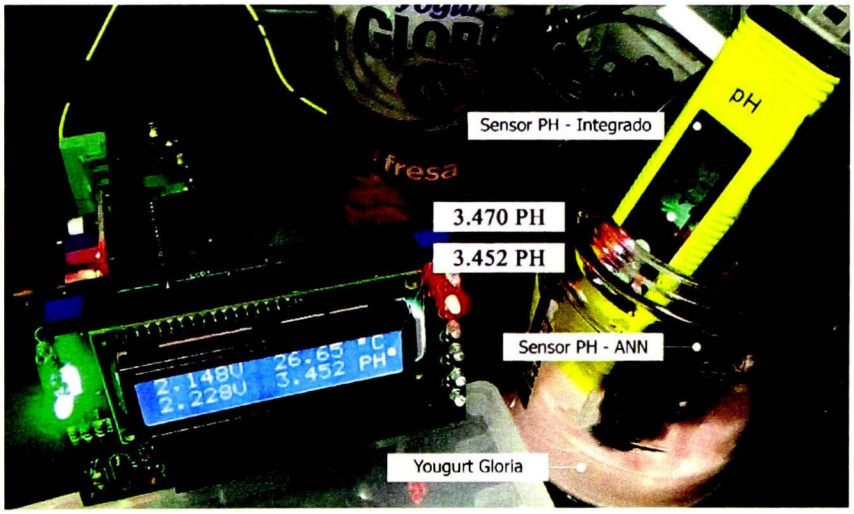


Figura 5.12. Medición del PH de Yogurt Gloria con el dispositivo sensor construido y un dispositivo sensor de fábrica. Error máximo 0.018PH.
Fuente: Elaboración propia del autor.

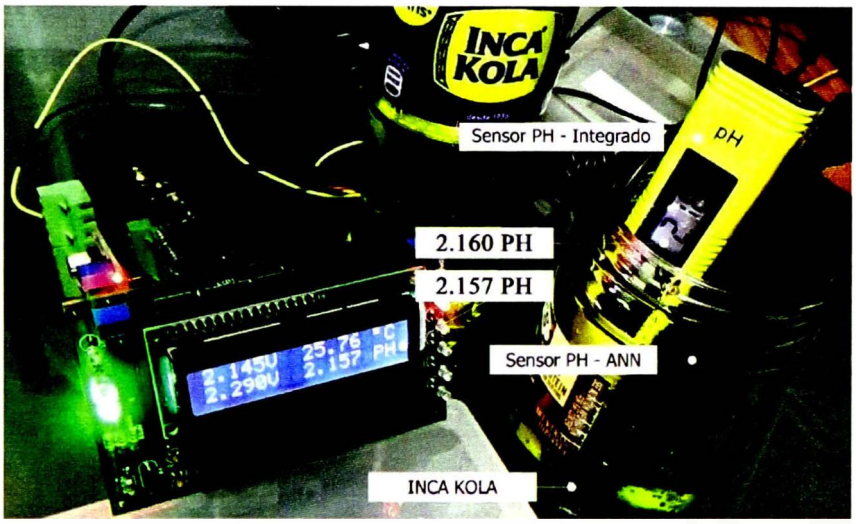


Figura 5.13. Medición del PH de una muestra de Inca Kola con el dispositivo sensor construido y un dispositivo sensor de fábrica. Error máximo 0.003PH. Fuente: Elaboración propia del autor.

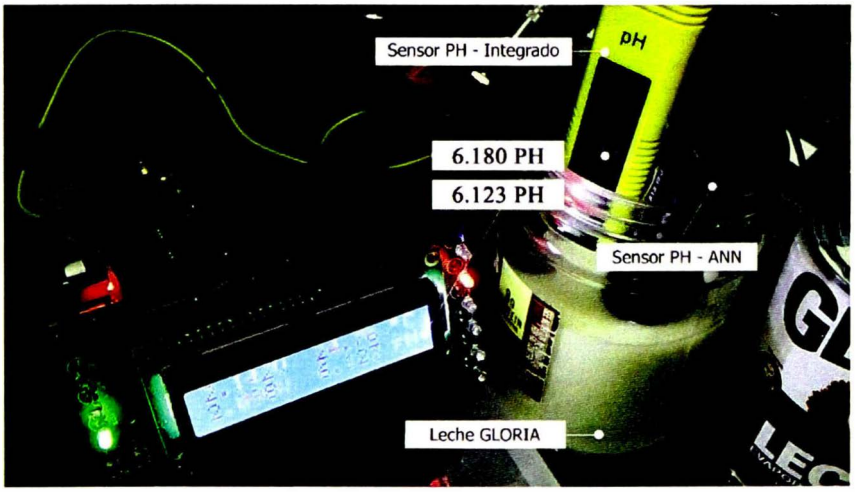


Figura 5.14. Medición del PH de una muestra de leche Gloria con el dispositivo sensor construido y un dispositivo sensor de fábrica. Error máximo 0.057PH. Fuente: Elaboración propia del autor.

CONCLUSIONES

La idea principal de este trabajo es la identificación de la curva característica de un sensor de Potencial de Hidrógeno, pH. Esto es requerido por la empresa WAPOSAT con el fin de posteriormente utilizar esta curva característica en el hardware a medida que pretenden desarrollar para adaptarse a las necesidades particulares de cada uno de sus clientes. Este trabajo hace uso de la capacidad de aprendizaje, generalización, tolerancia a fallos y facilidad de implementación en microcontroladores de las redes neuronales artificiales.

Del desarrollo de este trabajo se puede concluir que:

- En relación al objetivo específico 1 "Adquirir datos de pH y temperatura, y sus voltajes correspondientes del módulo electrónico Smart Water y Waspmote para entrenar la Red Neuronal Artificial", se concluye que, ha sido posible adquirir datos de entrenamiento, tratarlos y seleccionarlos adecuadamente según los resultados analizados en la sección de post entrenamiento. Esto gracias a teorías adquisición y tratamientos de datos citadas durante el desarrollo de problema. Estos datos han sido utilizados para el entrenamiento de la Red Neuronal Artificial como se muestran en el Capítulo 5 "Análisis de resultados", por lo tanto, se acepta la hipótesis específica 1: La adquisición de datos de pH y temperatura, y sus voltajes correspondientes del módulo electrónico Smart Water y Waspmote permite entrenar la Red Neuronal Artificial.

- En relación al objetivo específico 2 "Determinar y parametrizar una arquitectura de Red Neuronal Artificial para entrenar la misma mediante un algoritmo de entrenamiento", se concluye que, ha sido posible determinar y parametrizar una

arquitectura de red neuronal artificial que puede resolver el problema, para ello se ha tomado en cuenta la cantidad de variables de entrada y de salida, y el tipo de problema. Por lo tanto, se acepta la hipótesis específica 2: La determinación y parametrización de una arquitectura de Red Neuronal Artificial permite entrenar la misma mediante un algoritmo de entrenamiento.

- En relación al objetivo específico 3 "Elegir y ejecutar un algoritmo de entrenamiento para entrenar la Red Neuronal Artificial", se concluye que, ha sido posible elegir un algoritmo de entrenamiento de la RNA que resuelva el problema tomando en cuenta un análisis de diferentes algoritmos existentes, siguiendo criterios de prácticas recomendadas y teniendo en cuenta el tipo de problema que se intenta solucionar. Al ejecutar el algoritmo se logró satisfactoriamente el entrenamiento, optimización y validación de la Red Neuronal Artificial, esto se evidencia en el capítulo V "Análisis de resultados" en la cual los diversos indicadores lo confirman. Por lo tanto, se acepta la hipótesis específica 3: La elección y ejecución de un algoritmo de entrenamiento nos permite entrenar la Red Neuronal Artificial.

- En relación al objetivo específico 4, "Diseñar y construir un equipo electrónico para ejecutar la Red Neuronal Artificial entrenada y validada en una empresa de monitoreo de la calidad del agua", se diseñó y construyó un equipo electrónico como se evidencia en el en el Capítulo 4, subtítulo 4.3.4 "Diseño y construcción de hardware electrónico". En el capítulo V, subtítulo 5.2 "Análisis de resultados de hardware y software" se evidencia que en este equipo electrónico se ejecutó y validó satisfactoriamente la Red Neuronal Artificial al

realizar mediciones de PH de diversas sustancias y comparar con un dispositivo medidor de PH de fábrica. El error promedio de PH es de 0.026, este error es aceptable para las soluciones de monitoreo que requieren las empresas solicitantes. Por lo tanto, se acepta la hipótesis específica 4, El Diseño y construcción de un equipo electrónico permite ejecutar la red neuronal artificial entrenada en una empresa de monitoreo de la calidad del agua.

- Respecto al objetivo general de "Desarrollar una Red Neuronal Artificial para identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo en tiempo real de la calidad del agua", se concluye que debido a que se ha alcanzado satisfactoriamente cada uno de los objetivos específicos, y por ende el objetivo general, se ha logrado "Desarrollar una Red Neuronal Artificial para identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo en tiempo real de la calidad del agua.". Podemos afirmar esto tomando en cuenta los resultados óptimos de los análisis realizados a los resultados del entrenamiento en la sección de post entrenamiento y validación. Por lo tanto, se acepta la hipótesis general de que "El Desarrollo de una Red Neuronal Artificial permite identificar la Curva Característica de un Sensor de Potencial de Hidrógeno PH en una empresa de monitoreo de la calidad del agua."

Como resumen de estas conclusiones se indica que cada una de las hipótesis han sido validada y por ende también la hipótesis general.

RECOMENDACIONES

Después de realizar este trabajo, podemos recomendar, en base a los procedimientos, resultados y la experiencia adquirida durante la ejecución de este trabajo.

- La RNA entrenada debe ser utilizada en el rango de 0.6 a 13.9 de pH, ya que este es el rango en el que se ha tomado datos y a la vez entrenado y validado la RNA.
- La temperatura a la que debe operar el sistema que utilice estos resultados debe trabajar entre la temperatura de 0.2°C y 80°C (80°C es la temperatura máxima que recomienda el fabricante del sensor, [145].). Aunque el entrenamiento y testeo se ha realizado para comprender el rango de 0.2°C y 95.5°C.
- El rango de calibración del sensor debe realizarse entre 15°C y 30°C. Esto quiere decir que los 4 puntos de calibración se deben obtener de la tarjeta Smart Water con las soluciones de calibración en el rango de 15° y 30°C, aunque se recomienda calibrar los sensores a 25°C, [11]. Esto no quiere decir que la RNA trabaje solo en este rango indicado. Se recuerda que la RNA opera en el rango de 0.2°C y 80°C.
- La tarjeta electrónica ha sido diseñada con salida para alarmas en caso el pH o temperatura atraviesa un punto o un rango de valores deseados, sin embargo, no fue implementado físicamente, por lo que sería un punto de partida para ampliar este trabajo.
- Los valores de pH y temperatura se muestran de manera local en la pantalla LCD del equipo electrónico construido, por lo que para ampliar este trabajo y su utilidad estos datos podrían mostrarse en un aplicativo móvil o web para su visualización de manera remota en cualquier lugar con acceso a internet.

REFERENCIA BIBLIOGRÁFICA

[1] N. M. Sanctuaries, «Florida Keys National Marine Sanctuary,» U.S. Departament of Commerce, [En línea]. Available: https://nmsfloridakeys.blob.core.windows.net/floridakeys-prod/media/archive/scisummaries/wqfaq.pdf. [Último acceso: 10 October 2019].

[2] S. d. I. F. Fernandez, «Análisis Componentes Principales,» de *Componentes Principales*, Madrid, 2011, p. 1.

[3] S. A. M. K. Schölkopf B., «Springer Link,» Springer, 09 June 2005. [En línea]. Available: https://doi.org/10.1007/BFb0020217. [Último acceso: 09 September 2019].

[4] «IEEE,» IEEE - Connected & Autonomous Vehicles, [En línea]. Available: https://site.ieee.org/connected-vehicles/ieee-connected-vechicles/connected-vehicles/. [Último acceso: 09 September 30].

[5] R. P. BUCK, S. RONDININI y A. K. COVINGTON, «Measurement of pH. Definition, standards, and procedures - pH, a single ion quantity,» de *Pure and Applied Chemistry 74, 2169–2200*, INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY, 2002, p. 2171.

[6] R. P. BUCK, S. RONDININI y A. K. COVINGTON, «Measurement of pH. Definition, standards, and procedures - Hydrogen ion activity,» de *Pure and Applied Chemistry 74, 2169–2200*, INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY, 2002, p. 2173.

[7] R. P. BUCK, S. RONDININI y A. K. COVINGTON, «Measurement of pH. Definition, standards, and procedures - Relation to SI,» de *Pure and Applied Chemistry 74, 2169–2200*, INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY, 2002, p. 2173.

[8] R. P. BUCK, S. RONDININI y A. K. COVINGTON, «Measurement of pH. Definition, standards, and procedures - Primary method of measurement,» de *Pure and Applied Chemistry 74, 2169–2200*, INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY, 2002, p. 2173.

[9] R. P. BUCK, S. RONDININI y A. K. COVINGTON, «Measurement of pH. Definition, standards, and procedures - Harned cell,» de *Pure and Applied Chemistry 74, 2169–2200*, INTERNATIONAL UNION OF PURE AND APPLIED CHEMISTRY, 2002, pp. 2173, 2174.

[10] T. O. Blueprint, «Omega,» The Omega Blueprint, 28 August 2018. [En línea]. Available: https://www.omega.com/en-us/resources/ph-meter. [Último acceso: 16 October 2019].

[11] Libelium, «Calibration procedure - Document version: v7.6 - 09/2019,» de *Smart Water Technical Guide*, Libelium Comunicaciones Distribuidas S.L., 2019, p. 47.

[12] Libelium, «Calibration procedure,» de *Smart Water Technical Guide - Document version: v7.6 - 09/2019*, Libelium Comunicaciones Distribuidas S.L., 2019, p. 48.

[13] Libelium, «Calibration procedure,» de *Smart Water Technical Guide - Document version: v7.6 - 09/2019*, Libelium Comunicaciones Distribuidas S.L., 2019, p. 49.

[14] Libelium, «Calibration procedure,» de *Smart Water Technical Guide - Document version: v7.6 - 09/2019*, Libelium Comunicaciones Distribuidas S.L., 2019, pp. 47, 48, 49.

[15] I. I. AG, «Flow chart for pH electrode selection - 12.06/I.I. - CP010C/07/en/04.10,» de *pH measurement in industrial processes*, Reinach-Switzerland, Endress+Hauser, pp. 22-37.

[16] H. Research, Hexa Research, February 2016. [En línea]. Available: https://www.hexaresearch.com/research-report/ph-meters-market?utm_source=Quora&utm_medium=Referral&utm_campaign=Tanuja. [Último acceso: 16 October 2019].

[17] J. G. W. H. Eren, «Hysteresis and Backlash,» de *Mearument, Instrumentation and Sensors*, New York, Taylor & Francis Group, 2014, pp. 4-7, 4-8.

[18] J. G. W. H. Eren, «Hysteresis and Backlash,» de *Measument, Instrumentation and Sensors*, New York, Taylor & Francis Group, 2014, pp. 4-8 - 4-8.

[19] M. Hagan y H. Demut, «Taylor Series - Vector Case,» de *Neural Network Design*, New Yotk, Martin Hagan; 2 edition, 2014, pp. 8-4 - 8-5.

[20] M. T. Hagan y H. B. Demuth, «Introduction,» de *Neural Network Design - 2nd Edition*, Oklahoma, Martin Hagan; 2 edition (September 1, 2014), 2014, pp. 1-1.

[21] J. J. Velasco, «EL DIARIO,» 26 11 2014. [En línea]. Available: https://www.eldiario.es/turing/Ramon-Llull-Ars-Magna-pensantes_0_326517940.html. [Último acceso: 15 July 2019].

[22] «Wikipedia,» 04 July 2019 (last update). [En línea]. Available: https://es.wikipedia.org/wiki/Ars_Magna_Generalis. [Último acceso: 15 July 2019].

[23] M. R. Minervino, «Obras y Protagonistas,» 08 2016. [En línea]. Available: http://www.oyp.com.ar/nueva/revistas/244/1.php?con=5. [Último acceso: 15 07 2019].

[24] W. S. MCCULLOCH y W. PITTS, «A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY,» de *Bulletin of Mathematical Biology Vol. 52, No. I/2.*, Vol. 52 ed., 1990, pp. 99-115.

[25] W. McCulloch y W. Pitts, «Springer Link,» Springer Nature, [En línea]. Available: https://doi.org/10.1007/BF02478259. [Último acceso: 14 July 2019].

[26] D. Hebb, «Summation and Learning in Perception,» de *The Organization of Behavior*, New York, JOHN WILEY and SON, Inc., 1949, p. 17.

[27] D. Hebb, «Summation and Learning in Perception,» de *The Organization of Behavior*, New York, JOHN WILEY and SON, Inc., 1949, pp. 35-37.

[28] J. Copeland, «ALAN TURING,» May 2000. [En línea]. Available: http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI/What%20is%20AI10.html. [Último acceso: 16 Julio 2019].

[29] F. Rosenblantt, The Perceptron - A Perceiving and Recognizing Automation, Buffalo, N. Y.: CORNELL AERONAUTICAL LABORATORY, INC., January, 1957.

[30] J. C. Hay, B. E. Lynch y D. R. Smith, MARK I PERCEPTRON OPERATOR'S MANUAL, Buffalo, New York: CORNELL AERONAUTICAL LABORATORY, INC., 15 February 1960.

[31] «SEIDENBERG SCHOOL OF CSIS,» [En línea]. Available: http://csis.pace.edu/~ctappert/srd2011/rosenblatt-contributions.htm. [Último acceso: 16 July 2019].

[32] K. Steinbuch, «Die Lernmatrix,» Kybernetik , 1961, pp. 36-45.

[33] B. Widrow y M. E. Hoff, «Adaptive Switching Circuits,» de *1960 IRE WESCON Convention Record*, IRE, 1960, pp. 96-104.

[34] J. S. Albus, «A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC).,» de *Journal of Dynamic Systems, Measurement, and Control*, ASME The American Society Of Mechanical Engineers, 1975, pp. 220-227.

[35] K. Fukushima, «A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,» Tokyo, NHK Broadcasting Science Research Laboratories, 1980, pp. 1-10.

[36] C. G.A. y G. S., «Adaptive Resonance Theory,» de *Encyclopedia of Machine Learning*, Springer, Boston, MA, 2011, pp. https://doi.org/10.1007/978-0-387-30164-8.

[37] «Scholarpedia,» [En línea]. Available: http://www.scholarpedia.org/article/Adaptive_resonance_theory. [Último acceso: September 30 2019].

[38] Kohonen, «Springer-Verlag,» Biological Cybernetics, 1982. [En línea]. Available: https://doi.org/10.1007/BF00337288. [Último acceso: 30 September 2019].

[39] J. A. FELDMAN y D. H. BALLARD, «C0nnecti0nist Models and Their Properties,» de *COGNITIVE SCIENCE 6*, New York, Computer Science Department - University of Rochester, 1982, pp. 1-50.

[40] E. Hinton y T. Sejnowski, «Boltzmann Machines - Constraint Satisfaction Networks that Learn,» Pittsburgh, Pennsylvania, Carnegie-Mellon University, Dept. of Computer Science (1984), 1984, p. 1.

[41] E. Rumelhart, E. Hinton y J. Williams, «Learning Representations by back-propagation errors,» de *NATURE VOL. 323.*, Nature - International Journal of Sience, 1986.

[42] S. Hochreiter y J. Shmidhuber, «Long Short - Term Memory,» de *Neural Computation*, 1997.

[43] L. L. -. M. I. o. Technology, « The Importance of Surviving the Single-layer Perceptron,» de *DARPA Neural Network Study - Final Report*, Lexington, Massachussetts, Massachusetts Institute of Technology, 1989, p. 24.

[44] DARPA, de *DARPA Nural Network Study - Final Report*, Lexington, Massachusetts, MIT Lincoln Laboratory, 1989, p. Chapter 1..

[45] M. Hagan y H. Demuth, «Applications,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 1-5.

[46] M. Hagan y H. Demuth, «Applications,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 1-6.

[47] M. Hagan y H. Demuth, «Applications,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 1-7.

[48] Friedrichshafen, «AIRBUS,» 16 Noviembre 2018. [En línea]. Available: https://www.airbus.com/newsroom/press-releases/en/2018/11/Crew-assistant-CIMON-successfully-completes-first-tasks-in-space.html. [Último acceso: 01 Agosto 08].

[49] N. Muscettola, P. Nayak y B. Pell, «A Remote Agen t arc hitecture,» de *Remote Agent*, NASA Ames Research Center.

[50] N. Muscettola, P. Nayak y B. Pell, «A Remote Agent architecture,» de *Remote Agent: T o Boldly Go Where No AI System Has Gone Before* , NASA Ames Researc h Cen ter, p. 9.

[51] Mitsubishe Motors, 22 May 2017. [En línea]. Available: https://www.mitsubishi-motors.com/en/newsrelease/2017/detail1057.html. [Último acceso: 04 09 2019].

[52] «www.nvidia.com,» Nvidia Corporation, [En línea]. Available: https://www.nvidia.com/en-us/self-driving-cars/ap2x/. [Último acceso: 04 09 2019].

[53] «Nvidia,» Nvidia Corporation, [En línea]. Available: https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/. [Último acceso: 04 09 2019].

[54] «PC Magazine,» Ziff Davis, [En línea]. Available: https://www.pcmag.com/encyclopedia/term/65738/self-driving-car. [Último acceso: 04 09 2019].

[55] «UCSUSA,» The Union of Concerned Scientists, 21 February 2018. [En línea]. Available: https://www.ucsusa.org/clean-vehicles/how-self-driving-cars-work. [Último acceso: 04 09 2019].

[56] Tesla Motors, Inc., 19 October 2016. [En línea]. Available: https://www.tesla.com/BLOG/ALL-TESLA-CARS-BEING-PRODUCED-NOW-HAVE-FULL-SELF-DRIVING-HARDWARE. [Último acceso: 04 09 2019].

[57] Tesla Motors, Inc., [En línea]. Available: https://www.tesla.com/support/autopilot. [Último acceso: 04 09 2019].

[58] Tesla Motors, Inc., 22 April 2019. [En línea]. Available: https://www.youtube.com/watch?v=tlThdr3O5Qo. [Último acceso: 04 09 2019].

[59] TESLA MOTORS, [En línea]. Available: https://www.tesla.com/BLOG/ALL-TESLA-CARS-BEING-PRODUCED-NOW-HAVE-FULL-SELF-DRIVING-HARDWARE. [Último acceso: 04 09 2019].

[60] «Ignite,» [En línea]. Available: https://igniteoutsourcing.com/automotive/artificial-intelligence-in-automotive-industry/. [Último acceso: 04 09 2019].

[61] «IEE Innovation at Work,» IEEE , [En línea]. Available: https://innovationatwork.ieee.org/how-will-car-maintenance-evolve-autonomous-vehicles-era/. [Último acceso: 04 09 2019].

[62] CARFIT, [En línea]. Available: https://car.fit/wp-content/uploads/CARFIT_MediaClipping_20180124.pdf. [Último acceso: 04 09 2019].

[63] «Microsoft Corporation,» Microsoft Asia , 27 May 2019. [En línea]. Available: https://www.youtube.com/watch?time_continue=1&v=vRrkHVdYjr8. [Último acceso: 05 09 2019].

[64] «ICICI Lombard,» ICICI Lombard GIC Ltd., [En línea]. Available: https://www.icicilombard.com/mobile-self-inspection. [Último acceso: 05 09 2019].

[65] «CIO,» CIO, [En línea]. Available: https://www.cio.com/article/3355239/making-automotive-manufacturing-smarter-with-ai.html. [Último acceso: 05 09 2019].

[66] «H2O AI,» H2O AI, [En línea]. Available: https://www.h2o.ai/manufacturing/?gclid=Cj0KCQjw5MLrBRCIARIsAPG0WGzPFTITKrhwJ8dFDI2F8IhUn8d6TvbDDWFOjuLeKafBIQoGnL4_KIUaAm5qEALw_wcB. [Último acceso: 05 09 2019].

[67] H. Martin y D. Howard., «Single-Input Neuron,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-3.

[68] M. Hagan y H. Demuth, «Neuron Model and Network Architecture,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 2-1.

[69] H. Martin y D. Howard., «Transfer Functions,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-4.

[70] H. Martin y D. Howard., «Transfer Functions,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-5.

[71] H. Martin y D. Howard., «Transfer Functions,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-6.

[72] M. Hagan y H. Demuth, «Multiple-Input Neuron,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-7.

[73] H. Martin y D. Howard., «Multiple-Input Neuron,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-8.

[74] H. Martin y D. Howard., «A Layer of Neurons,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-9.

[75] H. Martin y D. Howard., «A Layer of Neurons,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-10.

[76] H. Martin y D. Howard., «Multiple Layers of Neurons,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-11.

[77] H. Martin y D. Howard., «Multiple Layers of Neurons,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 2-12.

[78] M. Hagan y H. Demuth, «Multilayer Perceptrons,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 11-2, 11-3.

[79] M. Hagan y H. Demuth, «Multilayer Perceptrons,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 11-3.

[80] M. Hagan y H. Demuth, «Function Approximation,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 11-5.

[81] H. Martin y D. Howard., «Multilayer Network,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 11-25.

[82] M. Hagan y H. Demuth, «Levenberg_Marquardt Algorithm,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 12-19 - 12-23.

[83] A. S. a. K.-R. M. Bernhard Schölkopf, de *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, Tübingen - Germany, Max-Planck-Institut, 1996, pp. 1-18.

[84] M. Hagan y H. Demuth, «Normalization,» de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 22-5 - 22-7.

[85] «Keras,» Keras, [En línea]. Available: https://keras.io/initializers/. [Último acceso: 13 September 2019].

[86] M. Hagan y H. Demuth, de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 22-13 - 22-14.

[87] M. Hagan y H. Demuth, «Weight Initialization,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 22-13.

[88] H. Martin y D. Howard., «Function Approximation,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 11-5.

[89] M. Hagan y H. Demuth, «Regularization,» de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2017, pp. 13-9.

[90] M. Hagan y H. Demuth, «Regularization,» de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 13-10.

[91] D. J. MacKay, «Abstract,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 7.

[92] D. J. MacKay, «Data modelling and Occam's razor,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 7,8,9,10.

[93] D. J. MacKay, «Where Bayesian inference fits into the data modelling process,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 8.

[94] D. J. MacKay, «Why Bayes embodies Occam's razor,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 9.

[95] D. J. MacKay, «The evidence and the Occam factor,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 10-14.

[96] D. J. MacKay, «Model fitting,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 10,10.

[97] D. J. MacKay, «Model comparison,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 11.

[98] D. J. MacKay, «A modern Bayesian approach to priors,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 12.

[99] D. J. MacKay, «The Occam factor,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 13.

[100] D. J. MacKay, «Evaluating the evidence,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 12,13.

[101] D. J. MacKay, «Interpretation of the Occam factor,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 13.

[102] D. J. MacKay, «Occam factor for several parameters,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 14.

[103] D. J. MacKay, «The noisy interpolation problem,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 15,16.

[104] D. J. MacKay, «The first level of inference,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 16.

[105] D. J. MacKay, «How the best interpolant depends on α,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 17.

[106] D. J. MacKay, «Selection of parameters alpha and beta,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 16, 17, 18.

[107] D. J. MacKay, «Misfit, x^2, and the effect of parameter measurements,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 18.

[108] D. J. MacKay, «Bayesian choice of α and β,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 18,19,20.

[109] D. J. MacKay, «David J.C. MacKay,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 20.

[110] D. J. MacKay, «Why not find the joint optimum in w, α, β?,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 20.

[111] D. J. MacKay, «Evaluating the evidence,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 21.

[112] D. J. MacKay, «Good and bad parameter measurements,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, p. 22.

[113] D. J. MacKay, «Properties of the evidence maximum,» de *Bayesian Methods for Adaptive Models*, Pasadena, California, California Institute of Technology, 1991, pp. 21,22,23.

[114] D. J. Cameron MacKay, Bayesian Methods for Adaptive Models, Pasadena, California: California Institute of Technology, 1992.

[115] M. Hagan y H. Demuth, de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 13-18.

[116] M. Hagan y H. Demuth, «Regularization,» de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 13-8 - 13-18.

[117] M. Hagan y H. Demuth, «Selection of Data,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, p. 0711.

[118] M. Hagan y H. Demuth, «Data Collection and Preprocessing,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, p. 076.

[119] M. Hagan y H. Demuth, «Multiple Layers of Neurons,» de *Neural Network Design*, Stillwater, Oklahoma, Neural Network Design, 2nd Edition, eBook, 2014, pp. 2-12.

[120] M. Hagan y H. Demuth, «Choise of Network Arquitecture,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 22-9.

[121] M. Hagan y H. Demuth, «Training the Network,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 23-5.

[122] M. Hagan y H. Demuth, «Bayesian Regularization Algorithm,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, pp. 13-16 - 13-17.

[123] T. Instrument, « Data Sheet,» 12 2014. [En línea]. Available: https://www.ti.com/lit/ds/symlink/lmp7721.pdf?ts=1634527401928&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLMP7721. [Último acceso: 17 10 2021].

[124] Texas Instruments, «Data sheets,» Texas Instrumenst, [En línea]. Available: https://www.ti.com/product/LMP2231. [Último acceso: 2021 12 18].

[125] Texas Instruments, «Productos,» [En línea]. Available: https://www.ti.com/product/LMP2232. [Último acceso: 2021 10 18].

[126] Texas Instruments, «Product,» [En línea]. Available: https://www.ti.com/product/REF4132. [Último acceso: 18 10 2021].

[127] Texas Instruments, «Product,» [En línea]. Available: https://www.ti.com/product/REF3320. [Último acceso: 2021 10 18].

[128] Texas Instruments, «Product,» [En línea]. Available: https://www.ti.com/product/ADS1115. [Último acceso: 18 10 2021].

[129] Crystalfontz, [En línea]. Available: https://www.crystalfontz.com/product/cfah1602btmijt-16x2-character-lcd. [Último acceso: 18 10 2021].

[130] Microchip Technology Inc., [En línea]. Available: https://ww1.microchip.com/downloads/en/devicedoc/39632e.pdf. [Último acceso: 18 10 2021].

[131] M. Hagan y H. Demuth, «Fitting,» de *Neural Network Design*, New Yotk, Martin Hagan; 2 edition, 2014, pp. 22-20.

[132] «NHTSA,» United States Department of Transportation, [En línea]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety. [Último acceso: 04 09 2019].

[133] «BMW,» [En línea]. Available: https://www.bmw.com/en/automotive-life/autonomous-driving.html. [Último acceso: 04 09 2019].

[134] Tesla Inc., [En línea]. Available: https://en.wikipedia.org/wiki/Tesla,_Inc.. [Último acceso: 04 09 2019].

[135] «IEE Connected and Autonomous Vehicles,» IEEE, [En línea]. Available: http://sites.ieee.org/connected-vehicles/ieee-connected-vechicles/connected-vehicles/. [Último acceso: 04 09 2019].

[136] M. Hagan y H. Demuth, «title2,» de *Neural Network Design*, Oklahoma, Martin Hagan; 2 edition, 2014, p. page2.

[137] «Hysteresis and Backlash,» de *Measurement, Instrumentation, and Sensors* , New York, Taylor & Francis Group, LLC, 2014, pp. 4-7.

[138] M. L. a. M. Cates, «THE ROYAL SOCIETY,» THE ROYAL SOCIETY, 05 July 2017. [En línea]. Available: https://royalsocietypublishing.org/doi/10.1098/rsbm.2017.0013. [Último acceso: 28 September 2019].

[139] M. Hagan y H. Demuth, «Applications,» de *Neural Network Design*, Stillwater, Oklahoma, 2014, pp. 1-5 - 1-7.

[140] Hach, «Product - Process pH - Dirty Water,» Hach, [En línea]. Available: https://www.hach.com/guides/ProcessPH/pr-guide-processpH-DirtyWater-en.jsp. [Último acceso: 16 October 2019].

[141] C. I. Journal, «Clean India Journal,» 11 December 2012. [En línea]. Available: https://www.cleanindiajournal.com/handling-ph-control-of-industrial_wastewater/. [Último acceso: 16 October 2019].

[142] F. Focus, Food Focus, 19 July 2019. [En línea]. Available: https://www.foodfocus.co.za/home/Industry-Topics/food-safety/Important-parameters-to-measure-for-meat-processors. [Último acceso: 16 October 2019].

[143] K. Hildebrant, «The Guide to pH Measurement in Food & Drink,» Our Daily Brine, [En línea]. Available: https://ourdailybrine.com/how-to-test-the-ph-of-food-and-drink/. [Último acceso: 16 Oc tober 2019].

[144] Repsol, «Repsol,» [En línea]. Available: https://www.repsol.com/es/conocenos/donde-trabajamos/refineria-pampilla/index.cshtml. [Último acceso: 16 October 2019].

[145] Libelium, «pH-Sensor,» de *Smart Water Technical Guide - Document version: v7.6 - 09/2019*, Libelium Comunicaciones Distribuidas S.L., 2019, p. 46.

[146] IUPAC, «WHO WE ARE,» IUPAC, [En línea]. Available: https://iupac.org/who-we-are/. [Último acceso: 21 October 2019].

ANEXOS

Anexo A

Acondicionamiento de datos

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Explanation  :       This script selects 21 data at most each temperature
%                     variation.
% Type file    :       lvm (LabView)
%
% Name format  :       data*_*.Lvm, the first and second
%                     asterisks are numbers that identify each data file.
%
% Next script  :       Data_Normalization.m
%
% Version      :       1.0
% Design       :       David Fernández
%                     dfernandezv@uni.pe

clc; clear all;       % Remove items from workspace, freeing up system memory

training_data=[];
for f= 1 : 96
    temp_min=1;        % minimum temperature: 0.1°C, aprox
    temp_max=22;       % maximum: 95.0°C, aprox


    for r=temp_min : temp_max
        clc
        filename=sprintf('data%d_%d.lvm',f,r);
        i=1;
        Aa=importdata(filename);
        z=length(Aa);
        A=[Aa(:,4)  Aa(:,8)  Aa(:,16) Aa(:,18) Aa(:,20) Aa(:,22) Aa(:,6)]

        % Select pH values between 0.6 and 13.9
        while 1
            if(Aa(i,6)>1)
                if(Aa(i+1,6)<=13.9)
                    index_ini=i+1;
                    break;
                end
            end
            i=i+1;
        end
        while 1
            if(Aa(z,6)>0.6)
                index_fin=z;
                break;
            end
            z=z-1;
        end
        A_new=A(index_ini:index_fin,:);
        r=r+1;

        % Select the pH value closest to 7
```

```
pH_sign=7;
zz=1;
ini_a = 1;
end_a = length(A_new);
while 1
    a=A_new(ini_a,7)-pH_sign;
    b=A_new(ini_a+1,7)-pH_sign;
    if(b==0)
        zz=ini_a+1;
        break;
    end
    if(b<0)
        if(b<=-a)
            zz=ini_a;
            break
        else
            zz=ini_a+1;
        end
        break;
    end
    ini_a=ini_a+1;

    if (ini_a>end_a)
        disp('ERROR, no found pH 7.NN');
        break;
    end
end
A_new2=A_new;
ii=1;
jj=2;
[nn,mm]=size(A_new2);
outer_while=1;

% Select values so that the difference between two consecutive
% pH data is at leas 0.7
delta_pH=0.7; % pH delta between two rows
while 1
    cc=A_new2(ii,7)-A_new2(jj,7);
    if (jj==nn)
       break;
    end
    if (outer_while==end_a)
       break;
    end
   if (cc<=delta_pH)
        while 1
            if (jj==nn)
                break;
            end
             A_new2(jj,:) = [];
             [nn,mm]=size(A_new2);
             if ((A_new2(ii,7)-A_new2(jj,7))>delta_pH)
                 if (jj==nn)
                     break;
                 end
                 ii=ii+1;
                 jj=jj+1;
                 break;
```

```
                end
            end
        elseif(cc>delta_pH)
            ii=ii+1;
            jj=jj+1;
        end
        outer_while=outer_while+1;
    end

    % This search the index of the row that contains the
    % closest value to pH 7, in the new matrix whit pH values
    % between 0.6 and 13.9.
    zzz=1;
    ini_aa = 1;
    end_aa = length(A_new2);
    while 1
        aa=A_new2(ini_aa,7)-pH_sign;
        bb=A_new2(ini_aa+1,7)-pH_sign;
        if(bb==0)
            zzz=ini_aa+1;
            break;
        end
        if(bb<0)
            if(bb<=-aa)
                zzz=ini_aa;
                break
            else
                zzz=ini_aa+1;
            end
            break;
        end
        ini_aa=ini_aa+1;

        if (ini_aa>end_aa)
            disp('ERROR, no found pH 7.NN');
            break;
        end
    end

    % This adds the row with the pH value closest to pH 7 in case
    % this row has been deleted
    if(A_new2(zzz,7)~=A_new(zz,7))
        if (A_new2(zzz,7)<pH_sign)
            A_new3=[A_new2(1:zzz-1,:); A_new(zz,:); A_new2(zzz:end,:)];
        else
            A_new3=[A_new2(1:zzz,:); A_new(zz,:); A_new2(zzz+1:end,:)];
        end
    else
        A_new3=A_new2
    end
    training_data=[training_data;A_new3];
  end
  f=f+1;
end

save('training_data.mat','training_data')
```

Anexo B

Normalización de datos

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Explanation   :   This script normalize the data between -1 and 1
%
% Type file input   :   .mat (MatLab)
%
%  File Input   :   training_data.mat, this file is the result of
%                   script data_select.m
%
% Next script   :   Select_Data_Sets.m
%
% Version       :   1.0
% Design        :   David Fernández
%                   dfernandezv@uni.pe
%%%
clc; clear all;

% T_SW  pH_volt  Cal_10  Cal_7 Cal_4  Cal_T pH_SW
load ('training_data.mat')
x=training_data; % x(1:1000,1:7)

xmin_1=0;   xmax_1=100;
xmin_2=1;   xmax_2=3;
xmin_3=1.5; xmax_3=2.3;
xmin_4=1.6; xmax_4=2.4;
xmin_5=1.7; xmax_5=2.5;
xmin_6=0;   xmax_6=100;
xmin_7=0;   xmax_7=14;

col_1=2*(x(:,1)-xmin_1)./(xmax_1-xmin_1)-1;
col_2=2*(x(:,2)-xmin_2)./(xmax_2-xmin_2)-1;
col_3=2*(x(:,3)-xmin_3)./(xmax_3-xmin_3)-1;
col_4=2*(x(:,4)-xmin_4)./(xmax_4-xmin_4)-1;
col_5=2*(x(:,5)-xmin_5)./(xmax_5-xmin_5)-1;
col_6=2*(x(:,6)-xmin_6)./(xmax_6-xmin_6)-1;
col_7=2*(x(:,7)-xmin_7)./(xmax_7-xmin_7)-1;

training_data_norm=[col_1 col_2 col_3 col_4 col_5 col_6 col_7];
%training_data_norm=2*(x-xmin)./(xmax-xmin)-1;

save('training_data_norm.mat','training_data_norm');
% save('xmin.mat','xmin');
% save('xmax.mat','xmax');

max_min_values.xmin_1 = xmin_1;    %
max_min_values.xmax_1 = xmax_1;    %
max_min_values.xmin_2 = xmin_2;    %
max_min_values.xmax_2 = xmax_2;    %
max_min_values.xmin_3 = xmin_3;    %
max_min_values.xmax_3 = xmax_3;    %
```

```
max_min_values.xmin_4 = xmin_4;    %
max_min_values.xmax_4 = xmax_4;    %
max_min_values.xmin_5 = xmin_5;    %
max_min_values.xmax_5 = xmax_5;    %
max_min_values.xmin_6 = xmin_6;    %
max_min_values.xmax_6 = xmax_6;    %
max_min_values.xmin_7 = xmin_7;    %
max_min_values.xmax_7 = xmax_7;    %

save('max_min_values.mat','max_min_values');
```

Anexo C

Separación de sets de entrenamiento y validación

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Explanation   :   This script separe data for training, and testing
% Type file input   :   .mat (MatLab)
% File Input    :   training_data_norm.mat, this file is the result of
%                   script data_Normalization.m
% Next script   :   training_bayesian_regularization_V7_final.m
% Version       :   1.0
% Design        :   David Fernández
%                   dfernandezv@uni.pe
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear all;
load ('training_data_norm.mat')
data_for_test=15; % in percentage

Dataset_train_testing=training_data_norm;
%data_set_train=eye(30); %training_data_norm;

[c,d]=size(Dataset_train_testing);
k_end=ceil(c*data_for_test/100);
Dataset_testing=[];

for k=1:k_end %33171
    [a,b]=size(Dataset_train_testing);
    i=ceil(a*rand(1));
    %bb=[i;bb];
    Dataset_testing=[Dataset_train_testing(i,:);Dataset_testing];
    Dataset_train_testing(i,:)=[];
end

Dataset_train_input=Dataset_train_testing(:,1:6)';
Dataset_train_target=Dataset_train_testing(:,7)';
save('Dataset_train_input.mat','Dataset_train_input');
save('Dataset_train_target.mat','Dataset_train_target');

Dataset_testing_input=Dataset_testing(:,1:6)';
Dataset_testing_target=Dataset_testing(:,7)';
save('Dataset_testing_input.mat','Dataset_testing_input');
save('Dataset_testing_target.mat','Dataset_testing_target');
```

Anexo D

Algoritmo de entrenamiento por Regularización Bayesiana

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Descripción:
%   - Este scritp permite entrenar una red neuronal  de dos capas,
%     con función de activación tangente sigmoidal para la primera capa
%     y función de activación pulerine para la capa de salida, utilizando
%     regularización bayesiana.
%   - Este script se ejecuta en modo bach.
%   - Este script está basdo en el escript desarrollado por Martin T.
%     Hagan and Howard B. Demuth: function [net,tr] = nnd13train_br
%     (trainParam,P,T,VV,TT,func_test, perf_plot,mer_plot,pause_time,
%     b1_plot,b2_plot,gamk_plot).
%   - P: matriz con los datos de entrada
%   - T: mector con los datod de salida
%
% Version       :   1.0
%               :   David Fernández
%                   dfernandezv@uni.pe
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc, clear all
warning('off','all')


% Leer datos de entrada y datos de salida para el entrenamiento
load('Dataset_train_input.mat')
load('Dataset_train_target.mat')

P=Dataset_train_input;
T=Dataset_train_target;

% AJUSTE DE PARÁMETROS DE CONVERGENCIA
mingrad = 1e-11;     % Gradiente mínima
mu = 0.01;           % Parámetro para ajustar el balance entre la velocidad
                     % de convergencia del método y de Newton y la seguridad
                     % de convergencia del metodo descenso más empinado
                     % (steepest descent).
v = 2;               % Parámetro para para multiplicar o dividir mu en cada
                     % iteración
maxmu = 1e10;        % mu máximo
err_goal = 0;        % Objetivo de error
time = inf;          %
max_epoch = 10;  % Número máximo de épocas


% INICIALIZAR PARÁMETROS DE LA ARQUITECTURA
S1 =35;              % Número de neuronas en la capa de entrada
```

```
[R,Q] = size(P);    % lee la cantidad de datos de entrada para el
                    % entrenamiento.
[S2,Q] = size(T);   % lee la cantidad de datos de salida para el
                    % entrenamiento.

W20 = (2*rand(S2,S1)-1)*1;  % Inicializa pesos de la capa oculta [-0.5:0.5]
B20 = (2*rand(S2,1)-1)*1 ;  % Inicializa bias de la capa oculta [-0.5:0.5]

% Pesos y bias iniciales para la capa de entrada según  Nguyen-Widrow
% Calculates Nguyen-Widrow initial conditions: abs(W10)=0.7*S1^(1/R)
[W10,B10] = Nu_Wid_initial(S1,P) %
%

% dimensiones del los parámetros
RS = S1*R;          %
RS1 = RS+1;         %
RSS = RS + S1;      %
RSS1 = RSS + 1;     %

RSS2 = RSS + S1*S2; %
RSS3 = RSS2 + 1;    %
RSS4 = RSS2 + S2;   %

% parámetros iniciales
W1=W10;     % pesos iniciales de la primera capa
B1=B10;     % bias iniciales de la primera capa

W2=W20;     % pesos iniciales de la segunda capa
B2=B20;     % bias iniciales de la segunda capa

dW1=W10;    % inicialización de la variación inicial de pesos de la
            % primera capa
dB1=B10;    % inicialización de la variación inicial de bias de la
            % primera capa
dW2=W20;    % inicialización de la variación inicial de pesos de la
            % segunda capa
dB2=B20;    % inicialización de la variación inicial de bias de la
            % segunda capa

stop = '';                  %
startTime = clock;          % inicio de tiempo
ii=eye(RSS4);               % matriz diagonal

A1 = nndtansig(W1*P,B1);    % salida de la capa oculta
A2 = nndpurelin(W2*A1,B2);  % salida de la capa de salida
E1 = T-A2;                  % error

w = getX(W1,B1,W2,B2);      % vector de parámetros (weights and biases)
ssE = (sum(sum(E1.*E1)));   % Suma de errores cuadrados
N = prod(size(E1));         % número de errores
numParameters = length(w);  % número de parámetros en la red
gamk = numParameters;       % número de parámetros en la red
                            % (número de pesos y bias en la red)

% Cálculo de alpha y betha
if ssE==0,
    beta = 1;
else
```

```
  beta = (N - gamk)/(2*ssE);     % beta=(N-gamma)/2Ed
end

if beta<=0,
  beta=1;
end

ssX = w'*w;                  % ssX: Suma de errores cuadrádos
alph = gamk/(2*ssX);         % alpha inicial

f1 = beta*ssE + alph*ssX;    % Objective function
                             % f1 = beta*Ed + alpha*Ew

% Bucle principal
%==============================
for epoch = 1:max_epoch

% CÁLCULO DEL JACOBIANO
  A1 = kron(A1,ones(1,S2));      %
  D2 = mdeltalin(A2);            %
  D1 = mdeltatan(A1,D2,W2);      %
  Pp=kron(P,ones(1,S2));         %
  jac1 = learn_marq(Pp,D1);      %
  jac2 = learn_marq(A1,D2);      %
  jac=[jac1,D1',jac2,D2'];       % matriz Jacobiana

% MAGNITUD DE LA GRADIENTE
  E1=E1(:);                      %
  je=jac'*E1;                    % transpose(jac)*v(x)

  w = getX(W1,B1,W2,B2);         % vector de parámetros (weights and biases)
  grd = 2*(beta*je+alph*w);      % gradiente
  normgX = norm(grd);            % magnitud vectorial de la gradiente
                                 % sqrt(sum of squared each element of grd)

  % Guardar el valor de las variables
  tr.mer(epoch)=ssE;         % suma de errores cuadráticos
  tr.meu(epoch)=mu;          % velocidad de convergencia
  tr.grad(epoch)=normgX;     % gradiente
  tr.gamk(epoch) = gamk;     % numero efectivo de parámetros
  tr.alph(epoch) = alph;     % alph = gamk/(2*ssX); (libro)
  tr.beta(epoch) = beta;     % beta = (numErrors - gamk)/(2*ssE);
  tr.ssX(epoch) = ssX;       % {inicial: ssX = w'*w;}
                             % ssX: Suma de parámetros cuadráticos

% condiciones de parada
  currentTime = etime(clock,startTime);
  if (f1 <= err_goal)                % err_goal=0;
    stop = 'Performance goal met.'; % f1 = beta*Ed + alph*Ew;
                                     % función objetivo;
  elseif (epoch == max_epoch)        % max_epoch = 101;
    stop = 'Maximum epoch reached, performance goal was not met.';
  elseif (currentTime > time)        % time = inf;
    stop = 'Maximum time elapsed, performance goal was not met.';
  elseif (normgX < mingrad)
    stop = 'Minimum gradient reached, performance goal was not met.';
  elseif (mu > maxmu)
    stop = 'Maximum MU reached, performance goal was not met.';
```

```
end

if length(stop)       %
    break
end

jj=jac'*jac;          % transpose(jac(x))*jac(x)

% Actualización de parámetros
% Incremento de mu hasta que los errores logren reducirce
while mu < maxmu
  dw = -(beta*jj + ii*(mu+alph)) \ (beta*je + alph*w);    %variación de
      % de los parámetros
  dW1(:)=dw(1:RS);        % variación de los pesos de la primera capa
  dB1=dw(RS1:RSS);        % variación del bias de la primera capa
  dW2(:)=dw(RSS1:RSS2);   % variación de los pesos de la segunda capa
  dB2=dw(RSS3:RSS4);      % variación del bias de la segunda capa

  % nuevos parámetros (aún no son actualizados los parámetros)
  W1n=W1+dW1;         %
  B1n=B1+dB1;         %
  W2n=W2+dW2;         %
  B2n=B2+dB2;         %

  A1 = nndtansig(W1n*P,B1n);      % salida de la primera capa
  A2 = nndpurelin(W2n*A1,B2n);    % salida de la segunda capa
  E2 = T-A2;                      % error

  X2 = getX(W1n,B1n,W2n,B2n); % vector de parámetros (weights and biases)
  ssX2 = X2'*X2;              % suma de parámetros cuadráticos
  ssE2 = (sum(sum(E2.*E2)));  % Suma de errores cuadráticos
  f2 = beta*ssE2 + alph*ssX2;      % función objetivo a minimizar

  if (f2 < f1)
    % actualización de parámetros (utiliza los nuevos parámetros)
    W1=W1n;       % actualización de los pesos de la primera capa
    B1=B1n;       % actualización de los bias de la primera capa
    W2=W2n;       % actualización de los pesos de la segunda capa
    B2=B2n;       % actualización de los bias de la segunda capa
    E1=E2;        % errores

    ssE = ssE2;   % Suma de errores cuadráticos
    ssX = ssX2;   % Suma de parámetros cuadráticos
    w = X2;       % vector de parámetros (weights and biases)
    mu = mu / v;  % Actualización de la velocidad de convergencia

    if (mu < 1e-20)
      mu = 1e-20;
    end
    break
  end
  mu = mu * v;    % Actualización de la velocidad de convergencia
end

% número efectivo de parámetros
gamk = numParameters - alph*trace(inv(beta*jj+ii*alph));

% actualización de alpha y betha
```

```
  if ssX==0,
      alph = 1;
  else
      alph = gamk/(2*(ssX));    % parámetro de regularización alpha
  end

  if ssE==0,
      beta = 1;
  else
      beta =(N-gamk)/(2*ssE);   % parámetro de regularización betha
  end
  f1 = beta*ssE + alph*ssX;          % función objetivo a minimizar

end

% empaquetado de datos
tr.mer=tr.mer(1:epoch);         % ssE; Ed
tr.ssX = tr.ssX(1:epoch);       % ssX; Ew of weights and biases
tr.meu=tr.meu(1:epoch);         % Hessian matrix to be invertible
tr.grad=tr.grad(1:epoch);       % Gradient
tr.gamk = tr.gamk(1:epoch);     % gamma
tr.alph = tr.alph(1:epoch);     % alpha

save('tr.mat','tr');


  % grafica de la suma de errores cuadráticos
  hold on
  figure(1)
  plot([1:epoch],tr.mer)
  title('Sum of Squares Error')
  ylabel('ssE')
  hold on

  % grafica de el numero efectivo de parámetos
  figure(2)
  plot([1: epoch],tr.gamk)
  title('gamma')
  ylabel('gamk')
  hold on

  % grafica de la gradiente
  figure(3)
  plot([1: epoch],tr.grad)
  title('gradient')
  ylabel('grad')
  hold on

  % grafica de la suma de parámetros cudráticos.
  figure(4)
  plot([1 : epoch],tr.ssX)
  title('Sum of Square Parameters')
  ylabel('ssX')
  hold on


 % Guardado de datos
  save('W1.mat','W1');
```

```
  save('B1.mat','B1');
  save('W2.mat','W2');
  save('B2.mat','B2');

%función para opbtener el vector de parámetros
function x = getx(W1,B1,W2,B2)

[S1,R] = size(W1);  %
[S2,S1] = size(W2); %
RS = S1*R;          %
RS1 = RS+1;         %
RSS = RS + S1;      %
RSS1 = RSS + 1;     %
RSS2 = RSS + S1*S2; %
RSS3 = RSS2 + 1;    %
RSS4 = RSS2 + S2;   %

x(1:RS)=W1(:);      %
                    % W1(:):    all content to rows, one column
                    % x(1:RS):  all contes W1 to columns, one row.
x(RS1:RSS)=B1;      %
x(RSS1:RSS2)=W2(:); %
x(RSS3:RSS4)=B2;    %
x=x(:);             %
end

%================================
function [net] = getW(x,R,S1,S2)

RS = S1*R; RS1 = RS+1; RSS = RS + S1; RSS1 = RSS + 1;
RSS2 = RSS + S1*S2; RSS3 = RSS2 + 1; RSS4 = RSS2 + S2;

net.W1 = zeros(S1,R);
net.W2 = zeros(S2,S1);
net.W1(:)=x(1:RS);
net.B1=x(RS1:RSS);
net.W2(:)=x(RSS1:RSS2);
net.B2=x(RSS3:RSS4);
end
```

Anexo E

Inicialización de parámetros

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Ss - Number of neurons.
%  Pp - Inputs. script data_Normalization.m
%  Copyright 1995-2015 Martin T. Hagan and Howard B. Demuth

s=Ss
p=Pp;

[r,q] = size(p);
pmin = min(p')';
pmax = max(p')';

pr = [min(pmin) max(pmax)];

magw = 0.7*s^(1/r);
ff=2*rand(s,r)-1;
dd=nnnormr(ff);
w = magw*dd;
if (s==1)
     b = 0;
else
     b = magw*linspace(-1,1,s)'.*sign(w(:,1));
end
% Adjust for input range
x = 2./(pr(:,2)-pr(:,1));
y = 1-pr(:,2).*x;
xp = x';
% b = w*y+b
w = w.*xp(ones(1,s),:);
% norm(w(5,:))

function n = nnnormr(m)
[mr,mc]=size(m);
if (mc==1)
     n = m ./ abs(m);
else
     n = sqrt(ones./(sum((m.*m)')))'*ones(1,mc).*m;
end
```

Anexo F

Gráfica de objetivos y salidas de red neuronal artificial

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description   :   plot the targets and neural outputs
%
% version       :   1.0
%               :   David Fernández
%                   dfernandezv@uni.pe
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear all; close all;

load('Dataset_testing_input.mat')
load('Dataset_testing_target.mat')
load('W1.mat')
load('B1.mat')
load('W2.mat')
load('B2.mat')
load('max_min_values.mat')

xmin_7=max_min_values.xmin_7;
xmax_7=max_min_values.xmax_7;

P=Dataset_testing_input;
T=Dataset_testing_target;

A1 = nndtansig(W1*P,B1);     % output layer1;
A2 = nndpurelin(W2*A1,B2);   % output layer2;
E1 = T-A2;                   % error

X=xmin_7+(A2+1)*(xmax_7-xmin_7)/(2);
Target_testing=xmin_7+(T+1)*(xmax_7-xmin_7)/(2);

Output_nn=X(1,2000:2025);
Output_target=Target_testing(1,2000:2025);

Yy=[1:length(Output_nn)];

figure(6)
plot(Yy, Output_nn,'b--o', Yy,Output_target,'r:+')
legend('Red neuronal)', 'Objetivo (test)')
title('Red neuronal vs Objetivo')
ylabel('pH')
xlabel('Muestras')
%==================
```

Anexo G

Análisis de regresión lineal

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Description  :   Linear regression analysis
%
% Version       :   1.0
%                 :   David Fernández
%                     dfernandezv@uni.pe
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

load('Dataset_testing_input.mat')
load('Dataset_testing_target.mat')
load('W1.mat')
load('B1.mat')
load('W2.mat')
load('B2.mat')
load('max_min_values.mat')

xmin_7=max_min_values.xmin_7;
xmax_7=max_min_values.xmax_7;

P=Dataset_testing_input;
T=Dataset_testing_target;

A1 = nndtansig(W1*P,B1);    % output layer1;
A2 = nndpurelin(W2*A1,B2);  % output layer2;
E1 = T-A2;                  % error

X=xmin_7+(A2+1)*(xmax_7-xmin_7)/(2);
Target_testing=xmin_7+(T+1)*(xmax_7-xmin_7)/(2);

y=X;
t=Target_testing;

%%%[r,m,b] = regression(t,y) takes these arguments,
% t     :Target matrix or cell array data with a total of N matrix rows
% y     :Output matrix or cell array data of the same size
% r     :Regression values for each of the N matrix rows
% m     :Slope of regression fit for each of the N matrix rows
% b     :Offset of regression fit for each of the N matrix rows

[r,m,b] = regression(t,y);
R=r;        % correlation coefficient
R2=r*r;     % coefficient of determination

Xx=[0:0.1:14];
Yy=m*Xx+b;

hold on
```

```
plot(Xx,Yy,'b','LineWidth',2);
hold on
plot(t,y,'ko');
hold on
plot(Xx,Xx,'r--');
xlim([0 14.5]);
ylim([0 14.5]);

%rr=strg(r);
rr=num2str(r);
s1 = 'R=';
s2 = strcat(s1,rr);
title(s2);

m_s=num2str(m);
b_s=num2str(b);

t1 = 'Output~=';
s3 = '*Target+';
s4 = strcat(t1,m_s,s3,b_s);

ylabel(s4);
xlabel('Targets');

legend('Regresión','Data','Linea a 45°');

%%=========================
```

Anexo H

Análisis de histograma

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Histogram
%
% Version       :   1.0
%               :   David Fernández
%                   dfernandezv@uni.pe
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; close all;

load('Dataset_testing_input.mat')
load('Dataset_testing_target.mat')
load('W1.mat')
load('B1.mat')
load('W2.mat')
load('B2.mat')
load('max_min_values.mat')

xmin_7=max_min_values.xmin_7;
xmax_7=max_min_values.xmax_7;

P=Dataset_testing_input;
T=Dataset_testing_target;

A1 = nndtansig(W1*P,B1);     % output layer1;
A2 = nndpurelin(W2*A1,B2);   % output layer2;
E1 = T-A2;                   % error

X_test_output=xmin_7+(A2+1)*(xmax_7-xmin_7)/(2);
Target_testing=xmin_7+(T+1)*(xmax_7-xmin_7)/(2);

Error_testing=(Target_testing-X_test_output);

nbins = 25;
h = histogram(Error_testing,nbins)
title('Histrograma de errores')
xlabel('e = t - a')
ylabel('Número de errores')
```

Anexo I

Notas sobre el entrenamiento de la red neuronal artificial

En la **¡Error! No se encuentra el origen de la referencia.** se muestra diferentes resultados al ejecutar el algoritmo de entrenamiento para diferentes valores de $S1$ ($S1 = S^1$, cantidad de neuronas de la capa oculta).

Aunque en la **¡Error! No se encuentra el origen de la referencia.** se puede observar que la suma de errores cuadráticos, ssE, es pequeña, sin embargo, la cantidad de neuronas sugerida $\mathbf{S1}(\boldsymbol{\gamma})$ por el número de parámetros efectivos $\boldsymbol{\gamma}$ tiende al valor de **S1** elegido, $(\mathbf{S1}(\boldsymbol{\gamma}) \approx \mathbf{S1})$, provocando que no se pueda tomar una buena decisión sobre cantidad de neuronas a utilizar en la capa oculta.

La tendencia de $\mathbf{S1}(\boldsymbol{\gamma})$ al valor de **S1** se debe a que en la expresión matemática utilizada para el cálculo de $\boldsymbol{\gamma}$ es la siguiente:

$$\gamma = n - 2\alpha \mathrm{tr}(\mathbf{H})^{-1},$$

donde el factor $2\alpha \mathrm{tr}(\mathbf{H})^{-1}$ tiende a cero, debido a que $\alpha = \frac{\gamma^*}{2E_W(\mathbf{x})}$ tiende a cero, y esto se debe a que $E_W(\mathbf{x})$ es relativamente numéricamente grande como se aprecia en la **¡Error! No se encuentra el origen de la referencia.**, y por lo tanto $\gamma \approx n$, provocando que $(\mathbf{S1}(\boldsymbol{\gamma}) \approx \mathbf{S1})$. Recordar que inicialmente el algoritmo parte de la igualdad entre $\mathbf{S1}(\boldsymbol{\gamma})$ y $\mathbf{S1}$, $(\mathbf{S1}(\boldsymbol{\gamma}) = \mathbf{S1})$.

El valor de ssX es bastante grande debido a que hay valores de entrada a la RNA que se encuentran en rangos numéricos pequeños y otros que se encuentran en rangos numéricos decenas de veces más grandes, y, por lo tanto, para los rangos pequeños de valores los pesos y los bias asociados principalmente a tales rangos toman valores relativamente grandes, esto es para que esa entrada pequeña sea apreciable en comparación al valor de las entradas que pertenecen a rangos mucho más grandes al

momento de ingresar a la función de activación. Por ejemplo, el rango de temperatura de entrada se encuentra entre 0.6°C y 95.5°C, y el rango de voltaje del sensor de pH está entre el valor de 1.5 y 2.5 aproximadamente. Si ambos rangos mencionados se normalizan utilizando el mismo valor máximo y mínimo entonces los resultados de la normalización serán desequilibrados, por ejemplo, si consideramos la normalización general de todas las variables de entrada a la RNA en el rango de -1 a +1, el rango de variación de temperatura normalizada variaría entre -1 a +1 (ya que es el rango más amplio), y la variable de voltaje normalizado variaría en 0.018 a 0.040. De lo anterior se observa que el rango de variable voltaje normalizado es bastante pequeño, en comparación al rango de la temperatura normalizada, por lo tanto, los pesos que se asocien en mayor medida a la variable de voltaje normalizado deben ser grandes para que esta variable tenga efecto al ingresar en la función de activación. El hecho de que el valor numérico de los pesos sea grande provoca que ssX sea bastante grande (recordar que ssX es la suma los pesos y bias cuadráticos), y por lo tanto $\alpha = \frac{\gamma^*}{2E_W(\mathbf{x})}$ tienda a cero, provocando que $\gamma \approx n$, y por lo tanto $\mathbf{S1}(\boldsymbol{\gamma}) \approx \mathbf{S1}$.

Una forma de evitar que el valor de ssX sea numéricamente grande es normalizando independientemente cada variable de entrada a la RNA.

Tomando el ejemplo anterior, en el que el rango de temperatura de entrada se encuentra entre 0.6°C y 95.5°C, y el rango de voltaje del sensor de pH está entre el valor de 1.5 y 2.5 aproximadamente, y se normaliza independientemente cada variable de entrada tomando en cuenta su rango de variación, se tendría que el rango de variación de temperatura normalizada variaría entre -1 a +1, y el rango de la variable de voltaje normalizado variaría entre -1 a +1, y así sucesivamente, cada una de las 6 variables de entra a la RNA variaría entre -1 a +1.

El hecho de que cada una de las variables al momento de ingresar a la RNA varíen entre -1 a +1 significa que los pesos y bias no tendrán que ser muy grandes para realzar

el valor de alguna de las 6 variables de entrada al momento de ingresar a la función de activación. Esto conlleva a que el valor de ssX no se bastante grande como se muestra en la **¡Error! No se encuentra el origen de la referencia.** en comparación de los valores respectivos de la **¡Error! No se encuentra el origen de la referencia.**, y por lo tanto factor $2\alpha \mathrm{tr}(\mathbf{H})^{-1}$ no tiende a cero, debido a que $\alpha = \frac{\gamma^{\cdot}}{2E_W(\mathbf{x})}$ tampoco se aproximaría a cero, dando como resultado que $\mathbf{S1}(\boldsymbol{\gamma})$ y $\mathbf{S1}$ no sean similares o próximos. En el entrenamiento y optimización de la RNA de este trabajo se utilizó estas mejoras, aunque se haya tenido que desechar semanas de trabajos realizados, en aras de la mejora continua y un trabajo riguroso.

Ejecución del algoritmo de entrenamiento, con diferentes números de S1, (fuente: elaborado por el autor).

S1	ssE	ssX	γ	gradient	Error_max	Error mean	S1(γ)
3	2.68	1.20E+07	22.0	0.060448	2.517	0.369	2.6
3	3.16	7.85E+05	22.1	0.010472	2.857	0.370	2.6
3	2.68	1.20E+07	22.0	8234.5	2.517	0.369	2.6
3	2.89	1.61E+06	23.0	0.076377	2.929	0.355	2.8
3	2.89	1.61E+06	23.0	0.076377	2.929	0.355	2.8
4	0.98	4.64E+05	32.0	3.6152	2.023	0.189	3.9
4	0.98	4.64E+05	32.0	0.026723	2.023	0.189	3.9
4	1.38	9.23E+06	28.9	2119.1	1.462	0.264	3.5
5	0.27	3.17E+06	36.7	90082	1.147	0.106	4.5
5	0.28	3.22E+06	37.4	1.80E+06	1.264	0.104	4.6
5	0.51	6.02E+06	39.0	14.573	1.507	0.146	4.7
6	0.22	7.39E+06	47.0	4.0764	1.454	0.095	5.7
6	0.22	3.21E+06	47.4	37.984	1.221	0.094	5.8
9	0.06	2.85E+07	71.0	9.49E+06	1.012	0.051	8.8
10	0.05	1.68E+07	62.4	12339	0.970	0.045	7.7
15	0.02	1.84E+07	120.4	2.15E+07	0.454	0.026	14.9
20	0.02	4.92E+06	157.0	4.87E+07	0.588	0.027	19.5
20	0.01	8.20E+06	151.6	1.72E+08	0.262	0.023	18.8
20	0.01	6.54E+06	160.2	6.12E+07	0.247	0.023	19.9
25	0.01	6.97E+06	192.7	6.10E+07	0.263	0.021	24.0
15	0.02	9.81E+06	117.9	7.88E+06	0.299	0.027	14.6
18	0.01	2.89E+07	144.3	1.37E+07	0.228	0.019	17.9
18	0.01	2.04E+07	144.6	2.94E+07	0.310	0.019	17.9
20	0.01	2.02E+07	160.2	2.79E+06	0.267	0.019	19.9
18	0.01	1.37E+07	144.9	8.33E+07	0.381	0.022	18.0
18	0.02	9.34E+06	123.6	1.70E+07	0.522	0.027	15.3
18	0.01	2.23E+07	145.9	1.83E+07	0.256	0.017	18.1
18	0.01	2.84E+07	145.1	1.09E+07	0.252	0.018	18.0
20	0.01	3.00E+07	161.3	3.53E+07	0.222	0.020	20.0
20	0.01	7.66E+06	172.5	2.08E+06	0.651	0.026	21.4

20	0.01	2.61E+07	160.5	2961.7	0.440	0.019	19.9
20	0.01	1.10E+07	158.0	6.34E+06	0.621	0.021	19.6

Ejecución del algoritmo de entrenamiento, con diferentes valores de S1 y cada variable de entrada normalizado respecto a su propio rango, (elaborado por el autor).

S1	ssE	ssX	γ	gradient	Error_max	Error mean	S1(γ)
15	0.5784	16915	113.53	5.6207	0.3682	0.02257	14.07
20	0.2667	15592	153.94	1546.6	0.2151	0.01544	19.12
25	0.2003	96166	186.94	7310.8	0.1574	0.01338	23.24
25	0.2307	20699	187.20	0.0034566	0.2348	0.01456	23.28
30	0.1766	91543	225.73	1.05E+05	0.1529	0.01273	28.09
30	0.1815	70425	225.36	60.206	0.1550	0.01290	28.05
30	0.1803	29047	224.90	0.018636	0.1612	0.01283	27.99
35	0.1951	33684	262.66	0.0078455	0.1387	0.01323	32.71
35	0.1758	39541	264.69	12981	0.1289	0.01249	32.96
40	0.1855	22701	302.44	292.35	0.1750	0.01272	37.68
15	0.5784	16915	113.53	5.6207	0.3682	0.0226	14.07
20	0.2667	15592	153.94	1546.6	0.2151	0.0154	19.12
25	0.2003	96166	186.94	7310.8	0.1574	0.0134	23.24
30	0.1803	29047	224.90	0.018636	0.1612	0.0128	27.99
35	0.1758	39541	264.69	12981	0.1289	0.0125	32.96
40	0.1855	22701	302.44	292.35	0.1750	0.0127	37.68

Anexo J

Programa para la lectura de pH y temperatura – Smart Water

```
/*
    ------ [SW_01] - pH sensor Reading for Smart Water--------

    Explanation: Turn on the Smart Water Board and reads the pH sensor
    extracting the value from the calibration values and temperature
    compensation

    Copyright (C) 2016 Libelium Comunicaciones Distribuidas S.L.
    http://www.libelium.com

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

    Version:           3.0
    Design:            David Gascón
    Implementation:    Ahmad Saad
*/

#include <WaspSensorSW.h>

float pHVol;
float temp;
float pHValue;

// Calibration values
#define cal_point_10  1.985
#define cal_point_7   2.070
#define cal_point_4   2.227

// Temperature at which calibration was carried out
#define cal_temp 23.7

pHClass pHSensor;
pt1000Class temperatureSensor;


void setup()
{
  USB.ON();
  USB.println(F("pH example for Smart Water..."));

  // Store the calibration values
  pHSensor.setCalibrationPoints(cal_point_10, cal_point_7, cal_point_4,
cal_temp);

  ////////////////////////////////////////////
  // 1. Turn ON the Smart Water sensor board
  ////////////////////////////////////////////
  Water.ON();
```

```
}

void loop()
{
  //////////////////////////////////////////
  // 2. read the sensors
  //////////////////////////////////////////

  // Read the ph sensor (voltage value)
  pHVol = pHSensor.readpH();
  // Read the temperature sensor
  temp = temperatureSensor.readTemperature();
  // Convert the value read with the information obtained in calibration
  pHValue = pHSensor.pHConversion(pHVol, temp);

  //////////////////////////////////////////
  // 3. Print the output values
  //////////////////////////////////////////

  USB.print(F("pH value: "));
  USB.print(pHVol);
  USB.print(F("volts  | "));
  USB.print(F(" Temperature: "));
  USB.print(temp);
  USB.print(F("degrees  | "));
  USB.print(F(" pH Estimated: "));
  USB.println(pHValue);


}
```

Anexo K

Programa PIC18F4550

```
////////////////////////////////////////////////////////////////////////////
//***** Lectura de un conversor ADC ADS1115 por medio de un PIC18F4550 *****//
//
/*
    Esta versión lee y muestra la temperatura y el PH
    Versión:            1.0
    Diseño:             David Fernández
    Implementación:     David Fernández
    e-mail:             dfernandezv@uni.pe // fernandezv.david@gmail.com
*/

#INCLUDE <18F4550.h> // librería del PIC18F4550
#fuses HS,HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#USE DELAY(CLOCK=20MHz, crystal) // Velocidad de cristal oscilador externo

//Configura dirección de memoria de los puertos A,B,C,D
#BYTE PORTA= 5
#BYTE PORTB= 6
#BYTE PORTC= 7
#BYTE PORTD= 8
#use standard _io(b)

// velocidad de comunicación I2C entre el ADC ADS1115 y el microcontrolador
PIC18F4550
#use i2c(MASTER,Fast=100000, sda=PIN_B2, scl=PIN_B3,force_sw)

//Librerias matemáticas y del display LCD
#include <lcd.c>
#include <stdio.h>
#include <stdint.h>
#INCLUDE <math.h>

// Configuración de display LCD 2X16
#define LCD_ENABLE_PIN  PIN_D0                                  ////
#define LCD_RS_PIN      PIN_D1                                  ////
#define LCD_RW_PIN      PIN_D2                                  ////
#define LCD_DATA4       PIN_D4                                  ////
#define LCD_DATA5       PIN_D5                                  ////
#define LCD_DATA6       PIN_D6                                  ////
#define LCD_DATA7       PIN_D7

//Variables temporales para acomular temporalmente valores leidos
// y almacenar promedios
unsigned int8 lb_a0 = 0x00;
unsigned int16 a0_conc = 0x00;
unsigned int8 hb_a0 = 0x0000;

unsigned int8 lb_a1 = 0x00;
unsigned int16 a1_conc = 0x00;
unsigned int8 hb_a1 = 0x0000;

unsigned int8 lb_a2 = 0x00;
unsigned int16 a2_conc = 0x00;
unsigned int8 hb_a2 = 0x0000;

unsigned int8 lb_a3 = 0x00;
unsigned int16 a3_conc = 0x00;
```

```
unsigned int8 hb_a3 = 0x0000;

int a=1;
int32 a0_conc_acom=0;
int32 a1_conc_acom=0;
int32 a2_conc_acom=0;
int32 a3_conc_acom=0;

//variables para almaceamiento de promedios finales de lectura de voltajes
float vol_a0 0;
float vol_a1 0;
float vol_a2 0;
float vol_a3 0;

// tiempo de lectura para el ADS1115
int ts_r=15; //tiempo para leer
int ts_co=2; //tiempo para cerrar e iniciar comunicación
int ts_ni=2; // tiempo entre una y otra instrucción común

//variable ´para la lectura de temperatura
float val=0;


////////////////////////////////////////////////////////////////////////////////
//****************** PARÁMETROS DE LA RED NEURONAL ENTRENADA*********************

// PESOS DE LA CAPA 1
const float W1[35][6]={
   { 0.1278,   50.5511,   -1.9148,  -18.3453,   -0.0146,   -0.1483},
   { 0.3466,    0.1352,   14.9281,   16.5050,  -27.7630,    8.6475},
   {-0.0425,   31.1630,    0.3771,  -12.9578,    0.0959,    0.0455},
   { 0.0176,  -10.5723,   -0.0657,    4.1926,    0.0803,   -0.0267},
   {-1.4615,  -13.2632,   19.9811,  -14.5734,    0.0219,    1.4109},
   {-1.4598,  -11.2829,   20.0665,  -15.4123,    0.0445,    1.4086},
   {-0.0377,   30.2300,    0.3365,   12.5550,    0.1060,    0.0404},
   {-0.6087,  -14.1800,    8.3316,   -2.8006,    0.0530,    0.6050},
   { 0.0321,    2.4931,    0.0351,   -1.5990,    0.4206,    0.0230},
   { 0.0971,   51.3997,   -1.5295,  -19.0565,   -0.0111,   -0.1066},
   { 0.7350,    0.1619,   42.0159,   24.6636,  -51.4476,   15.0023},
   {-0.5719,  -23.4402,    7.7899,    1.9213,   -0.0060,    0.5762},
   {-0.5552,   -0.0545,  -17.5441,  -21.9076,   37.6579,  -10.9869},
   { 1.2816,   13.5257,  -17.5366,   12.0484,    0.0018,   -1.2346},
   { 0.8969,  -10.0127,   -0.0382,   -7.7902,   12.0083,   -0.9232},
   {-0.2595,   36.5347,    1.1438,  -13.7672,   -1.7633,    0.1885},
   { 0.2284,   -9.6622,   -0.0083,    0.9023,    3.2756,   -0.2200},
   {-0.0049,  -10.3587,   -0.0368,    4.3440,   -0.1864,   -0.0024},
   {-0.2622,   -6.3975,   -0.3562,    6.9329,   -3.3222,    0.1890},
   {-1.2411,  -13.9400,   16.9674,  -11.3339,   -0.0029,    1.1981},
   { 0.5581,   23.5584,   -7.6146,   -2.1385,    0.0076,   -0.5636},
   { 0.0096,   47.1382,    0.0180,   19.0437,   -0.1831,   -0.0145},
   {-0.0345,   31.1563,    0.2954,  -12.8962,    0.1134,    0.0358},
   { 0.1083,   50.1973,   -1.6612,   18.4543,   -0.0078,   -0.1223},
   { 1.3718,    8.1254,  -20.3831,   16.8605,   -0.0766,   -1.3217},
   { 0.0338,  -10.5981,   -0.1044,    4.0657,    0.2541,   -0.0433},
   { 0.4811,    8.2137,   -0.0175,   -3.1408,    6.4474,   -0.4725},
   { 1.1578,   17.8651,  -15.6786,    8.5982,   -0.0677,   -1.1469},
   {-1.4222,    9.4134,   20.0580,  -16.0836,    0.0638,    1.3708},
   {-0.5494,  -23.8601,    7.5059,    2.3692,   -0.0105,    0.5561},
   { 0.0394,    2.7979,   -0.0233,   -1.7149,    0.4539,   -0.0309},
   { 0.0694,  -11.4671,   -0.1000,    3.9346,    0.7123,   -0.0814},
   {-0.2566,   -6.4132,   -0.3492,    6.8509,   -3.2541,    0.1851},
   { 0.0886,  -11.1696,   -0.1206,    3.5939,    0.9463,   -0.0999},
   {-0.2270,    9.4612,    0.0131,   -0.8476,   -3.2371,    0.2193}};


// PESOS DE LA CAPA 2
```

```
const float W2[1][35]={-8.5040,   -0.0120,  -14.4388,   30.7739,   25.2449,
-26.4490,   29.5620,    0.1264,    2.9620,   -8.6957,    0.0026,   -4.9377,
-0.0085,   15.1675,    0.4662,   -0.0349,   -2.1302,  -11.9549,   -1.8770,
 4.0064,    9.9959,    2.3842,  -13.3509,   18.4209,    6.1574,  -12.1440,
 0.1784,   -0.0787,   17.4323,   -4.9572,   -2.7288,  -14.4982,    1.9453,
 8.7814,   -2.3425 };


 // BIAS DE LA CAPA 1
const float B1[35][1]={{0.0150},   { 6.0360},   {-0.2143},   { 0.0582},   {-
4.8836},
   { 4.6456},   { 0.0106},   {-1.9868},   { 0.8452},   { 0.1892},   { 8.8842},
   { 2.0763},   { 8.5620},   { 4.5311},   { 3.2044},   {-1.7852},   { 1.7957},
   { 0.2106},   { 0.4093},   {-4.1140},   { 2.1272},   { 0.0581},   { 0.2181},
   { 0.0931},   { 4.5647},   {-0.1806},   { 1.6870},   { 3.2081},   { 4.5031},
   {-2.1943},   { 0.9794},   { 0.0726},   { 0.4245},   { 0.0352},   {-1.7423}};


// PESOS DE LA CAPA 2
const float B2[1][1]={{-1.6597}};

//*************** FIN DE PARÁMETROS DE LA RED NEURONAL ENTRENADA****************
////////////////////////////////////////////////////////////////////////////////


// Valores de calibración para la lectura de PH.
float P1[1][7]={{30.2000,    1.8130,    1.9020,    2.056,    2.204,   30.20,
6.6117}};

// valores máximos y mínimos de parámetros para normalizar. Tambien es usado
para
// convertir a su escala real (no convertido)
const float min_1=0;
const float max_1=100;

const float min_2=1;
const float max_2=3;

const float min_3=1.5;
const float max_3=2.3;

const float min_4=1.6;
const float max_4=2.4;

const float min_5=1.7;
const float max_5=2.5;

const float min_6=0;
const float max_6=100;

const float min_7=0;
const float max_7=14;

// varibles para normalizar parámetros de configuración y lectura de voltajes
// en los sensores
float Nor_T_SW=0;
float Nor_PH_Vol=0;
float Nor_Cal_10=0;
float Nor_Cal_7=0;
float Nor_Cal_4=0;
float Nor_Cal_T=0;
float Nor_PH_SW=0;

float Nor_P1[6][1]={};

//variables auxiliares
```

```
float TT=0;
float H=0;
float H2=0;
float nn=0;
float AA1[35][1]={};
float AA2=0;
float Out_NN=0;

////////////////////////////////////////////////////////////////////////////
//////////////////////////  FUNCIÓN DE LECTURA DE PH  //////////////////////////
// Esta función recibe la lectura de voltajes de los sensores, lo normaliza y
// utiliza la Red Neuronal entrenada para obtener el valor correspondiente de PH
// normalizado, luego el valor normalizado de PH es convertid en valor real
// (no normalizado) y finalmente es enviado para presentarlo al usuario.
float PH_function(float n, float nnn){
// la función recibe el voltaje "n" proveniente del sensor de PH, y la
// temperatura "nnn " en °C para determinar el valor de PH.
P1[0][1]=n;    // voltaje del sensor de PH.
P1[0][0]=nnn;  // temperatura actual de medición en °C.
output_high(PIN_B6);

// nomalización de parámetros y voltajes
Nor_T_SW=2*(P1[0][0]-min_1)/(max_1-min_1)-1;
Nor_PH_Vol=2*(P1[0][1]-min_2)/(max_2-min_2)-1;
Nor_Cal_10=2*(P1[0][2]-min_3)/(max_3-min_3)-1;
Nor_Cal_7=2*(P1[0][3]-min_4)/(max_4-min_4)-1;
Nor_Cal_4=2*(P1[0][4]-min_5)/(max_5-min_5)+1;
Nor_Cal_T=2*(P1[0][5]-min_6)/(max_6-min_6)+1;
Nor_PH_SW=2*(P1[0][6]-min_7)/(max_7-min_7)+1;

// Vector de parámetros de ingreso a la RNA
Nor_P1[0][0]=Nor_T_SW;
Nor_P1[1][0]=Nor_PH_Vol;
Nor_P1[2][0]=Nor_Cal_10;
Nor_P1[3][0]=Nor_Cal_7;
Nor_P1[4][0]=Nor_Cal_4;
Nor_P1[5][0]=Nor_Cal_T;
TT=Nor_PH_SW;

// variables de apoyo
int a;
int b;
int c;

// cálculo de valores para la salida de la primera capa
for(a=0;a<35;a++){
   output_low(PIN_B4);
   H=0;
   for(b=0;b<6;b++){
   H=H+ W1[a][b]*Nor_P1[b][0];
   }
   nn = H+B1[a][0];
   AA1[a][0]=2/(1+exp(-2*nn))-1 ;
}

H2=0;

// cálculo de valores para la salida de la segunda capa
for(c=0;c<35;c++){
   //output_low(PIN_B4);
   H2= H2+ W2[0][c]*AA1[c][0];
   }
// valor de PH normalizado
AA2= H2+B2[0][0];

// conversión del valor de PH normalizado a valor real (no normalizado)
```

```
Out_NN=min_7+(AA2+1)*(max_7-min_7)/(2);


// Envio del valor rela de PH
return Out_NN;
}
//////////////////////// FIN -  FUNCIÓN DE LECTURA DE PH  ////////////////////


////////////////////////////////////////////////////////////////////////////
//////////////////////  FUNCIÓN CONVERSOR ANALÓGICO A DIGITAL  //////////////////
void read_ADS115() {
///////////////////////////// ENTRADA ANALÓGICA A0  ////////////////////////////
  output_high(PIN_B4); // Encendido de led indicador de muestreo en curso

  delay_ms(ts_co);
  i2c_start();        // Inicia el protocolo de comunicación I2C.
  i2c_write(0x90);    // Envio de la dirección del ADS1115 (7 bits) y el modo de
                      // operación (1 bit): 10010000 (0x90).
                      // El bit 8, indica si va a escribir o leer (R/W bit).
                      // (first 7-bit I2C address followed by a low R/W bit).
  delay_ms(ts_ni);
  i2c_write(0x1);     // Envio de dirección de configuración de registro del
ADS1115
                      // 000000+01 (00000001: dirección paa configurar el
registro)
  delay_ms(ts_ni);
  i2c_write(0xC1);    // Configuración del byte más significativo del registro
                      // de configuración.
                      // 1+100+000+1 (0xC1).
                      // primer bit: inicio de conversión simple.
                      // tres bits siguientes 100: AINP = AIN0 y AINN = GND.
                      // tres bist siguientes 000: FSR = ±6.144 V.
                      // último bit 1: Modo de muestreo único (una lectura por
                      // muestra).
  delay_ms(ts_ni);
  i2c_write(0x83);    // Configuración del byte menos significativo del registro
                      // 100_0_0_0_11 (0x83).
                      // tres primeros bits 100: 128 SPS.
                      // siguiente bit 0: sin comparador tradicional.
                      // siguiente bit 0: sin comparador de polaridad.
                      // siguiente bit 0: sin comparador de enclavamiento.
                      // siguiente bits 11: sin comparador y establece el pin
                      // ALERT/RDY en alta impedancia.
  delay_ms(ts_r);
  i2c_stop();         // Fin de configuración del registro

  //configuración para lectura
  delay_ms(ts_co);
  i2c_start();        // Inicia el protocolo de comunicación I2C.
  i2c_write(0x90);    // Envio de la dirección del ADS1115 (7 bits) y el modo de
                      // operación (1 bit): 10010000.
                      // El bit 8, indica si va a escribir o leer (R/W bit)
                      // (first 7-bite I2C address followed by a low R/W bit)
  delay_ms(ts_ni);
  i2c_write(0x0);     // Envio de dirección hacia el registro de conversión del
                      // ADS1115.
                      // 000000+00 (00000000: dirección hacia el registro
                      // de conversión).
  delay_ms(ts_ni);
  i2c_stop();         // Fin de direccionamiento hacia el registro de converión

 // LECTURA DE DEL REGISTRO DE CONVERSIÓN
  delay_ms(ts_co);    //
  i2c_start();        // Inicia el protocolo de comunicación I2C.
  i2c_write(0x91);    // Envio de la dirección del ADS1115 (7 bits) y el modo de
```

```
                    // operación (1 bit): 10010001.
                    // El bit 8, indica si va a escribir o leer (R/W bit)
                    // (first 7-bite I2C address followed by a low R/W bit)
  delay_ms(ts_ni);
  hb_a0 = i2c_read();// Lectura del byte más significativo
  delay_ms(ts_ni);
  lb_a0 = i2c_read();// Lectura del byte menos significativo
  delay_ms(ts_ni);
  i2c_stop();        // Fin de lectura del canal A0
  //delay_ms(2);
  a0_conc=make16(hb_a0, lb_a0);  // Concatenamos el byte más y menos
significativo

  output_low(PIN_B4);   // apagado de led indicador de muestreo en curso
//////////////////////////// FIN ENTRADA ANALÓGICA A0  ////////////////////////////

// La configuración de las demás entradas analógicas es similar a la confi-
// guración de la entrada analógica A0. solo es necesario cambiar la direcciones
// de los puestos a leer (A1, A2 y A3).

//////////////////////////// ENTRADA ANALÓGICA A1  ////////////////////////////
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);

  delay_ms(ts_ni);
  i2c_write(0x1);
  delay_ms(ts_ni);
  i2c_write(0xD1);
  delay_ms(ts_ni);
  i2c_write(0x83);
  delay_ms(ts_r);
  i2c_stop();

  //configuración para lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);
  delay_ms(ts_ni);
  i2c_write(0x0);
  delay_ms(ts_ni);
  i2c_stop();

  // lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x91);
  delay_ms(ts_ni);
  hb_a1 = i2c_read();
  delay_ms(ts_ni);
  lb_a1 = i2c_read();
  delay_ms(ts_ni);
  i2c_stop();
  a1_conc=make16(hb_a1, lb_a1);
  output_high(PIN_B4);
//////////////////////////// FIN ENTRADA ANALÓGICA A1  ////////////////////////////

//////////////////////////// ENTRADA ANALÓGICA A2  ////////////////////////////
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);

  delay_ms(ts_ni);
  i2c_write(0x1);
  delay_ms(ts_ni);
  i2c_write(0xE1);
```

```
  delay_ms(ts_ni);
  i2c_write(0x83);
  delay_ms(ts_r);
  i2c_stop();

  //configuración para lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);
  delay_ms(ts_ni);
  i2c_write(0x0);
  delay_ms(ts_ni);
  i2c_stop();

  // lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x91);
  delay_ms(ts_ni);
  hb_a2 = i2c_read();
  delay_ms(ts_ni);
  lb_a2 = i2c_read();
  delay_ms(ts_ni);
  i2c_stop();
  a2_conc=make16(hb_a2, lb_a2);
  output_low(PIN_B4);
//////////////////////////// FIN ENTRADA ANALÓGICA A2  ////////////////////////////

//////////////////////////// ENTRADA ANALÓGICA A3  ////////////////////////////
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);

  delay_ms(ts_ni);
  i2c_write(0x1);
  delay_ms(ts_ni);
  i2c_write(0xF1);
  delay_ms(ts_ni);
  i2c_write(0x83);
  delay_ms(ts_r);
  i2c_stop();

  //configuración para lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x90);
  delay_ms(ts_ni);
  i2c_write(0x0);
  delay_ms(ts_ni);
  i2c_stop();

  // lectura
  delay_ms(ts_co);
  i2c_start();
  i2c_write(0x91);
  delay_ms(ts_ni);
  hb_a3 = i2c_read();
  delay_ms(ts_ni);
  lb_a3 = i2c_read();
  delay_ms(ts_ni);
  i2c_stop();
  a3_conc=make16(hb_a3, lb_a3);

//////////////////////////// FIN ENTRADA ANALÓGICA A3  ////////////////////////////

}
```

```
// FUNCIÓN PRINCIPAL
void main()
{
   output_high(PIN_B5); // Diodo verde indicador de ejecución del proceso
   delay_ms(500);
   output_low(PIN_B5);
   lcd_init();          // Inicialización de la pantalla LCD

   int sampl=15;        // Número de muestras para promediar

while (true)
{

// Lectura de los canales analógicos del ADS1115
   for(int a=1;a<(sampl+1);a++)
   {
      delay_ms(20);
      read_ADS115();
      a0_conc_acom=a0_conc_acom+a0_conc;
      a1_conc_acom=a1_conc_acom+a1_conc;
      a2_conc_acom=a2_conc_acom+a2_conc;
      a3_conc_acom=a3_conc_acom+a3_conc;
   }

   output_high(PIN_B5);

   // Promedio de muestras
   a0_conc=a0_conc_acom/sampl;
   a1_conc=a1_conc_acom/sampl;
   a2_conc=a2_conc_acom/sampl;
   a3_conc=a3_conc_acom/sampl;

   // Conversión de tipo de variable a punto flotante
   vol_a0=(float)a0_conc*6.144/32768;
   vol_a1=(float)a1_conc*6.144/32768;
   vol_a2=(float)a2_conc*6.144/32768;
   vol_a3=(float)a3_conc*6.144/32768;

   // conversion de temperatura en grados celcius (°C)
   delay_ms(1);
   val =(vol_a0 2.048);
   val = (val+2.048)/(2.048 val)*1000;
   val = 0.26048*val 260.83; // Temperatura correspondiente al voltaje Vadc
   delay_ms(1);


// Llama a la función de lectura de PH. Esta función devuel vel valor de PH.
   vol_a2=PH_function(vol_a3, val);

// Presentación de valores de temperatura y PH al usuario mediante el display
LCD
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   printf(lcd_putc,"%1.3f",vol_a0);//Visualiza la temperatura
   lcd_gotoxy(6,1);
   printf(lcd_putc,"V");//Visualiza la temperatura
   lcd_gotoxy(9,1);
   printf(lcd_putc,"%1.2f",val);//Visualiza la temperatura
   lcd_gotoxy(15,1);
   printf(lcd_putc,"\xDF");//Visualiza la temperatura
   lcd_gotoxy(16,1);
   printf(lcd_putc,"C");//Visualiza la temperatura

   lcd_gotoxy(1,2);
```

```
    printf(lcd_putc,"%1.3f",vol_a3);//Visualiza la temperatura
    lcd_gotoxy(6,2);
    printf(lcd_putc,"V");//Visualiza la temperatura
    lcd_gotoxy(9,2);
    printf(lcd_putc,"%2.3f",vol_a2);//Visualiza la temperatura
    lcd_gotoxy(15,2);
    printf(lcd_putc,"PH");//Visualiza la temperatura
    delay_ms(1000);

    a0_conc_acom=0;
    a1_conc_acom=0;
    a2_conc_acom=0;
    a3_conc_acom=0;
    val=0;

    output_low(PIN_B5);

}
}
```