UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA DE SISTEMAS Y ADMINISTRACIÓN DE EMPRESAS

CARRERA PROFESIONAL INGENIERÍA DE SISTEMAS

"IMPLEMENTACIÓN DE UNA SOLUCIÓN BASADA EN REDES NEURONALES PARA LA OPTIMIZACIÓN DEL PROCESO DE REFINAMIENTO DE ACEITE PARA UNA EMPRESA PESQUERA".

TRABAJO DE SUFICIENCIA PROFESIONAL
PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO DE SISTEMAS

PRESENTADO POR EL BACHILLER

CORREA AGUSTIN, JUAN NICOLÁS

Villa el Salvador
2016

DEDICATORIA

Este esfuerzo se lo dedico a Dios, a mis padres quienes confiaron en mí y estuvieron ahí cuando más los necesite.

AGRADECIMIENTOS

A mi casa de estudios la Universidad Nacional Tecnológica de Lima Sur por permitirme cumplir con mis objetivos e incrementar mis conocimientos en pos de ser un mejor profesional y para afrontar el mundo globalizado.

A mis docentes que siempre mostraban interés en que aprendamos de sus experiencias no solo académicas sino también laborales y enseñarnos que con esfuerzo y dedicación todo se puede lograr.

A mis compañeros que de alguna u otra manera estuvieron ahí para apoyarnos para poder llegar a la meta.

ÍNDICE

LISTADO DE FIGURAS

LISTADO DE TABLAS

INTRODUCCIÓN

La empresa “Pescamar” es una empresa pesquera líder en su sector, produciendo alimentos e ingredientes marinos de alta calidad, con un gran valor agregado y excelencia, lo cual les ha permitido convertirse en el primer productor y exportador de harina y aceite de pescado del mundo.

Actualmente Pescamar cuenta con una planta que se encarga exclusivamente del refinamiento y concentración de aceite de pescado con alto contenido de Omega 3. El Omega 3 es un ácido graso esencial poliinsaturado (el organismo humano no los puede fabricar sino es a partir de otras sustancias), que se encuentran en alta proporción en los tejidos de ciertos pescados, frutos secos, aceites vegetales (tales como el aceite de canola y de girasol).

Es por ello que Pescamar consciente de todos los beneficios que produce el Omega 3 (principalmente esto lo encontramos en productos farmacéuticos y nutracéuticos), le dedica mucho atención en el tratamiento y refinamiento del aceite de pescado para la obtención de nuevos aceites con altos contenidos de Omega 3.

Ahora el proceso para la obtención de nuevos aceites (y que estos sean los esperados), suele ser muy complejo puesto que no existe una fórmula matemática que te diga que al someter el aceite de pescado a ciertas condiciones puedas obtener un tipo de aceite con cierta cantidad de EPA (ácido eicosapentaenoico) y DHA (ácido docosahexaenoico) deseada, siendo estos los

más importantes ácidos grasos de la familia del Omega 3, mientras se obtenga mayor % de EPA y DHA el aceite de Omega 3 es más puro.

Es por ello que de ahí proviene la propuesta solución y el proyecto que he realizado. Analizando el proceso que se realiza para el refinamiento del aceite verifique que el mismo es secuencial e iterativo, y que este toma demasiado tiempo y coste de recursos para poder obtener un aceite de Omega 3 deseado. Y esto es porque no se cuenta con una fórmula matemática que te diga que al someter un aceite a ciertas condiciones puedes obtener un aceite con una alta concentración de Omega 3 en la menor cantidad de pasos posibles, y sin esa información no podrán analizar todos los tipos de aceite de Omega 3 generados y los costes que implican producirlos, y por ende no se podrá aplicar una buena estrategia de venta de su producto a las demás industrias interesadas en el aceite de Omega 3.

Esta problemática como tal es un proyecto de minería de datos, ya que se maneja grandes volúmenes de información, con esto me refiero a todas las posibilidades de aceites que se van a manejar para la elección de los aceites deseados.

Ahora como tema central a analizar es el modelo matemático que se va a aplicar para la solución de la problemática planteada, en donde los algoritmos más cercanos para la solución de este tipo de problemas son los algoritmos predictivos (pues estos se basan principalmente en la extracción de patrones de datos históricos, y es con lo que contamos ahora sólo datos históricos) y entre los más importantes tenemos las redes neuronales, árboles de decisión y

máquinas de vectores de soporte (SVMs), cada uno de estos dependiendo del proyecto te brinda un grado de error, por ello se deben analizar varios modelos para ver cuál de todos es el que te genera el menor porcentaje de error.

Para este proyecto se escogió como algoritmo predictivo el perceptrón multicapa (es un tipo de algoritmo de redes neuronales), y como algoritmo de aprendizaje el backpropagation.

Lo que se busca con este modelo es el poder predecir con un grado de exactitud (ya que cualquier algoritmo predictivo cuenta con un porcentaje de confiabilidad) el nuevo tipo de aceite generado basado en unas entradas (%EPA, %DHA, Temperatura y Tiempo).

Finalmente a manera de reflejar los nuevos registros generados (ya que el algoritmo te va a generar esos nuevos tipos de aceites), se utilizó una herramienta reporteadora tal es el caso de QlikView).

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción de la Realidad Problemática.

En el proceso de refinamiento de un aceite, este es sometido a varias transformaciones (condiciones tales como temperatura, tiempo, etc.) y al finalizar todo este proceso te genera un nuevo aceite, el cual puede tener un cierto grado de pureza de EPA y DHA, este proceso se puede repetir hasta encontrar el aceite buscado.

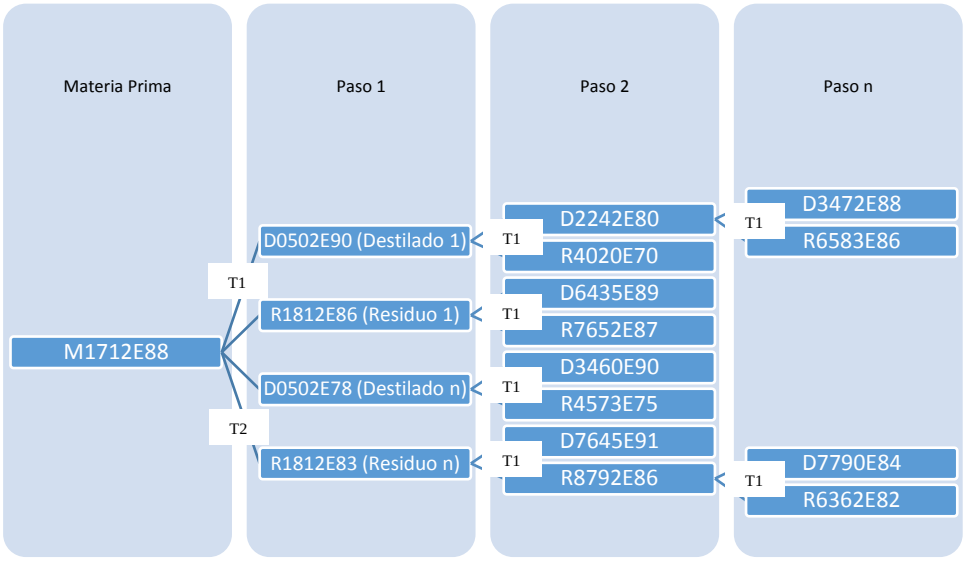


Figura 1
Proceso de Refinamiento de aceite

El problema radica en que no se conoce que tipo de aceite se va a obtener luego de someter un aceite inicial a ciertas condiciones de transformación (temperatura, presión, etc).

1.2. Justificación del Problema.

Para la empresa Pescamar el proceso de refinamiento de aceite puede incurrir a grandes costes de tiempo y dinero, si es que no se sabe cuáles son los valores de las variables de transformación necesarios para poder obtener un aceite con cierto grado de pureza de EPA y DHA determinado.

1.3. Delimitación del Proyecto.

1.3.1. Delimitación Temporal:

La presente investigación se inició a inicios de Junio hasta finales de Julio del presente año.

1.3.2. Delimitación Espacial:

El lugar en donde se realizó he implementó el proyecto fue en las instalaciones de su planta, en el distrito de Pucusana, provincia de Lima.

1.4. Formulación del Problema.

1.4.1. Problema Principal.

¿Cómo optimizará una solución basada en redes neuronales el proceso de refinamiento de aceite para una empresa pesquera?

1.4.2. Problemas Específicos.

- ¿Cómo se va a optimizar el proceso para la obtención de un nuevo tipo de aceite?

- ¿De qué manera se va a obtener el aprendizaje de la red neuronal?
- ¿Cómo se va a validar el correcto funcionamiento de la red neuronal y los datos que estos arrojan?
- ¿Qué se va a realizar para poder visualizar y analizar los registros de los nuevos aceites?

1.5. Objetivos.

1.5.1. Objetivo General.

Implementar de una solución basada en redes neuronales para la optimización del proceso del refinamiento de aceite en una empresa pesquera.

1.5.2. Objetivos Específicos.

- Diseñar el modelo de la red neuronal multicapa para la optimización del proceso para la obtención de un nuevo tipo de aceite.
- Entrenar la red neuronal a través del algoritmo de aprendizaje (backpropagation).
- Evaluar los resultados de la red neuronal (testeamos la función generada) y el grado de eficiencia del modelo.
- Diseñar e Implementar un sistema de información para la visualización de los registros de los nuevos aceites generados por el modelo.

CAPITULO II

2.1. Antecedentes de la Investigación

2.2.1 Antecedentes Internacionales.

Redes neuronales y pre-procesado de variables para modelos y sensores en bioingeniería, presentado por el Mg. Fernando Mateo Jiménez; Universidad Politécnica de Valencia, España; En muchos problemas de regresión (aproximación de funciones), clasificación (redes neuronales el cual se usó en este trabajo de tesis doctoral) y optimización, en general se hace uso de las nuevas metodologías basadas en la inteligencia artificial. La inteligencia artificial es una rama de las ciencias de la computación que busca automatizar la capacidad de un sistema para responder a los estímulos que recibe y proponer salidas adecuadas y racionales, en ese proyecto se buscó proponer una alternativa viable a la aproximación de modelos y procesos en el ámbito científico y, más concretamente, en aplicaciones complejas de bioingeniería, en las cuales es imposible o muy costoso encontrar una relación directa entre las señales de entrada y de salida mediante modelos matemáticos sencillos o aproximaciones estadísticas.

El aporte que encuentro en este trabajo de investigación es el hecho de aplicar redes neuronales a un problema real y tan importante (un gran aporte en el ámbito de bio-alimentario) como la predicción de presencia de toxinas (micotoxinas) asociadas a los hongos que pueden crecer en determinados alimentos debido a diferentes factores ambientales asociados al cambio climático y que son consideradas cancerígenas (Jiménez, 2012).

2.2.2 Antecedentes Nacionales.

MODELADO Y CONTROL BASADO EN REDES NEURONALES ARTIFICIALES DE UNA PLANTA PILOTO DE DESALINIZACIÓN DE AGUA DE MAR POR ÓSMOSIS INVERSA, presentado por el sr. Neil Neizer Carrasco Banda, Universidad Pontificia Universidad Católica Del Perú, 2016; Se plantea la hipótesis de que es posible aumentar la eficiencia de las plantas desalinizadoras de agua de mar utilizando sistemas de control inteligente, conformados por controladores multivariables fundamentados en redes neuronales artificiales y modelos matemáticos multivariables basados en las leyes de conservación de la materia y energía. Llegando a la conclusión de que aplicando las leyes de conservación de la materia y la energía, se obtuvo un modelo matemático multivariable para la unidad de ósmosis inversa de una planta desalinizadora de agua de mar, siendo los resultados satisfactorios alcanzados en la validación del modelo matemático con datos reales de operación de una planta piloto se permitió verificar que el modelo describe de forma adecuada el comportamiento dinámico de dicha planta, ya que se ajusta con un 92.37% a los datos experimentales de la dependencia del flujo

de permeado con la presión de la alimentación. Con un 74.57% a los datos experimentales de la conductividad del permeado en función de la presión de la alimentación. Y un 78.71% para la conductividad del permeado en función del pH de la alimentación (Carrasco 2016).

IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL RECONOCIMIENTO DE CARACTERES BASADO EN LA RED NEURONAL PERCEPTRON, presentado por la señorita Sammy Nahín Carranza Hernández, Universidad Pontificia Universidad Católica Del Perú, 2014; Proyecto de tesis que tiene como objetivo final construir una un sistema basado en el funcionamiento de redes neuronales para el reconocimiento de caracteres dibujados a mano.

Se llega a tomar como aporte la práctica correcta del proceso de aplicación de la red neuronal, dividiendo el mismo en 2 fases: de entrenamiento y testeo.

2.2. Bases Teóricas

2.2.1 Redes Neuronales Biológicas

Para entender las redes neuronales artificiales (RNA), se debe establecer cómo funciona una red neuronal biológica, porque la estructura del cerebro se fundamenta en la composición de millones de células interconectadas entre sí mediante la sinapsis, la unidad fundamental con la cual el cerebro percibe los estímulos físicos y químicos de la naturaleza. La unidad básica del sistema nervioso son las neuronas, las neuronas tienen tres componentes principales: las dendritas, el cuerpo y el axón. Las dendritas son los elementos receptores de la red, encargados de llevar las señales eléctricas de las excitaciones

fisicoquímicas. El cuerpo conduce y suma todas las señales eléctricas desde la entrada. El axón es una fibra larga que permite la conexión desde el cuerpo hasta otras neuronas, se ramifica en su extremo en pequeños bulbos finales que casi tocan las dendritas de las células vecinas. Un esquema para describir una red neuronal biológica es la que se presenta en la Figura 1.

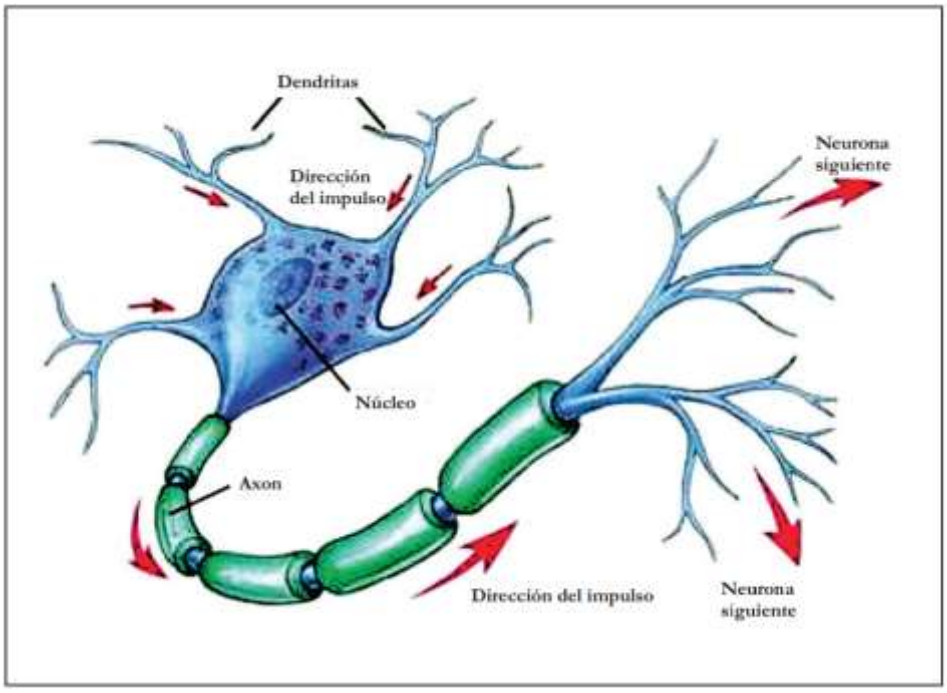


Figura 2
Componentes de una Red neuronal biológica

Las redes neuronales funcionan de igual forma como lo hace el resto del cuerpo humano: mediante impulsos eléctricos y reacciones químicas. Los impulsos eléctricos que usan las redes neuronales para intercambiar información con las demás, viajan por el axón que hace contacto con las dendritas de la neurona vecina mediante las sinapsis. La intensidad de la corriente transmitida depende de la eficiencia de la transmisión sináptica. El impulso eléctrico es transmitido mediante las dendritas y conducido por el axón hacia otras conexiones entre otras neuronas. De manera que el estímulo percibido por las

dendritas sea la causa su reacción, en los órganos o tejidos vinculados a esas neuronas, depende de la intensidad del estímulo y de la ponderación del mismo.

2.2.2 Redes Neuronales Artificiales (RNAs)

La red neuronal artificial trata de capturar la esencia de los sistemas neuronales biológicos e imitar su comportamiento. Una neurona artificial se puede describir de la siguiente manera:

- Recibe una serie de “entradas” (sean datos originales o “salidas” de otras neuronas). Cada entrada llega a través de una conexión que tiene una cierta “fuerza”, o “peso”, que equivale a la eficiencia sináptica de una neurona biológica. Cada neurona tiene también un determinado valor de umbral. La suma ponderada de las entradas menos el valor de umbral componen la “activación” de la neurona (también conocida como Potencial Post-Sináptico (PPS) de la neurona).
- La señal de activación pasa a través de una función de activación, o función de transferencia, para producir la “salida” de la neurona. Esta función limita el rango de valores que puede tomar la variable de salida de la neurona.

Por lo tanto, el nivel de actividad de cada neurona es función de las entradas que recibe, y el resultado se envía como una señal a través de sus conexiones con otras neuronas. Cada neurona sólo puede dar un valor de salida al mismo tiempo. Una neurona artificial puede estar implementada por medio de componentes hardware o ser simulada por medio de software en un ordenador. Para ilustrar lo explicado anteriormente consideremos que existe una señal x_j a la entrada de la sinapsis j de la neurona k. Al atravesar la j-ésima conexión

sináptica, la variable x_j se multiplica por el peso w_{kj} y un umbral o sesgo b_k (del inglés, bias) es agregado al resultado. El resultado pasa a ser mapeado por una función de activación ϕ que puede ser lineal o no lineal, aunque en general se utilizan funciones no lineales. En la Figura 2 se aprecia una neurona artificial que utiliza una función de activación lineal, es decir, es un sumador o combinador lineal de las m señales de entrada ponderadas por los pesos w_{kj}, j = [1, . . ., m]. En términos matemáticos, las ecuaciones que definen el cálculo que realiza una neurona artificial son las siguientes:

$$u_k = \sum_{j=1}^{m} (w_{kj} x_j) \tag{2.1}$$

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

o bien en una única ecuación:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right) \tag{2.3}$$

Es habitual considerar el umbral b_k como un peso extra w_{k0} que se multiplica por una supuesta entrada ficticia x_0 fijada a 1, por lo que solamente contribuye como una constante al resultado.

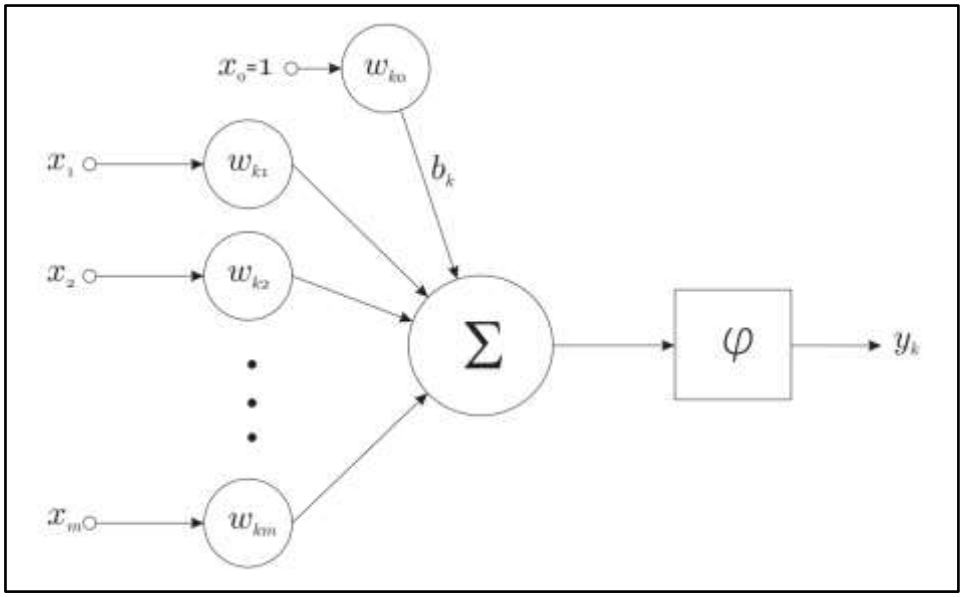


Figura 3
Diagrama de una neurona artificial.

2.2.3 Funciones de Activación

En este apartado vamos a hacer un inciso sobre las funciones de activación más típicas que se suelen utilizar. En concreto se trata de las funciones: escalón, lineal a trozos, sigmoide.

2.2.3.1 La función escalón:

Está definida por la Ecuación 2.4 y aparece representada en la Figura 3. A este tipo de neurona se refiere la literatura como neurona de McCulloch-Pitts (McCulloch-Pitts, 1943). Se emplea principalmente para clasificación, ya que la salida binaria es apropiada para la selección entre dos clases.

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v < 0 \end{cases} \qquad (2.4)$$

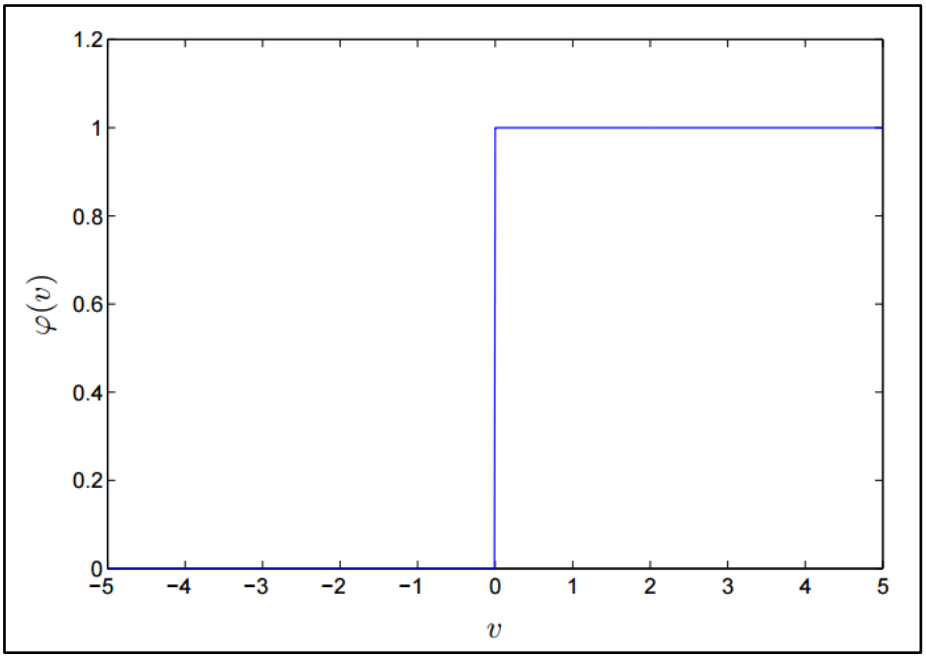


Figura 4
Función Escalón.

2.2.3.2 La función lineal a trozos:

Viene descrita por la Ecuación 2.5, y su representación correspondiente se muestra en la Figura 4.

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \qquad (2.5)$$

El factor de amplificación (pendiente) en la región lineal de operación se asume que es la unidad. La forma de esta función de activación se aproxima a la respuesta de un amplificador no lineal. Se pueden dar los siguientes dos importantes casos particulares de la función lineal a trozos:

- La función lineal a trozos se convierte en un combinador lineal si la región lineal de operación se mantiene sin entrar en saturación.

- La función lineal a trozos se convierte en un escalón si el factor de amplificación en la región lineal tiende a infinito.

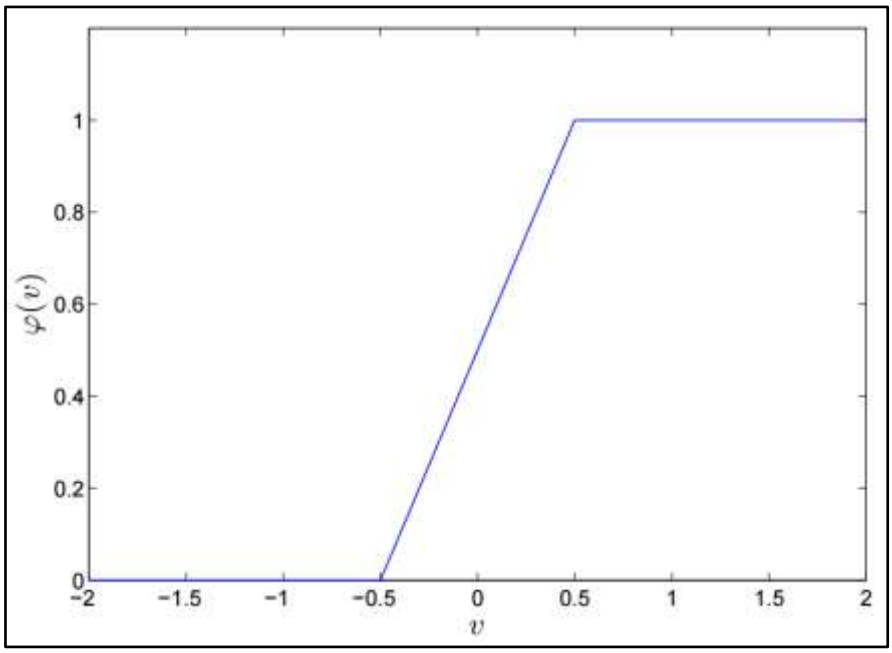


Figura 5
Función Lineal a trozos.

2.2.3.3 La función sigmoide:

Es una función de activación muy popular. Está definida por una función en forma de “s” y estrictamente creciente, mostrando un comportamiento suave. La ecuación que describe la sigmoide logística es la siguiente:

$$\varphi(v) = \frac{1}{1 + e^{-av}} \qquad (2.6)$$

donde a > 0 es el parámetro de pendiente. En la Figura 5 se representan varias sigmoides con distinto parámetro a. La importancia de esta función es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando v es cero. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la función escalón, con

la ventaja respecto a esta función, de que la derivada está definida para todo el intervalo.

Igualmente se definen otros tipos de funciones de activación como la Gaussiana,

$$\varphi(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} e^{-\frac{x^2}{2}} dx \qquad (2.7)$$

o bien, la función arcotangente:

$$\varphi(v) = \frac{1}{\pi} \arctan(v) + \frac{1}{2}. \qquad (2.8)$$

Las funciones de activación descritas abarcan el rango [0, 1]. A veces es deseable que cubran el rango [−1, 1]. En ese caso se pueden definir las mismas funciones de forma antisimétrica respecto al origen. Por ejemplo, la función escalón pasaría a estar definida como:

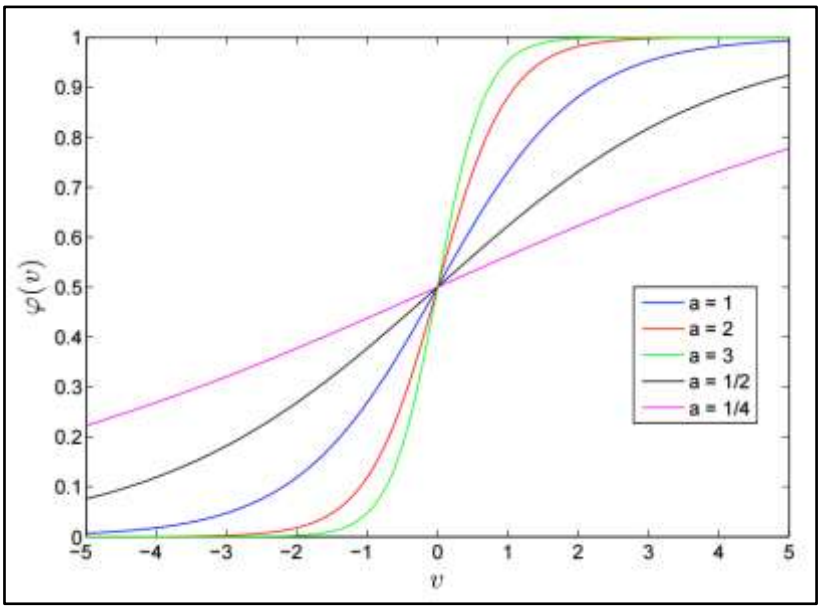


Figura 6
Función Sigmoide.

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v = 0 \\ -1 & \text{si } v < 0 \end{cases} \qquad (2.9)$$

lo que se conoce habitualmente como función signo. Para el caso de la sigmoide se obtiene una función muy utilizada como función de activación por sus buenos resultados denominada tangente hiperbólica:

$$\varphi(v) = \tanh(v) = \frac{e^{av} - e^{-av}}{e^{av} + e^{-av}} \qquad (2.10)$$

donde a es de nuevo el parámetro que controla la pendiente.

2.2.4 Perceptrón y Perceptrón Multicapa

2.2.4.1 Perceptrón:

El perceptrón (Rosenblatt, 1958) es la red neuronal que realiza la función más sencilla posible, la de clasificación binaria. Un perceptrón es una red compuesta por una única neurona. Básicamente, la función que realiza es mapear un vector de entrada x ∈ R d a un valor de salida binario (0 o 1) por medio de la siguiente función f:

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x + b > 0 \\ 0 & \text{en caso contrario} \end{cases} \qquad (2.11)$$

donde w ∈ R d es el vector pesos y w·x es el producto escalar. El valor escalar b corresponde al umbral. Esta función de activación no es más que un escalón

como el definido en la Sección 2.2.3 Como se ha mencionado, la salida binaria implica que el perceptrón se puede usar para clasificar una muestra en dos posibles clases.

El umbral representa la desviación u offset que tendrá el plano de decisión que separa las dos clases, respecto al origen de coordenadas. Suponiendo un caso de clasificación binaria entre las clases C_1 y C_2, el efecto del umbral sobre el plano o frontera de decisión se aprecia en la Figura 6.

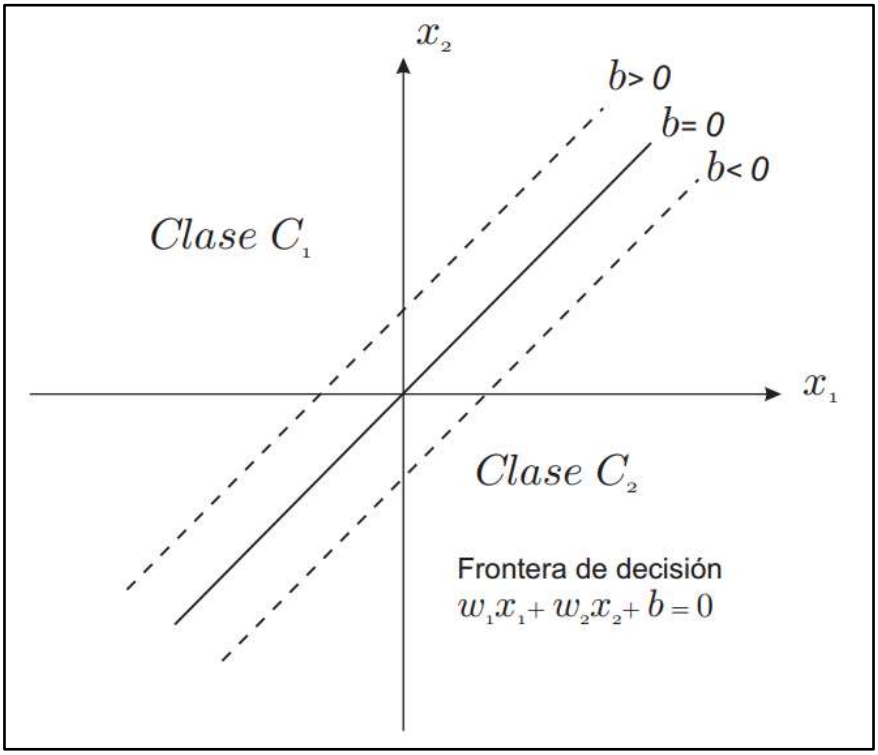


Figura 7
Efecto del signo del umbral b sobre la frontera de decisión en un problema de clasificación de dos clases.

Para explicar la forma en que un perceptrón “aprende”, supongamos que tenemos N pares de puntos (xi , yi), i = 1, . . ., N. Supongamos que el umbral b = 0, sin pérdida de generalidad, ya que se puede añadir un elemento más al vector

de entrada, x(0) = 1, en cuyo caso el peso $w_{(0)}$ reemplaza al umbral b. En cada iteración el vector de pesos w se actualiza, para cada una de las N muestras, usando la siguiente regla:

$$w(j+1) = w(j) + \alpha(y - f(x))x(j), \;\; j = 1, \ldots, d \qquad (2.12)$$

donde $0 < \alpha \leq 1$ es el factor de aprendizaje. Se puede observar que perceptrón aprenderá mientras la diferencia entre la salida obtenida y la salida deseada no sea 0. Los pesos se suelen inicializar a 0, o a valores pequeños aleatorios. La gran limitación del perceptrón, que fue sacada a la luz por Minsky y Papert en su libro “Perceptrones”, es que un perceptrón solamente puede resolver problemas linealmente separables, lo que deja fuera de su alcance problemas tan sencillos como una simple función OR exclusiva (XOR).

2.2.4.2 Perceptrón Multicapa:

El perceptrón multicapa (MLP) es una red neuronal de tipo feedforward que contiene una o más capas ocultas de neuronas. Tal como se muestra en la Figura 7, una red de tipo MLP se compone de varias capas de neuronas. Típicamente existen: a) una capa de entrada, b) una o varias capas ocultas y c) una capa de salida. El MLP de la Figura 7 aparece totalmente interconectado (cada neurona de una capa tiene una conexión a cada neurona de la anterior capa y de la siguiente capa). Hay casos en que existe interconexión parcial (cada neurona se conecta solamente con un subconjunto de las neuronas de la capa anterior o de la siguiente), aunque esto último no es muy habitual. También es posible la conexión entre neuronas de una misma capa en las denominadas redes competitivas, o incluso la realimentación de la señal de salida a las entradas, en las redes llamadas recurrentes. La capa de entrada de un MLP realmente sólo

realiza la función de permitir la entrada de muestras a la red neuronal, por lo que no es una capa de “neuronas” en el sentido estricto de la palabra. No contiene pesos ni umbrales, ni se realiza en ella ninguna operación matemática, sino que simplemente “lee” los datos que se presentan a su entrada y los distribuye a la primera capa oculta de neuronas. Visto de otra forma, se puede también considerar que la capa de entrada de una ANN contiene nodos cuya activación representa valores de las variables del dominio del problema en el cual la red es aplicada. Por su parte, la capa de salida representa las decisiones que toma la ANN. Existirán al menos tantas neuronas de salida como variables de salida tenga el problema a considerar. A menudo la función de activación de las neuronas de salida es lineal, de manera que sólo realiza la suma ponderada de las señales procedentes de la última capa oculta. También se pueden usar funciones sigmoidales según en qué problema. En problemas en que la salida debe ser binaria se utiliza una función escalón. Las capas ocultas representan relaciones entre las variables de entrada que no pueden interpretarse a simple vista. Representan variables que no pertenecen al dominio del problema, sino que codifican relaciones entre ellas a más alto nivel. El número de capas ocultas es un parámetro a optimizar cuando se diseña una ANN. El Teorema de Aproximación Universal especifica que “Una red MLP de una capa oculta es suficiente para realizar el mapeo entrada-salida que proporciona una realización aproximada de cualquier función continua en un intervalo dado”. En la práctica se observa que en muchas ocasiones, aun siendo suficiente el uso de una capa oculta, al utilizar dos capas ocultas, el rendimiento de la red MLP mejora considerablemente.

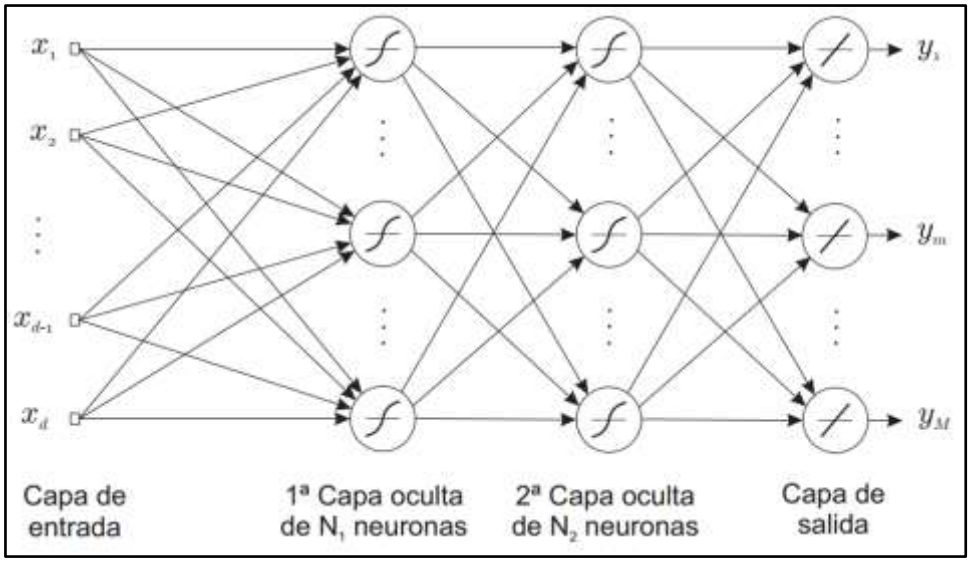


Figura 8

Estructura de una ANN de tipo MLP de d entradas, dos capas ocultas con activación sigmoidal, de N_1 y N_2 neuronas, respectivamente, y una salida con función de activación lineal. Los umbrales de cada neurona se han omitido en el diagrama por claridad. El sentido de las flechas indica la dirección en que se propagan las señales (feedforward).

Siguiendo la estructura del ejemplo descrito en la Figura 7, pasamos a describir matemáticamente el comportamiento de un MLP de dos capas ocultas. Aplicando la ecuación que describe una única neurona (Ecuación 2.3) y, asumiendo que el umbral de la neurona j-ésima es el producto de un peso w_{j0} por una entrada ficticia fijada a 1 ($x_0 = 1$), se puede extrapolar la ecuación analítica a la primera capa de neuronas ocultas del MLP de la siguiente manera:

$$a_j = \varphi_2 \left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right) \qquad (2.13)$$

siendo x_i ,(i = 1, . . ., d) la i-ésima entrada, y $w_{(1)ji}$ el peso que va desde la entrada i hasta la neurona j,(j = 1, . . ., N1) de la primera capa oculta. Del mismo modo, para la segunda capa oculta se tienen como entradas las a_j obtenidas y se aplica de nuevo la Ecuación 2.13:

$$b_k = \varphi_2 \left(\sum_{j=0}^{N_1} w_{kj}^{(2)} a_j \right) \qquad (2.14)$$

Finalmente, para la capa de salida de M nodos:

$$y_m = \varphi_3 \left(\sum_{k=0}^{N_2} w_{mk}^{(3)} b_k \right), \quad m = 1, \ldots, M \qquad (2.15)$$

y sustituyendo 2.13 en 2.14, y el resultado en 2.15 se tiene la ecuación que define la m-ésima salida del MLP:

$$y_m = \varphi_3 \left(\sum_{k=0}^{N_2} w_{mk}^{(3)} \varphi_2 \left(\sum_{j=0}^{N_1} w_{kj}^{(2)} \varphi_2 \left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i \right) \right) \right) \qquad (2.16)$$

Cabe destacar que la función de activación ϕ_3 es lineal en este caso concreto, por lo que $\phi_{3(u)} = u$.

2.2.5 El aprendizaje con redes neuronales

2.2.5.1 Entrenamiento, validación y test

La forma más común de aprendizaje, y también de adaptación, que tienen las redes neuronales es, como hemos visto, a través de la modificación de los pesos (w_{ij}) que cada neurona le asigna a la información que recibe de las neuronas que la preceden. Así, las ANNs van modificando los pesos en la etapa de

aprendizaje, que se conoce como “entrenamiento”, para reducir progresivamente el error que cometen al tratar de predecir las variables independientes. Con el fin de que una ANN aprenda exitosamente una tarea es necesario el entrenamiento. Esto requiere que se presente a su entrada de manera secuencial cada muestra de un conjunto de datos que será un subconjunto de los datos totales disponibles. En caso de ser entrenamiento supervisado también se necesitarán las salidas asociadas. Se conoce como época a una presentación completa del conjunto de entrenamiento durante el proceso de aprendizaje. El aprendizaje ocurre época a época hasta que los pesos y umbrales se estabilizan y el criterio de error (típicamente el error cuadrático medio (MSE)) sobre el conjunto de entrenamiento completo converge a algún valor mínimo. Además, es conveniente que las muestras se presenten de forma aleatoria para convertir en estocástica la búsqueda en el espacio de los pesos y así minimizar la posibilidad de convergencia a un mínimo local. En la etapa de entrenamiento las redes neuronales van calculando sus predicciones de las variables requeridas por el usuario y cada vez que obtienen un valor lo comparan con el valor real que tuvo la variable que se intentó predecir; así la red ve en qué medida se equivocó y modifica los pesos sinápticos para tratar de disminuir ese error. Para esto la red neuronal puede partir con pesos iguales a cero ($w_{ij} = 0$) o, más habitualmente, inicializar los pesos de forma aleatoria antes de empezar a “aprender”. De este modo tiene un punto desde donde empezar a adaptar los pesos. Una forma muy común de ir modificando los pesos es mediante una distribución proporcional del error cometido en la predicción entre todas las neuronas según el aporte de cada una al resultado final, es decir, cuanto más aportó la neurona o conexión al resultado, mayor parte del error se le asignará a ella para modificar sus pesos.

El método básico para estimar la eficiencia predictiva de un algoritmo de entrenamiento es medir el error que comete sobre un conjunto de muestras que se ha mantenido al margen durante el proceso de entrenamiento. Este conjunto se denomina conjunto de test o conjunto hold-out. El método hold-out de test es el más extendido y el más sencillo, puesto que sólo es necesario separar un conjunto de muestras y reservarlas para comprobar la generalización de un modelo una vez entrenado. Cuando las muestras disponibles para entrenar son escasas, es posible obtener una medida más exacta de la estimación utilizando la validación cruzada. La validación cruzada k-fold consiste en particionar los datos disponibles en k subconjuntos con aproximadamente el mismo número de muestras. El proceso de validación cruzada supone k iteraciones, en las cuales el algoritmo de entrenamiento utiliza $k - 1$ subconjuntos para entrenar, y deja fuera uno de ellos. El subconjunto que se deja fuera va rotando en cada iteración de manera que, al final del proceso, todos los datos han formado parte del entrenamiento y del test en algún momento (en concreto $k - 1$ veces para entrenar y una para testar). La medida del error cometido es, por lo tanto, la media aritmética de los errores cometidos en las diferentes iteraciones. Se puede observar gráficamente este proceso en la Figura 8 para $k = 6$. Este método llevado al extremo, para $k = N$, siendo N el número de muestras totales, produce el denominado método leave-one-out (LOO), que consiste en evaluar N veces el modelo, cada vez testando con una muestra distinta. También existe la posibilidad de utilizarla para optimizar parámetros durante el proceso de entrenamiento, o seleccionar modelos de entre varios posibles.

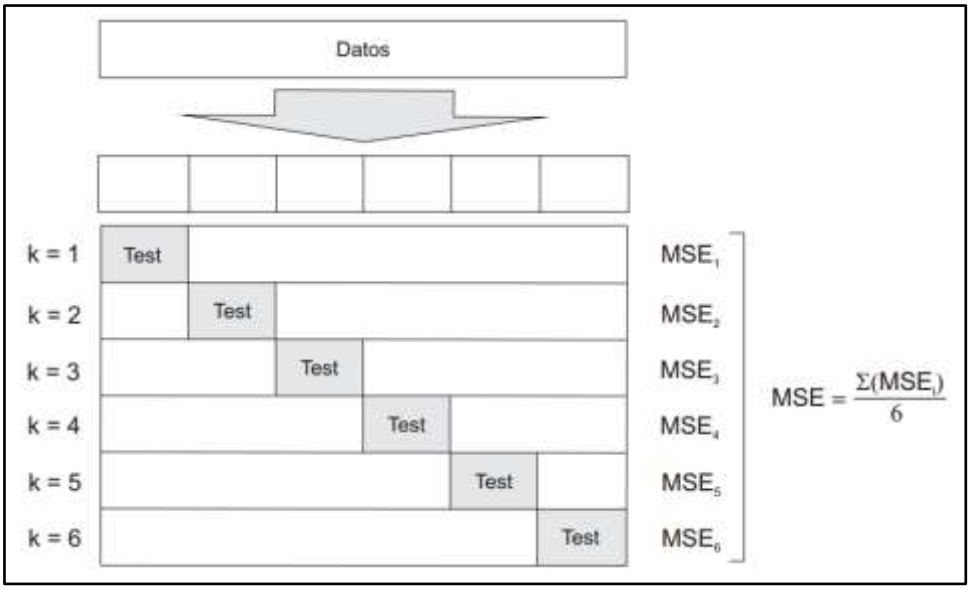


Figura 9
Esquema de una validación cruzada con k = 6 subconjuntos.

La validación cruzada por LOO funciona bien para funciones de error continuas, como el MSE, pero puede encontrar dificultades (además de las evidentes en cuanto a tiempo de computación) al utilizarse para funciones discontinuas, como las tareas de clasificación. Tampoco es recomendable para selección de modelos, ya que la falta de continuidad hace que un pequeño cambio en los datos pueda dar lugar a un cambio radical en el modelo escogido. Para la elección de subconjuntos de entradas en regresión lineal, Breiman y Spector encontraron que una validación cruzada k-fold con k = 10 o k = 5 daba mejor resultado que LOO. Kohavi también obtuvo mejores resultados para una validación cruzada 10-fold que para LOO para problemas de árboles de decisión. El método LOO a menudo se confunde con el método jackknife. Ambos tienen en común que las muestras de entrenamiento se omiten por turnos y se entrena la red sobre el conjunto restante. No obstante, la validación cruzada se utiliza para estimar el error de generalización, mientras que el jackknife se usa para

calcular el sesgo de una variable estadística de interés en cada subconjunto de datos. La media de esta variable estadística calculada para todos los subconjuntos se compara con la obtenida para todo el conjunto para estimar el sesgo. El jackknife también puede emplearse para calcular el error estándar de una variable. Por ejemplo, puede aplicarse al cálculo del sesgo del error de entrenamiento y, con ello, estimar el error de generalización, pero este proceso es más complicado que el LOO. Otro método de validación cruzada sería el llamado submuestreo aleatorio. Este método consiste en, dada una población de muestras, extraer un número determinado de ellas, entrenar la red con ellas y reservar el resto para testar. El proceso se repite un número determinado de veces, pudiéndose extraer varias veces la misma muestra para entrenar (se trata de un muestreo con reemplazamiento de tipo bootstrap [97]) y, finalmente, se promedia el error de test obtenido en cada una de las repeticiones. En cualquier caso, es obvio que la validación cruzada garantiza robustez frente a una elección sesgada de los conjuntos de entrenamiento y test. Otra pregunta que se puede plantear es: ¿cuándo se debe detener el entrenamiento? A priori la respuesta lógica sería que cuantas más iteraciones (normalmente llamadas épocas en la literatura) del algoritmo se hayan producido, mejor será el ajuste de los pesos y, por tanto, mejores prestaciones tendrá la red entrenada. Es evidente que esto se cumple para los datos de entrenamiento ya que una ANN de al menos una capa oculta puede aproximar, dadas suficientes iteraciones, cualquier conjunto de datos con una precisión arbitraria. Sin embargo, esto no se cumple por lo general para muestras nuevas. Se dice que una red se ha sobre-entrenado cuando “memoriza” las muestras que se le han presentado a la entrada durante el entrenamiento, de manera que incluso se ajusta al ruido existente en éstas

(ver Figura 9). Una red sobre-entrenada tendrá una mala generalización con respecto a muestras nuevas y, por lo tanto, es algo que hay que evitar deteniendo el entrenamiento a tiempo.

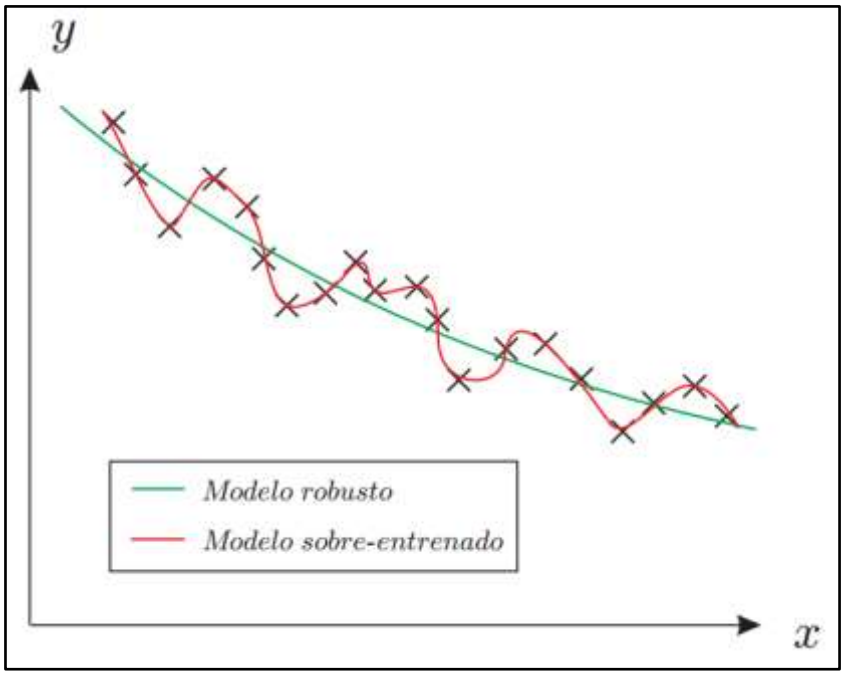


Figura 10
Modelo correctamente entrenado frente a modelo sobre-entrenado.

El modelo sobre-entrenado memoriza la forma de la función de entrada, modelando incluso el ruido de ésta. El modelo correctamente entrenado será más robusto frente al ruido y tendrá una manera de acometer el problema de sobre-entrenamiento (overfitting) es extraer un subconjunto de muestras del conjunto de entrenamiento (nótese que el conjunto de test se ha extraído previamente) y utilizarlo de manera auxiliar durante el entrenamiento.

Este subconjunto recibe el nombre de conjunto de validación. La función que desempeña el conjunto de validación es evaluar el error de la red tras cada época (o tras cada cierto número de épocas) y determinar el momento en que éste

empieza a aumentar. Ya que el conjunto de validación se deja al margen durante el entrenamiento, el error cometido sobre él es un buen indicativo del error que la red cometerá sobre el conjunto de test. En consecuencia, se procederá a detener el entrenamiento en el momento en que el error de validación aumente y se conservarán los valores de los pesos de la época anterior. Este criterio de parada se denomina early-stopping. La Figura 10 describe el procedimiento explicado.

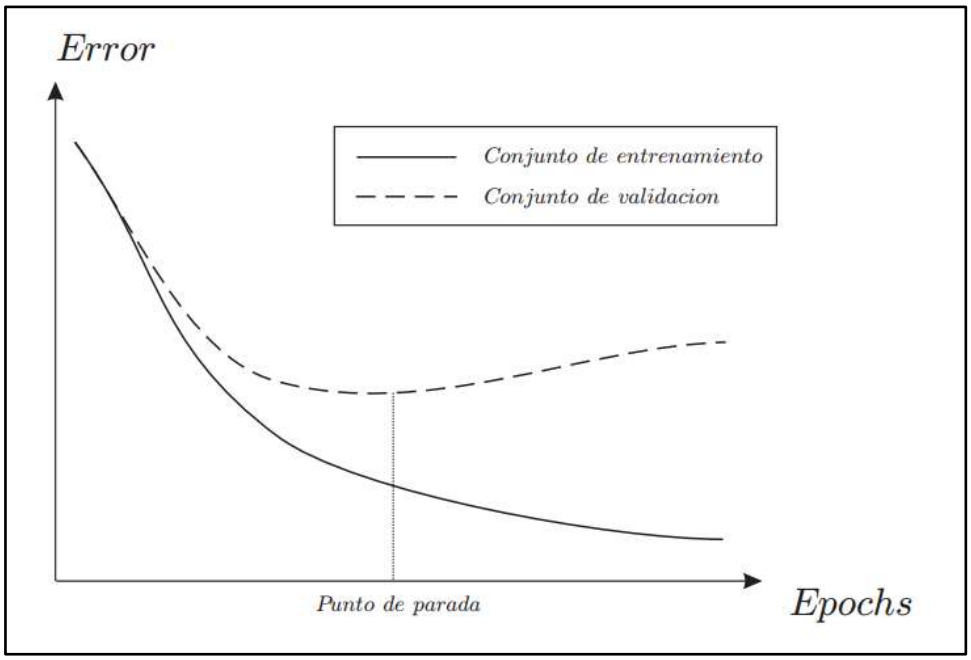


Figura 11
Representación del criterio de parada por early-stopping.

2.2.5.2 Entrenamiento supervisado y no supervisado

En relación a su manera de aprender, las ANNs pueden ser clasificadas en dos grupos:

- Supervisadas.
- No supervisadas

2.2.5.2.1 Entrenamiento supervisado:

Este es el tipo más popular de entrenamiento y el que utilicé en esta tesis. En este modo de entrenamiento, se puede considerar intuitivamente que existe un "profesor" que "enseña" a la red, proporcionando información para llevarla a emular la función deseada. Supongamos que un conjunto de muestras de entrada se aplica a la entrada de la red. Entonces la respuesta a la salida se compara con la respuesta deseada que proporciona el "profesor". Éste informa a la red, ya sea el resultado correcto o incorrecto, para actualizar los pesos en consecuencia. Para este tipo de aprendizaje hace falta conocer la salida deseada de las muestras de entrenamiento.

2.2.5.2.2 Entrenamiento no supervisado:

Las ANNs no supervisadas también reciben el nombre de redes auto-organizativas o, más frecuentemente, redes competitivas. Estas redes de aprendizaje competitivo no cuentan con la guía de un "profesor". Su base de funcionamiento son las técnicas de clustering o agrupamiento. Su función es agrupar muestras con características similares en el mismo cluster. Algunas redes no supervisadas son los SOM, las LVQ, los clasificadores k-NN3 y las RBFN (en su primera fase de entrenamiento). La idea inherente de todas estas redes es que la capa oculta de neuronas debe ser capaz de extraer las características estadísticas del conjunto de datos de entrada. En la mayoría de los casos las neuronas de la capa oculta compiten entre ellas para determinar cuál está más próxima al dato de entrada, y la neurona "ganadora" es la que se actualiza, lo que significa que se mueve más cerca del dato de entrada. Este tipo de red suele denominarse "winner-takes-all"

2.2.5.3 El algoritmo backpropagation

Para las ANNs que tienen funciones de activación diferenciables existe un método potente y computacionalmente eficiente denominado backpropagation (BP), o de “retropropagación”, para hallar las derivadas de la función de error con respecto a los pesos y umbrales de la red. En el método BP existe una función de error a minimizar, como puede ser el MSE entre las salidas obtenidas y las deseadas, y el algoritmo actúa modificando los pesos y umbrales en cada época. De manera adicional, puede incluir un término de complejidad que refleja una distribución previa sobre los valores que estos parámetros pueden tomar. Se asume que la propagación de la señal tiene dos fases, la fase forward, que es la fase de funcionamiento normal, en la que las señales se propagan desde las entradas hacia las salidas, y la fase backward, en la cual una vez obtenidas las estimaciones de salida se realiza la corrección de los pesos y umbrales en sentido contrario, es decir, desde las salidas hacia las entradas. El algoritmo BP, para el modo on-line, es decir, con actualización secuencial de los pesos, se resume en los siguientes pasos para una red de L capas:

1) Inicialización: Asumiendo que no existe información disponible a priori, se toman los pesos y umbrales de una distribución uniforme de media cero y cuya varianza hace que los productos entre pesos y entradas de las neuronas estén situados entre la zona lineal y la de saturación de las funciones de activación sigmoidales.
2) Presentación de las muestras de entrenamiento: Se presentan a la entrada de la red las muestras de entrenamiento correspondientes a una época. Para cada muestra se realiza la secuencia de operaciones en modo forward y backward indicadas en los puntos 3 y 4, respectivamente.

3) Fase forward: Denotemos una muestra correspondiente a la época n como ($x_{(n)}$, $d_{(n)}$), siendo el vector $x_{(n)}$ presentado a las entradas de la ANN, y $d_{(n)}$ el vector de salidas deseadas, presentado a la salida de la ANN. Se realiza la fase forward, capa por capa, según la ecuación que calcula el denominado "campo local inducido", $v_j^{(l)}(n)$, de la neurona j de la capa l:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \qquad (2.16)$$

donde m_0 es el número de neuronas en la capa anterior (l − 1), y $y_i^{(l-1)}$ es la salida de la neurona i de la capa l − 1 en la época n y $w_{ji}^{(l)}$ es peso de la conexión entre la neurona j de la capa l procedente de la neurona i de la capa l − 1. Para i = 0, se tiene $y_0^{(l-1)} = 1$ y, por tanto, $w_{j0}^{(l)} = b_j^{(l)}(n)$ es el umbral aplicado a la neurona j de la capa l. Asumiendo que se utiliza una función de activación sigmoidal, la salida de la neurona j de la capa l es:

$$y_j^{(l)} = \varphi_j(v_j(n)) \qquad (2.17)$$

En el caso particular de la capa de entrada (l=0) se tiene:

$$y_j^{(0)}(n) = x_j(n) \qquad (2.18)$$

donde $x_j(n)$ es el elemento j-ésimo del vector de entrada x(n). En el caso de la capa de salida (l = L) se cumple que para la neurona j la salida equivale al valor predicho p_j:

$$y_j^{(L)} = p_j(n) \qquad (2.19)$$

Se calcula la señal de error como:

$$e_j(n) = d_j(n) - p_j(n) \qquad (2.20)$$

siendo $d_j(n)$ el j-ésimo elemento de la salida deseada d(n).

4) Fase backward: Se calculan los gradientes locales (deltas) de la red por medio del método de descenso de gradiente, definidos como:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)), & \text{para la neurona } j \text{ de la capa de salida } L \\ \varphi_j'(v_j^{(l)}(n)) \sum \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), & \text{para la neurona } j \text{ de la capa oculta } l \end{cases} \quad (2.21)$$

donde $\phi'_j(\cdot)$ expresa la derivada de ϕ_j respecto al argumento. Los pesos se ajustan de acuerdo con la ecuación siguiente:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n) y_i^{(l-1)}(n) \quad (2.22)$$

donde η es la tasa de aprendizaje y α la constante de momento. La tasa de aprendizaje está relacionada con la trayectoria que toma el algoritmo BP en el espacio de los pesos por el método de descenso de gradiente. Cuando η es pequeño, la variación de los pesos entre iteraciones será pequeña, haciendo más “suave” la trayectoria, pero también haciendo más lento el entrenamiento. Para η grandes, la variación de los pesos será grande, acelerando el entrenamiento. Sin embargo, si η es demasiado grande, se corre el riesgo de que la red oscile y sea inestable. Existen métodos que mantienen la tasa de aprendizaje constante y otros que la hacen adaptativa según la evolución del error de una época a otra. Por su parte, el término de momento ayuda a evitar las oscilaciones producidas por las variaciones de los pesos en la función de error, que no decrece de manera monótona. El descenso de gradiente modificado con un término de momento se utiliza a veces, así como el método del gradiente conjugado, e incluso métodos de segundo orden.

5) Iteración: Se realizan los cálculos forward y backward de los puntos 3 y 4 presentando las muestras de entrenamiento de cada época a la entrada de la red hasta que el criterio de parada se cumpla.

2.3. Marco Conceptual

- ✓ EPA: Ácido eicosapentaenoico.
- ✓ DHA: Ácido docosahexaenoico.
- ✓ Triglicéridos (TG): Son la forma molecular natural de las grasas que se encuentran en los alimentos. Por ejemplo los omega-3 presentes en el pescado EPA & DHA están como triglicéridos. Consisten de una columna vertebral de glicerol unida a tres ácidos grasos. En el caso del aceite de pescado, los ácidos grasos presentes mayoritariamente son el eicosapentanoico (EPA) y docosahexanoico (DHA). Los ácidos grasos se oxidan rápidamente por lo que la columna de glicerol ayuda a estabilizar las moléculas de grasas y evitar la oxidación.

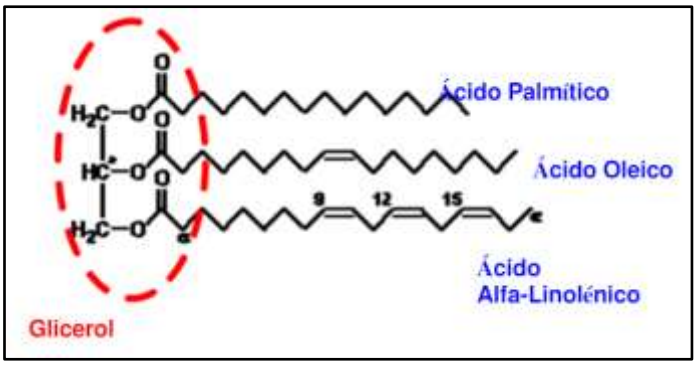


Figura 12
Estructura del triglicérido.

- ✓ Etil Ester (EE): Son una forma alternativa de grasas que son derivadas al reaccionar ácidos grasos libres con etanol. Durante este proceso, algunos ácidos grasos son separados de su columna natural de glicerol y

vinculados a una molécula de etanol. En este proceso llamado trans-esterificación se producen los etil ésteres de aceite de pescado (fish oil EE).

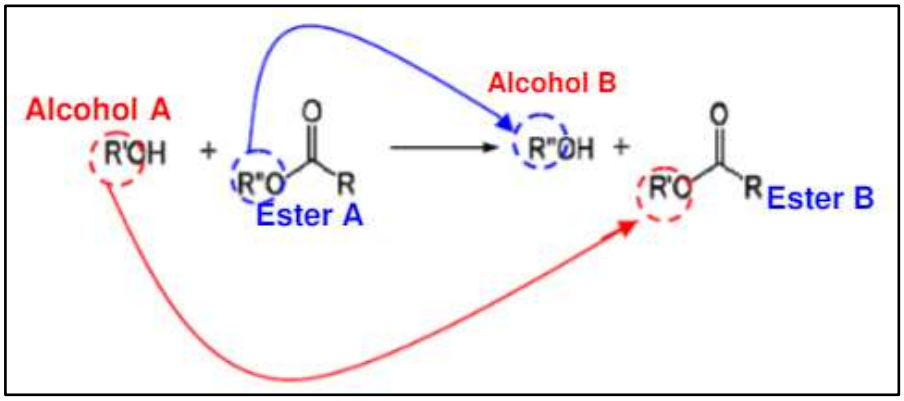


Figura 13
Estructura del etil ester.

A pesar de que tanto los ácidos grasos y el etanol son sustancias naturales, la esterificación de estas dos sustancias no se encuentra en ningún recurso alimenticio natural de los omega-3.

- ✓ Época: Conjunto de entrenamiento durante el proceso de aprendizaje.
- ✓ Umbral (threshold): Representa el grado de inhibición de la neurona, es un término constante que no depende del valor que tome la entrada.

CAPITULO III

DESARROLLO DE LA METODOLOGÍA

3.1. Análisis del sistema

3.1.1. Situación Actual:

El proyecto de tesis se centra en el análisis del aceite de pescado de Omega 3, en donde la misma cuenta con varios componentes siendo un total de 44 parámetros propios de la materia prima los cuales son:

Tabla 1
Perfil de aceite de pescado de Omega 3.

ITEM		CADENA
01	**MIRISTICO C14:0**	C14:0
02	**MIRISTOLEICO C14:1**	C14:1
03	**PENTADECYLICACID C15:0**	C15:0
04	**PALMITICO C16:0**	C16:0
05	**C16:1n11**	C16:1n11
06	**C16:1n5**	C16:1n5
07	**PALMITOLEICO C16:1n7**	C16:1n7
08	**I-17:0**	i-17:0

09	**C17:0**	C17:0
10	**HEXADECADIENOICO C16:2n4**	C16:2n4
11	**C17:1**	C17:1
12	**HEXADECATRIENOICO C16:3n4**	C16:3n4
13	**ESTEARICO C18:0**	C18:0
14	**C16:4n1**	C16:4n1
15	**OLEICO C18:1n9**	C18:1n9
16	**VACCENICO C18:1n7**	C18:1n7
17	**LINOLEICO C18:2n6**	C18:2n6
18	**C18:2n4**	C18:2n4
19	**C18:3n6**	C18:3n6
20	**OCTADECATRIENOICO C18:3n4**	C18:3n4
21	**ALFA LINOLENICO C18:3n3**	C18:3n3
22	**ARAQUIDICO C20:0**	C20:0
23	**MOROTICO C18:4n3**	C18:4n3
24	**EICOSENOICO C20:1n9**	C20:1n9
25	**C18:4n1**	C18:4n1
26	**C20:1n7**	C20:1n7
27	**EICOSADIENOICO C20:2n6**	C20:2n6
28	**C20:2n9**	C20:2n9
29	**ARAQUIDONICO C20:4n6**	C20:4n6
30	**EICOSATRIENOICO C20:3n3**	C20:3n3
31	**BEHENICO C22:0**	C22:0
32	**CETOLEICO C22:1n11**	C22:1n11
33	**EICOSATETRAENOICO ETA C20:4n3**	C20:4n3
34	**ERUCICO C22:1n9**	C22:1n9
35	**EICOSAPENTAENOICO EPA C20:5n3**	C20:5n3
36	**C22:1n7**	C22:1n7
37	**HENEICOSAPENTAENOICO HPA C21:5n3**	C21:5n3
38	**C22:5n6**	C22:5n6
39	**LIGNOCERICO C24:0**	C24:0
40	**C22:4n6**	C22:4n6
41	**C22:4n3**	C22:4n3
42	**NERVONICO C24:1n9**	C24:1n9
43	**DOCOSAPENTAENOICO DPA C22:5n3**	C22:5n3
44	**DOCOSAHEXAENOICO C22:6n3 - DHA**	C22:6n3

Este perfil de aceite tiene una codificación como tal la cual se conforma de la siguiente manera:

M1712E89,

Donde:

M: Etapa del aceite (Materia Prima, Destilado y Residuo), en donde dependiendo si el aceite es materia prima, destilado o residual se le coloca la primera letra de su etapa.

17: Es el valor porcentual que toma el EPA.

12: Es el valor porcentual que toma el DHA.

E: Es el tipo de aceite (Crudo, Esterina, Oleína), colocándose la primera letra del tipo en la codificación del aceite.

89: Es el % de Conversión del Etil Ester.

Este aceite en cuestión es sometido a ciertas condiciones de transformación, las cuales son las siguientes:

Tabla 2
Variables de transformación para el aceite de pescado.

Variable de Transformación	Unidad
Date	DD.MM.YYYY
Start of Test	hh:mm:ss
End of Test	hh:mm:ss
Duration	Min
Dosing Vessel	°C
Product -> Evaporator	°C
Product -> Evaporator (II)	°C
Evaporator heating	**°C**
Condenser	°C
Trace Heating Residue	°C
Cold Trap	°C
Vacuum stage	mbar
Wiper Speed	1/min
Flow	kg/h
Residue	**%**
Distillate	**%**
Cold Trap (I)	**%**
Residue gross (II)	g
Residue tare (II)	g
Residue net (II)	g
Distillate gross (II)	g

Distillate tare (II)	g
Distillate net (II)	g
Dosing Pump Feed	Hz
Residue Pump (I)	Hz
Distillate Pump (I)	Hz
Residue Pump (II)	Hz
Distillate Pump (II)	Hz

Ahora toda la información de las transformaciones que se realizan por cada test, lo tienen almacenado en archivos Excel, en donde la administrabilidad y el análisis de datos del mismo no son de gran ayuda a los analistas de dicha empresa.

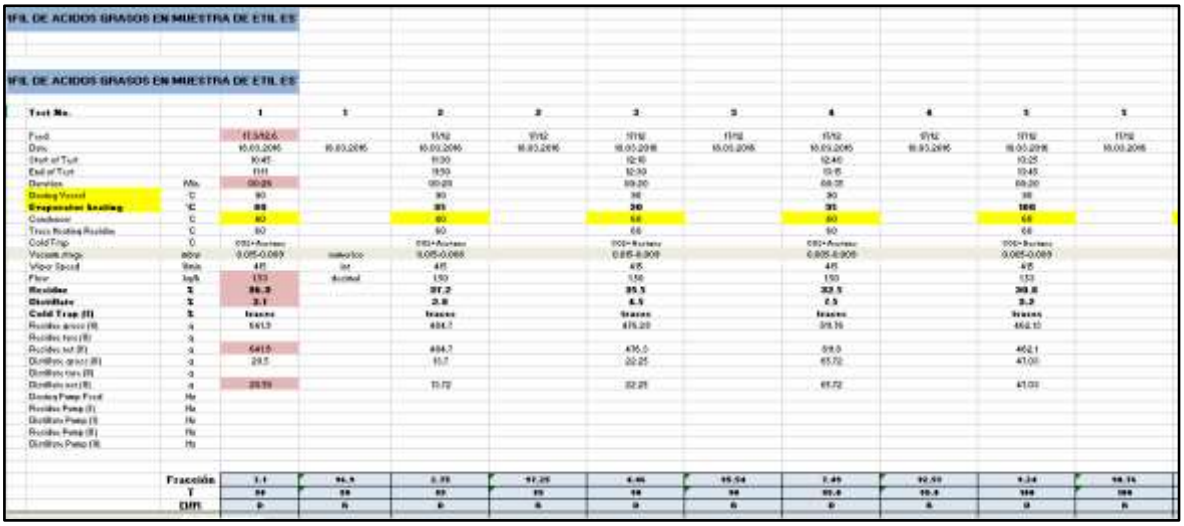


Figura 14
Fuente de datos actual de perfil y transformación de aceites.

El proceso entonces para el refinamiento y generación es el siguiente:

1) Se escoge un aceite materia prima para que pase por unas condiciones de transformación.
2) Cada transformación realizada es un test, todos estos test realizados a un aceite forman parte de un paso.

3) Luego de ello los ingenieros químicos deben escoger un aceite para volver a pasarlo por otra transformación, y su proceso de elección del mejor aceite sería: mayor % EPA, mayor % DHA y mayor rendimiento.

4) Este proceso continuará hasta que encuentren un aceite con alta concentración de EPA y DHA, el cual se ajuste a lo solicitado por el cliente.

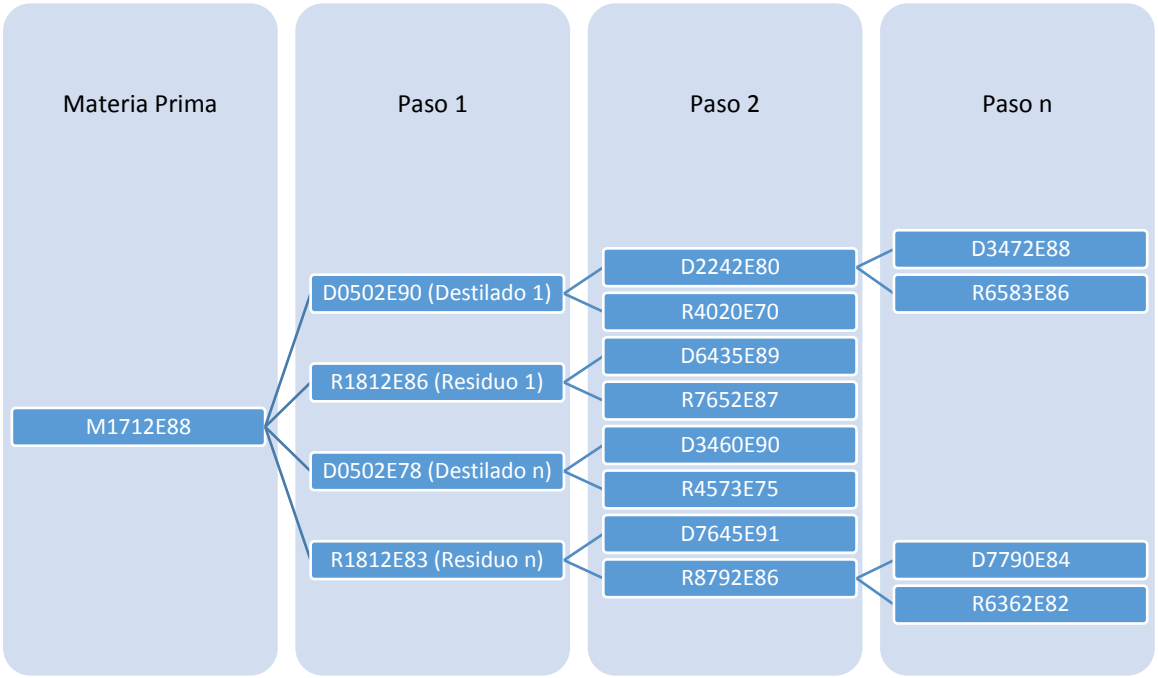


Figura 15
Proceso de Refinamiento de aceite

3.1.2. Necesidad:

La necesidad actual que presenta la empresa es el poder saber qué nuevo tipo de aceite se generará luego de someter a un perfil de aceite a ciertas condiciones (variables de transformación). Esto sería de gran ayuda puesto se ahorraría mucho tiempo y costes.

3.1.3. Planteamiento de solución:

Analizando los registros actuales con los que se cuentan tomamos como datos de muestra los más representativos para la empresa que en este caso son: EPA, DHA, Temperatura, Tiempo y el TIPO de aceite (campo que se quiere predecir).

Ahora como herramienta tecnológica que se utilizó para la programación de la red neuronal fue R Studio utilizando el lenguaje estadístico R, además aproveche el uso del paquete neuralnet (para creación de la red neuronal) y el aprendizaje de la red.

El proceso que seguí para la implementación de la solución realizada fue la siguiente:

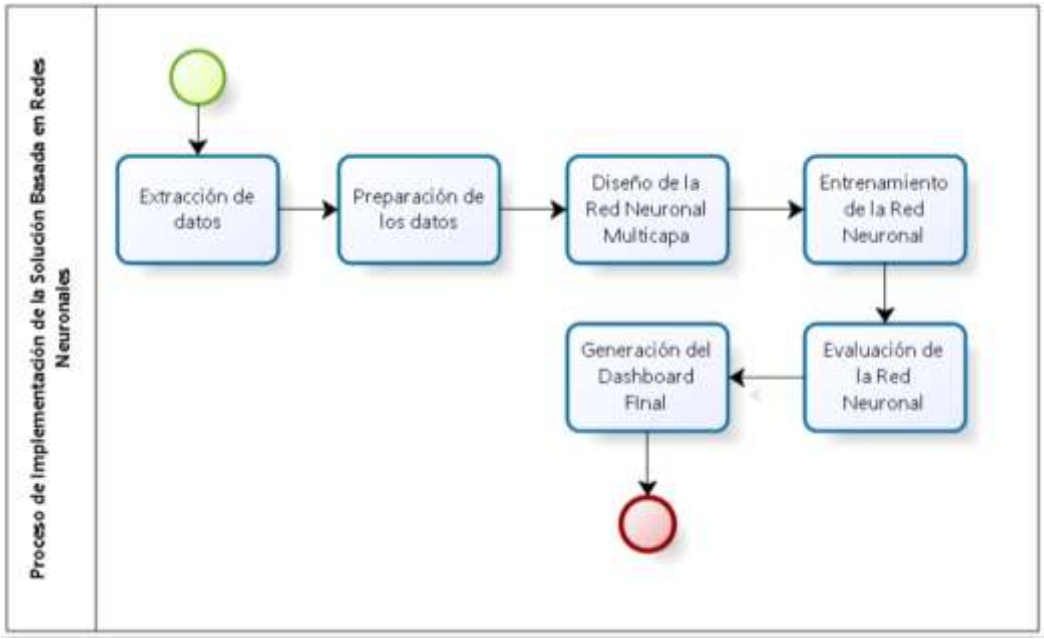


Figura 16
Proceso de Implementación de la Solución Basada en Redes Neuronales

3.2. Construcción del modelo predictivo.

3.1.1. Etapas:

- *Extracción de datos:*

Para ello contamos con una base de datos de aproximadamente 7068 registros.

Tabla 3
Fuente de Datos inicial

	EPA	DHA	Temperatura	Tiempo	TIPO
1	18	20	80	50	D1115R2023
2	18	20	85	50	D1316R2224
3	18	20	90	80	D1620R2628
4	18	20	95	60	D1719R2727
5	18	20	100	40	D1818R2826
6	18	20	105	60	D2121R3129
7	18	20	110	70	D2323R3331
8	18	20	115	80	D2625R3633
9	18	20	120	20	D2620R3528
10	18	20	125	100	D3029R4137
11	18	20	130	30	D3023R3931
12	18	20	135	70	D3328R4336
13	13	9	80	40	D0503R1511
14	13	9	85	20	D0702R1610
15	13	9	90	20	D0903R1811
16	13	9	95	50	D1207R2115
17	13	9	100	70	D1410R2418
18	13	9	105	90	D1713R2721

Toda esta data es cargada al Rstudio, para luego separar la data de la siguiente manera: Datos de entrenamiento (70%) y datos de test (30%). Cabe mencionar que la data que se tiene actualmente en la base de datos (sistematizado) sólo corresponde al paso 1, además de ello como ya se sabe cada aceite al ser transformado te genera 2 tipos de aceites, uno destilado y otro residual, ahora para reducir la cantidad de registros, y predecir de mejor manera

un aceite, se concatenó los códigos nuevos de aceites destilados y residuos, en un campo llamado TIPO.

- *Preparación de los datos:*

En este punto lo que se realiza es poder estandarizar todos los datos a tipo numérico (no valores categóricos tal es el caso del campo TIPO), luego de ello para que la predicción de la red neuronal sea más fina es necesario escalar o normalizar los datos, en este caso utilizaré el proceso de escalamiento de los datos (multiplicar por un factor a los datos que se tienen).

Tabla 4
Tabla de Entrenamiento Escalada

	EPA	DHA	Temperatura	Tiempo	TIPO
5560	-0.62216695413	-0.4177307447	-0.7260187509	-0.01221688551	-0.95787103117
5076	1.30249792680	-0.1258090052	1.5971473298	1.54744589206	1.95076401508
594	-0.94294443429	-0.1258090052	-0.1452272307	-0.79204827430	-0.68989697701
1843	-1.58449939460	-1.5854177028	0.1451685294	-0.01221688551	-0.56707553551
5680	1.30249792680	0.7499562134	-0.7260187509	1.54744589206	0.04982306835
135	-1.58449939460	0.7499562134	-1.0164145110	-1.18196396870	-1.66409431975
654	-0.30138947398	-0.7096524843	-0.1452272307	-0.40213257991	-0.29631008495
3444	-0.30138947398	-0.4177307447	1.5971473298	-1.57187966309	1.19429559135
2441	0.66094296649	0.1661127343	-0.4356229908	1.15753019767	0.03028329356
5838	-1.26372191444	1.0418779529	-0.1452272307	0.37769880888	-0.63406904905
5655	-1.26372191444	-0.1258090052	-1.0164145110	-0.40213257991	-1.44915679714
5718	0.66094296649	1.0418779529	-0.1452272307	-0.40213257991	0.15589613145
1057	-0.94294443429	0.4580344739	-1.5972060312	-1.57187966309	-1.87624044597
2161	-0.30138947398	1.6257214320	-1.5972060312	1.15753019767	-1.39053747279
6314	0.01938800618	-0.1258090052	-1.3068102711	0.76761450328	-1.07231828347
4527	0.98172044665	0.1661127343	-1.0164145110	1.54744589206	-0.38563476967
1842	-1.58449939460	-1.5854177028	-0.1452272307	-0.01221688551	-0.84621517527
2167	-0.30138947398	1.6257214320	0.1451685294	-0.79204827430	-0.08137256234

Tabla 5
Tabla de Test Escalada

	EPA	DHA	Temperatura	Tiempo	TIPO
1	0.953700237845	1.6105804450	-1.5834106872	-0.34497856790	-1.048325447658
2	0.953700237845	1.6105804450	-1.2955054950	-0.34497856790	-0.794298454115
7	0.953700237845	1.6105804450	0.1440204663	0.42280689119	0.583521869559
9	0.953700237845	1.6105804450	0.7198308508	-1.49665675654	0.956278870954
10	0.953700237845	1.6105804450	1.0077360430	1.57448507984	1.497466813719
19	-0.629579508922	-1.6746488226	0.1440204663	0.03891416165	-0.153708644311
20	-0.629579508922	-1.6746488226	0.4319256585	-0.34497856790	0.103079512206
21	-0.629579508922	-1.6746488226	0.7198308508	1.19059235029	0.525537447120
26	1.270356187198	1.3119232389	-1.2955054950	-0.34497856790	-0.670046120317
31	1.270356187198	1.3119232389	0.1440204663	1.57448507984	0.867921655808
34	1.270356187198	1.3119232389	1.0077360430	0.42280689119	1.478138672906
45	-1.262891407629	-0.4800199980	0.7198308508	-1.49665675654	-0.012889332673
50	1.270356187198	0.1172944143	-1.2955054950	1.19059235029	-0.529226808679
51	1.270356187198	0.1172944143	-1.0076003027	1.19059235029	-0.258632837296
54	1.270356187198	0.1172944143	-0.1438847260	-1.11276402699	0.235615334924
55	1.270356187198	0.1172944143	0.1440204663	-1.49665675654	0.503448143333
59	1.270356187198	0.1172944143	1.2956412353	1.57448507984	1.779105436995
60	1.270356187198	0.1172944143	1.5835464276	-0.72887129745	1.748732644289

- *Diseño de la red neuronal multicapa:*

Para ello definimos cuáles serán las entradas y cuál es la variable objetivo, el número de capas ocultas que tendrá la red y las neuronas por cada capa, además de definir un valor de umbral (threshold).

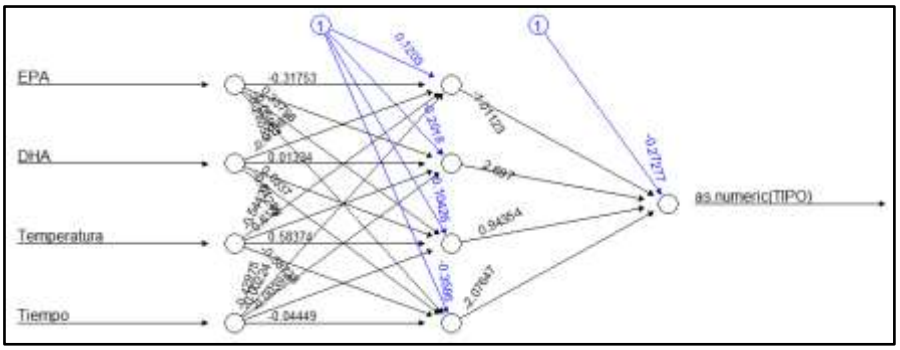


Figura 17
Topología de la red neuronal multicapa.

- ✓ **Entradas:** EPA, DHA, Temperatura y Tiempo.
- ✓ **Salida:** TIPO
- ✓ **Número de Capas Ocultas:** 1
- ✓ **Número de Neuronas:** 4
- ✓ **Umbral:** 0.04

- *Entrenamiento de la red neuronal:*

En este punto se emplea la data ya transformada y la función llamada neuralnet, entonces la configuración de mi red para el entrenamiento sería la siguiente:

```
ann <- neuralnet(as.numeric(TIPO) ~ EPA + DHA +
                 Temperatura + Tiempo , data= Train.scaled, hidden =4,
                 algorithm = "rprop+", threshold = 0.04, stepmax=1e6 )
```

Luego de ello podemos visualizar el resultado del entrenamiento de la red:

```
> ann
Call: neuralnet(formula = as.numeric(TIPO) ~ EPA + DHA + Temperatura +     Tiempo, data = Train.scaled, hid
den = 4, threshold = 0.04,     stepmax = 1000000, algorithm = "rprop+")

1 repetition was calculated.

        Error Reached Threshold Steps
1 2.570213137     0.03821348565  2245
```

Incluso podemos ver los pesos que se asignaron en cada capa.

```
> ann$weights
[[1]]
[[1]][[1]]
              [,1]           [,2]           [,3]           [,4]
[1,]  0.02394889202 -1.71562402747  0.96572299520  2.70996996789
[2,] -0.45498453727 -0.49737768768  0.46286281211 -0.82583864096
[3,] -0.04621252629 -0.10924990356 -0.02501150174 -0.05316460909
[4,] -1.05351825516 -1.22175839817  0.98660406391 -1.89355160825
[5,] -0.08939415988 -0.05622700594  0.21807836178 -0.27594417293

[[1]][[2]]
              [,1]
[1,]  0.7948068631
[2,] -1.6095588111
[3,] -0.7492837105
[4,]  1.4753233123
[5,] -0.9923839849
```

Figura 18
Entrenamiento – pesos por capas

Luego del proceso de entrenamiento de la red neuronal estamos listos para la siguiente etapa, de evaluación y testeo de la red.

- *Evaluación de la red neuronal:*

En esta etapa se realiza el testeo del modelo predictivo ya entrenado. Para ello se ingresa un conjunto de datos de test, para seguidamente poder analizar la validación con los datos reales y verificar el grado predicción que se logró.

```
output <- compute(ann, Test.scaled[ , c("EPA", "DHA", "Temperatura", "Tiempo")])
```

3.3. Revisión de Resultados:

Como último paso a realizar sería la revisión de los resultados los cuales se graficó en Qlikview dando estos valores:

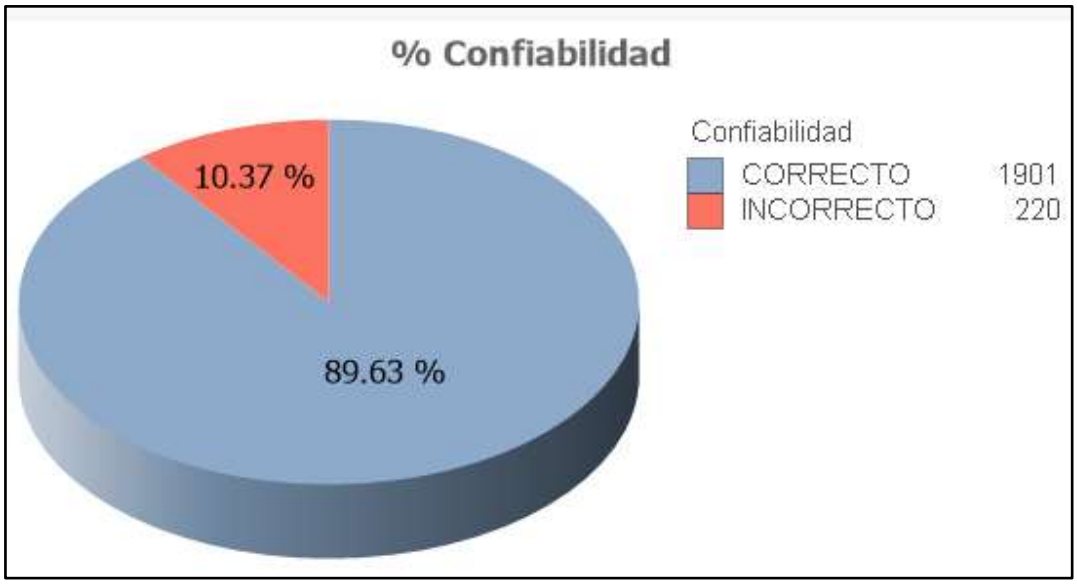
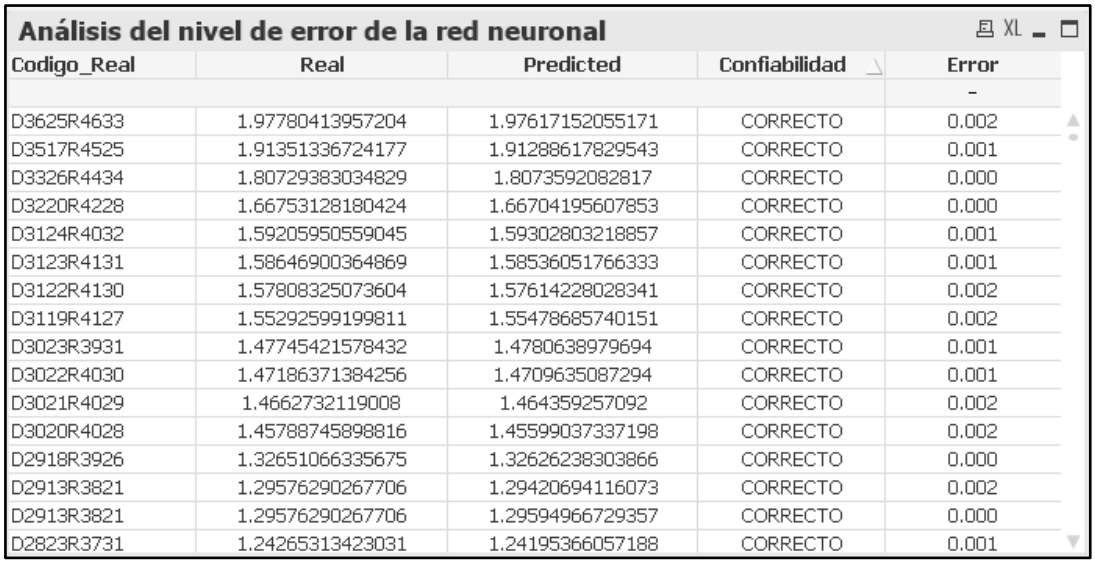


Figura 19
Resultados - % Confiabilidad

Además de ello podemos observar los valores que te indican si fue correcto o no la predicción.



Análisis del nivel de error de la red neuronal

Codigo_Real	Real	Predicted	Confiabilidad	Error
				-
D3625R4633	1.97780413957204	1.97617152055171	CORRECTO	0.002
D3517R4525	1.91351336724177	1.91288617829543	CORRECTO	0.001
D3326R4434	1.80729383034829	1.8073592082817	CORRECTO	0.000
D3220R4228	1.66753128180424	1.66704195607853	CORRECTO	0.000
D3124R4032	1.59205950559045	1.59302803218857	CORRECTO	0.001
D3123R4131	1.58646900364869	1.58536051766333	CORRECTO	0.001
D3122R4130	1.57808325073604	1.57614228028341	CORRECTO	0.002
D3119R4127	1.55292599199811	1.55478685740151	CORRECTO	0.002
D3023R3931	1.47745421578432	1.4780638979694	CORRECTO	0.001
D3022R4030	1.47186371384256	1.4709635087294	CORRECTO	0.001
D3021R4029	1.4662732119008	1.464359257092	CORRECTO	0.002
D3020R4028	1.45788745898816	1.45599037337198	CORRECTO	0.002
D2918R3926	1.32651066335675	1.32626238303866	CORRECTO	0.000
D2913R3821	1.29576290267706	1.29420694116073	CORRECTO	0.002
D2913R3821	1.29576290267706	1.29594966729357	CORRECTO	0.000
D2823R3731	1.24265313423031	1.24195366057188	CORRECTO	0.001

Figura 20
Resultados – Análisis del error de la red neuronal

Los valores predichos por la red neuronal esta siempre en un rango decimal (recordar que estos valores son los escalares), el cual para poder identificar a que aceite pertenece, se redondea su valor (el factor de cambio de un aceite a otro es de 0.02), entonces para poder saber si la predicción fue correcta realizamos la resta del valor real con el valor predicho y no debe exceder de dicho valor.

CONCLUSIONES

Las conclusiones del proyecto tesis son:

- Para la optimización del proceso de refinamiento de aceite de omega 3, se implementó una solución basada en redes neuronales con lo cual se logró brindar un conjunto de alternativas de aceites nuevos generados y así se logró tener un ahorro en costes y tiempo en relación a la búsqueda del aceite requerido.

- Se diseñó el modelo de la red neuronal multicapa basándonos en la información histórica presentada por la empresa, teniendo como datos entrada el %EPA, %DHA, temperatura y tiempo, y la variable objetivo el tipo de aceite (lo que se quiere predecir).

- Se entrenó la red neuronal con un conjunto de datos al azar de la data real (70% del total de datos), con el fin de poder tener el modelo ideal de la red (esto en relación a los pesos de cada neurona de todas las capas).

- Se evaluó los resultados de la red neuronal basándonos de un conjunto de datos al azar de la data real (30% del total de datos), obteniendo un porcentaje de confiabilidad de la red neuronal del 89.63%, con estos resultados podemos concluir que se logró con el objetivo planteado por la empresa (teniendo presente que un modelo predictivo siempre maneja % error de la red, la cual

puede disminuir en función a la cantidad y calidad de data con la que se cuenta y se entrena a la red neuronal).

- Para la visualización de los datos generados se diseñó un dashboard (en la herramienta QlikView) donde se muestra toda la red de aceites generados a partir de un aceite inicial y sus variables de transformación para cada etapa.

- Concluyo además que las redes neuronales son útiles para cualquier problema de clasificación, siempre y cuando haya data suficiente para el entrenamiento y que esta data sea una buena representación de todo el universo de la data.

RECOMENDACIONES

El presente proyecto queda como referencia para próximos proyectos relacionados a temas de clasificación, aportando la integración de 2 grandes herramientas tal es el caso de R (lenguaje de programación estadístico), el cual cuenta con 2 muy buenos paquetes para redes neuronales: “neurnalnet” y “nnet”, y la integración de la solución con la herramienta reporteadora Qlikview, el cual te brinda la facilidad de la integración con dicho lenguaje de programación (R).

Considerando de suma importancia también el poder conocer los conceptos básicos de redes neuronales y aprendizaje automático (utilizando el algoritmo de backpropagation). Todos estos conocimientos deben estar cimentados con conocimientos previos de estadística y matemática, esto para que la curva de aprendizaje no sea tan tediosa y para un mayor entendimiento de los temas en cuestión.

BIBLIOGRAFÍA

- ✓ Christopher M. Bishop. (2007). Pattern Recognition and Machine Learning. New York: Springer.
- ✓ Stuart Rusell and Peter Norvig. (2003). Artificial Intelligence a Modern Approach, Third Edition. Pearson.
- ✓ R.O. Duda, P.E. Hart, and D.G. Stork. Pattern Classification. John Wiley and Sons, New York, 2nd edition, 2001.
- ✓ Statsoft. Statsoft Electronic Textbook, 2003.
- ✓ W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys., 5(6,12a):115–133, 1943.
- ✓ F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psych. Rev., 65:386–408, 1958.
- ✓ D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values ot the adaptive weights. Proc. of the International Joint Conference on Neural Networks, IJCNN 1990, 3:21–26, 1990.
- ✓ A. Sorjamaa, N. Reyhani, and A. Lendasse. Input and structure selection for k-NN approximator. In J. Cabestany, A. Prieto, and F.S. Hernández, editors, Lect. Notes. Comput. Sci., volume 3512, pages 985–992. Springer, Berlin - Heidelberg, 2005.
- ✓ E.Z. Panagou and V.S. Kodogiannis. Application of neural networks as a non-linear modelling technique in food mycology. Expert Syst. Appl., 36(1):121–131, 2009.