



Universidad de Valladolid



**ESCUELA DE INGENIERÍAS
INDUSTRIALES**

**UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES**

Grado en Electrónica Industrial y Automática

**Diseño de sensores software no lineales para la estimación
de variables de calidad de los procesos**

Autor:

Ngua Ayecaba, Adolfo Cursillo Ngua

Tutor:

**De la Fuente Aparicio, María Jesús
Departamento de Ingeniería de Sistemas y Automática (ISA)**

Valladolid, febrero de 2019

Dedicatoria: este trabajo va dedicado a mis padres: Adolfo Ngua Nguema Mofuman y Gloria Ayecaba Ndong; a mi hermano Alfredo Ngua Ayecaba; a mi tío Salvador Nzi Ngua, y a mis hijos.

ÍNDICE DE CONTENIDOS.

CAPÍTULO I. INTRODUCCIÓN, OBJETIVOS, MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO.	10
1.1.- MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO.	10
1.2.- INTRODUCCIÓN Y OBJETIVOS.	10
1.2.1.- Introducción.	10
1.2.2.- Objetivos	11
1.3.- ORGANIZACIÓN DE LA MEMORIA.	12
CAPÍTULO II. TÉCNICAS Y MÉTODOS DE ESTUDIO: MARCO TEÓRICO DEL PROYECTO.	16
2.1.- CONCEPTO DE ESTADÍSTICA APLICADA AL ESTUDIO DE CALIDAD DE LOS PROCESOS.	16
2.2.- SENSORES SOFTWARE.	17
2.2.1.- Clasificación de los sensores software.	18
2.2.2.- Metodología de desarrollo de un sensor software.	19
2.2.2.1.- Primera inspección de datos.	20
2.2.2.2.- Selección de datos históricos e identificación de estados estacionarios.	21
2.2.2.3.- Preprocesamiento de datos.	21
2.2.2.4. Selección del modelo, entrenamiento y validación.	21
2.2.2.5.- Mantenimiento del sensor software	22
2.2.3.- Aplicaciones del sensor software	22
2.3.- REDES NEURONALES ARTIFICIALES.	23
2.3.1.- Clasificación de las redes neuronales artificiales.	26
2.3.1.1.- Clasificación según la arquitectura o topología de la red.	27
2.3.1.1.1.- Redes neuronales artificiales monocapa	27
2.3.1.1.2.- Redes neuronales artificiales multicapa	28
2.3.1.2.- Clasificación según el sentido en el que fluye la información	29
2.3.1.2.1.- Redes neuronales artificiales con conexiones hacia adelante o perceptrón multicapa (MLP)	29
2.3.1.2.2.- Redes neuronales artificiales recurrentes.	30
2.3.1.3.- Clasificación según su aprendizaje.	31
2.3.1.3.1.- Redes neuronales artificiales de aprendizaje supervisado	31
2.3.1.3.2.- Redes neuronales artificiales de aprendizaje no supervisado.	32
2.3.1.3.3.- Aprendizaje híbrido.	33
2.3.2.- Red perceptrón simple.	33

2.3.3.- Perceptrón multicapa	34
2.3.4.- Redes neuronales de funciones de base radial (RFBR)	35
2.3.4.1.- Entrenamiento y aprendizaje de la red.....	38
2.3.5.- Red neuro difusa ANFIS.....	40
2.3.6.- Ventajas e inconvenientes de las redes neuronales artificiales.	42
2.3.6.1.- Inconvenientes.....	42
2.3.6.2.- Ventajas.....	42
2.3.7.- Algunas aplicaciones de las redes neuronales	43
2.4.- MÁQUINAS DE VECTORES DE SOPORTE PARA REGRESIÓN	43
2.4.1.- Introducción.....	43
2.4.2.- Máquinas de soporte vectorial para regresión (SVR).	44
2.4.2.1. Caso cuasi ajustable linealmente	44
2.4.2.2. Caso no ajustable linealmente (Uso del kernel en las SVR)	46
2.5. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA).....	47
2.5.1.- Introducción	47
2.5.1.- Determinación de los componentes principales.	47
CAPÍTULO III. ESTACIÓN DEPURADORA DE AGUAS RESIDUALES (EDAR).....	50
3.1.- CONCEPTO Y DESCRIPCIÓN GENERAL DE UN PROCESO DE DEPURACIÓN DE AGUAS RESIDUALES.....	50
3.2 DESCRIPCIÓN DE LA PLANTA BSM2.....	51
3.2.1.- Descripción de las variables medidas en la planta BSM2	53
CAPÍTULO IV. RESULTADOS, ANÁLISIS Y DISCUSIÓN.....	56
4.1.- RECOPIACIÓN DE DATOS EN LA EDAR BSM2.	57
4.2.- PROCESAMIENTO DE DATOS.....	57
4.3.- REDES NEURONALES ESTÁTICAS	58
4.3.1.- Reducción dimensional con el coeficiente de correlación (R).	59
4.3.1.1.- Redes perceptrón multicapa (MLP) reduciendo variables con el coeficiente de correlación (R).	59
4.3.1.2.- Redes de funciones de base radial (RBF) reduciendo variables con el coeficiente de correlación (R).	63
4.3.1.3.- Redes anfis reduciendo variables con el coeficiente de correlación (R).	67
4.3.1.4.- Máquinas de vectores de soporte (SVM) con el coeficiente de correlación (R).	70
4.3.2.4.- Combinación de las mejores redes estáticas reduciendo las variables con el coeficiente de correlación (R).....	73

4.3.2.- Reducción dimensional basado en el análisis de componentes principales (PCA).....	78
4.3.2.1.- Red perceptrón multicapa (MLP) con PCA.	79
4.3.2.2.- Red de funciones de base radial (RBF) con PCA	81
4.3.2.3.- Red neuro-difusa anfis con PCA.....	83
4.3.2.4.- Máquinas de vectores de soporte (regresión de vectores de soporte) con PCA.	85
4.3.2.5.- Combinación de las mejores redes con PCA.....	86
4.3.3.- Comparación de las redes estáticas.....	88
4.4.- REDES NEURONALES ARTICIALES DINÁMICAS.....	90
4.4.1.- Reducción dimensional del conjunto de variables con R.	90
4.4.1.1.- Red perceptrón multicapa (MLP) dinámica reduciendo variables con R. 91	
4.4.1.2.- Red de funciones de base radial (RBF) reduciendo variables con R.....	94
4.4.1.3.- Red anfis reduciendo las variables con R.....	96
4.4.1.4.- SVR dinámica reduciendo variables con R.....	100
4.4.1.5.- Combinación de las redes dinámicas reduciendo los datos con R.	103
4.4.2.- Reducción dimensional utilizando el PCA.....	106
4.4.2.1.- Red MLP dinámica reduciendo variables con PCA	106
4.4.2.2.- Red RBF dinámica reduciendo variables con PCA	109
4.4.2.3.- Red ANFIS dinámica reduciendo variables con PCA	112
4.4.2.4.- SVR dinámica reduciendo variables con PCA	115
4.4.2.5.- Combinación de redes dinámica reduciendo variables con PCA.....	118
4.4.3.- Comparación de las redes dinámicas.	121
4.5.- COMPARACIÓN DE LAS MEJORES REDES ESTÁTICAS Y DINÁMICAS.....	122
CAPÍTULO V: CONCLUSIONES Y TRABAJO FUTURO	126
5.1- CONCLUSIONES.....	126
5.2.- TRABAJO FUTURO.....	126
CAPÍTULO VI. BIBLIOGRAFÍA	130

Agradecimientos:

A mi familia, La Casa Ngua, en especial, a mis finados padres, Adolfo Ngua Nguema Mofuman y Gloria Ayecaba Ndong, sin ellos no sería nadie en la vida.

Un especial agradecimiento a mis hermanos: Gabriel Ngua Ayecaba, Rodolfo Mba Ngua Ayecaba, Agueda Pilar Ngua Ayecaba, Reginaldo Ricardo Milam Ngua Ncham y Francisco Andrés Ngua Ayecaba, por su confianza, apoyo incondicional y emocional durante la carrera. Vuestra implicación fue imprescindible para mi estancia aquí en el Reino de España.

A mi cuñado, Carlos Micha Mba Bendom por su inestimable apoyo económico a lo largo de la carrera.

A mi pareja, Pilar Mercedes Bengobesam Micha Bengobesam, por mantenerse a mi lado hasta en los momentos más dedicados de la carrera.

A mi ínclita y conspicua mentora, María Jesús de la Fuente Aparicio, por sus benignas exigencias, por su dedicación, por su confianza en mí y por su asesoramiento a lo largo del desarrollo de este trabajo.

RESUMEN

El presente trabajo fin de grado, bajo título “Diseño de sensores software no lineales para la estimación de variables de calidad de los procesos” tiene como objetivo diseñar un sensor que sea capaz de medir la demanda química de oxígeno (DQO) en tiempo real para una estación depuradora de aguas residuales (EDAR) a partir de la información generada por los sensores hardware. El desarrollo del trabajo se lleva a cabo en el entorno de programación MATLAB. El sensor está basado en tres tipos de redes neuronales artificiales: el perceptrón multicapa (MLP) o red de propagación hacia adelante, la red de funciones de base radial (RBF), la red neuro difusa anfis y en la regresión con máquina de vectores de soporte (SVM). Los sensores se estudiarán tanto en modo estático como en modo dinámico. Se utilizará el coeficiente de correlación y el análisis de componentes principales para reducir la dimensionalidad.

Palabras clave: *red neuronal artificial, sensor software, predicción, coeficiente de correlación, análisis de componentes principales.*

CAPÍTULO I

INTRODUCCIÓN, OBJETIVOS, MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO

CAPÍTULO I. INTRODUCCIÓN, OBJETIVOS, MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO.

1.1.- MOTIVACIÓN Y JUSTIFICACIÓN DEL PROYECTO.

Conscientes de que el agua es un bien escaso e imprescindible para la vida de cualquier ser vivo, y que su cuidado tiene sus implicaciones en el bienestar humano, en la fauna y la flora, y, por tanto, del medio ambiente; desarrollar procesos de reciclaje y las correspondientes técnicas de control que contribuyan a mejorar la calidad del agua reciclada, adquiere especial importancia.

Debido a la naturaleza y complejidad propias del proceso de tratamiento de aguas residuales, éstos requieren ser automatizados dotándolos de equipos como sensores que puedan determinar en tiempo real los parámetros de calidad del agua objetivo. Sin embargo, por motivos económicos o tecnológicos, algunos de los parámetros de calidad no se pueden determinar con los sensores en línea, y dado que la fiabilidad de los sensores hardware tampoco es suficiente para garantizar la automatización y el control efectivo de todo el proceso, surge la necesidad de desarrollar mecanismos automáticos que puedan monitorizar los parámetros deseados no conocidos y que se pueden calcular fácilmente a partir de la información suministrada por los sensores en línea (Choi et. al, 2001).

En este sentido, trabajamos con la motivación de que el trabajo que estamos desarrollando podría ayudar a predecir los parámetros no conocidos, y que no se pueden medir fácilmente mediante los sensores en línea de variables que sí se pueden medir, y facilitar en la medida de lo posible las labores de control de calidad de la planta.

Así pues, este trabajo se justifica con el hecho de que plantea una solución tecnológicamente sencilla y económicamente barata, a algunos de los problemas y deficiencias que presentan los sensores en línea (sensores hardware).

1.2.- INTRODUCCIÓN Y OBJETIVOS.

1.2.1.- Introducción.

Las aguas residuales son en gran medida una consecuencia directa del uso que el ser humano hace del agua. Teniendo en cuenta que el agua es un bien escaso, y que su contaminación puede causar daños graves e irreversibles desde el punto de vista ambiental, se plantea ya no sólo la necesidad de darle un uso racional y sostenible, sino también de reciclar las aguas ya usadas, devolverlas a unos niveles de calidad aceptables, para así poder reutilizarlas para los fines deseados, o en su defecto, devolverlas al medio ambiente con la garantía de que no suponen una amenaza para la flora, la fauna y las poblaciones humanas del entorno receptor.

Para el tratamiento de las aguas residuales, se construyen estaciones depuradoras de aguas residuales (EDAR) que se automatizan, se dotan de sensores en línea con la finalidad de controlar de forma más efectiva los diferentes parámetros de calidad del agua.

Sin embargo, a pesar de lo dicho en el párrafo anterior, existen parámetros que no se pueden medir directamente sobre la planta con los sensores hardware, con lo que su medición requiere de muestras que luego deberán ser procesadas en el laboratorio después de cada cierto período de tiempo para su determinación. A esto se suma el hecho de que los tradicionales sensores hardware pueden no resultar fiables debido a los problemas de mantenimiento que a menudo presentan, como puede ser el ensuciamiento de sus electrodos que puede provocar que generen medidas erróneas (Choi et. al, 2001).

Para solventar estos inconvenientes, se plantea la necesidad de desarrollar técnicas sensoriales que sean capaces de estimar, en función de otros parámetros o variables, medidas por los sensores hardware, aquellos parámetros que con los sensores hardware no se pueden determinar. De esto se ocuparían los sensores software.

Los sensores software estiman el parámetro objetivo de calidad del agua (variable a predecir o variable predicha) utilizando la correlación entre otras variables de calidad del agua (variables predictoras). De esta manera, con los sensores software se podrá también detectar y advertir de posibles problemas en la planta, dado que, al depender su medida de otras variables (variables predictoras) del proceso, un mal funcionamiento de estas variables predictoras repercutirá en las medidas del sensor software.

Según lo anteriormente dicho, es evidente que los sensores software no están pensados para garantizar por sí solos el proceso de control de la planta, sino para trabajar simultáneamente con los sensores hardware y aprovechar la información que suministran estos de los muchos parámetros que pueden medir, para poder determinar con esta información, algunos parámetros o variables que los sensores hardware no pueden medir.

En este trabajo planteamos básicamente técnicas sensoriales basadas en redes neuronales artificiales y máquinas de vectores de soporte para predecir la variable objetivo.

1.2.2.- Objetivos

Puesto que en una estación de tratamiento de aguas residuales existen parámetros que difícilmente pueden determinarse con los sensores en línea, en este trabajo planteamos técnicas de desarrollo de sensores software que permiten predecir el valor de una

variable del agua objetivo a partir de otras variables determinadas con sensores en línea usando la correlación entre ellos.

Para lograr este objetivo, analizaremos diferentes técnicas de redes neuronales artificiales y máquinas de vectores de soporte.

Como la capacidad de predicción de una red neuronal artificial (RNA) depende del estado de sus datos de entrenamiento, para eliminar ruido o variables que no aportan información al sistema, se hará un procesamiento de los datos, logrando con esto eliminar en primera instancia, las variables que presenten muy poca dispersión (una varianza muy pequeña).

Después de este primer paso, se recurrirá a dos mecanismos estadísticos de reducción dimensional: mediante el coeficiente de correlación (r) y el análisis de componentes principales (PCA).

Para cada uno de los métodos mencionados de reducción dimensional, analizaremos tanto las redes neuronales artificiales (RNA) como las máquinas de vectores de soporte (SVM) por sus siglas en inglés, tanto en modo estático como en modo dinámico. Para cada caso, RNA y SVM estáticas, así como las RNA y SVM dinámicas, se ensamblarán de forma que se formará una red neuronal artificial híbrida (RNAH) para ver si el conjunto presenta mejor comportamiento o capacidad de predicción que cada uno de los métodos de forma aislada. Todo el proceso está pensado para buscar la mejor solución al problema.

1.3.- ORGANIZACIÓN DE LA MEMORIA.

Por cuestiones de organización, el presente trabajo lo dividimos en las siguientes partes:

- **Capítulo I: Introducción, objetivos, motivación y justificación del proyecto.**
En este punto explicamos lo que nos ha llevado a escoger y elaborar este trabajo, así como la importancia de este. Así mismo explicaremos de forma general de qué trata el proyecto y definiremos lo que se pretende conseguir con la elaboración de este proyecto.
- **Capítulo II: Técnicas y Métodos de estudio: marco teórico del proyecto.**
En este capítulo explicamos, desde el punto de vista teórico, los diferentes métodos utilizados para el diseño de los sensores software y definiremos aspectos y conceptos relacionados con el proyecto.

- **Capítulo III: Descripción de la planta depuradora de aguas residuales de estudio.**

En este capítulo se describirá brevemente la planta de procesamiento de aguas residuales en la que se recogieron los datos necesarios para el diseño de los sensores software desarrollados en este trabajo. Se definirá así mismo las variables medidas.

- **Capítulo IV: Resultados, análisis y discusión.**

En este punto se describe los métodos estadísticos de extracción de datos utilizados en el trabajo, los cuales fueron útiles en el procesamiento de datos; se describe igualmente otros aspectos relacionados con el desarrollo de la aplicación de los sensores desarrollados en este trabajo; se da a conocer los resultados del trabajo, se analizarán y se discutirán sobre los mismos.

- **Capítulo V: Conclusiones.**

Una vez estudiado el problema desde el punto de vista teórico y desarrollado la aplicación, y habiendo obtenido los resultados de la aplicación y discutido y analizados los mismos, procederemos a sacar las conclusiones pertinentes.

- **Capítulo VI: Bibliografía**

Por último, esta parte recogerá las citas bibliográficas. Se citará los textos, artículos y páginas web consultadas y usadas, en las cuales se ha recogió la información necesaria para el desarrollo de este trabajo.

CAPÍTULO II

TÉCNICAS Y MÉTODOS DE ESTUDIO: MARCO TEÓRICO DEL PROYECTO

CAPÍTULO II. TÉCNICAS Y MÉTODOS DE ESTUDIO: MARCO TEÓRICO DEL PROYECTO.

2.1.- CONCEPTO DE ESTADÍSTICA APLICADA AL ESTUDIO DE CALIDAD DE LOS PROCESOS.

Mantener unos niveles de calidad aceptables no es una cuestión menor, al contrario, es una cuestión crucial que afecta prácticamente a todo el proceso y como tal, hay que prestarle especial atención.

Debido a los elevados niveles de producción y el elevado coste que supondría descubrir al final del proceso desviaciones que se alejan de los estándares de calidad inicialmente propuestos, surge la necesidad de realizar inspecciones de calidad con más frecuencia en diferentes etapas del proceso. Esto es así porque lo que se pretende es reducir el número de unidades defectuosas o reducir el volumen de producción afectado en caso de fallo en el sistema de producción que afecte a la calidad del proceso; por tanto, la intención es reducir costes y producir con la máxima economía de medios. Para conseguir este objetivo, es necesario que el proceso esté dotado de medios de detección de desvíos rápidos, además de eficientes y fiables. Esta supervisión constante de los niveles de calidad se conoce como control estadístico del proceso (CEP).

El CEP es una de las técnicas más utilizadas para el diagnóstico a la hora de estimar los parámetros de calidad de un proceso. El CEP nos permitirá coleccionar, analizar e interpretar datos, para luego establecer las diferentes calidades, comparar desempeños y verificar posibles desvíos (Arroyo-Hernández et. al, 2016).

El CEP ayudará a determinar si el funcionamiento del proceso es el correcto, y si por tanto cumple con las especificaciones de calidad establecidas. Una aplicación correcta del CEP puede ayudar a corregir posibles fallos incluso antes de que esos aparezcan.

Entre otras cosas, el control de calidad del proceso ayuda a que, si bien es complicado mantener las variables de calidad dentro de un valor fijo y preciso, por lo menos se procura mantenerlos dentro de un rango aceptable, de forma que, si saliesen de ese rango, el producto resultante se considera defectuoso.

Como se ha dicho antes, la finalidad última de la aplicación de un control estadístico de proceso es disminuir los costes de proceso ya que la cantidad de producto rechazado será menor. Esto suele ser así porque se utilizan técnicas de muestreo estadístico para asegurarse de que las variables a medir están dentro de los rangos permitidos.

La inspección de calidad por muestreo presenta un inconveniente para procesos como el de tratamiento de aguas residuales que en este trabajo consideramos, y es que las muestras, por muy pequeñas que sean, se toman cada mucho tiempo, lo que podría provocar que, en caso de un fallo en el control de calidad del sistema ocurrido entre dos

períodos de muestreo consecutivos, en el sistema sensorial, por ejemplo, el volumen de producto que no cumple los límites de calidad sea elevado.

Para evitar mayores pérdidas, debido a la naturaleza del proceso, se prevé necesario dotar al proceso de control de calidad de medios que contribuyan a mejorar dicha acción de control de calidad. En esta tesitura surgen los sensores software.

Como se dijo anteriormente, los sensores software no sustituyen los análisis de laboratorio, los cuales están basados en las técnicas de muestreo estadístico, cuyo resultado se infiere luego en la población muestreada, sino que los refuerzan. Con los sensores software se podrá advertir con mayor antelación las posibles desviaciones y anticipar las acciones correctivas, como consecuencia de ello, el volumen de producción rechazado en caso de posibles fallos será menor, como también lo serían las pérdidas económicas asociadas.

2.2.- SENSORES SOFTWARE.

Desde el punto de vista conceptual, un sensor software se puede definir a partir de las dos palabras implicadas. Según esto, definimos:

Sensor: es un dispositivo diseñado para detectar una magnitud externa y transformarla en otro tipo de magnitud, generalmente eléctrica. Por tanto, un sensor está capacitado para detectar acciones o estímulos externos y responder en consecuencia (<https://definicion.de/sensor/>, no date).

Esta definición nos lleva a afirmar que un sensor nos permite obtener información de nuestro entorno para poder interactuar con esta información (*Güemes*, 2018).

Software: se puede definir como el equipamiento lógico intangible de un equipo informático. Este concepto incluye todas las aplicaciones informáticas necesarias para la realización de una tarea específica (<https://definicion.de/software/>).

Partiendo de estas dos definiciones y teniendo en cuenta todo lo anteriormente dicho, podemos **definir un sensor software** como el equipamiento lógico intangible que, mediante un algoritmo (software) se asocia con un sensor hardware que mide algunas variables de proceso, para estimar variables no medidas por los sensores hardware. Dicho de otra forma, un sensor software es un modelo predictivo basado en un programa informático, que, asociado con los sensores hardware, es capaz de suministrar la información que los sensores hardware no facilitan.

El hecho de no poder medir ciertas variables con los sensores en línea, siendo obligatorio en muchos casos tener que depender de recoger muestras del proceso objetivo periódicamente para procesarla fuera de línea o tener que reducir considerablemente la velocidad de proceso con la finalidad de determinar ciertos parámetros de calidad,

puede alargar los tiempos de proceso, pero también puede encarecerlo, ya que, al no poder determinar el valor de ciertas variables al mismo tiempo que los sensores hardware, algunos problemas pueden pasar inadvertidos y no ser detectados hasta obtener los resultados de laboratorio.

A esto se suma el hecho de que, las decisiones sobre el estado del proceso suelen basarse en ocasiones en estadísticas univariantes que dependen de la destreza y experiencia del operador del proceso a la hora de relacionar variables.

Para solucionar estos inconvenientes, se han desarrollado, entre otros métodos, los sensores software (Choi et. al, 2001). Los sensores software construirán características multivariantes y permitirán predecir con mayor celeridad el estado del proceso, permitiendo al operador de proceso tomar decisiones con mayor rapidez. Por tanto, podemos decir que la implementación y desarrollo de sensores software está pensado para:

- Minimizar el tiempo total de proceso
- Advertir de posibles problemas en el sistema de control; ya que su valor depende de los datos generados por los sensores en línea.
- Minimizar los costes, ya que se presentan como una opción barata en contraposición a lo que supondría la adquisición o desarrollo de equipos específicos.

Desde el punto de vista de control, los sensores software están pensados para maximizar la producción y mejorar la acción de control de la planta garantizando la calidad del proceso objetivo con la máxima economía de medios.

Si bien se señaló antes que los sensores software no están diseñados para reemplazar a los sensores hardware, sino para complementarlos, es también importante señalar que la presencia de los sensores software no sustituye los tradicionales trabajos de laboratorio, sino que los refuerzan.

2.2.1.- Clasificación de los sensores software.

El desarrollo de sensores software presenta en general dos grandes enfoques. Por un lado están los que estiman los parámetros deseados basándose en métodos deterministas; por otro está el modelo de caja negra, el cual depende sólo de los valores observados (Acha et. al, 1999),(Chéruy, 1997).

- **Sensores software basados en métodos deterministas o basados en el conocimiento.**

Son modelos basados en el conocimiento ya que requieren conocer todas leyes que rigen el sistema, y a partir de este conocimiento, establecer un modelo matemático que permita predecir las variables deseadas.

En este modelo, se establece una ecuación tal que la variable predicha se expresa en función del resto de las variables predictoras de forma lineal o no lineal. Es a partir de esta expresión donde se diseñará un algoritmo que constituirá el sensor software deseado.

Este enfoque es bastante complejo y exigente en términos de conocimientos. Se requiere de un conocimiento experto para el diseño de un algoritmo de estimación preciso. Este enfoque es menos popular que el de la caja negra.

- **Sensores software basados en datos o modelo de caja negra.**

Este enfoque de diseño de sensores software no requiere de un conocimiento previo de las leyes físicas y químicas que rigen el sistema. Simplemente utilizan, para su diseño, los datos proporcionados por los sensores en línea durante un período de tiempo para la predicción de ciertas variables no medidas por los sensores hardware. De ahí su denominación de “sensores basados en datos”.

Este modelo es más popular debido a su sencillez y facilidad de implementación, además de fiabilidad en los resultados.

A partir de estos dos enfoques, se pueden encontrar diferentes combinaciones que asocian ambos enfoques denominándose modelos híbridos o de caja gris.

En este trabajo, el enfoque utilizado para el diseño de nuestro sensor software es el basado en datos o modelo de caja negra.

2.2.2.- Metodología de desarrollo de un sensor software.

El esquema de la tabla 1.1 muestra una propuesta de secuencia de diseño a seguir en el diseño de sensores software:

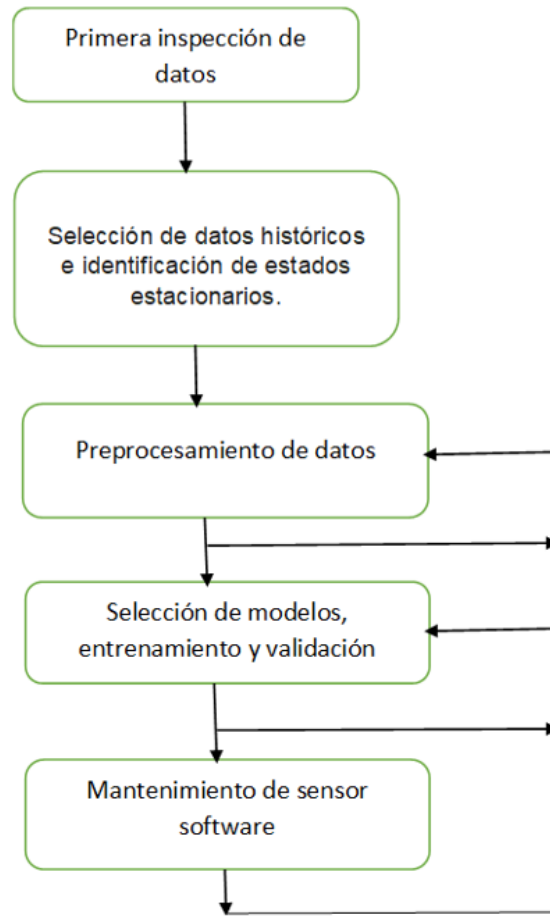


Tabla 2.1: Esquema para el desarrollo de un sensor software

Según (Kadlec et al, 2009) Por lo general, los pasos a seguir para el desarrollo de sensores software son los siguientes:

2.2.2.1.- Primera inspección de datos.

Esta etapa inicial exige hacer una inspección generalizada del conjunto de datos, de forma que se puedan eliminar variables que claramente no aportan ninguna información al sistema. Por otra parte, esta etapa le podría ayudar al desarrollador del sensor software a tomar una decisión razonable sobre si utilizar un modelo de regresión simple, un modelo de regresión PCA o una red neuronal. Como cabe la posibilidad de que la elección del modelo no sea el adecuado, es recomendable que el modelo elegido sea evaluado y comparado con otros modelos alternativos en las etapas de desarrollo posteriores.

Se verificará igualmente en esta etapa si la variable de salida presenta suficiente variación y si puede modelarse.

2.2.2.2.- Selección de datos históricos e identificación de estados estacionarios.

En esta fase se distinguen los datos de entrenamiento y validación del modelo. Una vez hecha esta separación, se procede a identificar las partes estacionarias de los datos del proceso. La identificación de los datos estacionarios sólo es necesaria en los procesos continuos, no en los procesos por lotes.

2.2.2.3.- Preprocesamiento de datos.

En esta etapa se acondiciona los datos para que sean fácilmente manejables por el sistema real. Una forma habitual de hacerlo es normalizar los datos a media cero y varianza uno, necesaria, por ejemplo, para el análisis de componentes principales (PCA).

El preprocesamiento de los datos es un proceso iterativo. En él, el desarrollador selecciona las características del modelo de forma iterativa hasta considerar que los datos están lo suficientemente preparados para ser utilizados para el entrenamiento y validación del modelo real. Es un paso que requiere mucho trabajo manual y de un conocimiento experto sobre el proceso.

2.2.2.4. Selección del modelo, entrenamiento y validación.

Esta fase es crítica en el diseño del sensor software ya que la elección del modelo es fundamental en términos de rendimiento del sensor. No existen ni teorías ni convenios sobre la elección del modelo, de forma que la elección del modelo y sus diferentes parámetros se eligen con vistas al objetivo o basándose en la experiencia del desarrollador y sus preferencias personales.

Una forma aconsejable de elegir el modelo es empezar a modelar con un modelo simple como puede ser el modelo de regresión lineal, para después ir subiendo en complejidad del modelo, siempre que se observen mejoras significativas en el rendimiento. Por ejemplo, se puede intentar con el modelo de Student, o en su caso recurrir a técnicas de estimación de error estadístico, o bien el remuestreo de métodos estadísticos, según sea el caso.

Una posible forma de mejorar el rendimiento en la generalización del sensor software es desarrollar y entrenar un conjunto de modelos de base para luego ensamblar o combinarlos.

Una vez entrenado el modelo, combinado y encontrado su punto óptimo, el modelo combinado tendrá que ser evaluado con otros datos independientes para verificar su capacidad de generalización. Si la evaluación es numérica, se puede utilizar, entre otros, el método del error cuadrático medio (MSE), que es por otro lado el más popular. Este

método mide la distancia media al cuadrado entre el valor pronosticado y el correcto. Una alternativa para evaluar el rendimiento del modelo es la observación visual, basado en el análisis de los residuos de predicción y en observar la relación entre las predicciones y los valores correctos. Uno de los inconvenientes de los métodos de observación visual es que, a falta de resultados objetivos, la decisión final depende de la valoración subjetiva del desarrollador.

2.2.2.5.- Mantenimiento del sensor software

El sensor software puede deteriorarse con el tiempo debido a las derivas y cambios en los datos. Para evitar su deterioro, habrá que mantenerlo regularmente compensándolo con un adaptador o re-desarrollándolo para que se adapte en todo momento con la nueva situación del proceso.

En la actualidad no existen mecanismos automáticos para el mantenimiento de sensores software, sin embargo, el mantenimiento del sensor software, como antes se dijo, es imprescindible debido a cambios y derivas en los datos. Eso hace que el coste del sensor software sea elevado, lo que es un inconveniente para su aplicación. Un inconveniente todavía mayor es el hecho de que, para determinar el estado del sensor, sólo se depende de la valoración subjetiva del operador, la cual está basada en la observación visual de la desviación entre el valor objetivo correcto y su predicción.

En relación con esos inconvenientes sobre el mantenimiento, se han planteado diferentes propuestas para solucionarlos, que van desde métodos adaptativos hasta el despliegue de un nuevo modelo neuronal cuando se encuentra un nuevo estado de los datos. A pesar de todas esas propuestas sobre la automatización del mantenimiento del sensor software, el operador sigue teniendo un rol fundamental por su juicio y conocimiento del proceso.

2.2.3.- Aplicaciones del sensor software

Las aplicaciones de los sensores software son muchas y las publicaciones sobre las mismas son exhaustivas. Lo que haremos nosotros en este punto es destacar algunas de las aplicaciones que desarrolla (Kadlec et. al, 2009) .

- **Predicción en línea.**

Existe un amplio campo de aplicación de los sensores software en la industria de procesos: industria química, industria de la pulpa y el acero.

La aplicación más común de este tipo de sensores es la predicción de variables o parámetros no medidos por los sensores en línea. Lo cual puede deberse a diferentes

razones. Algunas de ellas pueden ser por razones tecnológicas, es decir, no hay equipo disponible para medir el parámetro requerido, o por razones económicas, o sea, que el equipo necesario para medir la variable objetiva es demasiado caro. En estos casos, el sensor software puede ser de ayuda para proporcionar los valores de interés, suponiendo que está en condiciones para ser incorporado al sistema.

Este tipo de sensores han sido utilizados ampliamente en fermentación, polimerización y refinado.

- **Monitorización y detección de fallos de proceso.**

Otro campo de aplicación de los sensores software es la monitorización de procesos. Este monitoreo puede ser por aprendizaje no supervisado o una tarea de clasificación binaria. Para eso se entrena el sistema para que aprenda a reconocer el estado de funcionamiento normal, o en su defecto reconocer posibles fallos de proceso. Estas técnicas de monitoreo suelen estar basadas comúnmente en el análisis de componentes principales (PCA) o los mapas autoorganizativos.

- **Detección y reconstrucción de fallos del sensor**

Dado que los sensores software no son capaces de manejar datos defectuosos proporcionados por los sensores en línea, surge la necesidad de identificar el origen del fallo, o lo que es lo mismo identificar el sensor que falla para reemplazarlo antes de proceder a la reconstrucción del estado actual del proceso.

La detección de fallos se realiza generalmente con el análisis de componentes principales (PCA). Eso aporta dos ventajas: primero, los fallos del proceso se pueden identificar con relativa eficacia; segundo, mediante la manipulación del espacio del PCA, se puede también identificar el sensor o los sensores responsables del fallo.

2.3.- REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales están programadas para que emulen el comportamiento biológico del sistema neuronal de los seres vivos. Es un sistema de neuronas conectadas entre sí formando una red que es capaz de aprender, generalizar y abstraer. En la figura 2.1 presentamos la estructura funcional básica de una neurona biológica.

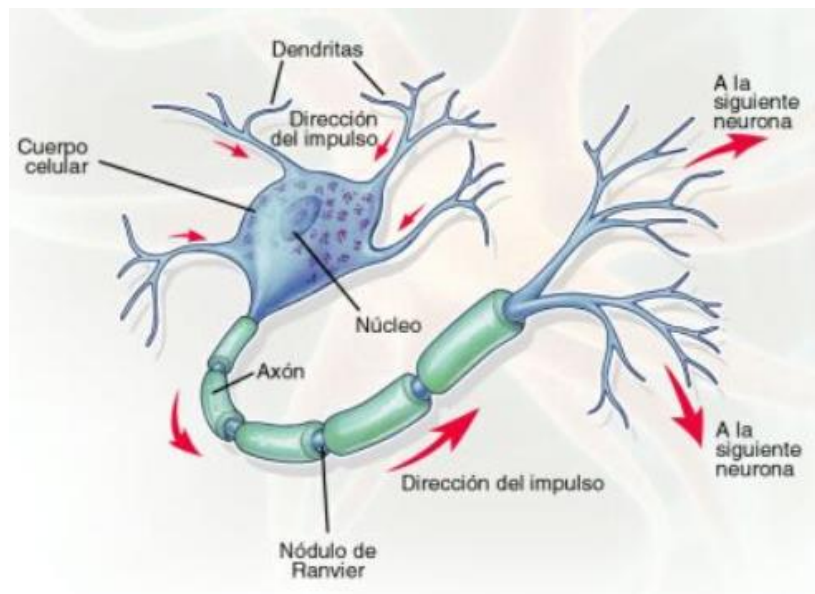


Figura 2.1: Estructura básica de una neurona biológica. Fuente:

<http://tusistemalnervioso.blogspot.com/2008/02/divisiones-del-sistema-nervioso.html>

La neurona es el elemento principal para el funcionamiento del sistema nervioso. Consta de un núcleo inmerso en un cuerpo celular del que salen las dendritas, y una rama principal, el axón. Las dendritas reciben señales, éstas son procesadas por el núcleo de la neurona que luego emite las señales de salida a través del axón por medio de sus terminales, las cuales distribuyen información hacia otras neuronas.

La comunicación entre neuronas se realiza mediante dos tipos de señales: las señales entre neuronas, transportadas a través del axón, son de tipo eléctrica, sin embargo, la generada entre los terminales axónicos, es de carácter químico.

En las neuronas biológicas, la señal llega a la neurona a través de las dendritas, las cuales están asociadas a un peso sináptico. En líneas generales, la sinapsis está formada por iones con características específicas que permiten activar o inhibir la señal eléctrica recibida; de forma que las señales de entrada pueden o no excitar a la neurona. Si el valor de las entradas a la neurona supera un valor umbral, ésta es excitada, el peso sináptico es positivo, y se envía un mensaje por medio del axón a otras neuronas; si no, éste es negativo y la neurona no se excita.

Partiendo del cuerpo de la neurona salen las dendritas hacia otras neuronas donde reciben mensajes que les llegan de otras neuronas. El punto de conexión entre neuronas es la sinapsis. Las entradas son orientadas al núcleo para ser sumadas.

Por analogía con la red neuronal biológica, las redes neuronales artificiales (RNA), el elemento principal será la neurona artificial, la cual está conectada a una o varias entradas, cada una de ellas asociadas a un peso que las pondera. En las RNA, las entradas que le llegan a la neurona modelan las dendritas; los pesos modelan la sinapsis; mientras

que la suma ponderada del producto entre cada una de las entradas y sus correspondientes pesos, modelan el cuerpo de la neurona biológica; el cuello del axón de la neurona biológica será modelado por la función de activación; por último, la salida modela el axón de la neurona biológica. En la figura 2.2 se puede ver este hecho en forma gráfica:

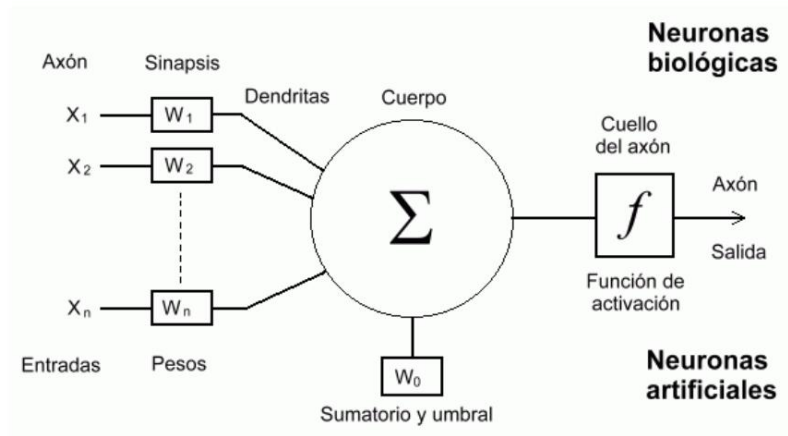


Figura 2.2: Analogía entre una neurona artificial y una neuronal biológica. Fuente: <https://inteligenciartificialmca.wordpress.com/category/sin-categoria/>

Así pues, desde el punto de vista matemático, tendremos que, ante n entradas x_i que reciben información del exterior, y sus correspondientes pesos w_i , cuyo valor será determinado por el algoritmo de aprendizaje, asignados de forma que la entrada de mayor peso tiene también mayor influencia en el procesamiento de la neurona para generar la respuesta; la entrada neta $netat$ de cada una de las neuronas será la suma ponderada del producto entre las salidas de las neuronas conectadas a ella por el peso de la conexión correspondiente. De forma que:

$$netat = \sum_{t=1}^n W_t X_t = \vec{X} \vec{W} \quad (2.1)$$

Una vez que la neurona procesa la información, el algoritmo de aprendizaje ajustará los pesos hasta que se produzca una salida. La salida de cada neurona será única y será el equivalente al valor neto de la función de activación $Y_i(t)$. La señal $Y_j(t)$ de salida es procesada por una función de activación F_{act} para generar la salida de la neurona correspondiente. Esta F_{act} puede ser lineal, una función umbral o una función no lineal que aproxime mejor las características de transferencia no lineales de las neuronas biológicas.

$$Y_j(t) = F_{act,j} (neta_j(t)) \quad (2.2)$$

La estructura de una neurona sería como se muestra en la figura 2.2.

Las principales funciones de activación de las neuronas artificiales se recogen en la tabla 2.2:

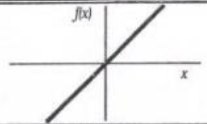
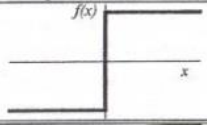
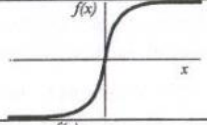
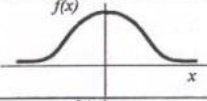
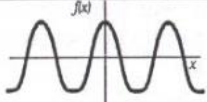
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \text{sen}(ax + \varphi)$	$[-1, +1]$	

Tabla 2.2: Funciones de activación de las neuronas artificiales. Fuente: <http://www.cs.us.es/~fsancho/?e=72>

2.3.1.- Clasificación de las redes neuronales artificiales.

Las redes neuronales artificiales se pueden clasificar en dos grandes grupos:

Atendiendo a la arquitectura o topología de la red, destacamos el número y tipo de capas de la red, las cuales pueden ser ocultas o visibles (capas de entrada y salida) y el sentido en el que fluye la información entre neuronas (hacia adelante o hacia atrás).

Por otro lado, podemos clasificar las redes neuronales artificiales en función de su aprendizaje. En función a eso destacaremos si el aprendizaje es supervisado o no, o si la red es competitiva o por refuerzo.

Basándonos en esta clasificación tendremos:

2.3.1.1.- Clasificación según la arquitectura o topología de la red

2.3.1.1.1.- Redes neuronales artificiales monocapa

Están formadas por una sola capa de neuronas en la capa de salida. No poseen capas ocultas. Esto se puede apreciar en la figura 2.3.

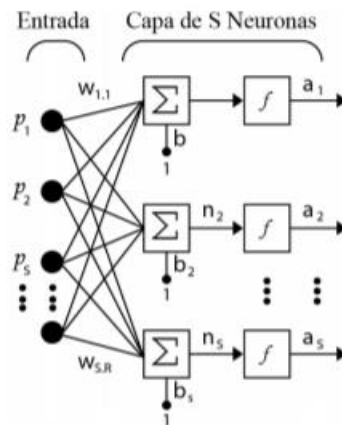


Figura 2.3: Estructura funcional de una red neuronal artificial de una capa. Fuente: <file:///C:/Users/cursi/Downloads/Dialnet-BalanceoYEstabilizacionDelPenduloInvertidoEmpleand-4269086.pdf>

En la figura 2.3, vemos la estructura de una red neuronal artificial de una capa. Si suponemos que tenemos una red neuronal de una sola capa de neuronas S , con un vector de datos de entrada P_i con R entradas y una bias b_i asociada a cada neurona; y suponiendo además que el vector de entrada a la neurona va asociado a un vector de pesos W , de forma que, cada elemento $W_{i,j}$, de W va asociado a la conexión entre la neurona y una componente de entrada R , tendremos que la salida a_i de cada neurona, una vez procesada la información que le llega de sus entradas será:

$$a_i = F(W_i * P_i + b_i) \quad (2.3)$$

Siendo F la función de activación o función de transferencia de la neurona.

Es importante señalar que R y S no tienen por qué ser iguales. Dicho de otra forma, el número de neuronas de la capa S no tiene por qué ser igual al número de entradas R que le llegan.

De este tipo de redes, las más comunes son la de Hopfield, la de BRAIN-STATE-in-a-Box y las máquinas de Boltzmann y cauchy.

Un ejemplo de aplicación de las redes neuronales artificiales monocapa es su implementación mediante hardware en circuitos eléctricos en matrices de diodos.

2.3.1.1.2.- Redes neuronales artificiales multicapa

Este tipo de redes neuronales están formadas por, además de una capa de neuronas de entrada y otra de salida, una o varias capas ocultas de neuronas. Es una generalización del caso anterior.

- La capa de entrada recibe información del exterior.
- Las capas intermedias o capas ocultas no tienen relación con el exterior. Reciben información de las capas precedentes y pueden estar conectadas de formas distintas.
- La capa de salida recoge información de las capas intermedias y la transfiere al exterior.

En la figura 2.4 presentamos la topología de una red neuronal multicapa con conexiones hacia adelante:

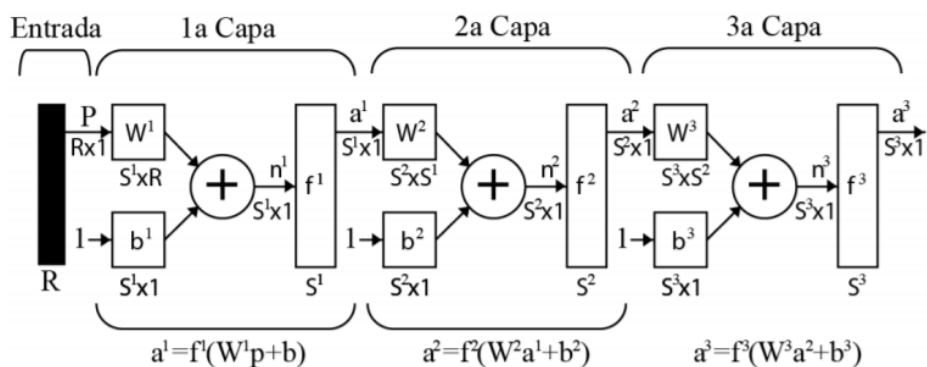


Figura 2.4: Estructura funcional de una red neuronal artificial multicapa. Fuente: <file:///C:/Users/cursi/Downloads/Dialnet-BalanceoYEstabilizacionDelPenduloInvertidoEmpleand-4269086.pdf>

Es un conjunto de capas de neuronas conectadas unas detrás de otras, cada una de ellas dotada de una función de activación propia (suponiendo que todas las neuronas de la misma capa tienen la misma función de activación), de forma que la salida será el resultado de las entradas y de cada una de las capas ocultas.

La expresión matemática resultante de la salida de una red multicapa, es también, una generalización de la expresión 2.3.

La salida para una red con tres capas ocultas sería:

$$\alpha^3 = F^3 \left((W^3 * F^2 (W^2 * F^1 (W^1 * P + b^1) + b^2) + b^3) \right) \quad (2.4)$$

En la expresión 2.4 se puede ver que, en términos matemáticos, la salida no es la suma algebraica de todas las entradas, sino que la función de activación de la primera capa está contenida en la función de activación de la segunda capa y esta última está contenido en la de la tercera capa. Eso demuestra una dependencia de las capas en sus precedentes. Con lo que, si suponemos que S_i es la capa de neuronas actual y S_{i+1} es la capa de neuronas siguiente, si no se activa S_i , S_{i+1} no puede activarse porque no le llega información alguna.

Dicho de otra forma, S_i es función de S_{i+1} . La no activación de S_i afecta a S_{i+1} haciendo imposible su activación.

Una consecuencia directa de eso es que, suponiendo que estamos en una programación de una red neuronal en la que la información sólo se propaga hacia adelante, basta con que la primera capa de neuronas esté desactivada para que todo el sistema lo esté.

2.3.1.2.- Clasificación según el sentido en el que fluye la información

2.3.1.2.1.- Redes neuronales artificiales con conexiones hacia adelante o perceptrón multicapa (MLP)

Esta red se denomina red de conexiones hacia adelante o perceptrón multicapa (MLP). A partir de ahora se utilizará indiferentemente cualquiera de estas denominaciones.

En esta red, las capas están dispuestas en el mismo orden en el que reciben y procesan la información, es de decir, desde la entrada hasta la salida. Con lo cual, no hay retorno de información desde las capas posteriores hacia sus precedentes. La información fluye siempre hacia adelante. Este tipo de redes se denominan estáticas o MLP. Cualquiera de las figuras mostradas en el criterio de clasificación anterior puede servir de ejemplo en este caso.

De estas redes podemos destacar: Adaline, Madaline, backpropagation, LQV y TMP de Kohonen.

2.3.1.2.2.- Redes neuronales artificiales recurrentes.

En este tipo de topologías, las capas pueden tener conexiones con capas anteriores, por lo que puede haber retorno de información desde la salida o desde las capas ocultas de la red a las capas anteriores. Este tipo de redes se denominan dinámicas o recurrentes.

Cuando en estas redes se da el caso de que hay neuronas capaces de retornar la información a sí mismas, se dice que la red es auto recurrente. Este hecho también se puede ver en la figura 2.5.

Este tipo de redes tienden a ser más potentes que las MLP, ya que producen una salida que no sólo se basa en la entrada actual, sino también de las entradas pasadas. Son capaces de resolver problemas de sistemas no lineales y modelos en tiempo continuo haciendo que la salida converja u oscile indefinidamente debido a su carácter temporal.

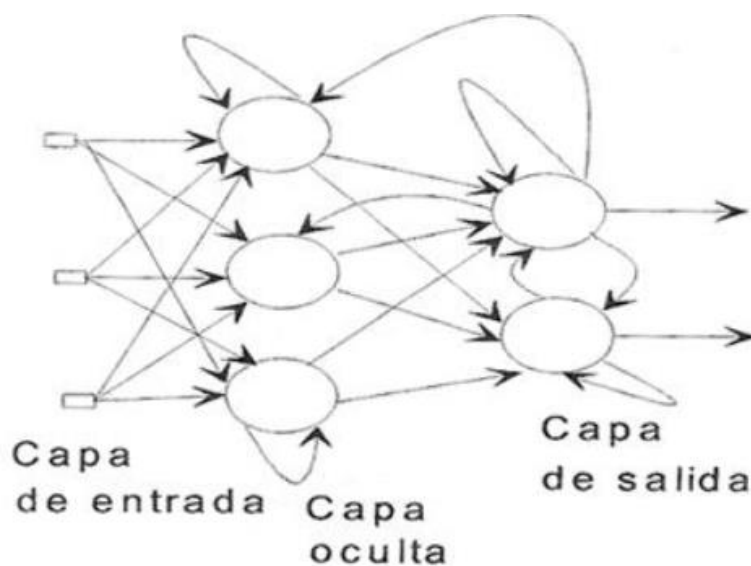


Figura 2.5: Esquema de una red neuronal recurrente o dinámica. Fuente: <http://grupo.us.es/qtocoma/pid/pid10/RedesNeuronales.htm>

El funcionamiento de este tipo de redes es tal que, parten de un vector de entradas inicial, y una vez que la red entra en ejecución, las salidas generadas se usan como entradas para el resto de los instantes.

Entre las redes dinámicas, están las ART, las BAM y las Cogniton.

Un ejemplo concreto de aplicación sería el reconocimiento de secuencias.

2.3.1.3.- Clasificación según su aprendizaje.

El proceso de aprendizaje de una red consiste en entregar a la red los patrones que debe aprender mediante cambios en los pesos de sus conexiones sinápticas haciendo uso de una regla de aprendizaje. Una regla de aprendizaje es un algoritmo consistente en expresiones matemáticas, que, mediante la aplicación de métodos como la minimización del error o la optimización de una función de energía, varía los pesos sinápticos en base a las entradas, consiguiendo con esto optimizar la respuesta de la red a las salidas estimadas. Podemos distinguir tres tipos de aprendizaje: supervisado, no supervisado e híbrido.

2.3.1.3.1.- Redes neuronales artificiales de aprendizaje supervisado.

A la red se le entrega los patrones de entrada y de la salida que deseamos estimar para esta entrada, en base a estos patrones, la red ajusta los pesos de forma iterativa hasta lograr un resultado lo más próximo posible a la salida deseada. La figura 2.6 muestra el esquema de entrenamiento supervisado de una red neuronal artificial.

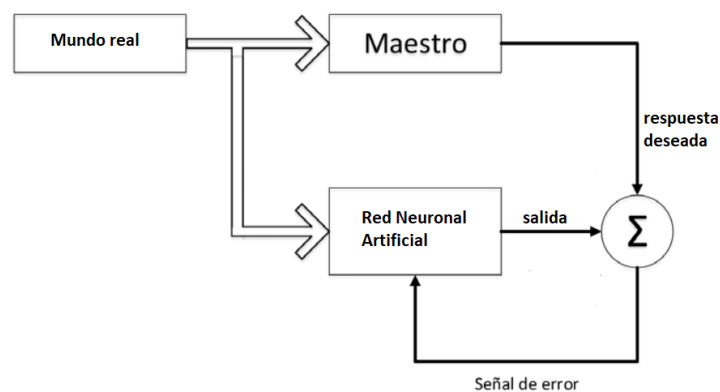


Figura 2.6: Esquema de entrenamiento supervisado.

El entrenamiento es gestionado por un supervisor externo que es el que controla el aprendizaje de la red, conoce los patrones de entrada y salida y se encargará de guiar a la red para obtener la salida requerida.

El desarrollo de la aplicación sobre el que se basa este proyecto utiliza este tipo de aprendizaje. Más adelante entraremos en detalle.

Algunas aplicaciones de este tipo de redes son: modelar funciones o relacionar patrones.

2.3.1.3.2.- Redes neuronales artificiales de aprendizaje no supervisado.

A la red no se le presenta ningún patrón de salida, sólo de entrada, y es la propia red la que, mediante el algoritmo de aprendizaje y ajuste iterativo de sus pesos, debe ser capaz de producir salidas consistentes, pudiendo encontrar características comunes de los patrones y agruparlos según su similitud, o destacar rasgos.



Figura 2.7: Esquema de un aprendizaje no supervisado. Fuente: <https://es.slideshare.net/hbanda/computacin-neuronal>

El aprendizaje no supervisado puede clasificarse a su vez en aprendizaje competitivo, aprendizaje reforzado y aprendizaje por componentes principales.

- **Aprendizaje competitivo.**

En este caso, las neuronas compiten entre sí para representar a un patrón de entrada. Si la neurona se selecciona, serán sus pesos sinápticos los que tendrán mayor incidencia en la representación del patrón de entrada. El aprendizaje del sistema se basa en debilitar las conexiones de las neuronas perdedoras y reforzar las de la neurona ganadora, de esta forma, los pesos de la neurona ganadora se aproximan más a la entrada.

- **Aprendizaje reforzado.**

Es parecido al aprendizaje supervisado, pero en este caso se limita a decir si la respuesta es correcta o incorrecta. El aprendizaje consiste en penalizar los pesos sinápticos cuando no se acierta la salida y premiarlos cuando aciertan la salida.

- **Aprendizaje por componentes principales.**

Consiste en encontrar características principales comunes a muchos patrones de entrada y con ellos conseguir representar el patrón de entradas con un número reducido de neuronas.

Este método de aprendizaje es más bien un método de reducción dimensional del patrón de entrada, que un método de aprendizaje de una red neuronal. De hecho,

nosotros en este proyecto lo hemos usado para eso, reducir la dimensionalidad de las variables de entrada y con ello conseguir que los componentes más representativos sean los que formen el patrón de entrada. Más adelante abordaremos el tema con más detalle.

2.3.1.3.3.- Aprendizaje híbrido.

El aprendizaje híbrido combina el entrenamiento supervisado y el entrenamiento no supervisado en la misma red. En general, este tipo de entrenamiento es aplicado en el proceso de entrenamiento de una red de forma que, cada capa de la red utiliza uno de los dos métodos de aprendizaje.

2.3.2.- Red perceptrón simple.

Es el primer modelo de red neuronal que se conoce. Lo crea Frank Rosenblatt en 1958. Es el modelo de red neuronal más simple. Es un modelo capaz de reconocer patrones y con capacidad para generalizar, pero presenta muchas limitaciones, una de ellas es su incapacidad para resolver patrones que no fueran separables linealmente.

Es un modelo de red formado por una capa de neuronas de entrada que recibe la información del exterior y una capa de salida formada por una sola neurona. El perceptrón simple consta de una única neurona de la capa de salida, que es la única responsable de decidir el patrón de salida. El patrón de salida puede ser +1 o -1 según sea de clase A o B respectivamente. La función de activación o función de transferencia de salida es un escalón. La topología del perceptrón simple se puede ver en la figura 2.8.

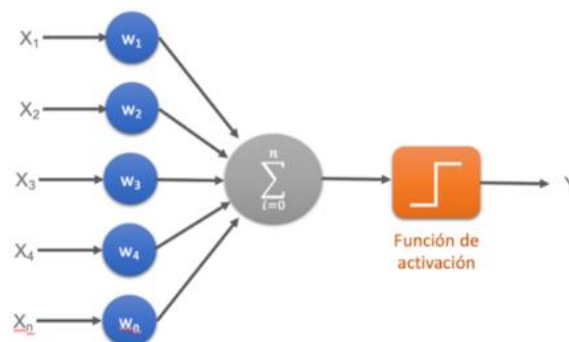


Figura 2.8: Topología del perceptrón simple. Fuente:

<http://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/>

2.3.3.- Perceptrón multicapa

Como ya se explicó antes cuando hablamos de redes neuronales multicapa, este tipo de redes se caracterizan por tener una capa de entrada que recibe información del exterior; una o varias capas intermedias o capas ocultas y una capa de salida.

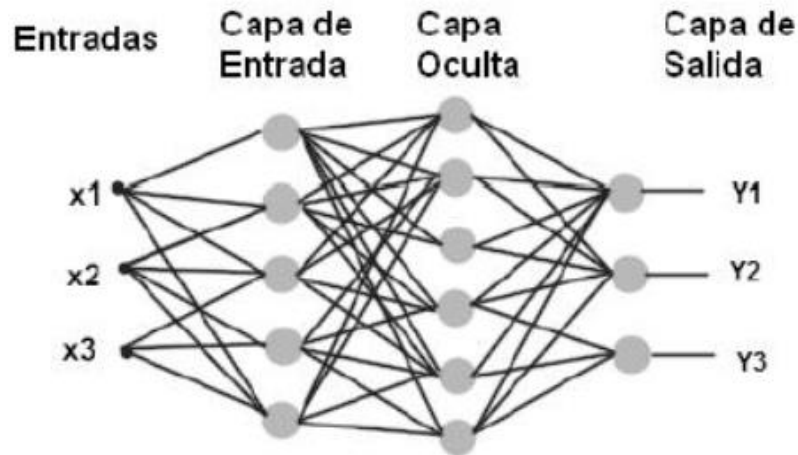


Figura 2.9: Topología del perceptrón multicapa. Fuente: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-548X2009000300006

El hecho de que el perceptrón multicapa tenga capas intermedias, lo hace más potente con respecto al perceptrón simple, pudiendo resolver problemas que este último era incapaz de resolver, como los problemas no lineales.

El perceptrón multicapa es una generalización de su anterior, el perceptrón simple. De hecho, está probado y demostrado que el perceptrón multicapa es un aproximador universal de funciones \mathbb{R}^n . El perceptrón multicapa (MLP) presenta conexiones hacia adelante, por tanto, el flujo de información es hacia adelante. La propuesta de asociar varios perceptrones simples llegó de la mano de Minsky y Papert en 1969 al considerar y demostrar que la combinación de varios perceptrones simples podía mejorar sus prestaciones y eliminar muchas de sus limitaciones. La idea tendría sus limitaciones, pero sentaría las bases para el desarrollo del perceptrón multicapa. El desarrollo de las redes neuronales alcanza un punto de inflexión en 1986 cuando David Rumelhart y G. Hinton desarrollan el algoritmo de propagación hacia atrás (backpropagation), a partir de este momento, el desarrollo y uso de redes neuronales artificiales en diferentes campos de investigación fue en aumento.

En principio, lo más normal en un perceptrón multicapa es que todas las neuronas de una capa anterior se conecten con todas las neuronas de la capa siguiente, pero eso no

siempre es así. Cabe la posibilidad de que falte alguna conexión o que no existan conexiones entre neuronas entre niveles consecutivos. En este tipo de casos, se considera que el peso asociado a las conexiones faltantes es nulo.

- **Entrenamiento y aprendizaje de la red perceptrón multicapa.**

Las RNA multicapa pueden utilizar diferentes tipos de algoritmos para el entrenamiento. Un ejemplo de algoritmo es el algoritmo de retropropagación.

En este algoritmo, los pesos sinápticos se van adaptando durante el entrenamiento de la red. Emplea un aprendizaje supervisado. El objetivo del proceso de aprendizaje consiste en minimizar la función de error cuadrático medio.

Para la minimización del error se utilizan técnicas de optimización no lineales, ya que el problema es no lineal. Estos métodos de minimización de error se caracterizan por ajustar los pesos de la red en función de la dirección negativa del gradiente de la función de error.

Los cambios en los pesos de la red se pueden hacer de dos formas:

la primera es ajustar los parámetros tantas veces como se introduzca el patrón, en este caso, el objetivo es minimizar $e(n)$ según la expresión:

$$e(n) = \frac{1}{2} \sum_{n=1}^t (s(n) - y(n))^2 \quad (2.5)$$

Donde:

$s(n)$: salida real o salida deseada de la red

$y(n)$: salida estimada por la red

t : número de neuronas.

N : número de patrones de salida.

La segunda posibilidad es, una vez metido todos los parámetros de entrenamiento, modificar los parámetros de la red. En este caso la función a minimizar es la E .

$$E = \frac{1}{N} \sum_{n=1}^N e(n) \quad (2.6)$$

2.3.4.- Redes neuronales de funciones de base radial (RFBR)

Fueron desarrolladas por Broomhead y Lowe (1988) y J. Moody y C. Darken (1989). Éstas están pensadas para mejorar las prestaciones del diseño inicial de las redes multicapa. Para ello se diseñó una red de una sola capa oculta de neuronas, de forma que su aprendizaje fuera mucho más rápido que el perceptrón multicapa. Estas redes tienen un carácter local, es decir, cuando reciben un patrón de entradas, se activan pocas

neuronas. El carácter local es la gran diferencia que tienen con el perceptrón multicapa. En común con el perceptrón multicapa, estas redes también son un aproximador universal de funciones en \mathbb{R}^n . La figura 2.10 muestra cómo se distribuyen las neuronas dentro de la capa de funciones de base radial de una red neuronal de funciones de base radial.

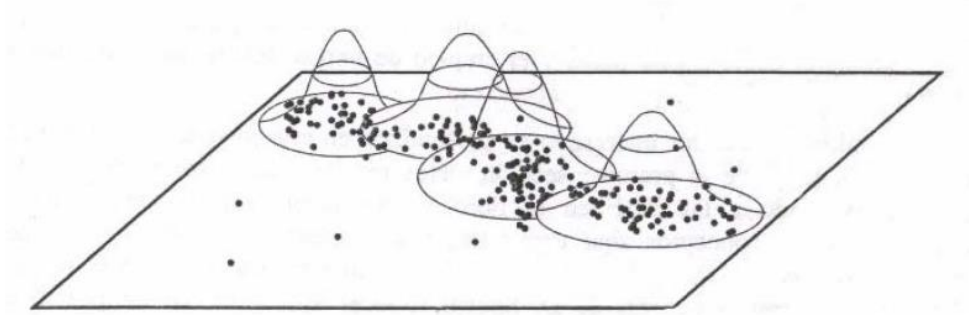


Figura 2.10: Distribución por sectores de las neuronas en la capa oculta de una red de funciones de base radial (RFBR). Fuente: https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-qlisa/redes_neuronales/curso-qlisa-redes_neuronales-html/x185.html

La arquitectura de una RFBR se puede ver en la figura 2.11.

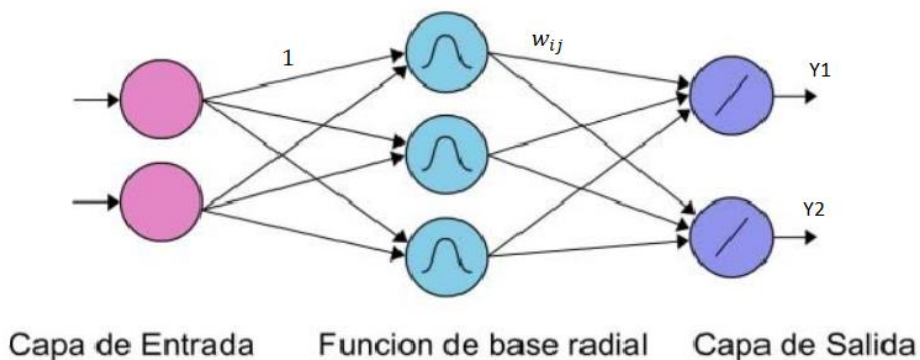


Figura 2.11: Arquitectura de una red de funciones de base radial RFBR

Como se puede ver en la figura anterior, estas redes constan de tres capas. Una capa de entrada, que se encarga de propagar la información hacia la capa oculta y el peso de sus conexiones es siempre igual a la unidad. Presenta una sola capa oculta que consta de una función de transferencia de funciones de base radial y una capa de salida dotada de una función de activación lineal. Las conexiones entre la capa oculta y la capa de salida tienen asociado un peso relativo. La capa de salida, al recibir información a través de sus

entradas procedente de la capa intermedia de funciones de base radial, realiza una combinación de lineal de las mismas. Es, en principio, una red de propagación hacia adelante.

Si consideramos C_{ij} como funciones centro y un vector de entradas X . Suponiendo que las funciones centro se mantienen fijas antes del entrenamiento de la red y que se leen durante la ejecución del entrenamiento de la red, cada neurona de la capa intermedia de funciones de base radial calculará una distancia medida entre el vector de entradas X y el vector de centros C_j . Esta distancia que llamaremos D_i se calculará de la forma:

$$D_i = \| X - C_j \|. \quad (2.7)$$

Asumiendo que d_i determina el perímetro de acción de la neurona, la distancia euclídea será;

$$r = \left(\frac{\|X - C_j\|}{d_i} \right) \quad (2.8)$$

Si suponemos, además, que ϕ_i es la salida de la i –ésima neurona de la capa intermedia, y ϕ es la función de base radial, tendremos:

$$\phi_i(X) = \phi \left(\frac{\|X - C_j\|}{d_i} \right) \quad (2.9)$$

De la ecuación 2.8 se deduce que la posibilidad de que una neurona se active aumenta cuanto más cerca se encuentra la entrada del centro de la neurona, por el contrario, las posibilidades de activación de la neurona se reducen si la entrada se encuentra lejos del centro de la neurona. Este detalle justifica lo anteriormente dicho sobre que las redes de base radical son de carácter local.

Suponiendo que j es cualquiera de las salidas, cada una de las salidas ejecutará una función lineal de la forma:

$$Y_j = \sum_{i=1}^n w_{ij} \phi_i(X) + u_j \quad (2.10)$$

Combinando la expresión 2.9 y 2.10, Y_j tendrá la forma de la expresión 2.11:

$$Y_j = \sum_{i=1}^n w_{ij} \phi \left(\frac{\|X - C_j\|}{d_i} \right) + u_j \quad (2.11)$$

La expresión 2.11 implica que, al recibir la capa de salida la información que le llega de la capa oculta, y al ser activada la capa de salida por una función lineal, la salida será la suma ponderada de las entradas de la capa de salida, las cuales son las salidas de la capa oculta.

Como consecuencia, de la expresión 2.11, la activación de cualquier neurona de la capa intermedia depende de la expresión 2.9.

Si n es el número de neuronas, el vector de centros C_j tendrá tantos elementos como elementos de entrada posea la red, porque de esta manera podrá situarlo en el espacio R^n .

2.3.4.1.- Entrenamiento y aprendizaje de la red

Las redes neuronales artificiales de funciones de base radial utilizan el aprendizaje híbrido, pero también utilizan un aprendizaje totalmente supervisado y un tercer método que combina esos dos.

- **Aprendizaje híbrido.**

Combina aprendizaje supervisado y no supervisado. La capa oculta de neuronas intermedia utiliza un método de aprendizaje no supervisado, mientras la capa de salida utiliza el aprendizaje supervisado.

En este método, se determina primero las amplitudes y centros de cada una de las neuronas empleando un sistema de aprendizaje no supervisado, para después determinar los pesos sinápticos de las conexiones, así como los valores umbrales de activación de los nodos o neuronas de salida mediante aprendizaje supervisado.

1.- Aprendizaje no supervisado de la capa oculta.

Las diferentes etapas de este procedimiento son las siguientes:

Primero: se definen los centros de las neuronas de la capa oculta. Se aplican algoritmos como k-medias o los mapas de Kohonen que optimicen la clasificación para hallar los centros de las neuronas para un número determinado de clases indicado por los patrones de entrada. Dado que en principio no se tiene conocimiento de en cuántos sectores se dividen los datos, este parámetro de diseño se determina en función de los requerimientos del problema y su definición estará basado en métodos de prueba y error (métodos heurísticos).

Segundo: se seleccionan los valores de los centroides de los L elementos a procesar en esta capa oculta.

Tercero: una vez superado el segundo paso, para cada iteración t , se asignan patrones de entrada (patrones de aprendizaje x) entre los L elementos a procesar en la capa oculta. La asignación de patrones en relación con los centros de los elementos ocultos se hace en base a la menor distancia entre el patrón de entrada y el centro de cada elemento a procesar, es decir, a cada uno de los patrones de entrada se le asigna el elemento cuyo centro diste menos.

Cuarto: se determinan los nuevos sectores según la ecuación 2.12:

$$c_j = \frac{1}{2} \sum_{i=1}^{p_j} x_i \quad (2.12)$$

Siendo p_j el número de patrones asignados a la neurona j –ésima.

Quinto: si los centros de las neuronas experimentan cambios, se vuelve al paso tercero, si no, el proceso de aprendizaje para esta etapa habría llegado a su fin.

Sexto: se determina la desviación típica σ_j para cada agrupación o sector de neuronas encontrado. Esa operación permite determinar el grado de concentración de las neuronas en un sector concreto. Dicho de otra forma, el cálculo de desviación típica permite definir el radio de acción del elemento dentro de la capa oculta.

$$\sigma_j = \sqrt{\frac{1}{p_j} \sum_{i=1}^{p_j} \|x_i - c_i\|^2} \quad (2.13)$$

2.- Aprendizaje de la capa de salida

El entrenamiento de la capa de salida es un proceso de aprendizaje supervisado. El algoritmo utilizado en este último caso es el de minimización de error. Para un problema de predicción en el que se conoce la salida estimada y_{pk} y la salida real d_{pk} , es posible determinar el error global de la red mediante la expresión 2.14:

$$E_p = \frac{1}{2} \sum_{p=1}^p \sum_{k=1}^M (d_{pk} - y_{pk})^2 \quad (2.14)$$

y_{pk} será la salida del k –ésimo elemento y se calculará mediante la expresión 2.10, la cual depende de las funciones de base radial. La expresión 2.10 se puede escribir en forma matricial. De esta forma tenemos la expresión 2.15:

$$y(x) = w\phi \quad (2.15)$$

De la expresión 2.15 se puede deducir que hay una dependencia lineal entre los pesos asociados a la capa de neuronas de salida y la salida de la red, por tanto, para la minimización del error, se aplicará el método de los mínimos cuadrados sobre la matriz de pesos.

- **Aprendizaje combinado.**

La conservación por parte del método híbrido del carácter local de la red, y, en consecuencia, el carácter propio de las RBF es una virtud de este método; sin embargo, esa cualidad hace que este método sea menos preciso que el aprendizaje totalmente supervisado. Para aprovechar las ventajas de esos dos métodos, se combinan ambos para que, por un lado, el carácter local del método de aprendizaje híbrido y, por otro lado, la precisión del método totalmente supervisado, mejoren los resultados de entrenamiento. Para esa tarea, se aplican algoritmos como k – medias, y para definir los parámetros de la red (pesos, amplitudes y centros) se emplea el método híbrido. Una vez determinado los pesos, amplitudes y centros, se emplearán como valores iniciales en el entrenamiento totalmente supervisado.

2.3.5.- Red neuro difusa ANFIS

Las redes neuronales vistas hasta ahora imitan el comportamiento del cerebro para resolver problemas. Las redes neuro difusas son algo diferentes. Están basadas en lógica borrosa, que lo que hace es modelar la forma en que el cerebro piensa (Martin del Brio et. al, 2001).

La red neuro difusa ANFIS es una combinación de las redes neuronales y la teoría difusa. Es una red neuronal artificial adaptativa Takagi-Sugeno. Las redes neuronales artificiales en este sistema se utilizan para configurar el sistema difuso. Según los patrones de entrada y salida, la propia red puede ajustar automáticamente los parámetros de diseño del sistema difuso y lograr después la función de autoaprendizaje y adaptación del sistema difuso. Este tipo de redes son aplicables a complejos sistemas de mapeo lineal y no lineal. La siguiente figura muestra la estructura de la red neuronal artificial neuro difusa (ANFIS).

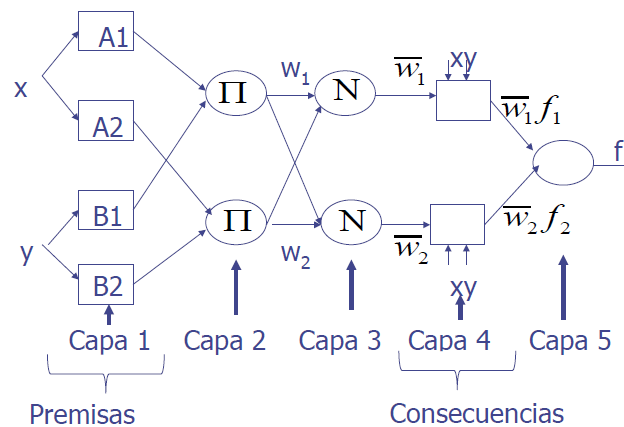


Figura 2.12: Topología de la red ANFIS. Fuente: (Correa et. al, 2013)

A diferencia de las redes neuronales antes descritas, las redes adaptativas no tienen pesos sinápticos asociados. En la figura 2.12, las líneas de conexión entre nodos sólo indican el sentido del flujo de información, y no están asociados a los pesos sinápticos como pasaría en el perceptrón simple, multicapa o en las redes de funciones de base radial. Los nodos cuadrados son nodos de parámetros ajustables o adaptativos; los nodos circulares son no adaptativos o simples. Se dice que un nodo es adaptativo cuando, a parte de la información que le llega a través de sus entradas, depende de un grupo de parámetros asociados para generar la salida. Los nodos no adaptativos sólo dependen de la señal que reciben a través de sus entradas para generar la salida.

La estructura de una red ANFIS presenta cinco niveles o capas como se puede apreciar en la figura 2.12:

Primera capa: esta capa se encarga de determinar los grados de pertenencia de la entrada x al nodo i para obtener la salida de la función de pertenencia μ_{A_i} en base a las ecuaciones:

$$\theta_i^1 = \mu_{A_i}(x) \quad (2.16)$$

$$\mu_{A_i} = \frac{1}{1 + \left(\frac{x-c_i}{a_i}\right)^{b_i}} \quad (2.17)$$

La ecuación (2.17) se corresponde con la función de pertenencia campana que es la que más se usa. Los parámetros a_i , b_i , c_i son los parámetros a calcular en las premisas de las reglas.

Cabe resaltar que, en el desarrollo y diseño de las redes neuronales, es especialmente importante manipular adecuadamente los datos de entrada, lo cual implica, entre otros métodos, una transformación o normalización de los datos de entrada. Sin embargo, en el caso de los sistemas neuro difusos, eso no es especialmente relevante, al contrario, es opcional, dado que las funciones de pertenencia permiten usar datos concretos en la primera capa donde serán difuminados (Correa et. al, 2013).

Segunda capa: esta capa se encarga de generar los pesos de disparo. Ésta presenta nodos no adaptativos, es decir, sólo dependen de las entradas. Los pesos de disparo se determinan en base a la siguiente expresión:

$$\theta_i^2 = w_i = \mu_{A_i}(x) * \mu_{B_i}(y) \quad (2.18)$$

En la figura 2.12 podemos ver que cada nodo de la capa 2 incluye un producto (π), eso quiere decir que puede tomar una T-norma para modelar el operador lógico &, el cual permitirá procesar los datos en las dos capas siguientes.

Tercera capa: al igual que la segunda capa, esta capa está formada por nodos no adaptativos, cuya misión es producir los pesos normalizados según la expresión 2.19.

$$\theta_i^3 = \bar{w}_i = \frac{w_i}{\sum w_i} \quad (2.19)$$

Cuarta capa: alberga nodos adaptativos. Esta capa se encarga de enviar a la última capa el producto entre la combinación lineal de las entradas y la capa de disparo normalizado:

$$\theta_i^4 = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i) \quad (2.20)$$

p_i, q_i, r_i : parámetros del consecuente

Para un sistema como el mostrado en la figura 2.12 con entradas x e y , se presentan estas reglas:

Si $x = A_1$ & $y = B_1 \rightarrow f_1 = P_1 * x + q_1 * y + r_1$

Quinta capa: posee un solo nodo no adaptativo. Esta capa determina la salida final del sistema según la ecuación:

$$\theta_1^5 = f(x) = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i \bar{w}_i} \quad (2.21)$$

2.3.6.- Ventajas e inconvenientes de las redes neuronales artificiales.

2.3.6.1.- Inconvenientes

- Falta de criterio definido para elegir la topología ante un problema específico.
- Ante la incapacidad de la RNA de ajustar los parámetros, es especialmente complicado identificar el problema.
- Dificultad de interpretar el funcionamiento una vez entrenada la red, es decir, una vez entrenada la red, es difícil entender cómo ha obtenido los parámetros de la red.

2.3.6.2.- Ventajas.

- Capacidad de aprendizaje: pueden aprender a partir de un conjunto de ejemplos representativos de un problema específico.
- Capacidad de generalización: capacidad de extender una respuesta aceptable a eventos o datos no usados en el entrenamiento.
- Extracción de características esenciales de los datos: ciertas RNA son capaces de extraer información relevante de los datos y rehusar la que no lo sea.
- Capacidad de asociación: las RNA tienen la capacidad de, a partir de su entrenamiento, asociar ciertas cualidades de otro conjunto de eventos o datos.
- Aprendizaje adaptativo: capacidad que tienen ciertas redes de modificar su estructura de pesos sinápticos durante el entrenamiento para adaptarse a nuevas situaciones.
- Tolerancia a fallos: soportan fallos en los datos (ruido, distorsión, etc.)

2.3.7.- Algunas aplicaciones de las redes neuronales

CAMPO DE APLICACIÓN	APLICACIÓN
Clasificación de patrones	<ul style="list-style-type: none"> • Fraudes con tarjetas de crédito. • Reconocimiento de caracteres impresos. • Reconocimiento de caracteres manuscritos. • Reconocimiento del habla • Control de calidad. • Prospecciones petrolíferas
Predicción y análisis financiero	<ul style="list-style-type: none"> • Concesión de préstamos. • Análisis de mercados. • Reservas de vuelos.
Control y optimización	<ul style="list-style-type: none"> • Fabricación de celulosa y papel. • Hornos de fundiciones. • Industria de semiconductores. • Control de procesos químicos • Refinería de petróleo
militar	<ul style="list-style-type: none"> • Guiado automático de misiles • Combate aéreo.

Tabla 2.3: Aplicaciones de las redes neuronales

2.4.- MÁQUINAS DE VECTORES DE SOPORTE PARA REGRESIÓN.

2.4.1.- Introducción.

Las máquinas de vectores de soporte fueron propuestas por Vapnik y sus colaboradores en la década de los 90. Inicialmente estaban pensadas para resolver problemas de clasificación binaria, en la actualidad se emplean para resolver problemas de regresión lineal, agrupamiento, multclasificación, visión artificial, reconocimiento de caracteres, etc. Este es un método de aprendizaje que, a diferencia de los otros métodos de aprendizaje, como las redes neuronales artificiales, que se empeñan en minimizar el error cometido por la red en el entrenamiento (error empírico), está basado en la minimización del error estructural. Lo que se pretende es elegir un hiperplano de separación que equidista de los ejemplos más próximos de cada clase y encontrar un margen máximo a cada lado del hiperplano. En el momento de determinar el hiperplano, se tendrá en cuenta sólo los ejemplos (vectores de soporte) de entrenamiento de cada una de las clases que se encuentran en la frontera de dichos márgenes.

En términos prácticos, el hiperplano tiene probado tener una gran capacidad de generalización, paliando el problema de sobre aprendizaje a los ejemplos de entrenamiento.

Algorítmicamente, la optimización del margen geométrico es un problema de optimización cuadrático con restricciones lineales que puede resolverse por métodos de programación cuadrática estándar. La propiedad de convexidad necesaria para solucionarlo garantiza una solución única, en comparación con la falta de unidad de la solución generada por una red neuronal artificial entrenada con el mismo grupo de ejemplos (Carmona-Suárez, 2014).

2.4.2.- Máquinas de soporte vectorial para regresión (SVR).

Como se señaló antes, las SVM también se pueden usar para solucionar problemas de regresión, en este caso se denominan SVR (support vector regression) o regresión vectores de soporte, en español.

2.4.2.1. Caso cuasi ajustable linealmente

Para un conjunto de entrenamiento:

$$S = \{(x_1, y_1), \dots, (x_n, y_n)\} \text{ tal que } x_i \in R^d \text{ e } y_i \in R \quad (2.22)$$

Asumiendo que todos los vectores de soporte (ejemplos) de y_i son ajustables o cuasi ajustables con una función lineal, con la regresión se pretende encontrar los parámetros $w = (w_1, \dots, w_d)$ con los que se puede definir esta función lineal:

$$f(x) = (w_1 y_1 + \dots + w_d y_d) + b = \langle w, x \rangle + b \quad (2.23)$$

Una forma de dar más flexibilidad a los ejemplos de entrenamiento para que puedan admitir algo de ruido es suavizar la condición de error entre el valor estimado y el real de forma que se pueda permitir cierta dispersión en la solución con la *función de pérdida ϵ – insensible* (L_ϵ). Los valores que queden entre $\pm\epsilon$ no son valores soporte, lo que disminuye el número de los valores soporte.

$$L_\epsilon(y, f(x)) = \begin{cases} 0 & \text{si } |y - f(x)| \leq \epsilon \\ |y - f(x)| - \epsilon & \text{en otro caso} \end{cases} \quad (2.24)$$

La expresión 2.24 es una función lineal con una zona insensible, su anchura es 2ϵ y el error de dicha anchura es cero.

Teniendo en cuenta que los ejemplos de entrenamiento siempre generan un error, se define un margen blando, ya que los datos no son perfectamente separables. Para el

margen blando se definen dos variables de holgura ξ_i^+ y ξ_i^- que cuantifican la magnitud del error mencionado.

$\xi_i^+ > 0$ si la predicción del ejemplo $f(x_i)$ es mayor que su valor real y_i para una cantidad superior a ϵ , o sea, $f(x_i) - y_i > \epsilon$ y será nulo en cualquier otro caso. Por otro lado $\xi_i^- > 0$ cuando el valor real del ejemplo es mayor que su predicción en una cantidad superior a ϵ , o sea $y_i - f(x_i) > \epsilon$ y nulo en cualquier otro caso.

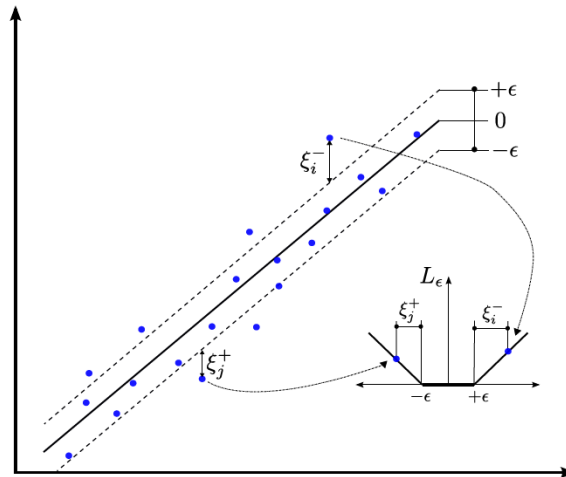


Figura 2.13: Regresión de soporte vectorial (SVR) con margen blando.

Fuente: (Carmona -Suárez, 2014)

Es imposible que la predicción de ejemplo sea a la vez mayor ($\xi_i^+ > 0$) y menor ($\xi_i^- < 0$) que su valor real. Se cumple por tanto que $\xi_i^+ * \xi_i^- = 0$.

El sumatorio de todas las variables de holgura permite determinar el coste vinculado al número de ejemplos con un error de predicción distinto de cero, de esta manera, el problema queda configurado de esta manera:

$$\min \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^n (\xi_i^+ + \xi_i^-)$$

$$(\langle w, x_i \rangle + b) - y_i - \epsilon - \xi_i^+ \leq 0 \tag{2.25}$$

$$y_i - (\langle w, x_i \rangle + b) - \epsilon - \xi_i^- \leq 0$$

$$\xi_i^+, \xi_i^- \geq 0, i = 1, \dots, n$$

A partir de aquí se nos plantea un problema dual que exige cinco pasos para llegar a su solución, empezando con la obtención de Lagrangiana hasta las ecuaciones finales. Esos pasos los omitiremos aquí por cuestiones objetivas, se pueden seguir en (Carmona-Suárez, 2014). Una vez hecho todo el desarrollo, se puede escribir la formalización del problema dual como:

$$\begin{aligned} \max \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) y_i - \epsilon \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) - \\ \frac{1}{2} \sum_{i,j=1}^n (\alpha_i^- - \alpha_i^+) (\alpha_j^- - \alpha_j^+) < x, x_i > \end{aligned} \quad (2.26)$$

$$\sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0$$

$$0 \leq \alpha_i^+, \alpha_i^- \leq C, i = 1, \dots, n$$

El regresor asociado a la función lineal deseada será:

$$f(x) = \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) < x, x_i > + b^* \quad (2.27)$$

Y finalmente, b^* se obtiene de las ecuaciones 2.28 y 2.29:

$$b^* = y_i - < w, x_i > + \epsilon \text{ si } 0 < \alpha_i^+ < C \quad (2.28)$$

$$b^* = y_i - < w, x_i > - \epsilon \text{ si } 0 < \alpha_i^- < C \quad (2.29)$$

Las expresiones 2.28 y 2.29 no pueden cumplirse a la vez, ya que $\alpha_i^+ \alpha_i^- = 0$, el valor de b^* será único, por tanto, si se da la condición $0 < \alpha_i^+ < C$ (2.28), entonces $\alpha_i^- = 0$, como consecuencia, la condición establecida en 2.29, no puede cumplirse, y al revés, $0 < \alpha_i^- < C, \alpha_i^+ = 0$ (2.29) y la restricción del primera expresión no puede cumplirse.

2.4.2.2. Caso no ajustable linealmente (Uso del kernel en las SVR)

Si los ejemplos (vectores de soporte) no pueden separarse linealmente, los ejemplos correspondientes al espacio inicial de entradas se transforman en un nuevo espacio (espacio de características) en el que sí es posible ajustar los ejemplos transformándolos mediante un regresor lineal. El tipo de transformación depende del kernel utilizado. El regresor de la función lineal en el nuevo espacio sería.

$$f(x) = \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) K(x, x_i) \quad (2.30)$$

Los coeficientes α_i^-, α_i^+ se determinan al solucionar el problema dual definido ahora de esta forma:

$$\begin{aligned} \max \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) y_i - \epsilon \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) - \\ \frac{1}{2} \sum_{i,j=1}^n (\alpha_i^- - \alpha_i^+) (\alpha_j^- - \alpha_j^+) K(x_i, x_j) \end{aligned} \quad (2.31)$$

$$\sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0$$

$$0 \leq \alpha_i^+, \alpha_i^- \leq C, i = 1, \dots, n$$

La ecuación 2.31 es análoga a la 2.26 con la diferencia de que ahora los productos escalares se sustituyen por las funciones kernel.

2.5. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA).

2.5.1.- Introducción

Uno de los dos métodos estadísticos que en este trabajo hemos utilizado para reducir el número de variables del problema es el análisis de componentes principales.

Esta técnica es aplicable a un grupo de *variables* p correlacionadas entre sí y con información superflua. Con el PCA se transforma el grupo inicial de *variables* p en un conjunto nuevo de *variables* m denominadas componentes principales, que, a diferencia del grupo original de *variables* p , no están correlacionadas entre sí y que por tanto, no incluyen información redundante. Las nuevas *variables* m son combinación lineal de las variables originales y se colocan en orden decreciente de su varianza.

Desde la óptica matemática, PCA se concibe como una transformación lineal ortogonal que transforma un conjunto de *variables* iniciales a un nuevo sistema de coordenadas tales que la mayor varianza se sitúa en la primera coordenada denominado primer componente principal (PC1), la segunda mayor varianza se sitúa en la segunda coordenada denominado PC2, la tercera mayor varianza se sitúa en la tercera coordenada, denominado PC3, etc. (Bajo Traver, 2014).

Como se ha dicho antes, PCA busca obtener a partir de un grupo de *variables* p un número menor de *variables* m de forma que $p > m$ no correlacionadas entre sí que representen sin casi pérdida de información el espacio original de variables. Podemos por tanto decir que con el PCA se pretende, ante un número elevado de *variables*, identificar la relación entre ellas con el fin de reducirlas, para obtener, en consecuencia, una reducción dimensional del problema.

2.5.1.- Determinación de los componentes principales.

Si suponemos que $x = [x_1, x_2, x_3 \dots x_p]$ es una variable estocástica p – *dimensional* de media μ . El problema reside en encontrar un nuevo grupo de variables $y_j = [y_1, y_2, y_3 \dots y_p]$ no correlacionadas, ordenadas en orden decreciente según su varianza, y que por tanto reflejen la variabilidad del conjunto inicial x_i . Cada una de las variables finales y_j será una combinación lineal de las variables x_i .

Si suponemos que la matriz X tiene n *observaciones* y m *variables*, y que por tanto es de tamaño $n \times m$:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \quad (2.32)$$

se puede hallar los vectores de carga calculando los valores singulares de (X):

$$\frac{1}{\sqrt{n-1}}X = U\Sigma V \quad (2.33)$$

siendo:

$U \in \mathbb{R}^{n \times m}$ y $V \in \mathbb{R}^{m \times m}$: matrices unitarias.

$V^T V = I, I$: matriz identidad.

$\Sigma \in \mathbb{R}^{n \times m}$: matriz de valores singulares positivos de X dispuestos de mayor a menor ($\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \sigma_m \geq 0$).

Los vectores de carga son ortonormales y dispuestos en columna en la matriz V ; mientras que la varianza del conjunto de los datos proyectados sobre la columna i de V es σ_i^2 .

Los autovalores singulares de la matriz X se pueden determinar a partir de (2.34), es decir, a través de la descomposición en valores singulares de la matriz de covarianzas de X , haciendo:

$$A = \frac{1}{n-1}X^T X = V\Lambda V^T \quad (2.34)$$

siendo:

$\Lambda = \Sigma^T \Sigma \in \mathbb{R}^{n \times m}$ = matriz de autovalores positivos dispuestos de mayor a menor

($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \lambda_m \geq 0$) con $\lambda_i = \sigma_i^2$

si se consideran los primeros valores propios más grandes y se almacenan en P , con $P \in \mathbb{R}^{m \times \alpha}$, el espacio inicial X se convierte en un espacio más reducido T :

$$T = XP \quad (2.35)$$

Por cada columna i del espacio transformado T se cumple:

1. La varianza está ordenada: $\text{var}(t_1) \geq \text{var}(t_2) \geq \dots \geq \text{var}(t_\alpha)$
2. Centrado en la media: $\text{media}(t_i) = 0; \forall i$
3. Descomposición ortogonal: $t_i^T t_j = 0; \forall i \neq j$
4. Es imposible otra expansión ortogonal de componentes que capture más variación de datos.

CAPÍTULO III

ESTACIÓN DEPURADORA DE AGUAS RESIDUALES (EDAR)

CAPÍTULO III. ESTACIÓN DEPURADORA DE AGUAS RESIDUALES (EDAR)

3.1.- CONCEPTO Y DESCRIPCIÓN GENERAL DE UN PROCESO DE DEPURACIÓN DE AGUAS RESIDUALES.

Una EDAR es una planta diseñada para reciclar las aguas usadas por el ser humano, que pueden ser de origen doméstico o industrial, para luego devolverlas al medio sin que supongan una amenaza para el medio receptor.

Aunque hay varios tipos de plantas y procesos a seguir en base al origen y/o destino del agua tratada, mostramos un ejemplo de un esquema de una EDAR en la figura 3.1 y describimos en líneas generales el proceso que se sigue en este tipo de plantas.

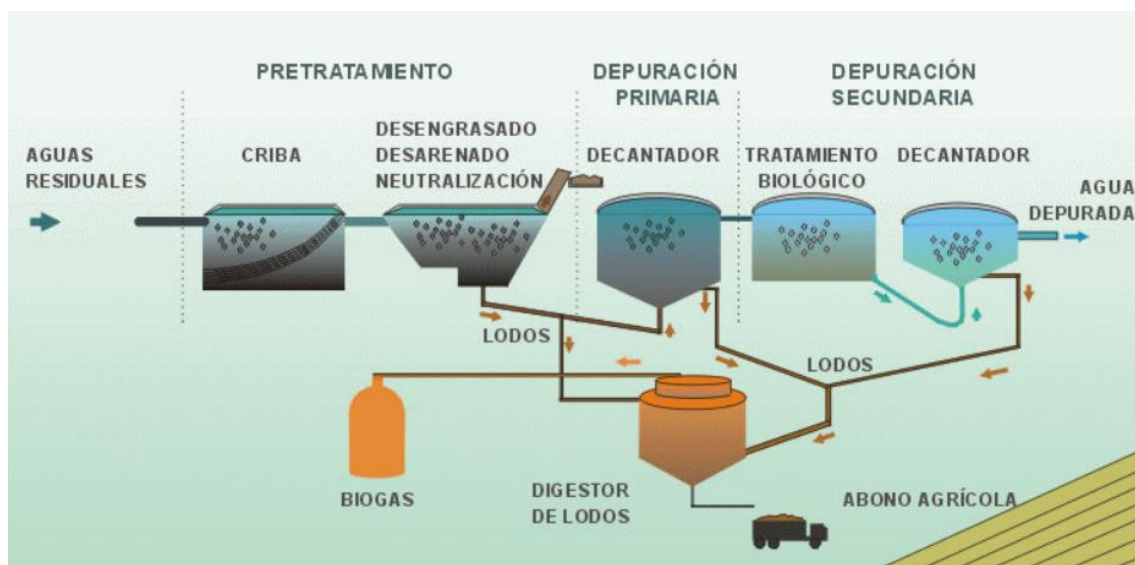


Figura 3.1: Esquema de tratamiento de aguas residuales. Fuente:

<http://www.celandigital.com/25/index.php/brujula-afondo/medio-ambiente/edar>

En el proceso de reciclado o tratamiento de aguas residuales se distinguen varias etapas, entre ellas:

Pretratamiento: consiste en una serie de tratamientos físicos destinados a separar los sólidos en suspensión en el agua. Entre los procesos del pretratamiento están:

- Desbaste: se elimina materiales como ramas, botellas, trapos, etc.
- Desarenado: se eliminan partículas no eliminadas en el desbaste como arenas, cáscaras, semillas, etc.
- Desengrasado: se eliminan grasas, aceites, espumas o materiales flotantes que fueran más ligeras que el agua y que no se han podido eliminar en tratamientos

anteriores. Mediante este proceso se evitan interferencias en procesos posteriores.

El pretratamiento tiene como objetivo evitar daños en los equipos posteriores.

Decantación primaria: mediante la sedimentación por gravedad o separación de las partículas por burbujas de aire, se separan sólidos sedimentables del material flotante. Se puede usar un material floculante que favorezca la unión de partículas para así facilitar a su vez la retirada de éstas. El tratamiento más usado es la sedimentación por gravedad, en el cual se separan los materiales por gravedad por diferencia de densidad.

Tratamientos biológicos: se conduce el agua hasta los reactores biológicos para que la materia orgánica presente en el agua sea degradada por los microorganismos, generándose un fango biológico. Este tratamiento incluye varios procesos dependiendo del origen y destino del agua tratada.

Decantación secundaria: después de la degradación orgánica por los microorganismos, el fango biológico se pasa por un decantador para separarlo del agua depurada. Una vez hecha esta separación, el fango se lleva a la línea de fangos donde será tratado. A partir de aquí, el agua será devuelto al medio natural (ríos o mares) o pasa a unos tratamientos terciarios.

Tratamientos terciarios: es un conjunto de tratamientos aplicados para definir la calidad del agua en base a su uso final.

3.2 DESCRIPCIÓN DE LA PLANTA BSM2

La planta en la que se recolectó los datos usados en este trabajo para el desarrollo de los sensores software es del modelo BSM2 (Benchmark Simulation Model 2). El esquema de esta planta se puede ver en la figura 3.2:

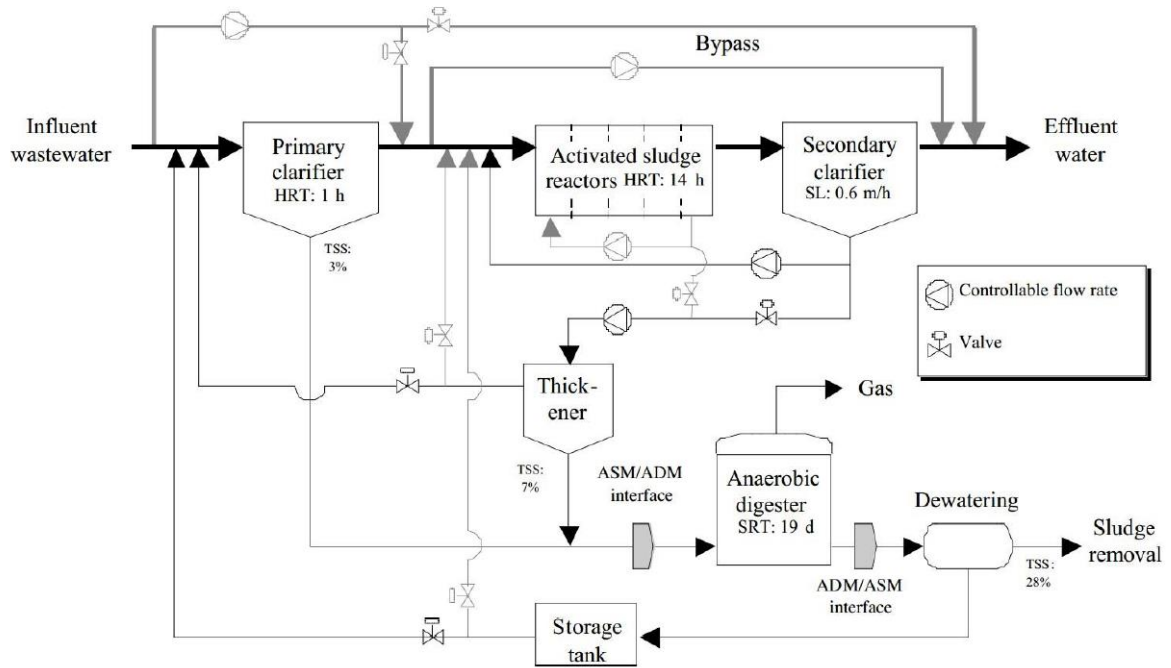


Figura 3.2: Esquema de distribución de una EDAR modelo BSM2. Fuente: (Hornik et al, 1989)

Como se puede ver en la figura 3.2, es un modelo que, como casi cualquier EDAR, incluye varios bloques que a su vez integran varias operaciones unitarias. Se distingue:

1. **Primary clarifier:** es el primer tratamiento que sufre el agua residual al entrar en el sistema. Es un decantador primario con el que se eliminan sólidos suspendidos en el agua.
2. **Activated sludge reactors:** una vez superado el punto uno, el caudal se dirige a los cinco reactores químicos dispuestos en serie y divididos en dos grupos, a través de los cuales se descompondrá la materia orgánica para reducir la demanda química de oxígeno (DQO) y convirtiendo el nitrógeno orgánico en amoníaco (NH_3). El primer grupo de dos reactores realiza la digestión anaeróbica y el otro grupo de tres reactores realiza la digestión aeróbica (Alex et. al, 2018).
3. **Secondary clarifier:** una vez superado los tratamientos biológicos, este proceso se encarga de reducir los sólidos todavía suspendidos en el agua tratada para dejarla en condiciones de poder ser retirada de la planta.
4. **Anaerobic digester:** es una de las operaciones de la línea de tratamiento de fangos. En el proceso de tratamiento de fangos generados durante el proceso, se genera gas combustible en el digester anaeróbico (Anaerobic digester) al reducir la cantidad de materia orgánica presente en fango rico en agua.

5. **Thickener y dewatering:** mediante esos dos procesos, al reducir la cantidad de materia orgánica presente en el fango, se separa el agua que contiene el fango del fango. El fango así obtenido pasa al storage tank.
6. **Storage tank:** una vez tratados los fangos en el proceso anterior, contienen unos bajos porcentajes de materia orgánica y de agua. Parte de este fango se recoge y se almacena en un tanque para ser usada de nuevo y la otra se elimina en los sitios debidamente asignados.

El sistema está dispuesto de un **by pass** de tuberías que permite desviar el flujo de agua si así lo requieren circunstancias.

3.2.1.- Descripción de las variables medidas en la planta BSM2

De la EDAR BSM2 consideramos siete variables, que son las que se han observado en diferentes puntos de la planta, generando un total de ciento cuarenta variables medidas (140 variables). Las variables consideradas las definimos brevemente a continuación:

- I. **Demanda química de oxígeno (DQO):** define la cantidad de oxígeno necesario para oxidar la materia orgánica presente en el agua residual en condiciones específicas de agente oxidante, temperatura y tiempo. Se mide en $\left[g \frac{DQO}{m^3} \right]$.
- II. **Oxígeno (O₂):** esta variable determina la cantidad de oxígeno disuelto en el agua residual. Se expresa en $\left[g \frac{O_2}{m^3} \right]$.
- III. **Alcalinidad (salk):** este parámetro permite determinar la cantidad de hidrógeno o iones ácidos (iones bicarbonatos, hidróxidos, carbonatos, etc...) presentes en el agua residual. Dicho de otra forma, la alcalinidad determina la capacidad del agua para neutralizar los ácidos presentes en el agua. Se expresa en $[ppm \text{ de } C_aCO_3]$ o $\left[\frac{meq}{L} \right]$.
- IV. **Nitrógeno (N₂):** esta variable indica la cantidad de nitrógeno presente en el agua residual. Se presenta como nitratos, nitrógeno orgánico, amonios. Sus unidades son $\left[g \frac{N}{m^3} \right]$.
- V. **Sólidos suspendidos (SS):** determina la cantidad de sólidos contenidos en el agua residual. Se mide en $\left[g \frac{SS}{m^3} \right]$.
- VI. **Caudal (Q):** cantidad de agua que atraviesa un punto por unidad de tiempo. Se mide en $\left[\frac{m^3}{s} \right]$.
- VII. **Temperatura:** indica la temperatura del agua residual en un punto determinado del proceso. Se mide en $[^{\circ}C]$.

CAPÍTULO IV

RESULTADOS, ANÁLISIS Y DISCUSIÓN

CAPÍTULO IV. RESULTADOS, ANÁLISIS Y DISCUSIÓN.

Cabe advertir que, aunque este capítulo se titule “resultados, análisis y discusión”, no aparecerán como tres partes diferentes a lo largo del capítulo, sino que iremos analizando y discutiendo cada caso según vayamos reflejando los resultados.

En este capítulo describiremos el proceso que se ha seguido para desarrollar el sensor software, se presentarán los resultados obtenidos en las simulaciones, se analizarán, se compararán y se discutirá sobre los mismos.

El presente trabajo ha sido desarrollado en el entorno de programación MATLAB mediante la implementación de algoritmos pensados para conseguir nuestro objetivo final, el cual es diseñar un sensor software con capacidad para predecir la demanda química de oxígeno (DQO) para una estación depuradora de aguas residuales (EDAR), a partir de los datos generados por los sensores en línea contenidos en variables explicativas.

Como ya se señaló en el capítulo II cuando se explicó el sensor software, aunque en principio la misión del sensor software sea la de predecir la variable DQO, su dependencia de otras variables le da la capacidad de advertir posibles problemas en el sistema de control de calidad de la planta.

Como también se anticipó en el capítulo II, el sensor software que se pretende diseñar, es un sensor basado en datos, porque sólo utiliza los datos generados por los sensores en línea para predecir una variable concreta; no es necesario, por tanto, un conocimiento de las leyes físicas y químicas del sistema en cuestión.

Para el diseño del sensor software se plantearon diferentes enfoques. Se usó la varianza para el preprocesamiento de los datos. El preprocesamiento de datos fue común a los dos criterios de reducción dimensional. Los dos criterios de reducción dimensional están basados en el análisis de componentes principales (PCA) y en el coeficiente de correlación. Para ambos criterios de reducción dimensional se estudiaron el comportamiento estático y dinámico de las redes perceptrón multicapa (MLP), las redes de funciones de base radial (RBR), las redes anfis, basadas en teorías difusas y las máquinas de vectores de soporte (SVM), en nuestro caso, regresión de vectores de soporte (SVR). Una vez simuladas las redes, se procedió a su ensamble, usando para ello la mejor red de cada tipo. La intención es estudiar el comportamiento del conjunto frente a cada una de las redes. El ensamble se hizo por cada método de reducción dimensional y atendiendo a si son redes estáticas o dinámicas. Después compararíamos los ensambles con el comportamiento individual de cada una de las redes; el comportamiento de las redes estáticas con el de las redes dinámicas o el comportamiento entre ensambles dinámicos y estáticos.

4.1.- RECOPIACIÓN DE DATOS EN LA EDAR BSM2.

Como se dijo y se describió en el capítulo III, la EDAR considerada, ya que de ella partieron los datos con los que trabajamos en este proyecto, es de tipo de BSM2. Los datos fueron tomados durante dos años. Se tomó un total de 58465 datos, a razón de un dato cada quince minutos, midiendo un total de 140 variables diferentes, aunque luego se añadiría una variable a mayores para que sean un total de 141.

El comportamiento de la planta con el que se recopiló los datos era normal, estable desde el punto de vista de control; sin presencia de turbulencias en el proceso ni situaciones indeseables que puedan suponer una desviación en las mediciones proporcionadas por los sensores en línea, que, por otro lado, es la situación deseada e indicada para recopilar datos útiles para una aplicación como la que en este proyecto desarrollamos.

4.2.- PROCESAMIENTO DE DATOS.

Lo primero que hacemos es importar los datos recogidos de la planta contenidos en una hoja de Excel a MATLAB, a partir de ahí construimos la matriz con la que trabajaremos.

Como antes se ha dicho, el total de los datos contenidos en la planta es de 58465 datos, de esos cogimos un dato de cada 16. Dicho de otra manera, hemos dividido la muestra de datos original entre 16, es decir, cambiamos el periodo de muestreo de la planta, cogiendo un dato cada 4 horas, que es mucho más similar al que se usa en una planta real, quedándonos, por tanto, con 3655 datos. Esta muestra es lo suficientemente grande para ser utilizada en el diseño del sensor software de forma que éste puede aprender bien, validar y posteriormente generalizar.

Como en este caso tenemos un patrón de salida contenida en una variable entre las 141 que contiene la muestra extraída de la original, lo que hacemos ahora es separar esta variable del resto de las variables. La variable así separada será el patrón de salida, que será utilizada como meta, a la cual tendrá que ser capaz de alcanzar ajustándose lo más posible el sensor software. Esto hace que las redes neuronales implementadas sobre las cuales se basa nuestro sensor utilicen un aprendizaje supervisado.

Una vez construida la matriz de datos a partir de la matriz de datos original, lo que hicimos es un preprocesamiento de datos con el fin de hacer un primer filtro para eliminar ruido. El preprocesamiento de los datos se hizo utilizando la varianza. Para discriminar las variables bajo este criterio, primero se calcula la varianza de cada variable, luego se elimina de la muestra las variables con varianza inferior a 0,001. Estas variables se eliminan de la muestra por considerar que sólo aportan ruido, ya que no

aportan ninguna información adicional al sistema; su varianza es próxima a cero y prácticamente son constantes.

Una vez superado este punto, lo que hacemos es distinguir los datos de entrenamiento de los de validación. En nuestro caso, utilizamos un setenta por ciento (70%) de datos para el entrenamiento y un treinta por cierto (30%) de datos para la validación.

Todo lo anteriormente comentado en este apartado es común a los diferentes sensores implementados, tanto para las redes estáticas como para las dinámicas.

A partir de aquí, se procede a reducir la dimensión de la muestra utilizando, o el análisis de componentes principales (PCA) o el coeficiente de correlación. Según el criterio que se utilice para reducir la dimensión, se analizarán las redes estáticas y dinámicas y el ensamble de éstas.

Cabe destacar que, como antes se señaló en el capítulo II, el tratamiento (preprocesamiento y reducción dimensional) de datos es de suma importancia, por lo menos en nuestro caso, ya que la capacidad de aprendizaje de las redes neuronales depende del estado de sus datos entrada. Si estos datos contienen ruido, pueden surgir problemas de sobre aprendizaje, que a su vez tienen sus implicaciones en la capacidad de validación y generalización de la red neuronal artificial.

Viendo el problema planteado, puede preguntarse por qué para desarrollar un sensor software basado en redes neuronales artificiales es necesario hacer todo lo que hemos hecho. La respuesta a la pregunta se puede hallar a lo largo de este mismo documento concretamente en el capítulo II, y es que, uno de los grandes inconvenientes de las redes neuronales es que no existen criterios ni convenios que permitan al desarrollador de la aplicación optar por un tipo de red concreto para una situación determinada; todo se hace según las preferencias y la experiencia del desarrollador de la aplicación. Por esta razón, nosotros optamos por probar con varios métodos para ver cuál de ellos aportaría mejor solución a nuestro problema.

A continuación, exponemos los resultados de las redes estáticas.

4.3.- REDES NEURONALES ESTÁTICAS.

En este tipo de redes neuronales artificiales, la información se propaga en una sola dirección, de las entradas a las salidas. Son redes que, ante unas entradas, proporcionan una salida, por lo que en ejecución convergen y garantizan la estabilidad (Martin del Brio et. al, 2001) .

Es importante señalar que cada modelo de red, estática, dinámica o ensamblada constituye un sensor software independientemente del criterio de reducción dimensional que se aplique.

Para este tipo de redes aplicaremos dos criterios de reducción dimensional; el basado en el coeficiente de correlación y el análisis de componentes principales. Para ambos criterios analizaremos las redes perceptrón multicapa (MLP) o redes de propagación hacia adelante, las redes de base radial, las redes de interferencia neuro-difusas o redes anfis y los regresores de vectores de soporte, así como la correspondiente combinación de todas estas redes. La combinación la asignaremos unos factores, que, en un primer caso serán iguales para todos, y en otro caso les asignaremos diferentes factores, quedándonos con la que mejor error de prueba presente. En todos los casos presentaremos los resultados de entrenamiento y los de validación (test).

Es importante señalar que, sólo en unos casos presentaremos las gráficas de los residuos, como tampoco presentaremos todas las gráficas de entrenamiento en todos los casos, sin embargo, los de validación siempre los mostraremos.

Es igualmente importante indicar que sólo hemos normalizado las redes MLP y la red de funciones de base radial entre 1 y -1, las otras dos redes, la red anfis y la SVM no han sido normalizadas, por eso, para evitar comparaciones viciadas, vamos a desnormalizar todas para poderlas comparar y ensamblar.

4.3.1.- Reducción dimensional con el coeficiente de correlación (R).

Lo que hicimos es determinar el coeficiente de correlación de todas las variables de la muestra para estudiar la relación entre las variables predictoras y éstas con la variable respuesta o predicha.

A partir de esto usamos un valor de referencia del coeficiente de correlación para discriminar una parte de las observaciones de la muestra y estudiar la red correspondiente. Tomamos varios valores de referencia generándose varios casos.

Queremos dejar claro que a partir de ahora R1 o simplemente R hacen referencia a lo mismo, el coeficiente de correlación. Se utilizarán indiferentemente a partir de ahora.

4.3.1.1.- Redes perceptrón multicapa (MLP) reduciendo variables con el coeficiente de correlación (R).

Usamos una sola capa oculta con varias neuronas en todos los casos. Optamos por usar una sola capa de neuronas porque usar varias no garantiza la mejora de los resultados obtenidos con una sola capa, al contrario, si no son casi los mismos, pueden ser peores. Este tipo de redes utilizan un aprendizaje totalmente supervisado. El número de neuronas de la capa oculta es el único parámetro a tener en cuenta, es el parámetro crítico. El resto de los parámetros se usan por defecto. Sobre la tabla señalamos el número de variables que quedan al variar el coeficiente de correlación (R1).

Hemos entrenado diferentes tipos de redes en todos los casos, y entre las muchas redes entrenadas, nos hemos quedado con tres en cada caso, diferenciadas por el número de neuronas en la capa oculta, y de todas ellas, basándonos en el valor del residuo de validación (test) elegimos la mejor. Ésta es la que se usaría para combinar con el resto de las redes. Cuanto más pequeño es el residuo de validación, mejor es la red. Según eso tenemos la tabla 4.1.

REDES PERCEPTRÓN MULTICAPA (MLP) ESTÁTICAS - REDES ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN R1.		
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 < 0.2$		
NÚMERO DE VARIABLES = 82		
N.º neuronas	Residuo entren.	Residuo test
5	0.0088	0.0126
10	0.0065	0.0082
15	0.0091	0.0116
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 > 0.3$		
NÚMERO DE VARIABLES = 25		
N.º neuronas	Residuo entren.	Residuo test
5	0.0335	0.0393
10	0.0045	0.0126
15	0.0090	0.0099
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 > 0.5$		
NÚMERO DE VARIABLES = 7		
N.º neuronas	Residuo entren.	Residuo test
5	0.0019	0.0255
10	0.0046	0.0158
15	2.7772e-04	0.0157

Tabla 4.1: Redes MLP estáticas entrenadas reduciendo variables con R.

Según la tabla 4.1, para $R1 < 0.2$, nos quedamos con las variables que tienen un coeficiente de correlación por debajo de 0.2 que son un total de 82 variables. En el segundo caso nos quedamos con las variables que tienen un coeficiente de correlación mayor de 0.3, reduciéndose el conjunto de variables en 25 variables, en el último caso sólo estudiamos la red con variables que tienen un coeficiente de correlación superior a 0.5, reduciéndose el número de variables a 7. Esta tarea será común a todas las redes en las que se apliquen la reducción de las variables con R, lo que puede variar es el valor de corte de R1, el cual hará que tengamos más o menos variables.

Cabe destacar que los datos de entrenamiento para esta red están normalizados entre menos uno y uno.

Remarcamos en amarillo la mejor red en cada caso. Se puede ver que la mejor red de todas, a juzgar por su residuo de validación, es la red de 10 neuronas utilizando el coeficiente de correlación ($R1 < 0.2$). Los resultados gráficos de entrenamiento y validación y los residuos generados en ambos casos generados por esta red se pueden ver en las figuras 4.1a, 4.1b, 4.1c, 4.2a, 4.2b y 4.2c.

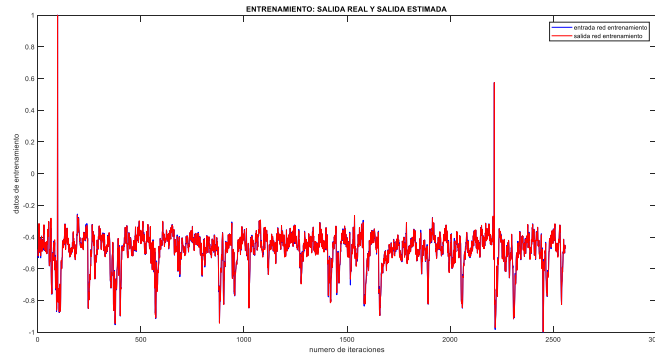


Figura 4.1a: Resultado de entrenamiento de la red estática MLP para predicción de DQO reduciendo variables con R

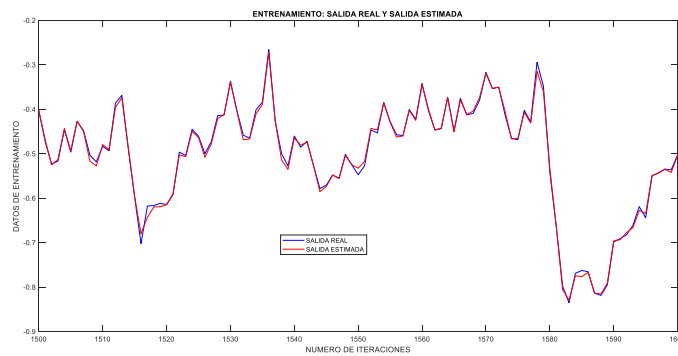


Figura 4.1b: Resultado de entrenamiento de la red estática MLP para predicción de DQO reduciendo variables con R

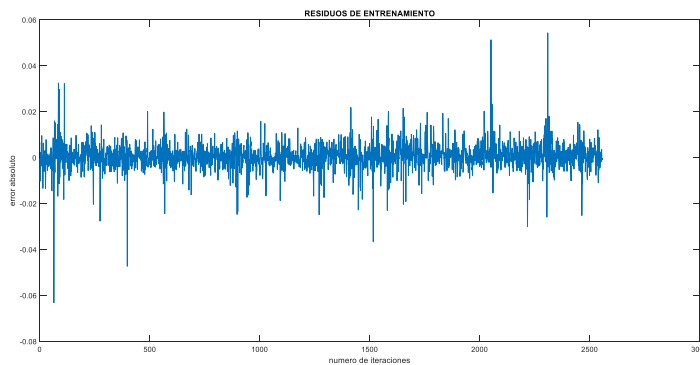


Figura 4.1c: Resultado del residuo de entrenamiento de la red estática MLP reduciendo variables con R

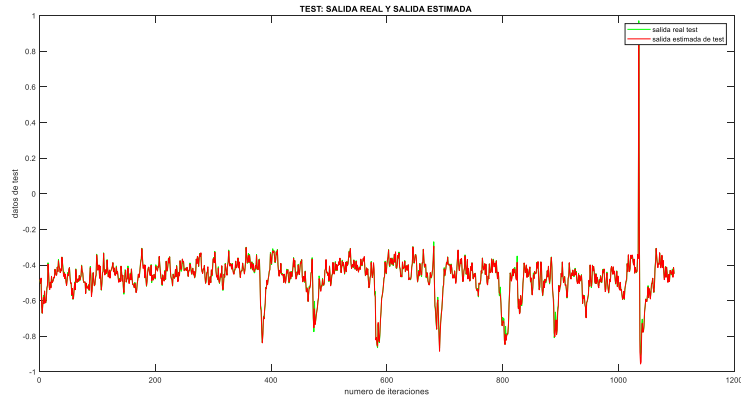


Figura 4.2a: Resultado de validación (test) de la red estática MLP para predicción de DQO reduciendo variables con R

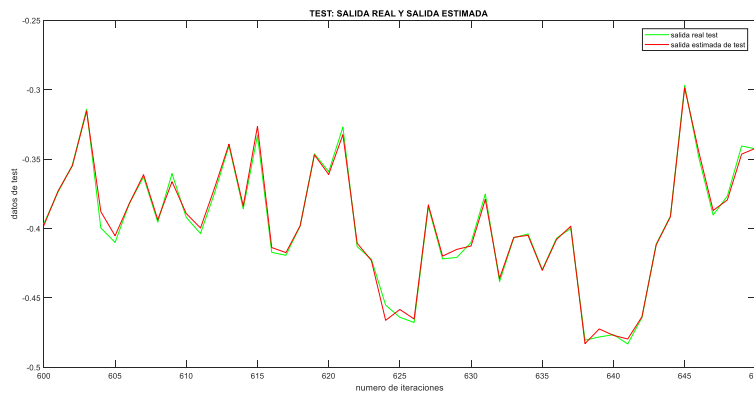


Figura 4.2b: Resultado de validación (test) de la red estática MLP para predicción de DQO reduciendo variables con R

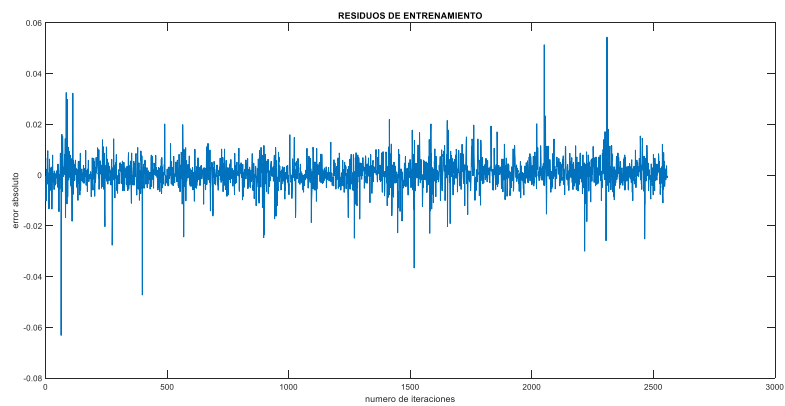


Figura 4.2c: Resultado del residuo de validación de la red estática MLP reduciendo variables con R

Se puede ver que con la red estática perceptrón multicapa (MLP) se obtiene un resultado aceptable, pues la salida estimada se ajusta relativamente bien con la salida real.

La distribución gráfica de los residuos que deja esta red en el entrenamiento y validación la presentamos en las figuras 4.1c y 4.2c. En estas dos tablas, lo que se ve es el valor medio de los residuos de entrenamiento y test. Es fácil deducir que este valor está influenciado por el mayor pico; éste puede ser debido a uno varios valores atípicos en el conjunto de datos o en los residuos que genera la red.

4.3.1.2.- Redes de funciones de base radial (RBF) reduciendo variables con el coeficiente de correlación (R).

Por lo expuesto anteriormente, para este tipo de redes también usamos en todos los casos una capa de neuronas en la capa oculta. El aprendizaje de este tipo de redes es híbrido como ya se explicó en el capítulo II. En este caso, el parámetro crítico es el objetivo. Con dicho parámetro, determinamos el objetivo de error cuadrático medio (MSE) máximo admisible por la red después de todas las iteraciones para regular sus pesos. El MSE se puede ver en la última columna de la tabla 4.2. El MSE se lo damos a la red para indicarle que éste es el error máximo con el que queremos que calcule la salida. La red, mediante la aplicación del método de aprendizaje y los algoritmos internos necesarios, calculará la salida, si no ajustándose al MSE que le fuera asignado previamente, se aproximará todo lo posible a él.

Se puede pensar que cuanto más pequeño es dicho objetivo, mejores resultados ofrece la red, pero eso tiene un límite. Por ejemplo, poner el objetivo a cero, eso a lo que lleva es alargar innecesariamente el tiempo de simulación, puesto que lo que se le pide a red es que reduzca el error cuadrático medio a cero, y que, por tanto, la salida estimada y la deseada se ajusten perfectamente, realidad que, por lo que hemos visto en las simulaciones de esta red, es bastante complicada. Al no haber un criterio ni método para elegir el parámetro objetivo, hay que basarse en métodos heurísticos para hallar el mejor objetivo posible que mejore el rendimiento y la efectividad de la red.

Como con las redes MLP, entrenamos diferentes redes variando el coeficiente de correlación (R_1), haciendo que éste sea mayor o menor a un valor, así, con respecto a este valor, discriminamos variables y conseguimos una reducción dimensional. Eso nos permite estudiar el comportamiento de la red con las variables que nos quedan. A continuación, en la tabla 4.2 presentamos las redes entrenadas.

REDES DE FUNCIÓN DE BASE RADIAL (RBF) ESTÁTICAS - REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN R1.			
Variando solo el goal y coeficiente de correlación R1<0.2 NÚMERO DE VARIABLES = 82			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7.5e-4	0.0274	0.1439	0.00085546
5e-3	0.0706	0.1127	0.00552109
3e-3	0.0546	0.1264	0.00325437
Variando solo el goal y coeficiente de correlación R1>0.3 NÚMERO DE VARIABLES = 25			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7.5e-4	0.0262	0.0469	0.00210496
2e-3	0.0447	0.0507	0.00210496
1e-4	0.0099	0.0455	0.000107842
Variando solo el goal y coeficiente de correlación R1>0.5 NÚMERO DE VARIABLES = 7			
goal	Residuo entren.	Residuo test	Mse (última iteración)
3e-6	0.0016	0.0403	3.117e-06
2e-6	0.0013	0.0409	3.117e-06
2.5e-6	0.0015	0.0403	3.117e-06

Tabla 4.2: Redes de funciones de base radial (RBF) estáticas entrenadas reduciendo variables con R.

Como en el caso de la red estática MLP, tomados los mismos valores del coeficiente de correlación para reducir el número de variables obteniéndose el mismo número de variables que antes en cada uno de los casos.

En amarillo señalamos las mejores redes de cada caso. La mejor red, debido a su reducido residuo de validación, en comparación con el resto de las redes, es la última red señalada en amarillo, la cual tiene un objetivo "goal" = 2.5e-6.

Es importante señalar que en muchos casos nos encontraremos con redes con mejor residuo de entrenamiento que de validación o test; eso quiere decir que la red se ha entrenado bien, pero que la validación no ha sido tan buena como el entrenamiento, en otros casos es al revés, aunque menos habitual.

Las gráficas 4.3a, 4.3b y las 4.4a, 4.4b recogen los resultados de entrenamiento y validación respectivamente de la mejor red de funciones de base radial (RBF).

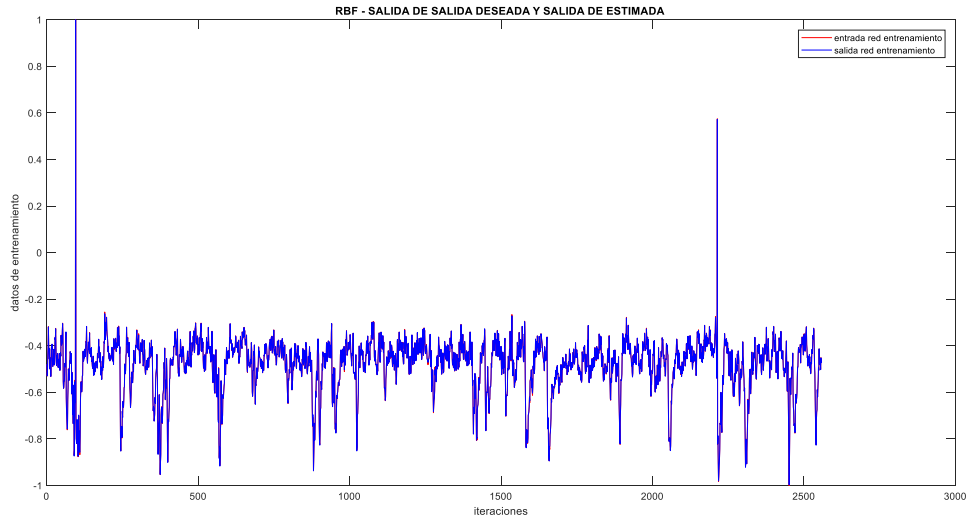


Figura 4.3a: Resultado del entrenamiento de la red estática RBF para predicción de DQO utilizando R para reducir variables

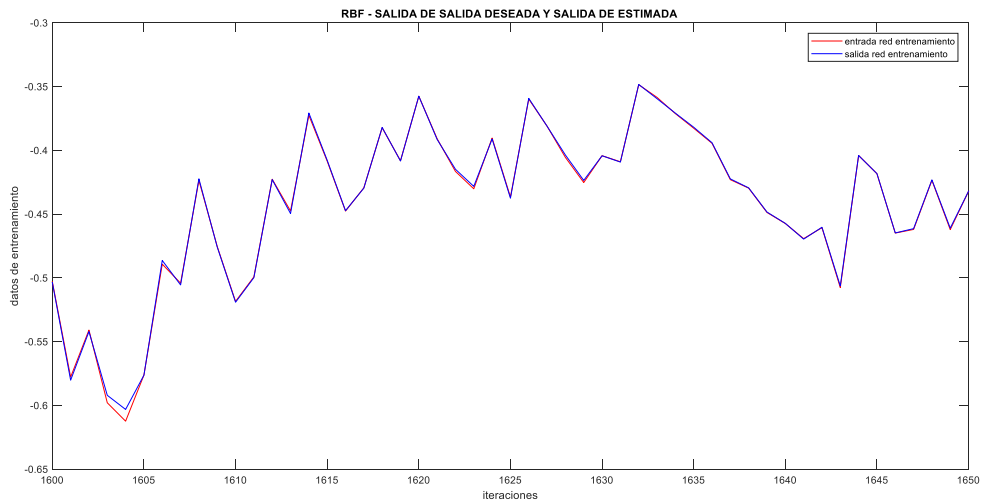


Figura 4.3b: Resultado del entrenamiento de la red estática RBF para predicción de DQO utilizando R para reducir variables

En la figura 4.3a y 4.3b, se puede ver que el entrenamiento es bastante bueno, ya que la salida estimada se ajusta bastante bien a la salida deseada.

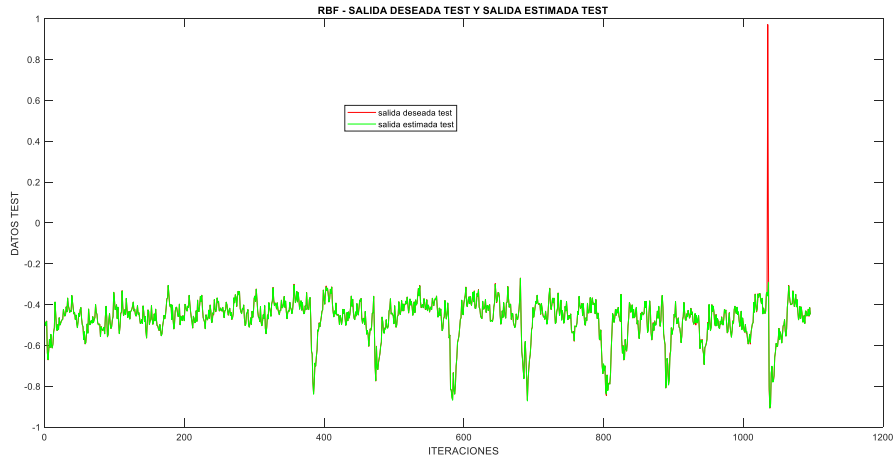


Figura 4.4a: Resultado de validación de la red estática RBF para predicción de DQO utilizando R para reducir variables

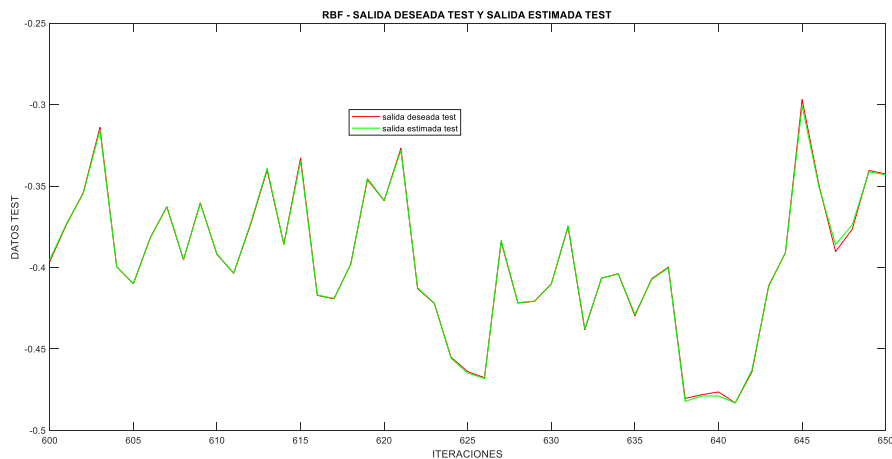


Figura 4.4b: Resultado de validación de la red estática RBF para predicción de DQO utilizando R para reducir variables

Visualizando las gráficas 4.4a y 4.4b, podemos ver el comportamiento de la salida estimada por la red con respecto a la salida deseada. Es fácil ver que ambas salidas están bastante bien sintonizadas, por lo que, consideramos que la red ha aprendido bien.

Una comparación entre la red de funciones de base radial y la red MLP nos permite afirmar que, aunque la red RBF aparenta estar mejor entrenada, a juzgar por su inferior error de entrenamiento, comparado con la red MLP, no aprende mejor que ésta, ya que la MLP obtiene mejor residuo de test.

4.3.1.3.- Redes anfis reduciendo variables con el coeficiente de correlación (R).

Las redes anfis están basados en conjuntos difusos como ya comentamos en el capítulo II cuando definimos este tipo de redes.

En ellas aplicamos también el método de reducción dimensional basado en el coeficiente de correlación utilizando los mismos valores de corte y el mismo número de variables en cada caso que en las dos redes anteriores. El parámetro crítico en este tipo de redes es el número de agrupaciones en los que la red divide a los datos para solucionar el problema.

Otra vez, se puede pensar que cuanto mayor es el número de agrupaciones, mejor son los resultados de la red, pero eso no es así. Elevar excesivamente el número de agrupaciones, aparte de generar carga computacional, no mejora el resultado de la red, de hecho, si no es el mismo a partir de un punto, éste empeora.

En la tabla 4.3 presentamos los resultados de las redes anfis entrenadas:

REDES ANFIS ESTÁTICAS - REDES ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN R1.		
Variando el número de agrupaciones y coeficiente de correlación $R1 < 0.2$		
NÚMERO DE VARIABLES = 82		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	0.3183	0.5025
10	0.4443	1.0658
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.3$		
NÚMERO DE VARIABLES = 25		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	0.0605	0.8815
10	0.0605	0.8815
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.5$		
NÚMERO DE VARIABLES = 7		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	0.0608	0.8818
10	0.0608	0.8818

Tabla 4.3: Redes anfis estáticas entrenadas reduciendo variables con R.

De la tabla 4.3, es fácil ver que la red de tres agrupaciones de 82 variables presenta mejores resultados porque presenta mejor error de test. Los resultados gráficos de entrenamiento, validación y residuos generados por esta red se pueden ver en las gráficas 4.5a, 4.5b y 4.5c (entrenamiento) y 4.6a, 4.6b y 4.6c (validación).

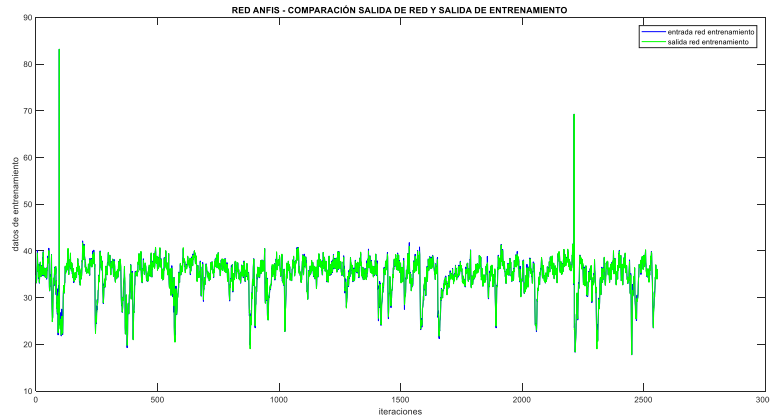


Figura 4.5a: Resultado del entrenamiento de la red estática anfis para predicción de DQO utilizando R para reducir variables

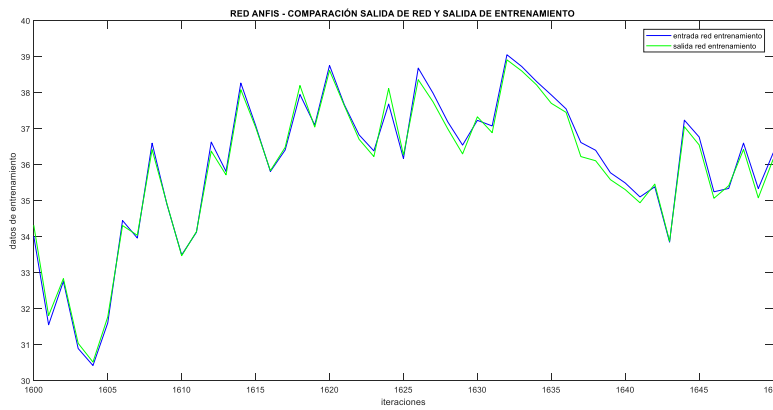


Figura 4.5b: Resultado del entrenamiento de la red estática anfis para predicción de DQO utilizando R para reducir variables

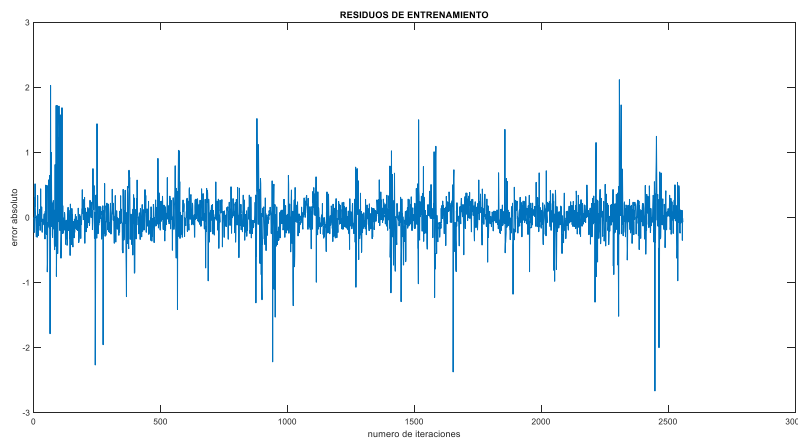


Figura 4.5c: Resultado del residuo del entrenamiento de la red estática anfis para predicción de DQO utilizando R para reducir variables

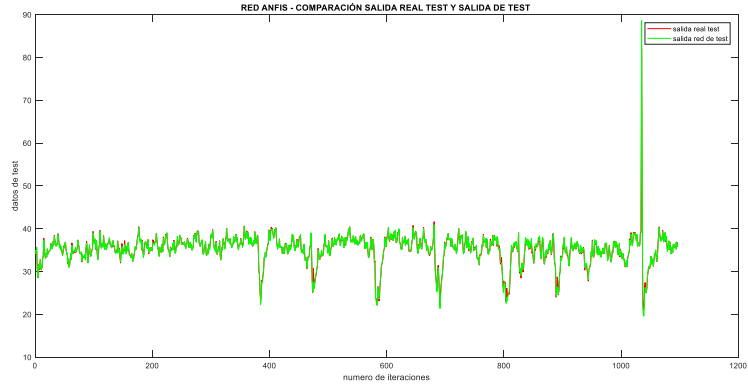


Figura 4.6a: Resultado del test de la red estática anfis para predicción de DQO utilizando R para reducir variables

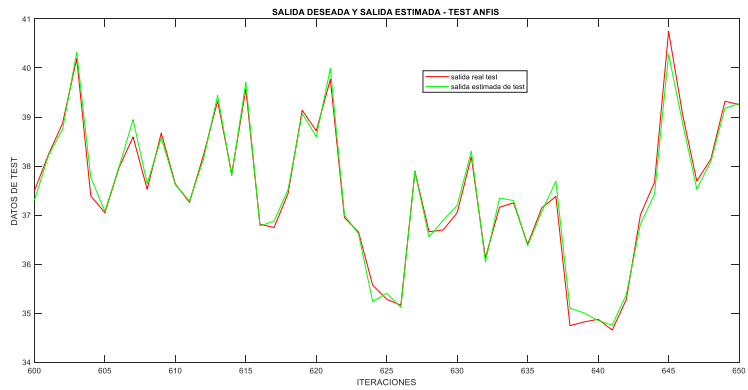


Figura 4.6b: Resultado del test de la red estática anfis para predicción de DQO utilizando R para reducir variables

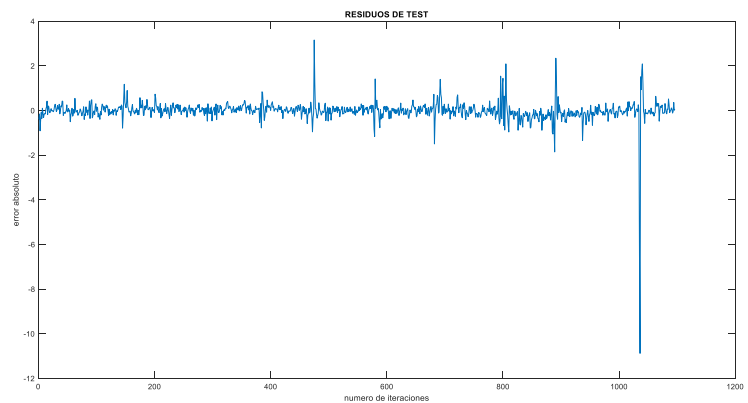


Figura 4.6c: Resultado del residuo de test de la red estática anfis para predicción de DQO utilizando R para reducir variables

Una observación visual de las gráficas nos permite deducir que las dos redes anteriores aprenden mejor que la red anfis ya que en esta última, la salida de la red (salida estimada) no se ajusta tan bien al patrón de salida (salida deseada). Aun así, consideramos que esta red ofrece un resultado aceptable.

La apreciación gráfica de los residuos de esta red da una idea de la diferencia entre las variables explicativas y la variable predicha. Aparecen picos relativamente más grandes en esta red que en las dos anteriores, lo cual indica que también aprendería peor.

4.3.1.4.- Máquinas de vectores de soporte (SVM) con el coeficiente de correlación (R).

En este caso, lo que utilizamos es la regresión de vectores de soporte (SVR) dado que estamos usando las SVM para resolver un problema de regresión. Al no ser un caso lineal, este método transforma el espacio inicial de entradas en un nuevo espacio utilizando el kernel.

Como en el resto de las redes estáticas, aplicamos la reducción dimensional utilizando el coeficiente de correlación con los mismos valores de corte y obteniéndose el mismo número de variables.

En este caso, el parámetro a tener en cuenta es el tipo de kernel, ya que de él depende el tipo de transformación que luego aplicará la red al espacio de variables de entrada. Hemos tenido en cuenta dos tipos de kernel como se puede ver en la tabla 4.4. En la misma tabla se puede ver las diferentes redes entrenadas en cada caso y los resultados obtenidos numéricamente.

Marcamos en amarillo la mejor red en cada caso. Por su inferior residuo de test, la mejor red es la de siete variables con kernel lineal.

Los resultados de entrenamiento de esta red se pueden ver en las gráficas 4.7a y 4.7b y los de validación en las gráficas 4.8a y 4.8b.

MÁQUINAS DE VECTORES DE SOPORTE (SVM) – REDES ESTÁTICAS ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN R1.		
Variando el tipo de función de ajuste coeficiente de correlación $R1 < 0.2$ NÚMERO DE VARIABLES = 82		
Kernel	Residuo entrenamiento	Residuo test
Lineal	2.3752	2.1399
gaussiano	2.5937	3.4158
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.3$ NÚMERO DE VARIABLES = 25		
Kernel	Residuo entrenamiento	Residuo test
Lineal	1.3633	1.5782
gaussiano	2.1777	2.6868
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.5$ NÚMERO DE VARIABLES = 7		
Kernel	Residuo entrenamiento	Residuo test
Lineal	0.0620	0.8627
gaussiano	1.2631	1.6100

Tabla 4.4: Regresión de vectores de soporte (SVR) estáticas entrenadas reduciendo variables con R.

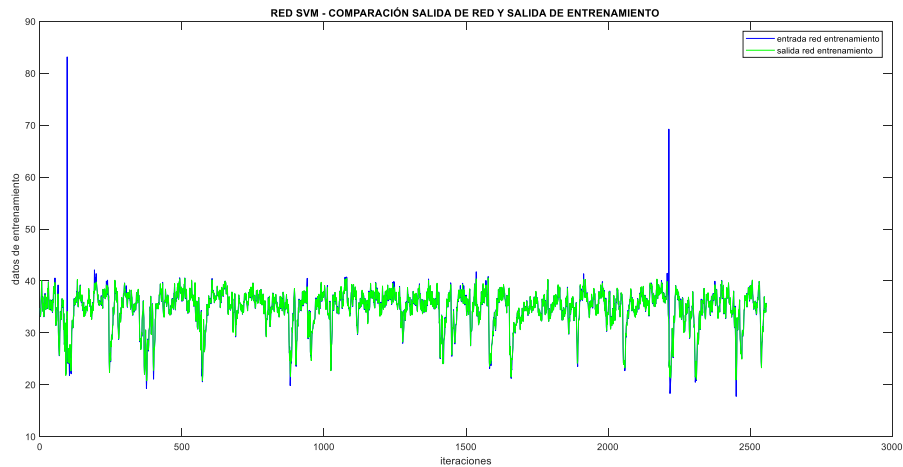


Figura 4.7a: Resultado de entrenamiento de SVR estática para predicción de DQO utilizando R para reducir variables

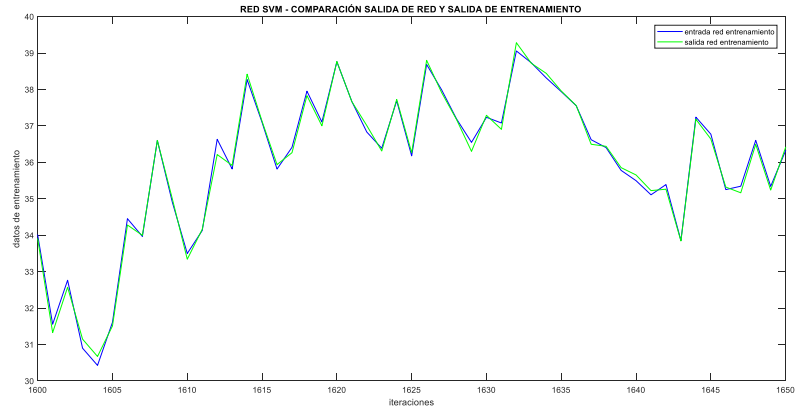


Figura 4.7b: Resultado de entrenamiento de SVR estática para predicción de DQO utilizando R para reducir variables

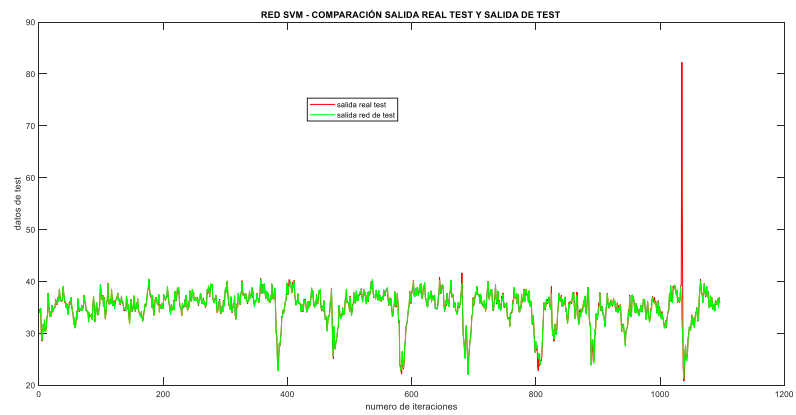


Figura 4.8a: Resultado de validación de la red estática SVR para predicción de DQO utilizando R para reducir variables

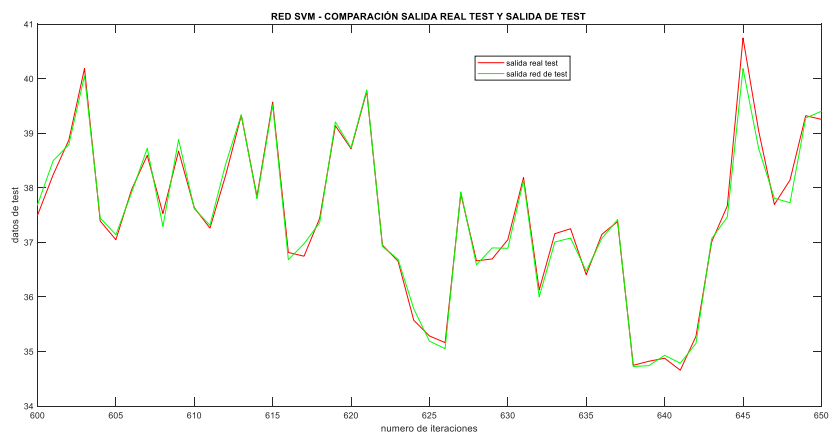


Figura 4.8b: Resultado de validación de la red estática SVR para predicción de DQO utilizando R para reducir variables

Aunque sea un juicio subjetivo, viendo los gráficos, se puede decir que con la SVR se obtiene un resultado bueno, puesto que la salida estimada se ajusta bastante bien al patrón de salida.

Cuando más adelante comparemos en una tabla los resultados de las diferentes redes sin normalizar ninguna, estaremos en condiciones de decir cuál de las redes se ajusta mejor a la salida.

4.3.2.4.- Combinación de las mejores redes estáticas reduciendo las variables con el coeficiente de correlación (R).

La combinación de las redes estáticas, en este caso, incluye a las mejores redes estáticas desarrolladas bajo el criterio de reducción dimensional del coeficiente de correlación.

Las mejores redes bajo este criterio de reducción dimensional son todas las que hemos ido presentando los resultados gráficos en las secciones anteriores.

Lo que se pretende es saber si la combinación de todas ellas da mejor resultado que cada una las redes de forma aislada.

Para ensamblar las redes, lo que vamos a hacer es desnormalizar las dos redes (red MLP y la red RBF) que habíamos normalizado entre -1 y 1 para evitar hacer una comparación sesgada y para que todas tengan la misma escala de datos.

Se plantean dos tipos combinaciones:

- **Una primera combinación** en la que todas las redes tienen el mismo factor (asignación porcentual). Eso es válido tanto para el entrenamiento como para la validación. De esta forma tendremos:

$$Y_{combinada_{entrenamiento}} = 0.25redff + 0.25redRBF + 0.25redanfis + 0.25SVM \quad (4.1)$$

$$Y_{combinada_{test}} = 0.25redff + 0.25redRBF + 0.25redanfis + 0.25SVM \quad (4.2)$$

Los resultados gráficos de entrenamiento y validación de esta combinación se pueden ver en las gráficas 4.9a, 4.9b, 4.10a y 4.10b.

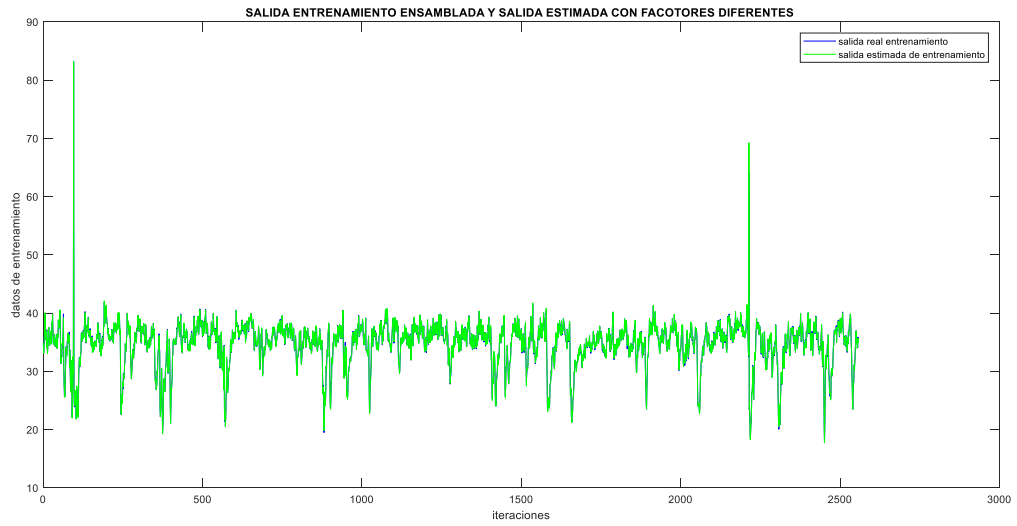


Figura 4.9a: Resultado del entrenamiento de la combinación de redes estáticas con factores iguales para predicción de DQO utilizando R para reducir variables

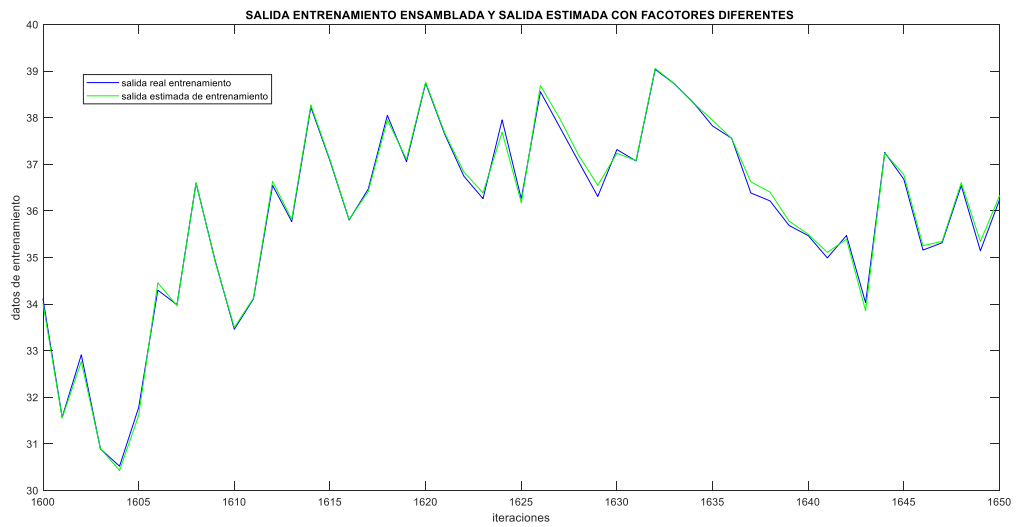


Figura 4.9b: Resultado del entrenamiento de la combinación de redes estáticas con factores iguales para predicción de DQO utilizando R para reducir variables

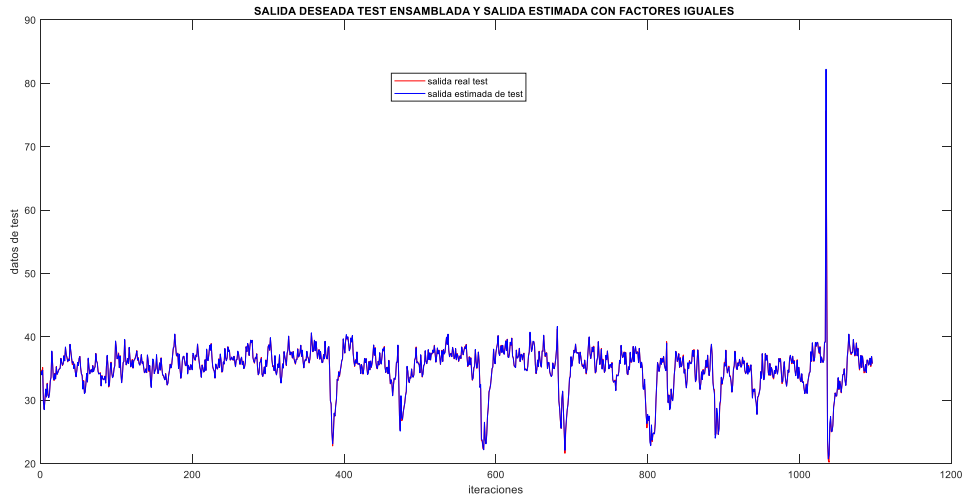


Figura 4.10a: Resultado de la validación de la combinación de redes estáticas con factores iguales para predicción de DQO utilizando R para reducir variables

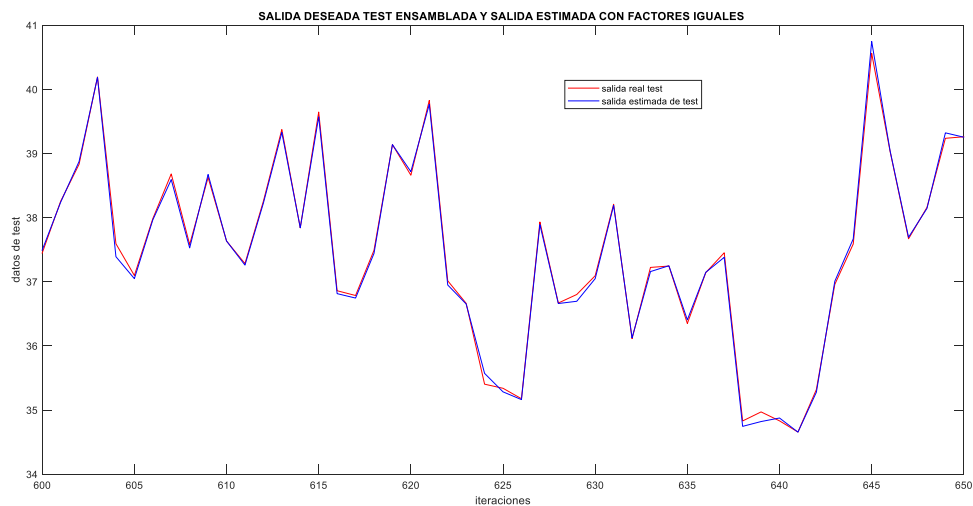


Figura 4.10b: Resultado de la validación de la combinación de redes estáticas con factores iguales para predicción de DQO utilizando R para reducir variables

- **La segunda combinación** supone buscar la mejor combinación de los factores posibles que mejore los resultados de la red. Esta combinación supone dar más peso a la red con la que mejor resultado de test se ha obtenido, un segundo peso a la siguiente mejor red, y así sucesivamente, pero teniendo en cuenta que la suma total de pesos sea uno. Esta combinación será la misma tanto para el entrenamiento que para la validación, así pues, tenemos:

$$Y_{combinada_{entrenamiento}} = 0.45redff + 0.10redRBF + 0.35redanfis + 0.10SVM \quad (4.3)$$

$$Y_{combinada_{test}} = 0.45redff + 0.10redRBF + 0.35redanfis + 0.10SVM \quad (4.4)$$

Las gráficas así obtenidas se pueden ver en las figuras 4.11a, 4.11b, 4.12a y 4.12b.

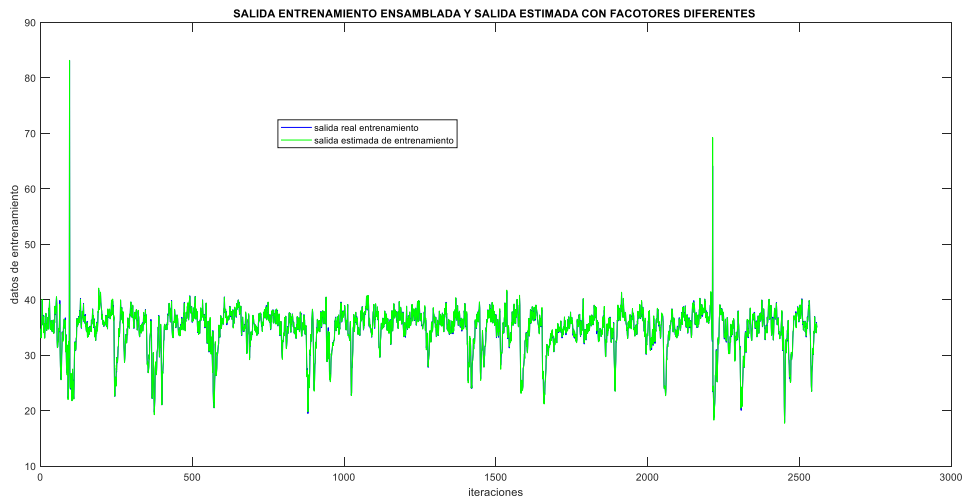


Figura 4.11a: Resultado del entrenamiento de la combinación de redes estáticas con factores diferentes para predicción de DQO utilizando R para reducir variables

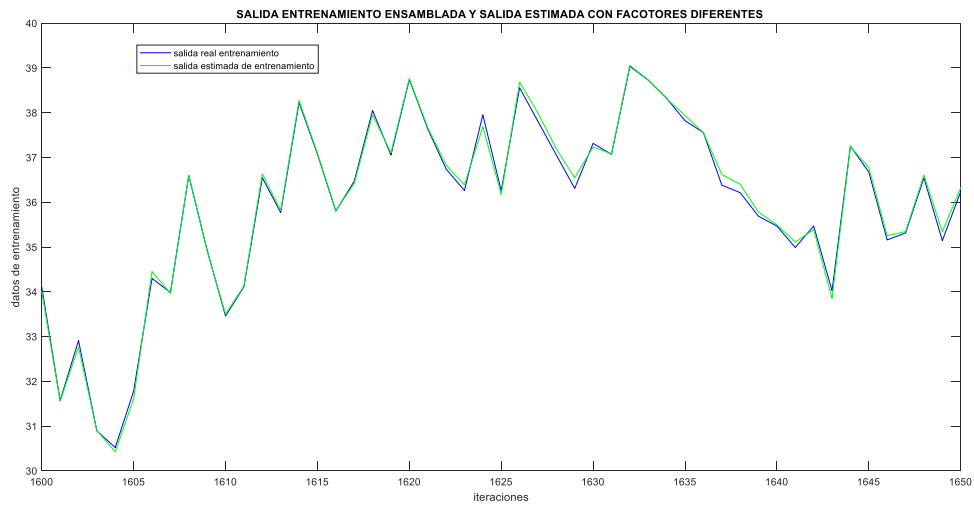


Figura 4.11b: Resultado del entrenamiento de la combinación de redes estáticas con factores diferentes para predicción de DQO utilizando R para reducir variables

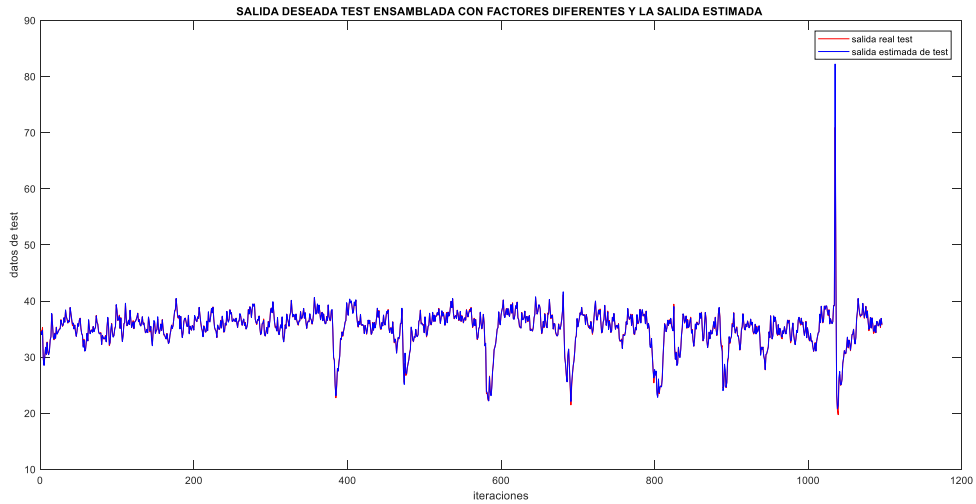


Figura 4.12a: Resultado de la validación de la combinación de redes estáticas con factores diferentes para predicción de DQO utilizando R para reducir variables.

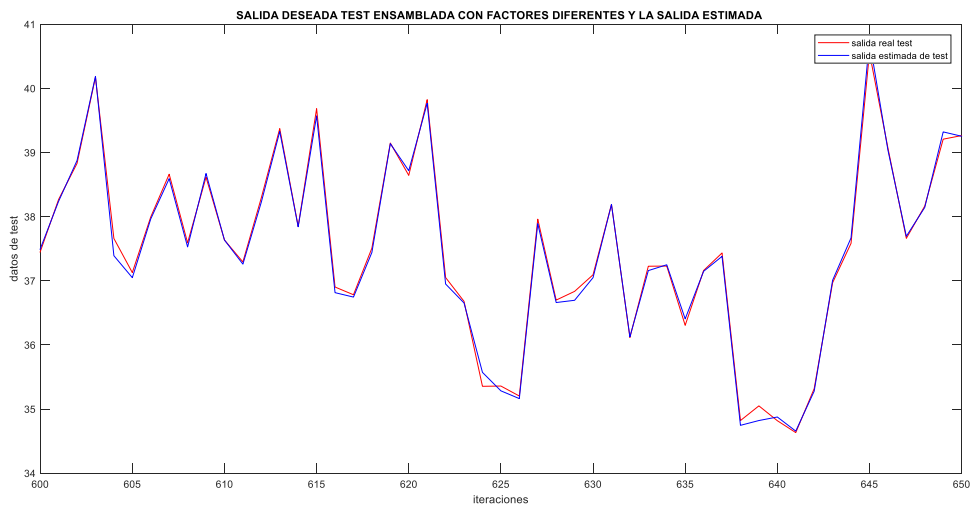


Figura 4.12b: Resultado de la validación de la combinación de redes estáticas con factores diferentes para predicción de DQO utilizando R para reducir variables.

Mirando las gráficas, la sensación es que esta red mejora los resultados de las RBF y SVM, pero no tanto los de las redes MLP y la anfis. Más adelante, haremos una comparación basada en los residuos de cada red para aclarar las diferencias entre unas redes y otras.

Presentamos la distribución gráfica de los residuos de estas dos combinaciones (ensambles) para el entrenamiento y validación.

4.3.2.- Reducción dimensional basado en el análisis de componentes principales (PCA).

El segundo método de reducción dimensional que hemos utilizado para las redes estáticas es el PCA. Como ya se explicó en el capítulo II en el apartado correspondiente al PCA, es una técnica estadística que, partiendo de un espacio inicial con muchas variables, algunas o muchas de ellas correlacionadas entre sí, conduce a un espacio de variables más reducido poco o nada correlacionadas. Las variables del nuevo espacio reducido denominados componentes principales (PC), son una consecuencia de la combinación lineal de las variables del espacio original de trabajo, y se colocan en orden decreciente según su varianza, de forma que, el componente principal tiene mayor varianza, el segundo, tiene la segunda mayor varianza, y así sucesivamente.

Partimos de un conjunto de 140 variables, al hacer la transformación mediante PCA, nos quedamos con las primeras 20 PC (variables) que representan el 99,5230% de la varianza en muchos casos, por considerar que este número de variables ofrecía una solución aceptable a nuestro problema; y en otros casos 30 PC, que representan el 99,9388% de la varianza total, siempre que sirviera para mejorar los resultados.

Un gráfico muy elocuente es el de la figura 4.13 (la mostramos ampliada para que se vea mejor), en las ordenadas tenemos el porcentaje de distribución de los datos que representa cada componente principal. El primer PC concentra en torno al 39% de los datos, el segundo entorno al 19%, el tercero un 13% aproximadamente, etc.

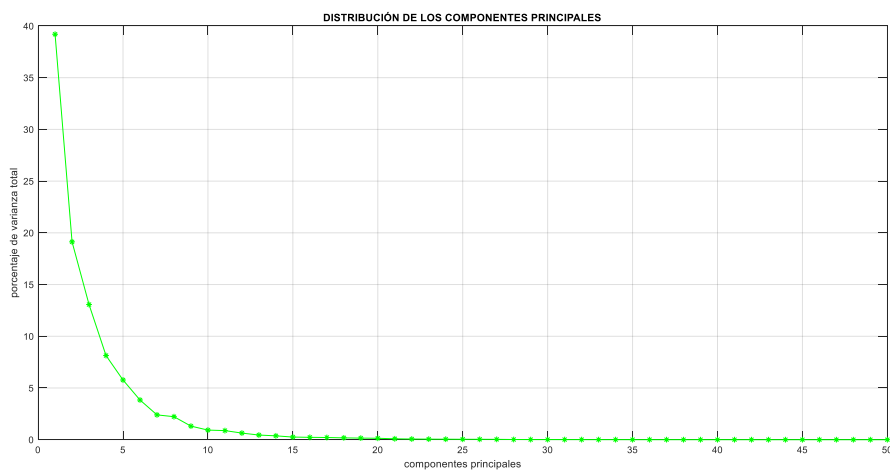


Figura 4.13: Distribución de los componentes principales PC frente al porcentaje de la varianza total

Se puede ver en la figura 4.13 que, prácticamente a partir del PC 25, el porcentaje de varianza total que concentra los siguientes PC se reduce a cero, con lo cual, influyen poco. Aun así, en algunos casos cogimos hasta 30 PC si era necesario.

Como en el otro criterio de reducción dimensional, entrenamos las redes MLP, las redes de funciones de base radial (RBF), las redes neuro difusas anfis y las máquinas de soporte vectorial.

A continuación, presentamos los resultados obtenidos con las diferentes redes.

4.3.2.1.- Red perceptrón multicapa (MLP) con PCA.

Esta red, que ya explicamos anteriormente, la hemos entrenado con 20 componentes principales, y los resultados de las redes entrenadas se muestran en la tabla 4.5.

REDES MLP ESTÁTICAS - REDES ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN PCA.		
NÚMERO DE COMPONENTES PRINCIPALES (PC) = 20		
Variando número de neuronas y reducción mediante PCA		
N.º neuronas	Residuo entren.	Residuo test
5	0.0025	0.0288
10	0.0027	0.0218
15	0.0020	0.0159

Tabla 4.5: Redes MLP estáticas entrenadas reduciendo variables con PCA.

En este caso, como en el anterior, el parámetro crítico es el número de neuronas. Entrenamos varias redes, aunque en la tabla 4.5 sólo mostramos tres: la red con 5,10 y 15 neuronas, obteniendo en términos de residuos, los resultados que en la tabla 4.5 se ven.

La que mejor resultado ofrece entre las tres fue la de 15 neuronas, que hemos resaltado en amarillo. Los resultados gráficos de esta red los mostramos a continuación en la figura 4.14a, 4.14b para el entrenamiento y 4.15a, 4.15b para la validación.

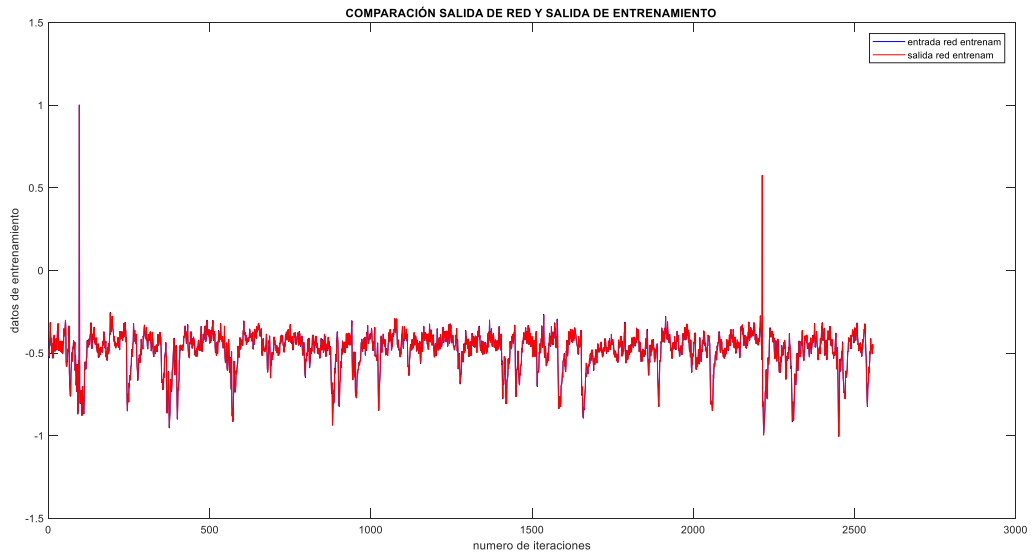


Figura 4.14a: Resultado del entrenamiento de la red estática MLP para predicción de DQO reduciendo variables con PCA.

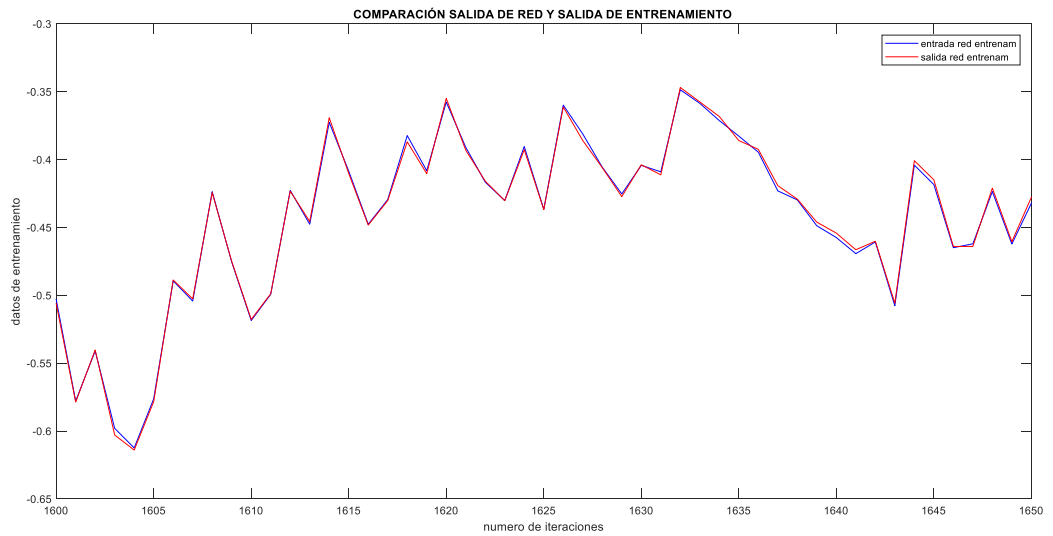


Figura 4.14b: Resultado del entrenamiento de la red estática MLP para predicción de DQO reduciendo variables con PCA.

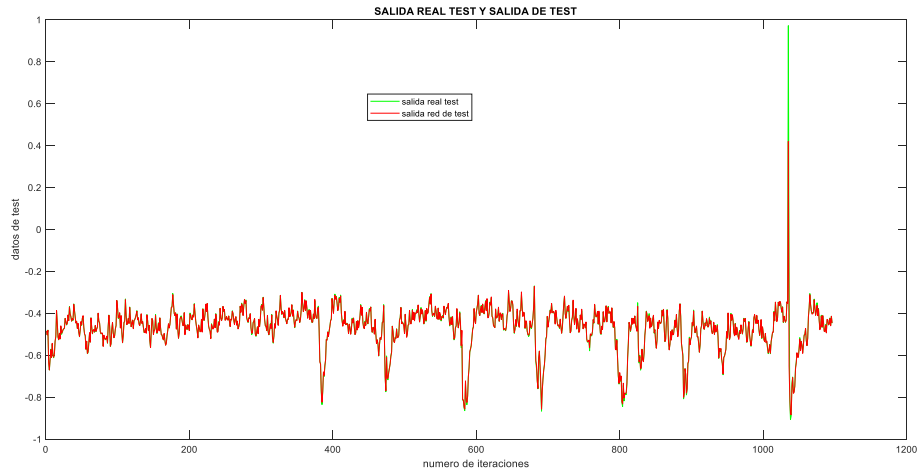


Figura 4.15a: Resultado de test de la red estática MLP para predicción de DQO reduciendo variables con PCA.

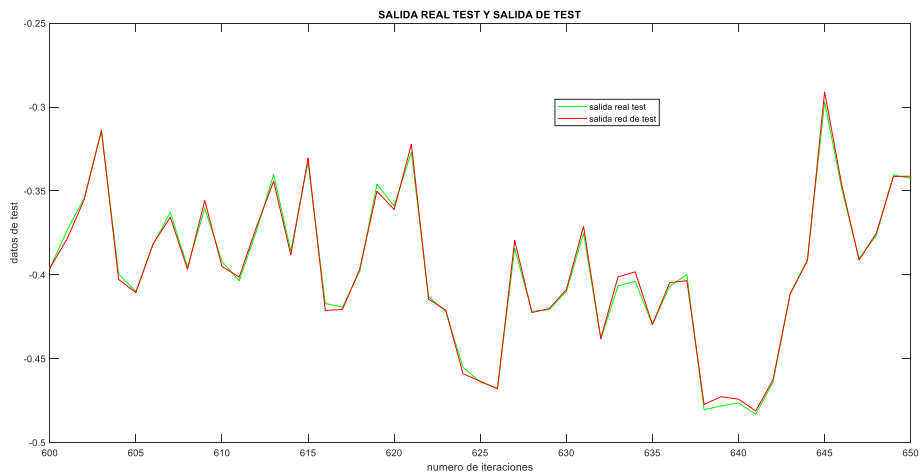


Figura 4.15b: Resultado de test de la red estática MLP para predicción de DQO reduciendo variables con PCA.

Como se puede apreciar en las gráficas, tanto en el entrenamiento como en la validación, la salida estimada se ajusta bastante bien con la salida deseada.

4.3.2.2.- Red de funciones de base radial (RBF) con PCA

El principio de funcionamiento de esta red también lo explicamos en el capítulo II cuando explicamos las redes de funciones de base radial. Sobre su utilidad en el diseño de nuestra aplicación, también queda explicado en el punto 4.3.1.2, con la diferencia de

que esta vez el método de reducción dimensional cambia y con él los resultados, como también cambian los parámetros objetivos (“goal”).

Las redes entrenadas en este caso los recogemos en la tabla 4.6.

REDES DE FUNCIÓN DE BASE RADIAL (RBF) ESTÁTICAS - REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN PCA. NÚMERO DE PC = 20			
Variando solo el goal y reducción mediante PCA			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7e-4	0.0263	0.0465	0.00147833
5e-4	0.0224	0.0452	0.000605984
6e-5	0.0077	0.0408	6.85514e-05

Tabla 4.6: Redes RBF estáticas entrenadas reduciendo variables con PCA.

Por su menor residuo en la validación, aunque en este caso, también tiene el menor residuo de entrenamiento, la mejor red es la que resaltamos en amarillo. Los resultados gráficos de ésta los presentamos en las figuras 4.16a, 4.16b, para la validación.



Figura 4.16a: Resultado del test de la red estática RBF para predicción de DQO reduciendo variables con PCA.

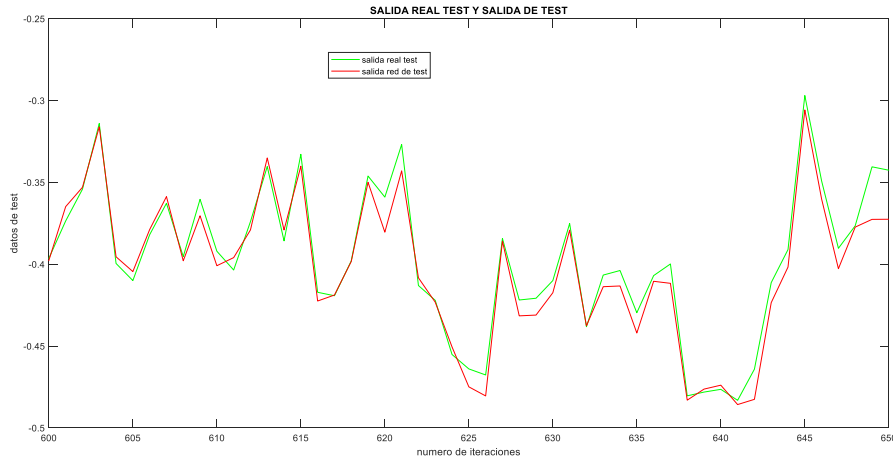


Figura 4.16b: Resultado del test de la red estática RBF para predicción de DQO reduciendo variables con PCA.

Como se puede apreciar en las gráficas, con esta red no se consiguió un resultado muy bueno, ya que la salida estimada por la red no se ajusta bien con la salida real (salida deseada).

4.3.2.3.- Red neuro-difusa anfis con PCA.

La red anfis también fue implementada bajo el criterio de reducción dimensional basado en el PCA. Se utilizó también 20 componentes principales, porque con ellos se obtenía mejor resultado.

Como esta red ya fue explicada en el capítulo II y comentada en el punto 4.3.1.3, nos limitamos a presentar sus resultados.

Como en el otro caso de reducción dimensional, el parámetro crítico es el número de agrupaciones. Se utilizó 10 y 100 agrupaciones.

Presentamos en la tabla 4.7 los resultados de las redes estáticas anfis con reducción dimensional basado en la PCA entrenadas:

REDES ANFIS ESTÁTICAS- REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN PCA. NÚMERO DE PC = 20		
Variando el número de agrupaciones y reducción mediante PCA		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
10	0.9563	1.1605
100	1.0077	1.0888

Tabla 4.7: Redes anfis estáticas entrenadas reduciendo variables con PCA.

El mejor resultado, puesto que mejora ligeramente el resultado sin causar pérdidas de tiempo en el entrenamiento, es la red resaltada en amarillo de 100 agrupaciones.

Los resultados gráficos de validación de esta red (la mejor red) se pueden ver en las gráficas 4.17a y 4.17b.

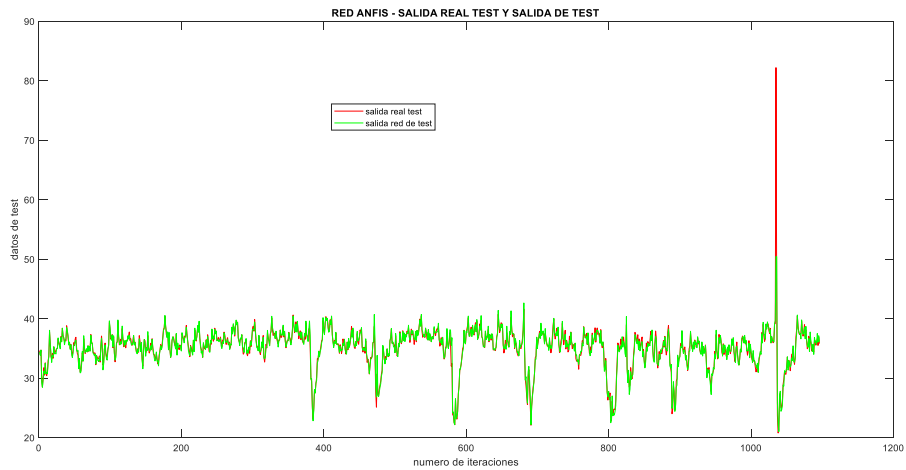


Figura 4.17a: Resultado del test de la red estática anfis para predicción de DQO reduciendo variables con PCA.

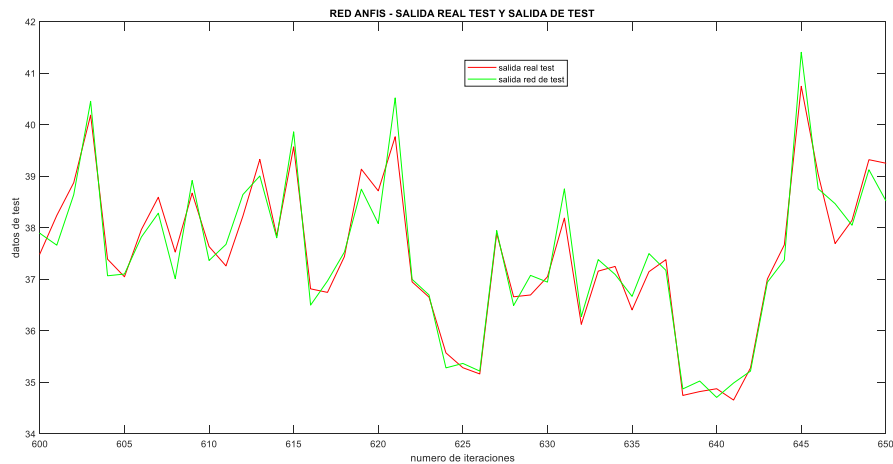


Figura 4.17b: Resultado del test de la red estática anfis para predicción de DQO reduciendo variables con PCA.

Un juicio subjetivo, pero no menos importante, sobre los resultados gráficos de esta red nos permiten decir que, con esta red, al igual que con la red RBF con el PCA, no se

obtiene un resultado óptimo, dado que el patrón maestro de salida y la salida estimada por la red, no se ajustan bien.

Luego presentaremos unos resultados con PCA más objetivos con datos sobre los residuos de esta red junto con el resto de las redes.

4.3.2.4.- Máquinas de vectores de soporte (regresión de vectores de soporte) con PCA.

Las máquinas de soporte vectorial también fueron entrenadas con 20 PC. Como ya explicamos, el parámetro crítico es el kernel. Se utilizó los kernel lineal y el gaussiano obteniéndose los resultados que en la tabla 4.8.

MÁQUINAS DE VECTORES DE SOPORTE (SVM) – REDES ESTÁTICAS ENTRENADAS NÚMERO DE COMPONENTES PRINCIPALES (PC) = 20		
Variando el tipo de función de ajuste y reducción mediante PCA		
Kernel	Residuo entrenamiento	Residuo test
Lineal	1.3110	1.5114
gaussiano	2.5911	3.4057

Tabla 4.8: Redes anfis estáticas entrenadas reduciendo variables con PCA.

La mejor SVR se resalta en amarillo. Con la SVR se obtuvo los resultados de validación que se muestran en las figuras 4.18a y 4.18b.

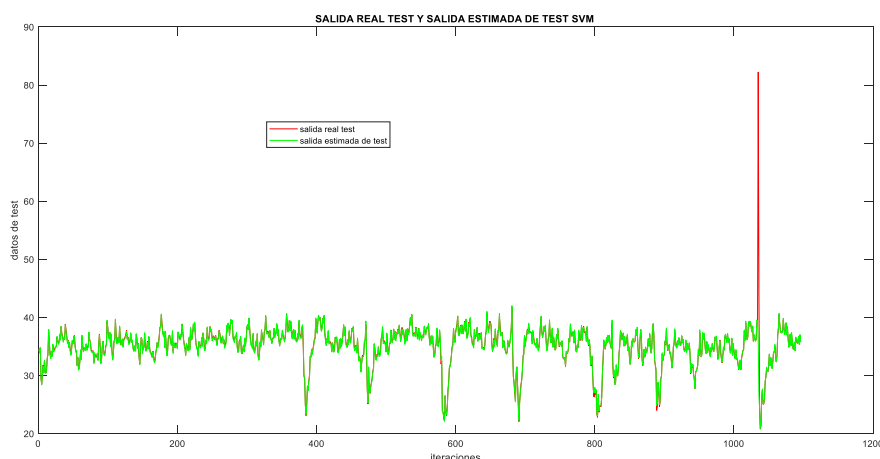


Figura 4.18a: Resultados del test de la SVR estática para predicción de DQO reduciendo variables con PCA.

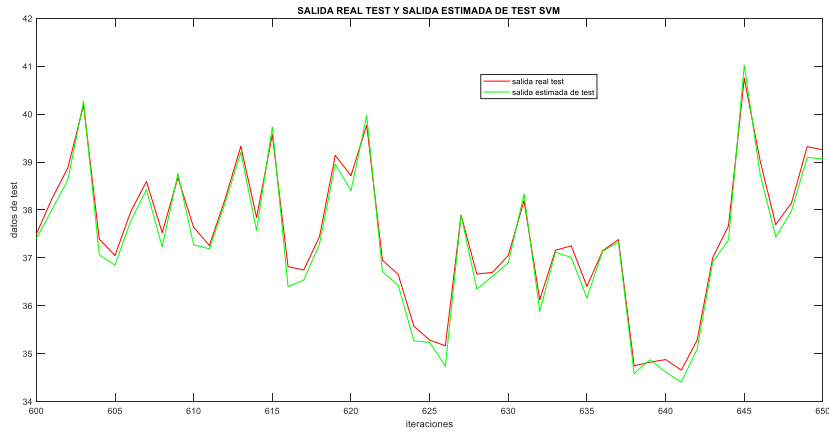


Figura 4.18b: Resultados del test de la SVR estática para predicción de DQO reduciendo variables con PCA.

Por lo que se aprecia en los resultados gráficos de esta red, se puede decir que presenta un resultado aceptable porque la salida deseada y la salida patrón se ajustan relativamente bien.

4.3.2.5.- Combinación de las mejores redes con PCA

En este caso se utiliza las mejores redes bajo el criterio de reducción dimensional del PCA y se las combina utilizando los mismos factores para todas las redes en un primer caso; en el segundo caso se utilizan diferentes factores, igual que antes.

$$Y_{combinada_{test}} = 0.25redff + 0.25redRBF + 0.25redanfis + 0.25SVM \quad (4.5)$$

Las gráficas así obtenidas se pueden ver en las figuras 4.19a y 4.19b, para validación.

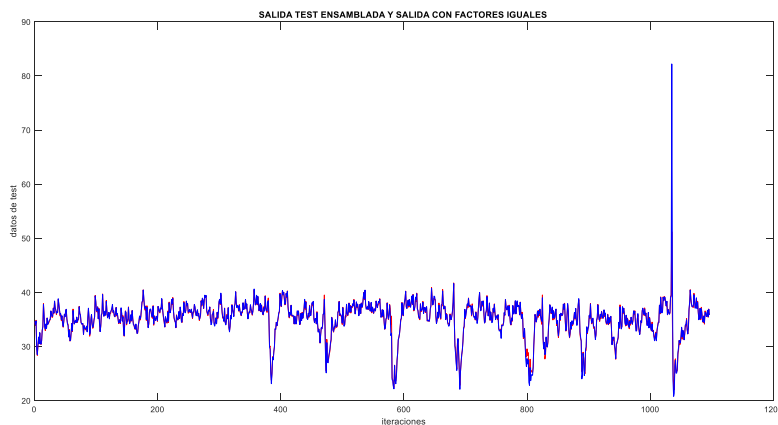


Figura 4.19a: Resultado del test de la combinación de redes estáticas para predicción de DQO con factores iguales reduciendo variables con PCA.

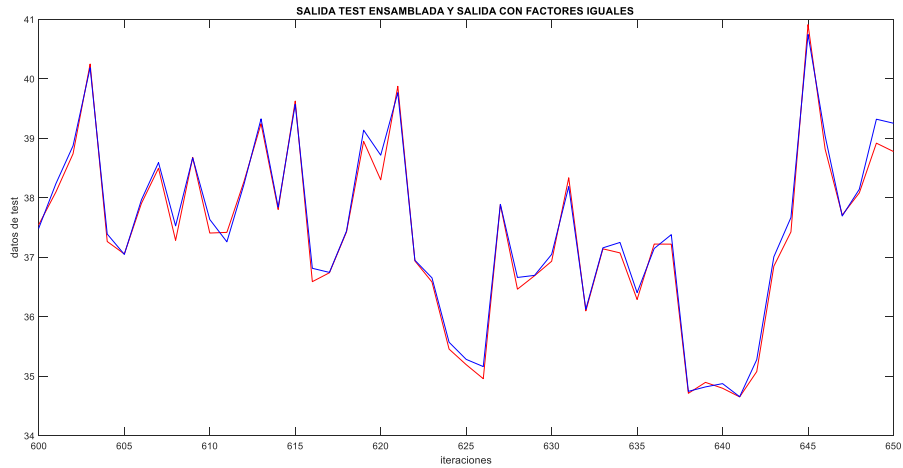


Figura 4.19b: Resultado del test de la combinación de redes estáticas para predicción de DQO con factores iguales reduciendo variables con PCA.

En las figuras 4.19c y 4.19d, se pueden ver los resultados gráficos del ensamble con factores diferentes.

$$Y_{combinada_{test}} = 0.45redff + 0.25redRBF + 0.20redanfis + 0.10SVM \quad (4.6)$$

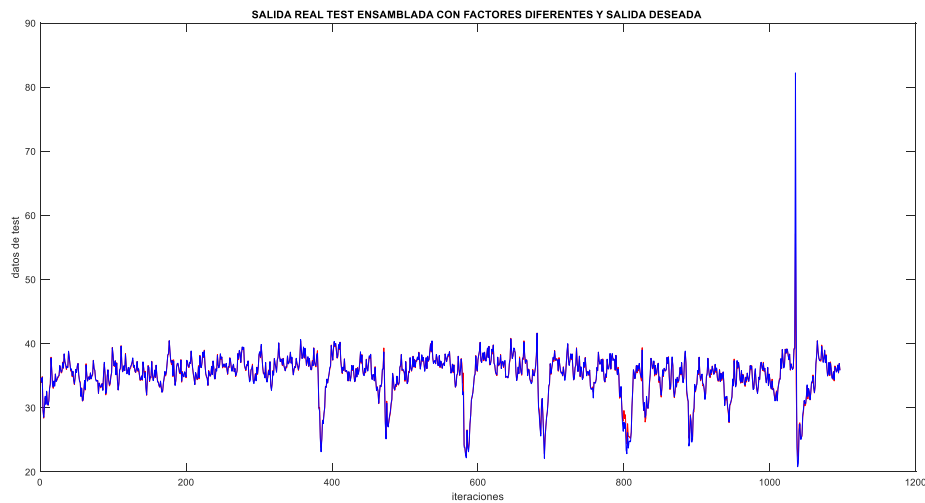


Figura 4.19c: Resultado del test de la combinación de redes estáticas para predicción de DQO con factores diferentes reduciendo variables con PCA.

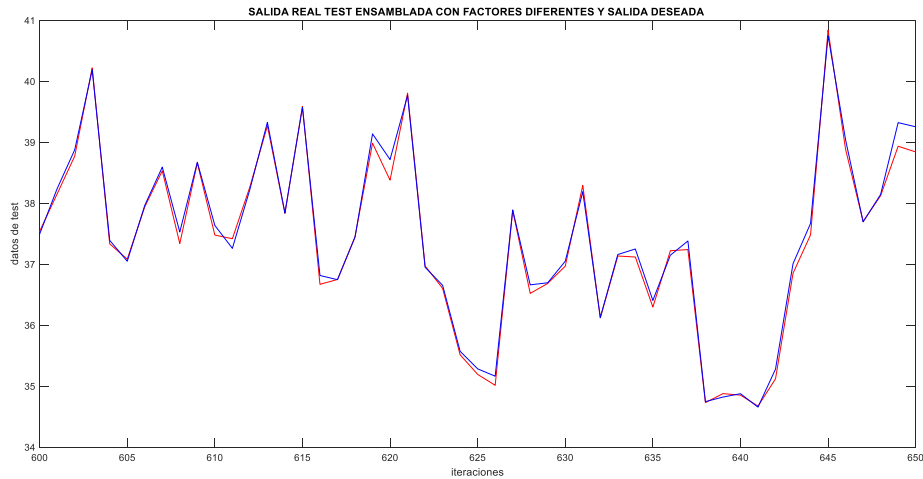


Figura 4.19d: Resultado del test de la combinación redes estáticas para predicción de DQO con factores diferentes. reduciendo variables con PCA.

Observando las gráficas, la primera impresión es que la red resultante de la combinación con el PCA ofrece peor resultado que las combinadas con el coeficiente de correlación (R). Eso puede deberse a que las redes desarrolladas bajo el criterio de R ofrecen mejores resultados individuales que las desarrolladas con el PCA, de forma que, al combinarlas también obtienen mejor resultado que las del PCA. Por otro lado, una comparación gráfica entre las redes con el PCA y la combinación de estas lleva a pensar que la combinación es mejor que cada una de redes individuales, cuestión que por otro lado coincidiría con la teoría. un resumen numérico de estas apreciaciones se discute en la sección 4.3.3.

Comparando los dos ensambles, la sensación es que el ensamble con factores diferentes es relativamente mejor. Con los residuos se podrá sacar una conclusión más objetiva.

4.3.3.- Comparación de las redes estáticas

En este punto presentamos todas las mejores redes estáticas y la combinación de éstas en cada caso, sus correspondientes resultados numéricos en residuos de entrenamiento y validación. Sus resultados gráficos ya los hemos expuesto anteriormente.

REDES ESTÁTICAS CON DATOS REDUCIDOS MEDIANTE COEFICIENTE DE CORRELACIÓN (R)		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (10neuronas) $R1 < 0.2$	0.2929	0.3631
Funciones de base radial (goal = $2.5e-6$) $R1 > 0.5$	0.0506	1.3187
Anfis (10 agrupaciones) $R1 < 0.2$	0.3183	0.5025
SVM (kernel lineal) $R1 > 0.5$	0.0620	0.8627
Ensamble (combinación) con factores iguales	0.1279	0.3241
Ensamble (combinación) con factores diferentes	0.1984	0.3968
REDES ESTÁTICAS CON DATOS REDUCIDOS MEDIANTE PCA		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (15 neuronas) 20 PC	0.0979	0.6654
Funciones de base radial (goal = $6e-5$) 20 PC	0.2525	1.3348
Anfis (100 agrupaciones) 20 PC	1.0077	1.0888
SVM (kernel lineal) 20 PC	1.3110	1.5114
Ensamble (combinación) con factores iguales	0.5489	1.0223
Ensamble (combinación) con factores diferentes	0.4652	0.8652

Tabla 4.9: Comparación de las mejores redes estáticas

Lo que se deduce de la tabla 4.9, fijándonos en el residuo de validación, es que, con la reducción del conjunto de variables con R, la mejor red se consigue con el ensamble con factores iguales; sin embargo, utilizando el PCA para reducir variables, el mejor resultado lo obtenemos con la red MLP de 15 neuronas. Esta red de 15 neuronas se corresponde con la mejor red entre las tres presentadas en la tabla 4.1 con $R1 > 0,3$. Si a su vez comparamos estas dos redes, es evidente que la red con mejor residuo de test, y, en consecuencia, la mejor de todas redes estáticas, es la del ensamble con factores iguales. Resaltamos en amarillo las dos mejores redes.

Por otro lado, lo que también se puede deducir de esos resultados es que, en general, los mejores resultados, por su reducido residuo de test, se obtienen con las redes combinadas.

Es especialmente importante insistir en que comparamos todas las redes sin normalizar. No tiene sentido comparar redes normalizadas con otras sin normalizar, porque las escalas en las que se sitúan los datos en ambos casos no es el mismo, por lo que se sacarían conclusiones poco o nada consistentes y la comparación, sería, como ya dijimos, sesgada.

4.4.- REDES NEURONALES ARTIFICIALES DINÁMICAS.

Hasta ahora sólo hemos presentado los resultados de las redes neuronales artificiales estáticas, cuyo principio de funcionamiento se basa en que, ante unas entradas, la red produce unas salidas. No hay realimentación de información de ningún tipo. en este punto presentamos los resultados obtenidos con las redes neuronales artificiales dinámicas.

Las redes neuronales artificiales dinámicas son una generalización de las redes estáticas. La diferencia entre ellas es que, en las redes dinámicas, aparte de propagar información hacia adelante, las neuronas son capaces de retornar información a sí mismas, a neuronas de otras capas anteriores o a neuronas de la misma capa. Por tanto, ante unas entradas, la red dinámica producirá una salida basada en la entrada actual y en las entradas pasadas, lo que supone una evolución temporal. Matemáticamente, las entradas a la red tienen la forma de la expresión 4.7.

$$X_{Dinámico} = [x(t) \ x(t - 1) \ y(t - 1) \ x(t - 2) \ y(t - 2), \dots, x(t - n) \ y(t - n)] \quad (4.7)$$

donde:

n: instante de tiempo cualquiera

x(t) : entrada actual

y(t): salida actual

x(t - n): entrada anterior

y(t - n): salida anterior

Como ya explicamos las redes dinámicas en el punto 2.3.1.2.2, no nos vamos a extender, pero señalar que, en teoría, estas redes dan mejores resultados en la resolución de los problemas no lineales. Estamos ante un problema no lineal, no podemos evitar la tentación de comparar los resultados de ambas redes. Más adelante presentaremos una tabla comparativa que incluya la mejor red de cada tipo. Lo que también haremos es comparar, por cada tipo de red (MLP, RBF, anfis y SVM), la mejor de ellas en las redes estáticas y las dinámicas; en esta comparación nos fijaremos básicamente en el error cuadrático medio (MSE) del residuo de validación de cada red y será mejor la que menor valor presente.

Con las redes dinámicas haremos lo mismo que hemos hecho con las redes estáticas.

4.4.1.- Reducción dimensional del conjunto de variables con R.

Como ya lo hicimos con las redes estáticas, analizamos las redes dinámicas de la misma manera, es decir, reducimos el conjunto inicial de variables utilizando el coeficiente de

correlación, después haríamos lo mismo utilizando el PCA, después compararemos los resultados.

4.4.1.1- Red perceptrón multicapa (MLP) dinámica reduciendo variables con R.

En principio, estas redes son de naturaleza estática, pero una modificación de éstas permite que tengan un comportamiento dinámico. Como decimos, se ha procedido de la misma forma que en las redes estáticas MLP tanto con la reducción dimensional del conjunto variables utilizando el coeficiente de correlación (R) que mediante la aplicación del análisis de componentes principales (PCA).

Esta red está normalizada, y en sus gráficas, que más abajo presentaremos, se podrá ver que los datos están en un rango que va de -1 hasta 1. Para el ensamble la desnormalizaremos para que tenga el mismo rango de datos que el resto de las redes. La comparación con la correspondiente red estática MLP se hará con las redes normalizadas.

En la tabla 4.10 presentamos las diferentes redes MLP dinámicas entrenadas.

REDES MLP DINÁMICAS - REDES ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN R1.		
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 < 0.2$		
NÚMERO DE VARIABLES = 420		
N.º neuronas	Residuo entren.	Residuo test
2	0.0253	0.0123
10	0.0088	0.0085
15	0.0087	0.0104
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 > 0.5$		
NÚMERO DE VARIABLES = 55		
N.º neuronas	Residuo entren.	Residuo test
2	0.0021	0.0257
10	0.0029	0.0162
15	0.0014	0.0236
VARIANDO NÚMERO DE NEURONAS Y COEFICIENTE DE CORRELACIÓN $R1 > 0.6$		
NÚMERO DE VARIABLES = 6		
N.º neuronas	Residuo entren.	Residuo test
2	0.0426	0.0501
10	0.0406	0.0497
15	0.0407	0.0474

Tabla 4.10: Redes MLP dinámicas entrenadas reduciendo variables con R.

Como antes, el parámetro distintivo es el número de neuronas. La mejor red en este caso es la red MLP de 10 neuronas con $R1 < 0,2$. Los residuos de entrenamiento y validación de esta red pueden verse en la tabla 4.10. La apariencia gráfica de esta red para entrenamiento y validación se puede ver en las figuras: 4.20a, 4.20b, 4.21a, 4.21b.

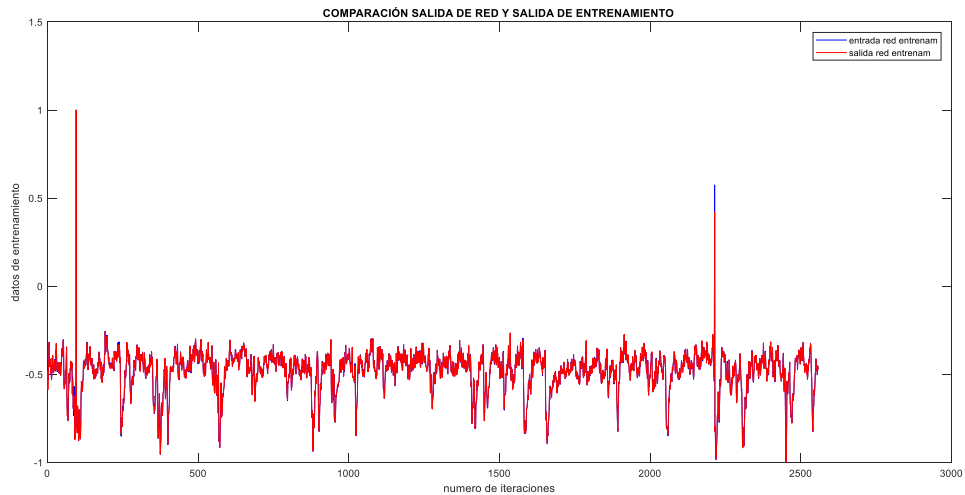


Figura 4.20a: Resultado del entrenamiento de la red dinámica MLP para predicción de DQO reduciendo variables con R.

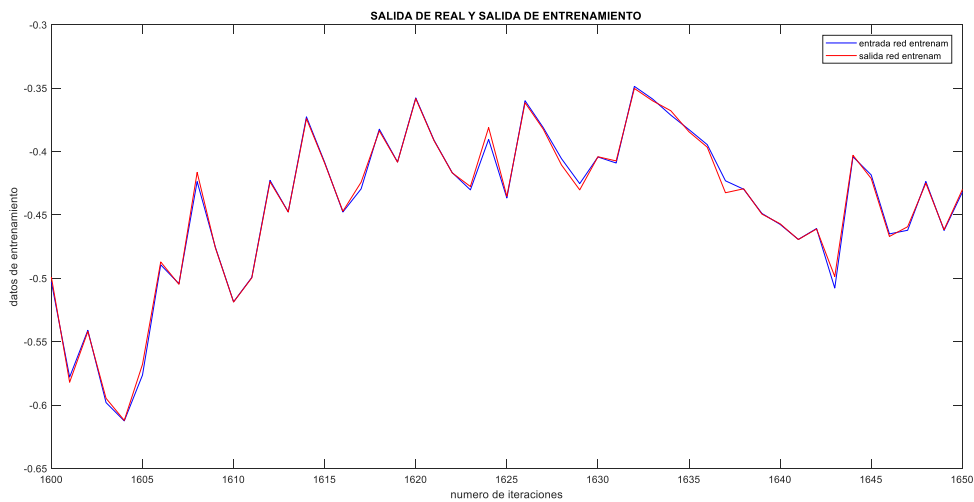


Figura 4.20b: Resultado del entrenamiento de la red dinámica MLP para predicción de DQO reduciendo variables con R.

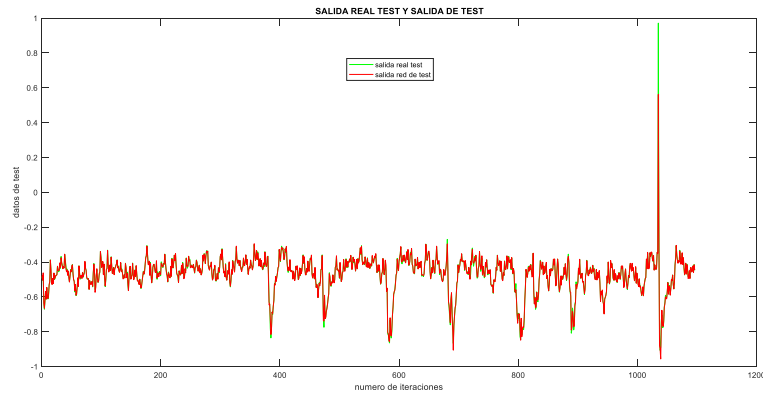


Figura 4.21a: Resultado de la validación de la red dinámica MLP para predicción de DQO reduciendo variables con R.

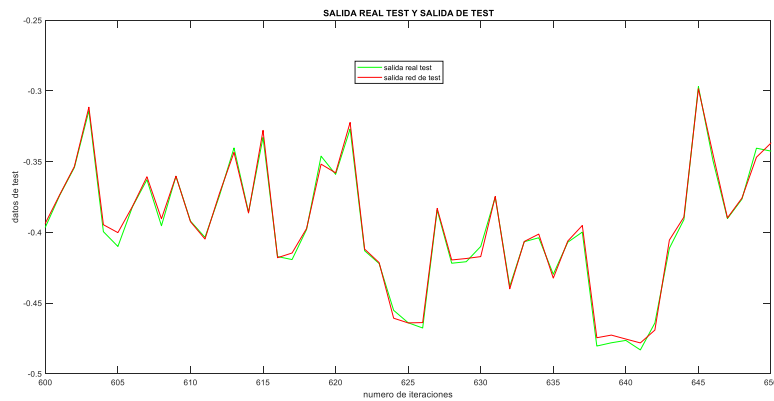


Figura 4.21b: Resultado de la validación de la red dinámica MLP para predicción de DQO reduciendo variables con R.

Lo que se desprende de la observación gráfica de esta red es que presenta mejor resultado de validación que de entrenamiento, eso significa que se entrena algo peor y valida ligeramente mejor, lo cual es bueno. También hay que decir que no es lo que se ve en muchos casos. Lo habitual es que sea al revés.

Ahora podemos comparar esta red MLP dinámica con la MLP estática para ver si la primera mejora los resultados de la segunda. Hacemos esta comparación basándonos en los residuos de cada red.

Red MLP	N.º neuronas	Residuo entren.	Residuo test
estática	10	0.0065	0.0082
dinámica	10	0.0088	0.0085

Tabla 4.11: Comparación entre la mejor red MLP estática y la mejor red MLP dinámica reduciendo variables con R.

Por los residuos, es evidente que ni la red dinámica MLP entrena mejor que la estática, ni valida mejor que ésta. Por casualidad, las dos mejores redes MLP tienen el mismo número de neuronas. Por los residuos de test podemos afirmar que la red MLP estática ofrece mejor resultado.

4.4.1.2.- Red de funciones de base radial (RBF) reduciendo variables con R.

Esta red también es de naturaleza estática, pero se puede hacer que adopte un comportamiento dinámico permitiendo que las neuronas puedan retornar información.

Con esta red se ha procedido de forma análoga a la que se hizo con su correspondiente red estática. Se analizó reduciendo el conjunto de datos con R y con PCA utilizando como parámetro diferenciador el objetivo o goal.

En la tabla 4.12 podemos ver las redes de funciones de base radial entrenadas.

REDES DE FUNCIÓN DE BASE RADIAL (RBF) DINÁMICAS - REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN R1.			
Variando solo el goal y coeficiente de correlación $R1 < 0.001$ NÚMERO DE VARIABLES = 21			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7e-4	0.0263	0.4365	0.000755591
3e-3	0.0548	0.1703	0.00317649
2e-3	0.0447	0.2075	0.00205307
Variando solo el goal y coeficiente de correlación $R1 > 0.5$ NÚMERO DE VARIABLES = 55			
goal	Residuo entren.	Residuo test	Mse (última iteración)
6.5e-4	0.0255	0.0595	0.000890335
6.5e-5	0.0080	0.0591	7.68674e-05
1e-4	0.0187	0.0591	0.000433794
Variando solo el goal y coeficiente de correlación $R1 > 0.6$ NÚMERO DE VARIABLES = 6			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7e-4	0.0264	0.2193	0.000845274
7.5e-4	0.0274	0.1386	0.000845274
7.8e-4	0.0278	0.2368	0.000845274

Tabla 4.12: Redes RBF dinámicas entrenadas reduciendo variables con R.

Como se puede comprobar en la tabla 4.12, la red que mejor residuo de test ofrece es la red con goal = 6.5e-5 con $R1 > 0,5$. Podemos ver las gráficas de entrenamiento y validación de esta red en las figuras: 4.22a, 4.22b, 4.23a y 4.23b.

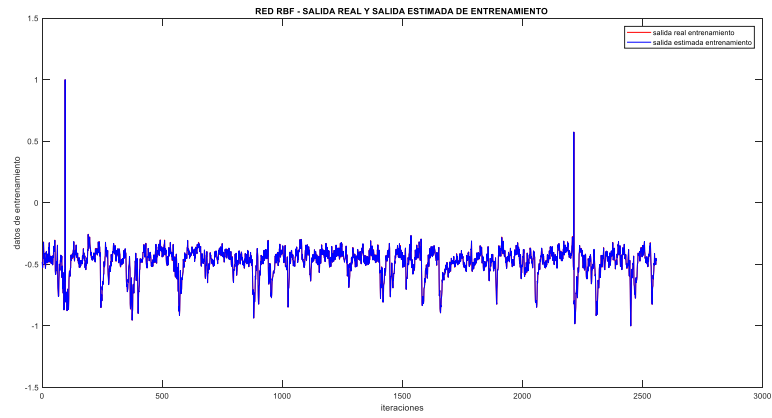


Figura 4.22a: Resultado del entrenamiento de la red dinámica RBF para predicción de DQO reduciendo variables con R.

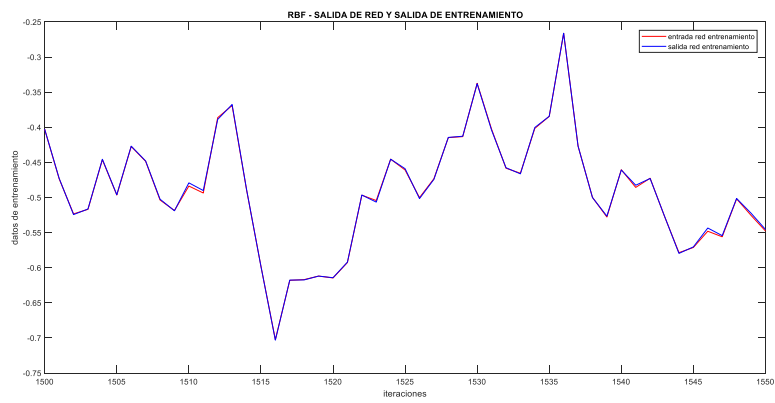


Figura 4.22b: Resultado del entrenamiento de la red dinámica RBF para predicción de DQO reduciendo variables con R.

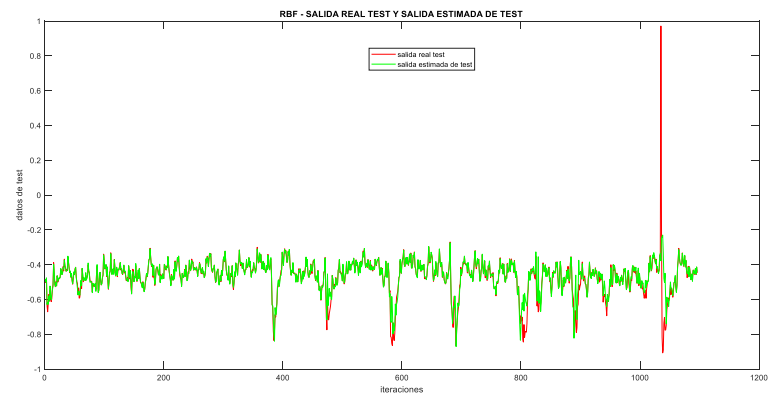


Figura 4.23a: Resultado de la validación de la red dinámica RBF para predicción de DQO reduciendo variables con R.

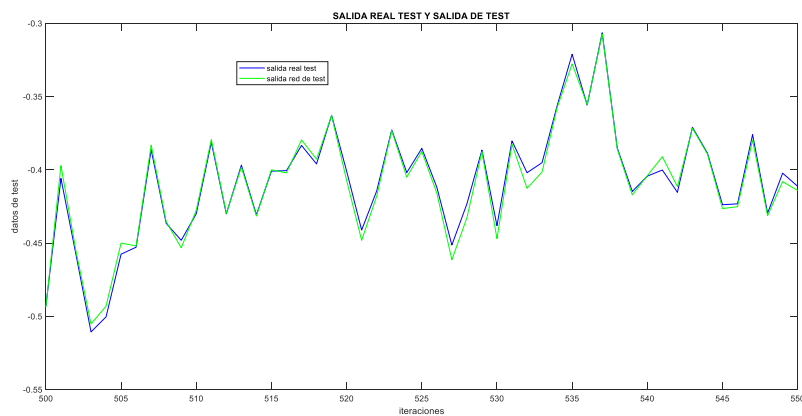


Figura 4.23b: Resultado de la validación de la red dinámica RBF para predicción de DQO reduciendo variables con R.

Visualizando las gráficas, se puede decir que la red ofrece buen resultado de entrenamiento, pero no así en la validación. Esto es así porque en el entrenamiento, la salida determinada por la red se ajusta mejor al patrón de salida en el entrenamiento que en la validación.

Ahora comparamos esta red con su correspondiente red estática. La comparación se hace con las dos redes normalizadas. Esta comparación basada en los residuos que deja cada red, se puede ver en la tabla 4.13.

Red RBF	goal	Residuo entren.	Residuo test
estática	2.5e-6	0.0015	0.0403
dinámica	6.5e-5	0.0080	0.0591

Tabla 4.13: Comparación entre la mejor red RBF estática y la mejor red RBF dinámicas reduciendo variables con R

Una vez más, los residuos que deja la red estática son inferiores a los que deja la red dinámica. Prestando especial atención a los residuos de test, podemos afirmar que la red dinámica RBF es peor que la red estática RBF.

4.4.1.3.- Red anfis reduciendo las variables con R.

Con esta red también haremos lo mismo que hicimos con ella cuando la estudiamos como red estática, o sea, la estudiaremos reduciendo las variables con R y con PCA. El

parámetro a tener en cuenta sigue siendo el número de agrupaciones. Esta red no está normalizada.

La tabla 4.14 recoge todas las redes anfis dinámicas entrenadas utilizando R para reducir el número de variables.

REDES ANFIS DINÁMICAS - REDES ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN R1.		
Variando el número de agrupaciones y coeficiente de correlación $R1 < 0.2$		
NÚMERO DE VARIABLES = 420		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	1.5618	1.5732
10	5.3338	4.4431
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.5$		
NÚMERO DE VARIABLES = 55		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	0.0603	0.8748
50	0.0603	0.6145
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.6$		
NÚMERO DE VARIABLES = 6		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
3	1.3329	1.5600
10	1.3329	1.5600

Tabla 4.14: Redes anfis dinámicas entrenadas reduciendo variables con R.

La mejor red de todas las que recoge la tabla 4.14 es la de 50 agrupaciones, con $R1 > 0,5$.

Los residuos generados tanto para el entrenamiento como para la validación se pueden ver sobre la misma tabla.

Los resultados gráficos de esta red los presentamos en las figuras: 4.24a, 4.24b, 4.24c 4.25a, 4.25b y 4.25c.

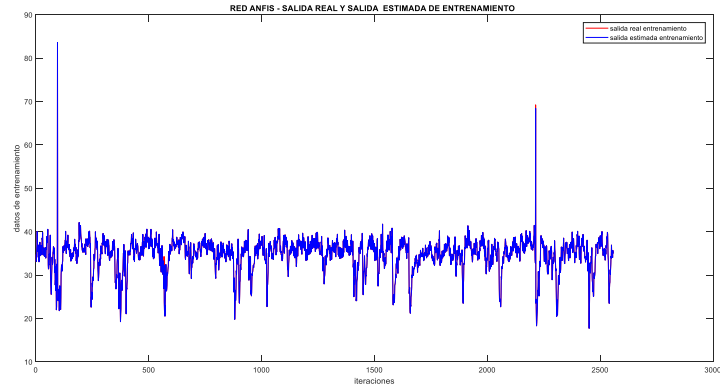


Figura 4.24a: Resultado de entrenamiento de la red dinámica anfis para predicción de DQO reduciendo variables con R.

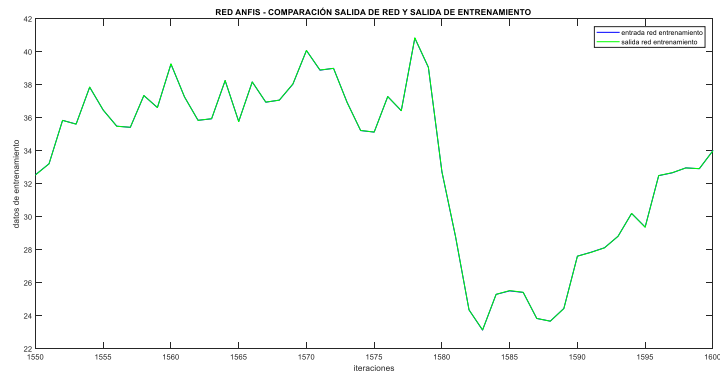


Figura 4.24b: Resultado de entrenamiento de la red dinámica anfis para predicción de DQO reduciendo variables con R.

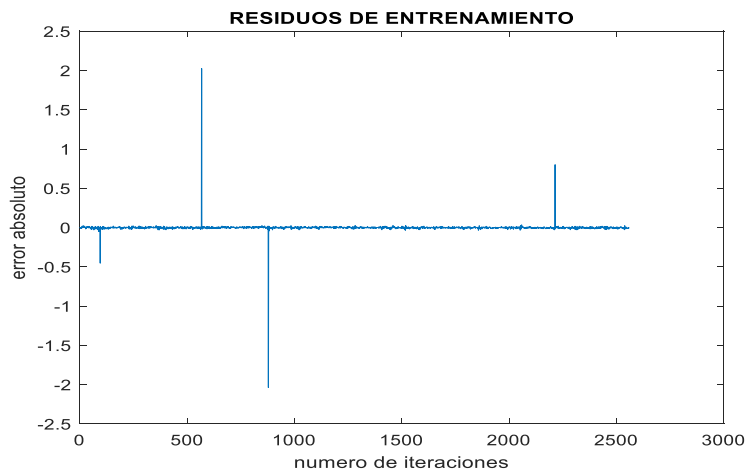


Figura 4.24c: Resultado del residuo de entrenamiento red dinámica anfis para predicción de DQO reduciendo variables con R.

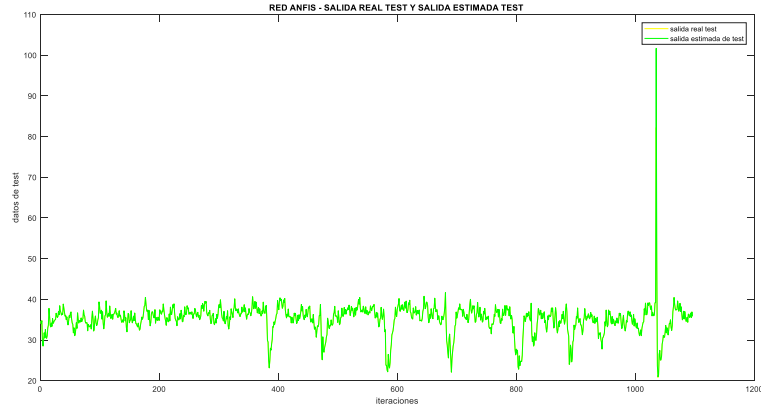


Figura 4.25a: Resultado del test de la red dinámica anfis para predicción de DQO reduciendo variables con R.

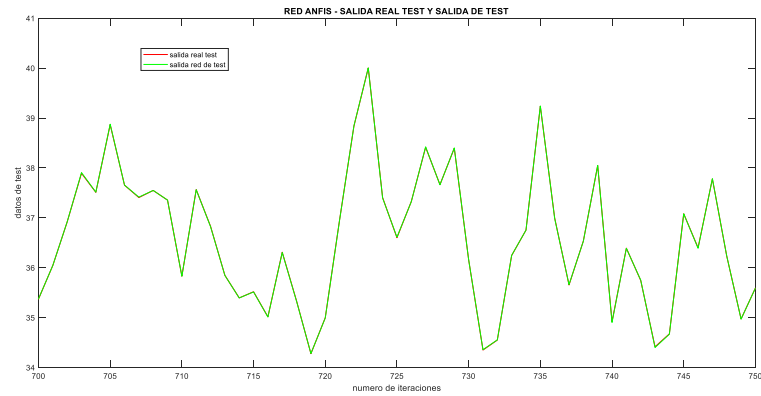


Figura 4.25b: Resultado del test de la red dinámica anfis para predicción de DQO reduciendo variables con R.

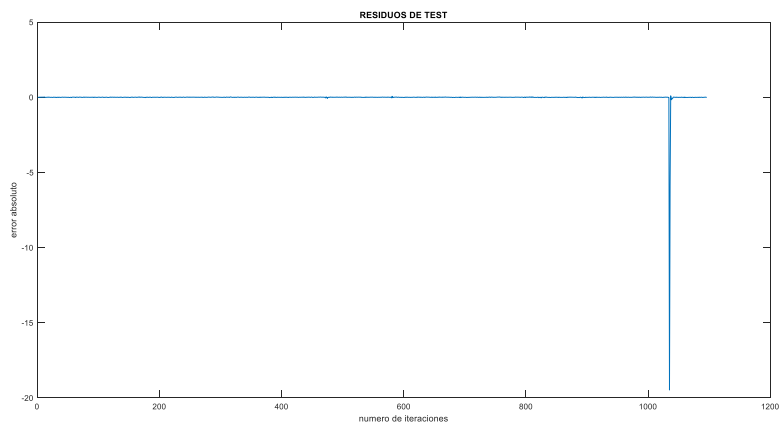


Figura 4.25c: Resultado del residuo de la red dinámica anfis para predicción de DQO reduciendo variables con R.

De las gráficas de esta red se observa que casi hay una sintonía perfecta entre la salida estimada por la red y el patrón de validación o salida deseada tanto en el entrenamiento como en la validación. La conclusión es que el aprendizaje es bueno y la validación también.

En los gráficos de los residuos de entrenamiento y validación, vemos que en ambos casos casi presentan un valor constante, salvo unos pocos picos. Estos picos pueden deberse a valores atípicos en los datos.

La tabla 4.15 hace una comparación entre la red anfis estática obtenida mediante la reducción de variables con R y su correspondiente red dinámica.

Red anfis	N.º de agrupaciones	Residuo entren.	Residuo test
estática	3	0.3183	0.5025
dinámica	50	0.0603	0.6145

Tabla 4.15: Comparación entre la mejor red anfis estática y la mejor red anfis dinámica reduciendo variables con R.

De la tabla 4.15 se deduce que el residuo de entrenamiento de la red estática es peor que su correspondiente red dinámica, sin embargo, la red estática, deja, aunque con muy poca diferencia, un residuo de validación más bajo que su correspondiente red dinámica. Dando mayor importancia al residuo de test, podemos decir que la red estática es ligeramente mejor que la red dinámica.

4.4.1.4.- SVR dinámica reduciendo variables con R.

El estudio de esta regresión en modo dinámico se hace de la misma manera que se hizo con red estática SVM.

Los resultados obtenidos de las diferentes redes entrenadas se pueden ver en la tabla 4.16.

MÁQUINAS DE VECTORES DE SOPORTE (SVM) - REDES DINÁMICAS ENTRENADAS		
REDUCCIÓN DIMENSIONAL SEGÚN R1.		
Variando el tipo de función de ajuste coeficiente de correlación $R1 < 0.2$		
NÚMERO DE VARIABLES = 82		
Kernel	Residuo entrenamiento	Residuo test
Lineal	2.3752	2.1399
gaussiano	2.5937	3.4158
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.3$		
NÚMERO DE VARIABLES = 25		
Kernel	Residuo entrenamiento	Residuo test
Lineal	1.3633	1.5782
gaussiano	2.1777	2.6868
Variando el número de agrupaciones y coeficiente de correlación $R1 > 0.5$		
NÚMERO DE VARIABLES = 7		
Kernel	Residuo entrenamiento	Residuo test
Lineal	0.0620	0.8627
gaussiano	1.2631	1.6100

Tabla 4.16: Redes SVM dinámicas entrenadas reduciendo variables con R.

De todas las SVRs dinámicas que presentamos en la tabla 4.16, la mejor de todas ellas es la del kernel lineal con $R1 > 0.5$ y 7 variables consideradas como entrada en la regresión.

Las figuras 4.26a, 4.26b, 4.27a y 4.27b recogen los resultados gráficos de entrenamiento y validación de esta red.

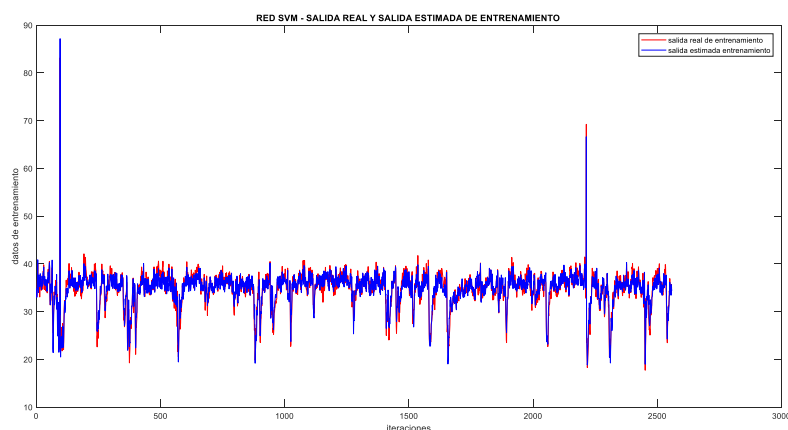


Figura 4.26a: Resultado del entrenamiento de la red dinámica SVM para predicción de DQO reduciendo variables con R.

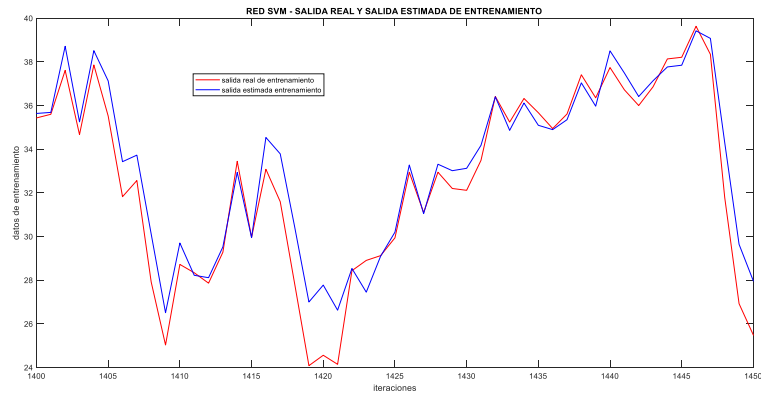


Figura 4.26b: Resultado del entrenamiento de la red dinámica SVM para predicción de DQO reduciendo variables con R.

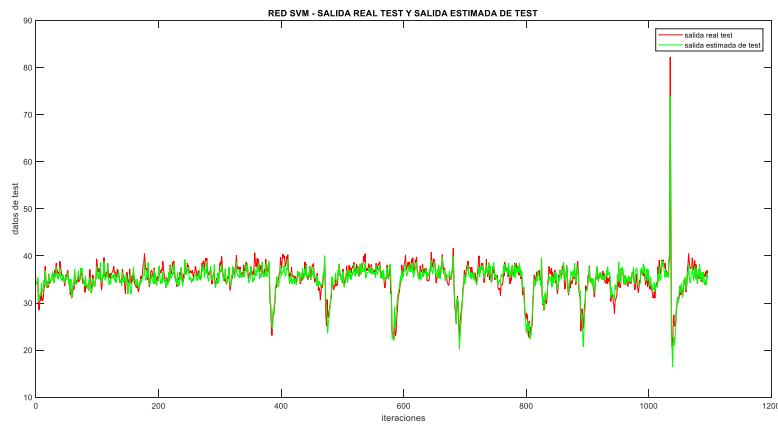


Figura 4.27a: Resultado de la validación de la SVR para predicción de DQO reduciendo variables con R.

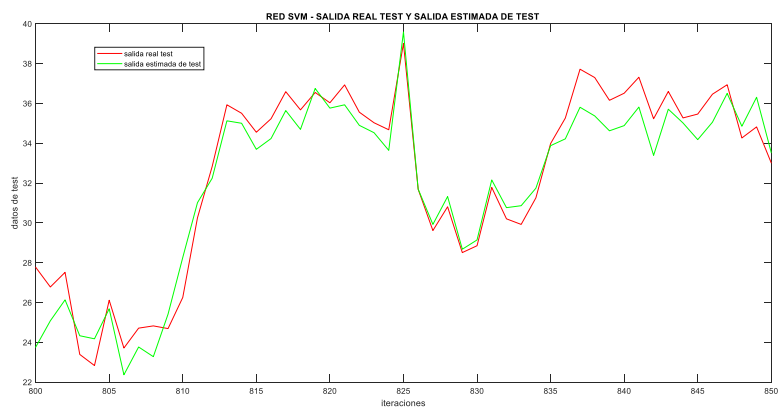


Figura 4.27b: Resultado de la validación de la SVR para predicción de DQO reduciendo variables con R.

De las gráficas de entrenamiento y validación de esta red podemos deducir que entrena igual que valida. No se obtiene un resultado bueno con ella.

Vamos a compararla con la SVM estática. Esta comparación basada en los residuos la presentamos en la tabla 4.17.

Red SVM	Kernel	Residuo entren.	Residuo test
estática	Lineal	0.0620	0.8627
dinámica	Lineal	1.1995	1.2841

Tabla 4.17: Comparación entre la mejor SVM estática y la mejor SVM dinámica reduciendo variables con R.

Teniendo en cuenta los residuos que deja esta red en modo estático y en modo dinámico, es fácil concluir que la SVR estática es mejor que su correspondiente dinámica.

4.4.1.5.- Combinación de las redes dinámicas reduciendo los datos con R.

Ahora vamos a hacer una combinación de las mejores redes dinámicas de la misma forma que lo hicimos con las redes estáticas, luego comparemos los resultados.

- Primero ensamblamos las redes asignándolas el mismo coeficiente o porcentaje a todas tanto para el entrenamiento que con la validación.

$$Y_{combinada_test} = 0.25redff + 0.25redRBF + 0.25redanfis + 0.25SVM \quad (4.8)$$

Los resultados gráficos de esta red se pueden ver en las figuras 4.28a y 4.28b.

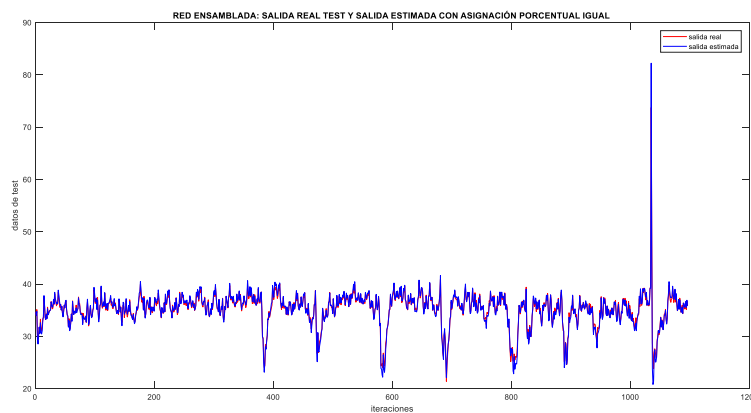


Figura 4.28a: Resultados del test de la red dinámica combinada con factores iguales para predicción de DQO reduciendo variables con R.

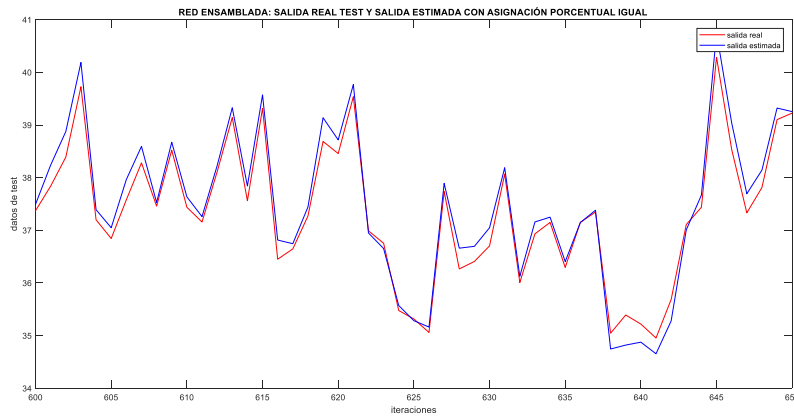


Figura 4.28b: Resultados del test de la red dinámica combinada con factores iguales para predicción de DQO reduciendo variables con R.

Luego comparemos los resultados de esta red con la red obtenida así en modo estático.

- Ahora vamos a ensamblar la misma red, pero buscando la mejor combinación de los factores que nos dé un mejor resultado. Eso se hace con la red para el entrenamiento y con la validación. Las redes que se van a combinar están todas sin normalizar.

$$Y_{combinada_test} = 0.45redff + 0.25redRBF + 0.20redanfis + 0.10SVM \quad (4.9)$$

Las gráficas de la red así obtenida se muestran en las figuras 4.29a y 4.29b, para la validación.

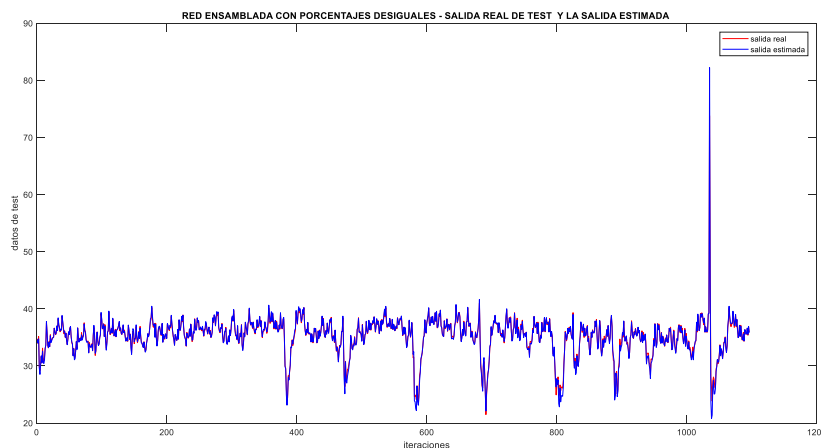


Figura 4.29a: Resultado de la validación del test de las redes dinámicas combinadas con factores diferentes para predicción de DQO reduciendo variables con R.

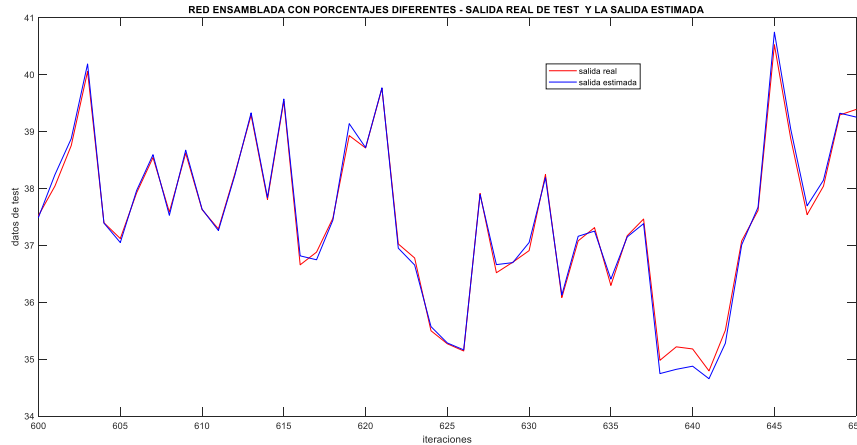


Figura 4.29b: Resultado de la validación del test de las redes dinámicas combinadas con factores diferentes para predicción de DQO reduciendo variables con R.

Una comparación entre estas dos combinaciones basada en las gráficas presentadas, nos permite decir que la combinación con factores diferentes valida relativamente mejor que la de los factores iguales.

De la misma forma que hemos comparado las redes individuales, vamos a comparar los ensambles estáticos con los dinámicos.

Comparación de redes ensambladas estáticas y dinámicas reduciendo las variables con R.		
Ensamble	Residuo entrenamiento	Residuo test
Estático con factores iguales	0.1279	0.3241
Estático con factores diferentes	0.1984	0.3968
Dinámico con factores iguales	0.3104	0.5412
Dinámico con factores diferentes	0.1858	0.4687

Tabla 4.18: Comparación entre ensambles estáticos y dinámicos usando R para reducir variables.

El mejor ensamble lo resaltamos en amarillo sobre la tabla 4.18, una vez más la red estática, esta vez en las redes combinadas, vuelve a resultar la mejor frente a las dinámicas.

Por lo que se ve en la tabla 4.18, las redes estáticas combinadas entrenan mejor y validan también mejor.

4.4.2.- Reducción dimensional utilizando el PCA.

Este método de reducción de la dimensionalidad del conjunto de variables ya se ha explicado en este mismo capítulo y en el capítulo II, así que no vamos a insistir en lo mismo. El PCA se usa en este punto para estudiar las redes que venimos estudiando, pero esta vez en modo dinámico. Estas redes las comparemos con sus correspondientes redes estáticas, entre otras comparaciones necesarias que en puntos precedentes ya hemos hecho.

Presentamos los resultados de los diferentes tipos de redes usando la reducción dimensional basada en el PCA.

4.4.2.1.- Red MLP dinámica reduciendo variables con PCA

La red MLP la hemos entrenado con 30 PC como se puede ver en la tabla 4.19. Se ha comprobado con un número inferior de PC, pero daban peores resultados que con 30, por eso optamos por quedarnos con 30 PC. Como en otros casos con la red MLP el parámetro a tener en cuenta es el número de neuronas, por lo que variando este parámetro obtenemos diferentes redes de este tipo. Todas las redes entrenadas tienen sólo una capa oculta de neuronas. Las redes que se presentan están normalizadas. Se desnormalizarán para las combinaciones y para las comparaciones con otras redes que no están normalizadas.

En la tabla 4.19 presentamos las tres redes entrenadas reduciendo variables con PCA.

REDES MLP DINÁMICAS - REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN PCA. NÚMERO DE COMPONENTES PRINCIPALES (PC) = 30		
Variando número de neuronas y reducción mediante PCA		
N.º neuronas	Residuo entren.	Residuo validación (test)
2	0.0535	0.0554
10	0.0334	0.0592
15	0.0414	0.0571

Tabla 4.19: Redes dinámicas MLP entrenadas reduciendo variables con el PCA

De las tres redes registradas en la tabla 4.19, la red remarcada en amarillo, la de 2 neuronas, es la que mejor resultado presenta, aunque también es cierto que la diferencia no es muy notoria si nos fijamos en los residuos de validación de todas esas redes.

En las figuras 4.30a, 4.30b, 4.31a y 4.31b. presentamos los resultados gráficos de la red señalada como la mejor.

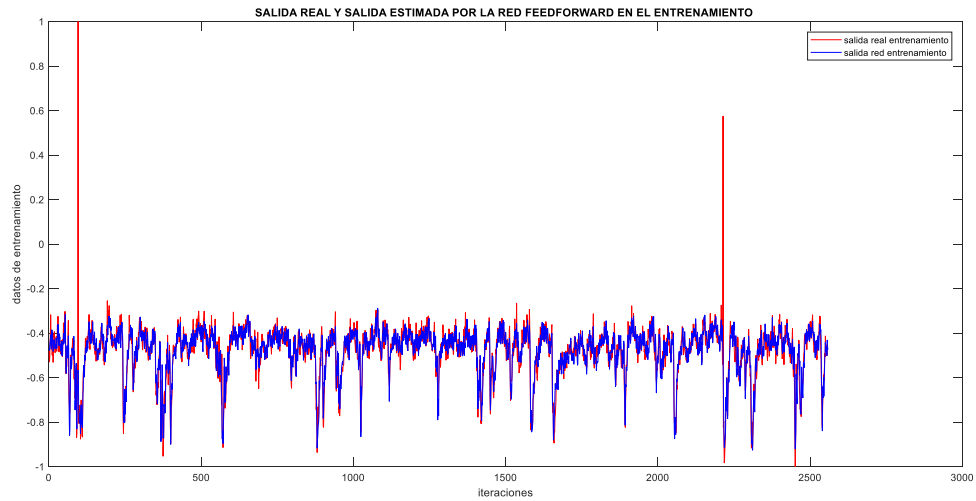


Figura 4.30a: Resultado del entrenamiento de la red dinámica MLP para predicción de DQO reduciendo variables con PCA

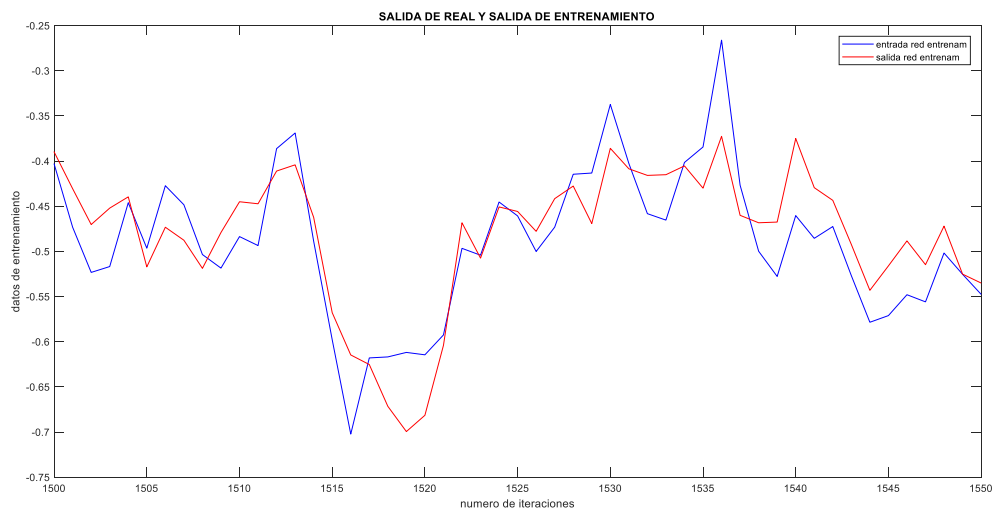


Figura 4.30b: Resultado del entrenamiento de la red dinámica MLP para predicción de DQO reduciendo variables con PCA.

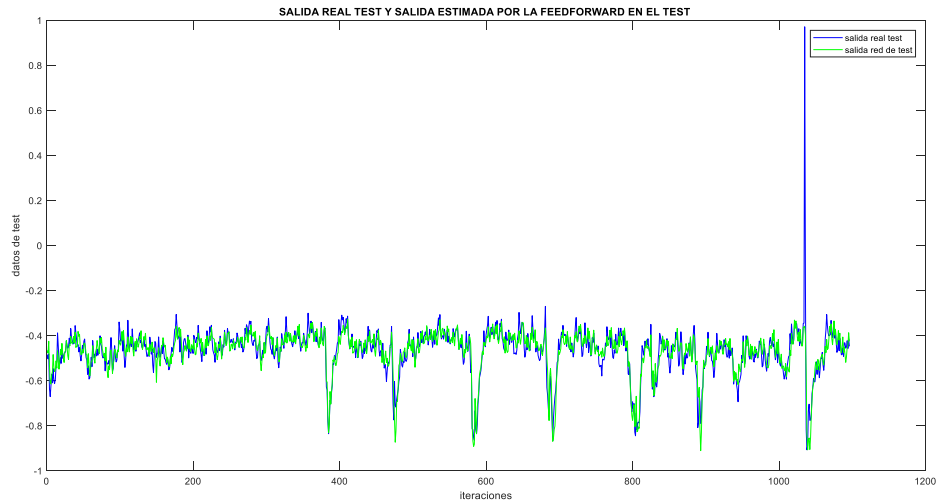


Figura 4.31a: Resultado del test de la red dinámica MLP para predicción de DQO reduciendo variables con PCA

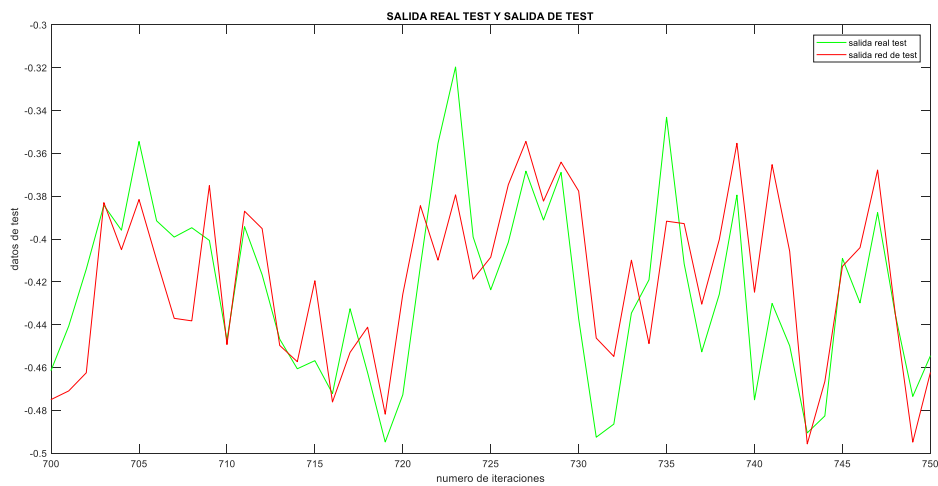


Figura 4.31b: Resultado del test de la red dinámica MLP para predicción de DQO reduciendo variables con PCA

A juzgar por la apariencia gráfica de entrenamiento y validación de esta red, podemos decir que no hemos obtenido un resultado bueno con ella.

Vamos a comparar ahora, usando los residuos, esta red dinámica MLP con su correspondiente red estática, así podemos ver cuál de ellas es mejor. La tabla 4.20 recoge esta comparación.

Red MLP	N.º neuronas	Residuo entren.	Residuo test
estática	15	0.0020	0.0159
dinámica	2	0.0535	0.0554

Tabla 4.20: Comparación entre la mejor red MLP estática y la mejor red MLP dinámica reduciendo variables con PCA

Como se puede ver en la tabla 4.20, la red estática MLP es mejor que su correspondiente red dinámica tanto en el entrenamiento como en la validación.

4.4.2.2.- Red RBF dinámica reduciendo variables con PCA

Las redes de funciones de base radial ya fueron explicadas en el capítulo II con más detalle, y también en este capítulo. Como en los demás casos con la red de funciones de base radial, el parámetro diferenciador con el que entrenamos varias redes es el objetivo o goal. Esta red también está normalizada. Se desnormalizará para el ensamble y para la comparación con otras redes que no están normalizadas.

Las diferentes redes entrenadas de RBF se pueden ver en la tabla 4.21.

REDES DE FUNCIÓN DE BASE RADIAL (RBF) DINÁMICAS - REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN PCA. NÚMERO DE PC = 20			
Variando solo el goal y reducción mediante PCA			
goal	Residuo entren.	Residuo test	Mse (última iteración)
7e-4	0.0265	0.0783	0.000714275
4e-4	0.0200	0.0798	0.000424079
7.5e-4	0.0274	0.0779	0.000777072

Tabla 4.21: Redes dinámicas RBF entrenadas reduciendo variables con el PCA

Como se puede ver en esta tabla, el resultado de estas redes es muy parecido, pero nos quedamos con la remarcada en amarillo porque su valor de test es ligeramente inferior al resto. Será considerada como la mejor red, que es la que después utilizaremos para el ensamble. Se muestra en la última columna el error cuadrático medio (MSE) alcanzado por la red al darle en cada caso los objetivos o goal de la primera columna.

Los resultados de esta red los presentamos en gráficas de las figuras 4.32a, 4.32b, 4.33a y 4.33b.

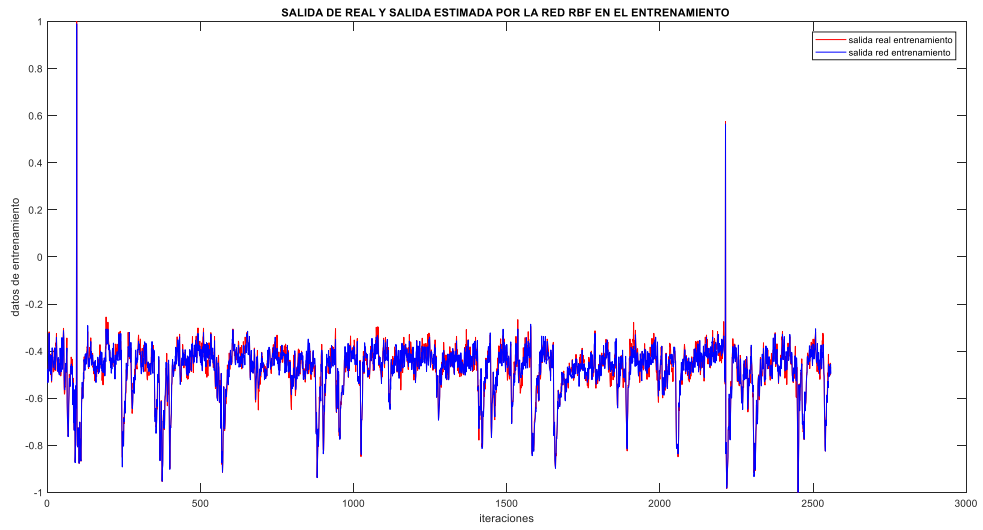


Figura 4.32a: Resultado del entrenamiento de la red dinámica RBF para predicción de DQO reduciendo variables con PCA

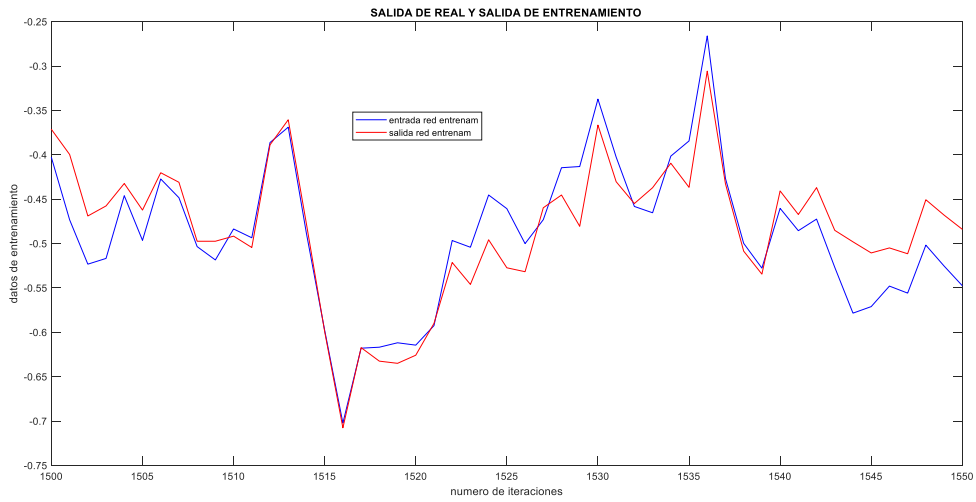


Figura 4.32b: Resultado del entrenamiento de la red dinámica RBF para predicción de DQO reduciendo variables con PCA

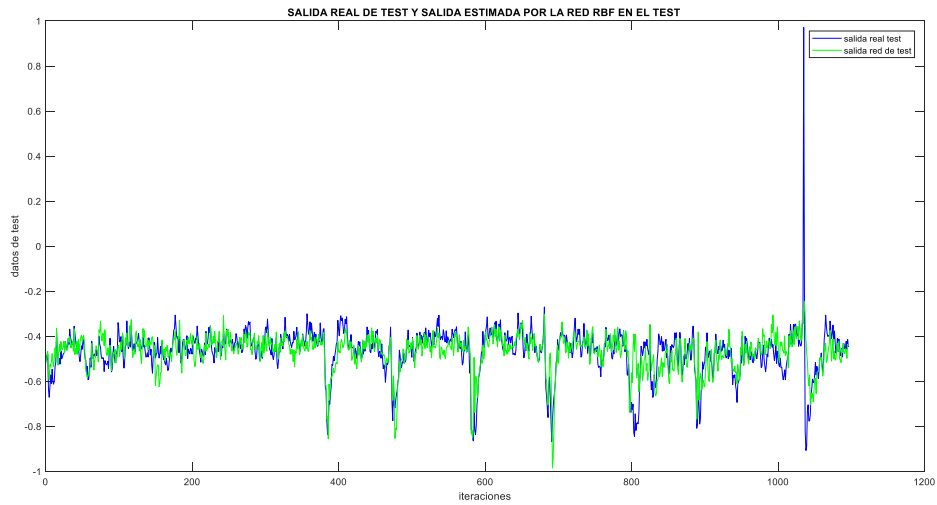
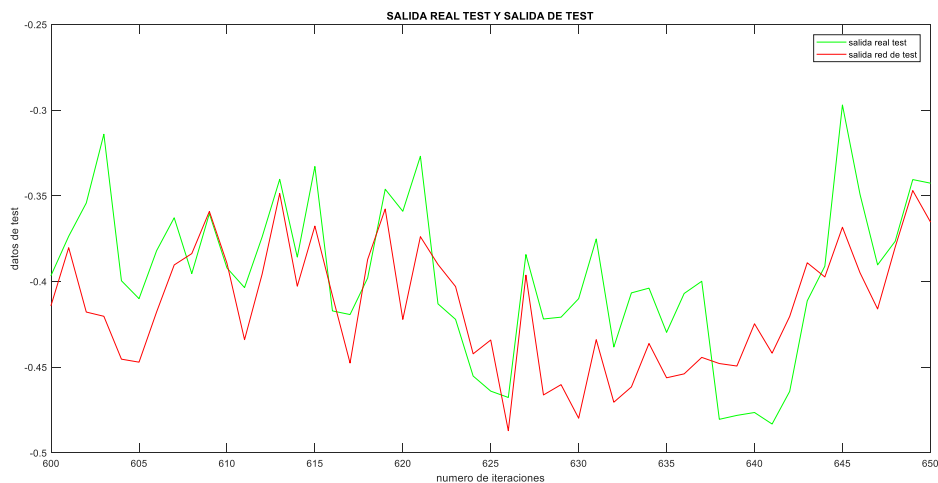


Figura 4.33a: Resultado del test de la red dinámica RBF para predicción de DQO reduciendo variables con PCA



4.33b: Resultado del test de la red dinámica RBF para predicción de DQO reduciendo variables con PCA

La observación de las gráficas resultantes de esta red sugiere que la red entrena mejor y valida peor. El resultado de esta red, a la espera de la comparación que hagamos con su correspondiente red dinámica, no es nada bueno.

Comparamos esta red con la red RBF estática. En la tabla 4.22 presentamos esta comparación utilizando los residuos.

Red RBF	goal	Residuo entren.	Residuo test
estática	6e-5	0.0077	0.0408
dinámica	7.5e-4	0.0274	0.0779

Tabla 4.22: Comparación entre la mejor red RBF estática y la mejor red RBF dinámica reduciendo variables con PCA.

En este caso, la mejor red RBF estática entrenada es mejor que la mejor red dinámica si nos fijamos en los residuos de entrenamiento, y también valida mejor. Como el objetivo es que la red valide bien para poder generalizar bien, podemos decir que la mejor red estática RBF es mejor que la mejor red dinámica.

4.4.2.3.- Red ANFIS dinámica reduciendo variables con PCA

Como las otras redes, la red neuro-difusa anfis ya fue explicada en el capítulo II al detalle y en este capítulo IV ya dijimos cómo se usaría. El número de agrupaciones sigue siendo el parámetro distintivo. De las muchas redes entrenadas variando este parámetro, nos quedamos con dos, las cuales presentamos en la tabla 4.23. Esta red no está normalizada.

REDES ANFIS DINÁMICAS- REDES ENTRENADAS REDUCCIÓN DIMENSIONAL SEGÚN PCA. NÚMERO DE PC = 20		
Variando el número de agrupaciones y reducción mediante PCA		
N.º de agrupaciones	Residuo entrenamiento	Residuo test
10	1.8901	1.8391
100	1.8901	1.8391

Tabla 4.23: Redes dinámicas anfis entrenadas reduciendo variables con el PCA

Los datos de las dos redes reflejadas en la tabla 4.23 indican que las dos redes son iguales tanto en el entrenamiento como en el test. Podríamos coger cualquiera de ellas, pero hemos querido basarnos en algún criterio, por lo que, por la carga computacional que supone simular cada una de las redes, marcamos la red de 10 agrupaciones como la mejor. Esta red tarda menos en simularse. Los resultados gráficos de esta red vienen recogidos en las figuras 4.34a, 4.34b, 4.35a y 4.35b.

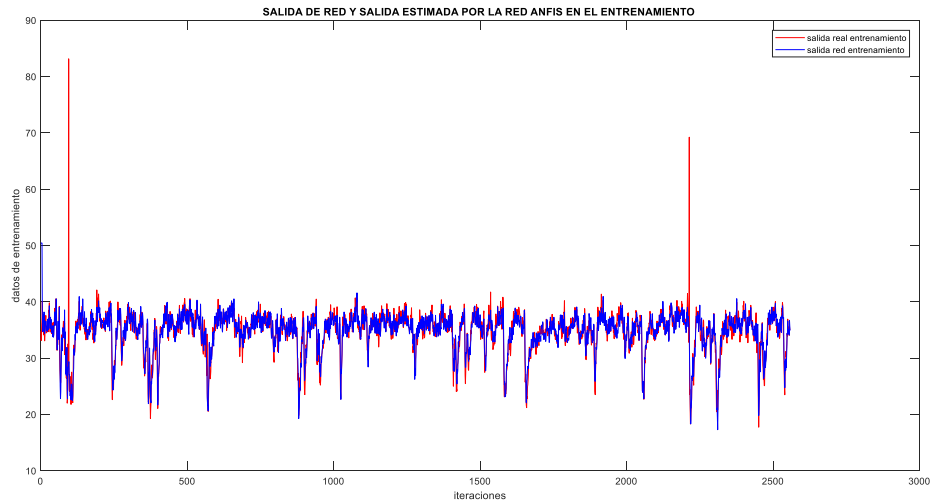


Figura 4.34a: Resultado del entrenamiento de la red dinámica anfis para predicción de DQO reduciendo variables con PCA

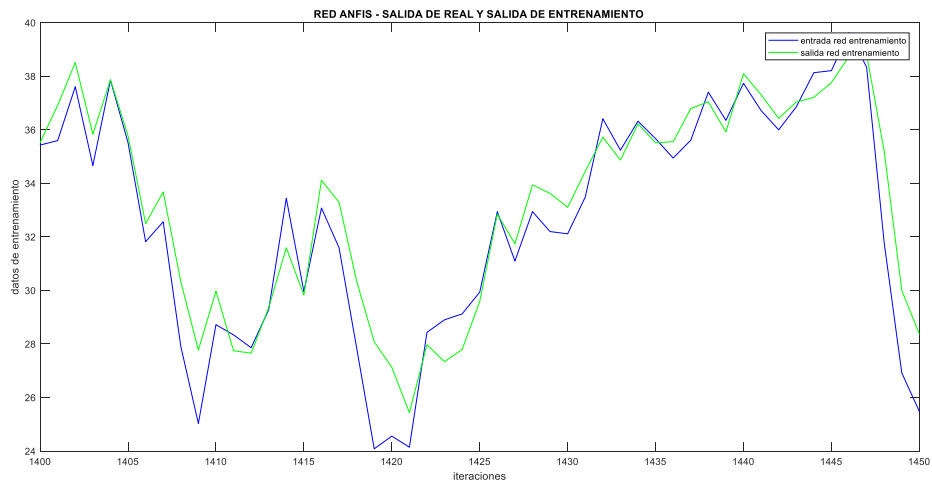


Figura 4.34b: Resultado del entrenamiento de la red dinámica anfis para predicción de DQO reduciendo variables con PCA

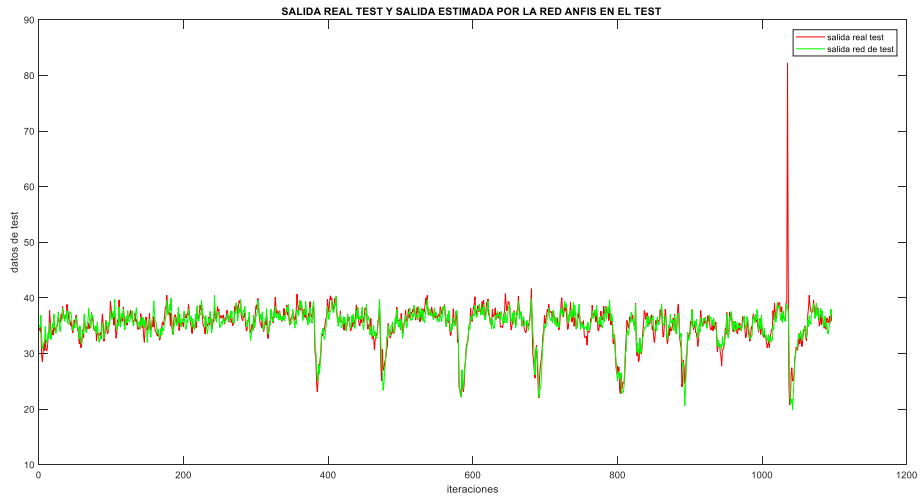


Figura 4.35a: Resultado del test de la red dinámica anfis para predicción de DQO reduciendo variables con PCA

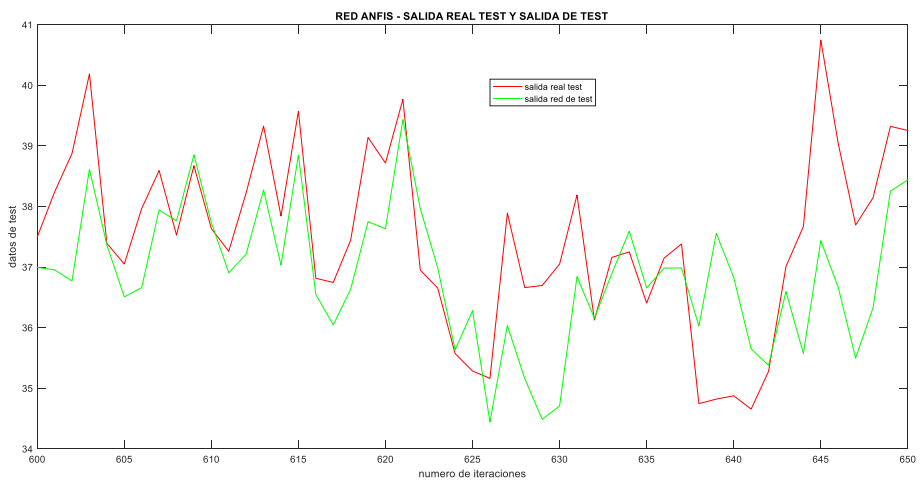


Figura 4.35b: Resultado del test de la red dinámica anfis para predicción de DQO reduciendo variables con PCA

La única lectura positiva de esta red es que la validación mejora ligeramente el resultado de entrenamiento, pero por lo demás, el resultado no es bueno.

Lo comparamos con la correspondiente red estática.

Red anfis	N.º agrupaciones	Residuo entren.	Residuo test
estática	100	1.0077	1.0888
dinámica	10	1.8901	1.8391

Tabla 4.24: Comparación entre la mejor red anfis estática y la mejor red anfis dinámica reduciendo variables con PCA.

Basándonos en la tabla 4.24, podemos decir que la mejor red estática anfis ofrece mejor resultado que la mejor red dinámica anfis; su residuo de entrenamiento es mejor, también lo es el de validación.

4.4.2.4.- SVR dinámica reduciendo variables con PCA

Las SVRs ya fueron explicadas anteriormente. Como en otros casos, probamos los resultados que obtenemos con la SVR en modo dinámico. Como en los demás casos anteriores con la SVR, el parámetro a tener en cuenta es tipo de kernel, que como siempre, serán de dos tipos, el kernel lineal y el gaussiano. Esta red no está normalizada.

Los resultados de las redes entrenadas se muestran en la tabla 4.25.

MÁQUINAS DE SOPORTE VIRTUAL DINÁMICAS(SVM) - REDES ENTRENADAS NÚMERO DE PC= 30		
Variando el tipo de función de ajuste y reducción mediante PCA		
Kernel	Residuo entrenamiento	Residuo test
Lineal	1.7856	1.8349
gaussiano	2.5852	3.4010

Tabla 4.25: SVRs dinámicas entrenadas reduciendo variables con el PCA

Por su residuo de entrenamiento y de test, la mejor SVR entre estas dos de la tabla 4.25 la resaltamos en amarillo. Los resultados gráficos de dicha SVR los mostramos en las figuras 4.36a, 4.36b, 4.37a y 4.37b.

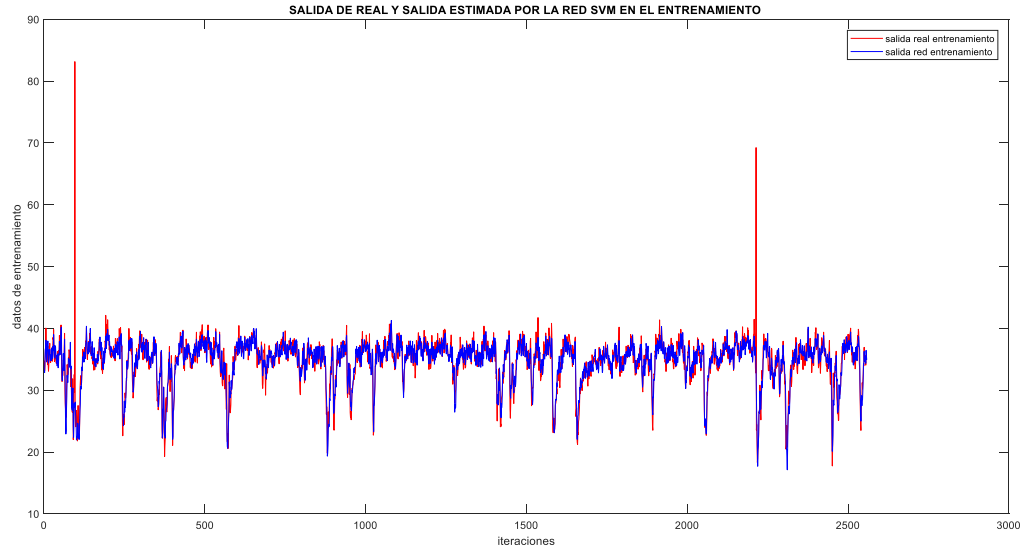


Figura 4.36a: Resultado del entrenamiento de la SVR dinámica para predicción de DQO reduciendo variables con PCA

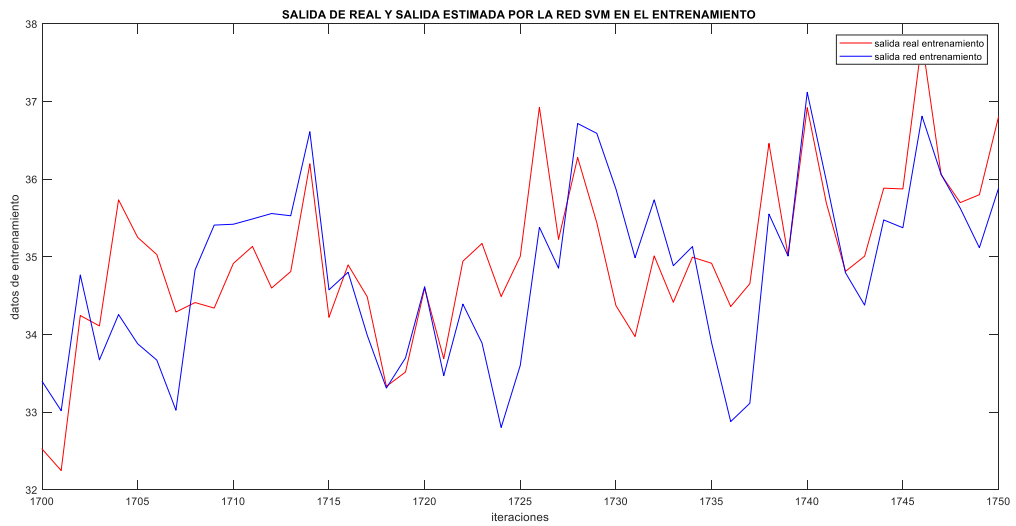


Figura 4.36b: Resultado del entrenamiento de la SVR dinámica para predicción de DQO reduciendo variables con PCA

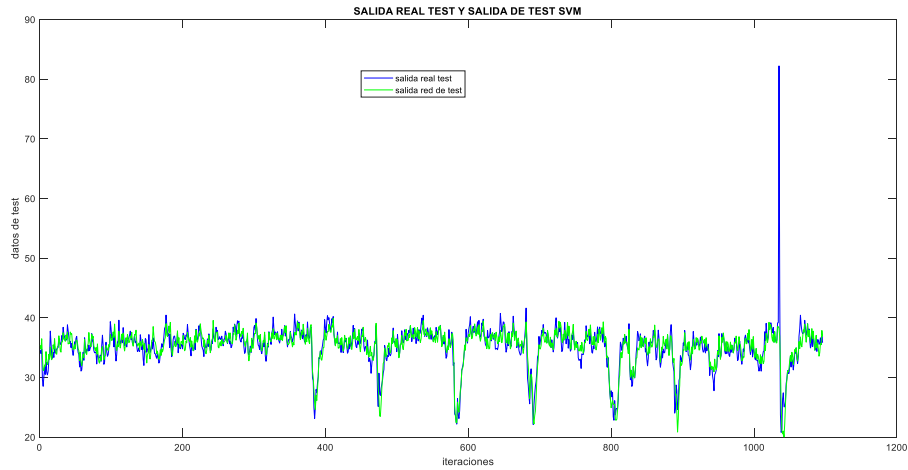


Figura 4.37a: Resultado del test de la SVM dinámica para predicción de DQO reduciendo variables con PCA.

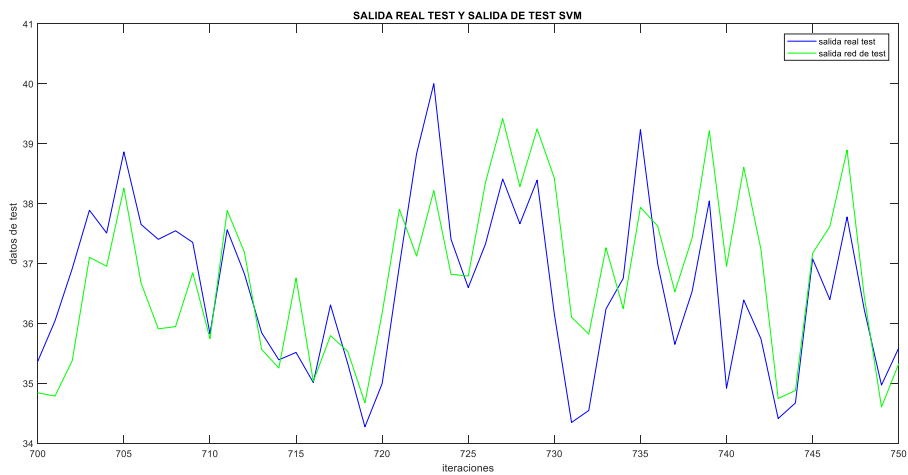


Figura 4.37b: Resultado del test de la SVR dinámica para predicción de DQO reduciendo variables con PCA.

Esta SVR, por lo observado en las gráficas y viendo el comportamiento de sus residuos, entrena mal y valida peor. Con esta red no obtenemos buen resultado.

Como hemos hecho con los demás casos, la SVM la comparamos con su correspondiente estática. Los resultados de esta comparación se recogen en la tabla 4.26.

Red anfis	Tipo de kernel	Residuo entren.	Residuo test
estática	Lineal	1.3110	1.5114
dinámica	Lineal	1.7856	1.8349

Tabla 4.26: Comparación entre la mejor SVM estática y la mejor SVM dinámica reduciendo variables con PCA

Todas las mejores SVR, según la tabla 4.26, se obtienen utilizando un kernel lineal.

Por los residuos de entrenamiento y validación, la mejor SVR vuelve a ser una vez más la SVR estática.

4.4.2.5.- Combinación de redes dinámica reduciendo variables con PCA.

Una vez estudiado todas las redes desarrolladas reduciendo las variables mediante la aplicación de la técnica del PCA, ahora procedemos a ensamblarlas. Para eso, como siempre en esos casos a lo largo de este trabajo, consideramos dos casos: el del mismo factor para todas las redes y el de la asignación de diferentes factores a las redes. Esto se hace como en otros casos, para el test y para la validación. Las dos redes obtenidas por combinación se compararán con sus correspondientes redes estáticas. Todas las combinaciones se hacen con todas las redes sin normalizar.

$$Y_{combinada_test} = 0.25redff + 0.25redRBF + 0.25redanfis + 0.25SVM \quad (4.10)$$

- Las gráficas de test de este primer tipo de ensamble se pueden ver en las figuras 4.38a y 4.38b, para validación.

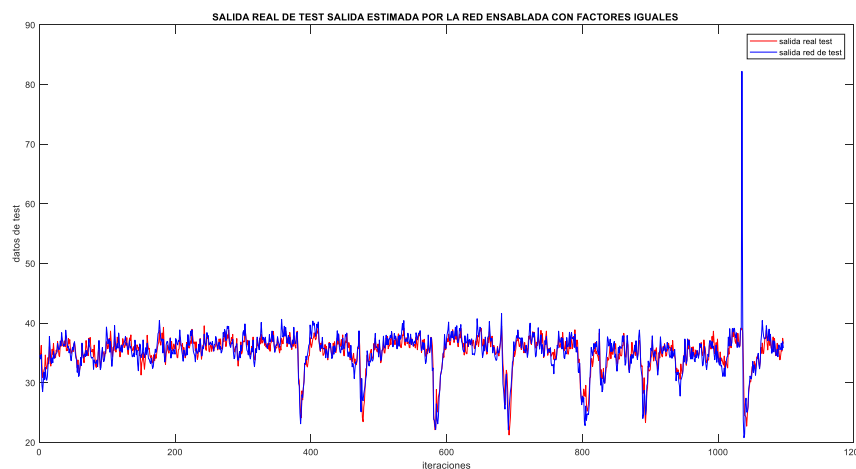


Figura 4.38a: Resultado del test de la red dinámica ensamblada con factores iguales para predicción de DQO reduciendo variables con PCA

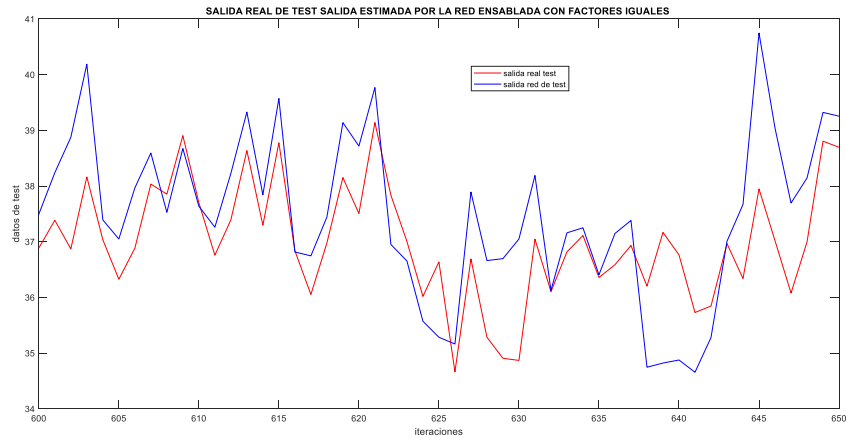


Figura 4.38b: Resultado del test de la red dinámica ensamblada con factores iguales para predicción de DQO reduciendo variables con PCA

La impresión, viendo las gráficas de esta red ensamblada con factores iguales, es que no aprende de bien, como tampoco valida bien. Con esta red el resultado es malo.

- El segundo método de ensamble al que hemos optado es el de encontrar la mejor combinación de los factores disponibles para intentar dar con la mejor solución posible. Según esto tenemos:

$$Y_{combinada_test} = 0.60redff + 0.10redRBF + 0.15redanfis + 0.15SVM \quad (4.11)$$

Las gráficas de esta red combinada se pueden ver en las figuras 4.39a y 4.39b, para la validación.

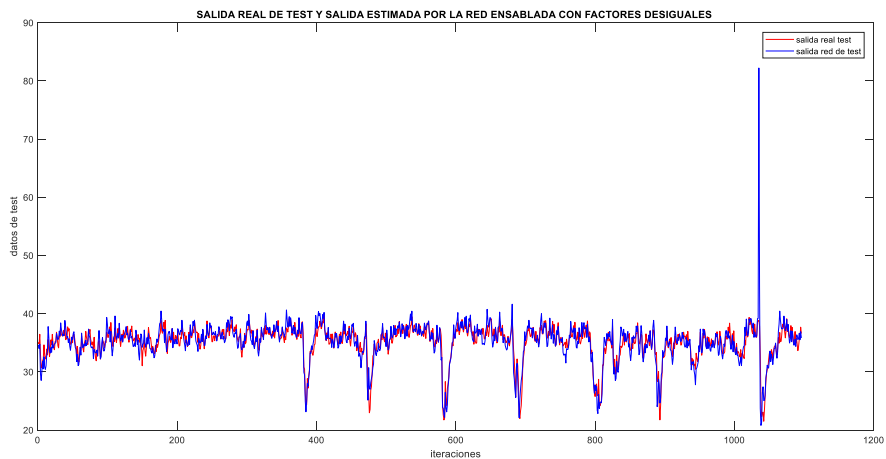


Figura 4.39a: Resultado del test de la red dinámica ensamblada con factores diferentes para predicción de DQO reduciendo variables con PCA

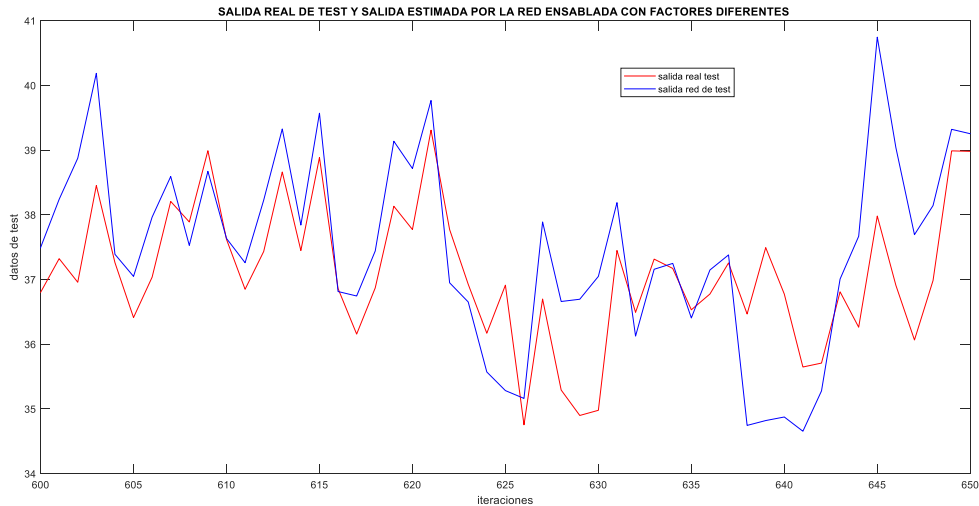


Figura 4.39b: Resultado del test de la red dinámica ensamblada con factores diferentes para predicción de DQO reduciendo variables con PCA

Esta red, como se puede observar en las gráficas, tampoco ofrece buenos resultados ni en el entrenamiento ni en la validación.

Ahora procedemos a comparar los ensambles de las redes dinámicas con los correspondientes ensambles estáticos reduciendo el conjunto de variables con PCA.

Comparación de redes ensambladas estáticas y dinámicas reduciendo las variables con PCA.		
Ensamble	Residuo entrenamiento	Residuo test
Estática con factores iguales	0.5489	1.0223
Estática con factores diferentes	0.4652	0.8652
Dinámica con factores iguales	1.3874	1.8200
Dinámica con factores diferentes	1.3874	1.7894

Tabla 4.27: Comparación entre ensambles de redes estáticas y dinámicas usando PCA para reducir variables.

Por lo que se ve en tabla 4.27, los dos métodos de ensamble tienen el mismo residuo de entrenamiento para las redes dinámicas; por otro lado, las redes que mejor resultado presentan en el ensamble son las redes de ensamble estático, siendo la mejor de todas, la red de ensamble estático con factores diferentes.

4.4.3.- Comparación de las redes dinámicas.

En este punto presentamos todas las mejores redes dinámicas y la combinación de éstas en cada caso, sus correspondientes resultados numéricos en residuos de entrenamiento y validación. Los resultados gráficos de las redes contenidas en la tabla 4.28 ya fueron expuestos anteriormente.

REDES DINÁMICAS CON DATOS REDUCIDOS APLICANDO R		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (10 neuronas) y $R1 < 0.2$	0.3126	0.3011
Funciones de base radial (goal = $6.5e-5$) y $R1 > 0.5$	0.0516	2.0063
Anfis (50 agrupaciones) y $R1 > 0.5$	0.0603	0.6145
SVM (kernel lineal) y $R1 < 0.2$	1.1995	1.2841
Ensamble (combinación) con factores iguales	0.3104	0.5412
Ensamble (combinación) con factores diferentes	0.1858	0.4687
REDES DINÁMICAS CON DATOS REDUCIDOS MEDIANTE PCA		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (2 neuronas) y 30 PC	1.6988	1.8045
Funciones de base radial (goal = $7.5e-4$) y 20 PC	0.8952	2.5463
Anfis (10 agrupaciones) 20 PC	1.8901	1.8391
SVM (kernel lineal) 30 PC	1.7856	1.8349
Ensamble (combinación) con factores iguales	1.3874	1.8200
Ensamble (combinación) con factores diferentes	1.3874	1.7894

Tabla 4.28: Comparación de las mejores redes dinámicas

Como se puede ver en la tabla 4.28, la mejor red entre todas las redes dinámicas es la red MLP de 10 neuronas con $R1 < 0.2$. esta red presenta mejor residuo de validación que el resto de las redes. Esta red es con diferencia mejor que la mejor que se obtiene reduciendo las variables mediante el PCA.

Las dos mejores redes las resaltamos en amarillo.

Otra conclusión que de esta tabla se puede sacar es que las redes dinámicas, si ensambladas, en general dan mejor resultado que cada red de forma individual; eso es así tanto reduciendo las variables con el PCA que, con el coeficiente de correlación, cuestión que, por otro lado, coincide con la teoría.

Como se puede ver en la tabla 4.28, si comparamos las redes dinámicas ensambladas, el mejor resultado se obtiene con las redes ensambladas reduciendo las variables con R.

4.5.- COMPARACIÓN DE LAS MEJORES REDES ESTÁTICAS Y DINÁMICAS.

En este punto comparamos, a través de la tabla 4.29, las mejores redes estáticas y las mejores redes dinámicas para los dos criterios de reducción dimensional aplicados. En esta tabla recogemos todas las mejores redes de cada tipo para cada caso.

La tabla 4.29 nos lleva a la tabla 4.30, que es un resumen de ésta. En esta última, nos quedamos sólo con la mejor red resultante de cada método de reducción dimensional aplicado, para finalmente elegir la mejor de las redes sobre ésta.

Las mejores redes quedan remarcadas en amarillo sobre la tabla 4.29, mientras que la mejor red obtenida de todas, la remarcamos en rojo en la tabla 4.30.

REDES ESTÁTICAS		
REDES ESTÁTICAS CON DATOS REDUCIDOS MEDIANTE COEFICIENTE DE CORRELACIÓN (R)		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (10neuronas) R1<0.2	0.2929	0.3631
Funciones de base radial (goal = 2.5e-6) R1>0.5	0.0506	1.3187
Anfis (10 agrupaciones) R1<0.2	0.3183	0.5025
SVM (kernel lineal) R1>0.5	0.0620	0.8627
Ensamble (combinación) con factores iguales	0.1279	0.3241
Ensamble (combinación) con factores diferentes	0.1984	0.3968
REDES ESTÁTICAS CON DATOS REDUCIDOS MEDIANTE PCA		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (15 neuronas) 30 PC	0.0979	0.6654
Funciones de base radial (goal = 6e-5) 20 PC	0.2525	1.3348
Anfis (100 agrupaciones) 20 PC	1.0077	1.0888
SVM (kernel lineal) 30 PC	1.3110	1.5114
Ensamble (combinación) con factores iguales	0.5489	1.0223
Ensamble (combinación) con factores diferentes	0.4652	0.8652
REDES DINÁMICAS		
REDES DINÁMICAS CON DATOS REDUCIDOS APLICANDO R		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (10 neuronas) y R1<0.2	0.3126	0.3011
Funciones de base radial (goal = 6.5e-5) y R1>0.5	0.0516	2.0063
Anfis (50 agrupaciones) y R1>0.5	0.0603	0.6145
SVM (kernel lineal) y R1<0.2	1.1995	1.2841
Ensamble (combinación) con factores iguales	0.3104	0.5412
Ensamble (combinación) con factores diferentes	0.1858	0.4687
REDES DINÁMICAS CON DATOS REDUCIDOS MEDIANTE PCA		
Red	Residuo entrenamiento	Residuo de validación (test)
MLP (2 neuronas) y 30 PC	1.6988	1.8045
Funciones de base radial (goal = 7.5e-4) y 20 PC	0.8952	2.5463
Anfis (10 agrupaciones) 20 PC	1.8901	1.8391
SVM (kernel lineal) 30 PC	1.7856	1.8349
Ensamble (combinación) con factores iguales	1.3874	1.8200
Ensamble (combinación) con factores diferentes	1.3874	1.7894

Tabla 4.29: Mejores redes estáticas y dinámicas

Redes estáticas			
Red	Método de reducción dimensional	Residuo entrenamiento	Residuo de validación
Ensamble(combina- ción) con factores iguales	R	0.1279	0.3241
MLP (15 (neuronas) 30 PC	PCA	0.0979	0.6654
Redes dinámicas			
Red	Método de reducción dimensional	Residuo entrenamiento	Residuo de validación
MLP (10 neuronas) y R1<0.2	R	0.3126	0.3011
Ensamble(combina- ción) con factores diferentes	PCA	1.3874	1.7894

Tabla 4.30: Mejores redes estáticas y dinámicas

Sobre la tabla 4.30 se puede ver que la mejor red de todas es la MLP dinámica de 10 neuronas con $R1 < 0.2$. Con ella se obtiene un residuo de validación más pequeño. Esta red se remarca en rojo.

Otra deducción que se puede ver sobre esta tabla es que, al hacer así la comparación entre las redes estáticas y dinámicas, las mejores redes se obtienen con las combinaciones y las redes MLP.

CAPÍTULO V

CONCLUSIONES Y TRABAJO FUTURO

CAPÍTULO V: CONCLUSIONES Y TRABAJO FUTURO

5.1- CONCLUSIONES

Como se dijo, el objetivo de este proyecto era diseñar, utilizando redes neuronales MLP, RBF y la red anfis, y una SVM, un sensor software que fuera capaz de predecir la demanda química de oxígeno en una estación depuradora de aguas residuales a partir de la información generada por los sensores hardware sobre la línea de proceso. Para lograr este objetivo, se utilizó dos métodos de extracción de datos: el coeficiente de correlación y el análisis de componentes principales. Las redes se han estudiado tanto en modo estático como en modo dinámico. Se ensamblaron las mejores redes según el método de extracción utilizado y atendiendo a si se trataba de redes estáticas o dinámicas, para luego hacer las comparaciones necesarias como se puede ver a lo largo del presente documento. A partir de los resultados obtenidos, y habiendo estudiado el marco teórico del proyecto, llegamos a las siguientes conclusiones:

Primero: de los tres tipos de redes desarrolladas y las SVR, al comparar un determinado tipo de red estática con su correspondiente dinámica, en general, las estáticas ofrecen mejores resultados que las dinámicas. Esto puede deberse a que los datos de la planta se tomaron en lazo cerrado entorno a un punto estacionario, y entorno a ese punto no hay mucha variación, por lo que el sistema se puede considerar en esas circunstancias lineal y estático.

Segundo: el criterio de reducción dimensional con el que se obtiene mejores resultados tanto para las redes estáticas como para las dinámicas, es el basado en el coeficiente de correlación (R).

Tercero: Como se puede ver en la tabla 4.29 y 4.30, al comparar las mejores redes estáticas con las mejores redes dinámicas, la red con la que se obtuvo mejor resultado es la red MLP dinámica. Sólo las MLP mejoran el resultado de las redes ensambladas.

Cuarto: se considera que se ha conseguido el objetivo para el que se ha desarrollado este trabajo fin de grado, el cual era diseñar un sensor software capaz de medir a tiempo real, la demanda química de oxígeno (DQO), y que, por tanto, cualquiera de los cuatro sensores señalados en la tabla 4.30, es apto para este fin, siendo especialmente aconsejable, el de MLP de 10 neuronas con $R1 < 0.2$, es decir, eligiendo las variables del proceso menos correlacionadas con la salida.

5.2.- TRABAJO FUTURO.

Sobre esta línea de trabajo se presentan varias alternativas a este, los cuales se pueden desarrollar y comparar los resultados obtenidos con este trabajo si se desea, Entre las cuales proponemos:

Utilizar las mismas redes neuronales artificiales y una SVR en modo estático y dinámico como en este trabajo se ha hecho, pero usando otros métodos de extracción de datos como pueden ser: el análisis de componentes principales con kernel o análisis de componentes principales probabilístico.

Utilizar métodos de regresión lineal y compararlos con los de regresión no-lineal para ver si los mejoran.

Desarrollar por uno lado, una o dos redes neuronales artificiales recurrentes o con conexiones hacia atrás como pueden ser: ART, las BAM y las Cogniton, y por otro, las mismas redes que se han desarrollado en este trabajo en modo estático o dinámico y comparar los resultados.

Desarrollar un sensor software basado en el conocimiento.

CAPÍTULO VI

BIBLIOGRAFÍA

CAPÍTULO VI. BIBLIOGRAFÍA

Libros y artículos consultados

1. Acha, V.; Meurens, M.; Naveau, H.; Dochain, D.; Bastin, G. & Agathos, S. N. "Model-based estimation of an anaerobic reductive dechlorination process via an attenuated total reflection-fourier transform infrared sensor," *Water Sci. Technol.*, vol. 40, No. 8, pp. 33–40, 1999.
2. Alex, J.; Benedetti, L.; Copp, J.; Gernaey, K.V.; Jeppsson, U.; Nopens, I.; Pons, M.N.; Rosen, C.; Steyer, J.P. & Vanrolleghem, P. (2018) "Benchmark Simulation Model nº 2 (BSM2)", *Water Science & Technology*.
3. Apuntes de tecnología ambiental y de procesos para el grado en electrónica industrial y automática (2011). Departamento de ingeniería química. Escuela de Ingenierías Industriales, Universidad de Valladolid.
4. Arroyo-Hernández, J. "Métodos de reducción de dimensionalidad : Análisis comparativo de los métodos APC , ACPP y ACPK," *UNICIENCIA* Vol. 30, No. 1, pp. 115–122, 2016.
5. Bajo Traver, M. "Aplicaciones prácticas del Análisis de Componentes Principales en Gestión de Carteras de Renta Fija (I). Determinación de los principales factores de riesgo de la curva de rendimientos. Practical applications of Principal Component Analysis in Fixed Inco," *Análisis Financ.*, Vol. 124, No. I, pp. 20–36, 2014.
6. Carmona Suárez, E. J. "Máquinas de Vectores Soporte (SVM)," Dpto. Inteligencia Artificial. ETS Ingeniería Informática, Universidad Nacional Educación a Distancia, documento técnico, pp. 1–25, 2014.
7. Chéruy, A. "Software sensors in bioprocess engineering," *J. Biotechnol.*, Vol. 52, No. 3, pp. 193–199, 1997.
8. Choi D. J. & Park, H. "A hybrid artificial neural network as a software sensor for optimal control of a wastewater treatment process," *Water Res.*, Vol. 35, No. 16, pp. 3959–3967, 2001.
9. Correa, G.J. & Montoya, L. M. "Aplicación del modelo ANFIS para predicción de series de tiempo," *Rev. Digit. la Fac. Ing. Lámpsakos*", Vol. 1, No. 9, pp. 12–25, 2013.

10. Güemes García, Enrique (2018). Diseño de sensores software para el control de la calidad de un proceso. Trabajo fin de grado. Departamento de Ingeniería de Sistemas y Automática, Escuela de ingenieros industriales Universidad de Valladolid, Valladolid.
11. Hernández-Pedreira, C. & Da Silva-Portofilipe, F. “Aplicación del control estadístico de procesos (CEP) en el control de su calidad.,” *Appl. Stat. Process Control it’s Qual. Control.*, Vol. 36, No. 1, pp. 130–145, 2016.
12. Hernando Gaviño, Ricardo (2010). Introducción a los sistemas de control: conceptos, aplicaciones y simulación con MATLAB. Ed. Pearson.
13. Hornik, Kurt; Stinchcombe, Maxwell & White, Halbert “Multilayer Feedforward Networks are Universal Approximators,” *Neural Networks*, Vol. 2, pp. 359–366, 1989.
14. Isasi Viñuela, Pedro; Galván León, Inés M. (2004) Redes Neuronales Artificiales: Un enfoque Práctico. Ed. Pearson Prentice Hall. Departamento de Informática, Universidad Carlos III de Madrid.
15. Kadlec, P.; Gabrys, B. & Strandt, S. “Data-driven Soft Sensors in the process industry,” *Comput. Chem. Eng.*, Vol. 33, No. 4, pp. 795–814, 2009.
16. Katsuhiko, Ogata (1998). Ingeniería de control moderna. Ed. Pearson - Prentice Hall.
17. Martín del Brío, Bonifacio; Sanz Molina, Alfredo (2001). Redes Neuronales y Sistemas Difusos. Ed. Alfaomega Ra – Ma.
18. Sánchez Fernández, A.; De la Fuente, María Jesús; Sainz Palmero, Gregorio (2015) “Fault detection in Wastewater Treatment Plants using Distributed PCA Methods”. IEEE 20th Conferencie on Emerging Technologies & Factory Automation (ETFA). doi: 10.1109/etfa.2015.7301504.

Páginas web consultadas

1. <file:///C:/Users/cursi/Downloads/Dialnet-BalanceoYEstabilizacionDelPenduloInvertidoEmpleand-4269086.pdf>

2. <http://grupo.us.es/qtocoma/pid/pid10/RedesNeuronales.htm>
3. <http://tusistemanervioso.blogspot.com/2008/02/divisiones-del-sistema-nervioso.html>
4. <http://www.cs.us.es/~fsancho/?e=72>
5. <http://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/>
6. http://www.profesormolina.com.ar/tecnologia/sens_transduct/que_es.htm
7. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-548X2009000300006
8. <https://definicion.de/sensor/> (no date)
9. <https://definicion.de/software/>
10. <https://es.slideshare.net/hbanda/computacin-neuronal>
11. <https://inteligenciartificialmca.wordpress.com/category/sin-categoria/>
12. https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/cursos-glisa-redes_neuronales.html/x185.html