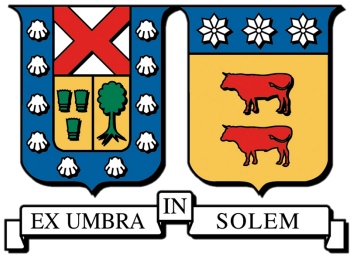
2017

ESTIMACIÓN DEL FLUJO AÉREO GLOTAL A PARTIR DE UNA SEÑAL DE ACELERÓMETRO UTILIZANDO REDES NEURONALES

FARÍAS CORREA, ROBERTO EUGENIO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



ESTIMACIÓN DEL FLUJO AÉREO GLOTAL A PARTIR DE UNA SEÑAL DE ACELERÓMETRO UTILIZANDO REDES NEURONALES

ROBERTO EUGENIO FARÍAS CORREA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES

PROFESOR GUÍA: MATÍAS ZAÑARTU
PROFESOR CORREFERENTE: MARÍA JOSÉ ESCOBAR

MAYO - 2017

Resumen

Muchos trastornos de la voz comunes son condiciones crónicas o recurrentes que pueden ser el resultado del uso inapropiado o excesivo de la voz (patrones defectuosos y/o abusivos del comportamiento vocal). A este tipo de trastornos de la voz se les refiere comúnmente como *hiperfunción vocal*, los cuales pueden llevar amplias consecuencias ya sea en lo social, profesional o personal.

Con el fin de proporcionar una solución no invasiva para la prevención y el monitoreo de este tipo de trastornos, han estado en desarrollo plataformas como el *monitor de la salud de la voz*, el cual busca a través del análisis de la aceleración proporcionada por una acelerómetro posicionado en la base del cuello, determinar medidas de relevancia médica (como lo es el flujo aéreo glotal) para el diagnóstico y prevención de trastornos de la voz tales como la hiperfunción vocal.

Este proyecto busca estudiar e implementar un algoritmo basado en un modelo de redes neuronales, el cual permita la estimación del flujo aéreo glotal utilizando una señal proveniente de un acelerómetro adherido a la piel del cuello de la persona. El objetivo es poner a prueba un acercamiento que use un modelo de redes neuronales y cuantificar su desempeño en comparación con otros acercamientos al problema, y de esta manera ayudar buscar una posible mejora en la eficacia de dispositivos ambulatorios que lo requiran, como el monitor de la salud de la voz.

Abstract

Many common voice disorders are chronic or recurrent conditions that may be the result of improper or excessive use of voice (defective and/or abusive patterns of vocal behavior). These types of speech disorders are commonly referred to as vocal hyperfunction, which can lead to farreaching social, professional or personal consequences.

In order to provide a non-invasive solution for the prevention and monitoring of this type of disorders, platforms such as the voice health monitor have been developed, which it looks for through the analysis of the acceleration provided by an accelerometer positioned at the base of the neck, determine measures of medical relevance (such as glottal airflow) for the diagnosis and prevention of voice disorders such as vocal hyperfunction.

This project seeks to study and implement an algorithm based on a neural network model, which allows estimation of glottal airflow using a signal from an accelerometer attached to the skin of the person's neck. The objective is to test an approach using a neural network model and quantify its performance in comparison with other approaches to the problem, and thus help to look for a possible improvement in the effectiveness of ambulatory devices that require it, such as the Voice Health Monitor.

Glosario

AC flow Amplitud peak to peak del flujo glotal inestable (AC flow) . 4

ACC Abreviación para acelerómetro, proviene de sus tres primeras letras de su palabra en inglés ”accelerometer” . 2

ANN Artificial Neural Network, red neuronal artificial en español. . 8

CPIF Filtrado inverso de fase cerrada (CPIF, por sus siglas en inglés) . 4

Deep Neural Networks Deep Neural Network, un tipo de red neuronal la cual se caracteriza por poseer varias capas ocultas. . 30

feedforward Corresponde a los tipos de redes de propagación hacia adelante. . 10

GPU Graphics processing unit, es un circuito electrónico especializado para acelerar la creación de imágenes con el propósito de mostrarlos en algun display . 30

GVV Glotal Volume Velocity en inglés, corresponde al flujo aéreo glotal . 4

IBIF Abreviación para Voice Health Monitor,o bien, filtrado inverso subglotal basado en impedancias en español . 1

MFDR Tasa máxima de caída del flujo (MFDR. maximum flow declination rate) . 4

MLP Multilayer Perceptron . 18

NARX Nonlinear autoregressive with exogenous inputs . 21

Pruning Técnica utilizada en redes neuronales para “podar” las neuronas o entradas no tan significativas, mejorando la velocidad de procesamiento y generalización de la red. . 30

RBF Radial Basis Function . 19

SPL Abreviación para Sound Pressure Level,o bien, nivel de presión del sonido en español . 3

VHM Abreviación para Voice Health Monitor,o bien, monitor de la salud de la voz en español . 2

Índice general

Capítulo 1

Introducción

El proyecto de "Estimación del flujo aéreo glotal a partir de una señal de acelerómetro utilizando redes neuronales" tiene por objetivo principal analizar el desempeño del uso de modelos de redes neuronales para la estimación del flujo aéreo glotal. Por esto, como introducción el primer tópico a tratar es acerca del contexto en que esta memoria está situada: acerca de los trastornos de la voz, el porqué del análisis del flujo aéreo glotal, la forma y el dispositivo que fue utilizado para las mediciones del acelerómetro. En general, del estado actual de las investigaciones. Dado que uno de los objetivos de esta memoria es también comparar el desempeño del uso de redes neuronales con la técnica de filtrado inverso subglotal (IBIF), el siguiente tema a tratar es sobre el uso de esta técnica, su metodología y resultados.

A continuación se prosigue con el estudio y análisis a nivel conceptual de lo que son las redes neuronales, sus características y aplicaciones. Todo esto para concluir el por que son una alternativa viable para el uso en esta memoria.

1.1 Trastornos de la voz

Según un estudio realizado el 2005 realizado en Estados Unidos [1], casi el 30% de la población adulta ha experimentado algún trastorno de la voz durante su vida, y casi el 7% informó que poseía algún problema de voz al momento del estudio. Aunque la mayoría de los trastornos de la voz reportados fueron de corta duración (es decir, < 4 semanas), estos tuvieron un impacto ocupacional significativo, con el 7,2% de los participantes en el estudio que poseían empleos llegaron a faltar al menos un día al trabajo durante el año a causa de su trastorno de la voz. El 2% informó de al menos 4 o más días de ausencia relacionada con sus problemas en la voz. Otros indicaron que los problemas de la voz le causaron restricciones en sus actividades relacionadas con el trabajo, lo que sugiere que los trastornos de la voz tienen un impacto funcional considerable. Teniendo en cuenta estas estimaciones, los costos económicos incurridos a causa de ausentismo relacionado con la voz y la pérdida

de productividad, junto con las intervenciones médicas y de comportamiento dirigidas a restaurar la voz, tanto en los EE.UU. como en muchos países del mundo, es muy posible sean muy considerables.

Muchos trastornos de la voz comunes son condiciones crónicas o recurrentes que pueden ser el resultado del uso inapropiado o excesivo de la voz (patrones defectuosos y/o abusivos del comportamiento vocal) a lo cual se refiere comúnmente como hiperfunción vocal [2]. Como se observó en el estudio previamente señalado, estos pueden interrumpir o impedir la comunicación oral normal y por lo tanto tener amplias consecuencias ya sea en lo social, profesional o personal.

Los trastornos de la voz más comunes relacionados con hiperfunción [2], se cree que [3] surgen de una variedad de patrones relacionados con el uso de voz que incluyen la sonoridad excesiva, el tono inapropiado, reducción de la eficiencia vocal, y el hablar por períodos excesivos de tiempo [4]. Los médicos actualmente confían en la auto-evaluación y auto-monitoreo del paciente para juzgar la prevalencia y persistencia de comportamientos vocalmente abusivos durante el diagnóstico y el tratamiento médico. Sin embargo, los pacientes tienden a ser altamente subjetivos y propensos a juicios poco confiables del uso de su propia voz, esto se debe a que muy probablemente los patrones del uso y mal uso de la voz han sido habituados por el paciente y por tanto se encuentran bajo el nivel de conciencia del individuo.

Debido la dificultad para realizar una evaluación precisa, un objetivo en curso en la evaluación clínica de la voz es el desarrollo y la utilización de las medidas derivadas de procedimientos no invasivos para cuantificar los trastornos de voz basados en el comportamiento, tales como la hiperfunción vocal. Los sistemas de seguimiento de voz portátiles buscan proporcionar medidas más confiables y objetivas del uso de la voz, y de esta manera, se mejoraría en gran medida el tratamiento de los trastornos hiperfuncionales debido a la capacidad de monitorear y cuantificar las conductas perjudiciales de forma no invasiva, además de la capacidad de proporcionar información en tiempo real, lo cual podría ayudar a corregir hábitos perjudiciales y llevar un funcionamiento vocal saludable.

Dados los importantes beneficios del monitoreo no invasivo, en el último tiempo ha estado en desarrollo una plataforma flexible y fácil de usar para el monitoreo ambulatorio de la voz, el cual es referido como ”Monitor de la salud de la voz” (VHM, por sus siglas en inglés) [5]. El sistema consiste en un smartphone como la plataforma de adquisición de datos y un acelerómetro en miniatura (ACC) como el sensor de la fonación, el cual va montado en la base del cuello. Además de sus características interactivas, el VHM (que se puede ver en la figura 1.1) permite la grabación de la señal de la aceleración sin procesar de la piel del cuello, con capacidad de almacenamiento suficiente como para grabar durante más de 18 horas por día, por al menos 7 días. El registro de la señal de aceleración de la piel del cuello permite la investigación de nuevas medidas relacionadas con el uso de voz sobre la base de un modelo de sistema vocal, y facilita la adquisición de datos para comprender mejor los comportamientos vocales relacionados con actividades diarias.

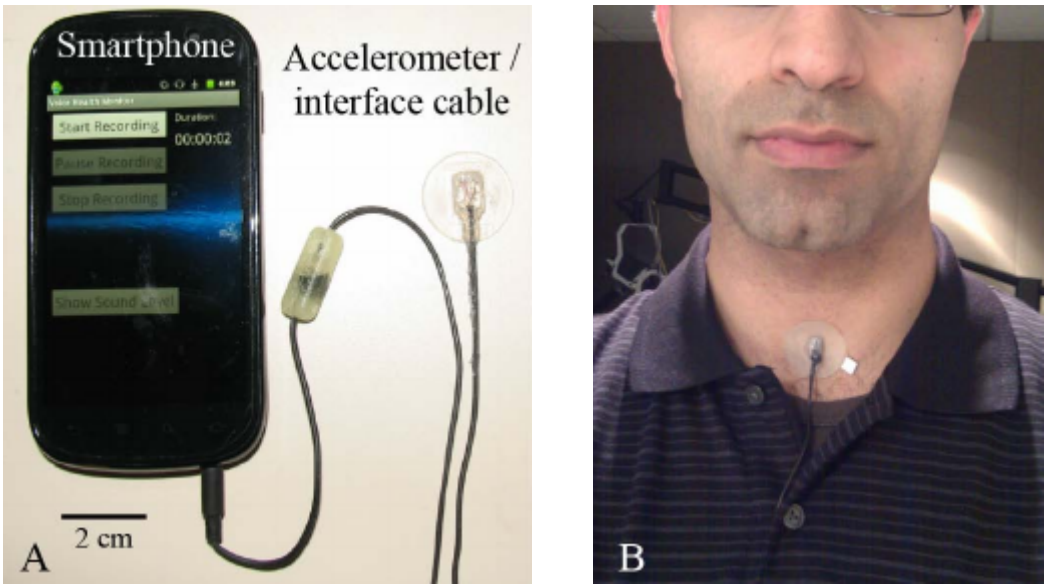


Figura 1.1: Monitor de la salud de la voz: (a) Smartphone, acelerómetro (b) Acelerómetro instalado para recolección de datos.

En las tecnologías actuales, los enfoques basados en parámetros acústicos que se han implementado para el biofeedback en tiempo real, utilizan un umbral para ya sea para la frecuencia fundamental (f0, tasa vibratoria básica de las cuerdas vocales) o el nivel de presión del sonido (sound pressure level, SPL), la cual está relacionada con la cantidad de flujo de aire de los pulmones y con la resistencia a este flujo ejercido por las cuerdas vocales [6, 7].

Sin embargo, el VHM ofrece una oportunidad para incorporar nuevos y más sofisticadas capacidades de biofeedback en tiempo real basados en medidas avanzadas de la función vocal para mejorar potencialmente la identificación y tratamiento de conductas vocales perjudiciales (por ejemplo, hiperfunción vocal). Además, la facilidad de uso de la plataforma basada en smartphone, permite estudios clínicos con grandes cantidades de muestras que pueden identificar medidas de gran utilidad estadística para diferenciar entre los patrones hiperfuncionales y normales del comportamiento vocal.

Un enfoque reciente proporciona un método para estimar las medidas aerodinámicas de la función vocal a partir de la señal del acelerómetro, utilizando un modelado del sistema vocal y realizando filtrado inverso subglotal basado en impedancias (IBIF, por sus siglas en inglés) [8].

1.2 Filtrado inverso subglotal basado en impedancias

El filtrado inverso de sonidos de la voz hablada es utilizado para estimar la fuente de la excitación en la glotis removiendo los efectos de las cargas acústicas de una señal de salida.

Esta técnica ha sido principalmente aplicada para el tracto vocal usando grabaciones de vocales sostenidas obtenidas de ya sea por la presión acústica radiada o el flujo de aire. La mayoría de los algoritmos de filtrado inverso se enfocan en eliminar las resonancias del tracto vocal, pero para el uso en el VHM se debe hacer uso de un enfoque distinto, en el cual se debe quitar las resonancias sub-glotales y efectos de la piel de las medidas hechas con el acelerómetro asentado en la base del cuello, ofreciendo grandes ventajas sobre los métodos tradicionales que tratan con el tracto supraglotal.

Por esto, ideó un modelo basado en analogías mecano-acústicas, principios de líneas de transmisión basados en la fisiología del sistema (cuya representación se muestra en la figura 1.2), con el fin de aplicar la técnica de filtrado inverso.

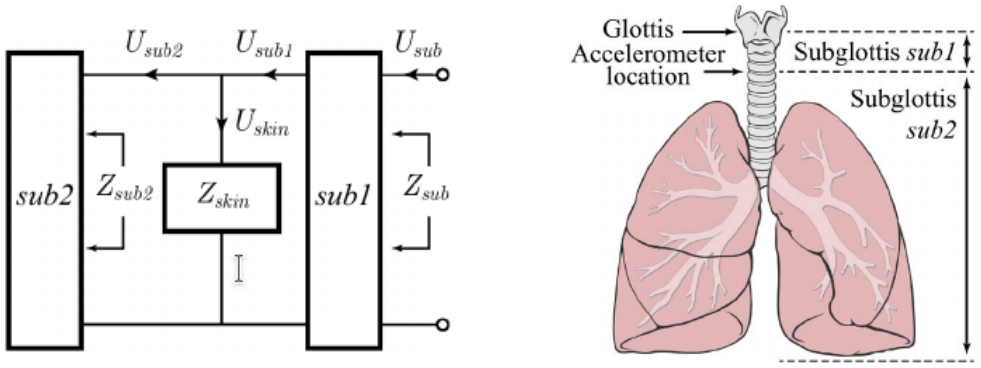


Figura 1.2: Representación del sistema utilizando principios de lineas de transmisión.

En este modelo, el tracto subglotal está compuesto de dos secciones, $sub1$ y $sub2$ que representan la porción de la tráquea extratorácica que está por sobre y debajo de la posición del acelerómetro, respectivamente. La impedancia de la piel del cuello Z_{skin} se encuentra en paralelo entre las secciones $sub1$ y $sub2$. Por otra parte, la variable que se desea estimar corresponde a $-U_{sub}$ el cual representa al flujo aéreo glotal, y se dispone la derivada de U_{skin} la cual representa la señal del acelerómetro.

Para poner a prueba este enfoque, se obtuvieron estimaciones del flujo volumétrico glotal (GVV, glottal volume velocity) y su derivada a partir de la señal del acelerómetro, y generaron las medidas que son relevantes para el estudio. Se contrastaron los resultados obtenidos con lo que se obtendría utilizando el filtrado inverso de fase cerrada (CPIF, por sus siglas en inglés), el cual es uno de los métodos más aceptados que se aplican realizando el filtrado inverso sobre el tracto vocal, y se observó que las formas onda obtenidas fueron bastante similares, indicando que este enfoque es capaz de recuperar los componentes claves de la fuente glotal. Más particularmente concluyó que para dos medidas derivadas del flujo volumétrico glotal; que son la tasa máxima de caída del flujo (MFDR, maximum flow declination rate) y la amplitud peak to peak del flujo glotal inestable (AC flow), el error y sus variaciones son consideradas lo suficientemente bajas para el uso de este enfoque en el ámbito clínico (un error medio porcentual menor al 10%).

El objetivo de este proyecto corresponde al uso de un nuevo enfoque, en el cual se haga uso de redes neuronales para realizar un mapeo entre la señal del acelerómetro y el flujo volumétrico glotal, para luego comparar su eficacia con los resultados obtenidos utilizando IBIF.

1.3 Redes Neuronales Artificiales

Desde sus inicios el trabajo en redes neuronales ha sido motivado por el reconocimiento de que el cerebro humano procesa tareas de una forma completamente distinta a la que una computadora convencional puede hacerlo. Nuestro cerebro puede interpretarse como un sistema de procesamiento de información altamente complejo, no lineal y que trabaja paralelo, el cual tiene la capacidad de organizar sus constituyentes estructurales, llamadas neuronas, para llevar a cabo determinadas tareas (como por ejemplo, reconocimiento de patrones, procesar las percepciones, etc) mucho más rápidamente y con mayor precisión que cualquier dispositivo que hoy exista. Solo basta con considerar la velocidad en que puede ejecutar una tarea rutinaria como reconocer a alguien en la calle (que a la más poderosa computadora se le puede hacer complejo), para darse cuenta del potencial que tiene, con respecto a una computadora.

Una de las características más interesantes del cerebro es la habilidad de construir sus propias reglas de comportamiento a través de lo que usualmente nos referimos como ”experiencia”. La experiencia nos permite aprender acerca de nuestros alrededores para así adaptarnos (a esto se le llama plasticidad).

Considerando todas las notables características que posee el cerebro, hacen de él una inspiración para el paradigma de procesamiento de información conocido como redes neuronales, y motivar la búsqueda de algún modelo que represente las características de interés de una red neuronal biológica, para así replicar su funcionamiento usando componentes electrónicos o realizando una simulación en software en una computadora digital.

1.3.1 Redes neuronales biológicas

Una neurona biológica genérica se ilustra en la figura 1.3, de ella se distinguen tres componentes que son de particular interés para el modelado de esta y, por ende, comprender una neurona artificial: dendrita, soma o núcleo y axón. Las muchas dendritas en nuestro cerebro son extensiones ramificadas pertenecientes a una neurona, las cuales actúan como receptores de las señales de otras neuronas. Estas señales corresponden a impulsos eléctricos que son transmitidos a través del espacio sináptico (espacio intermedio entre la neurona transmisora y receptora) por medio de un proceso químico, en una estructura especializada llamada sinapsis. Luego el soma, o cuerpo de una célula, suma las señales entrantes y cuando suficiente entrada es recibida la célula se gatilla, en otras palabras, transmite una

señal a otras neuronas u otros tipos celulares a través de su axón (estructura que representa el lado de salida de una neurona) [9].

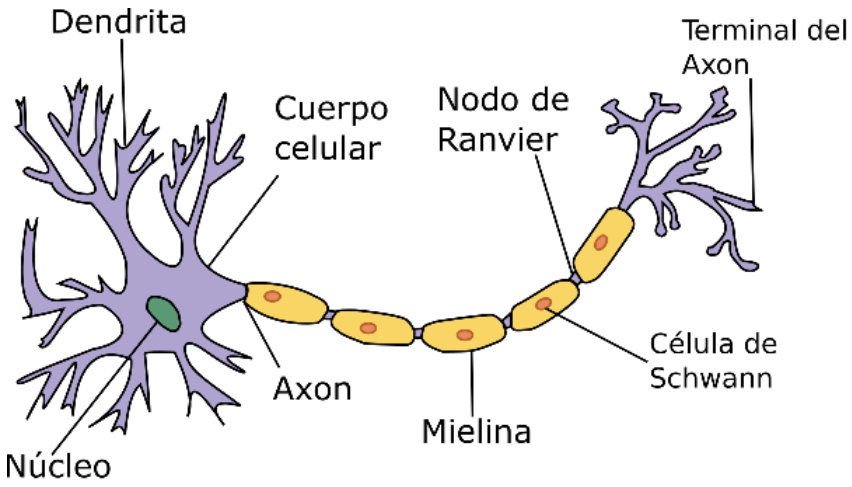


Figura 1.3: Diagrama básico de una neurona.

Varias de las características claves de las redes neuronales artificiales son sugeridas de propiedades de las redes neuronales biológicas [10], tales como:

- El elemento procesador recibe muchas señales.
- Las señales pueden ser modificadas por un peso en la neurona receptora.
- El elemento procesador suma las entradas ponderadas (afectadas por un peso).
- Bajo las circunstancias apropiadas (entrada suficiente), la neurona transmite una salida.
- La salida de una neurona puede llegar a muchas otras neuronas (las ramificaciones del axón).

1.3.2 Redes neuronales Artificiales

Según la definición detallada en [11],

Una Red Neuronal es un procesador paralelo y distribuido que tiene la facilidad natural para almacenar conocimiento experimental y hacerlo útil para su uso, asemejando al cerebro en dos aspectos:

- *El conocimiento es adquirido por la red a través de un proceso de aprendizaje.*

- *La "fuerza" de las conexiones interneurona, conocida como pesos sinápticos son utilizados para almacenar el conocimiento adquirido.*

Como ya se ha mencionado antes, una red neuronal artificial es un sistema de procesamiento de información que tiene ciertas características de funcionamiento en común con las redes neuronales biológicas. Las redes neuronales artificiales se han desarrollado como generalizaciones del modelo matemático de la cognición humana o la biología neuronal, basado en los supuestos de que:

1. El procesamiento de la información se produce en muchos elementos simples llamados neuronas.
2. Las señales se hacen pasar entre las neuronas a través de conexiones.
3. Cada conexión tiene un peso asociado, que, en una red neuronal típica, multiplica la señal transmitida.
4. Cada neurona se aplica una función de activación (usualmente no lineal) a su entrada (suma de las señales de entrada ponderados) para determinar su señal de salida.

Una red neuronal se caracteriza por (1) su patrón de conexiones entre las neuronas (llamado su arquitectura), (2) su método de determinación de los pesos de las conexiones (llamada su entrenamiento, o el algoritmo de aprendizaje), y (3) su función de activación.

Una red neuronal se compone de un gran número de elementos de procesamiento simples llamados neuronas, unidades, células o nodos. Cada neurona está conectada a otras neuronas por medio de enlace de comunicación dirigida, cada uno con un peso asociado. Los pesos representan información utilizada por la red para resolver un problema.

Cada neurona tiene un estado interno, llamado su nivel de actividad o activación, que es una función de las entradas que ha recibido. Típicamente, una neurona envía sus activaciones como una señal para varias otras neuronas. Es importante tener en cuenta que una neurona puede enviar una sola señal a la vez, a pesar de que la señal se transmite a varias otras neuronas.

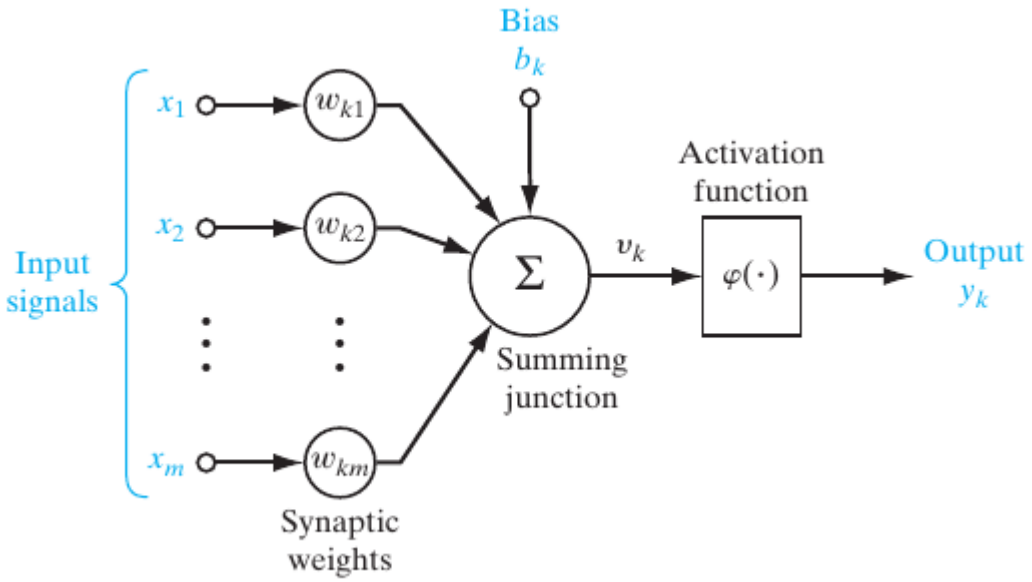


Figura 1.4: Modelo no lineal de una neurona, designada como k.

Por ejemplo, considérese el diagrama de bloques propuesto en la figura 1.4. Este muestra un modelo de neurona que forma la base para el diseño de una gran familia de redes neuronales. Por otra parte, en él se pueden identificar tres elementos básicos del modelo de una neurona:

1. Un set de sinapsis o enlaces (conexiones entre neuronas), cada uno de los cuales es caracterizado por una ponderación (peso) propia. A diferencia del peso de una sinapsis en el cerebro, el peso sináptico de una ANN puede ir en un rango que incluye valores tanto positivos como negativos.

2. Un sumador, el cual suma las señales de entradas, ponderadas por su respectivo peso sináptico de la neurona; la operación descrita aquí constituye un combinador lineal.

3. Una función de activación, para delimitar la amplitud de la salida de una neurona. La función de activación es también referida como función aplastante (squashing function), la cual aplasta el rango de la amplitud permisible (los límites) de la señal de salida a un valor finito.

Además, en el modelo presentado en la figura 1.4 también incluye un factor aplicado externamente llamado ”bias”, denotado por b_k. La bias tiene el efecto de escalar linealmente la entrada de la neurona a la función de activación.

1.3.2.1 Características importantes de las ANN

Función de Activación

Existen distintos tipos función de activación, entre los más utilizados, se encuentran:

Función umbral

Para este tipo de función de activación se tiene que:

$$\varphi(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{if } v < 0 \end{cases}$$

Gráficamente, es como se muestra a continuación:

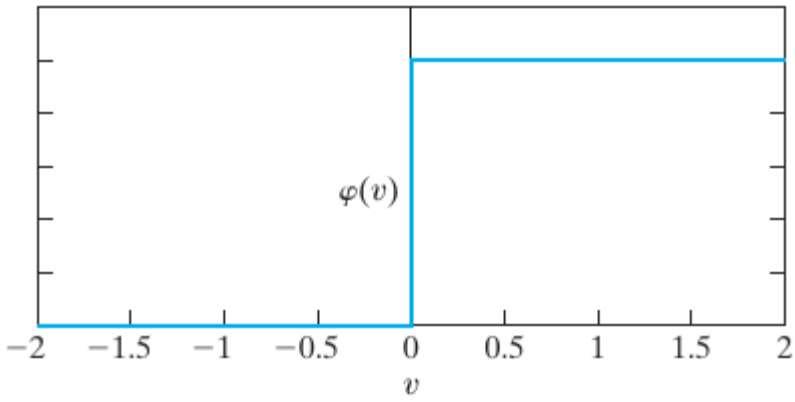


Figura 1.5: Representación gráfica de la función umbral

Función sigmoide

La función sigmoide es por lejos la más común utilizada en la construcción de redes neuronales. Es definida como una función estrictamente creciente y presenta un balance entre un comportamiento lineal y no lineal. La función sigmoide representa un caso particular de la función logística y está definida de la siguiente forma:

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

Gráficamente, variando el parámetro "a", posee el efecto que se muestra a continuación:

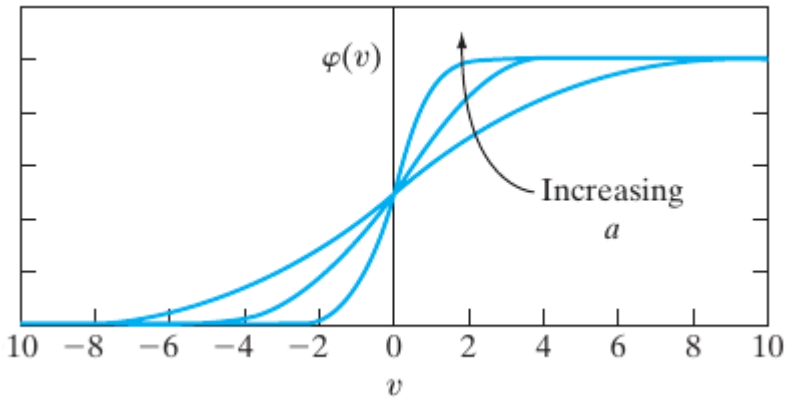


Figura 1.6: Representación gráfica de la función sigmoide, modificando el parámetro de pendiente "a".

Arquitectura

Otra propiedad importante que caracteriza una red neuronal, es su arquitectura. Las más comunes son dos tipos de redes de propagación hacia adelante (feedforward) o acíclicas (que van desde la entrada a la salida sin existir ningún ciclo), junto con las redes neuronales recurrentes, que se detallan a continuación:

Monocapa (single-layer feedforward)

La forma más simple de una red por capas, se tiene una capa de nodos correspondientes a la entrada a la red, directamente conectadas a la capa de salida de la red, una ilustración de esta red, de muestra a continuación:

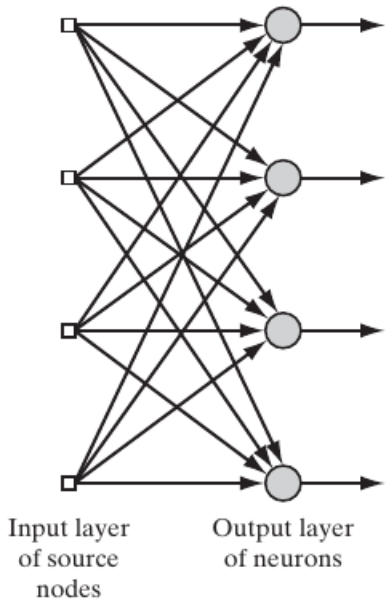


Figura 1.7: Red monocapa de propagación hacia adelante.

Multicapa (multilayer feedforward)

Similar a la monocapa, esta red tiene la presencia de una capa de entrada y otra de salida, pero además, posee entre estas capas la presencia de otras capas llamadas capas escondidas (término referente a que no se ven directamente desde la entrada o de la salida de la red). Agregando una o más capas escondidas, permite a la red extraer estadísticas de ordenes mayores de su entrada.

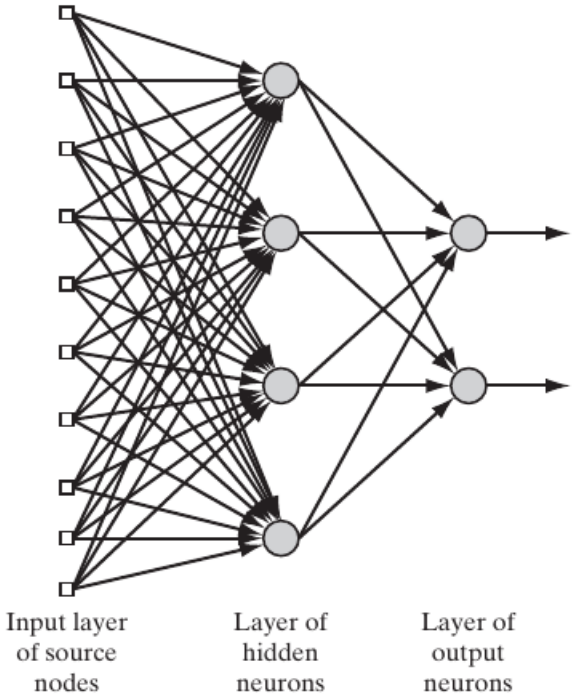


Figura 1.8: Red multicapa de propagación hacia adelante.

Redes recurrentes

Las redes neuronales recurrentes de distinguen de las redes de propagación hacia adelante en que estas poseen al menos un lazo de realimentación. La presencia de lazos de realimentación tiene un profundo impacto en la capacidad de aprender y el rendimiento de la red. Los lazos de realimentación implica el uso de elementos de retardo (denotados por z^{-1}), lo que resulta en un comportamiento no lineal dinámico, asumiendo que la red neuronal contiene unidades no dinámicas. Algunos ejemplos de esta arquitectura se presentan en la figura 1.9, en la izquierda una red recurrente sin realimentación propia y sin neuronas ocultas, a la derecha una red recurrente con neuronas ocultas.

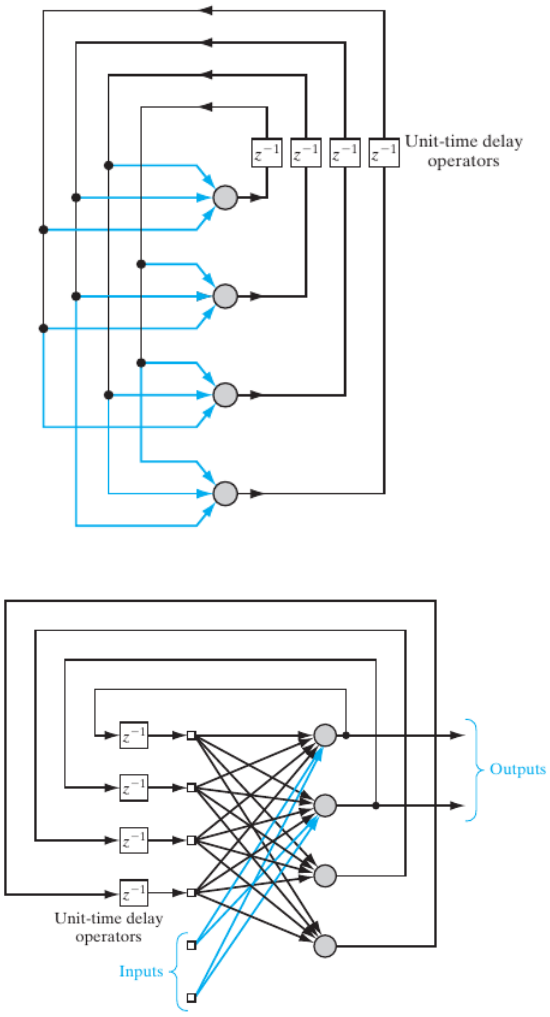


Figura 1.9: Ejemplos de redes recurrentes.

Paradigmas de Aprendizaje

Existen tres grandes paradigmas de aprendizaje en el tópico de las redes neuronales artificiales, cada uno de los cuales se enfoca en una tarea particular de aprendizaje, estas se detallan a continuación:

Aprendizaje Supervisado

Dado un conjunto de puntos (x, y), $x \in X$, $y \in Y$, la meta es encontrar una función $f: X \to Y$ en el dominio de funciones que coincida con los ejemplos. En otras palabras, deseamos inferir el mapeo implicado en los datos, y por tanto, la función de costo en este tipo de aprendizaje se relaciona con la diferencia en el mapeo entre los datos reales (conjunto de puntos (x, y)) y lo que se obtiene con la función f obtenida con el entrenamiento. Aprendizaje supervisado es principalmente usado en tareas reconocimiento de patrones (también conocido como clasificación) y regresión (también conocido como aproximación).

Aprendizaje no Supervisado

En este caso, solo se conoce un conjunto de datos x y la función de costo a ser minimizada, la cual puede ser cualquier función de los datos x y de la función resultado de nuestra red neuronal, f. Algunas de las tareas en las que el aprendizaje no supervisado es utilizado en tareas de estimación.

Aprendizaje reforzado

Es una técnica útil para resolver problemas de optimización de control, esto es, el problema de reconocer (o encontrar) la mejor acción en cada estado en que el sistema se encuentra optimizando una función objetivo, por ejemplo, la recompensa por unidad de tiempo y el total descontado en una ventana de tiempo. Las tareas en las que en general este modo de aprendizaje es utilizado son problemas de control, juegos y tareas que requieren la toma de decisiones secuenciales.

Ventajas y desventajas del uso de redes neuronales

Múltiples son las ventajas y desventajas del uso de redes neuronales [13], a continuación se muestra un breve listado de algunas de ellas:

Ventajas:

- Los modelos de redes neuronales requieren una formación estadística menos formal para desarrollarlas.
- Los modelos de redes neuronales pueden detectar implícitamente relaciones no lineales complejas entre variables independientes y dependientes.
- Los modelos de redes neuronales tienen la capacidad de detectar todas las posibles interacciones entre las variables predictoras.
- This is spartaLas redes neuronales pueden desarrollarse utilizando múltiples y variados algoritmos de entrenamiento distintos.

Desventajas:

- Las redes neuronales son una ”caja negra” y tienen una capacidad limitada para identificar explícitamente las posibles relaciones causales.
- Los modelos de redes neuronales pueden ser más difíciles de usar en la práctica.
- La implementación de redes neuronales requiere mayores recursos computacionales.
- Los modelos de redes neuronales son propensos a overfitting.
- El desarrollo de modelos de redes neuronales es empírico, y aún quedan por resolver muchos problemas metodológicos para su implementación.

Aplicaciones

Debido a la gran versatilidad de las redes neuronales artificiales, estas han sido utilizadas en diversos campos y también han sido utilizadas en variadas aplicaciones de las cuales algunas son:

- Control de sistemas [14]
- Problemas de regresión [15]
- Detección de rostros [16]
- Generar pronósticos [17]
- Detección de texto [18], entre otros.

Tantas como aplicaciones tienen las redes neuronales, existen variados desarrollos e implementaciones de éstas, las cuales buscan optimizar el rendimiento y funcionamiento de éstas. Un ejemplo, es la librería desarrollada por google “TensorFlow” la cual, provee implementaciones no solo de redes neuronales artificiales, sino que a una gran variedad de algoritmos de aprendizaje, buscando la optimización del funcionamiento y la versatilidad de este, ayudando a el funcionamiento de ésta en una gran variedad de sistemas desde celulares y tablets, hasta grandes sistemas distribuidos con cientos de máquinas [19].

Considerando el potencial que tienen las redes neuronales artificiales para las muchas aplicaciones en las que son utilizadas, y el creciente desarrollo en este campo hacen de las redes neuronales artificiales una gran propuesta para estudiar y resolver la aplicación que esta memoria trata.

1.4 Objetivos

Este proyecto busca estudiar e implementar un algoritmo basado en un modelo de redes neuronales, de modo que permita la estimación del flujo aéreo glotal utilizando una señal proveniente de un acelerómetro adherido a la piel del cuello de la persona. La idea es la utilización de este trabajo con el fin de poner a prueba un acercamiento que use un modelo de redes neuronales en ayuda del monitoreo y diagnóstico de trastornos de la voz y compararlo con otros acercamientos existentes, como el filtrado inverso basado en impedancia (IBIF).

En la figura 1.10, se muestra el esquema general del trabajo a realizar. En esta figura, se puede ver con mayor claridad el rol de este proyecto y su finalidad.

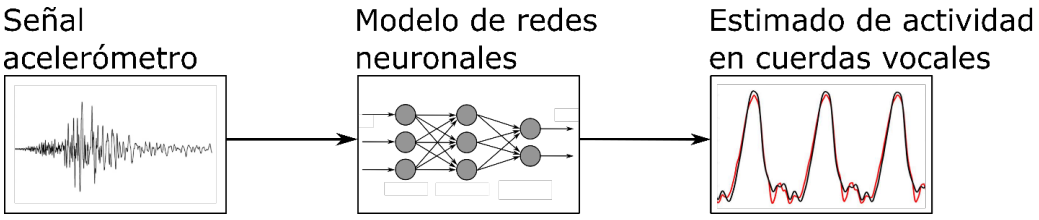


Figura 1.10: Esquema general del trabajo a realizar

Capítulo 2

Análisis y estudio de alternativas de solución

Dada la gran variedad de aplicaciones y de modelos existentes de redes neuronales, es necesario dar comienzo a este trabajo de memoria con un estudio de las posibles alternativas de solución, lo cual permite tener una mirada más amplia de los métodos que existen y son usados en algunas aplicaciones o investigaciones similares.

A continuación se muestran algunas alternativas para realizar la estimación de datos requerida en este proyecto, y se explica las razones por las cuáles éstas soluciones son viables para la resolución de la problemática, considerando sus ventajas y desventajas como también sus posibles implementaciones y usos para este proyecto. Dado que algunos tópicos pueden ser muy amplios, solo se dará una breve revisión enfocada en el uso que se le podría dar en este proyecto.

Antes de comenzar, es necesario señalar el paradigma de aprendizaje que cumpla con los requerimientos de este proyecto. Basado en el estudio hecho previamente capítulo 1, sabemos que son tres los principales, estos son: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje reforzado. Entre estos tres paradigmas, por razones obvias, el que cumple con las necesidades de nuestro proyecto es el aprendizaje supervisado, el cual, en palabras simples con un vector de entrada X y un vector de salida Y tiene por objetivo encontrar una función f tal que $f : X \to Y$, o bien, $y = f(x)$.

Entre los modelos existentes, los cuales corresponden a aprendizaje supervisado se han seleccionado tres modelos distintos los cuales pueden ser una alternativa válida (entre las muchas existentes), de los cuales todos tienen la capacidad de realizar un mapeo no lineal. Éstos se detallan a continuación.

2.1 Multilayer perceptron

Un perceptrón corresponde a una red de propagación hacia adelante (feedforward), la cual consta de: la capa de entrada (función identidad), luego dentro de la neurona, posee una etapa de suma ponderada (función de propagación), para finalizar pasando una función umbral(función de activación). Este modelo de una neurona se puede apreciar en la figura 2.1.

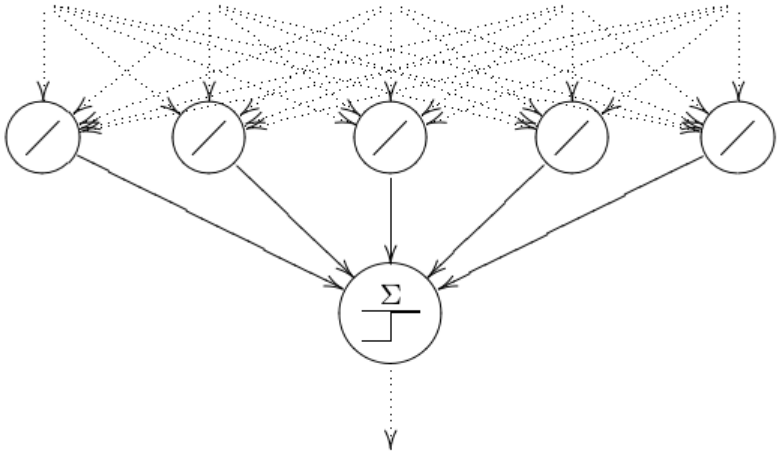


Figura 2.1: Arquitectura de un perceptrón con una capa de conexiones variables

Basado en el modelo descrito de perceptrón, una red perceptrón multicapa (MLP, por sus siglas en inglés) corresponde a una combinación de múltiples perceptrones, con dos o más capas de ponderaciones modificables. Un ejemplo se muestra a continuación.

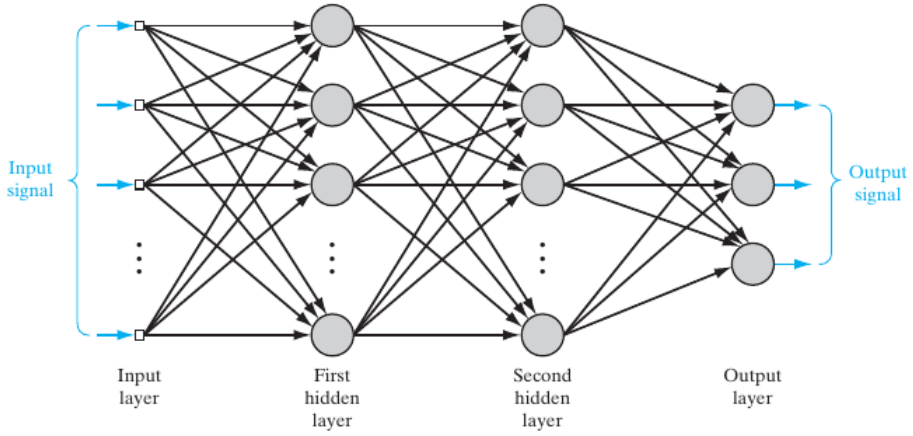


Figura 2.2: Gráfico de un ejemplo de perceptrón multicapa con dos capas escondidas

Esta red del tipo feedforward (propagación hacia adelante) es uno de los modelos más utilizados para tareas de reconocimiento de patrones, predicción y ajuste de funciones. Esta topología a diferencia del perceptrón descrito previamente, puede hacer uso de distintas funciones de activación, y entre las cuales, la más utilizada es la tangente hiperbólica [20] (entre otras) debido a que esta es: no lineal (lo que permite que una red neuronal de dos capas pueda probarse como un aproximador universal [21]), continuamente diferenciable (lo que permite usar métodos basados en gradientes para optimizar la red [22]), entre otras características.

En cuanto a la cantidad de capas y neuronas que la red debe poseer, debe ser analizada bajo un estudio experimental exhaustivo, debido a que no existe teoría bien desarrollada en este tema [11], pero debido a que el MLP representa un aproximado universal de funciones, se recomienda partir con 1 o 2 capas y verificar sus resultados [20] (dado que aumentar las capas y neuronas presentes en la red dificulta el problema de optimización y por otra parte aumenta el riesgo de overfitting). Este proceso de estudio exhaustivo de la red se puede ver aplicado en distintas investigaciones [23, 24, 25].

Ventajas y Desventajas de la alternativa

De acuerdo a lo antes expuesto, la alternativa del uso de redes neuronales del tipo MLP para la estimación de datos presenta las siguientes características:

- La elección de la cantidad de capas y neuronas puede dificultar la tarea de encontrar la topología óptima.
- Al ser tan popular, existe mucha literatura acerca de este modelo y su implementación se encuentra presente en la mayoría de los toolbox y librerías existentes sobre machine learning y redes neuronales.
- Al ser de tipo feedforward (aunque dependiendo de la función de activación) puede ser entrenada utilizando métodos numéricos altamente eficientes tales como Levenberg-Marquardt.
- Entrenamiento es más lento que las RBF (ver apartado 2.2).
- Ampliamente utilizada en investigaciones.
- Después de entrenamiento es más rápida que RBF.

2.2 Radial Basis Function

Similar a las redes MLP las redes RBF son del tipo feedforward, pero estas se caracterizan por utilizar solo tres capas (la capa de entradas, la capa de salida y solo una oculta). Un ejemplo de esta se puede ver a continuación en la figura 2.3.

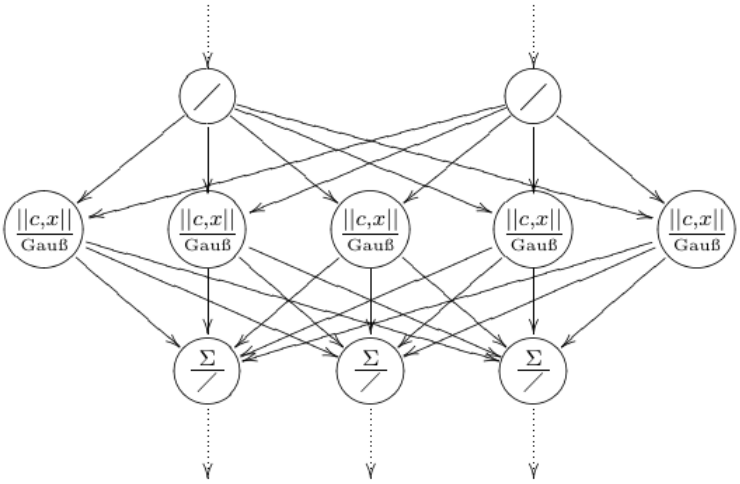


Figura 2.3: Un ejemplo de una red RBF con dos neuronas de entrada, cinco escondidas y tres de salida

La primera capa (la capa de entradas) solo traspasa sus valores sin ningún tipo de ponderación (función identidad). La capa escondida sin embargo, a diferencia de el perceptrón utiliza la distancia euclidiana, en vez de una suma ponderadas como función de propagación y como función de activación, una función gaussiana. Finalmente, la capa de salida le asigna a una suma ponderada a sus entradas, con función de activación la función identidad.

Debido a que este modelo solo posee tres capas, la única tarea posible es seleccionar la cantidad de neuronas que la capa escondida tendrá. Esta tarea al igual que el modelo MLP debe ser obtenido de forma exhaustiva buscando la combinación que posee mejores resultados.

Ventajas y desventajas de la alternativa

Bajo las características previamente descritas, algunas características de esta solución son:

La elección de la cantidad de capas ya no es un problema, la cantidad de neuronas y la elección de los centros puede dificultar la tarea de encontrar la topología óptima.

- Al ser casi tan popular como el modelo MLP, existe mucha literatura acerca de este modelo y su implementación se encuentra presente en la mayoría de los toolbox y librerías existentes sobre machine learning y redes neuronales.
- Al ser de tipo feedforward (aunque dependiendo de la función de activación) puede ser entrenada utilizando métodos numéricos altamente eficientes,
- No tan utilizada en investigaciones y por tanto menos reconocida.
- No permite extrapolar [20].

- Entrenamiento es más rápido que las MLP.
- Después de entrenamiento es más lenta que MLP.

2.3 Nonlinear autoregressive with exogenous inputs

Nuestra tercera alternativa a diferencia de las otras anteriores corresponde a una arquitectura recurrente. Un ejemplo de este modelo se muestra a continuación.

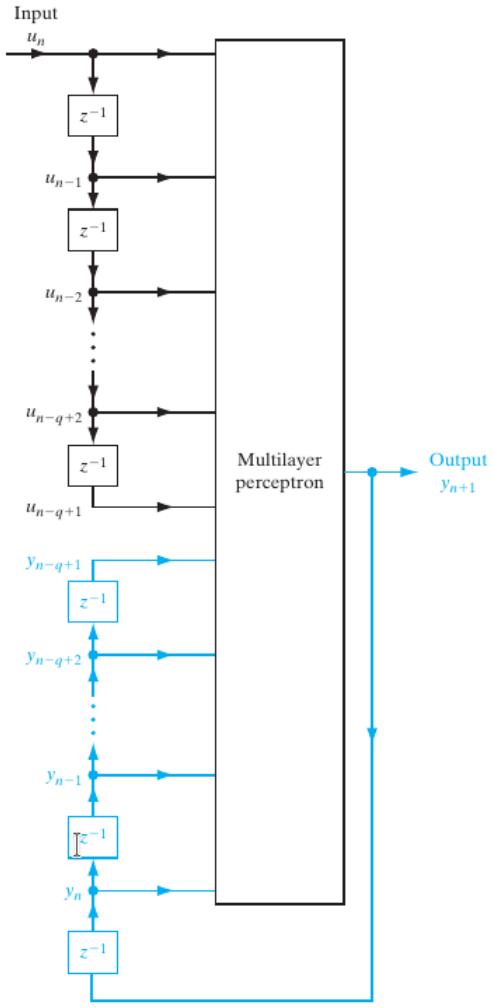


Figura 2.4: Representación del modelo no lineal autorregresivo con entradas exógenas (NARX).

Este tipo de arquitectura de diferencia de la arquitectura feedforward dado que presenta al

menos un lazo de realimentación, lo que hace que nuestra red neuronal depende ahora de la misma salida de esta. En el caso del modelo NARX, presenta dos tipos de entradas:

Los valores presentes y pasados de la entradas denotadas con el vector "u", las cuales representan las entradas exógenas de la red.

Valores pasados de la salida y.

Como es evidente, este tipo de modelo responde temporalmente a la señal externamente aplicada lo cual permite hacer uso de patrones temporales que existan en la señal. Las aplicaciones para este modelo son variadas algunas de ellas son utilizar este modelo como predictor, como filtro no lineal o bien para modelar sistemas dinámicos no lineales.

En cuanto a la topología general de la red, este deja aún más variables a considerar, pero cabe destacar que hacer uso de pares de entradas y salidas conocidas para entrenar esta red, permite hacer uso de todos los métodos conocidos para entrenar MLP.

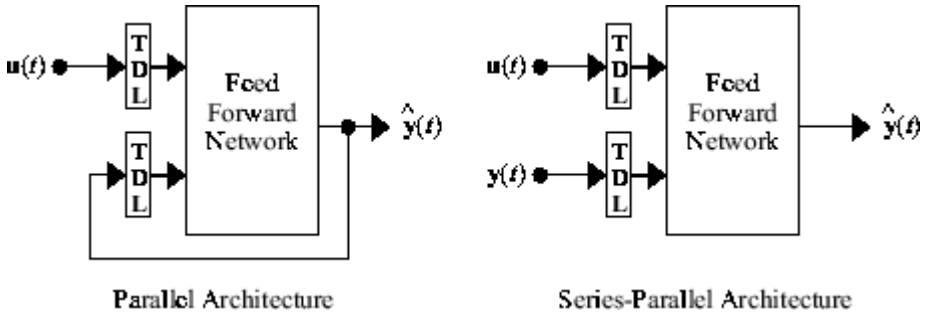


Figura 2.5: Simplificación del modelo de red recurrente.

Esto es de mucha utilidad ya que este modelo permite explotar características temporales de la red sin mayores dificultades de entrenamiento más que el modelo feedforward a utilizar (por ejemplo utilizar una red feedforward MLP entrega todas las optimizaciones de su implementación).

Ventajas y desventajas de la alternativa

Basado en lo expuesto previamente, se pueden identificar las siguientes características:

- Una clara ventaja, es la identificación de patrones temporales que puedan surgir en la red.
- Es posible utilizar todos los modelos feedforward y heredar sus optimizaciones.
- Existen variados tipos de implementación para este tipo de red, pero menos que sus otras alternativas.
- Debido a que se añaden características temporales, esto aumenta el número de variables a considerar para encontrar un modelo óptimo.

- Si es que no existen patrones temporales, no existe mayor beneficio.
- Al incorporar mayor cantidad de características a los modelos analizados anteriormente, esto disminuye su velocidad de entrenamiento, junto con su velocidad de ejecución.
- Es muy utilizada en investigaciones.

2.4 Conclusiones

Se han presentado tres alternativas de solución las cuales permiten realizar una buena estimación de datos, describiendo brevemente cada una de ellas y revisando sus características principales, para luego en base a ellas determinar las ventajas y desventajas de cada opción. Dos de estas alternativas corresponden a redes de tipo feedforward mientras que una es del tipo recurrente. Si bien aquí se detalla una selección modularizada de las alternativas que pueden ayudar a solucionar el problema, es necesario mencionar que estas pueden no ser excluyentes, esto es, es posible utilizar la combinación de dos soluciones o bien, hacer uso de ambas y comprobar su efectividad por separado.

Con la información aquí presentada, es posible determinar que las redes el tipo NARX poseen la versatilidad necesaria para tratar con series temporales, lo que la hacen la mejor opción entre las señaladas. Su capacidad de utilizar la historia de la señal a tratar, utilizando retardos temporales tanto a la entrada como a la realiemtación de la salida de la red, la hacen única en comparación con las otras dos alternativas y esta es una característica de suma importancia para el problema de estimación que se enfrenta en este trabajo. Esto es, debido a que se está analizando una serie de tiempo en la cual sus respuestas son contínuas (no existen saltos repentinos a magnitudes altas, sin haber pasado por estados intermedios) y periódicas (dado a que pertenecen a señales derivadas de la voz), con lo que conocer su pasado puede evidenciar estados futuros.

Otra característica interesante de las redes de tipo NARX, es que estas conforman una arquitectura más general de red neuronal, dado que sin retardos de ningún tipo estas pueden ser MLP o RBF, o bien, solo con retardos en la entrada se forma el un tipo llamado ”time-delayed”.

Todas estas características de las redes del tipo NARX la hacen una clara elección que promete obtener buenos resultados para el problema que se trata en este trabajo.

Capítulo 3

Datos Recibidos

Una temática importante para el desarrollo del proyecto es considerar la cantidad de datos que se disponen, dado que estos afectan la variedad de modelos que se pueden utilizar, su arquitectura y hasta incluso hacer imposible el uso de redes neuronales.

A continuación, se hace un breve resumen de los datos recibidos para la evaluación del uso de redes neuronales, junto con el detalle de su uso.

3.1 PAE

La secuencia de silabas /pae/ está diseñada para permitir estimaciones indirectas y no invasivas de la presión pulmonar y flujo de aire laríngeo para una vocal sostenida.

De esta secuencia de silabas se dispone el registro del acelerómetro y el flujo aéreo glotal obtenido a partir del flujo oral para 120 pacientes con 3 niveles de sonoridad (despacio, fuerte y medio), en ventanas que poseen 512 muestras.

Es necesario mencionar que este set de datos no posee ningún filtrado a diferencia del set de datos de Rainbow Passage, por lo que para utilizar estas señales será necesario diseñar uno.

3.2 Rainbow Passage

El llamado ”Rainbow Passage” corresponde a un texto fonéticamente balanceado que ha sido frecuentemente utilizado en la invesigación de la voz [26].

De pacientes leyendo algunos párrafos de este texto, se dispone de la grabación ininterrumpida de la señal del acelerómetro filtrada con un filtro pasabajos a aproximadamente $1[kHz]$, junto con su estimación del flujo aéreo glotal de las secciones de la grabación en donde se detectó habla humana.

Esta estimación del flujo aéreo glotal fue obtenida de dos formas:

1. A partir flujo aéreo glotal estimado a partir del flujo oral, el cual fue registrado de forma aproximadamente sincronizada con la señal de acelerómetro. Esta versión del flujo aéreo glotal se obtuvo usando un método aceptado en la investigación, lo cual nos permitiría utilizar como objetivo de la red neuronal.
2. A partir de la señal del acelerómetro, utilizando IBIF (ver apartado 1.2) lo que da espacio para contrastar el resultado de la red neuronal con este acercamiento.

Estas tres señales fueron registradas para 101 pacientes (los cuales 51 tienen voz normal y 50 poseen alguna patología), para estos pacientes, en 95 de ellos se detectaron más de 200 secciones (de 400 muestras) donde se pudo obtener el flujo aéreo glotal (ver figura 3.1).

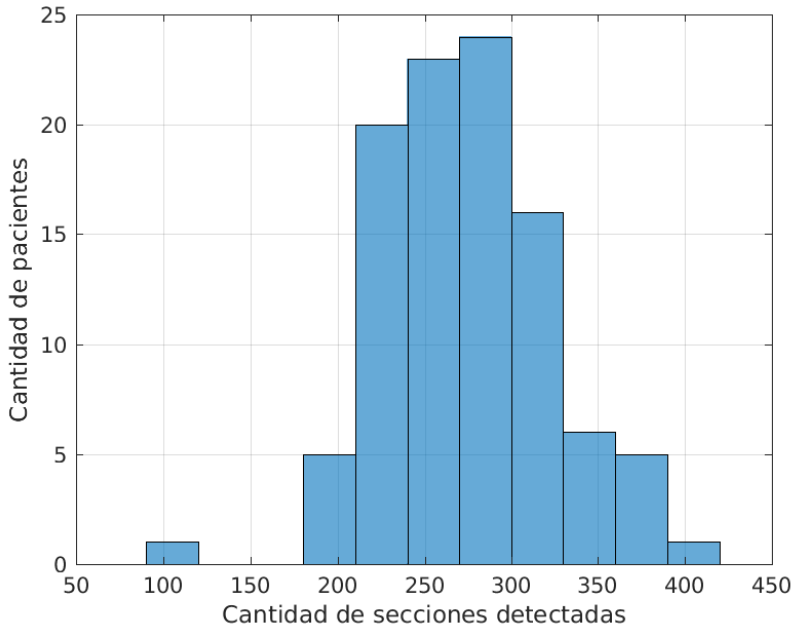


Figura 3.1: Histograma de los datos Rainbow Passage

Cabe señalar que es posible que el problema de sincronización entre los datos del acelerómetro y los del flujo aéreo glotal estimado a partir del flujo oral, dificulte el entrenamiento de la red (es posible que el acelerómetro tenga algunas muestras adelantadas respecto a GVV con lo cual reaccionaria antes, lo que es un comportamiento imposible en la práctica. Este problema se analiza con más detalle en el apartado 5.1.1).

3.3 Uso

A continuación se muestra un resumen de los datos recibidos por cada archivo:

Archivo	Pacientes	Secciones utilizables	Muestras por sección
/pae/	120	318	500
Rainbow Passage	101	27488	400

Cuadro 3.1: Resumen datos

Notar que para el archivo /pae/ se tiene una cantidad menor de segmentos utilizables a la indicada antes, dado que luego de una revisión se encontraron algunas señales vacías y por tanto, sin utilidad para entrenar las redes neuronales.

Si bien se posee una gran cantidad de datos (más de 27000 segmentos entre ambos archivos), es requerido que se realice un entrenamiento diferenciado para cada paciente. Esto es debido a que cada persona tiene características físicas distintas, como un distinto tono de voz, grosor de piel, etc. Lo cual evidentemente crea una restricción sobre el uso de estos.

Las mayores implicancias que este problema tiene, es que al realizar una calibración diferenciada para cada paciente limita la cantidad de datos que se pueden usar para cada entrenamiento, dejando al archivo /pae/ sin utilidad dado que se poseen muy pocos datos como para realizar un entrenamiento apropiado.

Por otra parte los datos del Rainbow Passage tienen una mayor cantidad de señales por paciente (mayor a 200 para la mayoría, ver figura 3.1), lo cual aún pueden ser pocos para entrenar redes neuronales, pero puede ser suficiente si es que éstas poseen las características esenciales del sistema. Sin embargo, dado que se conoce el número del paciente al que corresponde cada señal (en el archivo /pae/ y rainbow passage) estos pueden ser relacionados, lo que permite utilizar los datos del archivo /pae/ para una comprobación extra del desempeño de las redes.

Capítulo 4

Propuesta

A continuación se muestra en detalle la metodología que se utilizará para el desarrollo de este proyecto, mencionado todos los detalles de las configuraciones y cual es la finalidad de la implementación que se describe.

Como guía para el diseño de esta propuesta se ha utilizado el proceso de diseño de redes neuronales representado a continuación.

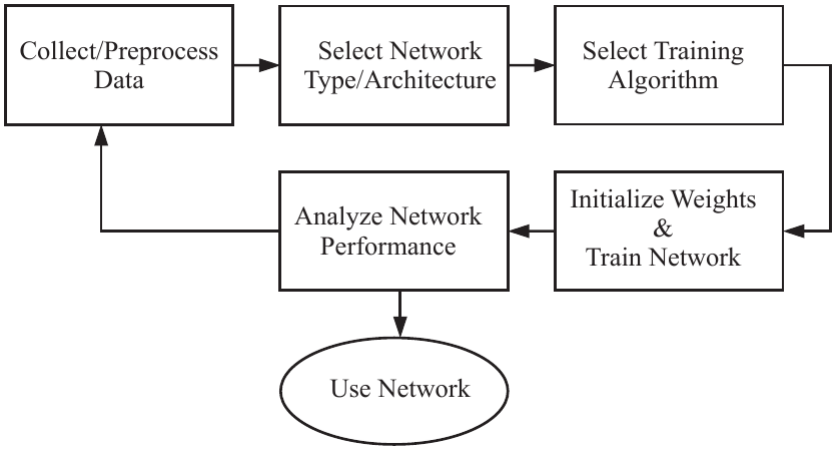


Figura 4.1: Diagrama de flujo para el proceso de diseño de redes neuronales

Como se detalló en el capítulo 3, el comienzo del flujo de la recolección de datos ya ha sido cubierto, por lo que corresponde comenzar el Pre-procesamiento de los datos.

4.1 Pre-procesamiento de los datos

El principal propósito de la etapa de pre-procesamiento de datos es facilitar el entrenamiento de la red. El pre-procesamiento de datos consiste en pasos tales como normalización, transformaciones no lineales, extracción de características, codificación de inputs/targets discretos, manejo de datos faltantes, etc. La idea es realizar un procesamiento preliminar de los datos para facilitar el entrenamiento de las redes neuronales de modo de extraer información relevante.

Es una practica común normalizar las entradas antes de aplicarlas a la red. De esta manera, inicializar los pesos de la red a valores aleatorios pequeños garantiza que el producto de entrada por el peso será pequeño. Además, cuando los valores de entrada están normalizados, las magnitudes de los pesos tienen un significado consistente con la función de activación.

Para esta aplicación, se opta por lo más común que es normalizar las entradas a valores entre -1 y 1. Esto puede ser realizado de la siguiente forma:

$$x_n^{normalizado} = 2\frac{x_n^{original} - x_{\min}}{x_{\max} - x_{\min}} - 1$$

Donde x_{min} y x_{max} corresponden al valor mínimo y máximo respectivamente de todo el se de datos.

Por otra parte, en el apartado 3.2, se mencionó que existe un problema de sincronización entre los datos entre la señal del acelerómetro y el flujo aéreo glotal obtenido a partir del flujo oral. Esto será resuelto utilizando correlación cruzada, en donde determinando el máximo de ésta determinará la diferencia en muestras las señales.

Finalmente, dado que los segmentos de señal pertenecientes al set de datos /pae/ no están filtrados a diferencia de los del set de datos RP, es necesario el diseño de un filtro se comporte de manera similar al filtro utilizado en RP.

Para esto se mira el espectro en varias señales del set de datos de RP y se concluye que este filtro tiene una frecuencia de corte cercana a 1[kHz]. Un ejemplo típico del espectro se puede ver a continuación en la figura 4.2.

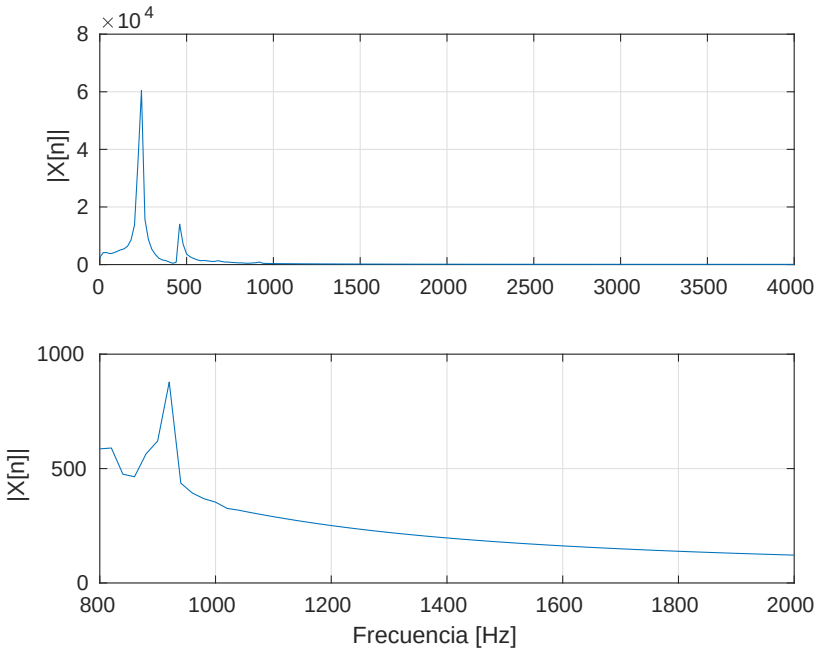


Figura 4.2: Espectro típico en los segmentos de señal del set de datos RP.

4.2 Arquitectura

Antes de definir una arquitectura, es necesario entender el tipo de problema que se trata de resolver, en nuestro caso es una aproximación (o regresión) pero como corresponde a un comportamiento que ocurre en una serie de tiempo, es necesario hacer uso de retardos temporales (que la salida de nuestra red, dependa de muestras anteriores).

Como se detalló en el capítulo 2, existen tres tipos de modelos de redes neuronales que se consideran los más utilizados en la investigación y los cuales por lo general se utilizan como punto de partida para el trabajo en redes neuronales. De aquellos tres, solo el modelo NARX (ver apartado 2.3) presenta características que permitirían el procesamiento de retardos temporales y por ende, posiblemente llegar a mejores resultados.

Por otra parte, existe una variación de las redes NARX que podría ser útil dado que es mucho menos compleja, llamada “time-delay” (retardada temporalmente). Esta variación tiene la única diferencia de que no posee la realimentación que las redes NARX tienen, y por ende, solo realizan cálculos utilizando valores del pasado de la entrada.

Existen muchos otros tipos de modelos de redes neuronales que se pueden utilizar para este tipo de problema, pero los mencionados corresponden a los más simples y útiles.

4.2.1 Selección de detalles específicos

Después de ya elegida la estructura básica de la red, es necesario seleccionar los detalles de la arquitectura (por ejemplo, el número de capas, el número de neuronas, etc.). En algunos casos, la elección de arquitectura básica determinará automáticamente el número de capas, pero en este caso, estos parámetros deben ser escogidos.

El procedimiento estándar es comenzar con una red con una capa oculta. Si el rendimiento de la red de dos capas no es satisfactorio, se puede utilizar una red de tres capas. Sería inusual utilizar más de dos capas ocultas [28].

Se debe considerar que el entrenamiento se vuelve cada vez más difícil a medida que se usan múltiples capas ocultas. Sin embargo, para problemas muy difíciles, se pueden usar las llamadas "Deep Neural Networks", las cuales poseen varias capas ocultas. Típicamente, esto requiere del uso de una GPU para realizar el entrenamiento dentro de un tiempo razonable.

También se debe seleccionar el número de neuronas en cada capa. El número de neuronas en la capa de salida es el mismo que el tamaño del vector de destino (en nuestro caso, una). El número de neuronas en las capas ocultas está determinado por que tan compleja es la función que se está aproximando o los límites de decisión que se están implementando (precisión a la que se desea llegar). Desafortunadamente, por lo general no se sabe cuán complejo es el problema hasta que tratamos de entrenar la red. Una forma de proceder es comenzar con más neuronas de las necesarias y luego utilizar alguna técnica de simplificación o de reducción de overfitting.

La mayor desventaja de utilizar demasiadas neuronas es que la red puede resultar en overfitting. Por otra parte, existen métodos como el llamado "Pruning", en donde se analizan los pesos luego de haber entrenado la red, y los que posea menos relevancia "podarlos".

Finalmente, los últimos parámetros que se deben escoger corresponden a la cantidad de retardos tanto en la entrada como en la realimentación para nuestras redes. El procedimiento que se usa, corresponde a comenzar con una baja cantidad de neuronas y retardos, e ir aumentando, analizando si es que no existen neuronas que "podar" además del impacto en el rendimiento que resultó el agregarlas (en otras palabras, realizar un análisis de sensibilidad).

4.3 Entrenamiento

El entrenamiento de una red neuronal es, en la mayoría de los casos, un ejercicio de optimización numérica de una función objetivo generalmente no lineal. No existe un único mejor método para llevar a cabo esta optimización, es necesario elegir un método basado en las características del problema que se va a resolver.

Para las redes multicapa, generalmente se utiliza algoritmos basados en gradiente o Jacobiano, estos pueden implementarse en modo “batch” o “secuencial” (también conocido como incremental). En la forma secuencial actualizamos los pesos después de cada entrada es procesada por la red. En el modo “batch”, todas las entradas se presentan juntas a la red y el gradiente total se calcula sumando los gradientes para cada entrada, antes de actualizar los pesos.

En algunas situaciones, se prefiere la forma secuencial, por ejemplo, cuando se requiere una operación en tiempo real o adaptable. Sin embargo, muchos de los algoritmos de optimización más eficientes son inherentemente algoritmos batch. Para esta aplicación, dado que los datos serán procesados de forma “offline”, se utiliza un entrenamiento del tipo batch.

En cuanto al algoritmo de entrenamiento, se sabe que para redes que se utilizan para la aproximación de funciones, las cuales posean múltiples capas con hasta unos pocos cientos de pesos y biases, el algoritmo de Levenberg-Marquardt suele ser el método de entrenamiento con la convergencia más rápida [28, 29, 30]. En muchos casos, este algoritmo de entrenamiento es capaz de obtener errores cuadráticos medios más bajos que cualquiera de otros algoritmos.

Cuando el número de pesos llega a mil o más, el algoritmo de Levenberg-Marquardt no es tan eficiente como algunos de los algoritmos de gradiente conjugado. Para grandes redes, el algoritmo Scaled Conjugate Gradient es muy eficiente. Este método también es atractivo para problemas de reconocimiento de patrones, tarea para la cual el algoritmo de Levenberg-Marquardt no posee un buen desempeño. Además este algoritmo es el que requiere mayor cantidad de memoria para ser procesado [29].

Considerando que las desventajas que tiene el algoritmo de Levenberg-Marquardt no aplican en este proyecto, lo hace la mejor opción para realizar el entrenamiento de las redes neuronales.

4.3.1 Función de rendimiento

Para las redes multicapa, el índice de rendimiento estándar es el error cuadrado medio (MSE), el cual se detalla a continuación:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

Donde $\hat{Y}_i$ es un vector de n predicciones, y Y_i corresponde al vector de los valores esperados para la señal.

Mientras el MSE es la función de rendimiento más utilizada, existen otras que se pueden utilizar. Una de ellas es el error absoluto medio (MAE), descrito a continuación:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{Y}_i - Y_i\right|$$

Donde $\hat{Y}_i$ es un vector de n predicciones, y Y_i corresponde al vector de los valores esperados para la señal.

Estas dos funciones son las más utilizadas como función de rendimiento, y la mayor diferencia entre ellas, yace en que el MSE utiliza el error al cuadrado, lo que resulta en que errores más grandes produzcan MSE más grandes (para errores mayor a 1), mientras que para MAE no presenta un aumento significativo en su magnitud, por lo que para esta aplicación es mejor hacer uso del MSE.

4.3.2 Múltiples Entrenamientos

Realizar solo una ejecución del entrenamiento (entrenar solo una red neuronal) puede no producir un rendimiento óptimo, debido a la posibilidad de alcanzar un mínimo local de la superficie de rendimiento. Lo mejor es reiniciar el entrenamiento con varias condiciones iniciales diferentes y seleccionar la red que produzca el mejor rendimiento. De cinco a diez entrenamientos casi siempre producirán un óptimo global [31].

4.4 Análisis del rendimiento

Previamente se ha descrito las características básicas de la red como su arquitectura, entrenamiento, etc. Pero también es necesario definir que tipo de métricas y metodología se utilizarán para dimensionar la eficacia las redes entrenadas.

A continuación se detalla un resumen de la propuesta de resolución para este proyecto.

4.4.1 Procedimiento

Primero, para determinar una arquitectura óptima para la red neuronal, o bien, que presente un buen rendimiento para la aplicación, se realizará un análisis de sensibilidad de las características que podemos modificar de la red: las neuronas, la cantidad de retardos a la entrada y la cantidad de retardos de la realimentación de la salida de la red. Esto se hará manteniendo constante una variable y modificando otra, observando la modificación en el desempeño de la red.

Una vez determinado una arquitectura óptima, se procede a probarlo con distintos pacientes del set de datos de Rainbow passage, de modo de poder dar una visión más generalizada del rendimiento que el modelo encontrado posee, junto con verificar si es posible realizar buenas estimaciones utilizando redes neuronales.

También se comprobará el rendimiento de las redes poniéndolas a prueba con las 3 señales por paciente disponibles en el set de datos /pae/. Dado que estos son la estimación más certera del flujo aéreo glotal, lo cual nos dará datos de que tan fiables son las redes entrenadas.

En cuanto a la metodología de entrenamiento, se utiliza un 90% escogido aleatoriamente de la totalidad de los datos para realizar el entrenamiento y un 10% para realizar las pruebas para cada red entrenada (el 10% será común para todas las redes), esto con la finalidad de comparar todas las redes con un mismo set de datos que no ha sido visto antes por las redes entrenadas. De el 90% antes mencionado, se utilizará un 70% como set de entrenamiento, 15% de validación y un 15% de prueba, todos escogidos aleatoriamente, lo cual se hace internamente configurando la red en Matlab. Esto permite asegurar que todas las redes posean entrenamientos con algunas variaciones con el fin de encontrar alguna que presente mayor generalidad.

4.4.2 Evaluación

Luego de hechos los respectivos entrenamientos señalados, se procede a calcular el rendimiento para éstas.

Para realizar el análisis de sensibilidad, se utilizará el error cuadrático medio, dado que nos provee una visión general del rendimiento y nos permite realizar comparaciones entre las mismas redes. Luego, para realizar las comparaciones con los resultados de IBIF, se utilizará el RMSE normalizado, dado que esta métrica nos permite verificar que tan desviada se encuentra nuestra predicción en un determinado segmento de señal.

Como existe un problema de sincronización ya mencionado antes, antes de calcular el error entre GVV IBIF y GVV OVV, estas deben ser sincronizadas de la misma forma que los datos antes de ser usados por las redes neuronales, dado que una ligero desfase puede provocar un aumento mayor en el MSE, y por tanto, también en el RMSE normalizado.

Capítulo 5

Implementación

A continuación se muestran detalles más específicos respecto a la implementación, como funciones, códigos y otros que se utilizaron. El código completo se puede ver en detalle en el anexo A.

Como framework, se usa el Toolbox de redes neuronales de Matlab [27] dado que es ampliamente utilizado en la investigación como punto de partida para el uso de redes neuronales en distintas aplicaciones.

5.1 Pre-procesamiento

Para el pre-procesamiento de los datos se usa el código que muestra en el apartado A.1, en este se pueden ver cada una de las etapas que se explican a continuación.

5.1.1 Sincronización de señales

Previo a entrenar las redes propuestas, se creó una función para sincronizar las señales del acelerómetro con las de el flujo aéreo glotal pertenecientes al archivo Rainbow Passage, esta se muestra en el apartado A.5. Esta función hace uso de la correlación cruzada para obtener el retardo entre las señales, detectando el desfase en muestras desde el centro al máximo de la correlación.

El efecto de esta función se puede en la figura 5.1, donde se ha normalizado la señal del acelerómetro y el GVV por sus respectivos máximos, solo para poder apreciar el efecto de sincronizar las señales utilizando la función creada.

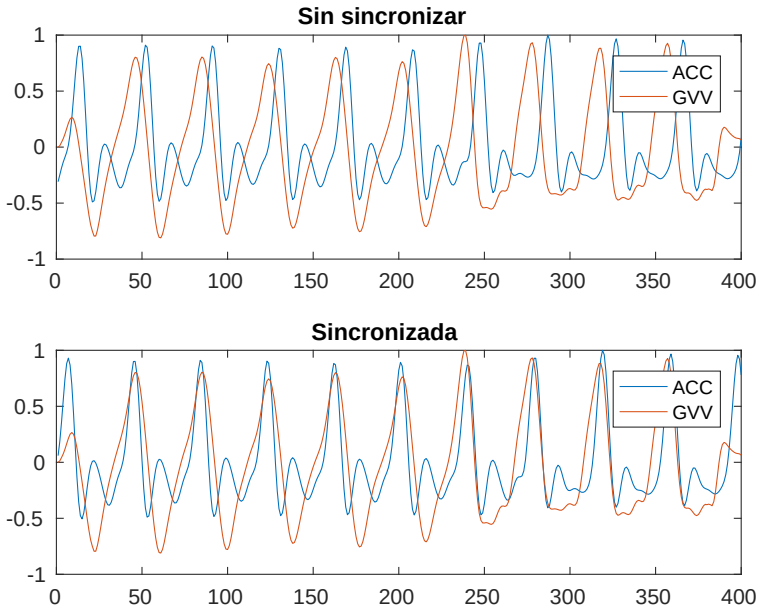


Figura 5.1: Efecto de la función sincw

5.1.2 Diseño de filtro

Como se señalo en el apartado 4.1, es necesario crear un filtro que tenga una frecuencia de corte cercana a 1[kHz] resguardando que no exista demasiada distorsión de fase. Para esto se utilizó el código que se muestra a continuación:

```
Fs  = 8e3;  %Frecuencia de Muestreo
Fp  = 800;  %Frecuencia de borde de la banda de paso
Fst = 1100; %Frecuencia a la cual comienza el filtrado con atenuación "
    Ast" definida
Ap  = 0.01; %Ripple permitido en la banda de paso
Ast = 80;   %Atenuación mínima permitida en la banda filtrada

d = fdesign.lowpass('Fp,Fst,Ap,Ast',Fp,Fst,Ap,Ast,Fs)
LP_filt = design(d,'equiripple');

fvtool(LP_filt)
```

La primera parte corresponde a la especificación de características y requisitos del filtro, las cuales fueron elegidas de acuerdo a lo requerido en esta aplicación. Existen varias formas de especificar los requerimientos del filtro, estas pueden ser listadas con:

```
d = fdesign.lowpass;
set (d, 'specification')
```

A continuación se hace uso de la función "`fdesign.lowpass`" la cual crea un objeto de clase "fdesign.lowpass" el cual compacta todas las especificaciones, para entregárselas a la función generadora del filtro "`design`". La función "`design`" se encarga de crear

un filtro el cual cumpla con las especificaciones requeridas. Existen varios algoritmos los cuales pueden ser usados, en este caso como se quiere evitar distorsión de fase se busca un algoritmo que implemente un filtro de tipo FIR, estos se pueden listar con el comando "`designmethods`":

```
>>designmethods(d,'FIR')

d =

  lowpass with properties:

                Response: 'Lowpass'
           Specification: 'Fp,Fst,Ap,Ast'
             Description: {4×1 cell}
    NormalizedFrequency: 0
                     Fs: 8000
                  Fpass: 800
                  Fstop: 1100
                  Apass: 0.0100
                  Astop: 80

FIR Design Methods for class fdesign.lowpass (Fp,Fst,Ap,Ast):

equiripple
ifir
kaiserwin
multistage
```

Dado que también es deseable que no exista demasiado ripple en la banda de paso se utiliza el algoritmo "equiripple".

Finalmente se puede ver la respuesta en frecuencia del filtro utilizando la función `fvtool`. El filtro diseñado se puede ver a continuación en las figuras 5.2 y 5.3.

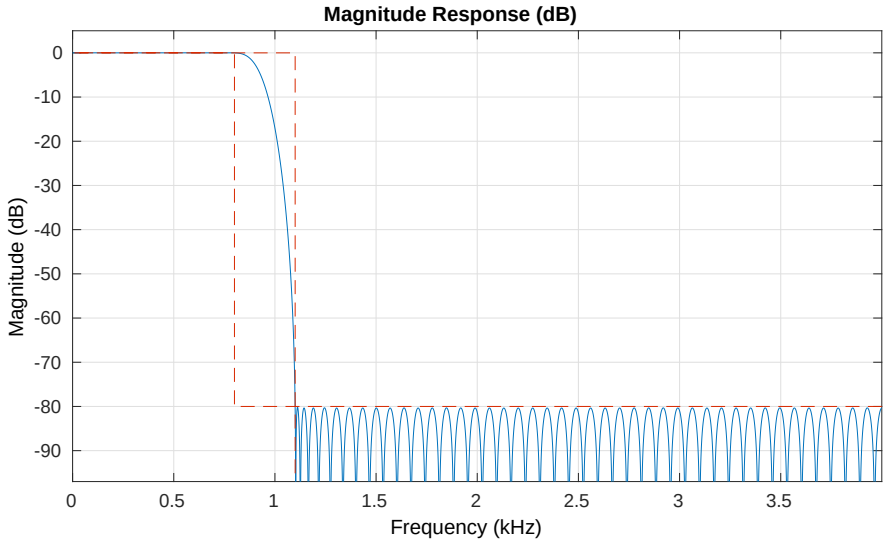


Figura 5.2: Magnitud de la respuesta en frecuencia de filtro diseñado

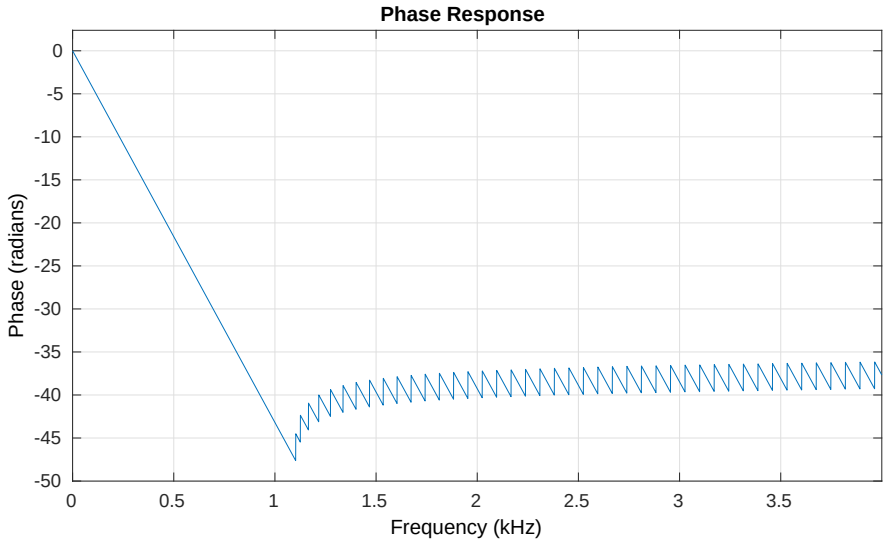


Figura 5.3: Fase de la respuesta en frecuencia de filtro diseñado

Como se puede ver, el filtro cumple con todas las especificaciones requerida. Para aplicarlo solo se debe hacer uso de la función "`filter`", por ejemplo:

```
filter(LP_filt, x)
```

5.1.3 Separación y encapsulación de datos

Además de sincronizar los datos, se debe separar el dataset en partes de 90% y 10% como se explicó en el apartado 4.4.1, junto con formatear el dataset para usar las funciónes de entrenamiento de Matlab, la programación hecha con este objetivo se muestra dentro del archivo `processData` en el apartado A.1. En esta función la división de las secciones de señal es hecha aleatoriamente y además se da el formato que corresponde para el procesamiento dentro de las redes neuronales.

Además de separar el dataset, se debe entregar todos los segmentos de señal simultáneamente para el procesamiento de la red neuronal (para entrenar en modo batch), por esto, se utiliza la función `catsamples`, la cual se encarga de concatenar las muestras utilizadas para entrenar la red, de modo que esta reciba y procese todas las señales en formas simultánea.

Finalmente, dado que las redes poseen retardos temporales, es necesario de condiciones iniciales para hacer cálculos de valores presentes, este seccionamiento y preparación es realizado con la función `preparets`. Esta función prepara las entradas y objetivos de una serie de tiempo para realizar la simulación y/o entrenamiento. El uso de esta función es particularmente útil para los tipos de redes que se aquí se usan, dado que permite extraer de forma genérica y fácil la cantidad de muestras que formarán parte de las condiciones iniciales de una red (las muestras correspondientes a los retardos utilizados).

5.2 Entrenamiento

A continuación se describe las funciones utilizadas para la creación y entrenamiento de las redes neuronales, el código completo se detalla en la apartado A.2

Antes de realizar entrenamiento alguno, es necesario crear una red neuronal. Esto se realiza con el uso de las funciones especializadas `timedelayednet` y `narxnet`, las cuales crean de forma rápida redes del tipo timedelayed y NARX con las configuraciones más usadas.

A continuación se muestra un ejemplo de su uso:

```
net = timedelaynet(0:I,N);
net = narxnet(0:I,1:F,N);
```

Donde “I” representa la cantidad de retardos a la entrada, “F” la cantidad de retardos de la rama de realimentación y “N” la cantidad de neuronas de la red.

Un ejemplo visual de lo que estas funciones generan se aprecia en las figuras 5.4 y 5.5

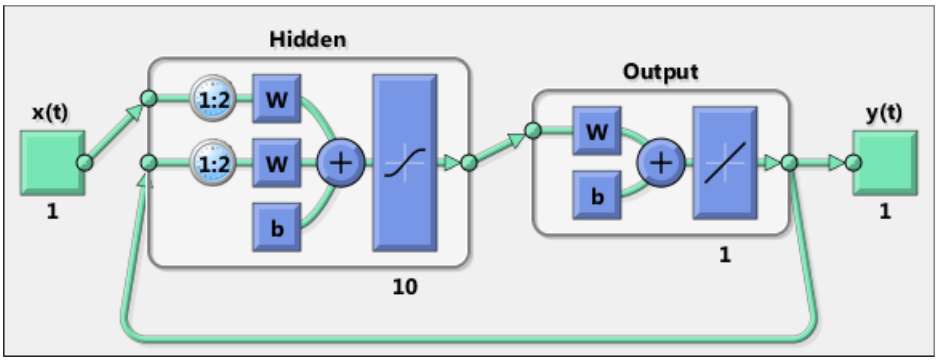


Figura 5.4: Ejemplo de red NARX generada por Matlab.

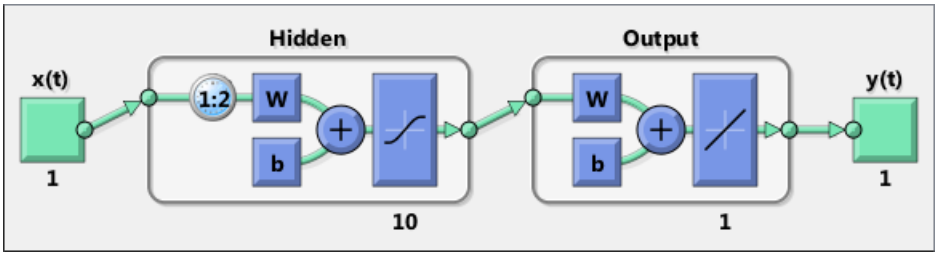


Figura 5.5: Ejemplo de red Timedelayed generada por Matlab.

La estructura básica de la red neuronal (un objeto clase ”network´´), con los objetos y subobjetos creados a partir de estas funciones, definen como esta será entrenada y ejecutada al momento de ser llamada, junto con todas sus configuraciones. Un ejemplo de su estructura se muestra continuación:

```
net =
    Neural Network
              name: 'NARX Neural Network'
          userdata: (your custom info)
    dimensions:
         numInputs: 2
         numLayers: 2
        numOutputs: 1
    numInputDelays: 2
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 10
        sampleTime: 1
    connections:
       biasConnect: [1; 1]
      inputConnect: [1 1; 0 0]
      layerConnect: [0 0; 1 0]
     outputConnect: [0 1]
    subobjects:
             output: Equivalent to outputs{2}
```

```
            inputs: {2x1 cell array of 2 inputs}
            layers: {2x1 cell array of 2 layers}
           outputs: {1x2 cell array of 1 output}
            biases: {2x1 cell array of 2 biases}
      inputWeights: {2x2 cell array of 2 weights}
      layerWeights: {2x2 cell array of 1 weight}
    functions:
          adaptFcn: 'adaptwb'
       adaptParam: (none)
         derivFcn: 'defaultderiv'
        divideFcn: 'dividerand'
      divideParam: .trainRatio, .valRatio, .testRatio
       divideMode: 'time'
          initFcn: 'initlay'
        performFcn: 'mse'
      performParam: .regularization, .normalization
          plotFcns: {'plotperform', plottrainstate, ploterrhist,
                     plotregression, plotresponse, ploterrcorr,
                     plotinerrcorr}
        plotParams: {1x7 cell array of 7 params}
          trainFcn: 'trainlm'
        trainParam: .showWindow, .showCommandLine, .show, .epochs,
                    .time, .goal, .min_grad, .max_fail, .mu, .mu_dec,
                    .mu_inc, .mu_max
    weight and bias values:
                IW: {2x2 cell} containing 2 input weight matrices
                LW: {2x2 cell} containing 1 layer weight matrix
                 b: {2x1 cell} containing 2 bias vectors
    methods:
             adapt: Learn while in continuous use
         configure: Configure inputs & outputs
            gensim: Generate Simulink model
              init: Initialize weights & biases
           perform: Calculate performance
               sim: Evaluate network outputs given inputs
             train: Train network with examples
              view: View diagram
       unconfigure: Unconfigure inputs & outputs
    evaluate:       [outputs,inputStates] = net(inputs,inputStates)
```

Por esto, luego de haber creado las redes con las funciones antes mencionadas, éstas deben ser configuradas para que estas posean las características que se desean. Es necesario aclarar que muchos de los parámetros que se muestran en la estructura básica no son útiles en el caso de este problema, por lo que a continuación se mostraran solo los parámetros relevantes para este trabajo los cuales fueron configurados.

5.2.1 Normalización entrada

Primero en cuanto a la normalización de la entrada, la creación de redes neuronales con las funciones mencionadas ya posee por defecto la normalización a la entrada. Esto se puede ver en los atributos de "Inputs".

El subobjeto "Inputs" es un conjunto de celdas las cuales representan las entradas a la red (en este caso una), aquí se puede configurar algunos parámetros de pre-procesamiento de los datos, como por ejemplo la normalización. A continuación se muestra un ejemplo del subobjeto "Input":

```
>> net.inputs{1}
ans =
    Neural Network Input
              name: 'x'
    feedbackOutput: []
       processFcns: {'mapminmax'}
     processParams: {1x1 cell array of 1 param}
   processSettings: {1x1 cell array of 1 setting}
     processedRange: [1x2 double]
      processedSize: 1
              range: [1x2 double]
               size: 1
          userdata: (your custom info)
```

Si bien, estos son configurados automáticamente al ejecutar el entrenamiento de la red neuronal, existe un par de parámetros que definen el como se hará esta configuración automática, estos son:

```
net.inputs{1}.processFcns
net.inputs{1}.processParams
```

En esos atributos se hace uso de la función `mapminmax` con `ymin=-1` y `ymax=1` lo que normaliza la entrada a los valores máximos y minimos del conjunto de señales utilizadas para entrenar la red neuronal.

De forma similar ocurre a la salida, cuando ocurre el proceso de desnormalización.

5.2.2 División de secciones de señal

También es necesario configurar de que forma se utilizaran los datos que se le entregarán a la red neuronal, por esto es necesario mirar los atributos de `divideFcn` y `divideParam`.

```
net.divideFcn
net.divideParam
```

Esta configuración corresponde a la forma interna en que Matlab dividirá el set de datos que se le entrega. Para la selección aleatoria de secciones de señal que se usaran para el entrenamiento, validación y prueba de la red, el atributo `divideFcn` debe estar asignado

en `dividerand` para que cada red entrenada use un conjunto distinto de secciones de señal para el entrenamiento, validación y pruebas. Junto con su respectiva división de datos que se hace modificando el parámetro `divideParam` señalando los porcentajes que se darán a cada; el 70% para entrenamiento 15% para validación y 15% para pruebas, como se menciono en el apartado 4.4.1. Para asignar estos porcentajes, se debe configurar el atributo `divideParam`.

Cabe mencionar que estas configuraciones son asignadas por defecto cuando se crea una red neuronal, por lo que el código del entrenamiento no muestra ninguna de estas fases.

5.2.3 Algoritmo de entrenamiento

Para seleccionar la función de entrenamiento deseada, se debe mirar al atributo `trainFcn`

```
net.trainFcn
```

Como se explicó en el apartado 4.3, el algoritmo de entrenamiento a utilizar es el de Levenberg-Marquardt, para esto se debe configurar el atributo de `trainFcn` a la función `trainlm`. Esto también forma parte de los valores por defecto al crear redes neuronales con los comandos `timedelayednet` y `narxnet`.

5.2.4 Función de Rendimiento

Para hacer uso como función de rendimiento el MSE, se debe configurar el atributo `performFcn`.

```
net.performFcn
```

Este debe especificar el uso del MSE con el valor “mse”, lo que también forma parte de las características por defecto al crear una red.

5.2.5 Entrenar

Finalmente, luego de configurar el comportamiento de la red, es necesario entrenarla, utilizando el comando `train`. Este comando no tiene muchas configuraciones, se le entrega la red a entrenar, los datos de entrada y salida, junto con las condiciones iniciales. Por ejemplo:

```
train(net,Xs,Ts, Xi, 'useParallel','yes')
```

La configuración interesante que se puede hacer, es la forma en que va a entrenar la red. En el caso del ejemplo, la directiva “useParallel” indica que al momento de realizar el entrenamiento utilice al menos dos núcleos (dependiendo de las configuraciones), lo cual puede mejorar la rapidez con que se realiza esta tarea. También es posible utilizar una tarjeta de video para realizar aún más rápido el entrenamiento, pero no se dispuso de una que fuera compatible con el toolbox al momento de desarrollar este proyecto.

5.3 Post-Procesamiento

A continuación, luego de que se ha entrenado la red y como post-procesamiento se realiza un análisis del desempeño de éstas ya sea para comparar su eficacia con IBIF o para realizar el análisis de sensibilidad por cantidad de neuronas o retardos.

Primero, para hacer el análisis de sensibilidad, se compara su desempeño utilizando la función `mse` perteneciente al toolbox de redes neuronales. Esta función permite obtener el MSE de forma rápida, entregándole como argumentos ambas señales a comparar.

Luego, para realizar la comparación de alguna red con IBIF se utiliza la función `normError` expuesta en el apartado A.4. Esta función es la misma que se usó para determinar el rendimiento al aplicar IBIF sobre las secciones de la señal del acelerómetro en etapas previas de esta investigación, por lo que es una buena forma de comprar en los mismos términos.

Esta función hace uso del RMS descrito a continuación:

$$x_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}$$

La función `normError` calcula una versión normalizada del RMS del error, la cual se calcula de la siguiente manera:

$$nRMSE = \frac{RMS(Y_{Objetivo} - Y_{Estimacion})}{RMS(Y_{Objetivo})}$$

Por otra parte, la señal objetivo se recorta desde el 25% hasta el 75% del largo de la señal objetivo de modo que se quitar los posibles transientes producto del filtrado inverso con el que se obtuvo GVV, y luego se comprara con la estimación buscando el mínimo (similar a realizar autocorrelación). Un ejemplo de cómo esta función realiza las comparación se puede ver en la figura 5.6.

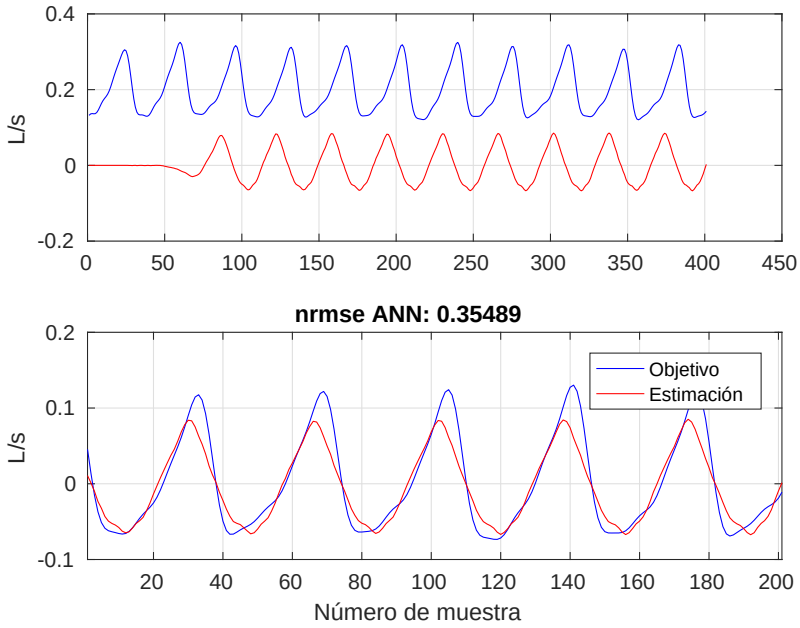


Figura 5.6: Ejemplo del ajuste hecho para el cálculo del nRMS.

En la figura 5.6, en la parte superior, se aprecia las salidas en bruto de la ANN (señal roja) y el objetivo (señal azul). Aplicando la función `normError` se obtiene el nRMS comparando las señales que hay abajo, donde se puede ver las señals sincronizadas y sin offset (se le restó el valor medio, que fue hecho fuera de la función). Notar que la gráfica superior tiene 400 muestras mientras que la inferior 200, debido al recorte entre el 25% y el 75% del largo de la señal objetivo.

Capítulo 6

Resultados

A continuación se muestran todos los resultados obtenidos a lo largo de este trabajo. Todas las gráficas generadas se encuentran en el anexo B.

6.1 Determinando un modelo óptimo

Como primera tarea se propuso elegir un modelo óptimo en cuanto a cantidad de neuronas y retardos temporales, por lo que se hizo un análisis de sensibilidad para dichos parámetros.

6.1.1 Cantidad de Retardos a la entrada

El primer parámetro que se analizó fue la cantidad de retardos a la entrada. En la figura 6.1 se muestra el avance típico del MSE a medida que se aumenta la cantidad de retardos de entrada a la red en una red de tipo timedelayed. Como se puede ver, aumentar al doble la cantidad de neuronas no posee un efecto demasiado significativo en la disminución del MSE (al rededor de 0.1×10^{-3}). En comparación con el aumento en el número de retardos, el cual presenta una significativa mejora hasta que se llegan a los 15 retardos a la entrada, donde su efecto llega a un estado en que no se mejora demasiado al aumentar la cantidad de retardos a la entrada.

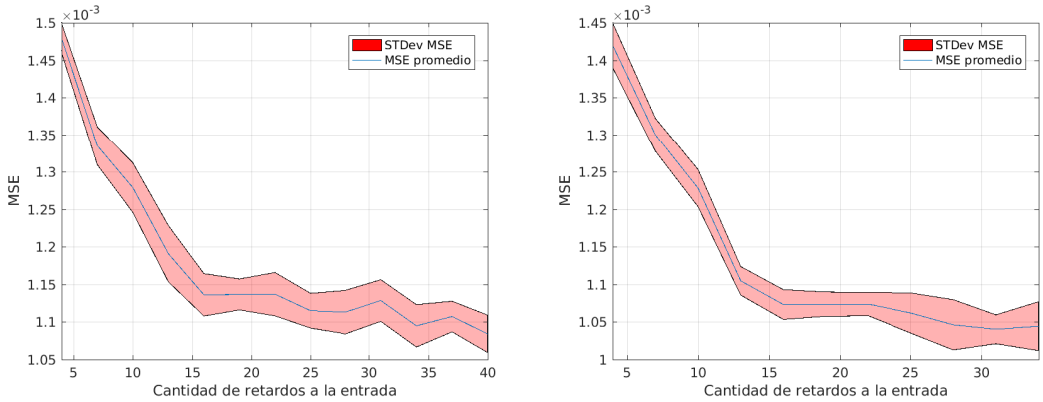


Figura 6.1: Comparación del avance típico del mse, a la izquierda con 15 neuronas a la derecha con 30.

Este experimento se realizó únicamente con 2 personas (con redes de 15 y 30 neuronas) y observando lo obtenido con este experimento, se puede ver que aproximadamente en 15 retardos a la entrada se encuentra el óptimo para este paciente. No obstante, considerando que se requiere de crear una arquitectura genérica, se concluyó que una cantidad de 25 retardos es una buena elección dado que el óptimo para distintos pacientes se verá desplazado, por lo que se debe usar una cantidad que supere el óptimo para todos (aunque no tan grande para evitar overfitting).

6.1.2 Cantidad de neuronas

A continuación se realiza el análisis de sensibilidad para la cantidad de neuronas que se utilizarán en la red.

Basado en la conclusión anterior, se utilizan 25 retardos a la entrada para crear redes del tipo timedelayed a las que se les modifica la cantidad de neuronas, determinando su rendimiento. El resultado de este experimento se puede ver en la figura 6.2.

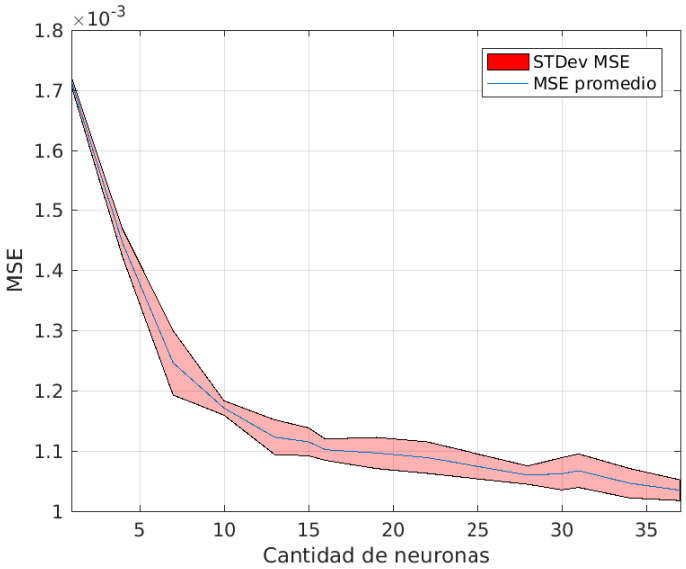


Figura 6.2: Movimiento del MSE a medida que aumentan las Neuronas.

Este experimento fue hecho solo con una persona, dado que se requirió de una gran cantidad de tiempo para llevarlo a cabo. Sin embargo, se puede notar una tendencia similar a la obtenida al analizar los retardos a la entrada, donde el error disminuye hasta llegar a un punto óptimo (en este caso aproximadamente con 15 neuronas). Haciendo el mismo análisis anterior, se concluye que una buena elección serian 22, debido a que el óptimo va a cambiar por persona y se debe escoger una arquitectura lo más generalizable posible.

6.1.3 Retardos en la realimentación

Por último, se analizó el parametro de retardos en la realimentación. Para esto se creo una red de tipo NARX con los parámetros óptimos obtenidos previamente (25 retardos a la entrada y 22 neuronas) y se modificó la cantidad de retados en la realiementación.

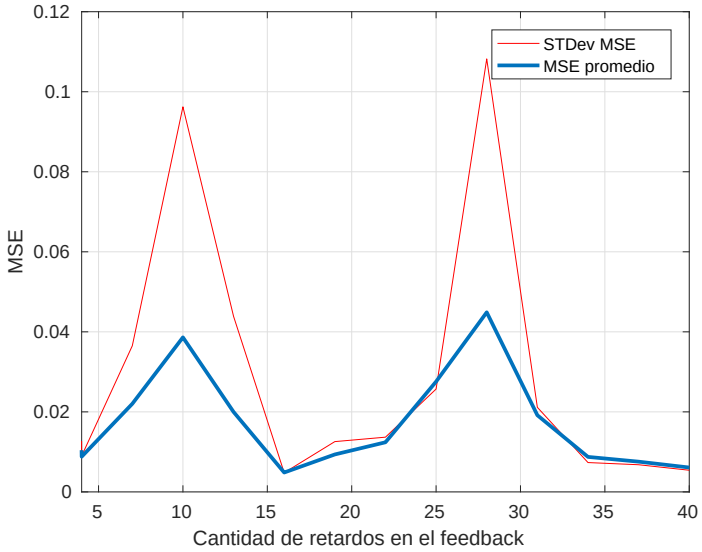


Figura 6.3: Movimiento del MSE a medida que aumentan los retardos en la realimentación.

Como se aprecia en la figura 6.3 utilzar retardos en la realimentación de la red neuronal no provee ninguna mejora evidente, de hecho entorpece y hace mas lento el entrenamiento sin mejorar de forma clara como lo hicieron los parámetros anteriores. Considerando que los retardos en la realimentación solo aumentaron el error en vez de disminuirlo, lo más lógico sería no utilizarlos. De este modo la arquitectura óptima lograda sería una red del tipo timedelayed con 25 retardos en la entrada y 22 neuronas.

6.2 Desempeño de la red diseñada

Con en análisis anterior, se determinó que la mejor arquitectura de red para este problema en particular corresponde a una red del tipo timedelayed con 25 retardos en la entrada y 22 neuronas. Para esta red, se muestran a continuación los resultados del desempeño para los distintos set de datos disponibles de modo de evidenciar su comportamiento para esta aplicación.

6.2.1 Rainbow Passage

Este set de datos corresponde al mismo con el que se entrenó la red neuronal, y por esto, se espera que tengan mucho mejores resultados en comparación al otro set de PAE. Para probar el desempeño con este set de datos, se observa el resultado de esta red con el 10% del set de señales que se separó en un comienzo y por lo que nunca ha procesado la red.

Se pueden apreciar algunas características interesantes en cuanto a la forma que toman las estimaciones.

Se tiene que en algunos casos el uso de redes neuronales se muestra muy superior al uso de otros métodos de estimación (IBIF en este caso). En la figura 6.4 se puede ver uno de estos casos, donde el uso de redes neuronales proporciona una estimación fiel de lo que corresponde al flujo aéreo glotal.

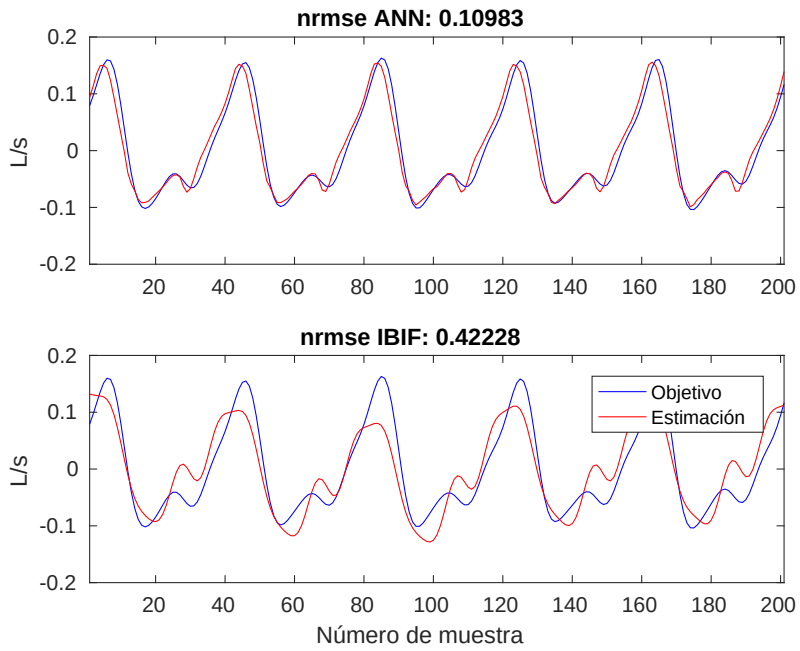


Figura 6.4: Comparación de la estimación del GVV con un buen resultado para las ANN.

También existen casos en los que las redes neuronales no parecen ajustar de buena forma la amplitud de la señal resultante, de modo que logran una mala estimación. Esto se muestra en la figura 6.5.

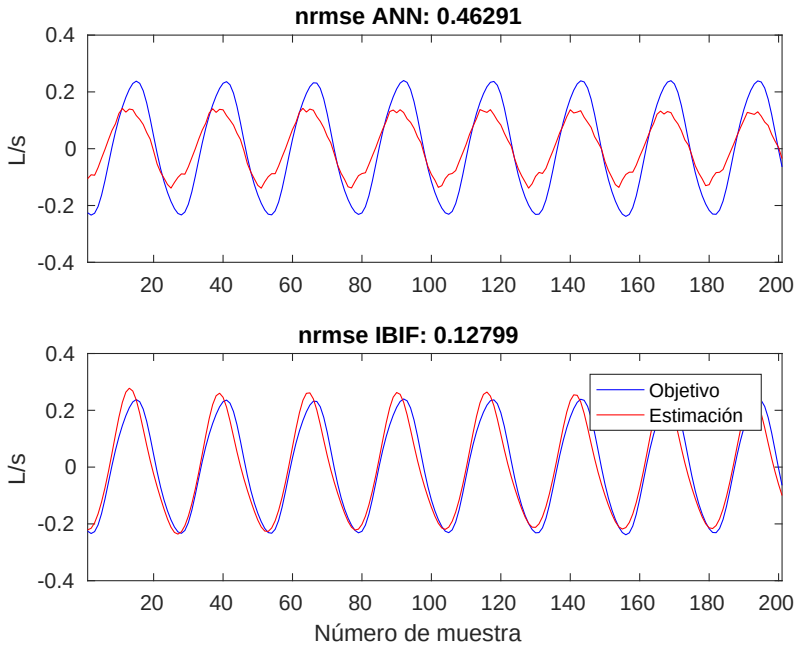


Figura 6.5: Comparación de la estimación del GVV no tan exacta para las redes neuronales.

Por otra parte, cabe mencionar que existen casos en que las señales presentan desafíos más difíciles de resolver, el cual es el caso 6.6. En la imagen, se muestra una señal que inicia con amplitud reducida y luego aumenta llegando a una señal similar a las anteriores. Como se puede ver, el uso de redes neuronales logró excelentes resultados.

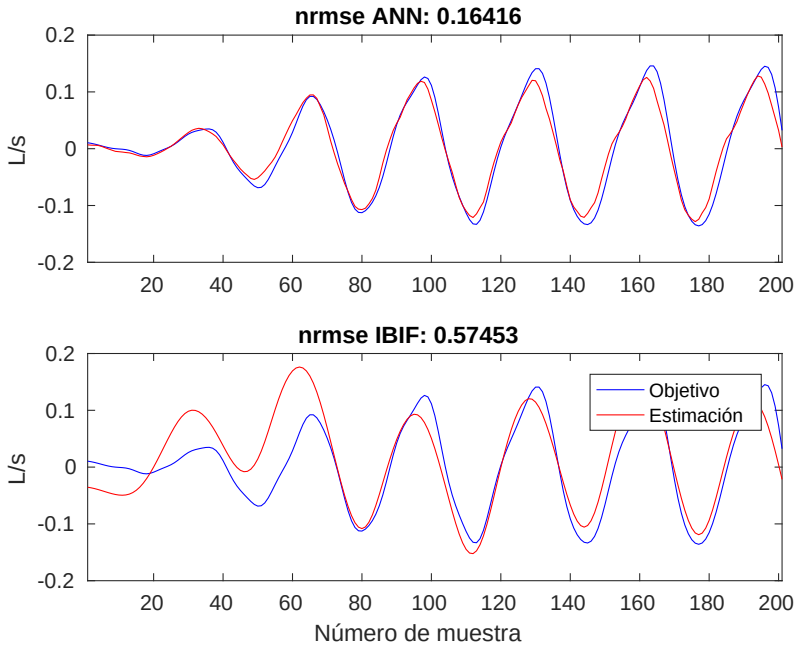


Figura 6.6: Comparación de la estimación del GVV para señales con comportamientos distintos.

Algunas señales de muestra se encuentran en el apartado B.4.

Finalmente, para ver un resumen del error presente en las señales de prueba para una determinada red neuronal, en la figura 6.7 se presenta la distribución típica del error para una persona.

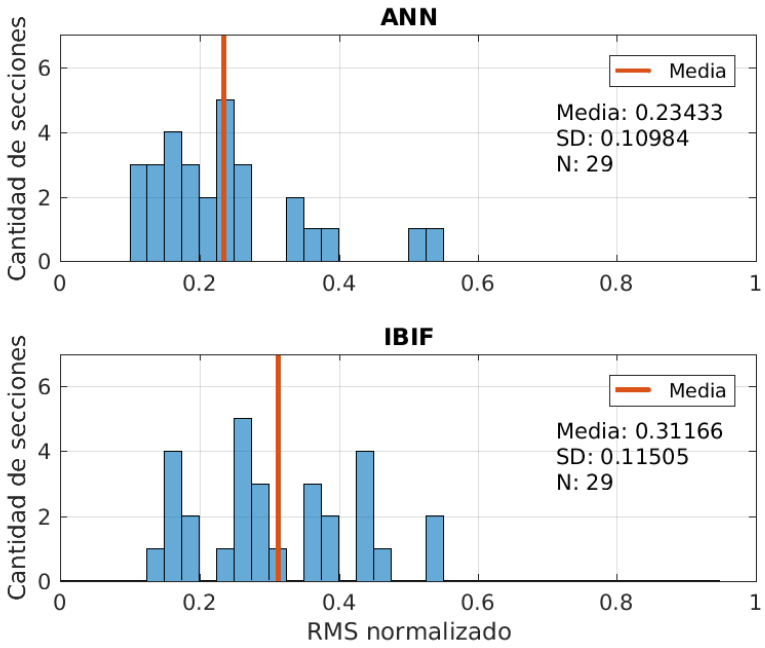


Figura 6.7: Histograma típico del error para una persona.

En la figura 6.7 se puede ver una tendencia de la red a tener menor porcentaje de error, pero para obtener una visión más general del desempeño de este modelo, se ejecuta este procedimiento con 30 distintos pacientes. La recopilación de los resultados obtenidos se muestran a continuación:

Paciente	N	Media ANN	SD ANN	Media IBIF	SD IBIF	t-Value	p
79	42	0.29573	0.10225	0.27269	0.13926	0.86396	0.39013
22	37	0.19517	0.068557	0.27987	0.11217	-3.919	0.00020048
54	37	0.19542	0.065444	0.20156	0.061552	-0.41558	0.67895
5	37	0.30491	0.10022	0.35827	0.12094	-2.0663	0.042394
64	36	0.22536	0.070452	0.16586	0.045223	4.2645	6.1563e-05
47	35	0.17627	0.059899	0.36865	0.18827	-5.7607	2.2153e-07
99	35	0.17546	0.066874	0.26862	0.11004	-4.2801	5.9945e-05
7	33	0.22906	0.060508	0.36341	0.15066	-4.7535	1.1743e-05
92	33	0.20832	0.054526	0.32995	0.11291	-5.5722	5.3798e-07
38	33	0.25796	0.084743	0.27614	0.11859	-0.71673	0.47615
9	32	0.21261	0.068538	0.27821	0.18616	-1.8705	0.066134
39	32	0.3188	0.21895	0.35364	0.31591	-0.51278	0.60993
4	32	0.25105	0.097496	0.3183	0.15826	-2.0467	0.044934
48	31	0.19444	0.052283	0.54856	0.31146	-6.243	4.8081e-08
43	31	0.30375	0.10617	0.25094	0.14963	1.6024	0.11431
19	30	0.21693	0.066429	0.3275	0.19736	-2.908	0.0051464
37	30	0.25068	0.069297	0.31308	0.16897	-1.8717	0.066294
100	29	0.23433	0.10984	0.31166	0.11505	-2.6179	0.011358
77	27	0.24863	0.081815	0.29116	0.052991	-2.2668	0.027591
20	25	0.2403	0.076924	0.32303	0.11506	-2.9885	0.0044096
33	25	0.24292	0.083364	0.32731	0.098047	-3.2787	0.0019443
94	25	0.16795	0.044505	0.31884	0.11365	-6.1815	1.3236e-07
71	24	0.18614	0.070937	0.25967	0.13591	-2.3497	0.023136
6	24	0.26545	0.056924	0.31085	0.11752	-1.7031	0.095305
61	24	0.27348	0.082167	0.26827	0.10473	0.19177	0.84877
63	23	0.26385	0.1139	0.34902	0.12587	-2.4064	0.020376
1	23	0.31483	0.11852	0.33791	0.14579	-0.58901	0.55886
101	23	0.17008	0.045128	0.36294	0.12916	-6.7603	2.57e-08
49	22	0.28717	0.16603	0.72321	0.58605	-3.3577	0.0016792
21	19	0.2496	0.07151	0.40518	0.2402	-2.7059	0.010345

Cuadro 6.1: Resumen resultados

Observando la tabla, se puede ver que existen solo 4 casos en los que el uso de IBIF supera a las redes neuronales (en los pacientes 79, 64, 43 y 61). Además en 9 pacientes (79, 54, 38, 9, 39, 43, 37, 6, 61 y 1) el valor p es mayor a 5% y por tanto no se puede decir que los resultados sen estadísticamente distintas.

Dado que es difícil concluir algo a partir de el resumen por paciente, se ha juntando cada uno de sus resultados (el error normalizado de cada una de las señales analizadas), y se ha graficado un único histograma. Este se puede ver en la figura 6.8.

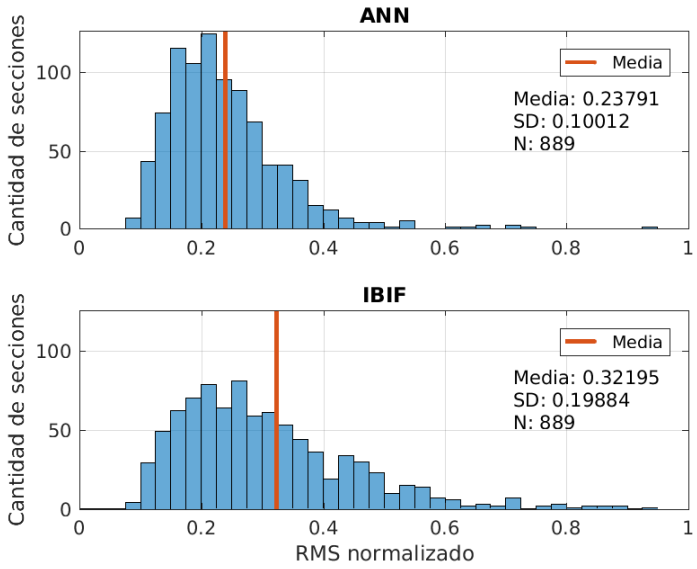


Figura 6.8: Histograma general de rendimiento.

Como se puede ver, el histograma para las redes neuronales tiene una clara tendencia a tener valores menores de nRMSE. Esto puede ser evidenciado con las estadísticas derivadas de estos datos, que se muestran a continuación:

Media ANN	SD ANN	Media IBIF	SD IBIF	t-Value	p
0.23791	0.10012	0.32195	0.19884	-11.255	1.957e-28

Comparando las medias, existe una clara mejora haciendo uso de redes neuronales. Además también los datos presentan menor dispersión dado que la desviación estándar es también mucho menor.

Por otra parte, para poder concluir que estas distribuciones son estadísticamente distintas se le ha hecho un t-test, en el cual consideraremos como valor seguro p de 0.05 (5%) corregido por la correción de bonferroni, descrita a continuación.

$$p \leq \frac{\alpha}{m}$$

Por tanto, el valor p debe ser menor a $\frac{0.05}{889} = 5.6243 \times 10^{-5}$, lo cual se cumple sin problemas.

6.2.2 PAE

A continuación, se pone a prueba el desempeño de las redes neuronales generadas con el segundo set de datos. Este set, contiene señales un mucho más distintas a las con que se

entrenaron la red, con distintos niveles de sonoridad (mas fuertes, medio y más despacio). Por esto, que un buen desempeño en este set de datos implicaría una buena generalización del modelo elegido.

Algo importante que se notó al usar este set de datos, es que tan suceptible es la red neuronal al ruido. En este caso como los datos no se habían filtrado como en RP, las estimaciones fueron muy inferiores, como se muestra a continuación:

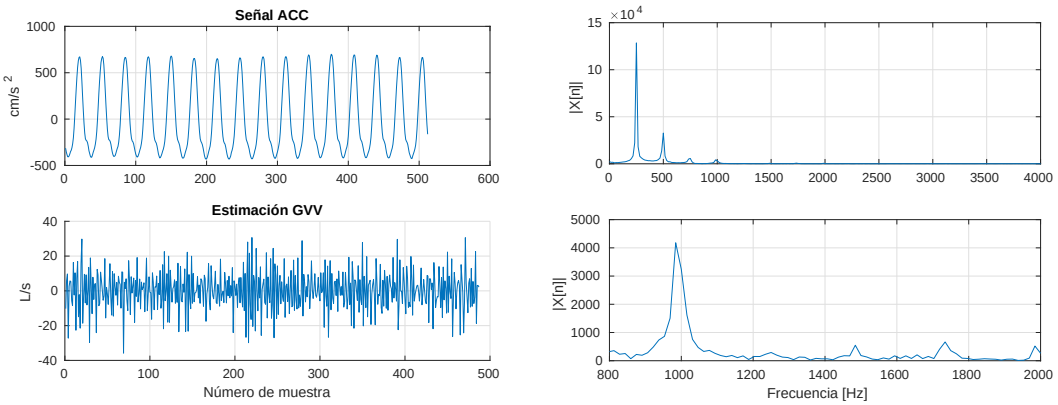


Figura 6.9: A la izquierda la señal del ACC antes de ser filtrada y su estimación usando ANN, a la derecha el espectro de la señal ACC.

En la figura 6.9 se puede observar que la estimación presenta un ruido muy alto, y si se observa en el lado derecho de la imagen el espectro de la señal de accelerómetro utilizada, se pueden ver algunos peaks luego de 1[kHz], que eventualmente se filtran, mejorando notablemente la estimación.

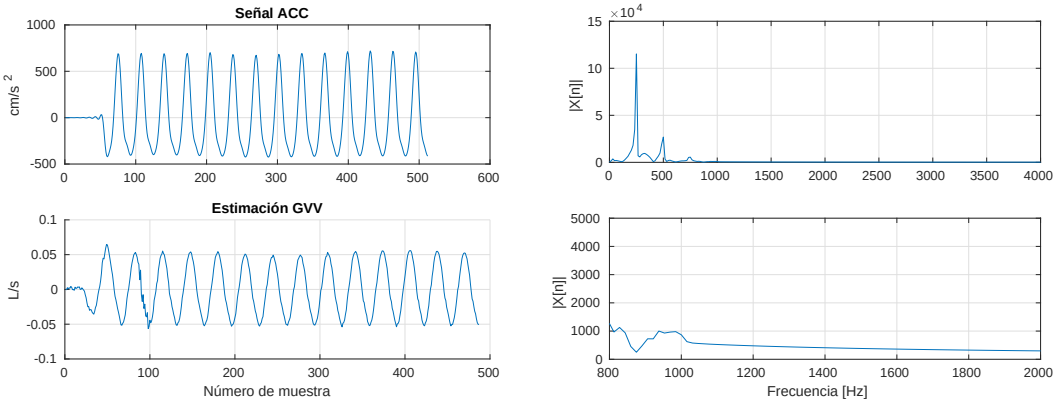


Figura 6.10: A la izquierda la señal del ACC luego de ser filtrada y su estimación usando ANN, a la derecha el espectro de la señal ACC.

Como se puede ver en la figura 6.10 el filtro diseñado en apartado 5.1.2 cumple con su objetivo y permite realizar una estimación mucho más certera. También se puede ver a la

derecha que el espectro también ha cambiado, disminuyendo las frecuencias cercanas a 1[kHz] y prácticamente eliminando todo lo que pase más allá.

Además de filtrar la entrada, también se filtró la salida de la red neuronal con el mismo filtro, y en base a eso se ha llegado los resultados que siguen.

Al igual que con el set de datos anteriores, existen casos de todo tipo. En varios casos se tiene que la estimación es bastante precisa, como se muestra a continuación.

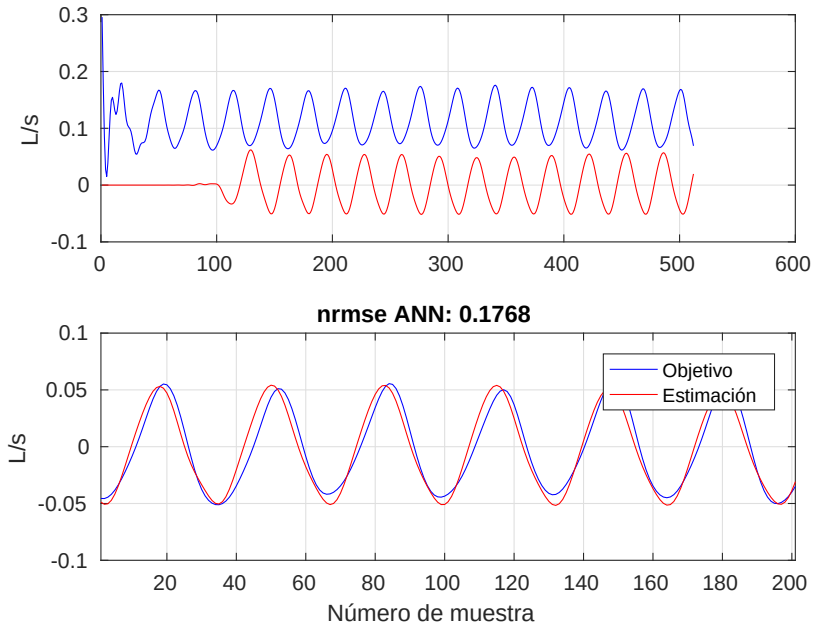


Figura 6.11: Buena estimación, set de datos PAE.

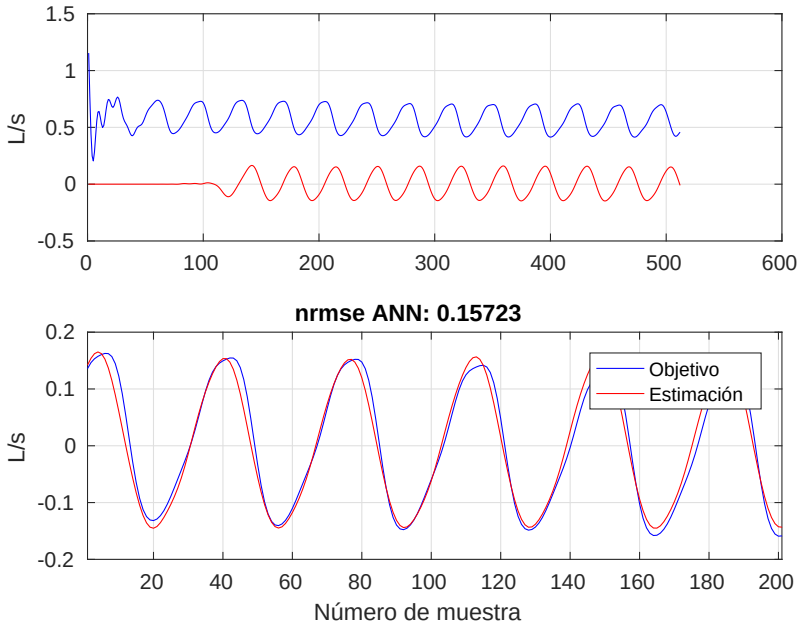


Figura 6.12: Buena estimación, set de datos PAE.

Como se ve en las figuras 6.11 y 6.12, se pudo lograr una buena estimación. En estas imágenes, como en todas las que siguen a continuación, en la parte superior se muestra la estimación sin ningún tipo de sincronización, mientras que abajo se puede ver las señales sincronizadas y sin offset. Se puede ver que el resultado de la red neuronal no posee ningún offset, esto se debe a que el set de datos con el que se entrenó la red (RP) tampoco tenía el offset que se puede ver en el objetivo de pae.

Además como también existen casos buenos, también hay casos en que las redes neuronales no pudieron dar una estimación certera:

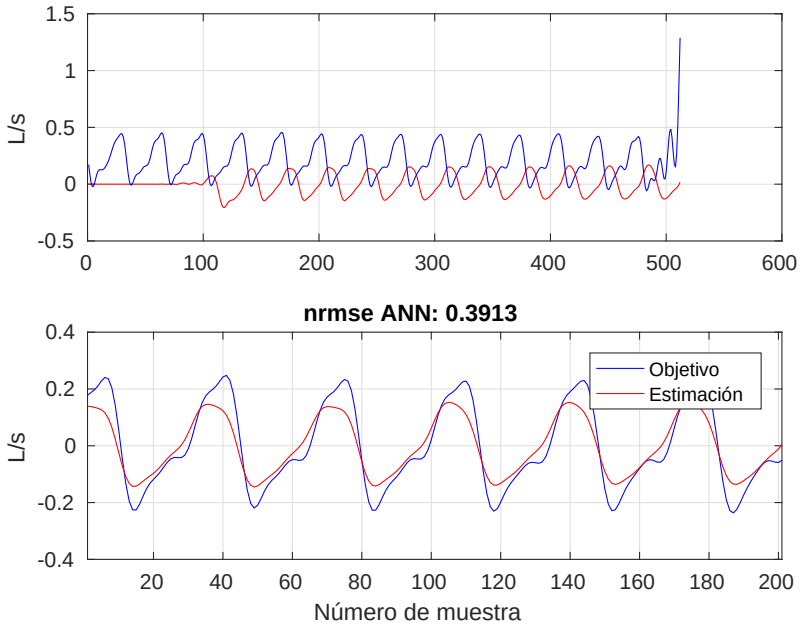


Figura 6.13: Mala estimación, set de datos PAE.

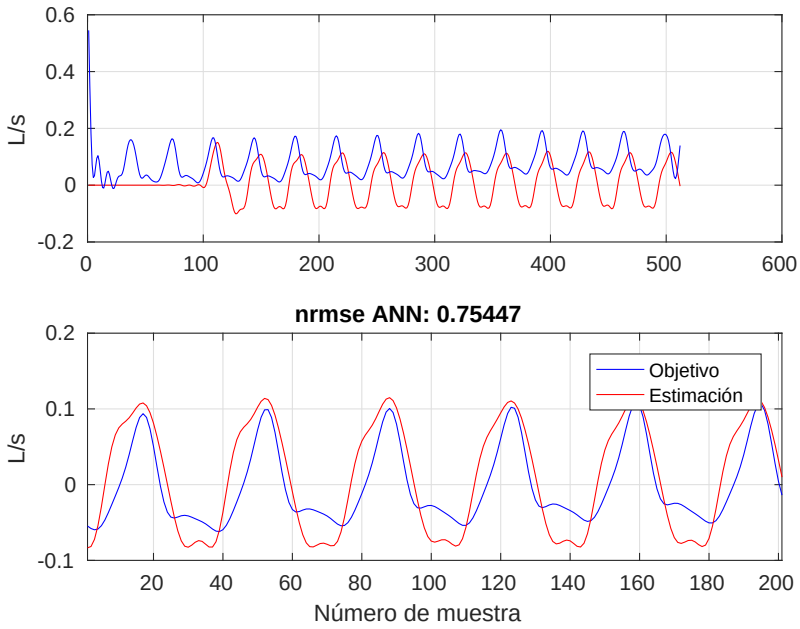


Figura 6.14: Mala estimación, set de datos PAE.

En las figuras 6.13 y 6.14 se puede ver como las ANN no entregaron buenos resultados para algunos casos. Los errores muy grandes se deben probablemente a que las señales que se analizan a que poseen rasgos diferentes a los del set con que se entrenó. Ya sea por que tienen rango de amplitudes distintas debido a los distintos niveles de sonoridad, o por el gesto vocal (nuevo para la red).

También existen casos en los que el error es elevado, pero muy problablemente debido a que el objetivo de la estimación se encontraba erróneo o con una tendencia muy distinta al común, lo cual es el caso de la figura 6.15.

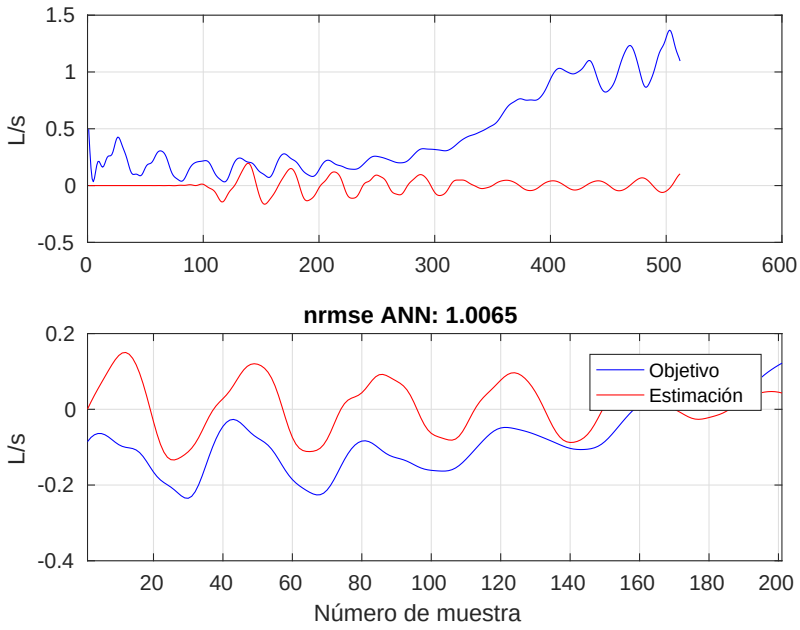


Figura 6.15: Mal Objetivo, set de datos PAE.

En general, si bien un error de 20%-30% pareciera ser alto, mirando a la forma de onda pareciera que no es una estimación tan mala. En las figuras figuras 6.16 y 6.17, se puede ver este hecho, en donde señales con errores altos poseen características muy similares a la forma de onda del objetivo.

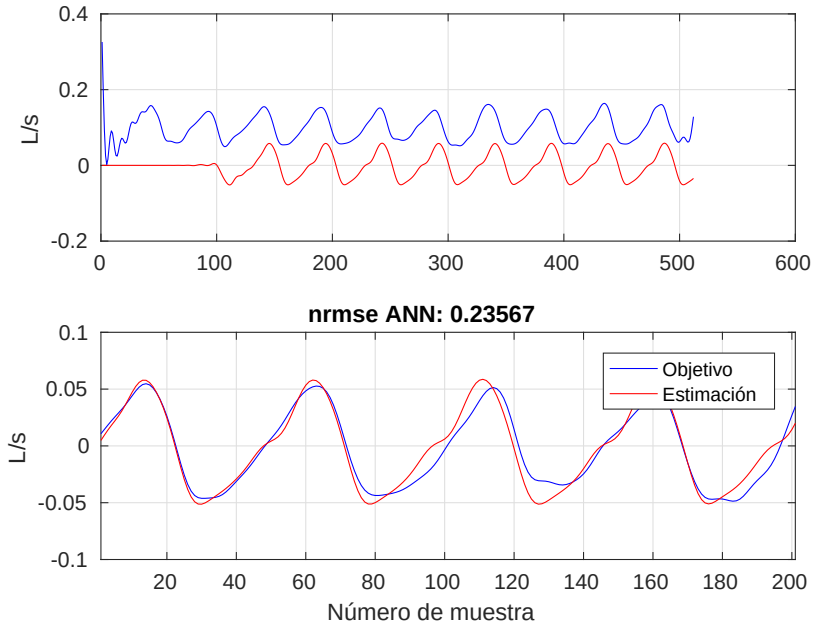


Figura 6.16: Estimación aceptable, set de datos PAE.

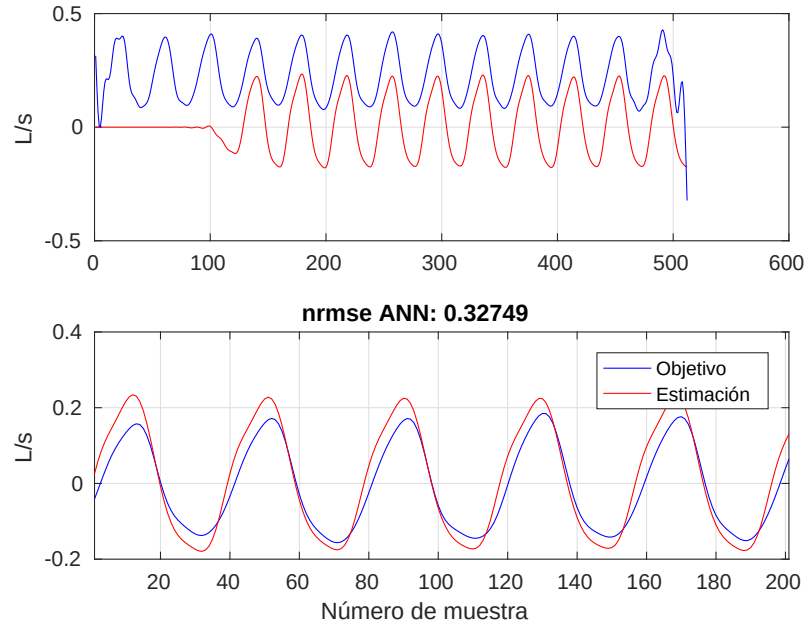


Figura 6.17: Estimación aceptable, set de datos PAE.

Por último una observación interesante se presenta en las figuras 6.18 y 6.19,en donde pareciera ser que las señales estimadas están invertidas respecto al objetivo. Eso puede ser interesante dado que cablear los sensores al revés pueden dar formas de ondas invertidas, lo cuál agregaría error al entrenamiento de las redes neuronales y dificultaría también la

medición del desempeño en este caso. Sin embargo, este hecho solo se cree que puede ser así y no se tiene forma inmediata de verificar.

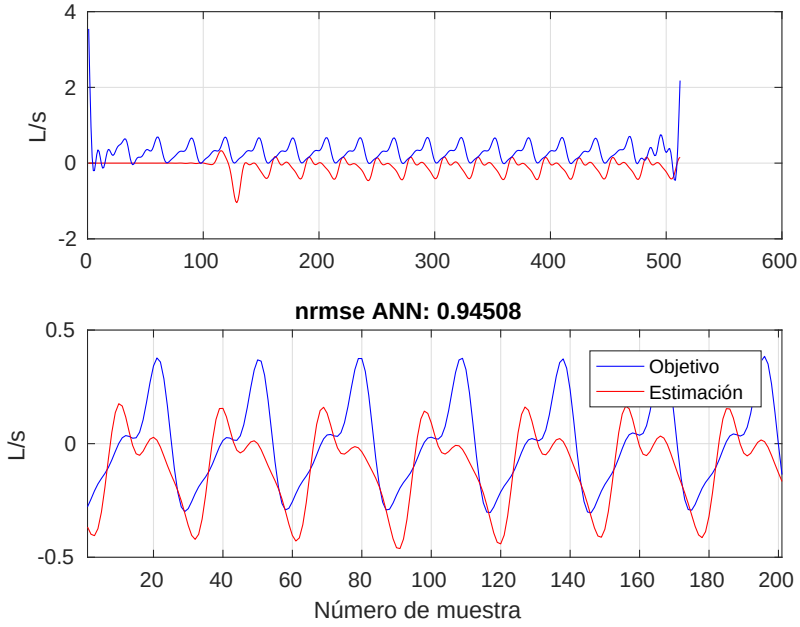


Figura 6.18: Estimación con fase invertida, set de datos PAE.

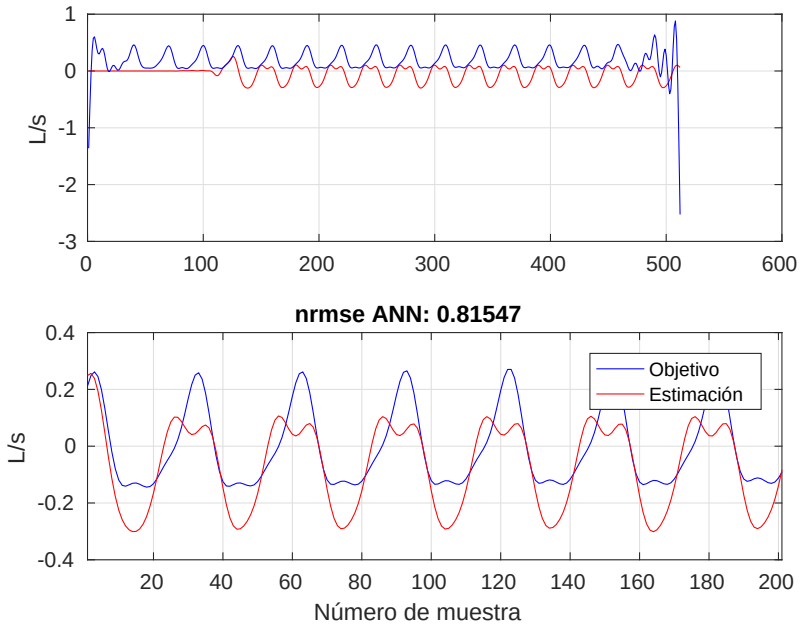


Figura 6.19: Estimación con fase invertida, set de datos PAE.

Algunas señales de muestra se encuentran en el apartado B.5.

A continuación se muestra una tabla resumen para los 27 pacientes analizados, con los errores para sus tres señales disponibles en el set de datos pae.

Paciente	Señal 1	Señal 2	Señal 3
79	0.48761	0.81547	0.48071
22	0.34298	0.39728	0.38628
54	0.21444	0.44331	0.32749
5	0.23469	0.40588	0.42802
64	0.24076	0.94508	0.31562
47	0.45527	0.47334	0.1768
99	0.32397	0.70897	0.28706
7	0.30734	0.34421	0.3371
92	0.44657	0.59873	0.39252
38	0.37723	0.49187	0.35698
9	0.39827	0.42065	0.35504
39	0.43446	0.71653	0.39376
4	0.35839	0.66417	0.30705
43	0.75447	0.66722	0.47917
19	0.2031	0.38429	0.391
37	0.24645	0.2399	0.28077
100	0.19874	0.35373	0.1422
77	0.31214	0.3913	0.24403
20	0.3833	0.35227	0.32484
33	0.30987	0.50172	0.25993
94	0.13731	0.38564	0.23342
71	0.11056	1.0065	0.15723
6	0.25829	0.90547	0.19641
61	0.50247	0.31762	0.57549
1	0.43551	0.60387	0.23567
101	0.22292	0.47682	0.25669
21	0.53813	0.50441	0.39408

Cuadro 6.2: Resumen resultados para set de datos PAE

Por último, y de manera similar a con el set de datos RP, se crea un histograma el cual compila todos estos resultados, con el fin de observar la tendencia general del modelo de red neuronal elegido en este set de datos.

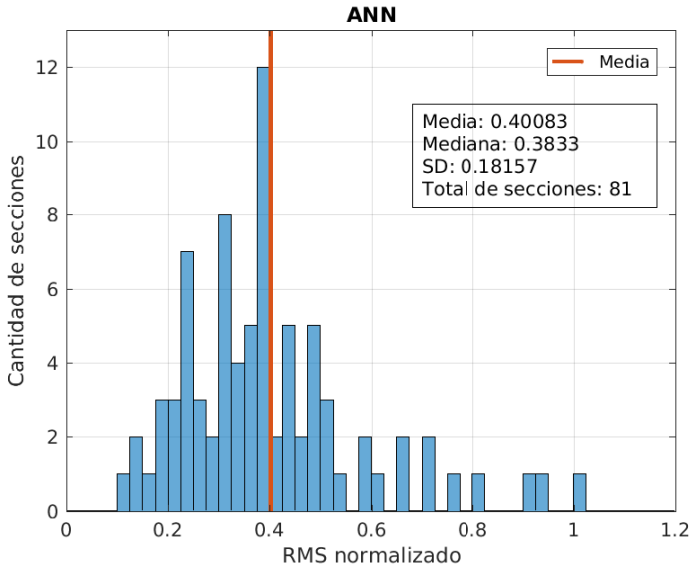


Figura 6.20: Histograma general de rendimiento para set de datos PAE.

Como se observa en el histograma, la media de 40% está aún lejos de IBIF en el set de datos RP de 32.2%, pero considerando que se entrenó con datos cuya estimación es poco precisa deja la posibilidad de que la estimación mejore mucho más a medida que se mejore la calidad los datos utilizados.

Capítulo 7

Conclusiones

Las redes del tipo timedelayed se han demostrado muy útiles y con un rendimiento bastante aceptable comparado con IBIF. Observando los resultados obtenidos para el set de datos PAE, se puede decir que las redes tienen un gran potencial y es muy posible que se pueda mejorar su desempeño.

El error promedio de 40% obtenido con el set de datos PAE no está tan alejado de lo obtenido con IBIF en el set de datos RP de 32.2%, lo cual para una primera aproximación utilizando redes neuronales pareciera ser un resultado prometedor.

Existe una gran diferencia del desempeño obtenido entre algunos pacientes, es posible que esto se deba al mismo problema de la cantidad de datos y su calidad, dado que existe una cantidad muy baja de datos para cada paciente (aproximadamente 270 secciones de señal en promedio) y la selección con la que se entrena y prueba es totalmente aleatoria.

En cuanto a las redes de tipo NARX, el lazo de realimentación no presento mucha mejora en cuanto a la estimación, es más la empeoró. El bajo rendimiento de este tipo de redes al aumentar la cantidad de retardos en la realimentación hizo que se descartaran este modelo en etapas tempranas de pruebas, pero es posible que combinaciones que no se probaron provean mejores resultados. Se podría intentar, por ejemplo, no usar retardos a la entrada y la realimentación de la salida más espaciada (no usar las muestras inmediatamente siguientes).

Finalmente, con los resultados obtenidos en este trabajo, es posible decir que las redes neuronales son totalmente útiles para esta aplicación. Basta con considerar que el error promedio llegó a 40% entrenando con datos limitados y en los cuales no se posee mucha confianza en la estimación, además de probarlos sobre segmentos en los que en varias ocasiones se presentan casos distintos y posiblemente no entrenados, o bien, con objetivos erróneos. El error puede parecer alto, pero en cuanto a forma de onda, poseen una gran similitud.

Por esto, como continuación de este trabajo podrían realizarse mejoras en la calidad de los datos usados para entrenar la red, esto sin duda mejoraría la calidad de las estimaciones, o

bien obtener una mayor cantidad de datos por paciente.

También probar otras arquitecturas o algoritmos para buscar la red mas óptima para este problema, podría tener una incidencia interesante y quizás una mejora muy considerable.

Todo este proyecto evidencia el potencial de las redes neuronales en esta aplicación. Si bien quizás no se llegaron a resultados tan notablemente buenos, establecen un punto de partida de que tan buena es esta técnica en este problema en particular.

Apéndice A

Código

A.1 Generación de Dataset

El siguiente código de utilizó para generar los datasets que se utilizan posteriormente para entrenar las redes neuronales.

```
format long
rng(1) % Para obetner resultados distintos, commentar

load ./DATASETS_V2/Multiple_Subjects_Q_Frame_RP_phono_pol1.mat

clearvars -except dataRP

%Cuadros que se usaron para el analisis
frames = dataRP.gvv.voicedFrame;

%Archivo completo del acelerometro
acc_raw = dataRP.acc.rawdata;

%Resultados GVV IBIF
gvv_IBIF = dataRP.acc.gvv_acc;

%GVV obtenifo por OVV
gvv_OVV = dataRP.flow.gvv_flow;

%RMS normalizado obtenido entre el GVV de IBIF y OVV
nrms_IBIF = dataRP.acc.nrms;

Win = 40;
for N=1:size(frames,2)
    count=1;
    for ii =frames{N}

        idx= (ii-1)*400+1;
        x = acc_raw{N}(idx-Win:idx+399+Win);
```

```
30
31          t=gvv_OVV{N}(count,:);
32
33          [xa,ta, D] = syncw(x,t, 2*Win);
34
35          acc_delay{N}(count,1) = D-Win;
36          acc_syncronized{N}(count,:) = xa;
37          acc_framed{N}(count,:) = acc_raw{N}(idx:idx+399);
38
39          count=count+1;
40      end
41  end
42
43  rows2remove = [];
44  Porcentaje_TEST = 0.90;
45
46  for i = 1:numel(acc_syncronized)
47      disp(['i = ', num2str(i)])
48      x = acc_syncronized{i};
49      t = gvv_OVV{i};
50
51      [ X_1, T_1, X_2, T_2, x_PS, t_PS, rev ,ind1, ind2] = processData(x,
        t,rows2remove,Porcentaje_TEST);
52
53      data{i}.X_1 = X_1;
54      data{i}.T_1 = T_1;
55      data{i}.X_2 = X_2;
56      data{i}.T_2 = T_2;
57      data{i}.x_PS = x_PS;
58      data{i}.t_PS = t_PS;
59      data{i}.rev  = rev;
60      data{i}.ind1  = ind1;
61      data{i}.ind2  = ind2;
62  end
63  %Se guarda el Dataset para usar el NN
64  save('./DATASETS/processed_RP_syncronized.mat', 'data')
65
66
67  clearvars data
68  %Guardar datos framed
69  for i = 1:numel(acc_framed)
70      disp(['i = ', num2str(i)])
71      x = acc_framed{i};
72      t = gvv_IBIF{i};
73
74      data_framed{i}.acc_framed = acc_framed{i};
75      data_framed{i}.gvv_IBIF = gvv_IBIF{i};
76      data_framed{i}.gvv_OVV = gvv_OVV{i};
77      data_framed{i}.nrms_IBIF = nrms_IBIF{i};
78  end
79  %Se guarda el Dataset para usar el NN
80  save('./DATASETS/processed_RP_framed.mat', 'data_framed')
```

```
function [ X_1, T_1, X_2, T_2, x_PS, t_PS , Version, ind1, ind2 ] =
    processData( x, t, rows2remove, Porcentaje_TEST )

Version = 5;

%Dejar todos las señales en el vector columna
x(:,:,rows2remove) = [];
x = permute(x,[1 3 2]);
x = reshape(x,[],size(x,3),1);

t(:,:,rows2remove) = [];
t = permute(t,[1 3 2]);
t = reshape(t,[],size(t,3),1);

%Verificar que no existan constantes (por lo general vacias)
[ ~,x_PS] = removeconstantrows(x);
[ ~,t_PS] = removeconstantrows(t);

%Si es que exsiten vacías quitarlas de todas las señales
if(isempty(t_PS.remove)~= 1 || isempty(t_PS.remove) ~= 1)
    disp('Warning: some rows are empty, check PS')

    dif_row=[];
    rows = [x_PS.remove t_PS.remove];
    for i = rows
        dif_row= [dif_row rows(1)];
        rows = rows(rows ~= rows(1));
        if(isempty(rows)); break; end
    end

    disp(dif_row)
    for i = 0:size(dif_row,2)-1
        disp(['Deleting row: ' num2str(dif_row(i+1))])
       x(dif_row(i+1)-i,:)=[];
       t(dif_row(i+1)-i,:)=[];
    end

end

for i=1:size(x,1)
    eval(sprintf('T%d = num2cell(t(%d,:)); X%d = num2cell(x(%d,:));', i
    ,i,i,i));
end

%Separar los datasets, uno para entrenar las redes, otro para probar y
%comparar rendimiento entre las redes
Q=size(x,1);
Q1 = floor(Q*Porcentaje_TEST);
Q2 = Q-Q1;

ind = randperm(Q);
ind1 = ind(1:Q1);
```

```
ind2 = ind(Q1+(1:Q2));


sx='X = catsamples(';
st='T = catsamples(';
for i = ind1
   eval(sprintf('sx = strcat(sx,''X%s,''); st = strcat(st,''T%s,'');',
    num2str(i), num2str(i)))
end
sx= strcat(sx,''',''pad'');');
st= strcat(st,''',''pad'');');

eval(sx); eval(st); %se genera X y T

X_1 = X;
T_1 = T;

%hacer catsample del conjunto2
% [sx,st] = gen_str(ind2);

sx='X = catsamples(';
st='T = catsamples(';
for i = ind2
   eval(sprintf('sx = strcat(sx,''X%s,''); st = strcat(st,''T%s,'');',
    num2str(i), num2str(i)))
end
sx= strcat(sx,''',''pad'');');
st= strcat(st,''',''pad'');');

eval(sx); eval(st); %se genera X y T

X_2 = X;
T_2 = T;

end
```

A.2 Generación y entrenamiento de redes NARX

A continuación se muestra el código utilizado para entrenar redes el tipo NARX.

```
clear; clc; close all;
format long

load ./DATASETS/processed_RP_syncronized.mat

N_Sujeto = 49;

X_1 = data{N_Sujeto}.X_1;
T_1 = data{N_Sujeto}.T_1;
```

```
X_2 = data{N_Sujeto}.X_2;
T_2 = data{N_Sujeto}.T_2;
x_PS = data{N_Sujeto}.x_PS;
t_PS = data{N_Sujeto}.t_PS;
rev = data{N_Sujeto}.rev;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rango_entradas = 19;
rango_neuronas = 10;
rango_feedback = 1:3:40;
num_exp = 19;              %Cantidad de experimentos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for I=rango_entradas
    for N =rango_neuronas
        for F = rango_feedback
            net = narxnet(0:I,1:F,N);
            net.performParam.normalization = 'percent';
            net.trainParam.showWindow = false;

            NN = cell(1,num_exp);
            perfs = zeros(1,num_exp);

            for j=1:num_exp
              [Xs,Xi,Ai,Ts, EWs, shift] = preparets(net, X_1, {} ,T_1);

              NN{j} = train(net,Xs,Ts, Xi, 'useParallel','yes');

              NN{j} = closeloop(NN{j});
              [Xs,Xi,Ai,Ts, EWs, shift] = preparets(NN{j}, X_2, {} ,T_2
    );

              Y = NN{j}(Xs, Xi, Ai);

              perfs(j) = mse(T_2(end-size(Y,2)+1:end),Y); disp(['
    Performance: ' num2str(perfs(j))])
              disp(' ')
            end
            save(sprintf('NARX_%dI_%dN_%dF_%d.mat', I, N, F, num_exp),
    'NN', 'perfs')
        end
    end
end

load handel
sound(y,Fs)
```

A.3 Generación y entrenamiento de redes timedelayed

Este código se utiliza para entrenar redes del tipo timedelayed.

```
clear; clc; close all;
format long

load ./DATASETS/processed_RP_syncronized2.mat

N_Sujeto = 49;

X_1 = data{N_Sujeto}.X_1;
T_1 = data{N_Sujeto}.T_1;
X_2 = data{N_Sujeto}.X_2;
T_2 = data{N_Sujeto}.T_2;
x_PS = data{N_Sujeto}.x_PS;
t_PS = data{N_Sujeto}.t_PS;
rev = data{N_Sujeto}.rev;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rango_neuronas = 15;
rango_entradas = 1:3:40;
num_exp = 10;              %Cantidad de experimentos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for N =rango_neuronas
    for I=rango_entradas
        net = timedelaynet(0:I,N);
        net.performParam.normalization = 'percent';
        net.trainParam.showWindow = false;

        NN = cell(1,num_exp);
        perfs = zeros(1,num_exp);
        t = zeros(1,num_exp);
        for j=1:num_exp
          disp(['Training ' num2str(j) '/' num2str(num_exp) ' - IN='
    num2str(I) ' - Neu=' num2str(N)])

          [Xs,Xi,Ai,Ts, EWs, shift] = preparets(net, X_1, T_1);

          NN{j} = train(net,Xs,Ts, Xi, 'useParallel','yes');

          [Xs,Xi,Ai,Ts, EWs, shift] = preparets(NN{j}, X_2, T_2);

          Y = NN{j}(Xs, Xi, Ai);

          perfs(j) = mse(T_2(end-size(Y,2)+1:end),Y); disp(['
    Performance: ' num2str(perfs(j))])

          disp(' ')
        end
```

```
46
47          save(sprintf('timedelaynet_%dI_%dN_%d.mat', I, N, num_exp), 'NN
    ', 'perfs')
48      end
49  end
50
51  load handel
52  sound(y,Fs)
```

A.4 RMSE normalizado

Para el cálculo del RMSE normalizado se utilizó la siguiente función.

```
1  function [NRMS Index]=normError(x,y)
2
3  x=x(:);
4  y=y(:);
5  aux1=x(end*.25:end*.75);
6  for pp=-50:50;
7      aux2=y(end*.25+pp:end*.75+pp);
8      nRMS(pp+51)=rms(aux2-aux1)/rms(aux1);
9      ndRMS(pp+51)=rms(diff(aux2)-diff(aux1))/rms(diff(aux1));
10 end
11
12 [~, Index]=min(nRMS);
```

A.5 Sincronización

Para sincroninizar las señales del GVV porviniente del OVV y la señal del acelerómetro se utilizó la siguiente función.

```
1  function [xa, ya, delay]= syncw(x,y,lim)
2
3  corr = xcorr(x,y);
4  delay=-1;
5  while(delay<0 || delay > lim)
6      [~, idx] = max(corr);
7      delay = idx-numel(x);
8      corr(idx)=0;
9  end
10
11 xa=x(delay+1:end);
12
13 if(numel(y)<numel(xa))
14     xa= xa(1:numel(y));
15 end
16 ya=y;
```

Apéndice B

Gráficas

B.1 MSE vs Retardos a la entrada

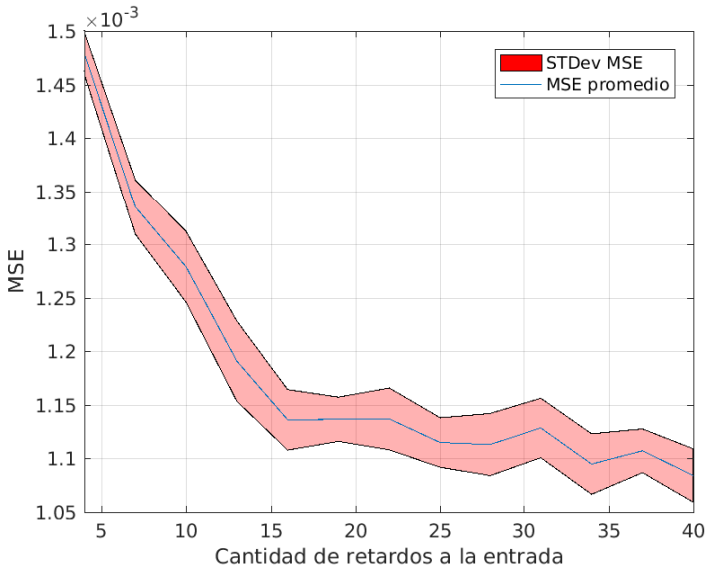


Figura B.1: MSE vs retardos a la entrada con 15 neuronas, para paciente 22.

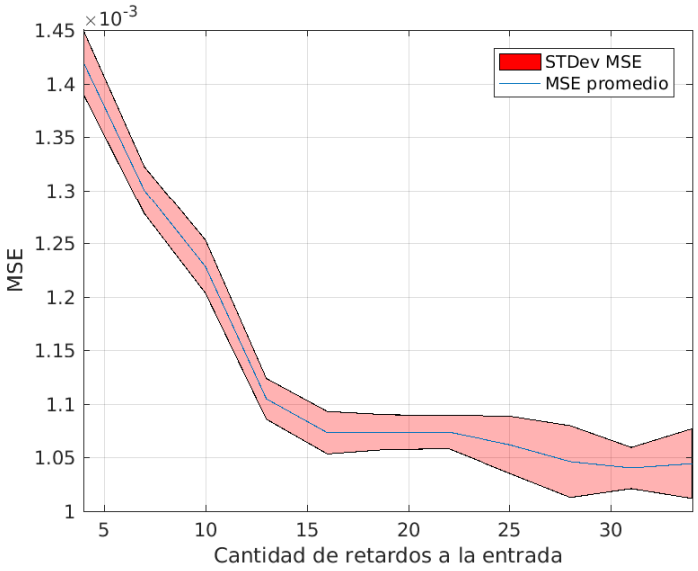


Figura B.2: MSE vs retardos a la entrada con 30 neuronas, para paciente 22.

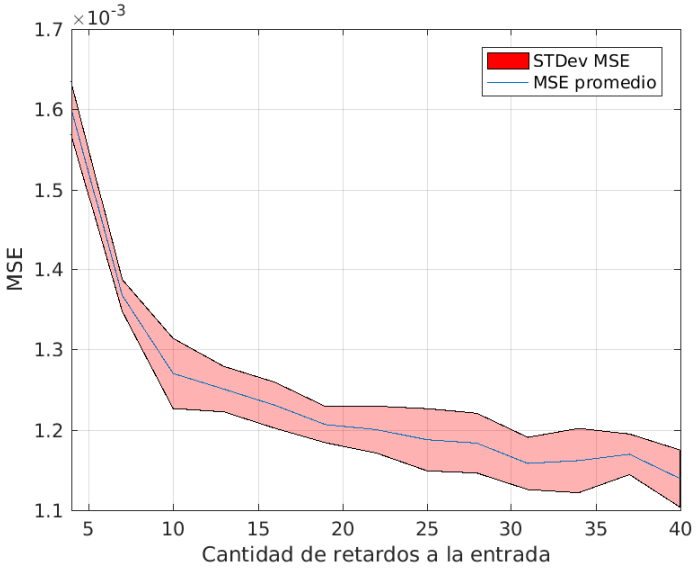


Figura B.3: MSE vs retardos a la entrada con 15 neuronas, para paciente 79.

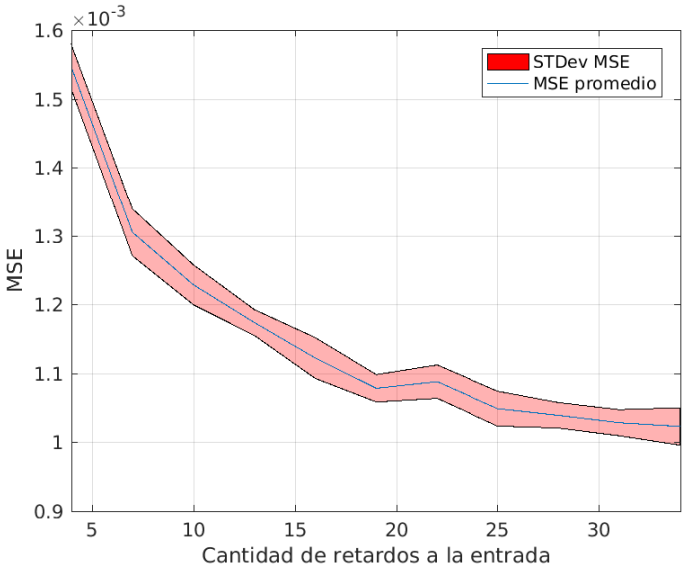


Figura B.4: MSE vs retardos a la entrada con 30 neuronas, para paciente 79.

B.2 MSE vs Cantidad de Neuronas

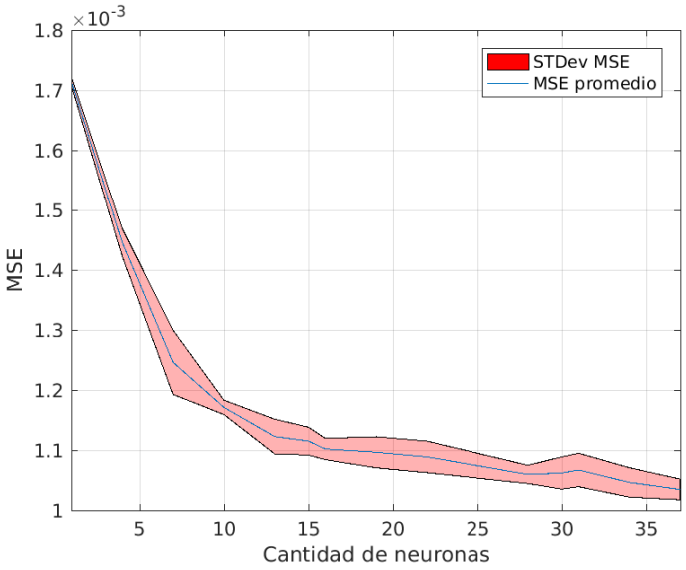


Figura B.5: MSE vs cantidad de neuronas con 25 retardos a la entrada, para paciente 22.

B.3 MSE vs Retardos a la realimentación

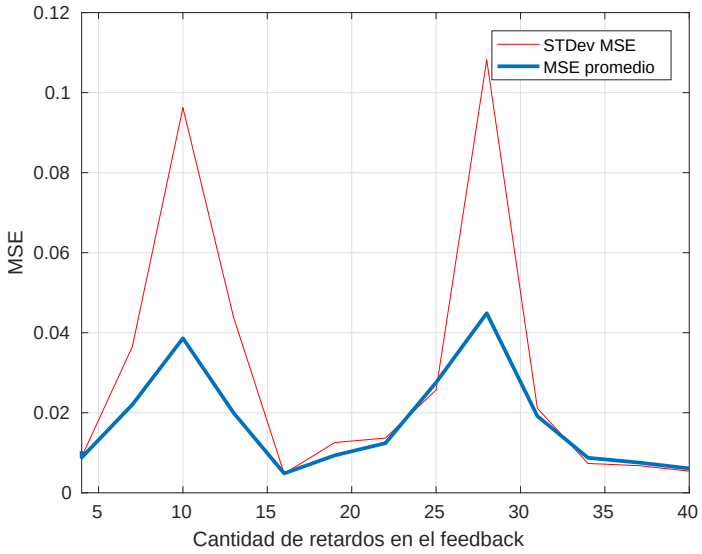
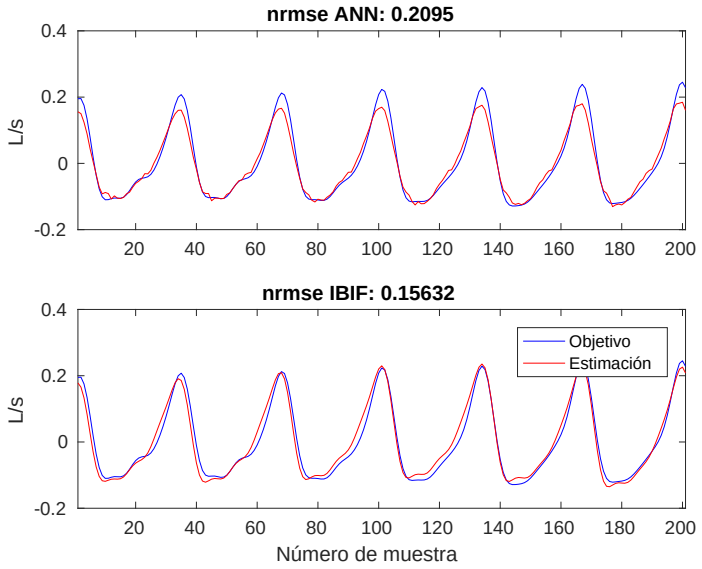
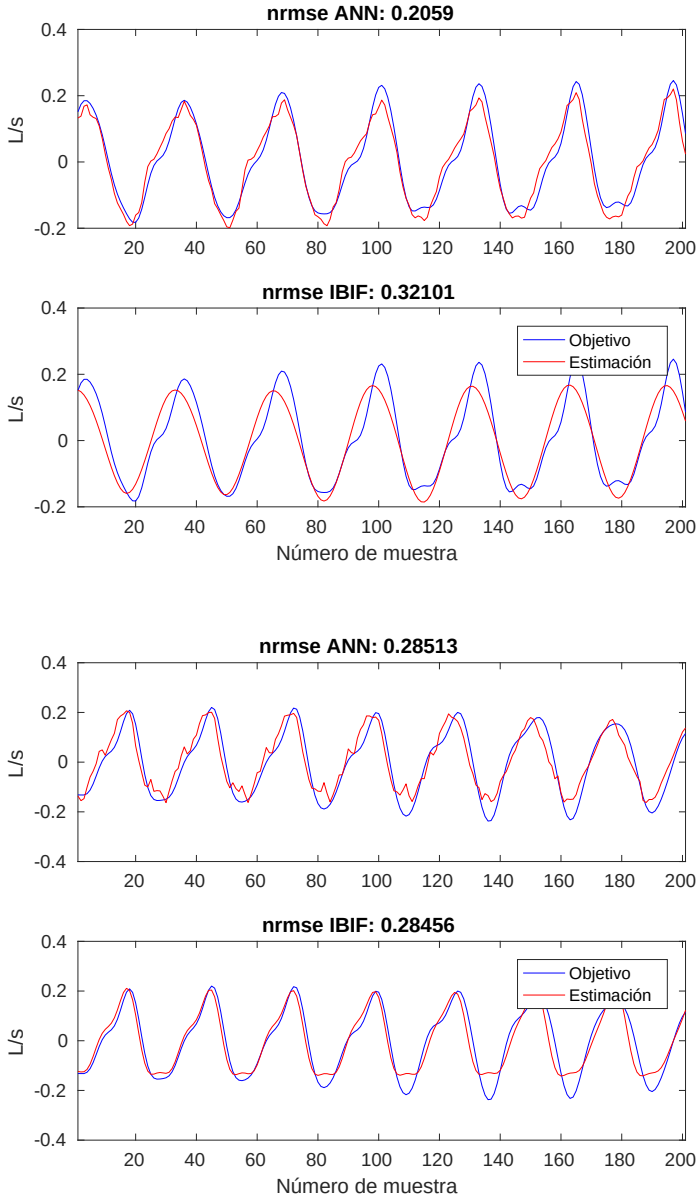
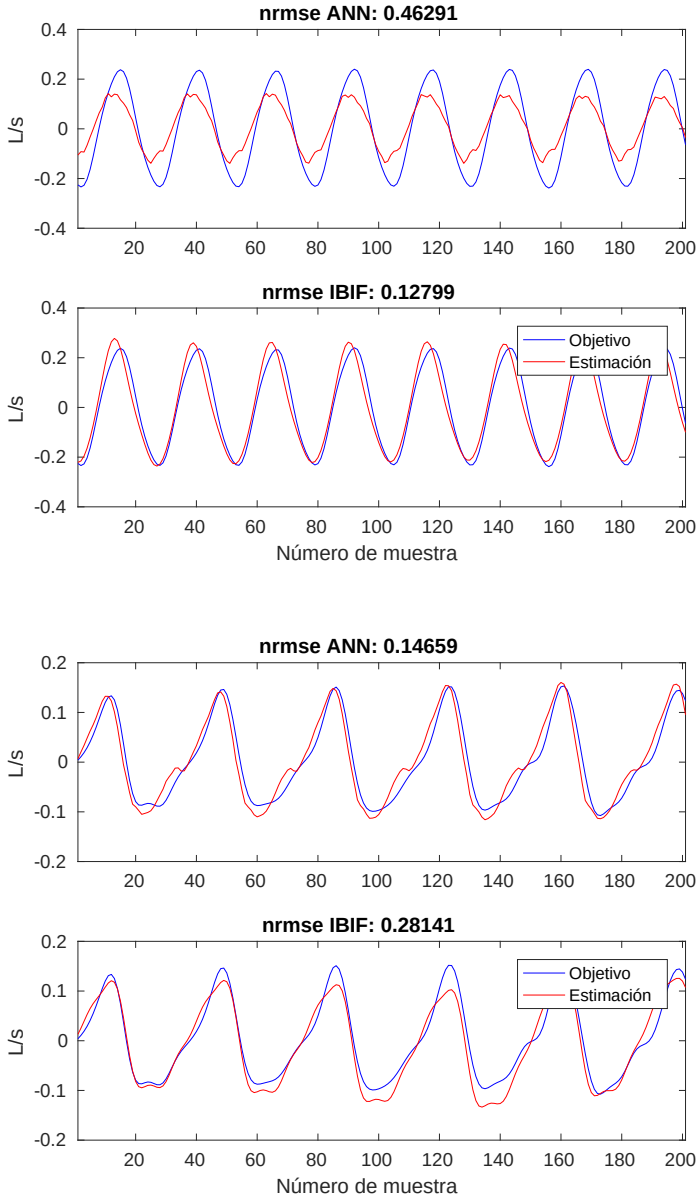


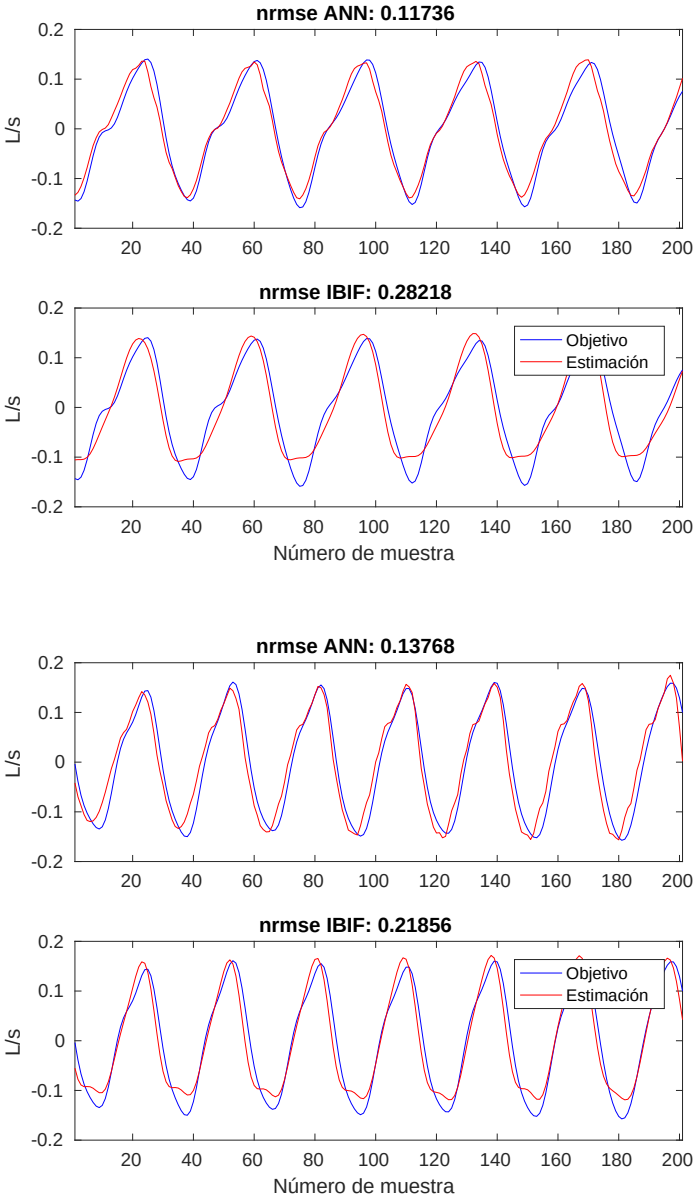
Figura B.6: MSE vs cantidad de retardos en la realimentación, para una red NARX con 22 neuronas y 25 retardos a la entrada, para paciente 22.

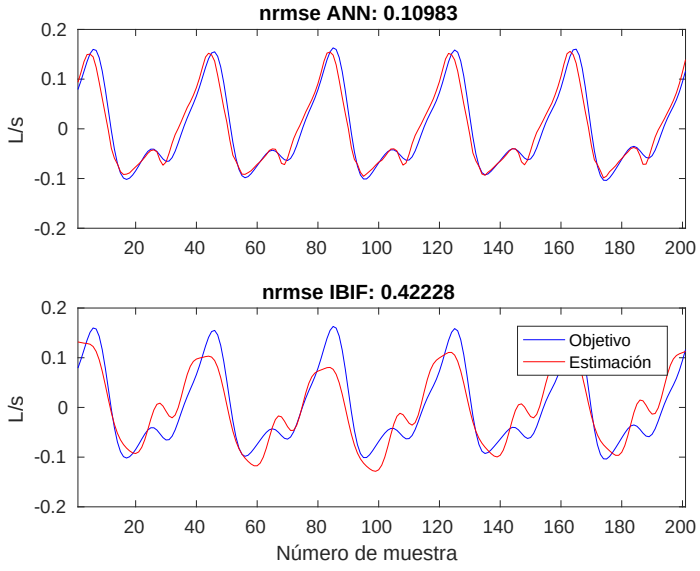
B.4 Ejemplos de Estimaciones RP



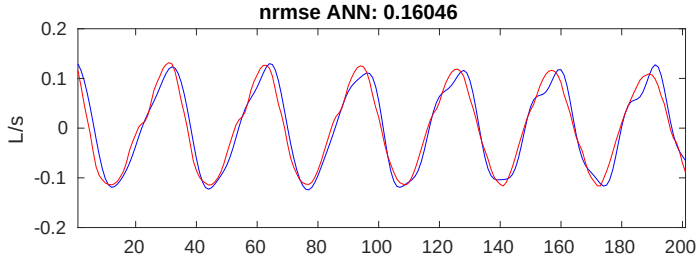

nrmse ANN: 0.2059
nrmse IBIF: 0.32101
Objetivo
Estimación
L/s
Número de muestra
nrmse ANN: 0.28513
nrmse IBIF: 0.28456
Objetivo
Estimación
L/s
Número de muestra

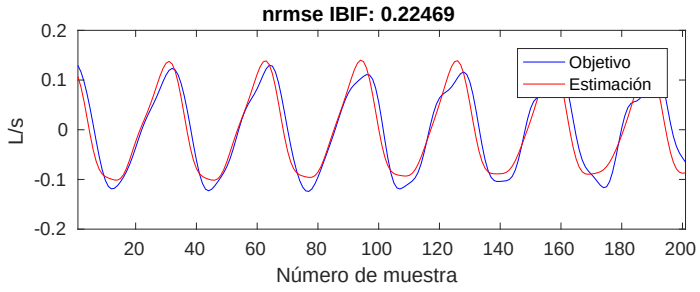

nrmse ANN: 0.46291
L/s
nrmse IBIF: 0.12799
Objetivo
Estimación
Número de muestra
nrmse ANN: 0.14659
L/s
nrmse IBIF: 0.28141
Objetivo
Estimación
Número de muestra

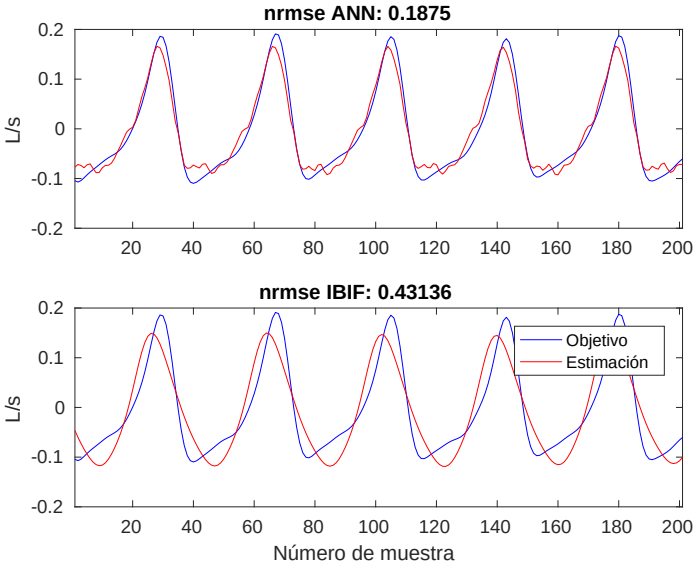

nrmse ANN: 0.11736
nrmse IBIF: 0.28218
Objetivo
Estimación
L/s
Número de muestra
nrmse ANN: 0.13768
nrmse IBIF: 0.21856
Objetivo
Estimación
L/s
Número de muestra

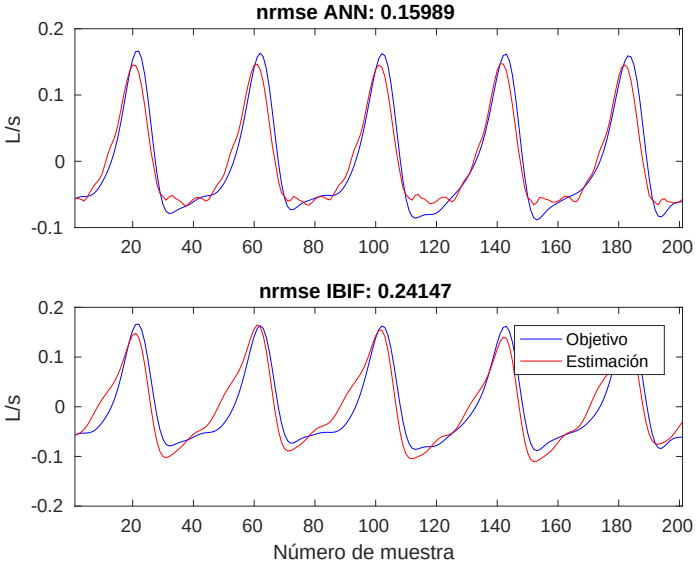

nrmse ANN: 0.10983
L/s
0.2
0.1
0
-0.1
-0.2
20
40
60
80
100
120
140
160
180
200

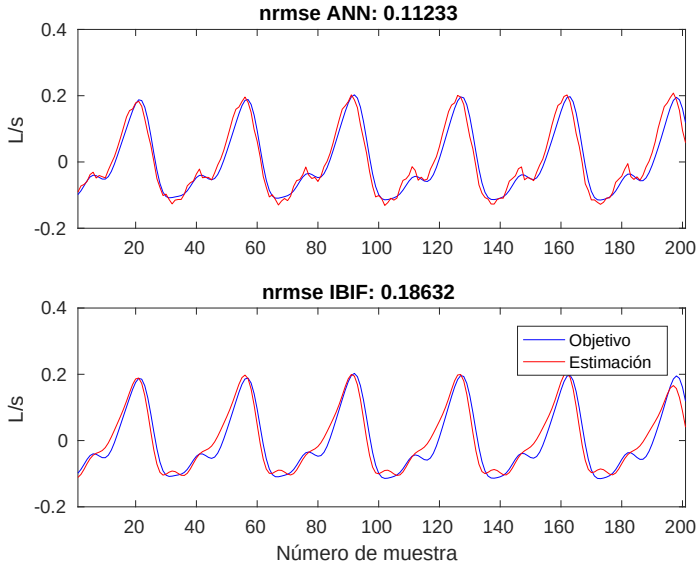
nrmse IBIF: 0.42228
L/s
Objetivo
Estimación
0.2
0.1
0
-0.1
-0.2
20
40
60
80
100
120
140
160
180
200
Número de muestra


nrmse ANN: 0.16046
L/s
0.2
0.1
0
-0.1
-0.2
20
40
60
80
100
120
140
160
180
200

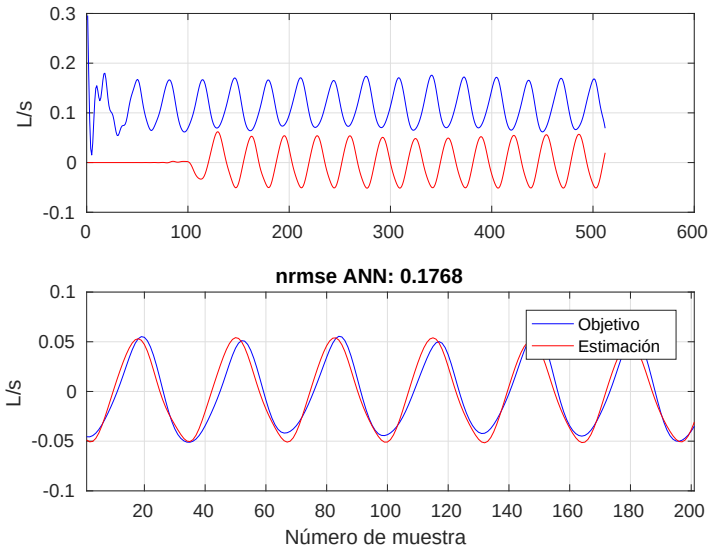

nrmse IBIF: 0.22469
L/s
Objetivo
Estimación
0.2
0.1
0
-0.1
-0.2
20
40
60
80
100
120
140
160
180
200
Número de muestra

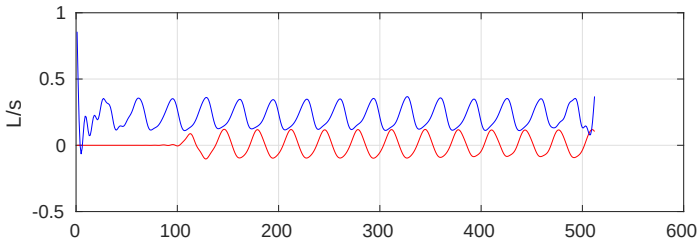

nrmse ANN: 0.1875
nrmse IBIF: 0.43136
Objetivo
Estimación
L/s
Número de muestra

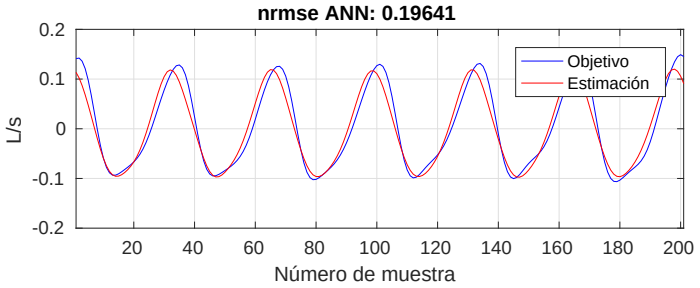

nrmse ANN: 0.15989
nrmse IBIF: 0.24147
Objetivo
Estimación
L/s
Número de muestra

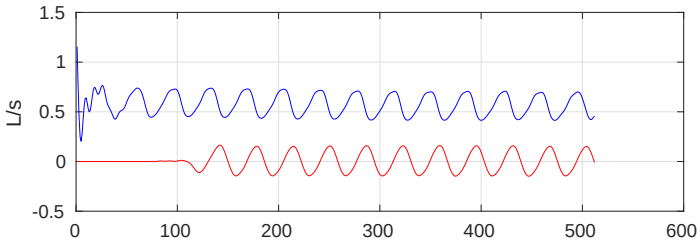


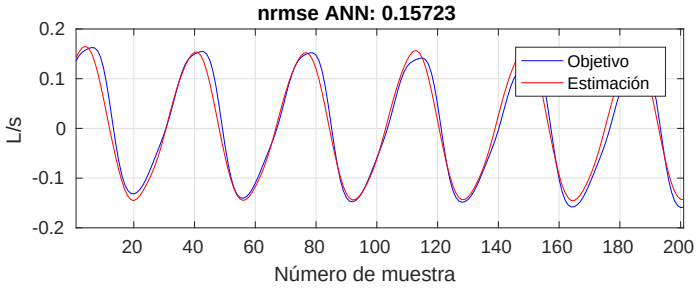
B.5 Ejemplos de estimaciones PAE

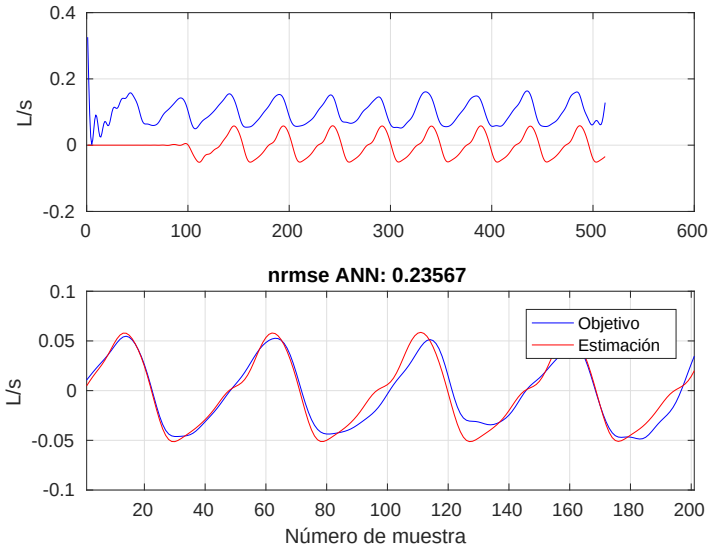



1
0.5
0
-0.5
L/s
0
100
200
300
400
500
600

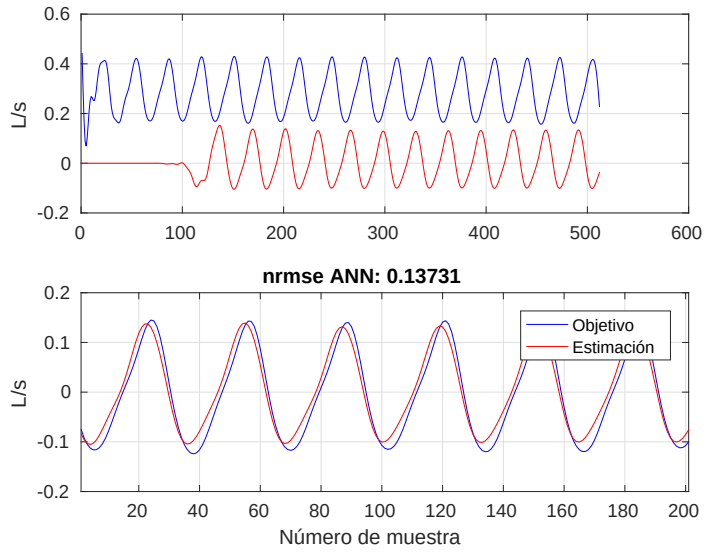

nrmse ANN: 0.19641
0.2
0.1
0
-0.1
-0.2
L/s
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra


1.5
1
0.5
0
-0.5
L/s
0
100
200
300
400
500
600

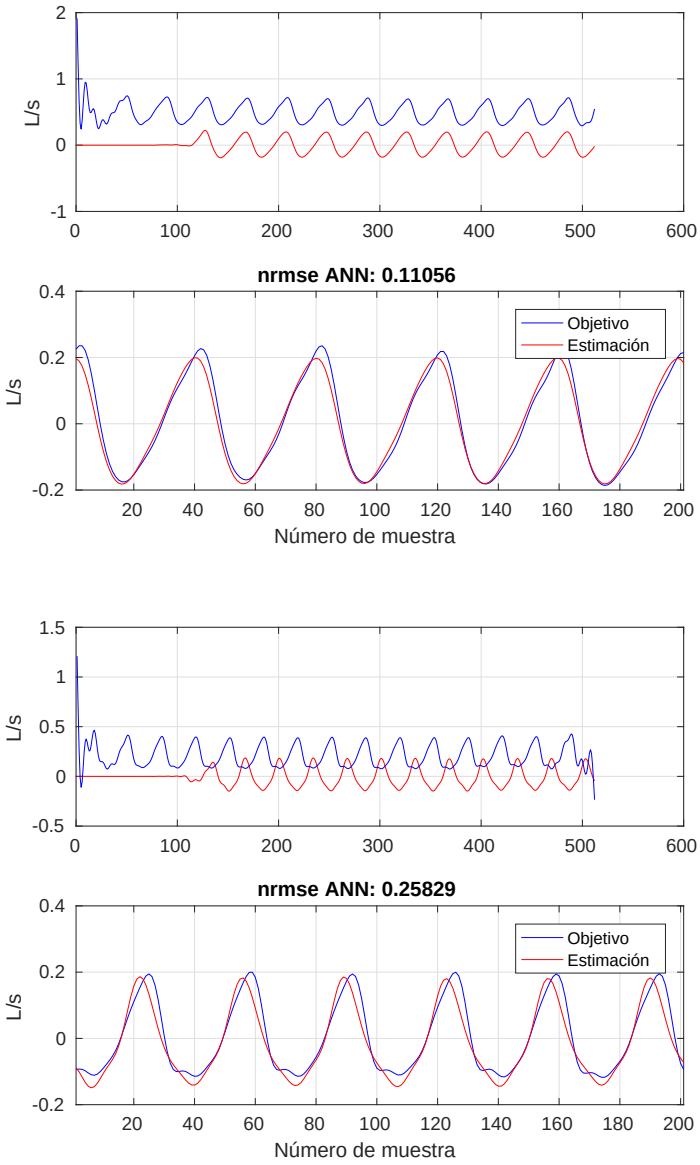

nrmse ANN: 0.15723
0.2
0.1
0
-0.1
-0.2
L/s
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra

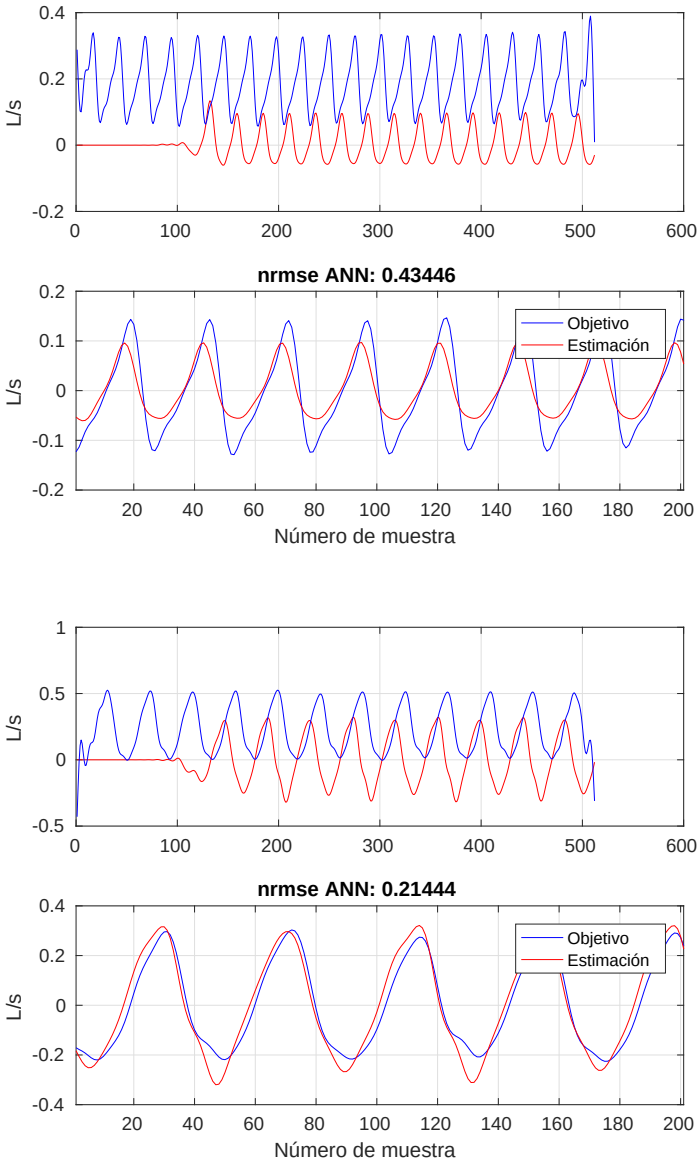

L/s
0.4
0.2
0
-0.2
0
100
200
300
400
500
600

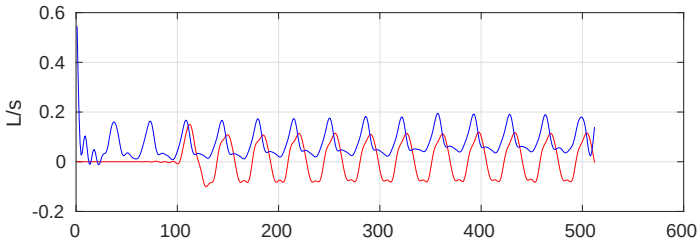
nrmse ANN: 0.23567
L/s
0.1
0.05
0
-0.05
-0.1
20
40
60
80
100
120
140
160
180
200
Número de muestra
Objetivo
Estimación

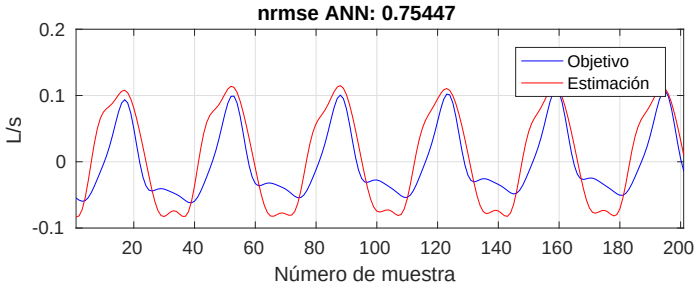

L/s
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600

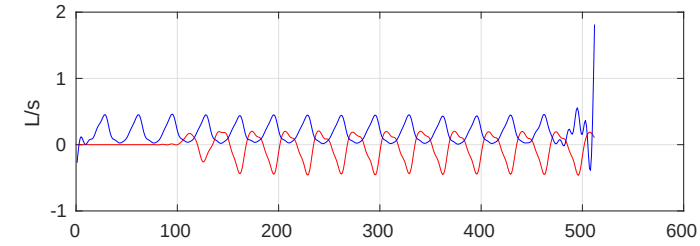
nrmse ANN: 0.13731
L/s
0.2
0.1
0
-0.1
-0.2
20
40
60
80
100
120
140
160
180
200
Número de muestra
Objetivo
Estimación

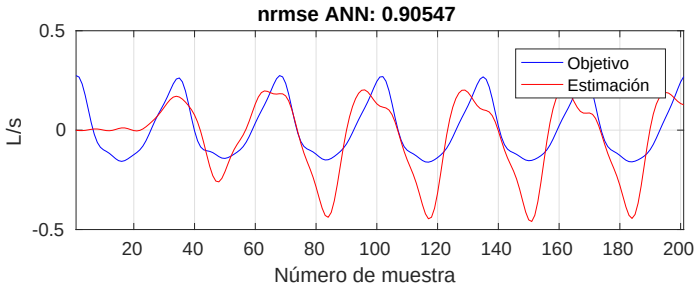

L/s
nrmse ANN: 0.11056
Objetivo
Estimación
Número de muestra
L/s
nrmse ANN: 0.25829
Objetivo
Estimación
Número de muestra

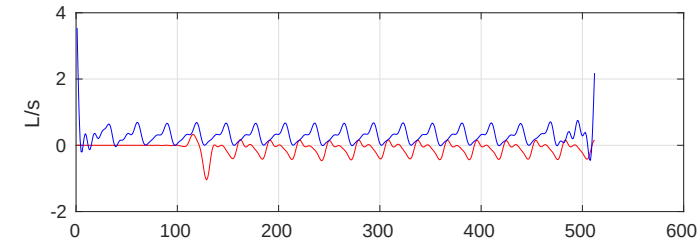

L/s
nrmse ANN: 0.43446
Objetivo
Estimación
Número de muestra
L/s
nrmse ANN: 0.21444
Objetivo
Estimación
Número de muestra

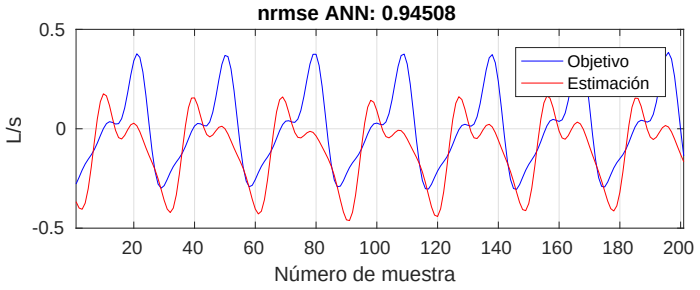

0.6
0.4
0.2
0
-0.2
L/s
0
100
200
300
400
500
600

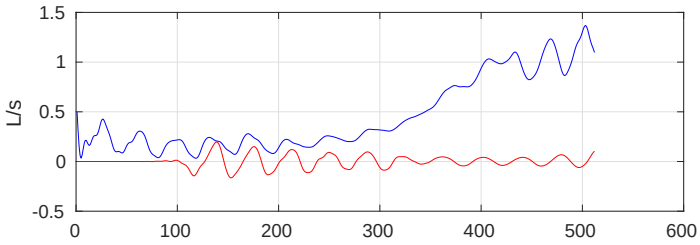

nrmse ANN: 0.75447
0.2
0.1
0
-0.1
L/s
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra

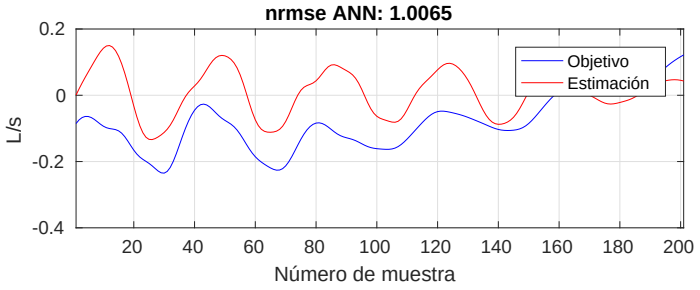

2
1
0
-1
L/s
0
100
200
300
400
500
600


nrmse ANN: 0.90547
0.5
0
-0.5
L/s
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra


L/s
4
2
0
-2
0
100
200
300
400
500
600


nrmse ANN: 0.94508
L/s
0.5
0
-0.5
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra


L/s
1.5
1
0.5
0
-0.5
0
100
200
300
400
500
600


nrmse ANN: 1.0065
L/s
0.2
0
-0.2
-0.4
Objetivo
Estimación
20
40
60
80
100
120
140
160
180
200
Número de muestra

Bibliografía

[1] N. Roy, R. M. Merrill, S. D. Gray, and E. M. Smith, “Voice disorders in the general population: Prevalence, risk factors, and occupational impact,” Laryngoscope, vol. 115, no. 11, pp. 1988–1995, 2005.

[2] R. E. Hillman, E. B. Holmberg, J. S. Perkell, M. Walsh, and C. Vaughan, “Objective assessment of vocal hyperfunction: An experimental framework and initial results,” J. Speech Hear. Res., vol. 32, no. 2, pp. 373–392, 1989.[f]

[3] R. E. Hillman, E. B. Holmberg, J. S. Perkell, M. Walsh, and C. Vaughan, “Phonatory function associated with hyperfunctionally related vocal fold lesions,” J. Voice, vol. 4, no. 1, pp. 52–63, 1990.

[4] K. Verdolini, C. Rosen, and R. C. Branski, Classification Manual for Voice Disorders-I, Special Interest Division 3, Voice and Voice disorders, American Speech-Language Hearing Division. Mahwah, NJ: Lawrence Erlbaum, 2005.[h]

[5] D. D. Mehta, M. Zañartu, S. W. Feng, H. A. Cheyne II, and R. E. Hillman, “Mobile voice health monitoring using a wearable accelerometer sensor and a smartphone platform,” IEEE Trans. Biomed. Eng. 59(11), 3090–3096 (2012).

[6] J. H. Van Stan, J. Gustafsson, E. Schalling, and R. E. Hillman, “Direct comparison of commercially available ambulatory voice monitors: A clinical perspective,” Perspect. Voice Voice Disorders 24(2), 80–86 (2014).[i]

[7] JUSTICE, Laura M. ”Communication Sciences and Disorders: An Introduction” 2a ed. Estados Unidos, Prentice Hall, 2005. 608 p.

[8] M. Zañartu, J. C. Ho, D. D. Mehta, R. E. Hillman, and G. R. Wodicka, “Subglottal impedance-based inverse filtering of speech sounds using neck surface acceleration,” IEEE Trans. Audio Speech Lang. Process. 21(9), 1929–1939 (2013).

[9] GAZZANIGA, Michael S., IVRY, Richard B., MANGUN, George R., ”Cognitive Neuroscience: The Biology of the Mind” 4a ed. Estados Unidos, W. W. Norton & Company, 2013. 752 p.

[10] FAUSETT, Laurene V. ”Fundamentals of Neural Networks: Architectures, Algorithms And Applications” 1a ed. Estados Unidos, Pearson, 1993. 461p.

[11] HAYKIN, Simon O. ”Neural Networks and Learning Machines” 3a ed., Canada, Pearson, 2008. 936 p.

[12] Y. Liu, Y. C. Lee, H. H. Chen and G. Z. Sun, ”Speech recognition using dynamic time warping with neural network trained templates,” Neural Networks, 1992. IJCNN., International Joint Conference on, Baltimore, MD, 1992, pp. 326-331 vol.2. doi: 10.1109/IJCNN.1992.226967

[13] Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes Jack V. Tu Journal of Clinical Epidemiology Volume 49, Issue 11, November 1996, Pages 1225-1231

[14] K. S. Narendra and K. Parthasarathy, ”Identification and control of dynamical systems using neural networks,” in IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 4-27, Mar 1990.

[15] D. F. Specht, ”A general regression neural network,” in IEEE Transactions on Neural Networks, vol. 2, no. 6, pp. 568-576, Nov 1991.

[16] H. A. Rowley, S. Baluja and T. Kanade, ”Neural network-based face detection,” Computer Vision and Pattern Recognition, 1996. Proceedings CVPR ’96, 1996 IEEE Computer Society Conference on, San Francisco, CA, 1996, pp. 203-208.

[17] B. R. Szkuta, L. A. Sanabria and T. S. Dillon, ”Electricity price short-term forecasting using artificial neural networks,” in IEEE Transactions on Power Systems, vol. 14, no. 3, pp. 851-857, Aug 1999.

[18] Huiping Li, D. Doermann and O. Kia, ”Automatic text detection and tracking in digital video,” in IEEE Transactions on Image Processing, vol. 9, no. 1, pp. 147-156, Jan 2000.

[19] “Tensorflow: Large-scale machine learning on heterogeneous systems” por Abadi Martn, [et. al]. . 2015.

[20] KRIESEL David, “A Brief Introduction to Neural Networks” [en linea], <http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf>, [consulta: 07 de Abril 2017]

[21] Cybenko, G., 1989: Approximation by superposition of sigmoidal functions, in mathematics of control. Signals Syst., 2, 303–314.

[22] Snyman, J., “Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms.”, Vol. 97. Springer Science & Business Media, 2005.

[23] Manuela Battipede, Piero Gili, Angelo Lerro, “Neural Networks for Air Data Estimation: Test of Neural Network Simulating Real Flight Instruments”, Engineering Applications of Neural Networks,Volume 311 of the series Communications in Computer and Information Science pp 282-294

[24] Antonios Papaleonidas, Lazaros Iliadis, “Employing ANN That Estimate Ozone in a Short-Term Scale When Monitoring Stations Malfunction” Engineering Applications of Neural Networks, Volume 311 of the series Communications in Computer and Information Science pp 71-80

[25] Krystyna Kuzniar, Lukasz Chudyba, “Neural Networks for the Analysis of Mine-Induced Vibrations Transmission from Ground to Building Foundation”, Engineering Applications of Neural Networks, Volume 311 of the series Communications in Computer and Information Science pp 162-171.

[26] Fairbanks, G. (1960). Voice and Articulation Drillbook. New York, NY: Harper and Row

[27] Mathworks Matlab - Neural Network Toolbox [en linea] <https://www.mathworks.com/products/neural-network.html/> [consulta: 16 Febrero 2017]

[28] Neural Network Design por Martin Hagan [et. al]. 2a ed. Estados Unidos, Martin Hagan, 2014. 800p

[29] BEALE Mark H., HAGAN Martin T.,DEMUTH Howard B. Neural Network Toolbox™User’s Guide [en linea] <https://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf> [consulta: 16 Febrero 2017]

[30] SARLE, Warren S., Neural Network FAQ, [en linea] <ftp://ftp.sas.com/pub/neural/FAQ.html> [consulta: 13 Enero 2017]

[31] L. Hamm, B. W. Brorsen and M. T. Hagan, “Comparison of Stochastic Global Optimization Methods to Estimate Neu- ral Network Weights,” Neural Processing Letters, Vol. 26, No. 3, December 2007.