Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Modelo farmacodinámico *in silico* de almeja pismo con digoxina usando técnicas de inteligencia artificial

Tesis presentada por

Rubén Alfonso Castañeda Martínez

para obtener el grado de

Bioingeniero

Directora: Dra. Dora Luz Flores Gutiérrez

Ensenada, México, septiembre de 2017

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

MODELO FARMACODINÁMICO *IN SILICO* DE ALMEJA PISMO CON DIGOXINA USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

TESIS

PARA CUBRIR LOS REQUISITOS NECESARIOS PARA OBTENER EL TÍTULO DE

BIOINGENIERO

PRESENTA:

RUBÉN ALFONSO CASTAÑEDA MARTÍNEZ

Aprobada por:

Dra. Dora Luz Flores Gutiérrez
Director
(Presidente)

Dr. David Cervantes Vásquez
Sinodal
(Secretario)

Dra. Claudia Mariana Gómez Gutiérrez
Sinodal
(Vocal)

Dra. María de los Ángeles Cosío León
Sinodal
(Vocal)

Dr. Rubén César Villarreal Sánchez
Sinodal
(Vocal)

A mis padres y hermanos

Gracias por tanto apoyo y amor incondicional

Agradecimientos

Estoy inmensamente agradecido con mi familia, quienes me lo han dado todo. Especialmente con mi madre Oxana Martínez Gómez y mi padre Paúl Alfonso Castañeda Delgado, no podría estar más bendecido por tenerlos. Los amo profundamente, y les agradezco todo el inmenso apoyo, manifestado de muchísimas maneras, que me han brindado durante todo este tiempo, así como la paciencia que me han tenido.

Quiero agradecer a mi equipo, Alberto Abaroa Villanueva y Carlos Alberto Castro Estrada, mis hermanos académicos, con quienes trabajé durante más de dos años. Gracias por el apoyo brindado cuando tuve dificultades para avanzar, sin ustedes este trabajo no hubiera sido posible. La mayoría de los códigos utilizados en la metodología de esta tesis fueron desarrollados por los tres.

Un agradecimiento muy especial a la Dra. Dora Luz Flores Gutiérrez, mi madre académica. Gracias por permitirme trabajar con usted, por asesorarme, por enseñarme tanto y por darme tantas oportunidades de aprender, conocer, crecer y autocorregirme, y sobre todo, por no dejarme solo. Mi vida se expandió gracias a usted, tanto en lo académico como en otros aspectos. Admiro mucho sus ganas de siempre aprender más y mejorar, sin duda es para mí todo un ejemplo que me motiva a seguir adelante en todo momento.

También quiero agradecer profundamente a Samuel García Zarate, por su gran disposición de apoyarme para entender algunos códigos de Java.

Al Dr. Rubén César Villarreal Sánchez y al Dr. David Cervantes Vásquez, por mostrarme tal calidad de seres humanos, gracias por ser un ejemplo a seguir.

A la Dra. María de los Ángeles Cosío León, por sus valiosos comentarios para mejorar este trabajo de tesis.

A la Dra. Claudia Mariana Gómez Gutiérrez, por los datos proporcionados para la realización de este trabajo.

A la Universidad Autónoma de Baja California, y a la Facultad de Ingeniería, Arquitectura y Diseño, por darme la oportunidad de estudiar una carrera tan bella.

Resumen

Las redes neuronales artificiales (RNA) son un poderoso método computacional capaz de resolver problemas no lineales y encontrar patrones complejos en sistemas, que han sido utilizadas extensamente en diversos campos de aplicación para llevar a cabo predicciones, optimizaciones y clasificación de datos.

En este trabajo se utilizaron RNA en comparación con otras técnicas del estado del arte de *machine learning* para predecir parámetros biofísicos específicos de la actividad cardíaca de la almeja *Tivela stultorum* al adicionarle una concentración específica de digoxina. Esta almeja ha sido previamente validada como modelo biológico en la farmacología. Para validar los modelos se utilizaron los parámetros de error medio cuadrado (MSE), raíz del error medio cuadrado (RMSE), exactitud al 5 %, los coeficientes de regresión R^2 y la raíz del error medio cuadrado relativo (RRMSE), para el caso de las RNA. Para los algoritmos de *machine learning* se utilizó únicamente el RRMSE. Se utilizaron los algoritmos de entrenamiento Levenberg-Marquardt (LM), *Bayesian Regulation* (BR), *Scaled Conjugated Gradient* (SCG) para las RNA, en comparación contra los algoritmos *Stacked Single-Targets* (SST), *Ensemble of Regressor Chains* (ERC), *Random Linear target Combinations* (RLC), y *Single-Target* (ST) como métodos de regresión múltiple utilizando como base los algoritmos de salida única *Bagged regression trees* (BAG) y *support vector regression* SVR. Se obtuvieron modelos farmacodinámicos con un MSE muy cercano al 0 y una R^2 alta en el caso de las RNA, y con un RRMSE bajo en todos los casos. No se encontró diferencia significativa entre las

RNA-PMC y el mejor algoritmo utilizado de *machine learning*, ERC. Para las RNA el mejor algoritmo de entrenamiento fue BR seguido de LM, mientras que para el resto de los métodos de *machine learning* fue ERC utilizando como base el algoritmo BAG. Los resultados muestran que el algoritmo SVR no es una buena base para regresión múltiple.

Se necesitan nuevos experimentos que comparen RNA con topologías más actualizadas como las redes neuronales evolutivas, adaptativas de Newmann-Watts, recurrentes, entre otras, y diferentes métodos de validación para los algoritmos de *machine learning* para encontrar la configuración óptima para este caso.

Palabras clave: Almeja pismo, aprendizaje automático, cardioactividad, redes neuronales artificiales, regresión múltiple, digoxina, modelado farmacodinámico, modelo *in silico*, *Tivela stultorum*.

Abstract

Artificial neural networks (ANNs) are a powerful computational method capable of solving nonlinear problemas and find complex relationships in a extensive range of systems, and they have been widely used in different fields of application to carry out predictions, optimizations and data classification.

In this work, ANNs in comparison with other state-of-the-art machine learning algorithms were used to predict specific biophysical parameters of the cardioactivity of the pismo clam *Tivela stultorum* when a specific concentration of digoxin was added to his heart. The pismo clam is a bivalve mollusc that has been previously validated as a biological model in pharmacology. To validate the models, the mean-squared error (MSE), accuracy at 5 %, the regression coefficients R^2 and the relative root mean-squared error (RRMSE) were used in the case of the MLP-ANNs, and the RRMSE was used in the rest of the machine learning algorithms. The algorithms of Levenberg-Marquardt (LM), Bayesian Regulation (BR), Scaled Conjugated Gradient (SCG), Stacked Single-Targets (SST), Ensemble of Regressor Chains (ERC), Random Linear target Combinations (RLC), and Single-Target (ST) were used to train the models. Pharmacodynamic models with good fitting and an excellent prediction capability were obtained, with a low MSE and a high R^2, and with a low RRMSE in every case. No significative difference was found between the performance of MLP-ANNs and the rest of the algorithms. The BR algorithm, followed by LM, were the best to train the MLP-ANNs, and in the case of the rest of the algorithms, the best performance was achieved by ERC using

BAG algorithm as base regressor. The results show that SVR is not a good base algorithm for multi-output regression.

New experiments are needed in order to compare MLP-ANNs with more updated topologies, such as evolutive neural networks, adaptive Newmann-Watts, recurrent, among others, and different validation methods for machine learning algorithms to find the optimum configuration applied to the pharmacological-related dataset.

Keywords: Artificial neural networks, cardioactivity, digoxin, *in silico* modelling, machine learning, multioutput regression, pharmacodynamic modelling, pismo clam, *Tivela stultorum*.

Índice general

Índice de tablas

Tabla **Página**

Índice de figuras

Figura **Página**

Lista de abreviaciones

ADBE	Absorción, Distribución, Biotransformación y Excreción
ANOVA	Análisis de varianza
ATP	Adenosín trifosfato
aRRMSE	*average Relative root mean-square error*
BAG	*Bagged regression trees*
BR	*Bayesian Regulation*
ERC	*Ensemble of Regressor Chains*
FC/FD	Farmacocinético/Farmacodinámico
IA	Inteligencia artificial
***k*-NN**	*k-Nearest Neighbor*
LM	Levenberg-Marquardt
MSE	*Mean-square error*
NKA	Bomba adenosín trifosfatasa dependiente de sodio y potasio
OMS	Organización Mundial de la Salud
PMC	Perceptrón multicapa
PSO	*Particle swarm optimization*
RLC	*Random Linear target Combinations*
RMSE	*Root mean-square error*
RNA	Redes neuronales artificiales
RRMSE	*Relative root mean-square error*
SCG	*Scaled Conjugated Gradient*
SST	*Stacked Single-Target*
ST	*Single-Target*
SVM	*Support Vector Machine*
SVR	*Support Vector Regression Machine*

Prefacio

Las redes neuronales artificiales (RNA) han sido extensamente aplicadas para predicción y optimización de sistemas en diversos campos de la ciencia y la industria; específicamente, la topología perceptrón multicapa (PMC) ha sido optimizada a lo largo de décadas de estudio.

Sin embargo, varios revisores han comentado al equipo de trabajo de Biología Computacional de la Facultad de Ingeniería, Arquitectura y Diseño - Universidad Autónoma de Baja California, al cual el autor perteneció durante dos años, que la topología PMC suele converger la generalización de la salida a un mínimo local y no a un mínimo general, provocando que el modelo tenga un sesgo ante nueva información que se presente; aunado a esto, el tiempo requerido para el entrenamiento de una RNA-PMC es bastante alto. Consecuentemente, se considera como una técnica computacional si bien no obsoleta, desactualizada, superada por técnicas más recientes del *machine learning*. Las máquinas de soporte vectorial (SVM, del inglés *support vector machines*) superan la capacidad predictiva de las RNA-PMC desarrollando modelos más efectivos, con una mayor exactitud en sus predicciones y en un tiempo mucho menor, comentan algunos revisores de varias revistas científicas. El trabajo que desarrollamos como equipo ha demostrado lo contrario en repetidas ocasiones. Se construyeron modelos basados en RNA-PMC con diferentes conjuntos de datos, algoritmos de entrenamiento y variables de entrada y salida. En todos los casos se obtuvieron RNA con una correlación y capacidad de predicción muy altas con respecto de las variables de salida, utilizando técnicas

de entrenamiento y validación que previenen el sobreentrenamiento de las redes.

La idea de este trabajo es comparar a las RNA-PMC con algoritmos de *machine learning*, con el fin de definir cuál algoritmo sería mejor para aplicaciones en farmacología, específicamente, en el modelado para predicción de la respuesta fisiológica de corazones de la almeja *Tivela stultorum* a un fármaco cardiotónico.

Estructura de la tesis

En el capítulo 1 se expone el marco teórico, antecedentes, justificación, hipótesis y objetivos, entre otros aspectos. En el capítulo 2 se describe la metodología utilizada para construir los modelos farmacodinámicos, los métodos de validación utilizados, los parámetros para medir su desempeño y el análisis estadístico aplicado. En el capítulo 3 se presentan los resultados obtenidos del análisis de los modelos generados. En el capítulo 4 se hace un análisis de estos resultados. En el capítulo 5, se exponen las conclusiones extraídas del análisis de la información obtenida y se plantea el trabajo a futuro que podría llevarse a cabo para complementar este proyecto. Finalmente, en pro del *open source* y bajo el ideal de que la ciencia debería estar disponible para todos, en el apéndice A se anexa el código fuente de las funciones más importantes utilizadas para construir las RNA en MATLAB.

De manera adicional, el lector puede encontrar en el siguiente repositorio de GitHub, los códigos presentes en el apéndice A, los *workspaces* de MATLAB (*.mat) en donde se encuentran guardados los modelos de RNA-PMC con las configuraciones más eficientes y los resultados desglosados del análisis estadístico: `https://github.com/ruben1294/biocomp.git`.

Capítulo 1

Introducción

La farmacocinética se define como el estudio de los procesos de absorción, distribución, biotransformación y excreción (ADBE) de un fármaco, mientras que la farmacodinámica se refiere a la relación que existe entre la concentración del fármaco y la respuesta farmacológica traducida en cualquier efecto resultante, incluyendo su intensidad, duración y posibles efectos adversos (Campbell and Cohall, 2017; Kareem and Pathak, 2016). Tanto la farmacocinética como la farmacodinámica son áreas de estudio fundamentales para la evaluación cuantitativa de la eficacia y potencia del efecto farmacológico durante la etapa de desarrollo preclínico de nuevos fármacos (Campbell and Cohall, 2017; Marian and Seghezzi, 2013).

La bomba adenosín trifosfatasa dependiente de sodio y potasio, Na^+/K^+-ATPasa (NKA, EC 3.6.3.9) es una enzima presente en la membrana plasmática de todas las células de mamíferos (Goldstein et al., 2005). Su función consiste en hidrolizar el adenosín trifosfato (ATP) y utilizar la energía disponible para transportar, en contra del gradiente electroquímico, potasio al interior y sodio al exterior de la célula (Goldstein et al., 2005; Svrckova et al., 2017).

La digoxina es un fármaco de tipo glucósido, que corresponde al grupo de los digitálicos, compuestos obtenidos de la planta *Digitalis purpurea*. Estas sustancias han sido utilizadas

por más de 200 años como fármacos para el tratamiento de enfermedades cardíacas en humanos, como fallas cardíacas congestivas y taquiarritmias auriculares (Wilkerson et al., 1980; Yang et al., 2012; Ziff and Kotecha, 2016). Los mecanismos de acción de la digoxina sobre el corazón han sido bien estudiados e identificados; su efecto bioquímico consiste en inhibir reversiblemente la actividad enzimática de transporte e hidrólisis de la NKA en el tejido cardíaco (Dawson and Buckley, 2016; Smith, 1988; Ziff and Kotecha, 2016). Esta inhibición se da gracias a tres mecanismos farmacológicos: hemodinámico, neurohormonal y electrofisiológico (Ziff and Kotecha, 2016). Como mecanismo hemodinámico, la digoxina inhibe el enlace de las subunidades α de la NKA en la membrana de los cardiomiocitos, causando un incremento en la concentración intracelular de sodio, que a su vez estimula la actividad del intercambiador de sodio-calcio, aumentando la concentración intracelular de calcio y por lo tanto la interacción entre las proteínas contráctiles miocárdicas (Ziff and Kotecha, 2016). La respuesta fisiológica del corazón a este efecto hemodinámico consiste en aumentar el máximo de fuerza de contracción del músculo cardíaco y la velocidad con la que se llena la cámara ventricular de fluido sanguíneo, efecto conocido como inotrópico positivo. En segundo lugar, la digoxina induce la activación del nervio vago, llevando de esta manera el equilibrio del control del corazón del sistema simpático hacia el parasimpático, reduciendo así la norepinefrina en el plasma sanguíneo (Flapan et al., 1997; Ziff and Kotecha, 2016). Por último, como efecto electrofisiológico se aumenta el período refractario —un corto período muerto en el que las células cardíacas son incapaces de generar un nuevo potencial de acción cardíaco— en el nodo auriculoventricular. El corazón responde a este aumento reduciendo la frecuencia cardíaca, efecto conocido como cronotrópico positivo (Berman et al., 1980; Dolphen and Lesne, 1980; Smith, 1988; Ziff and Kotecha, 2016). Los digitálicos también son conocidos como cardiotónicos por los efectos que tienen en la actividad cardíaca.

En otro sentido, un modelo se define como una representación abstracta y simplificada de una realidad dada, ya sea existente o planeada. El objetivo de diseñarlos es estudiar y

explicar fenómenos observados (modelos descriptivos) o prever futuros fenómenos (modelos predictivos) (Abar et al., 2017). Los modelos ayudan a visualizar partes de un sistema que son relevantes para responder a una pregunta específica, por lo que el tipo y la complejidad del modelo dependerán directamente de la pregunta que se desea responder (Gross, 2013). Existen tres tipos de modelos: físicos, matemáticos y computacionales. Los modelos computacionales (en adelante, modelos *in silico*) tienen la ventaja de resolver los modelos matemáticos y ser superiores a estos en el sentido de que un modelo *in silico* en ciertas ocasiones no podría haber sido descrito o representado de otra manera más que a través de código (Gross, 2013). Las aplicaciones de los modelos son incontables. Se utilizan en diversas ramas de la ciencia como la biología, economía, ingeniería electrónica, industrial, aeronáutica, aeroespacial, química, farmacéutica, entre otras. Por otra parte, una simulación es la manifestación perceptible de un modelo, representada por un programa de computadora que proporciona ideas sobre un sistema (Abar et al., 2017). Dentro de los modelos *in silico* existen dos tipos: los determinísticos, que producen perfiles de salida idénticos con un estado y condiciones iniciales idénticos; y los estocásticos, que permiten eventos aleatorios o probabilísticos, con los que se puede obtener una familia de perfiles de respuesta en una sola simulación con un estado y condiciones iniciales idénticos.

La almeja pismo (*Tivela stultorum*) es una especie de molusco bivalvo perteneciente a la familia Veneridae. Su distribución geográfica abarca desde Half Moon Bay, California, E.U.A., hasta Bahía Magdalena, Baja California Sur, México, a lo largo de la costa occidental de las tres Californias, así como en Isla Socorro, Colima (Fitch, 1950; Skoglund, 1991). El corazón de *T. stultorum* ha sido validado como modelo biológico para farmacología en bioensayos en los que se ha evaluado su respuesta bioquímica (Cuéllar-Roehri, 1991) y fisiológica (Guerra Rivas, 1994) a compuestos digitálicos, en donde se ha llegado a la conclusión de que dicha respuesta es uniforme.

Los corazones de los moluscos están compuestos de una cámara auricular y una ventri-

cular. Al igual que en los vertebrados, cada cámara está constituida por cardiomiocitos que se contraen simultáneamente y, similar a los mamíferos, los valores de la presión sanguínea varían a lo largo del sistema cardiovascular, alcanzando el valor más alto en el ventrículo. Al igual que en corazones humanos, la superficie endocárdica de los corazones de moluscos es similar a los músculos suaves de mamíferos (Collis et al., 2006; Kodirov, 2011). La frecuencia cardíaca de los moluscos puede llegar a valores comparables a los de humanos (60 latidos por minuto), que puede ser alterada en respuesta a cambios de temperatura y otros factores de estrés o compuestos químicos como metales pesados, acetilcolina y otros (Bini et al., 2006; Kodirov, 2011; Prosser, 1940). Aunado a las razones arriba mencionadas, la almeja pismo tiene un bajo costo de adquisición y es fácil de manipular, por lo que representa una excelente opción como modelo animal de prueba para digitálicos.

El gran avance en la tecnología en las últimas décadas ha permitido aplicarla al estudio y modelado de sistemas naturales. De manera paralela, se ha observado un crecimiento exponencial en la cantidad de datos experimentales y biológicos que se generan cada día; esta situación demanda técnicas eficientes de procesamiento y análisis de información para ayudar a interpretarla y poder extraer conclusiones válidas. Esto justifica el uso de los modelos *in silico* y de las ciencias computacionales en general, que facilitan el procesado de información (Yu and Bagheri, 2016). Otras ramas de la ciencia que han ayudado al procesado de información son la biología de sistemas, la biología sintética y la biología computacional, disciplinas que se complementan entre sí (Gramelsberger, 2013). Cabe resaltar que para que un modelo *in silico* sea válido, debe estar basado, construido o entrenado utilizando información real.

El modelado y simulación de sistemas es una herramienta útil en la optimización de problemas. Su finalidad es encontrar el diseño o la solución ideales a un problema, con el mínimo costo y máximo desempeño, buscando la sustentabilidad, reciclabilidad y eficiencia de energía. Los modelos y las simulaciones también se utilizan para llevar a cabo predicciones y así proyectar el comportamiento de sistemas basándose en suposiciones con condiciones

iniciales específicas (Gross, 2013; Yang et al., 2014). El modelado y simulación aplicado a sistemas y fenómenos biológicos es un área dentro de la biología computacional que se enfoca en medir la expresión global de diferentes componentes de sistemas y procesos biológicos. Esta área resulta ser bastante importante y útil dentro de la bioingeniería, ya que ayuda a entender procesos como cascadas bioquímicas, regulación de expresión de genes, cinética de poblaciones microbianas y cómo se llevan a cabo actividades naturales en general a partir de un todo, además de ayudar a realizar hipótesis sobre fenómenos biológicos y acelerar el diseño de productos como fármacos (Muniyandi and Mohd Zin, 2014; Yu and Bagheri, 2016). En este sentido, el modelado farmacocinético consiste en predecir la concentración y los perfiles temporales del fármaco en el cuerpo después de la administración. El objetivo del modelado farmacocinético/farmacodinámico (FC/FD) es determinar correlaciones importantes entre variables medibles que estén relacionadas con el efecto del fármaco, para obtener la respuesta terapéutica deseada (Kareem and Pathak, 2016).

Al implementar modelos de sistemas biológicos se deben de considerar diversos factores que definirán la complejidad y la salida (o salidas) del modelo, como la pregunta que se desea responder, el contexto del problema, el tipo de respuesta que se desea obtener y los datos experimentales disponibles. Si el modelo es muy simple, es posible que no pueda representar las relaciones subyacentes entre las diversas variables del fenómeno que el conjunto de datos proporciona. En el otro extremo, si es muy complejo o detallado, el conjunto de datos podría no ser compatible con la estructura del modelo o la estimación de los parámetros de salida, lo cual generaría un ajuste perfecto a los datos y con esto la incapacidad de responder fielmente a nuevos datos no mostrados anteriormente, y un modelo innecesariamente difícil de resolver en términos de procesamiento computacional (Yu and Bagheri, 2016). Dada la naturaleza aleatoria que caracteriza a los fenómenos naturales, los modelos estocásticos generalmente son más adecuados para modelar fenómenos biológicos, considerando el ruido y la heterogeneidad inherente de estos sistemas.

Una vez que se construye un modelo, debe ser validado. La validación de un modelo se refiere al proceso de establecer si «el modelo reproduce fiablemente el comportamiento crucial y las cantidades de interés dentro del contexto de uso previsto» (Rykiel, 1996). Este proceso se utiliza para determinar el grado de exactitud con el que el modelo representa el fenómeno real, haciendo énfasis en que se hace desde la perspectiva del uso que se le desea dar al modelo (Gross and MacLeod, 2016). En el modelado y simulación de sistemas biológicos, la validación se considera necesaria y útil para evaluar la bondad de nuevos modelos (Lecca et al., 2016, cap. 1). No existe un procedimiento general para la validación debido a la diversidad de modelos de sistemas biológicos, pero es posible definir dos formas generales de llevarla a cabo. Una es la validación operacional, que consiste en analizar qué tan bien se ajustan los modelos a los datos disponibles acerca del comportamiento del sistema que se desea modelar. Consecuentemente, es posible llevar a cabo esta validación con métodos que midan qué tan robusta y exactamente se simula a un sistema, como el análisis de sensibilidad o el análisis estadístico. El objetivo principal de esta validación es aumentar el desempeño del modelo sin tomar en cuenta necesariamente qué tan bien capture el fenómeno que se desea modelar (Rykiel, 1996). La otra forma es la validación conceptual, que se enfoca en definir si el modelo representa la estructura y propiedades del fenómeno que modela. Esta validación se basa en las teorías, reglas y conocimientos disponibles acerca del fenómeno a modelar para evaluar qué tan bien se captura esta información en las abstracciones e idealizaciones del modelo (Gross and MacLeod, 2016). Es preciso hacer una distinción entre validación y verificación, consistiendo la última en el proceso de determinar si la implementación de un modelo representa fielmente la descripción conceptual que el desarrollador menciona acerca del mismo (Gross and MacLeod, 2016); en otras palabras, si el modelo hace lo que el desarrollador dice que hace.

La computación natural es una rama de las ciencias computacionales que está enfocada en desarrollar herramientas informáticas inspiradas en la naturaleza. Los expertos en esta

área diseñan modelos y técnicas computacionales e investigan, en términos de procesamiento informático, fenómenos que se llevan a cabo en la naturaleza. Debido a su carácter interdisciplinario, la investigación dentro de la computación natural abarca un amplio espectro de metodologías, desde investigación teórica pura y algoritmos, hasta aplicaciones de software a investigación experimental en laboratorio enfocada a la biología, química y física (Rozenberg et al., 2012).

La inteligencia artificial (IA) se ha establecido como un área de la computación dedicada a la producción de software capaz de realizar cálculos sofisticados e inteligentes, similares a los que lleva a cabo el cerebro humano diariamente (Agatonovic-Kustrin and Beresford, 2000; Rozenberg et al., 2012, cap. 10). Esta área de la computación incluye métodos, herramientas y sistemas enfocados a simular los procesos lógicos humanos, de adquisición de conocimiento y de razonamiento para la resolución de problemas.

El aprendizaje automático (en adelante, *machine learning*) es un campo de estudio multifacético de la computación y rama de la IA cuyo objetivo es dotar a las máquinas (específicamente, a las computadoras) con la capacidad de aprendizaje y lograr que razonen inductiva o deductivamente para optimizar, aproximar, reducir, generalizar de ejemplos específicos a reglas generales, clasificar, realizar predicciones, proponer explicaciones y formas de agrupar cosas (Rozenberg et al., 2012, cap. 30), con el objetivo general de automatizar y facilitar tareas (Michalski et al., 1983; Shoham, 1994, cap. 9).

En el área científica es común ajustar series de datos a una función o modelo matemático que sea fácilmente reproducible, con el fin de extrapolar o interpolar información para así predecir muestras sin necesidad de tomar nuevas medidas. Dada una función real de variable real, es posible simplificarla y representarla por una sumatoria de funciones que sean más fáciles de manejar. A este proceso se le conoce como aproximación de funciones (Mateo Jiménez, 2012). Las normas de proximidad más utilizadas son la lineal y la cuadrática, aunque existe una multitud de normas más complejas. La regresión es una tarea en la que se busca

aproximar una función de regresión que sea apropiada para llevar a cabo predicciones de variables de salida continuas (Mateo Jiménez, 2012).

1.1. Regresión múltiple

También conocida en la literatura como regresión multivariable, multiobjetivo o multisalida, en la regresión múltiple existe un conjunto de algoritmos diseñados para predecir simultáneamente múltiples variables de salida (también conocidas como variables objetivo) a partir de un número específico de variables de entrada (Borchani et al., 2015). La regresión múltiple ha tenido aplicaciones en campos muy diversos en sistemas complejos en los que influyen muchas variables y es difícil predecir fenómenos debido a las relaciones no lineales, como el modelado ecológico, la economía o la producción energética (Spyromitros-Xioufis et al., 2016). Existe otro tipo de aproximación de funciones conocido como clasificación, en donde las salidas que se obtienen son discretas. La diferencia entre regresión y clasificación radica en que la primera se usa para predecir fenómenos, por lo tanto sus salidas son continuas; mientras que la segunda se enfoca en etiquetar de acuerdo a una información dada, pudiendo sus salidas ser binarias (clasificación simple) o con más de dos posibles resultados (clasificación multietiqueta).

Actualmente, existen dos grupos generales de algoritmos para la regresión múltiple: a) métodos de adaptación del algoritmo, que toman un algoritmo específico utilizado para una sola salida y lo adaptan para que pueda manejar salidas múltiples; y b) métodos de transformación del problema, los cuales utilizan un algoritmo base de regresión de salida única que construye un modelo para cada variable objetivo y posteriormente se construye el modelo de regresión múltiple utilizando un algoritmo de varias salidas que concatena todas las predicciones. La principal desventaja de estos métodos es que al construir modelos independientes para cada una de las salidas, se ignoran las posibles relaciones que pudiesen existir entre las variables de salida y reducen de esta manera la calidad general de las predicciones (Borchani et al., 2015).

Supongamos que se tiene un conjunto de datos de entrenamiento D, formado por N instancias que contienen un valor asignado para cada variable $X_1, \cdots, X_m, Y_1, \cdots, Y_d$; o lo que es lo mismo, $D = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(N)}, y^{(N)})\}$. Cada instancia está definida por un vector de entrada de m variables descriptivas o predictivas $\mathbf{x}^{(l)} = (x_1^{(l)}, \cdots, x_j^{(l)}, \cdots, x_m^{(l)})$, y un vector de salida de d variables objetivo $\mathbf{y}^{(l)} = (y_1^{(l)}, \cdots, y_i^{(l)}, \cdots, y_d^{(l)})$, con $i \in 1, \cdots, d, j \in 1, \cdots, m$, y $l \in 1, \cdots, N$. El objetivo de la regresión múltiple es encontrar una función h que asigne para cada instancia dada por el vector $\mathbf{x}$, un vector $\mathbf{y}$ de d valores objetivo:

$$h : \Omega_{\mathrm{X}_1} \times \cdots \times \Omega_{\mathrm{X}_m} \longleftrightarrow \Omega_{\mathrm{Y}_1} \times \cdots \times \Omega_{\mathrm{Y}_d}$$

$$\mathbf{x} = (x_1, \cdots, x_m) \longmapsto \mathbf{y} = (y_1, \cdots, y_d),$$

en donde Ω_{X_j} y Ω_{Y_i} son lo espacios de ejemplo para cada variable predictiva X_j, para todo $j \in 1, \cdots, m$, y cada variable objetivo Y_i, para todo $i \in 1, \cdots, d$, respectivamente (Borchani et al., 2015). El modelo construido se puede utilizar para predecir simultáneamente los valores de $\hat{\mathbf{y}}^{N'}$ de todas las variables objetivo de los nuevos conjuntos de instancias $\mathbf{x}^{N'}$ que se pudiesen presentar.

Los algoritmos de *machine learning* resultan ser muy efectivos para llevar a cabo la tarea de regresión múltiple porque consideran las relaciones subyacentes entre las entradas y sus respectivas salidas, además de las relaciones que existen entre una salida y otra, resultando en modelos capaces de representar de manera muy eficaz los conjuntos de datos con salidas múltiples (Borchani et al., 2015).

1.1.1. Algunos algoritmos de *machine learning* para regresión múltiple

El área de *machine learning* provee una amplia gama de algoritmos para aproximar funciones que se pueden aplicar a la regresión múltiple. La selección de los algoritmos a utilizar en este trabajo surge de una revisión en la literatura de los algoritmos del estado del arte de *machine learning*. Se seleccionaron aquellos que mostraban el mejor desempeño en tareas de regresión. Sólo dos trabajos previos se han encontrado en el tema del modelado de moluscos utilizando redes neuronales artificiales (Flores et al., 2017a,b). Con respecto del modelado FC/FD, el algoritmo más utilizado anteriormente ha sido el de las redes neuronales artificiales (Kareem and Pathak, 2016). Con respecto del análisis de la actividad cardíaca, de nuevo las redes neuronales artificiales con diferentes topologías han sido la única técnica utilizada previamente, además de un clasificador basado en relación de equivalencia difusa (Atkov et al., 2012; Chesnokov, 2008; Hosseini et al., 2006; Joo et al., 2012; Jovic and Bogunovic, 2011; Rai et al., 2013; Rajendra Acharya et al., 2003; Silipo et al., 1995). A la fecha no se encontraron trabajos en los que se utilizan el resto de los algoritmos mencionados a continuación para el análisis FC/FD, de la actividad cardíaca o asociados al modelado de la almeja pismo, inclusive, al modelado de moluscos; esta cuestión supone una innovación de este trabajo en el modelado farmacodinámico con técnicas de *machine learning*.

1.1.1.1. Redes neuronales artificiales

Las redes neuronales artificiales (RNA) son modelos abstractos y digitalizados del cerebro humano, diseñados para simular cómo este procesa información (Agatonovic-Kustrin and Beresford, 2000). Las RNA no se acercan en lo absoluto a la complejidad del cerebro humano; sin embargo, existen similitudes entre las redes neuronales artificiales y las biológicas. Las unidades elementales que componen a ambas redes son las neuronas, dispositivos simples

altamente interconectados y que se comunican entre sí, con una capacidad de procesamiento paralelo muy alta. Esta capacidad permite a las RNA reconocer relaciones complejas no lineales y realizar predicciones acertadas en áreas muy diversas (Agatonovic-Kustrin and Beresford, 2000; Mateo Jiménez, 2012). A diferencia de otros métodos computacionales, las RNA no son programadas, sino que obtienen la capacidad de predicción a partir de un proceso de aprendizaje, mediante un entrenamiento que consiste en mostrar datos asociados a una respuesta a la red (Ponce Cruz, 2010). Los datos mostrados a la red se conocen como entradas, que son multiplicadas por los pesos asociados a cada neurona, lo cual genera una señal. Esta señal pasa a través de una función de transferencia (también conocida como función de activación) definida para cada neurona, a la siguiente capa de la red. Cuando la señal llega a la última neurona, el valor obtenido es comparado con el valor esperado y se calcula el error; después los pesos son modificados de manera que se reduzca el error. Este proceso es repetido hasta que el error es el mínimo.

Suponiendo que se tiene un modelo de una sola neurona con una entrada y una salida, la ecuación que describe al sistema se define como $a = f(wp + b)$, en donde f es la función de activación que se aplica sobre la entrada p para generar la salida escalar a de la neurona; w es el peso de conexión escalar de la neurona, el cual se multiplica por p para formar wp; y b es el sesgo o error (del inglés *bias*) asociado a esa neurona (Hagan et al., 2014). Generalmente, una neurona posee varias entradas, en cuyo caso la ecuación del sistema con R entradas se transforma en $a = f(\mathbf{Wp} + b)$, en donde $\mathbf{W}$ representa la matriz de los pesos de las entradas y $\mathbf{p}$ corresponde a las entradas $p_1, p_2, \cdots, p_R$. Por otra parte, un modelo de una sola neurona generalmente no es suficiente para llevar a cabo procesos de optimización, automatización, predicción o clasificación, ya que la simplicidad del modelo no permite manejar las variables y relaciones típicas de esta clase de problemas. Una red de S neuronas y R entradas se representa por la ecuación $\mathbf{a} = \mathbf{f}(\mathbf{Wp} + \mathbf{b})$. El vector de salida $\mathbf{a}$ está formado por las salidas $a_i, \cdots, a_S$ correspondientes a cada neurona, mientras que $\mathbf{f}$ corresponde al vector que agrupa

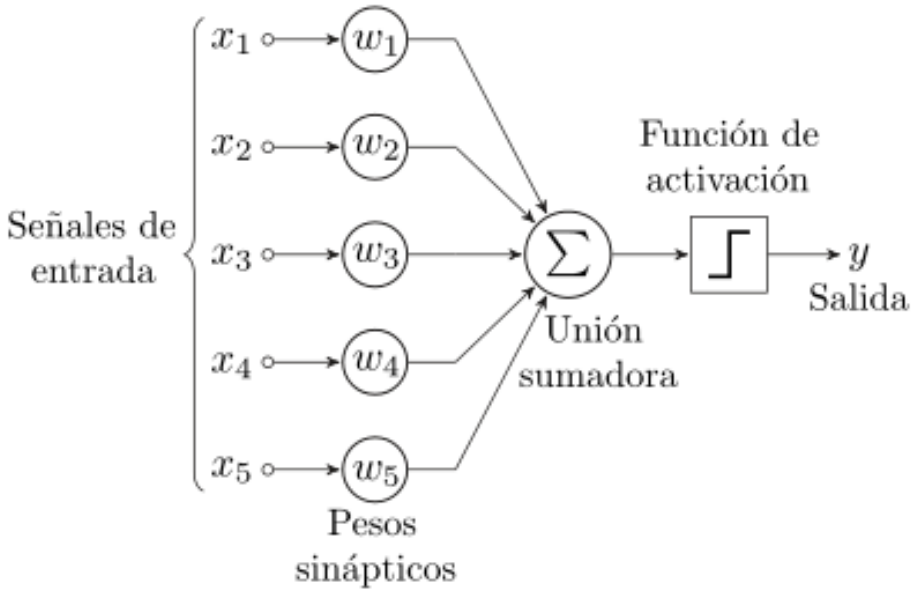

Figura 1.1: El perceptrón, unidad elemental de las redes neuronales artificiales que realiza la suma de los valores numéricos de las entradas para generar la salida.

a las funciones de activación, **b** representa el vector de b_i errores en el cual i señala la neurona correspondiente, y **W** de nuevo es la matriz de $w_{S,R}$ pesos de las entradas, en donde el subíndice S indica la neurona y R la salida.

El enfoque de esta tesis es el de aplicar métodos de IA al modelado farmacodinámico. Si el lector desea profundizar en el fundamento matemático de las RNA, se sugiere consultar las referencias Hagan et al. (2014), Mateo Jiménez (2012) y Haykin (1999).

Una de las topologías de RNA más utilizadas es la del perceptrón multicapa (PMC), en el cual la unidad básica es el perceptrón (Figura 1.1). Esta unidad suma todas las señales de entrada y las multiplica por la suma de los pesos previamente inicializados de manera aleatoria. Al principio el perceptrón no es capaz de distinguir patrones complejos en los datos de entrada, sin embargo adquiere esta capacidad a través del entrenamiento, el cual consiste en presentar nueva información a la red asociada a su respectiva salida (Agatonovic-Kustrin and Beresford, 2000; Ponce Cruz, 2010).

1.1.1.2. *Single Target*

En este algoritmo base, se construye un modelo h de salidas múltiples compuesto de m modelos de salida única $h_j : \mathcal{X} \rightarrow R$, en donde $\mathcal{X} = R^d$ es el dominio de X, el vector de las variables de entrada, y R representa el espacio de la variable Y a predecir. Cada modelo h_j está entrenado con un conjunto de datos de entrenamiento transformado $D_j = (x^1, y_j^1), \cdots, (x^n, y_j^n)$ para así predecir el valor de la única variable objetivo Y_j. Las posibles relaciones que puedan existir entre las salidas no pueden ser explotadas con este método (Spyromitros-Xioufis et al., 2012).

1.1.1.3. *Stacked Single-Target*

En 2016, un equipo de investigadores expertos en *machine learning* modificó dos algoritmos que anteriormente se aplicaban en la clasificación multietiqueta (SST, de las siglas en inglés de *Stacked Single-Target*, y ERC, del inglés *Ensemble of Regressor Chains*), adaptándolos para tareas de regresión múltiple. Esta innovación se llevó a cabo usando las variables de salida como entradas adicionales, generando una especie de meta-modelos. Posteriormente, realizaron una comparación de sus novedosos algoritmos contra otros algoritmos del estado del arte de la regresión múltiple (Spyromitros-Xioufis et al., 2016). El algoritmo SST está basado en el algoritmo *Stacked Binary Relevance*, diseñado por Godbole and Sarawagi (2004). En el trabajo de Spyromitros-Xioufis et al. (2016), la innovación consistió en adecuar este método a la regresión de salidas múltiples. El entrenamiento está dividido en dos etapas: en la primera, se construyen m modelos independientes de salida única $h_j : \mathcal{X} \rightarrow R$ (Spyromitros-Xioufis et al., 2016), la cual es la primera etapa de la versión de una sola salida de este método, el algoritmo *Single-Target* (ST). La segunda etapa de entrenamiento es la adecuación a salidas múltiples, en donde se construye un segundo conjunto de meta-modelos $h'_j : \mathcal{X} \times R^m \rightarrow R$, uno para cada variable de salida Y_j. Se les conoce como meta-modelos ya que se entrenan

con vectores expandidos; los vectores de entrada ($\mathbf{x}^i$) utilizados para el entrenamiento se aumentan estimando los valores de sus variables de salida ($\hat{y}_1^i, \cdots, \hat{y}_m^i$) para generar los vectores expandidos $\mathbf{x}' = [\mathbf{x}^i, \hat{y}_1^i, \cdots, \hat{y}_m^i]$.

1.1.1.4. *Ensemble of Regressor Chains*

La versión para clasificación de este algoritmo se conoce como cadenas de regresión (del inglés *regressor chains*). Su entrenamiento consiste en elegir una cadena aleatoria del conjunto de las variables de salida y construir un modelo de regresión separado para cada salida. En un contexto hipotético se selecciona la cadena $C = Y_1, Y_2, \cdots, Y_m$, en donde C representa un conjunto ordenado. El primer modelo se enfocaría en la predicción Y_1 y tendría la forma $h_1 : \mathcal{X} \rightarrow R$. Los siguientes modelos $h_j (j > 1)$ aprenden con los conjuntos de entrenamiento transformados $D'_j = (\mathbf{x'}^1_j, y_j^1), \cdots, (\mathbf{x'}^n_j, y_j^n)$, en donde los vectores de entrada originales han sido aumentados por los valores actuales de todas las variables objetivo anteriores de la cadena para formar los vectores de entrada expandidos $\mathbf{x'}^i_j = [x_1^i, \cdots, x_d^i, y_1^i, \cdots, y_{j-1}^i]$.

1.1.1.5. *Random Linear target Combinations*

El algoritmo de combinaciones aleatorias lineales de objetivos (RLC por sus siglas en inglés) consiste en crear nuevas variables objetivo al considerar combinaciones lineales aleatorias de k variables objetivo originales (Tsoumakas et al., 2014). Considérese un conjunto de p variables de entrada $\mathbf{x} \in R^p$ y un conjunto de q variables objetivo $\mathbf{y} \in R^q$, en donde R^p y R^q son los dominios de los conjuntos $\mathbf{x}$ y $\mathbf{y}$, respectivamente. Considérese también que se tiene un conjunto de m datos de entrenamiento: $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x^{(i)}}, \mathbf{y^{(i)}})\}_{i=1}^m$, en donde $\mathbf{X}$ y $\mathbf{Y}$ son matrices de tamaño $m \times p$ y $m \times q$, respectivamente. El algoritmo RLC construye $r >> q$ nuevas variables objetivo utilizando combinaciones lineales aleatorias de $\mathbf{y}$ (Tsoumakas et al., 2014).

1.1.1.6. Algoritmos de salida única *Support Vector Regression* y *Bagged regression trees*

En la metodología propuesta para este trabajo se utilizaron estos dos algoritmos de salida única como base para los algoritmos de regresión múltiple. El algoritmo *Support Vector Regression* (SVR por sus siglas en inglés) se eligió debido a la sugerencia de varios revisores, como se explica en el prefacio. La versión utilizada es la regularizada L2, diseñada por Drucker et al. (1997). La representación de este método se da por medio de la ecuación $F(x, \hat{w}) = \sigma_{i=1}^{N}(\alpha_i^* - \alpha_i)(v_i^t x + 1)^P + b$, en donde x es el vector de entrada con N instancias, $\hat{w}$ es el vector w que se busca estimar y que tiene el error mínimo posible, v^t es el subconjunto de entrenamiento, el superíndice t indica la transpuesta del vector v, α^* y α son constantes, y b es el error.

El término *bagging* proviene de un algoritmo diseñado por Breiman (1996). Supongamos que se tiene un conjunto L de datos $(y_n, x_n), n = 1, \cdots, N$, en donde se busca predecir a y a partir de $\varphi(x, L)$. Este método toma muestras repetidas de *bootstrap* de L, para realizar la predicción utilizando $\varphi(x, L^{(B)})$. Este proceso se puede describir con la ecuación $\varphi_B(x) = av_B\varphi(x, L^{(B)})$. El acrónimo *bagging* se refiere a "***b**ootstrap **agg**regat**ing***"(Breiman, 1996). El algoritmo *Bagged regression trees* (en adelante, BAG) es un acoplamiento del método *bagging* sobre los árboles de regresión, y se eligió debido a que es un algoritmo base con uno de los mejores desempeños actualmente cuando se acopla a algoritmos de salida múltiple. Además, permite extraer aspectos importantes aún cuando la información con la que se trabaja tiene mucho ruido; y posee una elevada robustez, ya que su desempeño no se reduce cuando se utiliza con uno u otro algoritmo de salida múltiple (Spyromitros-Xioufis et al., 2016).

1.2. Justificación

Según cifras de la Organización Mundial de la Salud (OMS), las enfermedades cardiovasculares son la principal causa de defunción en todo el mundo (OMS, 2017). Para combatir estos padecimientos, una de las propuestas es el desarrollo de nuevos fármacos que deben ser probados en organismos modelo antes de ser utilizados en humanos. La elección del organismo modelo a utilizar depende del estudio que se debe de llevar a cabo.

El modelado y simulación *in silico* ha sido utilizado en el análisis FC/FD desde hace varios años debido a las ventajas que aporta, como la optimización de concentraciones y bioprocesos de producción, predicción de diagnóstico y prognóstico en pacientes, mejor entendimiento de los fenómenos *in vivo*, y la predicción de la eficacia terapéutica y efectos adversos de fármacos (Siepmann, 2013), entre otras. Una de las ventajas más notables que ofrece el modelado en la farmacodinámica y la farmacocinética es el ahorro de recursos durante el desarrollo de nuevos potenciales fármacos (Dingemanse and Krause, 2017). En la farmacología se observa un interés creciente y una tendencia en desarrollar modelos farmacodinámicos basados en la fisiología, disciplina que se encuentra en una etapa temprana y cuyo progreso podría tener un impacto significativo en la farmacología y en el desarrollo de nuevos fármacos (Dingemanse and Krause, 2017).

Las RNA han sido extensamente utilizadas en la investigación farmacológica, principalmente en el modelado FC/FD (Kareem and Pathak, 2016), un área en la que han tenido aceptación y un excelente desempeño. Específicamente se han utilizado para la predicción de respuesta farmacológica y análisis FC/FD (Belič et al., 2005; Chen et al., 1999, 2011; Gobburu and Chen, 1996; Haidar et al., 2002; Mager et al., 2005; Takayama et al., 2003), así como en aplicaciones clínicas a partir del perfil farmacodinámico y farmacocinético de pacientes (Albert et al., 2010; Camps-Valls et al., 2000, 2003; De Matas et al., 2010; Masetic and Subasi, 2016; Yamamura, 2003). Trabajos recientes han demostrado un buen desempeño de las RNA en el

modelado farmacodinámico aplicado a la almeja pismo con digoxina (Flores et al., 2017a,b). Otros algoritmos para regresión de *machine learning* han sido utilizados, en menor medida, en aplicaciones en farmacología como predicción de niveles de actividad de inhibidores protéicos, activadores, bloqueadores y sustratos enzimáticos, predicción de actividad de acuerdo a la estructura molecular, etcétera; es preciso mencionar a las RNA, SVM, regresión logística, *k-Nearest Neighbor* (kNN), y a los árboles de decisión como algoritmos comúnmente utilizados para estos fines (Li et al., 2007)

Por otra parte, a lo largo de muchos años se han utilizado moluscos (Cuéllar-Roehri, 1991; Guerra Rivas, 1994), gallinas (Wernke and Cacini, 1990), conejillos de indias (Del Valle-Mondragón et al., 2008; Schäfer et al., 1985), ratas (Eyer et al., 2012; Goldstein et al., 2005) y otras especies, ya sea animales completos o sus tejidos, para realizar experimentos de cuantificación de parámetros de farmacocinética y farmacodinámica de digitálicos. Estos ensayos involucran un sacrificio considerable de animales para lograr este propósito.

Un modelo *in silico* que prediga la cantidad de fármaco necesaria para obtener una respuesta fisiológica específica podría complementar a los modelos *in vivo*; de esta manera se podría ahorrar tiempo y dinero al reducir el sacrificio de animales y los experimentos en laboratorio. Por esta razón, una predicción aproximada de la respuesta farmacológica a la digoxina, reflejada en la actividad cardíaca de ventrículos de *T. stultorum*, puede ser de gran interés en el área de la farmacología y la biosimulación. Debido a que los modelos son una abstracción de algo más complejo, la predicción de la actividad cardíaca puede realizarse a partir de cuatro parámetros característicos que se ven directamente afectados por el compuesto cardiotónico digoxina: el máximo de contracción muscular, el mínimo de contracción muscular, la frecuencia cardíaca y la velocidad con la que la cámara ventricular se llena de sangre. En el capítulo 2 se hablará en detalle al respecto.

El objetivo de este trabajo, es realizar una comparación entre técnicas de *machine learning* previamente utilizadas en el modelado farmacológico contra técnicas que no han

sido previamente utilizadas en esta área, para desarrollar modelos *in silico* que representen la relación farmacodinámica entre la dosis administrada de digoxina y la respuesta fisiológica en la actividad cardíaca del ventrículo de la almeja pismo.

1.3. Pregunta de investigación

¿Cuál es el mejor algoritmo, en términos de capacidad de predicción, para el modelado de la respuesta fisiológica de la almeja *Tivela stultorum* al fármaco digoxina?

1.4. Objetivos

1.4.1. Objetivo general

Utilizar las redes neuronales artificiales en comparación con otras técnicas de *machine learning* para desarrollar modelos farmacodinámicos que describan la relación que existe entre la concentración de digoxina adicionada al medio de perfusión y la respuesta fisiológica reflejada en la actividad cardíaca del ventrículo de *Tivela stultorum*.

1.4.2. Objetivos específicos

- Utilizar las redes neuronales artificiales con topología perceptrón multicapa y determinar la configuración óptima, de acuerdo al error medio cuadrado mínimo, para el modelado farmacodinámico de corazones de almeja *Tivela stultorum*
- Conocer y seleccionar algoritmos relevantes en la literatura asociados al modelado farmacocinético y farmacodinámico, al análisis de la actividad cardíaca y al modelado de moluscos

- Comparar y seleccionar la mejor técnica de *machine learning*, de acuerdo al promedio de la raíz del error medio cuadrado relativo más bajo, para predicción en farmacodinámica con el conjunto de datos utilizado en este trabajo

1.5. Hipótesis

Las redes neuronales artificiales son el algoritmo de *machine learning* con el mejor desempeño en predicción de la relación farmacodinámica entre la cantidad de digoxina administrada a ventrículos de *Tivela stultorum* y su respuesta farmacológica reflejada en la actividad cardíaca.

Capítulo 2

Metodología

La metodología comienza con una limpieza y extracción de los datos relevantes del total proporcionado. Posteriormente, se realiza un pretratamiento a los datos, con el fin de mejorar la capacidad de predicción del modelo; se construyen los modelos con las técnicas mencionadas, y finalmente, se hace un análisis de la capacidad de predicción y exactitud del modelo. El capítulo termina con el análisis estadístico utilizado.

2.1. Conjunto de datos

Los experimentos de los que se obtuvieron los datos consistían en colocar un ventrículo, previamente extirpado, de *T. stultorum* en un medio de perfusión (agua de mar sintética), colgado de un transductor que medía con cuánta fuerza (en gramos) se contraía el ventrículo con respecto del tiempo. Este transductor medía a una frecuencia de 1 kHz, por lo que cada segundo de muestreo contenía mil datos de fuerza. Lecturas de la actividad cardíaca de 25 ventrículos de *T. stultorum* fueron proporcionadas por el Laboratorio de Farmacología Marina de la Facultad de Ciencias Marinas, en donde se midió la fuerza de contracción del músculo cardíaco en función de las condiciones del medio (con y sin digoxina). Debido a que

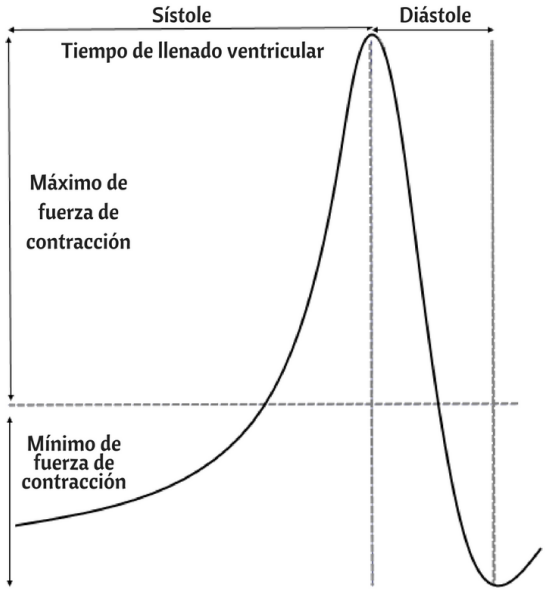

Figura 2.1: Ejemplo de un latido del ventrículo de *T. stultorum*.

la contracción cardíaca puede ser afectada por la temperatura, todos los experimentos se realizaron a 17 ± 1°C. En la Figura 2.1 se muestra un latido de un ventrículo de *T. stultorum*, en donde se representan parámetros importantes de la actividad cardíaca.

2.2. Preprocesado de la información

Se exportaron las lecturas de archivos de LabChart 8 Reader (ADInstruments Pty Ltd., Bella Vista, New South Wales, Australia) a archivos de texto (*.txt), y se importaron al software MATLAB 8.5 R2015a (MathWorks Inc., Natick, MA, E.U.A.). Se separaron en tres vectores, uno para el tiempo y dos para los canales de lectura de LabChart. Para reducir el tiempo de procesamiento computacional, el tipo de datos fue cambiado de *double* a *int32*, con lo que se redujo cada dato de 64 a 32 bits; para poder realizar este cambio se escalaron los datos a enteros. Las regiones en blanco o ilegibles fueron eliminadas. Las lecturas contenían más de un millón de datos en crudo. Para reducir la cantidad de información se realizó un muestreo de la señal de acuerdo al teorema de Nyquist-Shannon para el muestreo de señales biológicas, el

cual afirma que una señal puede ser totalmente reconstruida a partir de una señal discreta si la frecuencia de muestreo es igual o mayor a un dato muestreado por cada dos datos originales. En la práctica se tomó uno de cada 100 datos, resumiendo de esta manera la cantidad de datos de 1,000 a 10 por segundo.

Se extrajeron los parámetros biofísicos relevantes de la información de acuerdo a Guerra Rivas (1994) utilizando un código escrito en MATLAB. Cada máximo local se tomó como un máximo de la contracción, y cada mínimo local como un mínimo de la contracción; el tiempo de llenado se tomó como el tiempo que tarda la cámara ventricular en llenarse de fluido sanguíneo y vaciarse, esto es, el tiempo que transcurre entre un latido y otro; y la frecuencia cardíaca se tomó como la cantidad de latidos que ocurren en un minuto. El resto de las variables de entrada fueron proporcionadas de los experimentos *ex vivo*. Se utilizaron 10 variables de entrada para construir los modelos: máximo de fuerza de contracción ventricular, mínimo de fuerza de contracción ventricular, tiempo de llenado ventricular y frecuencia cardíaca antes de la adición de digoxina, peso del corazón, volumen del corazón, largo del corazón, ancho del corazón, concentración de digoxina y volumen utilizado para la dilución de digoxina; mientras que las salidas fueron el máximo de contracción, mínimo de contracción, tiempo de llenado y frecuencia cardíaca que se obtendrían después de agregar la digoxina (Tabla 2.1). Se construyó un conjunto de 3,975 x 14 datos con esta información.

Las entradas se estandarizaron a valores de z (media = 0, desviación estándar = 1) mediante la Ecuación 2.1:

$$z_n = \frac{x_n - \overline{X}}{\sigma_x} \tag{2.1}$$

en donde z_n es el valor estandarizado de la observación n, x_n es el valor original de la observación n, y $\overline{X}$ y σ_x son la media y la desviación estándar de la variable X, respectivamente. Los datos de salida fueron normalizados de una forma centralizada en el rango [-1, 1] con la

Ecuación 2.2:

$$X_i = \frac{x_i - \left(\frac{x_{max} + x_{min}}{2}\right)}{\left(\frac{x_{max} + x_{min}}{2}\right)} \tag{2.2}$$

donde X_i es el valor normalizado de la observación i, x_i es el valor original de la observación i, y x_{min} y x_{max} son el valor mínimo y máximo, respectivamente, del conjunto de datos x. La estandarización y normalización de las variables de entrada y salida se realizan para reducir los diferentes rangos de medida de cada variable, de modo que el mismo porcentaje de cambio en la suma de los pesos de entrada provoque el mismo porcentaje de cambio en la salida (Olden and Jackson, 2002).

Variable	Entrada/salida	Definición
maxIN	Entrada	Máximo valor de fuerza de contracción que alcanzó el ventrículo durante la sístole en ese latido.
minIN	Entrada	Mínimo valor de fuerza de contracción que alcanzó el ventrículo durante la diástole en ese latido.
timeIN	Entrada	Tiempo que tardó el ventrículo en llenarse de sangre en ese latido.
bpmIN	Entrada	Frecuencia cardíaca promedio del minuto en el que ocurrió ese latido.
weight	Entrada	Peso del ventrículo usado en ese experimento.
VolHeart	Entrada	Volumen del ventrículo usado en ese experimento.
Height	Entrada	Largo del ventrículo usado en ese experimento.
Wide	Entrada	Ancho del ventrículo usado en ese experimento.
VolDig	Entrada	Volumen de líquido utilizado para diluir la digoxina agregada a ese ventrículo.
ConcDig	Entrada	Concentración de digoxina agregada al ventrículo.
maxOUT	Salida	Máximo valor de fuerza de contracción que alcanzó el ventrículo durante la sístole en ese latido, posterior a la adición de digoxina.
minOUT	Salida	Mínimo valor de fuerza de contracción que alcanzó el ventrículo durante la diástole en ese latido, posterior a la adición de digoxina.
timeOUT	Salida	Tiempo que tardó el ventrículo en llenarse de sangre en ese latido, posterior a la adición de digoxina.
bpmOUT	Salida	Frecuencia cardíaca promedio del minuto en el que ocurrió ese latido, posterior a la adición de digoxina.

Tabla 2.1: Definición de las variables de entrada y salida utilizadas en la construcción de los modelos.

2.3. Construcción de modelos farmacodinámicos

2.3.1. Predicción utilizando redes neuronales artificiales

2.3.1.1. Entrenamiento

Para las neuronas de la capa oculta, se utilizó la tangente hiperbólica como función de activación, descrita por la Ecuación 2.3:

$$f(p) = \frac{2}{1 + e^{-2p}} - 1 \tag{2.3}$$

en donde p es cada dato de entrada de la red. Esta es una función sigmoide que ayuda a que las RNA aprendan más rápido en relación con otras funciones, y se utiliza cuando la normalización de los datos va de -1 a 1 (Mateo Jiménez et al., 2009; Ponce Cruz, 2010). Para las capas de entrada y de salida se utilizó una función lineal de activación.

La topología de PMC que se utilizó se describe como $10 - N - 4$, en donde 10 y 4 son el número de entradas y salidas, respectivamente, y N es el número de neuronas en la capa oculta del PMC, el cual es un número par que va de 2 a 30.

Para entrenar a las RNA-PMC se utilizó la información obtenida de los experimentos *ex vivo*. El algoritmo utilizado para el entrenamiento fue el de retropropagación (del inglés *backpropagation*), uno de los algoritmos más utilizados para el entrenamiento de RNA-PMC debido a su efectividad (Agatonovic-Kustrin and Beresford, 2000; Flores et al., 2017a,b; Hagan et al., 2014; Mateo Jiménez, 2012; Mateo Jiménez et al., 2011; Ponce Cruz, 2010). Este algoritmo consiste en propagar el error de las salidas de vuelta a todas las neuronas, de manera que los pesos se actualizan de adelante hacia atrás (Erb, 1993). Se utilizaron tres variantes de este algoritmo para comparar su desempeño y seleccionar la mejor configuración: Levenberg-Marquardt (LM), *Bayesian Regulation* (BR) y *Scaled Conjugated Gradient* (SCG).

2.3.1.2. Validación

Para validar las RNA se utilizó el método de submuestreo aleatorio (del inglés *random sub-sampling*), un tipo de validación cruzada que consiste en dividir de manera aleatoria los datos en subconjuntos para entrenar y probar las redes, diferentes para cada iteración (Mateo Jiménez et al., 2011). Se hace énfasis en que este método se aplicó a todas las RNA. A su vez, se utilizó un método conocido como validación *hold-out* para el cual se utilizó el criterio de parada *early-stopping*, que consiste en detener el entrenamiento de la red cuando el error del subconjunto de validación deja de disminuir y comienza a aumentar (Hagan et al., 2014; Mateo Jiménez, 2012; Mateo Jiménez et al., 2011). Para esta validación, el conjunto de datos se dividió en tres subconjuntos: 50 % para el entrenamiento (*train*), 25 % para validación (*validation*) y el 25 % restante para probar el desempeño de la red (*test*). Cuando no se utilizó la validación *hold-out*, las redes se entrenaron hasta llegar a 100 épocas —se les conoce así a las iteraciones pertenecientes al entrenamiento de la red— utilizando el 75 % de los datos, mientras que el 25 % restante se utilizó para probar la red. Con el fin de evitar una elección sesgada de datos para el entrenamiento, cada arquitectura de red diferente con cada algoritmo se corrió 20 veces con y sin validación *hold-out*, obteniendo un total de 1,800 RNA-PMC; posteriormente se eligió la mejor red de las 20 con cada configuración de acuerdo al error medio cuadrado mínimo (MSE). Este proceso puede entenderse como un método adicional de validación cruzada conocido como validación k-fold; para este caso fue 20-fold.

2.3.1.3. Evaluación

Para evaluar el desempeño y la precisión de la predicción del modelo se utilizaron los parámetros de error medio cuadrado del subconjunto de datos de prueba (MSE_{test}), la raíz del error medio cuadrado del subconjunto de datos de prueba ($\text{RMSE}_{\text{test}}$), el promedio de la raíz del error medio cuadrado relativo del subconjunto de datos de prueba ($\text{aRRMSE}_{\text{test}}$) y los

coeficientes de regresión R^2, de acuerdo a los trabajos de Lou and Nakai (2001), García-Gimeno et al. (2005), Mateo Jiménez et al. (2011), y Spyromitros-Xioufis et al. (2016). Para evaluar la exactitud de diagnóstico del modelo, se llevó a cabo una prueba de predicción, introduciendo cada fila de entradas del conjunto de datos a los modelos construidos (Babaoglu et al., 2009); las predicciones de las salidas resultantes fueron comparadas con el valor real obtenido en los ensayos. Se tomó la predicción como verdadera si el error entre el valor predicho (vp) y el valor observado (vo) es menor al 5 % (error relativo). Este procedimiento se describe en la Ecuación 2.4:

$$\begin{aligned} \text{exactitud} &= \frac{find_n(re_i) \leq 5\,\%}{n} \\ re_i &= \left|\frac{vp - vo}{vo}\right| * 100\,\% \\ n &= 3,975 \end{aligned} \tag{2.4}$$

donde la función $find_n$ cuenta el número de veces que se cumplió la condición, es decir, la cantidad de valores re_i que fueron menores o iguales al 5 %. Este parámetro fue calculado para cada variable de salida.

Se utilizó el Neural Network Toolbox™ de MATLAB 8.5 R2015a para construir las RNA-PMC. El entrenamiento se optimizó de acuerdo con el criterio predeterminado del Toolbox, el cual asume que mientras más bajo sea el MSE, el modelo simula mejor la información.

2.3.1.4. Análisis de sensibilidad

Para evaluar las contribuciones relativas (también reportada en la bibliografía como importancia o influencia relativa) de las entradas sobre las salidas, se realizó un análisis de sensibilidad. Este método consiste en calcular la magnitud de la contribución de las entradas

sobre las salidas (Mateo Jiménez et al., 2009). Para obtener los valores de las contribuciones se usan los pesos de las neuronas y se asocian con las variables de entrada (Chen et al., 2011). Las variables con una importancia relativa baja pueden ser omitidas del modelo sin tener una pérdida significativa del desempeño del mismo, resultando en redes con menor número de variables de entrada y por lo tanto un tiempo de procesamiento más bajo para ser resuelta (Mateo Jiménez et al., 2011). Para llevar a cabo esta evaluación, se tomaron las seis configuraciones con el $\mathrm{MSE}_{\mathrm{test}}$ más pequeño y se realizaron 20 corridas independientes de cada configuración. A estas nuevas RNA se les evaluó las contribuciones relativas. El porcentaje relativo se promedió para cada configuración utilizando los valores de las 20 corridas, para evitar máximos o mínimos locales en los valores de las contribuciones. Se utilizó el algoritmo de Garson (1991), posteriormente modificado por Goh (1995), para evaluar las contribuciones relativas, el cual está descrito por la Ecuación 2.5:

$$Q_{ik} = \frac{\sum_{j=1}^{L} \left(\frac{|w_{ij}|}{\sum_{m=1}^{N} |w_{mj}|} |p_{jk}| \right)}{\sum_{i=1}^{N} \left(\sum_{j=1}^{L} \left(\frac{|w_{ij}|}{\sum_{m=1}^{N} |w_{mj}|} |p_{jk}| \right) \right)} \tag{2.5}$$

en donde Q_{ik} es el porcentaje de influencia de la variable de entrada x_i sobre la salida y_k, $\Sigma_{m=1}^{N} w_{mj}$ es la suma de los pesos de las conexiones entre las neuronas de entrada N y las neuronas de la capa oculta j, w_{ij} es el peso de conexión entre la neurona de entrada i y la neurona oculta j; y p_{jk} es el peso de conexión entre la neurona oculta j y la neurona de salida k.

La construcción de las RNA y la evaluación de las contribuciones relativas se realizaron con códigos hechos en MATLAB, modificados de Flores et al. (2017a,b).

2.3.2. Regresión utilizando otras técnicas de *machine learning*

2.3.2.1. Entrenamiento

Se utilizaron los algoritmos SST (variante *train* y *cv*), ERC (variante *train* y *cv*), RLC y ST para construir los modelos de regresión, utilizando como algoritmos base BAG y SVR. Para llevar a cabo los experimentos con estos algoritmos se utilizó la clase MTRExperimentMLJ.java, de la versión extendida de la biblioteca para *machine learning* MULAN, un programa *open source* basado en Java y desarrollado por Spyromitros-Xioufis et al. (2016), el cual puede ser descargado desde su repositorio en GitHub (`https://github.com/lefman/mulan-extended.git`). El software utilizado como entorno de programación fue Eclipse Neon 3.0 (Eclipse Foundation Inc., Ontario, Ottawa, Canadá).

2.3.2.2. Validación

Para validar los modelos se utilizó el método de validación cruzada 10-*fold*, de acuerdo a la metodología expuesta por Spyromitros-Xioufis et al. (2016).

2.3.2.3. Evaluación

Como parámetro de evaluación se utilizó el RRMSE de cada salida y el promedio de todas (aRRMSE), de acuerdo a Spyromitros-Xioufis et al. (2016). Adicionalmente, se registró el tiempo de procesamiento computacional requerido para construir cada uno de los modelos de regresión entrenados con los algoritmos seleccionados de *machine learning*.

2.4. Análisis estadístico

Se realizó una prueba estadística t de Student para verificar que el conjunto de datos muestreado era estadísticamente igual al original, es decir, que no había una pérdida de

información. Se eligió la hipótesis nula como $\mu_1 - \mu_2 = 0$, en donde μ_1 es el promedio de N de las lecturas originales de los corazones, mientras que μ_2 es el promedio de n de las lecturas muestreadas de los corazones.

Se realizó una prueba t para comparar los valores reales contra los predichos, y un análisis multifactorial de varianza (ANOVA) para confirmar los resultados del análisis de las contribuciones relativas. Todas las pruebas estadísticas se realizaron con el software Minitab 17 (Minitab Inc., State College, PA, E.U.A.).

Capítulo 3

Resultados

A continuación se presenta, a manera de una representación gráfica, resultados de los experimentos `ex vivo` *de los cuales se obtuvieron los datos utilizados para construir los modelos. Posteriormente, se muestran los parámetros de desempeño y análisis de los modelos construidos con las técnicas de inteligencia artificial.*

3.1. Actividad cardíaca

En la Figura 3.1 se muestra un ejemplo de la señal obtenida a partir de la actividad cardíaca de uno de los ventrículos. La barra roja indica la adición de digoxina, después de la cual se puede observar el efecto sobre la actividad cardíaca. Esta figura muestra la actividad un minuto antes de la adición del fármaco (eje x), posteriormente se deja pasar un tiempo (durante el cual el máximo y mínimo de contracción van reduciendo su intensidad gradualmente) y se muestra un minuto de actividad posterior a la adición, en donde se observa un cambio significativo principalmente en el máximo y mínimo de contracción. En el eje y se observa la fuerza de contracción en gramos.

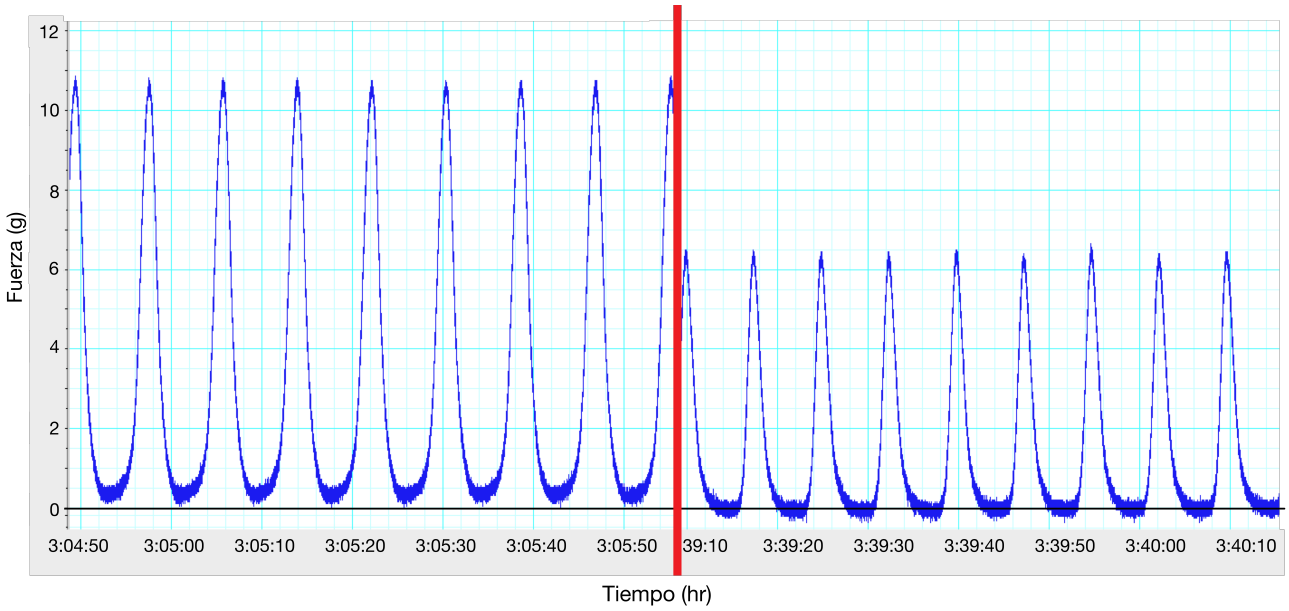

Figura 3.1: Lectura de la actividad cardíaca de uno de los corazones utilizados en los experimentos *ex vivo*.

3.2. Análisis estadístico

La Tabla 3.1 muestra los resultados de la prueba t aplicada al conjunto de datos original y granulado. Se logró una reducción considerable en la cantidad de datos sin tener una pérdida de la representatividad del total de los datos.

3.3. Desempeño de los modelos farmacodinámicos

En la Tabla 3.2 se muestran los valores de MSE, RMSE y aRRMSE de las mejores configuraciones de PMC, parámetros que determinan el desempeño del modelo en general. Las mejores RNA fueron seleccionadas de acuerdo al menor MSE_{test} de cada combinación de configuraciones.

En cuanto al análisis por salidas, se muestra en la Tabla 3.3 los valores de R^2_{test}, exactitud al 5 % y RRMSE, parámetros de desempeño, capacidad de predicción y exactitud por cada

Tabla 3.1: Prueba t de Student para los conjuntos de datos originales y muestreados.*

Corazón	N[a]	n[b]	Valor de t	Valor de p
1	1507010	15070	0.04	0.966
2	3860002	38600	-0.02	0.984
3	2426002	24260	0.07	0.942
4	2328694	23287	-0.12	0.904
5	3007002	30070	0.01	0.993
6	2654002	26540	0.00	0.998
7	1310002	13100	-0.10	0.918
8	1432003	14320	0.07	0.946
9	3494002	34940	0.15	0.879
10	2635002	26350	0.00	0.998
11	2819002	28190	0.00	0.997
12	1256002	12560	0.09	0.929
13	2016002	20160	-0.37	0.710
14	2671002	26710	0.02	0.987
15	6724002	67240	-0.02	0.987
16	2197002	21970	0.00	0.996
17	2702002	27020	0.04	0.968
18	2857341	28420	-0.02	0.984
19	2527002	25270	0.00	0.996
20	2134381	21220	-0.05	0.962
21	1840041	18340	-0.29	0.775
22	4164002	41640	0.01	0.994
23	4213002	42130	-0.10	0.923
24	4985002	49850	-0.02	0.983
25	3654002	36540	-0.13	0.900

[a] N: tamaño del conjunto de datos original.

[b] n: tamaño del conjunto de datos muestreado.

* Se eligió la hipótesis nula como $\mu_1 - \mu_2 = 0$, en donde μ_1 es el promedio de N y μ_2 es el promedio de n.

Tabla 3.2: Parámetros de desempeño por modelo de las redes neuronales artificiales.

Identificador/ Arquitectura	Algoritmo de entrenamiento[a]	Validación *hold-out*	N[b]	MSE_{test}	$RMSE_{test}$	$aRRMSE_{test}$
BR28T	BR	Sí	28	0.0051	0.0716	0.1550
BR10F		No	10	0.0146	0.1208	0.2893
LM30T	LM	Sí	30	0.0053	0.0733	0.1650
LM14F		No	14	0.0134	0.1159	0.2597
SCG18T	SCG	Sí	18	0.0242	0.1557	0.3788
SCG30F		No	30	0.0584	0.2417	0.5070

[a] BR: *Bayesian Regulation*, LM: Levenberg-Marquardt, SCG: *Scaled Conjugated Gradient*.
[b] N: Número de neuronas en la capa oculta.

salida del modelo.

De igual manera, se muestra una comparación del MSE_{test} de los tres algoritmos sin validación (Figura 3.2) y con validación (Figura 3.3).

En la Tabla 3.4 se muestran los porcentajes de importancia relativa de las variables de entrada sobre las salidas, obtenidos mediante el algoritmo de Goh (1995). Para obtener las influencias relativas, se seleccionaron las seis mejores configuraciones de redes y se corrieron 20 repeticiones de cada una, diferentes a las utilizadas para calcular los parámetros de desempeño. La Tabla 3.4 muestra el promedio de las importancias de las 20 corridas independientes. Como se observa, ninguna de las variables tiene una importancia relativa mucho más alta que las demás, por lo que todas las entradas afectan a las salidas con una magnitud similar, y por lo tanto ninguna puede ser omitida del modelo.

Para llevar a cabo la comparación entre los algoritmos seleccionados de *machine learning*, se utilizó el parámetro de RRMSE. En la Figura 3.4 se observa el RRMSE de las diferentes arquitecturas de RNA, en donde es posible notar que los algoritmos BR y LM tienen un desempeño superior al de SCG, y a su vez las configuraciones en las que se utilizó la validación *hold-out* sobrepasan a aquellas en las que no se utilizó. Del mismo modo, no hay una diferencia significativa entre la mejor configuración entrenada con BR (28 neuronas con validación) y la

Tabla 3.3: Parámetros de desempeño por variable de salida de las redes neuronales artificiales.

Arquitectura	Parámetro	Salida			
		maxOUT	minOUT	timeOUT	bpmOUT
BR28T	R^2_{test}	0.9787	0.9954	0.806	0.988
	Exactitud (%)	92.30	93.79	91.40	92.78
	RRMSE	0.179	0.1199	0.1706	0.1505
BR10F	R^2_{test}	0.9332	0.9591	0.5947	0.9692
	Exactitud (%)	88.53	90.57	89.61	91.17
	RRMSE	0.3483	0.251	0.2953	0.2625
LM30T	R^2_{test}	0.9783	0.9953	0.7926	0.9891
	Exactitud (%)	91.01	91.65	89.79	91.88
	RRMSE	0.1837	0.1221	0.187	0.1673
LM14F	R^2_{test}	0.939	0.9649	0.6302	0.9728
	Exactitud (%)	85.74	91.55	88.35	88.15
	RRMSE	0.3022	0.2181	0.2729	0.2455
SCG18T	R^2_{test}	0.8461	0.945	0.5272	0.9456
	Exactitud (%)	83.85	87.40	85.16	84.20
	RRMSE	0.4721	0.3334	0.3759	0.3338
SCG30F	R^2_{test}	0.764	0.7599	0.4229	0.7062
	Exactitud (%)	83.75	85.99	84.98	87.88
	RRMSE	0.5763	0.4628	0.4908	0.498

Tabla 3.4: Importancia relativa (%) de las variables de entrada sobre las salidas en las mejores configuraciones de redes neuronales artificiales.*

Arquitectura/	Validación	Variables de entrada[b]									
Algoritmo[a]	*hold-out*	MaFC	MiFC	TL	FC	Peso	VolC	Lar	Anc	VolD	ConcD
10-10-4/BR	No	9.52	10.63	6.31	7.35	14.59	10.86	10.08	9.91	8.42	12.29
10-28-4/BR	Sí	6.22	4.91	6.13	5.33	14.50	10.86	10.68	13.89	7.15	14.92
10-14-4/LM	No	9.84	11.45	6.58	10.43	13.39	9.61	10.22	10.01	7.57	10.85
10-30-4/LM	Sí	9.21	10.18	8.17	9.67	11.94	10.66	10.48	11.37	7.66	10.61
10-30-4/SCG	No	10.40	9.96	9.69	10.13	9.60	10.28	9.59	9.95	10.23	10.12
10-18-4/SCG	Sí	11.04	11.30	8.74	9.85	10.71	9.63	9.51	9.18	9.98	10.01

* Los valores mostrados son los promedios de 20 corridas independientes a las usadas para determinar el desempeño de las redes.

[a] BR: *Bayesian Regulation*, LM: Levenberg-Marquardt y SCG: *Scaled Conjugated Gradient*.

[b] MaFC: máximo de fuerza de contracción, MiFC: mínimo de fuerza de contracción, TL: tiempo de llenado, FC: frecuencia cardíaca, VolC: volumen del corazón, Lar: largo del corazón, Anc: ancho del corazón, VolD: volumen de la dilución de digoxina, y ConcD: concentración de digoxina.

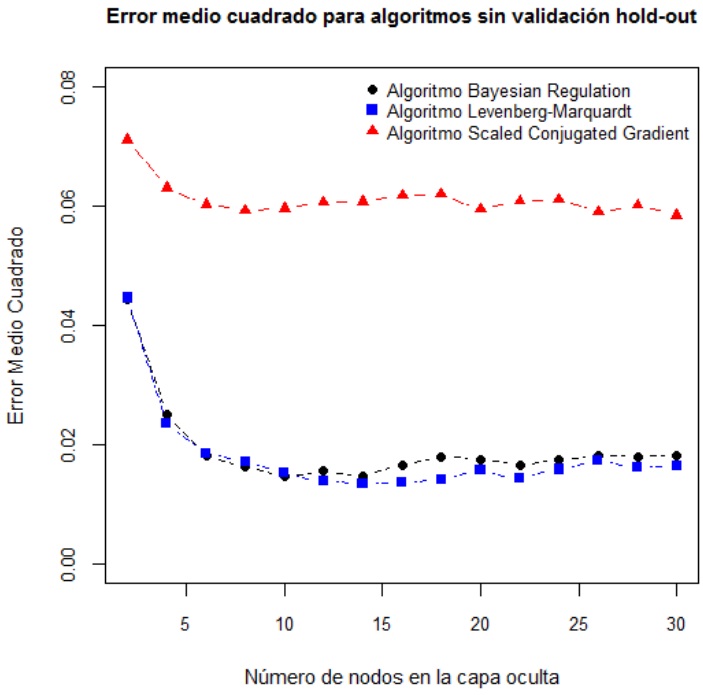

Figura 3.2: Error medio cuadrado de las redes neuronales artificiales entrenadas sin validación *hold-out*.

análoga entrenada con LM (30 neuronas sin validación), al igual que ocurre lo mismo para estos algoritmos sin validación; en este caso, la configuración LM14F superó ligeramente a la configuración BR10F.

En las Figuras 3.5 y 3.6 se muestran los valores de RRMSE obtenidos para los algoritmos multisalida utilizando como base los algoritmos BAG y SVR, respectivamente. En el caso de los métodos basados en el algoritmo BAG, no existe una diferencia significativa entre los seis algoritmos utilizados, posiblemente debido a que BAG es uno de los algoritmos de salida única más fuertes y con uno de los mejores desempeños (Spyromitros-Xioufis et al., 2016). En estos métodos es posible notar que el RRMSE de la salida timeOUT, que corresponde al tiempo de llenado, es mucho mayor que aquel del resto de las salidas. Este fenómeno puede deberse a que la digoxina no causó una modificación significativa en este parámetro durante los experimentos *ex vivo*, y por lo tanto no existe una diferencia notable entre este parámetro

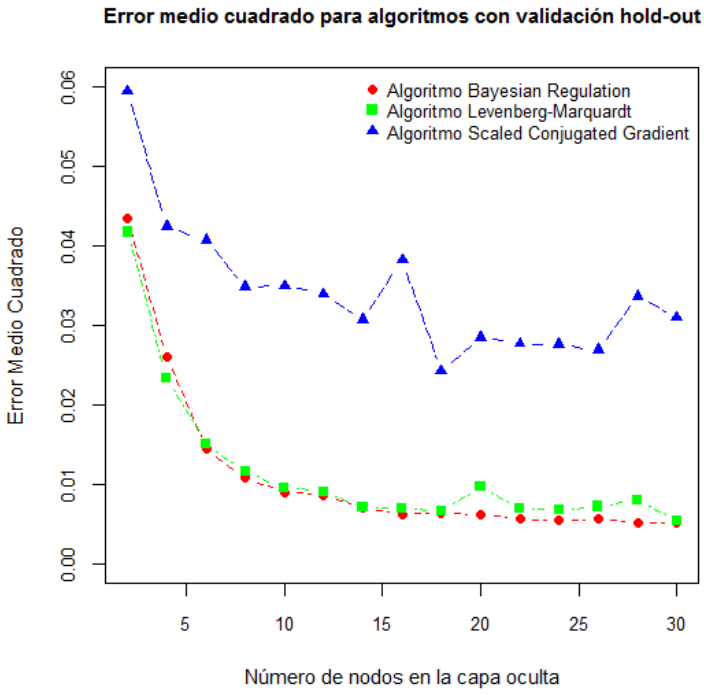

Figura 3.3: Error medio cuadrado de las redes neuronales artificiales entrenadas con validación *hold-out*.

antes y después de la adición de digoxina al medio.

Para el caso de los algoritmos con base SVR se observa de nuevo un RRMSE considerablemente más alto en el tiempo de llenado en comparación con el resto de las salidas. Estos algoritmos tuvieron un desempeño considerablemente más bajo en comparación con el resto de las técnicas de *machine learning*.

Cabe destacar que no se encontró una diferencia significativa en el RRMSE de la mejor configuración de RNA contra el mismo parámetro del mejor método de *machine learning*.

Con respecto de los tiempos requeridos para la construcción de los modelos, cabe mencionar que las RNA necesitan un tiempo considerable para ser entrenadas, principalmente cuando se utilizan los algoritmos LM y BR, siendo el último el más minucioso para la etapa de entrenamiento. Para el resto de algoritmos, los más tardados fueron ERC seguido de RLC,

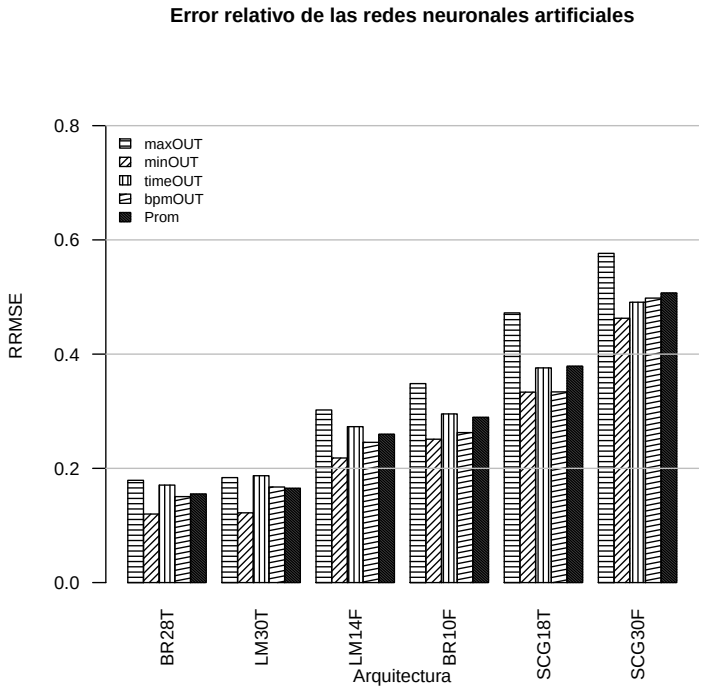

Figura 3.4: Raíz del error medio cuadrado relativo para las salidas de las redes neuronales artificiales.

mientras que los más rápidos fueron SST y ST.

3.4. Predicción de respuesta fisiológica

El modelo final se tomó como la red neuronal con 28 neuronas entrenada con el algoritmo BR y con validación *hold-out*, el cual obtuvo el MSE_{test} más bajo de entre todos los algoritmos utilizados. En la Figura 3.7 se muestra una predicción de la fuerza máxima de contracción conforme varía la cantidad de digoxina administrada. Para hacer esta predicción se fijó en un valor constante ocho de las 10 variables de entrada y se predijo el volumen y la concentración de digoxina con el modelo final. Se aprecia una correlación lineal entre ambas variables, tal como demuestran los experimentos llevados a cabo por Cuéllar-Roehri (1991), Guerra Rivas (1994) y Lumbreras Martínez (2012).

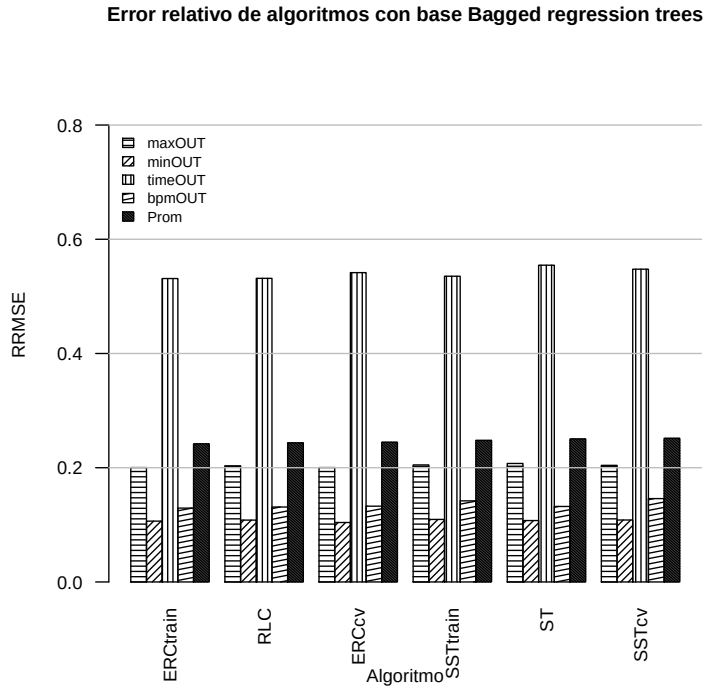

Figura 3.5: Raíz del error medio cuadrado relativo para las salidas de los algoritmos multisalida utilizando como algoritmo base *Bagged regression trees*.

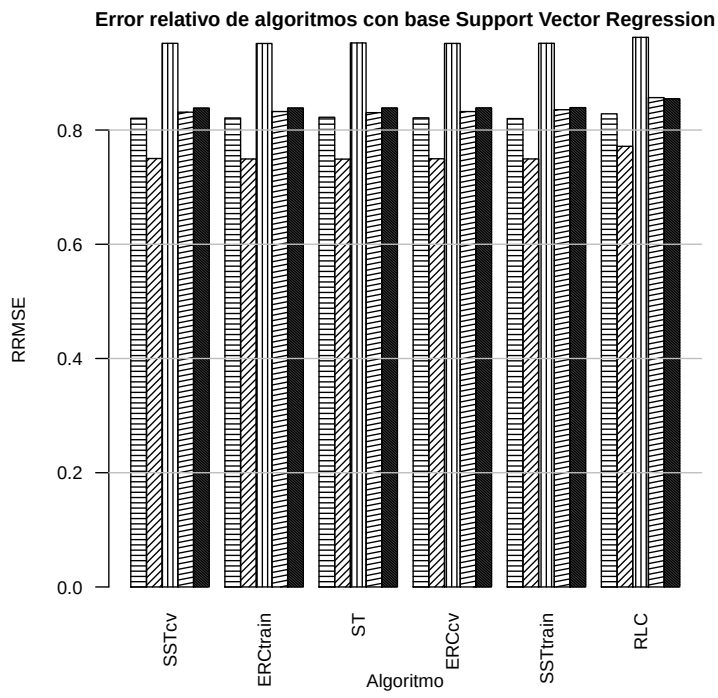

Figura 3.6: Raíz del error medio cuadrado relativo para las salidas de los algoritmos multisalida utilizando como algoritmo base *Support Vector Regression*.

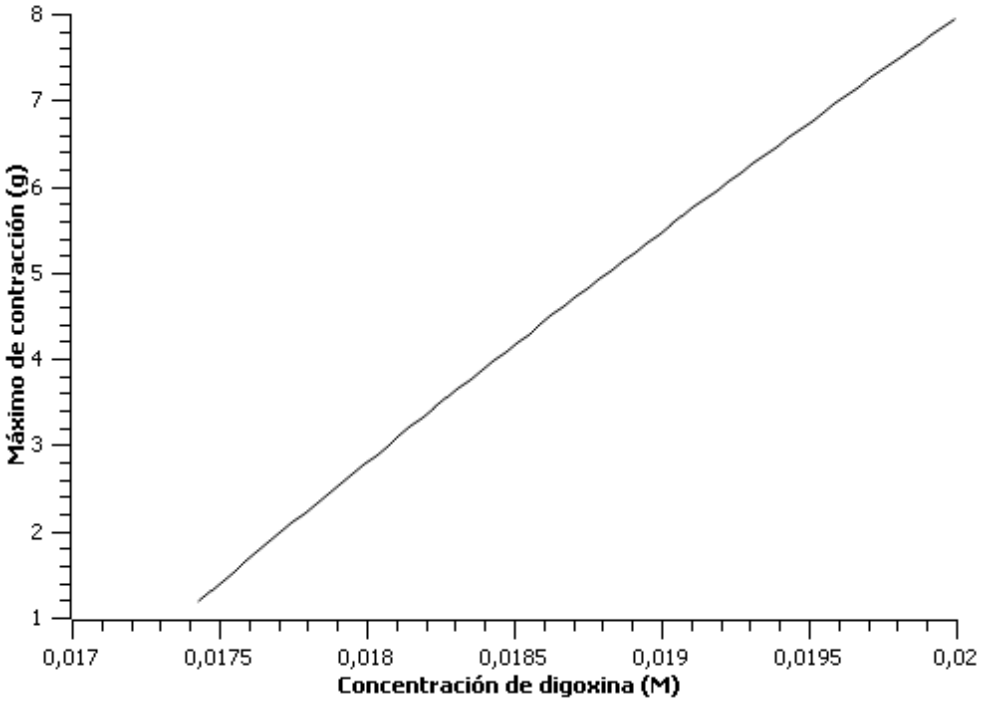

Figura 3.7: Predicción de máximo de fuerza de contracción de uno de los corazones utilizados en los experimentos.

Capítulo 4

Discusiones

Los parámetros MSE_{test} y $R^2{}_{test}$ de las mejores redes resultaron muy cercanos a cero y uno, respectivamente. Esto es muy improbable que suceda, ya que en realidad son números que demuestran un modelo ideal; y puede deberse a que las RNA se hayan sobreentrenado y convergido a un mínimo local (*overfitting*), un suceso no deseado en el que la red se adapta demasiado a los datos de entrada presentados en el entrenamiento, y como resultado tendrá una mala generalización ante nuevos datos de entrada (Fogel et al., 2015; Garson, 1991; Mateo Jiménez, 2012). Sin embargo, en este trabajo se llevaron a cabo varios métodos para prevenir este fenómeno: el haber utilizado el método de *random sub-sampling* para elegir los datos del subconjunto de entrenamiento (diferente para cada construcción de red) de todo el *dataset*, significa que los valores obtenidos de MSE_{test} y $R^2{}_{test}$ pueden variar en otras repeticiones aunque se esté utilizando la misma configuración de red. Al utilizar un doble método de validación (*random sub-sampling* y validación *hold-out*) se asegura la robustez del modelo frente a una selección sesgada de los datos de los subconjuntos de prueba y validación, y esto indica que las redes no están sobreentradas. Aunado a esto, las 20 iteraciones de cada configuración se llevan a cabo con el fin de elegir la que tenga el error más bajo utilizando la misma configuración. Del mismo modo, el tamaño del *dataset* (3,975 x 4 datos) previene

dicha selección sesgada, en dado caso de que se considerara esa hipótesis, ya que es un tamaño considerable.

Los algoritmos LM y BR con validación *hold-out* mostraron ser los mejores para este experimento (Tabla 3.2) de acuerdo al menor MSE_{test}, superando al algoritmo SCG, como ha sucedido en trabajos previos. El desempeño de las RNA-PMC siempre mejoró cuando se utilizó la validación *hold-out*, para todos los casos, al igual que ha sucedido en trabajos previos (Flores et al., 2017a,b; Mateo Jiménez, 2012; Mateo Jiménez et al., 2011).

Para el resto de las técnicas de *machine learning*, el mejor método resultó ser $\text{ERC}_{\text{train}}$-BAG seguido de RLC-BAG, lo que concuerda con trabajos previos utilizando estos algoritmos (Spyromitros-Xioufis et al., 2016). En realidad, no hubo una diferencia significativa en el aRRMSE entre los seis algoritmos de multisalida utilizados. Con respecto de los algoritmos base, el algoritmo BAG resultó ser considerablemente superior al algoritmo SVR, lo que también concuerda con el trabajo de Spyromitros-Xioufis et al. (2016), en donde BAG fue considerado como el mejor algoritmo del estado del arte como base para regresión múltiple, y SVR resultó ser uno de los que tenía el desempeño más bajo, en una comparación con otros algoritmos de salida única utilizados frecuentemente en *machine learning*. No obstante, se debe tomar en cuenta que las técnicas de *machine learning* utilizadas para regresión múltiple tienen un desempeño diferente dependiendo de la naturaleza del conjunto de datos con el que se esté trabajando, por ejemplo, un algoritmo puede tener un buen desempeño para ciertos datos de naturaleza ecológica, pero reducir su capacidad de predicción para un conjunto de datos correspondientes a la actividad solar en un determinado tiempo (Spyromitros-Xioufis et al., 2016).

Uno de los hallazgos más importantes de este trabajo, es que el aRRMSE de las mejores configuraciones de RNA fueron menores a aquellos de los mejores algoritmos de *machine learning*, lo cual pone de manifiesto que aunque la topología PMC ha quedado un poco obsoleta debido al surgimiento de topologías más actualizadas, sigue siendo una herramienta

computacional competitiva y muy eficaz, con una alta capacidad de aprendizaje, predicción y adaptación (*fitting*), y por lo tanto útil para aplicaciones de regresión en el modelado farmacodinámico.

Capítulo 5

Conclusiones y trabajo futuro

Los algoritmos BR y LM sobrepasan al algoritmo SCG en cuanto a las RNA, con la desventaja de requerir un tiempo mucho más largo para el entrenamiento. El algoritmo LM representa una excelente opción con un buen balance entre desempeño y tiempo requerido para el entrenamiento, ya que la diferencia en la capacidad de predicción entre este y BR puede despreciarse. Para el resto de algoritmos utilizados en este trabajo, el algoritmo SST utilizando como base el regresor BAG posee un buen equilibrio entre desempeño y tiempo necesario para el entrenamiento.

Para el caso específico de este trabajo, no existe una diferencia significativa en el desempeño de las RNA en comparación con el desempeño de algoritmos del estado del arte de *machine learning*, como SST o ERC, y por lo tanto, las RNA-PMC son un método computacional de predicción capaz de competir contra algoritmos más actualizados para regresión múltiple. Sin embargo, ambos grupos de métodos demostraron una buena capacidad de predicción aplicados en datos de origen farmacodinámico. Se debe de tomar en cuenta que diferentes algoritmos y métodos tienen un diferente desempeño en cuanto a capacidad de predicción y exactitud dependiendo de la naturaleza de los datos, es decir, del fenómeno al que corresponden.

Como trabajo futuro, se necesitan llevar a cabo nuevos experimentos que comparen las técnicas utilizadas en este proyecto contra otras más actualizadas, como las redes neuronales evolutivas, adaptativas de Newmann-Watts, recursivas, diferenciales o de base radial (Fogel et al., 2015). Aunado a esto, es necesario probar distintos métodos de validación para los algoritmos de *machine learning* con el fin de encontrar el método que mejor se ajuste para este trabajo. Por otra parte, técnicas de optimización como algoritmos genéticos o *Particle Swarm Optimization* (PSO), se pueden utilizar acoplados a las redes neuronales artificiales para aumentar aún más su capacidad de predicción, lo cual sugiere una posible línea de investigación a futuro.

Otro aspecto importante que se puede trabajar es la reconstrucción de la señal de la actividad cardíaca a partir de los cuatro parámetros predichos con los modelos, para así validar la predicción de la respuesta fisiológica. Es posible realizar esta reconstrucción utilizando herramientas de procesamiento digital de señales y métodos numéricos, entre otras técnicas de ingeniería y computación.

Bibliografía

S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M.P. O'Hare. Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review*, pages 1–21, 2017. doi: 10.1016/j.cosrev.2017.03.001.

S. Agatonovic-Kustrin and R. Beresford. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5):717–727, 2000. doi: 10.1016/S0731-7085(99)00272-1.

A. Albert, A. J. Serrano-López, E. Soria-Olivas, and N. V. Jimenez. Clinical Decision Support System to Prevent Toxicity in Patients Treated with Digoxin. In A. Shukla and R. Tiwari, editors, *Intelligent medical technologies and biomedical engineering: Tools and applications*, pages 1–21. Hershey, 2010. doi: 10.4018/978-1-61520-977-4.ch001.

O. Y. Atkov, S. G. Gorokhova, A. G. Sboev, E. V. Generozov, E. V. Muraseyeva, S. Y. Moroshkina, and N. N. Cherniy. Coronary heart disease diagnosis by artificial neural networks including genetic polymorphisms and clinical parameters. *Journal of Cardiology*, 59(2):190–194, 2012. doi: 10.1016/j.jjcc.2011.11.005.

I. Babaoglu, O. K. Baykan, N. Aygul, K. Ozdemir, and M. Bayrak. Assessment of exercise stress testing with artificial neural network in determining coronary artery disease and predicting lesion localization. *Expert Systems with Applications*, 36(2 parte 1):2562–2566, 2009. doi: 10.1016/j.eswa.2007.11.013.

A. Belič, I. Grabnar, I. Belič, R. Karba, and A. Mrhar. Predicting the anti-hypertensive effect of nitrendipine from plasma concentration profiles using artificial neural networks. *Computers in biology and medicine*, 35(10):892–904, 2005. doi: 10.1016/j.compbiomed.2004.07.006.

W. Berman, J. Musselman, and R. Shortencarrier. The physiologic effects of digoxin under steady-state drug conditions in newborn and adult sheep. *Circulation*, 62(6):1165–1171, 1980.

G. Bini, A. M. Pugliese, G. Pepeu, and G. Chelazzi. Tetrodotoxin prevents copper-induced bradycardia in gastropod limpets. *Environmental Pollution*, 139(1):79–85, 2006. doi: 10.1016/j.envpol.2005.04.029.

H. Borchani, G. Varando, C. Bielza, and P. Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015. doi: 10.1002/widm.1157.

L. Breiman. Bagging Predictors. *Machine Learning*, 24(421):123–140, 1996. doi: 10.1007/BF00058655.

J. E. Campbell and D. Cohall. *Pharmacodynamics—A Pharmacognosy Perspective*. Elsevier Inc., 2017. doi: 10.1016/B978-0-12-802104-0.00026-3.

G. Camps-Valls, E. Soria-Olivas, and N. V. Jimenez-Torres. Artificial neural networks for the classification of potentially intoxicated patients treated with digoxin. *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1: 13–16, 2000. doi: 10.1109/IEMBS.2000.900655.

G. Camps-Valls, B. Porta-Oltra, E. Soria-Olivas, J. D. Martin-Guerrero, A. J. Serrano-Lopez, J. J. Perez-Ruixo, and N. V. Jimenez-Torres. Prediction of cyclosporine dosage in patients after kidney transplantation using neural networks. *IEEE Transactions on Biomedical Engineering*, 50(4): 442–448, 2003. doi: 10.1109/TBME.2003.809498.

Y. Chen, T. W. McCall, A. R. Baichwal, and M. C. Meyer. The application of an artificial neural network and pharmacokinetic simulations in the design of controlled-release dosage forms. *Journal of Controlled Release*, 59(1):33–41, 1999. doi: 10.1016/S0168-3659(98)00171-0.

Y. C. Chen, W. W. Cao, Y. Cao, L. Zhang, B. B. Chang, W. L. Yang, and X. Q. Liu. Using neural networks to determine the contribution of danshensu to its multiple cardiovascular activities in acute myocardial infarction rats. *Journal of Ethnopharmacology*, 138(1):126–134, 2011. doi: 10.1016/j.jep.2011.08.069.

Y. V. Chesnokov. Complexity and spectral analysis of the heart rate variability dynamics for distant prediction of paroxysmal atrial fibrillation with artificial intelligence methods. *Artificial Intelligence in Medicine*, 43(2):151–165, 2008. doi: 10.1016/j.artmed.2008.03.009.

L. P. Collis, Y. Sun, and R. B. Hill. Length-dependent deactivation of ventricular trabeculae in the bivalve, Spisula solidissima. *Journal of Comparative Physiology B: Biochemical, Systemic, and Environmental Physiology*, 176(4):371–385, 2006. doi: 10.1007/s00360-005-0060-9.

M. Cuéllar-Roehri. *Utilización de la almeja pismo Tivela stultorum (Mawe, 1823) como organismo de prueba para glucósidos cardiotónicos*. Tesis de licenciatura, Universidad Autónoma de Baja California, 1991.

A. H. Dawson and N. A. Buckley. Digoxin. *Medicine*, 44(3):158–159, 2016. doi: 10.1016/j.mpmed.2015.12.006.

M. De Matas, Q. Shao, M. F. Biddiscombe, S. Meah, H. Chrystyn, and O. S. Usmani. Predicting the clinical effect of a short acting bronchodilator in individual patients using artificial neural networks. *European Journal of Pharmaceutical Sciences*, 41(5):707–715, 2010. doi: 10.1016/j.ejps.2010.09.018.

L. Del Valle-Mondragón, M. Ramírez-Ortega, G. Zarco-Olvera, A. Sánchez-Mendoza, G. Pastelín-Hernández, and F. A. Tenorio-López. Capillary zone electrophoretic determination of cytochrome c in mitochondrial extracts and cytosolic fractions: Application to a digitalis intoxication study. *Talanta*, 74(4):478–488, 2008. doi: 10.1016/j.talanta.2007.06.010.

J. Dingemanse and A. Krause. Impact of pharmacokinetic-pharmacodynamic modelling in early clinical drug development. *European Journal of Pharmaceutical Sciences*, (May):0–1, 2017. doi: 10.1016/j.ejps.2017.05.042.

R. Dolphen and M. Lesne. Uptake and pharmacological effect of gitoxin and gitaloxin in rat and guinea-pig perfused hearts. Comparison with digitoxin and digoxin. *Arzneimittel-Forschung*, 30 (4):614–618, 1980.

H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Dystems*, 1:155–161, 1997. doi: 10.1.1.10.4845.

R. J. Erb. Introduction to Backpropagation Neural Network Computation, 1993.

F. Eyer, W. Steimer, T. Nitzsche, N. Jung, H. Neuberger, C. Müller, M. Schlapschy, T. Zilker, and A. Skerra. Intravenous application of an anticalin dramatically lowers plasma digoxin levels and reduces its toxic effects in rats. *Toxicology and Applied Pharmacology*, 263(3):352–359, 2012. doi: 10.1016/j.taap.2012.07.009.

J. E. Fitch. The pismo clam. *California Fish Game*, 36:285–312, 1950.

A. D. Flapan, N. E. Goodfield, R. A. Wright, C. M. Francis, and J. M. Neilson. Effects of digoxin on time domain measures of heart rate variability in patients with stable chronic cardiac failure: withdrawal and comparison group studies. *International Journal of Cardiology*, 59(1):29–36, 1997. doi: 10.1016/S0167-5273(96)02893-8.

D.-L. Flores, C. Gómez, D. Cervantes, A. Abaroa, C. Castro, and R. A. Castañeda-Martínez. Predicting the physiological response of Tivela stultorum hearts with digoxin from cardiac parameters using artificial neural networks. *BioSystems*, 151:1–7, 2017a. doi: 10.1016/j.biosystems.2016.11.002.

D.-L. Flores, C. Gómez, D. Cervantes, A. Abaroa, C. Castro, and R. A. Castañeda-Martínez. Prediciendo la actividad cardíaca de la almeja Tivela stultorum con digoxina utilizando redes neuronales artificiales. *Revista Mexicana de Ingeniería Biomédica*, 38(1):208–216, 2017b. doi: 10.17488/RMIB.38.1.15.

G. B. Fogel, S. L. Lamers, E. S. Liu, M. Salemi, and M. S. McGrath. Identification of dual-tropic HIV-1 using evolved neural networks. *BioSystems*, 137:12–19, 2015. doi: 10.1016/j.biosystems.2015.09.007.

R. M. García-Gimeno, C. Hervás-Martínez, R. Rodríguez-Pérez, and G. Zurera-Cosano. Modelling the growth of Leuconostoc mesenteroides by Artificial Neural Networks. *International Journal of Food Microbiology*, 105(3):317–332, 2005. doi: 10.1016/j.ijfoodmicro.2005.04.013.

G. D. Garson. Interpreting neural-network connection weights. *AI Expert*, 6(4):46–51, 1991.

J. V. S. Gobburu and E. P. Chen. Artificial neural networks as a novel approach to integrated pharmacokinetic-pharmacodynamic analysis. *Journal of Pharmaceutical Sciences*, 85(5):505–510, 1996. doi: 10.1021/js950433d.

S. Godbole and S. Sarawagi. Discriminative Methods for Multi-labeled Classification. *Lecture Notes in Computer Science*, 3056:22–30, 2004. doi: 10.1007/978-3-540-24775-3_5.

A. T. C. Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence Expert*, 9:143–151, 1995.

I. Goldstein, T. Levy, D. Galili, H. Ovadia, R. Yirmiya, H. Rosen, and D. Lichtstein. Involvement of Na+, K+-ATPase and Endogenous Digitalis-Like Compounds in Depressive Disorders. *Biological Psychiatry*, 60(5):491–499, 2005. doi: 10.1016/j.biopsych.2005.12.021.

G. Gramelsberger. The simulation approach in synthetic biology. *Studies in History and Philosophy of Biological and Biomedical Sciences*, 44(2):150–157, 2013. doi: 10.1016/j.shpsc.2013.03.010.

F. Gross and M. MacLeod. Prospects and problems for standardizing model validation in systems biology. *Progress in Biophysics and Molecular Biology*, pages 1–10, 2016. doi: 10.1016/j.pbiomolbio.2017.01.003.

L. J. Gross. Computer Systems And Models, Use Of. *Encyclopedia of Biodiversity*, 1:845–853, 2013. doi: 10.1016/B0-12-226865-2/00058-4.

G. Guerra Rivas. *Evaluación experimental del uso del corazón de almeja pismo (Tivela stultorum) como modelo biológico para Farmacología Marina*. Tesis de maestría, Universidad Autónoma de Baja California, 1994.

M. T. Hagan, H. B. Demuth, and M. Beale. *Neural Network Design*. 2014.

S. H. Haidar, S. B. Johnson, M. J. Fossler, and A. S. Hussain. Modeling the pharmacokinetics and pharmacodynamics of a unique oral hypoglycemic agent using neural networks. *Pharmaceutical Research*, 19(1):87–91, 2002. doi: 10.1023/A:1013611617787.

S. Haykin. *Neural networks: a comprehensive foundation*. Pearson Education, 2da edition, 1999. doi: 10.1017/S0269888998214044.

H. G. Hosseini, D. Luo, and K. J. Reynolds. The comparison of different feed forward neural network architectures for ECG signal diagnosis. *Medical Engineering and Physics*, 28(4):372–378, 2006. doi: 10.1016/j.medengphy.2005.06.006.

S. Joo, K. J. Choi, and S. J. Huh. Prediction of spontaneous ventricular tachyarrhythmia by an artificial neural network using parameters gleaned from short-term heart rate variability. *Expert Systems with Applications*, 39(3):3862–3866, 2012. doi: 10.1016/j.eswa.2011.09.097.

A. Jovic and N. Bogunovic. Electrocardiogram analysis using a combination of statistical, geometric, and nonlinear heart rate variability features. *Artificial Intelligence in Medicine*, 51(3):175–186, 2011. doi: 10.1016/j.artmed.2010.09.005.

S. S. Kareem and Y. Pathak. *Clinical Applications of Artificial Neural Networks in Pharmacokinetic Modeling*. Elsevier Inc., 2016. doi: 10.1016/B978-0-12-801559-9.00020-X.

S. A. Kodirov. The neuronal control of cardiac functions in Molluscs. *Comparative Biochemistry and Physiology - A Molecular and Integrative Physiology*, 160(2):102–116, 2011. doi: 10.1016/j.cbpa.2011.06.014.

P. Lecca, A. Re, A. E. Ihekwaba, I. Mura, and T.-P. Nguyen. *Computational Systems Biology: Inference and Modeling*. Elsevier Ltd, 2016. doi: 10.1016/B978-0-08-100095-3.09983-2.

H. Li, C. W. Yap, C. Y. Ung, Y. Xue, Z.R. Li, L. Y. Han, H. H. Lin, and Y. Z. Chen. Machine learning approaches for predicting compounds that interact with therapeutic and ADMET related proteins. *Journal of Pharmaceutical Sciences*, 96(11):2838–2860, 2007. doi: 10.1002/jps.

W. Lou and S. Nakai. Artificial neural network-based predictive model for bacterial growth in a simulated medium of modified-atmosphere-packed cooked meat products. *Journal of Agricultural and Food Chemistry*, 49(4):1799–1804, 2001. doi: 10.1021/jf000650m.

H. Lumbreras Martínez. *Caracterización de compuestos cardiotónicos del Ofiuroideo Ophioderma panamense Lütken, 1859 y validación del corazón de Mytilus californianus como método de cribado*. Tesis de maestría, Universidad Autónoma de Baja California, 2012.

D. E. Mager, J. D. Shirey, D. Cox, D. J. Fitzgerald, and D. R. Abernethy. Mapping the dose-effect relationship of orbofiban from sparse data with an artificial neural network. *Journal of Pharmaceutical Sciences*, 94(11):2475–2486, 2005. doi: 10.1002/jps.20384.

M. Marian and W. Seghezzi. *Novel Biopharmaceuticals: Pharmacokinetics, Pharmacodynamics, and Bioanalytics*. Elsevier, 2013. doi: 10.1016/B978-0-12-394810-6.00004-6.

Z. Masetic and A. Subasi. Congestive heart failure detection using random forest classifier. *Computer Methods and Programs in Biomedicine*, 130:54–64, 2016. doi: 10.1016/j.cmpb.2016.03.020.

F. Mateo Jiménez. *Redes neuronales y preprocesado de variables para modelos y sensores en bioingeniería*. Tesis doctoral, Universidad Politécnica de Valencia, 2012.

F. Mateo Jiménez, R. Gadea, Á Medina, R. Mateo, and M. Jiménez. Predictive assessment of ochratoxin A accumulation in grape juice based-medium by Aspergillus carbonarius using neural networks. *Journal of Applied Microbiology*, 107(3):915–927, 2009. doi: 10.1111/j.1365-2672.2009.04264.x.

F. Mateo Jiménez, R. Gadea, E. M. Mateo, and M. Jiménez. Multilayer perceptron neural networks and radial-basis function networks as tools to forecast accumulation of deoxynivalenol in barley seeds contaminated with Fusarium culmorum. *Food Control*, 22(1):88–95, 2011. doi: 10.1016/j.foodcont.2010.05.013.

Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors. *Machine learning, an artificial intelligence approach*. Morgan Kaufmann Publishers, Palo Alto, 1983.

R. C. Muniyandi and A. Mohd Zin. Modeling framework for membrane computing in biological systems: Evaluation with a case study. *Journal of Computational Science*, 5(2):137–143, 2014. doi: 10.1016/j.jocs.2013.12.004.

J. D. Olden and D. A. Jackson. Illuminating the "black box": a ramdomization approach for understanding variable contributions in artificial neuronal networks. *Ecological Modelling*, 154: 135–150, 2002. doi: 10.1016/S0304-3800(02)00064-9.

OMS. Las 10 principales causas de defunción, 2017. URL http://www.who.int/mediacentre/factsheets/fs310/es/.

P. Ponce Cruz. *Inteligencia artificial con aplicaciones a la ingeniería*. Alfaomega, D.F., 1ra edition, 2010. doi: 10:0-8400-5444-0.

C. L. Prosser. Acetylcholine and nervous inhibition in the heart of Venus mercenaria. *The Biological Bulletin*, 78(1):92–102, 1940. doi: 10.2307/1537803.

H. M. Rai, A. Trivedi, and S. Shukla. ECG signal processing for abnormalities detection using multi-resolution wavelet transform and Artificial Neural Network classifier. *Measurement*, 46(9): 3238–3246, 2013. doi: 10.1016/j.measurement.2013.05.021.

U. Rajendra Acharya, P. Subbanna Bhat, S. S. Iyengar, A. Rao, and S. Dua. Classification of heart rate data using artificial neural network and fuzzy equivalence relation. *Pattern Recognition*, 36 (1):61–68, 2003. doi: 10.1016/S0031-3203(02)00063-8.

G. Rozenberg, T. Bäck, and J. N. Kok, editors. *Handbook of Natural Computing*. Springer-Verlag, Heidelberg, 2012. doi: 10.1007/978-3-540-92910-9.

E. J. Rykiel. Testing ecological models: The meaning of validation. *Ecological Modelling*, 90(3): 229–244, 1996. doi: 10.1016/0304-3800(95)00152-2.

S. G. Schäfer, G. Schuhmann, W. Doering, and B. Fichtl. Influence of quinidine on the intestinal secretion of digoxin and digitoxin in Guinea pigs. *Chemico-Biological Interactions*, 55:203–213, 1985.

Yoav Shoham. *Artificial intelligence techniques in prolog*. Morgan Kaufmann Publishers, San Francisco, 1994. doi: 10.1049/ic:19971179.

J. Siepmann. In-silico simulations of advanced drug delivery systems: What will the future offer? *International Journal of Pharmaceutics*, 454(1):512–516, 2013. doi: 10.1016/j.ijpharm.2013.07.018.

R. Silipo, M. Gori, A. Taddei, M. Varanini, and C. Marchesi. Classification of arrhythmic events in ambultory electrocardiogram, using artificial neural networks.pdf. *Computers and Biomedical Research*, 28:305–318, 1995.

C. Skoglund. Additions to the Panamic Province Bivalve (Mollusca) literature 1971 to 1990. *Festivus*, 22:1–63, 1991.

T. W. Smith. Digitalis, mechanisms of action and clinical use. *The New England Journal of Medicine*, 318(6):358–365, 1988.

E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas. Multi-Label Classification Methods for Multi-Target Regression. 2012.

E. Spyromitros-Xioufis, G. Tsuomakas, W. Groves, and I. Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016. doi: 10.1007/s10994-016-5546-z.

Marika Svrckova, Martina Zatloukalova, Petra Dvorakova, Dominika Coufalova, David Novak, Lenka Hernychova, and Jan Vacek. Na+/K+-ATPase Interaction with Methylglyoxal as Reactive Metabolic Side Product. *Free Radical Biology and Medicine*, 2017. doi: 10.1016/j.freeradbiomed.2017.03.024.

K. Takayama, M. Fujikawa, Y. Obata, and M. Morishita. Neural network based optimization of drug formulations. *Advanced Drug Delivery Reviews*, 55(9):1217–1231, 2003. doi: 10.1016/S0169-409X(03)00120-0.

G. Tsoumakas, E. Spyromitros-Xioufis, A. Vrekou, and I. Vlahavas. Multi-Target Regression via Random Linear Target Combinations. pages 1–16, 2014. doi: 10.1007/978-3-662-44845-8_15.

M. J. Wernke and W. Cacini. Concentrative uptake of digoxin by slices of chicken renal cortex. *Biochemical Pharmacology*, 39(4):655–659, 1990. doi: 10.1016/0006-2952(90)90142-8.

R. D. Wilkerson, P. B. Mockridge, and G. K. Massing. Effects of selected drugs on serum digoxin concentration in dogs. *The American Journal of Cardiology*, 45(6):1201–1210, 1980. doi: 10.1016/0002-9149(80)90479-8.

S. Yamamura. Clinical application of artificial neural network (ANN) modeling to predict pharmacokinetic parameters of severely ill patients. *Advanced Drug Delivery Reviews*, 55(9):1233–1251, 2003. doi: 10.1016/S0169-409X(03)00121-2.

E. H. Yang, S. Shah, and J. M. Criley. Digitalis toxicity: A fading but crucial complication to recognize. *American Journal of Medicine*, 125(4):337–343, 2012. doi: 10.1016/j.amjmed.2011.09.019.

X. S. Yang, S. Koziel, and L. Leifsson. Computational optimization, modelling and simulation: Past, present and future. *Procedia Computer Science*, 29(Coms):754–758, 2014. doi: 10.1016/j.procs.2014.05.067.

J. S. Yu and N. Bagheri. Multi-class and multi-scale models of complex biological phenomena. *Current Opinion in Biotechnology*, 39:167–173, 2016. doi: 10.1016/j.copbio.2016.04.002.

O. J. Ziff and D. Kotecha. Digoxin: The good and the bad. *Trends in Cardiovascular Medicine*, 2016. doi: 10.1016/j.tcm.2016.03.011.

Apéndice A

Código fuente

A.1. Función principal SW1C.m

```
1  %El contenido de este script es para realizar las corridas de las distintas
       configuraciones de redes neuronales
2  %ademas de realizar las 20 iteraciones, guarda un archivo .xlsx con los
       distintos parametros MSE's y R's.
3  %Este codigo construye todas las configuraciones de las redes mandando llamar
        a la funcion RedValid.m, la cual es parte del Neural Network Toolbox de
       MATLAB.
4  %Se inicializan las variables
5  [FCN,F,c,HOV,t1,t2] = deal(['trainscg';'trainlmh';'trainscg'],['SCG';'LMh';'
       BRh'],1,1,5,1);
6  NumNe = Neuronas1C();
7  %Se carga el conjunto de datos y se divide en entradas (p) y salidas (t).
8  p =(xlsread('NF.xlsx','Tesis','A2:J3976'))';
9  t =(xlsread('NF.xlsx','Tesis','K2:N3976'))';
10 norm_data = mapminmax(t,-1,1);
11 z=zscore(p);
12 for HOV = 1:2
13     if HOV == 1
14         val = 'T';
15     else
16         val = 'F';
17     end
18     for c = 1:3
19         switch c
20             case 1
21                 [v,testMSE,trainingMSE,R2,rMSE,IRGT,acc5,acc1,nena,pred] =
                       deal(1,zeros(1,300),zeros(1,300),zeros(300,4),zeros(1,300),
                       zeros(300,10),zeros(300,4),zeros(300,4),cell(1,300),zeros
                       (1,4));
22                 for l = 1:15
23                     for i = 1:20
24                         [tr,performance,testPerformance,net,r,testTargets] =
```

```
                    RedValid(NumNe(l),z,norm_data,HOV,FCN(c,1:7));
25                  save(['N1_' F(c,1:3) num2str(NumNe(l)) '_' val num2str
                        (i) '.mat']);
26                  %Calcula los parametros de validacion.
27                  [testMSE(v,1),trainingMSE(v,1),R2(v,:),rMSE(v,1)] =
                        Parametros(testPerformance,performance,r);
28                  R2(v,:) = r';
29
30                  %Calcula las contribuciones relativas de las
                        variables de entrada.
31                  [IRG] = Garson(NumNe(l),net);
32                  for q = 1:10
33                      IRGT(v,q) = IRG(q);
34                  end
35                  %Realiza las predicciones con el modelo.
36                  [acc5(v,:),acc1(v,:)] = Accuracy(net,z,norm_data,t1,t2
                        );
37
38                  if l <=4
39                      if i <= 9
40                          nena{v} = ['N1_' F(c,1:3) '0' num2str(NumNe(l)
                                ) '_' val '0' num2str(i)];
41                      else
42                          nena{v} = ['N1_' F(c,1:3) '0' num2str(NumNe(l)
                                ) '_' val num2str(i)];
43                      end
44                  else
45                      if i <= 9
46                          nena{v} = ['N1_' F(c,1:3) num2str(NumNe(l)) '_
                                ' val '0' num2str(i)];
47                      else
48                          nena{v} = ['N1_' F(c,1:3) num2str(NumNe(l)) '_
                                ' val num2str(i)];
49                      end
50                  end
51                  v = v+1;
52              end
53          end
54          EW1C(nena,testMSE,trainingMSE,rMSE,R2,IRGT,F(c,1:3),val,acc5,
                acc1);
55          clc;
56      case 2
57          [v,testMSE,trainingMSE,R2,rMSE,IRGT,acc5,acc1,nena,pred] =
                deal(1,zeros(1,300),zeros(1,300),zeros(300,4),zeros(1,300),
                zeros(300,10),zeros(300,4),zeros(300,4),cell(1,300),zeros
                (1,4));
58          for l = 1:15
59              for i = 1:20
60                  [tr,performance,testPerformance,net,r,testTargets] =
                        RedValid(NumNe(l),z,norm_data,HOV,FCN(c,1:7));
61                  save(['N1_' F(c,1:2) num2str(NumNe(l)) '_' val num2str
                        (i) '.mat']);
```

```
62                          % Calcula los parametros de validacion.
63                          [testMSE(v,1),trainingMSE(v,1),R2(v,:),rMSE(v,1)] =
                                Parametros(testPerformance,performance,r);
64                          R2(v,:) = r';
65
66                          % Calcula las contribuciones relativas de las
                                variables de entrada.
67                          [IRG] = Garson(NumNe(l),net);
68                          for q = 1:10
69                              IRGT(v,q) = IRG(q);
70                          end;
71                          % Realiza las predicciones con el modelo.
72                          [acc5(v,:),acc1(v,:)] = Accuracy(net,z,norm_data,t1,t2
                                );
73                          if l <=4
74                              if i <= 9
75                                  nena{v} = ['N1_' F(c,1:2) '0' num2str(NumNe(l)
                                        ) '_' val '0' num2str(i)];
76                              else
77                                  nena{v} = ['N1_' F(c,1:2) '0' num2str(NumNe(l)
                                        ) '_' val num2str(i)];
78                              end
79                          else
80                              if i <= 9
81                                  nena{v} = ['N1_' F(c,1:2) num2str(NumNe(l)) '_
                                        ' val '0' num2str(i)];
82                              else
83                                  nena{v} = ['N1_' F(c,1:2) num2str(NumNe(l)) '_
                                        ' val num2str(i)];
84                              end
85                          end
86                          v = v+1;
87                      end
88                  end
89                  EW1C(nena,testMSE,trainingMSE,rMSE,R2,IRGT,F(c,1:3),val,acc5,
                        acc1);
90                  clc;
91              case 3
92                  [v,testMSE,trainingMSE,R2,rMSE,IRGT,acc5,acc1,nena,pred] =
                        deal(1,zeros(1,300),zeros(1,300),zeros(300,4),zeros(1,300),
                        zeros(300,10),zeros(300,4),zeros(300,4),cell(1,300),zeros
                        (1,4));
93                  for l = 1:15
94                      for i = 1:20
95                          [tr,performance,testPerformance,net,r,testTargets] =
                                RedValid(NumNe(l),z,norm_data,HOV,FCN(c,1:8));
96                          save(['N1_' F(c,1:2) num2str(NumNe(l)) '_' val num2str
                                (i) '.mat']);
97                          % Calcula los parametros de validacion.
98                          [testMSE(v,1),trainingMSE(v,1),R2(v,:),rMSE(v,1)] =
                                Parametros(testPerformance,performance,r);
99
```

```
100                         % Calcula las contribuciones relativas de las
                                variables de entrada.
101                         [IRG] = Garson(NumNe(l),net);
102                         for q = 1:10
103                             IRGT(v,q) = IRG(q);
104                         end
105
106                         % Realiza las predicciones con el modelo.
107                         [acc5(v,:),acc1(v,:)] = Accuracy(net,z,norm_data,t1,t2
                                );
108                         if l <=4
109                             if i <= 9
110                                 nena{v} = ['N1_' F(c,1:2) '0' num2str(NumNe(l)
                                        ) '_' val '0' num2str(i)];
111                             else
112                                 nena{v} = ['N1_' F(c,1:2) '0' num2str(NumNe(l)
                                        ) '_' val num2str(i)];
113                             end
114                         else
115                             if i <= 9
116                                 nena{v} = ['N1_' F(c,1:2) num2str(NumNe(l)) '_
                                        ' val '0' num2str(i)];
117                             else
118                                 nena{v} = ['N1_' F(c,1:2) num2str(NumNe(l)) '_
                                        ' val num2str(i)];
119                             end
120                         end
121                         v = v+1;
122                     end
123                 end
124                 % Crea el archivo de Excel con todos los parametros.
125                 EW1C(nena,testMSE,trainingMSE,rMSE,R2,IRGT,F(c,1:2),val,acc5,
                        acc1);
126                 clc;
127             otherwise
128                 disp('other value')
129         end
130     end
131 end
```

A.2. Función principal RedValid.m

```
1 function [tr,performance,testPerformance,net,r,testTargets] = RedValid(NEU,p,
      norm_data,HOV,FCN)
2 % Esta es una version ligeramente modificada del script que viene incluido en
      el Neural Network Toolbox de MATLAB. Fue adaptado para automatizar y
      facilitar la creacion de redes.
3 % Aqui se cargan las entradas (x) y salidas (t).
4 x = p;
5 t = norm_data;
6 % Create a Fitting Network
```

```
% Para mas de una capa cambiar NEU con valor de Matriz [C1 C2 C3]
net = fitnet(NEU);
% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};
% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
if HOV == 1
    net.divideParam.trainRatio = 75/100;
    net.divideParam.valRatio = 25/100;
    net.divideParam.testRatio = 25/100;
else
    net.divideParam.trainRatio = 50/100;
    net.divideParam.valRatio = 0/100;
    net.divideParam.testRatio = 25/100;
end
% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = FCN;  % Levenberg-Marquardt
% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};
if HOV == 1
    net.trainParam.epochs = 1000;
else
    net.trainParam.epochs = 100;
end
% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);
[r,b,m]=regression(t,y);

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t  .* tr.valMask{1};
testTargets = t  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y);
valPerformance = perform(net,valTargets,y);
testPerformance = perform(net,testTargets,y);
```

A.3. Función secundaria Garson.m

```
function [IRGarson] = Garson(nodos,net)
%j es el numero de neuronas de la red, y net es la red.
weights = getwb(net); %Carga los pesos y las biases de la red.
iw = cell2mat(net.IW); %Pesos de las entradas
lw = cell2mat(net.LW); %Pesos de las capas ocultas
P = zeros;
in = 10; %Numero de entradas
out = 4; %Numero de salidas

for j = 1 : out
    for c = 1 : in
        for i = 1 : nodos
            P(i, c, j) = abs(iw(i,c)) * abs(lw(j,i));
        end
    end
end

suma = zeros;
Q = zeros;

for x = 1 : out
    for c = 1 : in
        for i = 1 : nodos
            suma = zeros;
            for a = 1 : in
                suma = suma + P(i, a, x);
            end
            Q(i, c, x) = P(i, c, x) / suma;
        end
    end
end

S = zeros;
for x = 1 : out
    for c = 1 : in
        suma = 0;
        for i = 1 : nodos
            suma = suma + Q(i, c, x);
        end
        S(c, x) = suma;
    end
end

for x = 1 : out
    suma = zeros;
    for a = 1 : in
        suma = suma + S(a, x);
    end
end
```

```
50
51 IRGarson = zeros;
52
53 for x = 1 : out
54     for c = 1 : in
55         IRGarson(c, x) = S(c, x) * 100 / suma;
56     end
57 end
```