UNIVERSIDAD DE EL SALVADOR
FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA
ESCUELA DE MATEMÁTICA

NOMBRE DEL TRABAJO DE GRADUACIÓN

"ESTIMACIÓN DE LA SUSCEPTIBILIDAD A DESLIZAMIENTOS DE TIERRA EN EL ÁREA METROPOLITANA DE SAN SALVADOR (AMSS) UTILIZANDO REDES NEURONALES ARTIFICIALES"

MAESTRIA EN ESTADÍSTICA

ASESOR:

MSC. MARIO GIOVANNI MOLINA MASFERRER

INVESTIGADOR:

LIC. RICARDO SALVADOR RÍOS MÁRQUEZ

CIUDAD UNIVERSITARIA, 15 de junio de 2012

UNIVERSIDAD DE EL SALVADOR

RECTOR :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIA GENERAL :

DRA. ANA LETICIA DE AMAYA

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA

DECANO :

MSC. MARTÍN ENRIQUE GUERRA CÁCERES

SECRETARIO :

LIC. NELSON GÓMEZ CEDILLOS

ESCUELA DE MATEMÁTICA

DIRECTOR :

DR. JOSÉ NERYS FUNES TORRES

TRABAJO DE GRADUACIÓN APROBADO POR:

COORDINADOR :

DR. JOSÉ NERYS FUNES TORRES

ASESOR :

MSC. MARIO GIOVANNI MOLINA MASFERRER

Índice general

Capítulo 1

INVESTIGACIÓN PRELIMINAR

1.1. INTRODUCCIÓN

El Salvador se encuentra vulnerable ante la ocurrencia de deslizamientos de tierra, en 1982 un flujo de escombros se desencadenó debido al temporal entre los días 17 y 20 de septiembre. En la estación meteorológica de El Boquerón se registraron las intensidades de lluvia mayores del siglo XX en El Salvador, con énfasis entre la noche del 18 y la mañana del 19. El inicio del aluvión fue a 1,600 msnm y descendió por la quebrada el Níspero hasta las zonas urbanizadas sobre 790 msnm en el reparto Montebello. El área del deslave fue de cerca de 700 metros de largo por 65 metros de ancho y un aproximado de 7 metros de profundidad. El volumen estimado de material removido fue de 425,000 metros cúbicos. Los reportes oficiales de los cuerpos de socorro estimaron que la cifra de muertos alcanzaba los 300, pero los habitantes de la zona consideran que pudieron ser mucho más, quizá el doble.

El 13 de Enero de 2001 un terremoto ocurrido en las costas de El Salvador disparo una gran cantidad de deslizamientos de tierra en muchas partes de El Salvador. El mayor deslizamiento de tierra fue en la colonia Las Colinas

en Santa Tecla. El deslizamiento se produjo en una elevación cercana a los 1070 metros y viajo en dirección norte una distancia entre 700 y 800 metros. El volumen del material que movio el deslizamiento fue de 250,000 metros cúbicos. Este terremoto causó 944 perdidas humanas de las cuales un 58 % se generaron en las Colinas. Además la mayor proporción de pérdidas humanas durante el terremoto de enero del 2001 fueron causadas por los deslizamientos que se generaron y por la ocupación humana de espacios cercanos a laderas.

Más recientemente podemos mencionar las lluvias que azotaron el territorio nacional los primeros días del mes de octubre de 2005 las cuales fueron provocadas en su mayor parte por la tormenta tropical Stan. Los informes presentados por el SNET revelaron que después de los primeros cinco días de lluvia, la cantidad de agua acumulada fue de 340 milímetros, la cual estuvo a punto de superar la cantidad registrada durante todo el mes de septiembre de 355.7 milímetros; convirtiéndose así, los primeros días de octubre, en el período más copioso del año. El suelo presentaba niveles críticos de saturación, provocando que los ríos incrementaran sus niveles, y por consiguiente, aumentaran considerablemente las probabilidades de desbordamiento, provocando que el riesgo de deslizamientos en todo el país ascienda del 65 % hasta el 90 %.

Debido a lo anteriormente mencionado se hace necesario implementar mecanismos que nos permitan cuantificar la susceptibilidad de una determinada zona geográfica a deslizamientos de tierra, esto se realizará con la elaboración de mapas de susceptibilidad los cuales presentan de manera gráfica las zonas más propensas a deslizamientos de tierra y representan una práctica herramienta en la planificación urbana y de espacios naturales. Así, puede aplicarse a la determinación de los usos de terrenos, en el diseño de construcción civil y para la planificación de gran variedad de actividades.

Entre las técnicas que recientemente se han usado para estimar la susceptibilidad a deslizamientos de tierra se encuentra la teoría de Redes Neuronales Artificiales (RNA) las cuales han representado un nuevo paradigma de solución de problemas, este nuevo enfoque es inspirado en las redes neuronales biológicas, las RNA han mostrado mayor efectividad para resolver problemas que los enfoques tradicionales no han solucionado o lo han hecho de manera parcial.

En la presente investigación se desarrollará un modelo de RNA que nos permita estimar la susceptibilidad a deslizamientos de tierra en el AMSS; y posteriormente, a partir de la información que nos de el modelo generar el mapa de susceptibilidad a deslizamiento de tierra por medio de algun visualizador de mapas.

1.2. PLANTEAMIENTO

Cuando los suelos y rocas están expuestos en la superficie de la Tierra, la acción desintegradora del agua, viento, sol y materia viviente, conocida por intemperismo, comienza a actuar inmediatamente para establecer un equilibrio entre los materiales térreos y su medio. Otros factores se unen al intemperismo, como por ejemplo la gravedad, para mover los suelos y las rocas hacia los niveles más bajos del relieve. A este movimiento de material en la superficie terrestre, causado por la gravedad, es lo que se denomina movimientos de masa, movimientos gravitatorios, movimientos de ladera, movimientos de terreno, inestabilidad de laderas, deslizamientos de tierra o simplemente, deslizamientos.

Los factores que influyen en los deslizamientos de tierra se clasifican en dos grupos: a) Condicionantes y b) Desencadenantes. Los primeros, también conocidos como pasivos o intrínsicos, son aquellos que dependen de la naturaleza, estructura y forma del terreno, mientras que los segundos, también llamados activos o externos, son factores que actúan desde fuera del medio que se estudia, provocando o desencadenando un deslizamiento.

El Salvador presenta un alto grado de vulnerabilidad debido a la presencia de estos factores, por un lado el territorio salvadoreño se caracteriza por una topografía quebrada y escabrosa y la presencia de cadenas montañosas proclives a deslizamientos de tierra, además de esto se debe mencionar la presencia de factores desencadenantes como son las lluvias y la alta sismicidad, lo cual ha provocado desgracias relacionadas a los deslizamientos de tierra.

Los mapas constituyen el método más efectivo de presentar información referente a la susceptibilidad, peligrosidad y riesgo de una zona o región,

y deben ser usados por planificadores, arquitectos, ingenieros, científicos y técnicos encargados de las labores de emergencia. Tienen por finalidad dividir el territorio en zonas o unidades con diferente grado de susceptibilidad, peligro o riesgo potencial. La susceptibilidad puede definirse como la posibilidad de que una zona quede afectada por un deslizamiento de tierra. Depende de los factores que controlan o condicionan la ocurrencia de este proceso, los cuales, como ya se mencionó, pueden ser intrínsecos o externos.

A través de la superposición del conocimiento de las áreas que han sufrido deslizamientos de tierra junto con la información de los factores intrínsecos de la zona, puede elaborarse un mapa de susceptibilidad que nos permite cuantificar el riesgo de estas zonas a sufrir deslizamientos a tráves de una relación funcional entre los factores intrínsecos y que se presenta acotada, con la cota superior significando riesgo alto y la cota inferior riesgo bajo.

Existen muchos métodos que se han usado para modelar los mapas de susceptibilidad a deslizamientos de tierra, sin embargo, últimamente se han aplicado los modelos conocidos como RNA los cuales representan una mejora, ya que incorporan relaciones no-lineales que los métodos tradicionales les resulta muy difícil modelar; sin embargo, estos modelos tienen el inconveniente de presentar un comportamiento tipo "caja negra", es decir, que una vez estimado uno de estos modelos no se tiene en cuenta la interpretación de sus parámetros y cómo las variables explicativas inciden en la variable respuesta.

Recientemente se han incorporado Metodologías basadas en el concepto de remuestreo que nos permiten realizar ciertas inferencias con respecto al comportamiento e interpretación de las variables explicativas en los modelos de RNA, pero la mayoría de las investigaciones no han venido aplicando estas nuevas metodologías, debido en parte a la carencia de software que las implemente y a su alto costo computacional.

En virtud de lo anterior, el presente trabajo investigativo consiste en construir un modelo de redes neuronales que clasifique a una determinada zona geográfica del área metropolitana de San Salvador si es susceptible o no a deslizamientos de tierra, tomando como base variables tales como: Geomorfología, Pendiente, Aspecto físico, Relieve interno, Densidad del drenaje, Geología, Distancia a la carretera, Distancia a una falla geológica, Modifica-

ción del entorno natural, Precipitaciones, Aceleración que ejerce un terremoto sobre la zona, etc.

1.3. JUSTIFICACIÓN

Ante la gran vulnerabilidad que presenta el territorio de El Salvador a desastres y la situación que viven muchos salvadoreños obligados a vivir en zonas propensas a deslizamientos de tierra, se hace necesario implementar mecanismos que nos permitan identificar estas zonas. En este sentido, la elaboración de mapas de susceptibilidad a deslizamientos de tierra es una metodología que nos permite lograr este fin.

Varias técnicas matemáticas han sido usadas para modelar la susceptibilidad a deslizamientos, entre estas estan: técnicas multivariadas, el método Mora-Vahrson, métodos probabilísticos, lógica difusa y Teoría de Dempster Shafer.

En años recientes, la teoría de RNA ha encontrado una amplia aplicación en modelar la susceptibilidad de deslizamientos de tierra. Esto se debe a que las RNA son aproximadores universales de funciones y así ellos pueden encontrar amplia aplicabilidad en complejos problemas no-lineales como el de la susceptibilidad a deslizamientos de tierra.

Usualmente cuando se utiliza RNA el nivel de análisis y discusión de los resultados es mucho menor que con las técnicas estadísticas clásicas. Probablemente esto es debido al gran esfuerzo necesario para implementar estos modelos y la dificultad de aplicar métodos que nos permitan interpretar la conducta de la red. En varios casos los investigadores solo aplican las RNA como una caja negra sin pretender conocer como influencian las variables explicativas a la variable respuesta. Pese a este inconveniente recientemente se ha logrado introducir a la metodología de RNA lo que se conoce como análisis de sensibilidad. Estos métodos se realizan calculando derivadas parciales locales en determinados puntos de la distribución, ya que a diferencia de los modelos lineales, la importancia de un factor puede ser distinta en dos puntos de la distribución. Estos análisis nos permiten llevar a cabo cierta inferencia

acerca de la influencia de cada variable explicativa y tomando en cuenta el potencial uso de estos modelos (planificación, reducción del riesgo), es claro que un modelo no validado y debidamente controlado no puede encontrar amplio uso.

Consideramos que este proyecto reviste interés, ya que tiene valor tanto teórico como práctico, entre las razones principales que motivaron su estudio podemos mencionar dos:

La Proyección Social: El territorio de El Salvador presenta una alta vulnerabilidad a deslizamientos de tierra, por lo que cualquier nueva metodología que permita identificar de forma precisa las zonas más vulnerables a este tipo de sucesos y sus lamentables consecuencias representa un gran aporte para el país.

La Investigación: En este caso se está realizando un proyecto basado en la estadística para apoyar a la mitigación de desastres en nuestro país, enmarcado en la activa línea de investigación conocida como mapas de susceptibilidad a deslizamientos de tierra.

Por todo lo anterior y debido que se cuenta con recursos técnicos tales como amplia bibliografía, base de datos, equipo adecuado, conocimiento de técnicas de programación, herramientas de apoyo como Internet y listas de correo sobre el tema, etc. Por otra parte los recursos económicos que implica el desarrollo del proyecto son mínimos; lo que garantiza la factibilidad de, llevar a cabo este proyecto según lo planificado.

1.4. OBJETIVOS

1.4.1. GENERALES

1. Construir un modelo de RNA para la evaluación de la amenaza a deslizamientos de tierra en el AMSS.

1.4.2. ESPECÍFICOS

1. Crear una base de datos con la información existente acerca de los elementos necesarios para el estudio de la amenaza a deslizamientos de tierra referentes al área de estudio.

2. Desarrollar un paquete en el software estadístico R para la estimación y evaluación del modelo de red neuronal propuesto.

3. Estimar y evaluar un modelo neuronal con la aplicación informática mencionada arriba, para obtener la mejor estimación de probabilidad de que una zona en el AMSS sea susceptible a deslizamientos de tierra.

4. Presentar el resultado de la estimación de la susceptibilidad a deslizamientos de tierra en el AMSS mediante un vizualizador gratuito de mapas.

1.5. METODOLOGIA

Se parte del modelo neuronal denominado Perceptron Multicapa (MLP) con función de activación de tipo sigmoideo, en la cual las variables explicativas o de entrada son cuantitativas y la variable respuesta o de salida es binaria. La topología o arquitectura del modelo estará compuesta por una capa de entrada, una capa oculta y una capa de salida. Para desarrollar la investigación seguiremos una metodología estructurada la cual consta de las siguientes fases:

1.5.1. Preparación de la base de datos y selección de variables relevantes

En nuestra investigación, primero se realizará un levantamiento de la información correspondiente a las variables de interés a partir de información almacenada en bases de datos geográficas. Después de obtener estos datos se procederá a realizar un análisis exploratorio de datos y posteriormente el análisis respectivo de aquellas variables explicativas que están relacionadas con nuestra variable respuesta.

Las variables de entrada que hemos considerado para nuestra investigación se pueden clasificar en tres categorías:

1. Condiciones del terreno

   Las condiciones del terreno representan características geomorfológicas, geológicas y el tipo de tierra.

2. Distancia a factores relacionados

   Otro grupo de potenciales factores causales son aquellos relacionados a la distancia a carreteras y a fallas geológicas.

3. Factores geomorfométricos

   Estos se refieren a factores como puntos de elevación, líneas de contorno, red de drenaje y lagos.

Considerando las categorias de variables anteriormente descritas podemos formular la susceptibilidad a deslizamientos de tierra de una determinada zona como un modelo MLP con una neurona en la capa de salida que representa la probabilidad de ocurrencia a deslizamientos de tierra de la zona (Entre más cerca está a 1, la zona es más susceptible a deslizamientos de tierra y si está más cerca a 0 la zona no es susceptible a deslizamientos de tierra).

Se dispone de una base de datos que proviene del SNET-MARN la cual contiene las siguientes variables:

1. Ocurrencia a deslizamiento: representa si en la zona geográfica ocurrió un deslizamiento (con valor igual a 1) o no (con valor igual a 0). Esta variable se obtuvo a partir de la identificación visual de las zonas donde hubo deslizamientos de tierra por medio del sistema de información geográfico ILWIS.

2. Geomorfología: se refiere a las formas del relieve que son resultado de la dinámica litosférica de la zona geográfica. Fuente SNET-MARN

3. Pendiente: Derivada del modelo de elevación digital del SNET-MARN.

4. Geología: Descripción de la geología de la zona a partir del mapa geológico de la misión alemana (escala 1:100000). Fuente SNET-MARN

5. Precipitaciones: Precipitaciones máximas registradas en la zona geográfica.

6. Aceleración que ejerce un terremoto sobre la zona: Aceleracion máxima del suelo (en GAL) para un período de retorno de 500 años, esto es lo mínimo que se tiene con informacion detallada. Esta información fue obtenida a partir del proyecto RSIS II. Fuente SNET-MARN

7. Distancia a la red vial: Distancia en kilómetros a la carretera más cercana.

8. Distancia a una falla geológica: Distancia en kilómetros a la falla geológica más cercana. La base de fallas geológicas usadas provienen del mapa geológico de la misión alemana (1:100000). Fuente SNET-MARN

1.5.2. Elección de los conjuntos de aprendizaje, test y validación

Una vez que hemos obtenido los datos con los cuales vamos a realizar nuestro análisis, procederemos a dividirlos en tres grupos: uno de aprendizaje, uno de test y uno de validación. De esta forma, se procederá al entrenamiento del MLP solamente con los datos del conjunto de aprendizaje, mientras que se comprobará la capacidad predictiva o de generalización de la red con los datos del conjunto de test. En caso de tener pocos datos con respecto al número de variables consideradas se tendrá que usar una técnica basada en el bootstrap.

1.5.3. Búsqueda de software para estimar el modelo propuesto

Se realizará una búsqueda de los posibles programas que nos permitan estimar el modelo propuesto, esta búsqueda se orientará bajo la premisa de que los programas sean de distribución libre, lo cual puede posibilitar la sustentabilidad desde el punto de vista económico de futuras investigaciones relacionadas con la estimación de la susceptibilidad a deslizamientos. Actualmente se dispone como posible software el R, que permite estimar el modelo.

1.5.4. Preprocesamiento de los datos

Esta fase es muy importante para la posterior estimación de los parámetros del perceptron multicapa, ya que nos permite reducir el espacio de búsqueda de los parámetros. En esta etapa se debe tener en cuenta que dado que nuestro problema es de clasificación binaria la mejor elección será estandarizar las variables de entrada, esto se debe al siguiente hecho: Supongamos que tenemos un modelo MLP con una capa oculta aplicado a un problema de clasificación binaria y por consiguiente se está interesado en los hiperplanos generados por cada neurona en la capa oculta. Cada hiperplano es el con-

junto de puntos donde las entradas de la red a la neurona de la capa oculta son cero (ver la ecuación (2.3)) y por lo tanto se puede considerar como una región de clasificación generada por las neuronas ocultas consideradas de forma aislada. Los pesos de las entradas a las neuronas ocultas determinan la orientación de los hiperplanos. El bías o umbral determina la distancia del hiperplano al origen. Luego si los bías o umbrales son todos pequeños valores aleatorios, entonces todos los hiperplanos pasaran cerca del origen. Por lo tanto, si al realizar la transformación los datos transformados no están centrados en el origen, los hiperplanos pueden fallar en pasar a tráves de la nube de puntos. Con una pobre inicialización, los mínimos locales ocurrirán muy probablemente. En particular, transformar los datos a [-1, 1] trabajará mejor que [0,1], aunque cualquier transformación que centre los datos probablemente será mejor, pero puede ser incluso mejor si en las variables de entrada existen datos atípicos[1].

1.5.5. Proceso de entrenamiento

En esta fase se obtendrá la estimación de los parámetros o pesos que mejor ajusten el modelo, para tal efecto encontraremos el mínimo global de la función de error del MLP; como esta función es no-lineal tenemos que usar un método adecuado para tal efecto, existen multitud de algoritmos para estimar los parámetros o pesos de una red neuronal, pero en el presente trabajo investigativo se decantará por usar algoritmos genéticos sobre los métodos tradicionales, por la ventaja que tienen de poder escapar a mínimos locales de la función de error de la red neuronal.

Uno de los puntos más críticos será la cantidad de neuronas en la capa oculta, pues, no existe una receta que indique el número de neuronas ocultas que para un problema dado deben emplearse. Gracias a algunos resultados como el de Funahashi, se tiene constancia teórica de que basta una capa oculta para resolver cualquier problema de clasificación con un perceptron. Debe recordarse que si se coloca un número excesivo de neuronas ocultas ocurrirá algo parecido a la sobreparametrización en los modelos de regresión, modelos de series de tiempo y aunque ajusten perfectamente, los casos que

[1] http://www.faqs.org/faqs/ai-faq/neural-nets/part2/

nos proporcionan (conjunto de aprendizaje), nos apartamos de la tónica general y fallamos ante nuevos casos (conjunto de test). Por otro lado, si el número de neuronas no es suficiente no obtendremos un error aceptable ni siquiera para los datos que queremos ajustar[2].

Para decidir el número de neuronas ocultas recurriremos a la siguiente técnica:

Prueba y error. Partiendo de un número sumamente pequeño de neuronas ocultas (por ejemplo, dos), se procede a realizar el entrenamiento. El proceso se repite para distintas arquitecturas, cada vez con más neuronas ocultas, hasta llegar a la arquitectura que proporciona el resultado óptimo para los conjuntos de aprendizaje y de test.

1.5.6. Evaluación de resultados

Finalizada la fase de entrenamiento y almacenadas las mejores estimaciones de los parámetros, ya estaremos en disposición de aplicar el modelo sobre casos nuevos (los del conjunto de validación) para medir su eficacia de forma completamente objetiva. Además se realizará un análisis de sensibilidad para determinar como influencian las variables de entrada o explicativas en la salida de la red neuronal. Si se comprueba que se siguen obteniendo buenos resultados, se procederá á presentar la estimación de la susceptibilidad a deslizamientos de tierra en el AMSS mediante un vizualizador de mapas.

[2]Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes Neuronales y sistemas borrosos, Editorial RA-MA, Universidad de Zaragoza 2001, Págs 217-218.

Capítulo 2

FUNDAMENTO TEÓRICO

2.1. INTRODUCCIÓN

En el presente capítulo se expone el fundamento teórico de la investigación. En primer lugar comenzaremos repasando los orígenes y los conceptos básicos de los modelos de redes neuronales. A continuación, expondremos la estimación de los modelos de redes neuronales usando una técnica poco tradicional denominada algoritmos genéticos. También, se exponen los tres tipos de criterio para evaluar una red neuronal: criterio dentro de la muestra, criterio fuera de la muestra y pruebas de significancia y plausibilidad de los resultados.

2.2. TEORÍA DE REDES NEURONALES

2.2.1. Historia de las redes neuronales

Inteligencia Artificial

La historia de la informática tal y como la conocemos hoy en día es reciente. Hacia finales de los años 30 y durante la década de los 40, los trabajos de investigadores como Alan Turing y von Neumann asientan las bases de la informática moderna. En un principio se orienta hacia la computación algorítmica, es decir, la resolución de un determinado problema obteniendo un algoritmo que manipula los datos relativos al problema. El binomio hardware más software se muestra como una potente herramienta para la resolución de problemas que el ser humano no podría resolver o que tardaría mucho tiempo en hacerlo.

La evolución del hardware ha hecho que la potencia de cálculo haya crecido de tal forma que los ordenadores hoy en día son indispensables en muchas áreas de actividad del ser humano. La computación algorítmica, sin embargo, no es suficiente cuando nos enfrentamos a ciertas tareas. Por ejemplo, algo tan sencillo para el ser humano como reconocer una cara de otra persona es el tipo de problema que no es tan fácil ser resuelto por la vía algorítmica. Debido a este tipo de problemas desde finales de los 50 se ha venido investigando en un conjunto de técnicas que utilizan un enfoque diferente para resolver los problemas. Este conjunto de técnicas y herramientas se bautizó con el nombre de Inteligencia Artificial (IA), porque lo que se pretendía era que los ordenadores presentaran un comportamiento inteligente, entendiendo por esto que supieran hacer frente a ciertos problemas de una manera similar a como lo hacen los seres humanos.

Dentro de la IA, se trabajó en dos enfoques distintos. Por un lado, se desarrolló lo que se conoce como el enfoque simbólico. Este enfoque asienta sus bases en la manipulación de símbolos en vez del mero cálculo numérico, tradicional de la computación algorítmica. La realidad se plasma por medio de una serie de reglas. Herramientas como la lógica de predicados, nos permiten

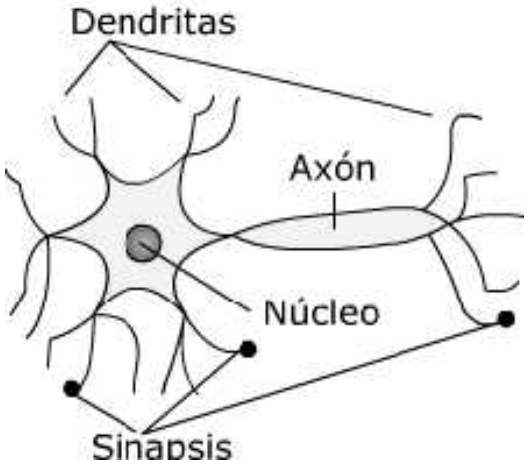
manipular los símbolos y las reglas para obtener nuevas reglas. Este enfoque se presta a ser muy útil en ciertos tipos de problemas, aunque en general tiene la desventaja de que a la hora de buscar la solución a un determinado problema los métodos de deducción presentan una explosión combinatoria, que hace que requiera bastante tiempo de cálculo.

El otro enfoque tradicional es el enfoque conexionista y es donde se encuadran las redes neuronales. La idea aquí es desarrollar un sistema formado por pequeñas unidades de cálculo en cierta medida muy simples y hacer mediante conexiones entre ellas, que todo el conjunto sea capaz de resolver cierta clase de problemas.

Redes Neuronales

La idea que animó el modelo conexionista fue la de imitar el sistema de computación más complejo de los que se conocen hasta ahora, que es el cerebro. El cerebro esta formado por millones de células llamadas neuronas. Estas neuronas son unos procesadores de información muy sencillos con un canal de entrada de información (dendritas), un órgano de cómputo (soma) y un canal de salida de información (axón)

Figura 2.1: Representación de una neurona biológica



La unión o comunicación entre dos neuronas se denomina sinápsis. En

relación a la sinápsis se habla de neuronas presinápticas (La neurona que envía las señales) y postsinápticas (La que recibe las señales). Las sinápsis son unidireccionales, es decir, la información fluye siempre en un único sentido.

De esta forma, las redes neuronales imitan en cierto modo la estructura física y el modo de operación de un cerebro. Teniendo en cuenta que el cerebro presenta las cualidades de procesamiento paralelo, procesamiento distribuido y adaptabilidad, un sistema de redes neuronales tiene, también, estas características.

El sistema resulta ser intrínsecamente paralelo porque esta formado por unidades elementales de procesamiento llamadas neuronas. Cada neurona realiza un tipo de procesamiento muy simple.

El sistema es distribuido. Esto quiere decir que la información no se almacena localmente en ciertas zonas concretas de la red neuronal sino que se halla presente por toda ella, en concreto, se almacena en la sinapsis entre las neuronas. De igual forma, la computación es, también, distribuida. Al calcular la respuesta de la red neuronal, intervienen todos y cada uno de los procesadores elementales, los cuales se hallan distribuidos por toda la arquitectura de la red, además, este carácter distribuido hace que la red presente tolerancia a fallos (sí se pierde una parte de las neuronas no se pierde toda la información).

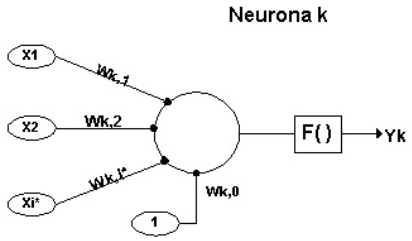
Una red neuronal presenta, además, un grado de adaptabilidad que se concreta en las capacidades de aprendizaje y generalización. Por aprendizaje entendemos la capacidad para recoger información de las experiencias y utilizarlas para actuar ante situaciones futuras. Íntimamente relacionada con el aprendizaje esta la generalización, que podría definirse como la capacidad para abstraer la información útil, más allá de los casos particulares. De esta manera, la red neuronal es capaz de responder ante casos desconocidos.

2.2.2. Conceptos Básicos sobre Redes Neuronales

La neurona artificial

La unidad básica de una red neuronal es la neurona. Aunque hay varios tipos de neuronas diferentes, la más común es la de tipo McCulloch-Pitts. En la figura (2.2) puede verse una representación de la misma.

Figura 2.2: Representación de una neurona artificial tipo McCulloch-Pitts con i* entradas



La representación matemática de la neurona y_k es:

$$n_k = w_{k,0} + \sum_{i=1}^{i*} w_{k,i} * x_i \tag{2.1}$$

$$y_k = f(n_k) \tag{2.2}$$

Una neurona artificial es un procesador elemental, en el sentido que procesa un vector $x(x_1, x_2, ...x_n)$ de entradas y produce una respuesta o salida única. Los elementos claves de una neurona artificial los podemos ver en la figura anterior y son los siguientes:

Las entradas que reciben los datos de otras neuronas en la figura son los elementos etiquetados con x_i con i variando de 1 hasta $i*$, además, existe una entrada con valor constante igual a 1. En una neurona biológica las entradas corresponderían a las dendritas, estas a su vez pueden ser binarias(digitales)

o continuas(analógicas), en los modelos orientados a la predicción se suelen usar variables continuas.

Los pesos sinápticos $w_{k,i}$. Al igual que en una neurona biológica se establecen sinápsis entre las dendritas de una neurona y el axón de otra, en una neurona artificial a las entradas que vienen de otras neuronas se les asigna un peso o factor de importancia, este factor de importancia $w_{k,i}$ representa la intensidad de interacción entre la neurona postsináptica k y la neurona presináptica i, como en nuestro ejemplo solo se trata de una neurona las neuronas presinápticas son las entradas de la neurona. Este peso, se modifica durante el entrenamiento de la red neuronal, y es aquí, donde se almacena la información que hará que la red sirva para un propósito u otro, además, se observa en el ejemplo que existe un peso denotado por $w_{k,0}$ el cual representa la interacción entre la entrada constante con valor *1* y la neurona k, este peso proporciona a la neurona un grado más de libertad y una mayor riqueza computacional.

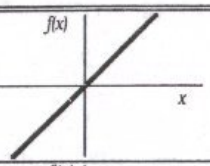
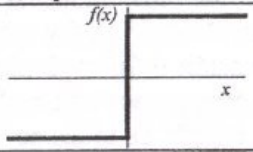
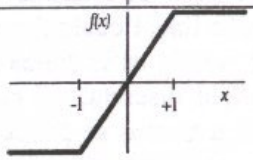
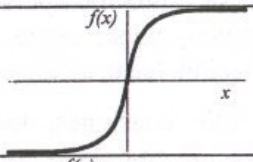
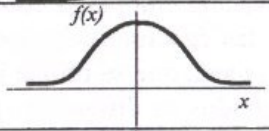
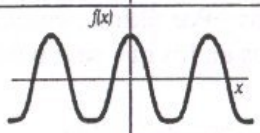
Una regla de propagación. Con esas entradas y los pesos sinápticos, se suele hacer algún tipo de operación para obtener el valor del potencial postsináptico (valor que es función de las entradas y los pesos y que es el que se utiliza en último término para realizar el procesamiento) Una de las operaciones más comunes es sumar las entradas, pero teniendo en cuenta la importancia de cada una (el peso sináptico asociado a cada entrada) es lo que se llama suma ponderada, aunque otras operaciones, también, son posibles, para nuestra investigación la regla de propagación viene representada por la ecuación (2.1).

La otra regla de propagación más habitual es la distancia euclídea.

Una función de activación. El valor obtenido con la regla de propagación, se filtra a través de una función conocida como función de activación y es la que nos da la salida de la neurona. Según para lo que se desee entrenar la red neuronal, se suele escoger una función de activación u otra en ciertas neuronas de la red. En la figura (2.3) se muestran las funciones de activación más usuales.

En muchas ocasiones la razón para la aplicación de una función de activación distinta de la identidad surge de la necesidad de que las neuronas

Figura 2.3: Funciones de activación más usuales

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

produzcan una salida acotada. Esto desde un punto de vista de similitud con el sistema biológico, no es tan descabellado, ya que las respuestas de las neuronas biológicas están acotadas en amplitud. Además, cada neurona tiene asociado un número denominado bías o umbral el cual lo representamos por el peso $w_{k,0}$ que puede verse como un número que indica a partir de que valor del potencial postsináptico la neurona produce una salida significativa, esta interpretación se suele dar cuando la función de activación son de tipo escalón. Más adelante, comentaremos las funciones de activación más usadas para predicción. A continuación mostraremos como simular la función lógica AND a través de una neurona artificial tipo McCulloch-Pitts.

La función lógica AND

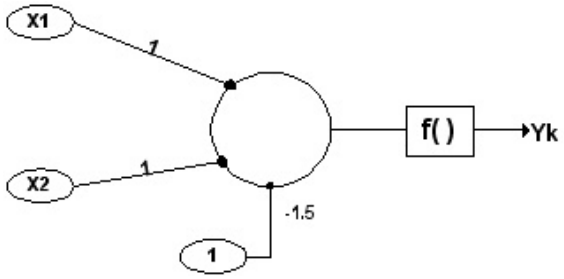
Como ejemplo clásico de los problemas que, en principio, modelaron las redes neuronales y que a su vez nos permitirá mostrar cómo funciona una neurona, simularemos la función lógica AND la cual viene definida por la tabla (2.1).

Tabla 2.1: Función Lógica AND

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

En la figura(2.4) se muestra una neurona que reproduce la tabla lógica AND.

Figura 2.4: Neurona que emula la función lógica AND



Donde:

$$w_{k,1} = 1$$

$$w_{k,2} = 1$$

$$w_{k,0} = -1{,}5$$

$$n_k = w_{k,0} + \sum_{i=1}^{2} w_{k,i} * x_i$$

$$y_k = F(n_k)$$

$$F(n_k) = \begin{cases} 1 & \text{si } n_k \geq 0 \\ 0 & \text{si } n_k < 0 \end{cases}$$

A continuación verificaremos que esta neurona reproduce la tabla lógica AND.

Tabla 2.2: Actividad de la neurona de la figura (2.4)

x_1	x_2	n_k	x_1 AND x_2
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

Por ejemplo para $x_1 = 1$ y $x_2 = 0$ realizamos las operaciones necesarias para calcular la salida de neurona:

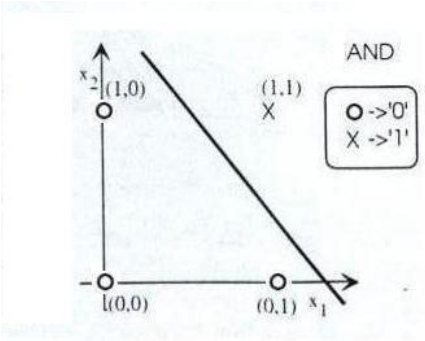
$$n_k = w_{k,0} + \sum_{i=1}^{2} w_{k,i} * x_i$$
$$n_k = -1{,}5 + (1 * 1) + (1 * 0)$$
$$n_k = -0{,}5$$
$$y_k = F(n_k) = 0$$

Con lo que se verifica que efectivamente el valor de 0 se corresponde cuando $x_1 = 1$ y $x_2 = 0$ en la función lógica AND. Si realizamos los cálculos para los otros tres valores observaremos que la neurona reproduce la función lógica AND, a este modelo se le denomina perceptron simple. Si consideramos x_1 y x_2 situadas sobre los ejes de abscisas y ordenadas en el plano, la condición:

$$n_k = w_{k,1}x_1 + w_{k,2}x_2 + w_{k,0} = 0 \tag{2.3}$$

Representa una recta (un hiperplano, si trabajamos con n entradas) que divide el plano en dos regiones, aquellas para las que la neurona proporciona una salida 0 ó 1, respectivamente. Por lo tanto, una neurona tipo perceptron representa un discriminador lineal, al implementar una condición lineal que separa dos regiones en el espacio, que representa dos diferentes clases de patrones. Como podemos observar en la figura (2.5) para la función lógica AND se puede determinar una recta que separe perfectamente dos regiones en el espacio, que representa a las dos diferentes clases de patrones.

Figura 2.5: Función lógica AND



Arquitectura de una red neuronal

Desde un punto de vista matemático, se puede ver una red neuronal como un grafo dirigido y ponderado donde cada uno de los nodos son neuronas artificiales y los arcos que unen los nodos son las conexiones sinápticas. Al ser dirigido, los arcos son unidireccionales. ¿Que quiere decir esto? En el lenguaje de neuronas y conexiones significa que la información se propaga en un único sentido, desde una neurona presináptica (neurona origen) a una neurona postsináptica (neurona destino), por otra parte, es ponderado lo que significa que las conexiones tienen asociado un número real, un peso que indica la importancia de esa conexión con respecto al resto de las conexiones. Si dicho peso es positivo la conexión se dice que es excitadora, mientras que si es negativa se dice que es inhibidora.

Lo usual es que las neuronas se agrupen en capas de manera que una red

neuronal esta formada por varias capas de neuronas. Aunque todas las capas son conjuntos de neuronas, según la función que desempeñan, suelen recibir un nombre específico. Las más comunes son las siguientes:

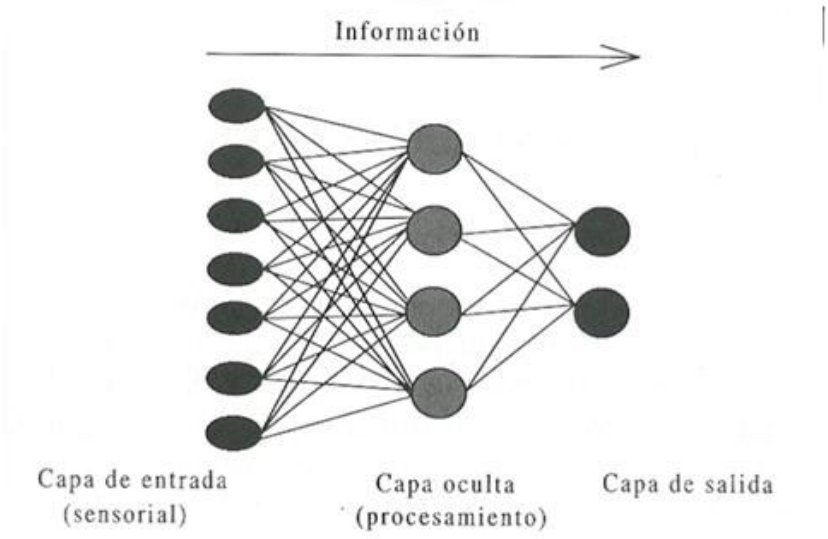
Capa de entrada: las neuronas de la capa de entrada, reciben los datos que se proporcionan a la red para que los procese.

Capas ocultas: estas capas introducen grados de libertad adicionales en la red. El número de ellas puede depender del tipo de red que estemos considerando. Este tipo de capas realiza gran parte del procesamiento.

Capa de salida: Esta capa proporciona la respuesta de la red neuronal. Normalmente, también, realiza parte del procesamiento.

La figura (2.6) ejemplifica lo anteriormente dicho:

Figura 2.6: Tipos de capas que conforman una red neuronal



La diferencia entre las redes neuronales y otros métodos estadísticos radica en que las primeras hacen uso de capas ocultas en la cual las variables de entradas son transformadas por una función de activación, por lo general se usa la función logsigmoidea, en cambio en los métodos estadísticos el pro-

cesamiento de la información es secuencial. Mientras que este enfoque podría parecer esotérico, esto representa un camino muy eficiente para modelar procesos estadísticos no lineales[1].

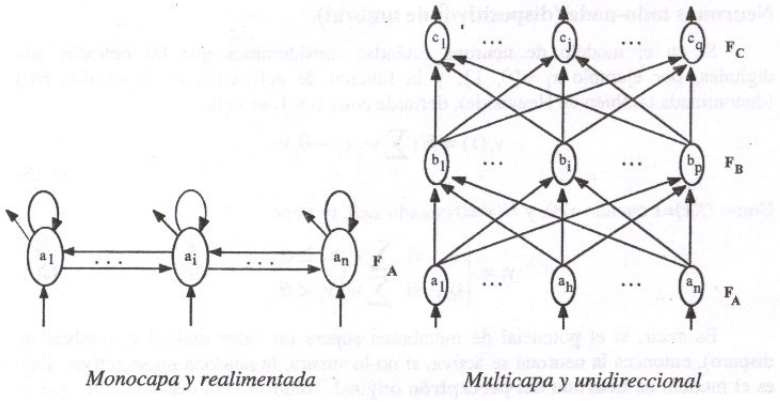
Tipos de redes neuronales

Según el criterio que escojamos para clasificar las redes neuronales tendremos una clasificación u otra. Lo más común es usar la arquitectura y el tipo de aprendizaje como criterios de clasificación.

Si nos fijamos en la arquitectura podemos tener dos posibilidades distintas. Si la arquitectura de la red no presenta ciclos, es decir, no se puede trazar un camino de una neurona a sí misma, la red se llama unidireccional (feedforward).

Por el contrario, si podemos trazar un camino de una neurona a sí misma la arquitectura presenta ciclos. Este tipo de redes se denominan recurrentes o realimentados (recurrent).

Figura 2.7: Representación de redes unidireccionales y realimentadas



En el contexto de nuestro trabajo investigativo nos limitaremos a tratar con redes unidireccionales, ya que presentan un menor grado de complejidad y, además, trataremos con redes con una capa oculta.

[1]Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005, Pág 21.

El otro criterio más habitual para clasificar las redes neuronales es el tipo de aprendizaje que se utilice. Hay cuatro clases de aprendizaje distintos:

Aprendizaje Supervisado: En este tipo de aprendizaje se le proporciona a la red una serie de ejemplos consistentes en unos patrones de entrada junto con la salida que debería dar la red. El proceso de entrenamiento consiste en el ajuste de los pesos, para que la salida de la red sea lo más parecida posible a la salida deseada. Es por ello que en cada iteración se usa alguna función que nos dé cuenta del error o el grado de acierto que se esta cometiendo en la red.

Aprendizaje no supervisado o autoorganizado: En este tipo de aprendizaje se presenta a la red una serie de ejemplos, pero no se presenta la respuesta deseada. Lo que hace la red es reconocer regularidades en el conjunto de entradas, es decir, estimar una función densidad de probabilidad *p(x)* que describe la distribución de patrones x en el espacio de entrada $\mathbb{R}^n$.

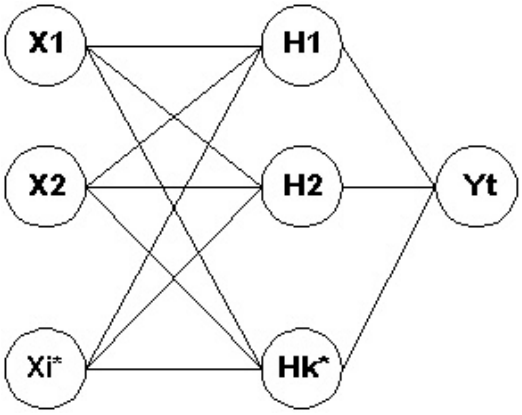
Aprendizaje Híbrido: Es una mezcla de los anteriores. Unas capas de la red tienen un aprendizaje supervisado y otras capas de la red tienen un aprendizaje de tipo no supervisado. Este tipo de entrenamiento es el que tiene las redes de base radial.

Aprendizaje reforzado (reinforcement learning): Es un aprendizaje con características del supervisado y con características del autoorganizado. No se proporciona una salida deseada, pero si se le indica a la red en cierta medida el error que comete, aunque es un error global.

Dentro de nuestro trabajo investigativo usaremos redes unidireccionales con aprendizaje supervisado, en la figura (2.8) se muestra la arquitectura neuronal más frecuentemente usada en problemas de pronóstico.

La red neuronal tiene i^* variables de entrada x_i para $i=1$ hasta $i=i^*$, k^* neuronas ocultas H_k para $k=1$ *hasta* $k=k^*$ y una neurona de salida. Supongamos que de estas variables existen un total de T observaciones, las cuales están enumeradas con un correlativo t , por lo tanto, x_t representa la observación t del vector de entradas y $x_{i,t}$ representa la observación t en la variable i, por lo anteriormente dicho, la actividad de la neurona k la podemos expresar como (2.4):

Figura 2.8: Red neuronal unidireccional multicapa



$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i*} w_{k,i} * x_{i,t} \tag{2.4}$$

Donde $w_{k,i}$ representa el peso entre la neurona de la capa oculta k y la variable de entrada i, $w_{k,0}$ representa el peso de una variable constante con valor igual a 1, luego la salida de la neurona k en la observación t está dado por (2.5).

$$N_{k,t} = L(n_{k,t}) \tag{2.5}$$

Donde L es de la forma:

$$L(n_{k,t}) = \frac{1}{1 + e^{-n_{k,t}}} \tag{2.6}$$

La salida de la red neuronal en la observación t está dada por la ecuación (2.7).

$$y_t = \gamma_0 + \sum_{k=1}^{k*} \gamma_k N_{k,t} \tag{2.7}$$

Donde γ_k representa el peso de la neurona k de la capa oculta a la neurona de salida, y γ_0 representa el peso de una neurona con valor constante igual a 1 de la capa oculta a la neurona de salida, a continuación, comentaremos detalladamente las funciones de activación más comúnmente usadas con la arquitectura neuronal anteriormente propuesta.

Funciones de activación más comúnmente usadas

Como mencionamos anteriormente las neuronas procesan los datos de entrada primero formando combinaciones lineales de los datos de entrada y entonces se acotan estas entradas, a través, de la función log-sigmoidea. Las entradas son entonces transformadas por la función de activación, antes de ser transmitido su efecto a la salida.

El atractivo de la función log-sigmoidea viene de su conducta tipo umbral, la cual, caracteriza a muchos fenómenos económicos. Kuan y White describen la característica tipo umbral como la fundamental característica de los procesos no lineales en el paradigma de las redes neuronales. Ellos describen esto como la tendencia de ciertos tipos de neuronas de estar inactivas a modestos niveles de actividad del vector de entrada, y solo se hacen activas después de que la actividad de las entradas pasa a un determinado limite, mientras que más allá de este umbral el incremento de la actividad en el vector de entrada resulta en pequeños efectos[2].

La ecuacion (2.6) representa la función de activación log-sigmoidea con la forma $\frac{1}{1+e^{-n_{k,t}}}$, en este sistema hay $i*$ variables de entrada $\{x\}$, y $k*$ neuronas. Una combinación lineal de estas variables de entrada en la observación t, $\{\ x_{i,t}\ \}$, $i = 1, \cdots, i*$, y $t = 1, \cdots, T$ con el vector de coeficientes o conjunto de pesos de la capa de entrada observadas $w_{k,i}$ $i = 1, \cdots, i*$, además, el término constante $w_{k,0}$ forma la variable $n_{k,t}$, esta variable es transformada

[2]Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005, Pág 25.

por la función logsigmoidea, y se obtiene la variable $N_{k,t}$ en la observación t . El conjunto de k^* neuronas son combinadas linealmente con el vector de coeficientes $\{\gamma_k\}$, $k = 1, \cdots, k*$, y agregándole el término constante γ_0 , se construye la salida de la red y_t en la observación t . Las redes unidireccionales junto con la función de activación log-sigmoidea son, también, conocidas como perceptron multicapa o MLP.

Otra alternativa como función de activación es la tangente hiperbólica. Esta función acota las combinaciones lineales del vector de entradas al intervalo [-1,1], en vez del intervalo [0,1] de la función log-sigmoidea.

La representación matemática de las redes unidireccionales con esta función de activación, es dada por el siguiente sistema:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i*} w_{k,i} * x_{i,t}$$

$$N_{k,t} = T(n_{k,t})$$

$$T(n_{k,t}) = \frac{e^{n_{k,t}} - e^{-n_{k,t}}}{e^{n_{k,t}} + e^{-n_{k,t}}} \tag{2.8}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k*} \gamma_k N_{k,t}$$

Donde (2.8) es la función tangente hiperbólica para la neurona $n_{k,t}$

Otra función de activación comúnmente usada, es la función acumulada gaussiana, conocida en estadística como la función normal.

La función Gaussiana no es tan usada como la función logsigmoidea, la diferencia principal que presentan estas dos funciones es que la pendiente de la función Gaussiana es mucho más empinada. La representación matemática

de una red unidireccional con función de activación gaussiana está dada por el siguiente sistema:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i*} w_{k,i} * x_{i,t}$$

$$N_{k,t} = \phi(n_{k,t})$$

$$\phi(n_{k,t}) = \int_{-\infty}^{n_{k,t}} \sqrt{\frac{1}{2\pi} e^{-0,5{n_{k,t}}^2}}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k*} \gamma_k N_{k,t}$$

Aunque de todas estas funciones de activación la que más amplio uso tiene es la función de activación log-sigmoidea, muchas veces la elección de una u otra función de activación obedece a las particularidades del problema en cuestión.

La función lógica XOR

La función lógica XOR puede ser simulada a través de una red neuronal con una capa oculta. La arquitectura de la red neuronal esta integrada por: Dos variables de entrada, dos neuronas en la capa oculta y una neurona en la capa de salida, el total de observaciones es T=4. La función lógica XOR viene definida en la tabla (2.3).

La red neuronal representada en la figura (2.9) reproduce la tabla lógica XOR.

Donde los pesos que van de las neuronas de la capa oculta a las entradas estan dados por la tabla (2.4).

Tabla 2.3: Función Lógica XOR

t	x_1	x_2	x_1 XOR x_2
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Figura 2.9: Red neuronal que emula la función lógica XOR

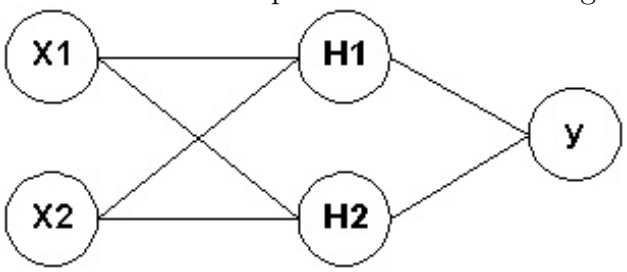


Tabla 2.4: Pesos Sinápticos de las neuronas de la capa oculta a las entradas

$w_{1,0} = -0{,}790$	$w_{2,0} = 0{,}612$
$w_{1,1} = 1{,}7062$	$w_{1,2} = -1{,}6725$
$w_{2,1} = 1{,}1207$	$w_{2,2} = -1{,}0978$

Y los pesos de la neurona de salida a la capa oculta estan especificados en la tabla (2.5).

Tabla 2.5: Pesos Sinápticos de las neuronas de salida a la capa oculta

$\gamma_0 = 1{,}4805$	$\gamma_1 = -1{,}1127$	$\gamma_2 = 1{,}4805$

La representación matemática de la red neuronal es:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{2} w_{k,i} * x_{i,t}$$

$$N_{k,t} = T(n_{k,t})$$

$$y_t = \gamma_0 + \sum_{k=1}^{2} \gamma_k * N_{k,t}$$

Donde $T(n_{k,t})$ es la función de activación tangente hiperbólica para la neurona $n_{k,t}$ la cual viene expresada por:

$$T(n_{k,t}) = \frac{e^{n_{k,t}} - e^{-n_{k,t}}}{e^{n_{k,t}} + e^{-n_{k,t}}}$$

A continuación verificaremos que la red neuronal reproduce la tabla lógica XOR, los cálculos estan desarrollados en la tabla ().

Tabla 2.6: Actividad de la red neuronal de la figura 2.9

t	x_1	x_2	$\sum_{k=1}^{2} \gamma_k N_{k,t}$	$y_t = \gamma_0 + \sum_{k=1}^{2} \gamma_k * N_{k,t}$	x_1 XOR x_2
1	0	0	-1.4805	0	0
2	0	1	-0.4805	1	1
3	1	0	-0.4805	1	1
4	1	1	-1.4805	0	0

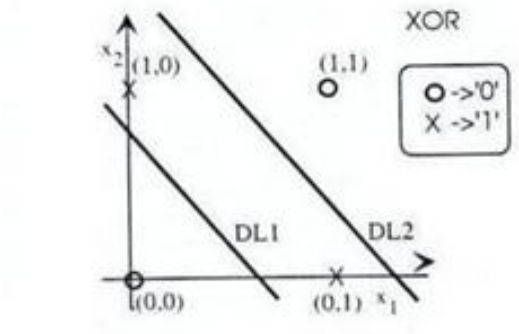
Podemos preguntarnos si el perceptron simple puede reproducir la función lógica XOR, sin embargo, como se puede observar en la figura (2.10) no se puede encontrar una única condición lineal que separe las regiones correspondientes a los valores de salida 0 y 1, por lo que se dice que la función XOR no es separable linealmente. Como el perceptron simple representa un discriminador lineal, no puede implementar la función XOR. Por lo tanto, concluimos que las funciones no separables linealmente no pueden ser representadas por un perceptron simple.

Por lo tanto, el perceptron simple presenta serias limitaciones pues solo puede representar funciones linealmente separables. Así aunque puede representar complejas funciones booleanas o resolver con éxito muchos problemas de clasificación, en otras ocasiones fallará irremediablemente.

"Reflexionemos un poco sobre el problema de la función XOR para intentar encontrar una solución. Un perceptron simple implementa una decisión lineal observando la figura (2.10) podemos encontrar dos perceptrones simples, uno implementa la decisión lineal DL1 y el otro la decisión lineal DL2. Consideremos una capa adicional compuesta por una única capa de neuronas encargada de componer las regiones en las que el plano queda dividido por

las dos neuronas anteriores: si esta neurona se activa únicamente cuando la neurona correspondiente a DL1 esta activada y DL2 desactivada, tendremos una red de tres capas (una de ellas oculta) que implementa la función lógica XOR. Luego una solución a las limitaciones del perceptron simple es incluir mas capas en la arquitectura, con lo que tendremos el modelo conocido como perceptron multicapa"[3]

Figura 2.10: Función lógica XOR

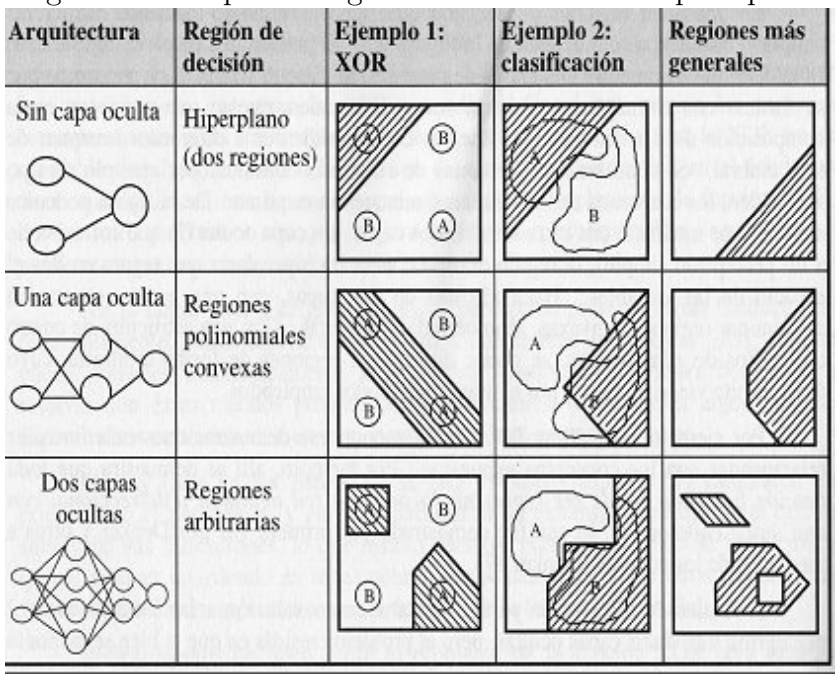


"Los tipos de regiones de decisión que pueden formarse mediante estructuras simples y multicapa se muestran en la figura (2.11). El problema a resolver en este caso es la discriminación entre dos clases de patrones, la clase A y la B. Los contornos que se forman con estructuras multicapa son polinomiales, puesto que consisten en la composición de discriminaciones lineales correspondientes a diferentes neuronas de tipo umbral. Si consideramos neuronas de respuesta continua por ejemplo de tipo sigmoideo, los contornos serán suaves pero sin esquinas. De la figura (2.11) podemos apreciar que mediante una estructura de dos capas, sin capa oculta (la que corresponde al perceptron simple), la región de decisión es un hiperplano que separa en dos el espacio de las variables. Haciendo uso de tres capas, con una oculta se pueden discriminar regiones convexas, sean cerradas o abiertas. Con una estructura de cuatro capas, dos de ellas ocultas se pueden discriminar regiones de forma arbitraria, cuyo único limite viene impuesto por el número de

[3]Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes Neuronales y sistemas borrosos , Editorial RA-MA, Universidad de Zaragoza 2001, Pág 50.

nodos empleados"[4]

Figura 2.11: Tipos de regiones de decisión en el perceptron



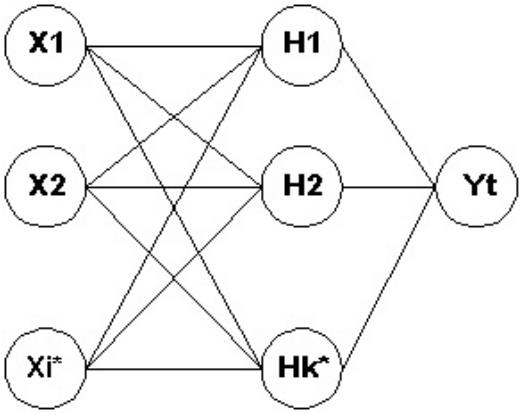
Conexiones con saltos

Hasta este momento hemos visto ejemplos de redes neuronales unidireccionales cuyas conexiones o pesos corresponden a capas que están una a la par de otra, una alternativa a este tipo de redes neuronales es una red unidireccional con conexiones que presentan saltos, es decir, las entradas tienen enlaces o pesos lineales directos a la salida y, además, esta red presenta capas ocultas.

La figura (2.12) muestra una red unidireccional la cual posee conexiones con saltos, con tres entradas, una capa oculta compuesta por dos neuronas (i*=3, k*=2) y una neurona en la capa de salida.

[4]Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes Neuronales y sistemas borrosos , Editorial RA-MA, Universidad de Zaragoza 2001, Pág 51.

Figura 2.12: Red neuronal que tiene conexiones con saltos



La representación matemática de una red unidireccional incorporándole conexiones con saltos, con función de activación log-sigmoidea, es dada por el siguiente sistema:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i*} w_{k,i} * x_{i,t}$$

$$N_{k,t} = \frac{1}{1 + e^{-n_{k,t}}}$$

$$\hat{y}_t = \gamma_0 + \sum_{k=1}^{k*} \gamma_k N_{k,t} + \sum_{i=1}^{i*} \beta_i * x_{i,t}$$

Donde $x_{i,t}$ representa el valor de la variable i en la observación t, $\hat{y}_t$ representa la respuesta que da la red neuronal al presentarle la observación t del vector de entradas. Se observa que en este modelo se incrementa él número de parámetros de la red por $i*$ él número de entradas en la red. La ventaja de trabajar con este modelo es que se anida en el un modelo lineal, además, de la red neuronal, esto permite la posibilidad de que una función no lineal tenga tanto una componente lineal y otra no lineal. Sí el proceso subyacente entre las entradas y las salidas es solo lineal, entonces, el conjunto de coeficientes $\{\beta_i\}$ para $i = 1, \ldots, i*$ deberían ser significativos, sin embargo, si la relación es no lineal uno esperaría que el conjunto de coeficientes $\{w\}$

y $\{\gamma\}$ sean altamente significativos, y el conjunto de coeficientes $\{\beta\}$ ser relativamente insignificantes, finalmente puede existir la posibilidad que la relación entre las variables de entrada y de salida pueda ser descompuesta tanto en su componente lineal como en su componente no lineal, y entonces esperaríamos que todos los tres conjuntos de coeficientes $\{\beta\}$ $\{w\}$ y $\{\gamma\}$ sean significativos.

Una aplicación de agregar conexiones con saltos en las redes neuronales es detectar no-linealidad en la relación entre las variables de entrada y de salida.

Modelos de redes neuronales con respuesta dicotómica

La regresión logística es un caso especial de red neuronal con respuesta discreta, ya que la regresión logística representa una red neuronal con una neurona en la capa oculta. La siguiente adaptación de las redes unidireccionales multicapa (MLP) puede ser usada para un modelo con variable respuesta binaria, donde p_i es la probabilidad de ocurrencia del suceso de interés, para efectos de clasificación si el valor de p_i es mayor que 0.5 entonces p_i es redondeado a 1, en caso contrario el valor de p_i es 0. Para una red con k^* variables explicativas o de entrada y j^* neuronas en la capa oculta, la representación matemática del modelo será:

$$n_{j,i} = w_{j,0} + \sum_{k=1}^{k^*} w_{j,k} x_{k,i}$$
$$N_{j,i} = L(n_{j,i})$$
$$\tilde{p}_i = L(\gamma_0 + \sum_{j=1}^{j^*} \gamma_j N_{j,i})$$

Donde L es la función logística definida como:

$$L(t) = \frac{1}{1 + e^{-t}}$$

A continuación comentaremos dos conceptos muy importantes a la hora de modelar un problema con redes neuronales.

Generalización

Hasta ahora hemos visto que una red neuronal es capaz de "aprender" a representar funciones lógicas como el XOR. El proceso para determinar los pesos de la red consiste en sintetizar la relación entrada-salida, partiendo de un conjunto de datos de entrenamiento. Si este fuese todo el proceso, la red neuronal no sería más que un dispositivo que permite almacenar y evocar información conocida, similar a las memorias.

No obstante, existe una intención más profunda que la anterior, y es el emplear la red neuronal, una vez entrenada, como un dispositivo útil en el procesamiento de información que no ha participado en el proceso de entrenamiento.

En otras palabras, es de interés que la red neuronal sea capaz de "generalizar" el conocimiento adquirido en forma significativa. En este sentido, al proceso para determinar los pesos se le adiciona un proceso de validación, en el cual se determina este grado de generalización. La experiencia ha mostrado que esto puede lograrse en mayor o menor medida, para lo cual existen varias razones y condiciones, como las siguientes:

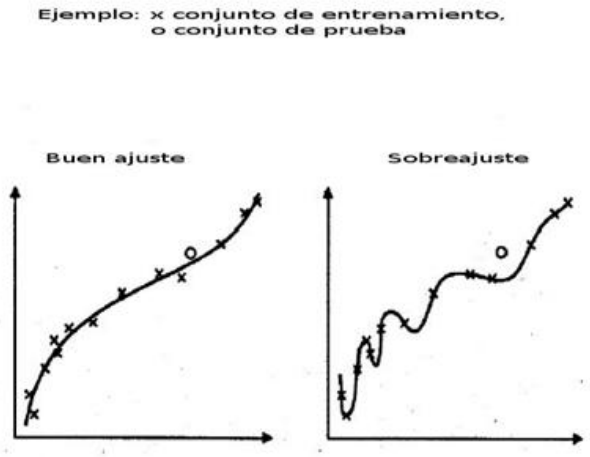
- Los ejemplos sobre los cuales se desea una buena capacidad de generalización no deben diferir mucho de la data de entrenamiento. De manera que esta debe contener la mayor cantidad de información posible sobre el dominio del problema.
- La información requerida para lograr un modo adecuado de generalización no debe estar oculta por otros rasgos o propiedades de los datos de entrenamiento. En este sentido, en muchos casos es necesario un proceso previo de extracción de rasgos.
- La información necesaria para una buena capacidad de generalización debe aparecer más o menos explícita en los datos de entrada. Para ello,

en muchos casos es de relevancia la selección de un buen esquema de representación.

Sobreajuste

A la hora de estimar un modelo neuronal puede ocurrir que el error alcanzado durante la etapa de entrenamiento es muy pequeño, pero frente a nuevos datos (etapa de validación) la red neuronal arroja soluciones con un alto error. La red neuronal memorizó el conjunto de entrenamiento pero no es capaz de generalizar frente a una nueva situación. La generalización no es garantizada cuando el error durante el entrenamiento se reduce a cero. En la figura (2.13) se ilustra el problema del sobreajuste.

Figura 2.13: El problema de un modelo sobreajustado



En el gráfico ubicado a la izquierda se observa que posee buena generalización, ya que el que el patrón etiquetado con "o" es bien aproximado por la función, en cambio en el gráfico de la izquierda se observa que solo ha memorizado los puntos etiquetados con "x", por lo tanto es un ejemplo de un modelo sobreajustado y una situación que debemos evitar.

2.2.3. Estimación de una red neuronal

Las redes neuronales están inspiradas en la biología, de igual manera la mejor forma para estimar estos modelos esta inspirado por la genética y la evolución. Como hemos observado las redes neuronales son sistemas no lineales que aprenden a través de ejemplos, la estimación de modelos no lineales es difícil, debido a problemas de convergencia o convergencia a un mínimo local, A continuación verificaremos los pasos necesarios para ajustar un modelo neuronal y mostraremos que el mejor camino para estimar una red neuronal es aprovechar el poder de búsqueda de los algoritmos genéticos.

Preprocesamiento

Antes de pasar al proceso de estimación debemos primero ajustar los datos, a este proceso se le llama preprocesamiento. Cuando las variables de entrada $\{x\}$ y las variables de salida $\{y\}$ son usadas en una red neuronal, el preprocesado o escalamiento facilita el proceso de estimación no lineal, el cual consiste en modificar los valores de los datos originales para que los valores transformados estén contenidos en un intervalo numérico, el motivo por el cual escalamos los datos es práctico, hasta crucial, la presencia de números con altos o bajos valores o con outlier puede causar problemas como señala Judd[5], la computadora continuará asignando valores de cero a los pesos de la red desde el principio del proceso de estimación ante la presencia de valores muy altos o muy bajos, además, cuando usamos funciones de activación como la log-sigmoidea o la tangente hiperbólica el escalamiento es obligado, si los datos no son escalados a un intervalo como [0,1] o [-1,1] entonces las neuronas ante valores razonablemente grandes responderán con el valor de uno, y ante valores razonablemente pequeños responderán con 0 (para la función de activación logsigmoidea) o –1 (para la función de activación tangente hiperbólica) , si no se escalan los datos posiblemente habrá una gran perdida de información ya que las neuronas simplemente transmitirán valores de menos uno, cero y más uno para muchos valores del vector de entrada.

Dentro del preprocesado o escalamiento existen dos principales rangos

[5]Judd. Kenneth L, Numerical Methods in Economics, MIT Press, 1998.

numéricos que los especialistas en redes neuronales usan en las funciones de escalamiento: cero a uno, denotado por [0,1] y menos uno a mas uno denotado por [-1,1].

Las funciones de escalamiento por lo general hacen uso de los máximos y mínimos valores de los datos. Suponiendo que tenemos la variable x_k y que $max(x_k)$ y $min(x_k)$ son respectivamente los valores máximos y mínimos que puede tomar dicha variable. La función de escalamiento para el intervalo [0,1] transforma a la observación t en la variable $x_{k,t}$ de la siguiente manera:

$$x^*_{k,t} = \frac{x_{k,t} - min(x_k)}{max(x_k) - min(x_k)}$$

La función de escalamiento para el intervalo [-1,1] transforma una variable $x_{k,t}$ en $x^{**}_{k,t}$ de la siguiente manera:

$$x^{**}_{k,t} = 2 * \frac{x_{k,t} - min(x_k)}{max(x_k) - min(x_k)} - 1$$

Finalmente James DeLeo del Instituto Nacional de Salud de Maryland U.S.A sugiere escalar los datos en dos pasos: Primero estandarizar la variable x_k, para obtener la variable z_k y entonces aplicar la función log-sigmoidea a la variable estandarizada z_k:

$$z_k = \frac{x_k - \bar{x}_k}{\sigma_{x_k}}$$

$$x^*_k = \frac{1}{1 + e^{-z_k}}$$

Decir cual de estas funciones de escalamiento trabaja mejor depende de la calidad de los resultados en el proceso de estimación. No existe un método que nos permita decidir cual función de escalamiento trabaja mejor dada las características de los datos. La mejor estrategia es estimar el modelo con

diferentes funciones de escalamiento y examinar cual de ellas tiene un mejor rendimiento.

El problema de la Estimación no lineal

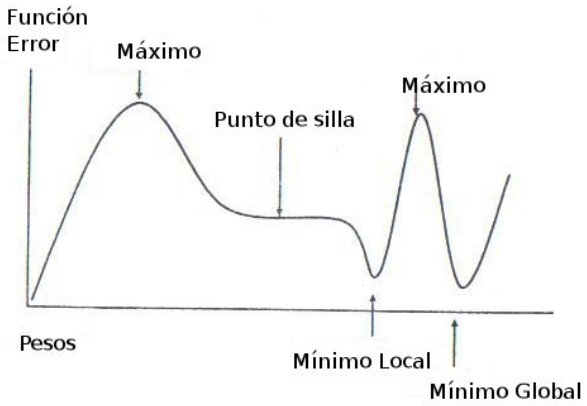
Encontrar los coeficientes para una red neuronal, o cualquier modelo no lineal no es trabajo fácil. Una red neuronal es un sistema no lineal muy complejo. Este sistema podría tener una multiplicidad de soluciones locales optimas, ninguna de las cuales es la mejor solución en términos de minimizar la diferencia entre las predicciones del modelo $\hat{y}$ y los valores actuales y. De esta manera, la estimación de una red neuronal toma tiempo y involucra el uso de métodos alternativos. Brevemente, en cualquier sistema no lineal, nosotros necesitamos iniciar el proceso de estimación con las condiciones iniciales, o valores aleatorios de los parámetros que deseamos estimar. Desgraciadamente, algunos valores iniciales podrían ser mejores que otros para el proceso de estimación de los coeficientes para un pronóstico óptimo. Algunos valores iniciales podrían llevarnos a mínimos locales, que es el mejor pronóstico en el vecindario de los valores iniciales, pero no nos proporciona los coeficientes para el mejor pronóstico si nosotros buscamos un poco mas allá de estos valores iniciales.

La figura (2.14) ilustra el problema de encontrar óptimos globales o mínimos globales en una superficie altamente no lineal.

Como muestra la figura (2.14), un conjunto inicial de pesos o coeficientes de la red en cualquier lugar del eje x podrían falsamente decirnos que estamos cerca del máximo global de la función en vez de estar en un máximo local. Un punto máximo o mínimo tiene una pendiente o derivada igual a cero. En un punto máximo, la segunda derivada o el cambio en la pendiente es negativo, mientras en un punto mínimo el cambio en la pendiente es positivo. En un punto de silla, ambos la pendiente y el cambio en la pendiente son cero.

Cuando los pesos son ajustados o actualizados uno puede atascarse en cualquiera de las muchas posiciones donde la derivada es igual a cero, o cuando la superficie tiene la pendiente aplanada (como lo es en un punto de silla), los ajustes en los pesos deben ser suficientemente estables como

Figura 2.14: Pesos y función de error



para recordar los patrones antiguos (un ritmo de aprendizaje excesivamente grande los borraría, pues, la presente actualización seria de gran magnitud), pero suficientemente plástico como para aprender los nuevos (un ritmo muy pequeño, provoca actualizaciones diminutas, por lo que se corre el riesgo de quedar atrapado en un punto de silla por un largo tiempo, durante el periodo de entrenamiento), lo anterior constituye el dilema de la plasticidad frente a la estabilidad.

Desgraciadamente no existe un método estándar para evitar los problemas de los mínimos locales dentro del problema de la estimación no lineal, sólo existen estrategias que involucran reestimación o búsquedas estocásticas evolutivas.

Se podría encontrar el mínimo o máximo global de una función, pero dado que en un intervalo numérico real existen una cantidad infinita de números, se requerirían tiempos infinitos para llegar a la convergencia al mínimo o máximo global, por lo que nuestro objetivo será encontrar soluciones subóptimas del conjunto de coeficientes o pesos $\Omega = \{w_{k,j}, \gamma_k\}$ de una red con una capa oculta,nosotros minimizaremos la función de perdida ψ, definida como la su-

ma de los cuadrados de las diferencias entre la salida actual observada y, y $\hat{y}$ la salida pronosticada de la red, es decir.

$$min\psi(\Omega) = \sum_{t=1}^{T}(y_t - \hat{y}_t)^2$$

$$\hat{y}_t = f(x_t; \Omega)$$

Donde T es él número de observaciones del vector de salidas y $f(x_t; \Omega)$ es la representación de la actividad de la red neuronal.

Claramente $\psi(\Omega)$ es una función no lineal de Ω Toda optimización no lineal inicia con una solución inicial de la solución, Ω_0, y busca la mejor solución hasta encontrar una posible buena solución dentro de una razonable cantidad de búsquedas.

Para problemas de clasificación binaria se suele definir como función de perdida la entropía que viene dada por:

$$\psi(\Omega) = -\sum_{t=1}^{T}(y_t * log(\hat{y}_t) + (1 - y_t) * log(1 - y_t))$$

Utilizando esta función de perdida conseguimos que las salidas de la red puedan ser interpretadas como probabilidades a posteriori. La entropía puede interpretarse como el grado de información que suministra la observación de la variable respuesta de la red. Cuanto más impredecible y menos estructurada este la variable respuesta de la red, mayor es su entropía.

Para minimizar $\psi(\Omega)$ nosotros emplearemos una búsqueda evolucionaría estocástica comúnmente conocida como algoritmos genéticos, la cual inicia con una población de p supuestas inicializaciones, $[\Omega_{01}, \Omega_{02}, \ldots, \Omega_{0P}]$, y actualiza esta población a través de selección genética, reproducción y mutación por muchas generaciones, hasta que el mejor vector de coeficientes o parámetros es encontrado entre la población que corresponde a la última generación.

Diferencias entre los algoritmos genéticos y los métodos tradicionales de optimización

- Las técnicas evolutivas usan una población de soluciones potenciales en vez de un solo individuo, lo cual las hace menos sensibles a quedar atrapadas en mínimos locales.
- Las técnicas evolutivas usan operadores probabilísticas, mientras las otras técnicas usan operadores determinísticos.
- Las técnicas evolutivas no necesitan conocimientos específicos sobre el problema que intentan resolver.

Ventajas de las técnicas evolutivas

- Simplicidad Conceptual.
- Amplia aplicabilidad.
- Superior a las técnicas tradicionales en muchos problemas de la vida real.
- Tiene el potencial para hibridarze con otras técnicas de búsqueda/optimización.
- Pueden explotar fácilmente las arquitecturas en paralelo.
- Capaces de resolver problemas para los cuales no se conoce solución alguna.

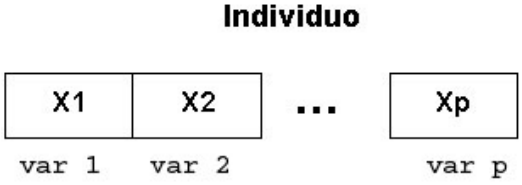
Búsqueda Estocástica Evolucionaría: Algoritmos Genéticos

Los algoritmos genéticos (AG) son métodos sistemáticos para la resolución de problemas de búsqueda y optimización basados en los mismos métodos de la evolución biológica: Selección basada en la población, reproducción sexual y mutación. La principal ventaja de los algoritmos genéticos es reducir la probabilidad de obtener soluciones optimas locales ya que son intrínsecamente paralelos. La mayoría de los otros algoritmos son en serie y sólo pueden

explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo, y si la solución que descubren resulta ser un mínimo o óptimo local, no se puede hacer otra cosa que abandonar todo el trabajo hecho y empezar de nuevo. Sin embargo, ya que los AG tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. Si un camino resulta ser un callejón sin salida, pueden eliminarlo fácilmente y continuar el trabajo en avenidas más prometedoras, dándoles una mayor probabilidad en cada ejecución de encontrar la solución.

Los algoritmos genéticos requieren que los elementos o vectores se codifiquen en un cromosoma. Cada cromosoma tiene varios genes, que corresponden a las variables del problema. Para poder trabajar con estos genes en el ordenador, es necesario codificarlos en una cadena, es decir, una secuencia de símbolos (números) en la figura (2.15) se observa un cromosoma con p variables, para nuestro caso las variables representaran los pesos de la red.

Figura 2.15: Ejemplo de un cromosoma



El algoritmo genético consta de los siguientes pasos:

1. **Creación de la Población**
   Este método inicia con una población N^* de vectores de coeficientes aleatorios. Sea p el número de elementos de cada columna, representando el número total de coeficientes o pesos a estimar en la red neuronal, nosotros creamos una población N^* de vectores aleatorios de dimensión px1.

2. **Selección**
   El siguiente paso es aplicar el operador de selección a dos parejas de coeficientes de la población seleccionadas aleatoriamente, con reemplazamiento. Evaluamos la aptitud o puntuación(fitness) de los vectores de

coeficientes a través de la función de perdida ψ y la cual hemos definido con anterioridad, estos cuatro vectores de coeficientes agrupados en dos parejas, le calculamos la función de perdida. El vector de coeficientes que es más próximo a minimizar la suma al cuadrado de los errores, recibe una puntuación mayor.

Luego se hace un torneo simple entre dos parejas de vectores para observar su aptitud. El ganador de cada torneo es el vector con la mejor puntuación. Estos dos vectores ganadores (i, j) son retenidos para propósitos de cruza o reproducción. Aunque este operador no siempre es usado, la selección a probado ser extremadamente útil para acelerar la convergencia del proceso de búsqueda genética.

3. **Cruza**
   El siguiente paso es la cruza, en la cual dos padres(vectores de coeficientes o pesos) engendran dos hijos(vectores de coeficientes o pesos). No pasa nada si se emparejan dos descendientes de los mismos padres; ello garantiza la perpetuación de un individuo con buena puntuación (algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada inbreeding, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún individuo que, además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización atranque en un mínimo local, y es uno de los principales problemas que debemos estar atentos al aplicar esta técnica. El algoritmo permite que la cruza se realice en cada pareja de los vectores de coeficiente i y j, con una probabilidad prefijada $p > 0$. Si la cruza es realizada, el algoritmo usa una de tres diferentes operaciones de cruza, estos tres métodos tienen una probabilidad de (1/3) de ser seleccionados:

   Cruza aleatoria: Para cada pareja de vectores se realizan k extracciones aleatorias de una distribución binomial. Si la k-ésima extracción es igual a uno, los coeficientes de $\Omega_{i,p}$ y $\Omega_{j,p}$ son intercambiados, en caso contrario no hay cambios.

   Cruza Aritmética:Para cada pareja de vectores un número aleatorio es seleccionado $w \in (0, 1)$. Este número es usado para crear dos nuevos vectores de parámetros que son combinaciones lineales de los vectores

padres de la siguiente manera:

$$(1-w)\Omega_{j,p}, (1-w\Omega_{i,p}+w)\Omega_{j,p}$$

Cruza de punto único: Para cada par de vectores, un número entero I es seleccionado aleatoriamente del conjunto $[1, k-1]$. Los dos vectores son entonces cortados en el entero I y los coeficientes a la derecha de este punto de corte, $\Omega_{i,I+1}, \Omega_{j,I+1}$ son intercambiados.

La cruza de punto único es el método estándar, pero no existe un consenso en la literatura de los algoritmos genéticos de cual cruza es la mejor.

Siguiendo con las operaciones de cruza, los padres son asociados con dos vectores hijos, los cuales son denotados por $C1(i)$y $C2(j)$. Si la cruza ha sido aplicada a parejas de padres, los vectores de hijos deberían generalmente diferir de los vectores de padre.

4. **Mutación**
Consiste en la mutación de los hijos. Con alguna pequeña probabilidad $\tilde{p}_r$ la cual decrece sobre el tiempo, cada elemento o coeficiente de los dos vectores hijos es sometido a mutación. La probabilidad de que cada elemento sea sujeto a mutación en las generaciones $G = 1, 2, \ldots, G^*$, es dada por la probabilidad:

Si la mutación va a ser realizada en un elemento del vector, nosotros usamos el siguiente operador de mutación no-uniforme. Comenzando por extraer aleatoriamente dos números reales r_1 y r_2 del intervalo $[0, 1]$ y un número aleatorio s de una distribución normal estándar.El coeficiente de mutación $\Omega_{i,p}$ es dado por la siguiente formula:

$$\tilde{\Omega}_{i,p} = \begin{cases} \Omega_{i,p} + s[1 - r_2^{(1-G/G^*)^b}] & \text{si } r_1 > 0{,}5 \\ \Omega_{i,p} - s[1 - r_2^{(1-G/G^*)^b}] & \text{si } r_1 \geq 0{,}5 \end{cases}$$

Donde G es él número de generaciones, G^* es él número máximo de generaciones, y b es un parámetro que gobierna el grado, en el cual el operador de mutación actúa de manera uniforme, usualmente se establece b=2. Notemos que la probabilidad de crear a través de mutaciones

un nuevo coeficiente que este lejos del valor actual del coeficiente disminuye cuando $G \rightarrow G^*$. Así la probabilidad de que un elemento mute evoluciona a través del tiempo.

Algunos investigadores como Fogarty[6], han sugerido usar porcentajes altos de mutación al inicio de la búsqueda, y luego decrementarlos exponencialmente para favorecer el desempeño de un algoritmo genético.

5. **Selección mediante Torneo**
   Después de aplicar el operador de mutación, los cuatro miembros de la "familia" (P1, P2, C1, C2) los reunimos en un torneo de aptitud. Los hijos son evaluados por el mismo criterio de aptitud o puntuación usado para evaluar a los padres. Los dos vectores con las mejores aptitudes, ya sea padres o hijos, sobreviven y pasan a la siguiente generación, mientras los dos que tienen peor aptitud se eliminan. Este operador se considera que controla de manera endógena la taza de mutación. Nosotros repetimos el proceso de selección mediante torneo, hasta seleccionar N^* elementos o vectores, con lo que habremos logrado crear una nueva generación.

6. **Elitismo**
   Una vez la siguiente generación es poblada, nosotros podemos introducir elitismo (o no). Evaluamos todos los miembros de la siguiente generación y la pasada generación de acuerdo al criterio de aptitud. Si el mejor miembro de la generación anterior domina al mejor miembro de la nueva generación, entonces este miembro desplaza al peor miembro de la nueva generación y así es elegible para ser seleccionado en la siguiente generación.

Convergencia

Este proceso desde la selección hasta el elitismo es realizado en G^* generaciones. Desgraciadamente, la literatura nos da pocas sugerencias acerca de seleccionar el valor de G^* . Ya que nosotros evaluamos la convergencia por los valores de los mejores individuos de cada generación en la función de perdida ψ , el valor de G^* podría ser muy grande , pero sin embargo, podría pasar

[6]T.C Fogarty, Varying the probability of mutation in the genetic algorithm, 3rd International Conference on Genetic Algorithms, Págs 104-109.

que no veamos ningún cambio en los valores de aptitud del mejor individuo por varias generaciones.

Resumiendo los algoritmos genéticos es un proceso de búsqueda evolutiva para encontrar el mejor conjunto de coeficientes Ω de p elementos, los parámetros de los algoritmos genéticos, tales como el tamaño de la población, probabilidad de cruze, probabilidad inicial de mutación, uso o no de elitismo, pueden evolucionar ellos mismos, sin supervisión externa. Dentro de nuestro trabajo investigativo los algoritmos genéticos serán usados para encontrar un conjunto de pesos óptimos, dejando la arquitectura de la red neuronal fija.

2.2.4. Evaluación de la estimación de una red neuronal

Hasta ahora nosotros hemos discutido la estructura o arquitectura de una red neuronal, además, de cómo entrenar o estimar los coeficientes o pesos de la red. Surge la pregunta: ¿Como nosotros interpretamos los resultados obtenidos por las redes neuronales?

Hay tres tipos de criterios para evaluar una red: Criterio dentro de la muestra, Criterio fuera de la muestra, y pruebas de significancía y plausibilidad de los resultados.

Criterio dentro y fuera de la muestra

Cuando nosotros evaluamos un modelo de clasificación, lo primero que deseamos conocer es el ajuste del modelo con respecto a los datos usados para obtener las estimaciones de los coeficientes. En la literatura de las redes neuronales, este tipo de ajuste es conocido como aprendizaje supervisado. A esto se le llama criterio dentro de la muestra.

Sin embargo, la real prueba para verificar el rendimiento de un modelo neuronal es observar su rendimiento ante datos que no han sido mostrados a la red. Las pruebas fuera de la muestra evalúan que tan bien el modelo generaliza un conjunto de datos que no han sido usados en el proceso de esti-

mación. Buen rendimiento dentro de la muestra, puede simplemente significar que el modelo esta tomando peculiaridades o aspectos idiosincrásicos de una muestra particular o sobreajustandola pero el modelo podría no ajustarse bien al amplio conjunto de datos de la población.

Para evaluar el rendimiento de un modelo fuera de la muestra, comenzamos dividiendo los datos en tres conjuntos uno de estimación dentro de la muestra o conjunto de entrenamiento para obtener los coeficientes, el conjunto de validación que nos permitirá determinar la arquitectura de la red y el conjunto de prueba. Con él último conjunto de datos y con el conjunto de coeficientes estimados por el conjunto de entrenamiento, observamos que tan bien se ajustan estos con el nuevo de conjunto de datos, el cual no tuvo nada que ver en él calculo de los coeficientes de la red.

En la mayoría de estudios con redes neuronales, un relativamente alto porcentaje de los datos, 25 % o más, son dejados afuera o retenidos del proceso de estimación para usarlos en el conjunto de validación y prueba.

Hay parámetros estadísticos los cuales son específicos de problemas de clasificación. Estos parámetros se aplican a cada clase o categoría del problema de clasificación.

Ellos reflejan el rendimiento del modelo tanto dentro de la muestra como afuera de la muestra. En primer lugar, supongamos que tenemos una base de datos de un estudio medico, la cual esta clasificada en dos clases: clase A y clase B. Estas las llamaremos clasificaciones reales. Posteriormente un modelo neuronal fue desarrollado para hacer clasificaciones en la misma base de datos. Supongamos que la red neuronal no necesariamente realiza la clasificación de tal manera que coincidan exactamente con las clasificaciones reales.

Los siguientes 4 parámetros para la clase A pueden ser calculados de la comparación de la clasificación real y la clasificación que proporciona la red.

Verdaderos-positivos: Estos son los casos, los cuales son clasificados por la red como A y que realmente pertenecen a la clase A. La red neuronal responde correctamente en este caso. El ratio de verdaderos-positivos para la clase A es el cociente del número total de verdaderos-positivos de la clase

A clasificados entre el número total de casos tipo A en la clasificación real.

Falsos-positivos: Estos son los casos, los cuales son clasificados por la red neuronal como A, pero estos casos no están realmente clasificados en la clase A (ellos están clasificados en la clase B). La red neuronal responde incorrectamente en este caso; estas son clasificaciones incorrectas. El ratio de falsos-positivos para la clase A es el cociente de el número total de falsos-positivos de la clase A clasificados entre el número total de casos en la clasificación real en todas las clases con excepción de la clase A.

Verdaderos-negativos: Estos son los casos, los cuales no son clasificados por la red neuronal como de la clase A (ellos están clasificados en la clase B), y estos casos realmente no pertenecen a la clase A (ellos están en la clase B). La red neuronal contesta adecuadamente en este caso. El ratio de verdaderos-negativos para la clase A es el cociente de el número total de verdaderosnegativos clasificados entre el número total de casos en la clasificación real en todas las clases con excepción de la clase A.

Falsos-negativos: Estos son los casos, los cuales no son clasificados por la red neuronal como de la clase A (ellos están clasificados en la clase B), pero estos casos realmente pertenecen a la clase A. La red neuronal responde incorrectamente en este caso; estas son clasificaciones incorrectas. El ratio de falsosnegativos para la clase A es el cociente de el número total de falsos negativos de la clase A entre el número total de casos tipo A en la clasificación real.

La misma lógica aplica a la definición de los parámetros de la clase B. Es de aclarar que si solo existen dos categorías o clases, entonces los verdaderos-positivos de la clase A son iguales a los verdaderos-negativos de la clase B y los falsos-positivos de la clase A son iguales a los falsos-negativos de la clase B. Esta propiedad no es valida si hay mas de dos categorías o clases en el problema.

Sensibilidad y Especificidad

En la literatura médica probablemente no encontraremos los términos de verdaderos-positivos, verdaderos-negativos, etc. En lugar de estos se usan los conceptos de sensitividad y especificidad, aplicados a la categoría o clase de interés. Estos son cocientes, calculados de el número de verdaderos-positivos

y falsos-positivos para una clase especifica. Sensibilidad y especificidad son usualmente expresadas como porcentajes, ambos en el rango de 0 % (muy mala clasificación) a 100 % (clasificación perfecta). Las siguientes formulas solo aplican en el caso de tener dos clases o categorías.

Sensibilidad de la clase A = (Número de verdaderos-positivos de la clase A)/(Número total de casos de la clase A de la clasificación real)

Especificidad de la clase A = 1-(Número falsos-positivos de la clase A)/(Número total de casos de la clase B de la clasificación real)

Es fácil observar que por definición la sensibilidad de la clase A es exactamente igual a el ratio de verdaderos-positivos de la clase A. También, por definición, la especificidad de la clase A es exactamente igual al ratio de verdaderos-negativos de la clase A.

En un problema con mas de 2 categorías o clases, la definición de la sensibilidad sigue siendo la misma. La especificidad, sin embargo, es calculada un poco diferente:

Especificidad de la clase A = 1-(Número de falsos-positivos de la clase A)/(Número total de casos de la clasificación real – Número total de casos en la clase A de la clasificación real)

El método bootstrap 0.632

Desgraciadamente, muchas veces los problemas del mundo real no tienen un numero suficiente de observaciones para una buena estimación dentro del conjunto de entrenamiento y para una correcta verificación del modelo en el conjunto de validación.

Un enfoque simple es dividir el conjunto inicial de datos en k subconjuntos de tamaño aproximadamente igual. Entonces se estima el modelo k veces, cada vez dejamos fuera uno de los subconjuntos. Calculamos las medidas de las raíces de los errores cuadráticos medios, para observar el sesgo en el subconjunto omitido. Para un tamaño igual a k del conjunto inicial de datos,

este método es llamado "dejar fuera a uno".

LeBaron propone una prueba mucho más amplia llamada el bootstrap 0.632, debido originalmente a Efron. La idea básica, es estimar el sesgo del conjunto de entrenamiento original repetidamente extrayendo nuevas muestras como conjuntos de estimaciones, con el resto de los datos de la muestra original no apareciendo en el conjunto de nuevas estimaciones. En cada una de las extracciones llevamos la cuenta de los puntos que están en el conjunto de estimación y en el conjunto de validación. Dependiendo de las extracciones en cada repetición, el tamaño del conjunto fuera de la muestra o de validación debería variar. En contraste a la validación cruzada, el método bootstrap 0.632 permite una selección aleatoria de las submuestras para probar el rendimiento del modelo.

El procedimiento bootstrap 0.632[7] aparece en la tabla (2.7).

Criterio interpretativo y significancía de los resultados

En el momento de realizar el análisis, él más importante criterio descansa en las preguntas planteadas por los investigadores. ¿Las redes neuronales se prestan a interpretaciones que tengan sentido para la toma de decisiones? El objetivo de los cálculos y el trabajo empírico es tener una idea de la precisión de los resultados.

Efectivamente la interpretación de un modelo depende del porque se realiza la investigación. Si el objetivo es obtener mejores y más creíbles pronósticos, y nada más, entonces es aplicable la metodología de las redes neuronales.

Nosotros podemos interpretar un modelo de distintas maneras. Una interpretación es simplemente simular un modelo con condiciones iniciales dadas, agregar pequeños cambios a una de las variables y observar como se comporta el modelo. Podríamos, también, estar interesados en conocer si alguna de las variables usadas en el modelo son realmente importantes o estadísticamente significativas. Por ejemplo, ¿Ayuda el desempleo a explicar la inflación?

[7]Lee Baron (1998) señala que el valor de 0.632 viene de la probabilidad que un punto dado este realmente en la extracción de la muestra $1 - [1 - (\frac{1}{n})]^n \approx 1 - e^{-1} = 0{,}632$

Tabla 2.7: Prueba Bootstrap 0.632 para el sesgo fuera de la muestra.

Definición	Operación
Obtener el error cuadrático medio para todo el conjunto de datos	$MSSE^0 = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2$
Extraer una muestra de longitud n con reemplazamiento	z_1
Estimación de los coeficientes del modelo	Ω^1
Obtener los datos omitidos del conjunto total de datos	$\tilde{z}$
Pronosticar fuera de la muestra con los coeficientes Ω^1	$\hat{\tilde{z}}_1 = \hat{\tilde{z}}_1(\Omega)$
Calcular el error cuadrático medio para los datos fuera de la muestra	$MSSE^1 = \frac{1}{n_1}\sum_{b=1}^{B} MSSE^b$
Repetir el experimento B veces	
Calcular el promedio de los errores cuadráticos medios para las B muestras	$MSSE^{'} = \frac{1}{B}\sum_{b=1}^{B} MSSE^b$
Calcular el sesgo ajustado	$\bar{\Omega}^{(0,632)} = 0{,}632[MSSE^0 - MSSE^{'}]$
Calcular el error estimado ajustado	$MSSE^{(0,632)} = 0{,}368 * MSSE^0 + 0{,}632 * MSSE^{'}$

Nosotros podemos simplemente estimar la red con la variable desempleo y entonces podar la red dejando fuera la variable desempleo, estimamos otra vez la red, y observamos si el poder explicatorio global de la red se deteriora después de eliminar esta variable. De esta manera probamos la significancía del desempleo. Sin embargo, este método es difícil de manejar, ya que a menudo los resultados tienden a corromperse en una red, que da buenos resultados, después de eliminar una variable clave.

Otro camino para interpretar un modelo estimado es examinar algunas de las derivadas parciales o los efectos de ciertas variables explicativas en la variable dependiente o respuesta. Por ejemplo, ¿Es el desempleo más importante para explicar la inflación que la tasa de interés? ¿El gasto publico tiene un positivo efecto en la inflación? Con las derivadas parciales, podemos evaluar, la calidad y cantidad, relativas a la intensidad de cómo las variables

explicativas afectan a la variable dependiente.

Es importante proceder con precaución y críticamente. Un modelo ajustado, usualmente una red neuronal sobreajustada, por ejemplo, podría producir derivadas parciales que muestren un incremento firme en el beneficio actual, incrementándose el riesgo de bancarrota. En estimaciones no lineales complejas tal absurda posibilidad pasa cuando el modelo esta sobreajustado con muchos parámetros.

El proceso de estimación debería ser nuevamente realizado podando la red a una más simple, y luego verificar si el resultado de la mala estimación se debe a la presencia de muchos parámetros. Resultados absurdos pueden, también, venir de la falta de convergencia o convergencia a un óptimo local o punto de silla.

En la evaluación del sentido común de un modelo neuronal, es importante recordar que los coeficientes estimados o pesos de la red, los cuales comprenden los coeficientes de las entradas a las neuronas de la capa oculta, y los coeficientes de las neuronas ocultas a la salida, no representan las derivadas parciales de las salidas y con respecto a cada una de las variables de entrada. La estimación de una red neuronal es no paramétrica, en el sentido que los coeficientes no tienen una clara interpretación, como es el caso de los modelos lineales, en los cuales, los coeficientes y las derivadas parciales son iguales.

Así para descubrir si una red neuronal tiene sentido, podemos fácilmente calcular las derivadas asociadas a los cambios en las variables de entrada. Afortunadamente, calcular tales derivadas es una tarea relativamente fácil. Aquí hay dos enfoques: Analítico y método de diferencias finitas.

Una vez obtenidas las derivadas de la red, podemos evaluar su significancía estadística por medio del método bootstrap. A continuación comentamos estos métodos.

Derivadas Analíticas

Uno puede calcular las derivadas analíticas de la salida y con respecto a las variables de entrada en una red unidireccional de la siguiente forma, dada la

red:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i^*} w_{k,i} x_{i,t}$$

$$N_{k,t} = \frac{1}{1 + e^{-n_{k,t}}}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

La derivada parcial de y_t con respecto a $x_{i^*,t}$ es dada por:

$$\frac{\partial y}{\partial x_{i^*,t}} = \sum_{k=1}^{k^*} \gamma_k N_{k,t}(1 - N_{k,t}) w_{k,i^*}$$

Este resultado viene de la aplicación de la regla de la cadena.

$$\frac{\partial y}{\partial x_{i^*,t}} = \sum_{k=1}^{k^*} \frac{\partial y}{\partial N_{k,t}} \frac{\partial N_{k,t}}{\partial n_{k,t}} \frac{\partial n_{k,t}}{\partial x_{i^*,t}}$$

Y del hecho que la derivada de la función $N_{k,t}$ tiene la siguiente propiedad:

$$\frac{\partial N_{k,t}}{\partial n_{k,t}} = N_{k,t}[1 - N_{k,t}]$$

Los modelos lineales implican derivadas parciales que son independientes de los valores de las variables explicativas. Desgraciadamente, con modelos no lineales uno no puede hacer declaraciones tan generales, acerca de cómo las entradas afectan a las salidas sin el conocimiento previo sobre los valores de las variables explicativas.

Diferencias Finitas

El camino más común usado para calcular las derivadas son los métodos de diferencia finita.

Dada la función de una red neuronal, $y = f(x)$, $x = (x_1, x_2, \ldots, x_{i^*})$ un camino para aproximar $f_{i,t}$ es a trávez de la formula de diferencias finitas (2.9).

$$\frac{\partial y}{\partial x_i} = \frac{f(x_1, \ldots, x_i + h_i, \ldots, x_{i^*}) - f(x_1, \ldots, x_i, \ldots, x_{i^*})}{h_i} \tag{2.9}$$

Donde el denominador es el valor de $max(\epsilon, \epsilon x_i)$, con $\epsilon = 10^{-6}$

La segunda derivada parcial es calculada por la expresión dada en (2.10)
.

$$\begin{aligned}\frac{\partial^2 y}{\partial x_i \partial x_j} = \frac{1}{h_j h_i}[&f(x_1, \ldots, x_i + h_i, x_j + h_j, \ldots, x_{i^*}) - f(x_1, \ldots, x_i, \ldots, x_j + h_j, \ldots, x_{i^*}) \\ &f(x_1, \ldots, x_i + h_i, x_j, \ldots, x_{i^*}) - f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_{i^*})]\end{aligned} \tag{2.10}$$

Mientras que la derivada parcial directa de segundo orden es dada por la ecuación (2.11).

$$\begin{aligned}\frac{\partial^2 y}{\partial x_i^2} = \frac{1}{h_i^2}[&f(x_1, \ldots, x_i + h_i, x_j, \ldots, x_{i^*}) - 2f(\ldots, x_i, \ldots, x_j, \ldots, x_{i^*}) + \\ &(x_1, \ldots, x_i - h_i, x_j, \ldots, x_{i^*})]\end{aligned} \tag{2.11}$$

El valor de h_i se establece como $h_i = max(\epsilon x_i, \epsilon)$, donde el escalar ϵ es igual al valor de 10^{-6}.

Importancia
En la práctica, conociendo la forma exacta de la función, las derivadas analíticas nos dan resultados exactos. Sin embargo, para formas funcionales más

complejas, la diferenciación se hace más difícil, y el método de diferencias finitas evita errores que pueden originarse de otras fuentes.

Otra razón para usar el método de las diferencias finitas para calcular las derivadas parciales de una red, es que se puede alterar la forma funcional de la red, o él número de capas ocultas en la red, sin tener que volver a derivar una nueva expresión. Las derivadas analíticas son una mejor elección cuando los valores de las derivadas parciales deben ser exactos o para aumentar la velocidad del programa.

Evaluación de la importancia del método de bootstrap
Evaluar la significancía estadística de una variable de entrada en una red neuronal es un proceso sencillo. Supongamos que tenemos un modelo con varias variables de entrada. Estamos interesados, por ejemplo, en si o no el crecimiento en el gasto publico afecta la inflación. En un modelo lineal, podemos usar el estadístico t , con la estimación no lineal de una red neuronal, sin embargo, el número de parámetros es muy grande. Los estadísticos son a menudo inestables. Un método más seguro, pero que consume mucho tiempo es usar el método original de bootstrap. En este método trabajamos con la muestra completa,$[y, x]$ obteniendo las mejores predicciones, con una red neuronal,$\hat{y}$, y obteniendo el conjunto de residuales, $\hat{e} = y - \hat{y}$, entonces aleatorizamos este vector $\hat{e}$, con reemplazamiento y obtenemos el primer conjunto de efectos para el primer experimento del bootstrap, $\hat{e}^{b1}$. Con este conjunto, generamos una nueva variable dependiente para el primer experimento del bootstrap, $y^{b1} = \hat{y} + \hat{e}^{b1}$, y usamos el nuevo conjunto de datos $[y^{b1}, x]$ para reestimar una red neuronal y obtener las derivadas parciales y otros estadísticos de interés de la estimación no lineal. Repetimos este proceso 500 o 1000 veces, obteniendo e^{bi} y y^{bi} para cada experimento, y su reestimación. A continuación ordenamos el conjunto de derivadas parciales estimadas (además, de otros estadísticos) de menor a mayor y obtenemos una distribución de probabilidad de estas derivadas. De esta distribución podemos calcular los p-valores del bootstrap para cada derivada, dada la probabilidad de que la hipótesis nula de cada una de estas derivadas sea igual a cero.

La desventaja del método de bootstrap es su alto consumo de tiempo, ya que nosotros realizamos un remuestreo de las redes 500 o 1000 veces. Sin embargo, esto es más fiable, si podemos rechazar la hipótesis nula que las derivadas parciales son iguales a cero, basado en el remuestreo de los

residuales originales y reestimando el modelo 500 o 1000 veces, podemos estar razonablemente confiados que los resultados del modelo son significativos.

Implementando el modelo

Cuando se encara la tarea de estimar un modelo, el material precedente indica que tenemos un gran número de elecciones para realizar todas las etapas del proceso, dependiendo si analizamos los datos dentro o fuera de la muestra. Por ejemplo, ¿Que tipo de función de escalamiento deberíamos de usar? ¿Que tipo de arquitectura debemos de usar? ¿Cuándo evaluamos los resultados, a cual diagnostico le podríamos dar mayor importancia y a cual menor importancia? Finalmente, ¿Debemos observar las derivadas parciales?

En general, el objetivo de una investigación con redes neuronales es evaluar su rendimiento con respecto a los modelos lineales. La más simple función de escalamiento debería ser usada primero, a saber, la función lineal de escalamiento lineal en el intervalo [0,1]. Después de esto, podemos verificar la robustez de los resultados en conjunto con respecto la función de escalamiento. Generalmente, la más simple alternativa de arquitectura de red neuronal, debería ser usada, con pocos neuronas al principio de la estimación. Un buen inicio podría ser una red unidireccional o una red unidireccional con conexiones con saltos, las cuales usan combinaciones lineales y las funciones log-sigmoidea.

Para el proceso de estimación la solución no es simple;la mejor opción es usar algoritmos genéticos en el entrenamiento de la red.

Para evaluar el criterio dentro de la muestra, debemos vigilar que las características o patrones que la red haya asimilado sean generalizables a datos no anteriormente mostrados a la red.

Resumiendo, Para evaluar el rendimiento de una red deberíamos de usar tanto un criterio dentro de la muestra, como fuera de la muestra y un criterio de sentido común. Podríamos usar una red neuronal simplemente para pronosticar o simplemente para evaluar propiedades particulares en los datos, tales como la significancía de una o más variables de entrada para explicar

la conducta de la variable dependiente. En este caso, no necesitamos evaluar la red con el mismo conjunto de pesos en los tres criterios. Pero en general nosotros deberíamos estar satisfechos si el modelo cumple los tres criterios.

Algunos aspectos a tener en cuenta para el uso de redes neuronales

Expondremos a continuación en términos generales las características que debe poseer un problema para que su resolución con redes neuronales proporcione buenos resultados, así como aquellas que sugieren que el empleo de estas técnicas puede no resultar aconsejable o viable.

a) Características que debe cumplir el problema:

- No se dispone de un conjunto de reglas sistemáticas que describan completamente el problema.
- En cambio, si se dispone de muchos ejemplos o casos históricos (esta es una condición indispensable para poder aplicar técnicas de redes neuronales).
- Una circunstancia frecuente es que estos métodos proporcionan una alternativa mucho mas rápida y sencilla de desarrollar que otras técnicas convencionales.
- No se necesita tener un conocimiento profundo del problema de estudio.
- Si las condiciones de trabajo son cambiantes se puede hacer uso de la capacidad de la red neuronal para adaptarse a esos cambios reentrenando el sistema con nuevos ejemplos.

b) Características que hacen desaconsejable el empleo de redes neuronales:

- Existe un algoritmo o técnicas estadísticas que resuelve con total eficacia el problema.
- No se dispone de un número adecuado de casos (ejemplos) para entrenar la red neuronal.

- Tareas críticas o potencialmente peligrosas, cuya resolución deba ser siempre perfectamente predecible y explicable. A veces no resulta fácil interpretar la operación de la red neuronal o predecir con total fidelidad el resultado que pueda proporcionar en todos los casos posibles.

Capítulo 3

DISEÑO DE LA INVESTIGACIÓN

3.1. INTRODUCCIÓN

En este tercer capítulo del documento que se refiere al diseño de la investigación, se muestran los aspectos previos para la aplicación del análisis de redes neuronales entre ellos se incluyen la preparación de la base de datos, la selección de variables relevantes, la elección de los conjuntos de aprendizaje y prueba con fines de validación.

3.2. PREPARACIÓN DE LA BASE DE DATOS

Para aplicar la metodología de redes neuronales primero especificaremos la variable dependiente: Ocurrencia a deslizamiento (Ocurrio/No Ocurrio) que es una variable dicotómica; y luego las variables independientes: geomorfología, pendiente, geología, precipitaciones, distancia a la red vial y distancia

a una falla geológica. Se asocia a la base de datos utilizada, un libro de códigos en el que se detallan los nombres de las variables utilizadas, su tipo, su rango de valores, su significado así como la escala que se utiliza. (Ver Anexo A).

3.2.1. TAMAÑO MUESTRAL

Partiendo de la base de datos proporcionada sobre la ocurrencia a deslizamientos de tierra en el AMSS, que contiene 978252 observaciones, de los cuales en 4792 de estas observaciones se pudo observar la ocurrencia a deslizamiento, las variables no presentaron casos perdidos.

A continuación se presenta una tabla donde se observa el cruce de la variable dependiente con las dos variables categoricas ordinales de la investigacion a saber geomorfología y geología.

Tabla 3.1: Datos de la Muestra Total: Ocurrencia a deslizamiento según geomorfología y geologia

		Ocurrencia a deslizamiento		Total
		No ocurrio	Ocurrio	
Geomorfologia	Recuento	973460	4792	978252
	0	36261	0	36261
	3	774584	3454	778038
	5	162615	1338	163953
Geologia	Recuento	973460	4792	978252
	0	37816	0	37816
	1	783	0	783
	2	119547	768	120315
	3	110736	374	111110
	4	658724	3217	661941
	5	45771	433	46204
	6	1	0	1
	7	82	0	82

Como se observa en la tabla anterior en las variables geología y geomorfología existen niveles que no tienen observaciones cuando la variable

respuesta toma el valor de 1 (ocurrencia de deslizamiento), al realizar una interpretacion sobre estos niveles se concluyo que la variable respuesta esta completamente determinada, por lo que estos casos se excluyeron del análisis los cuales representaban aproximadamente un 4 % (38845 casos) del total de observaciones.

3.2.2. ANÁLISIS ESTADÍSTICO UNIDIMENSIONAL

En este apartado se hace un análisis estadístico univariado de cada una de las variables en estudio, con el fin de tener una idea de la información contenida en la base de datos y así poder tomar decisiones con respecto a la inclusión o exclusion de las variables de entrada. Dado que la variable dependiente y las 7 variables explicativas tienen una escala de medida nominal, ordinal y cuantitativa, el análisis se basa en tablas de distribución de frecuencias, diagramas de sectores, histogramas y gráficos de cajas.

En la tabla (3.2) se presenta la distribución de frecuencias para la variable respuesta o de salida **Ocurrencia a deslizamiento**. En ella, se observa que la gran mayoria de las observaciones se dan cuando no ocurre deslizamiento (99.5 %), en cambio cuando ocurre deslizamiento es bastante bajo (0.5 %).

Tabla 3.2: Distribución de frecuencias por Ocurrencia de deslizamiento

Ocurrencia a deslizamiento	Frecuencia	Porcentaje	Porcentaje acumulado
Ocurrio	4792	0.5	0.5
No ocurrio	934615	99.5	100.0
Total	939407	100.0	

En la tabla (3.3) se muestra la distribución de frecuencias para la variable independiente **Geomorfología** Así mismo, en la gráfica se muestra el diagrama de sectores correspondiente a esta variable. La Geomorfología con mayor frecuencia es la 3, también se observa que el 17.4 % de los datos provienen de suelos con geomorfología 5.

En la tabla (3.4) se muestra la distribución de frecuencias para la variable independiente **Geología**. La Geología con mayor frecuencia es la que tiene el valor de 4 con aproximadamente un 70.46 % de los casos.

Tabla 3.3: Distribución de frecuencias por Geomorfología

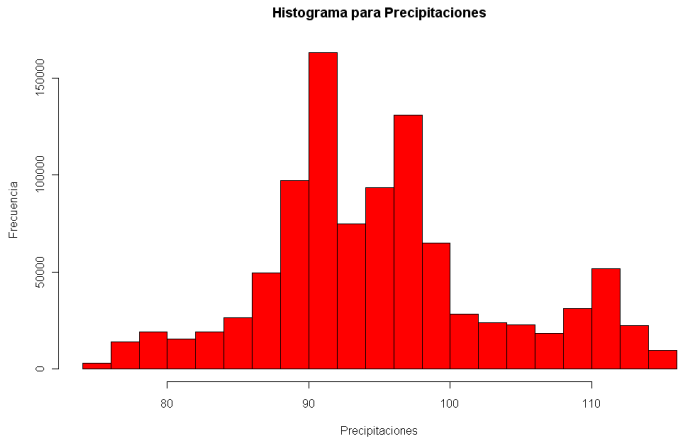
Geomorfología	Frecuencia	Porcentaje	Porcentaje acumulado
3	778038	82.6	82.6
5	163953	17.4	100.0
Total	941991	100.0	

Tabla 3.4: Distribución de frecuencias por Geología

Geología	Frecuencia	Porcentaje	Porcentaje acumulado
2	120315	12.81	12.81
3	111110	11.82	24.63
4	661941	70.46	95.09
5	46204	4.91	100.0
Total	939570	100.0	

En la figura (3.1) se muestra el histograma para la variable precipitaciones máximas, se observa que la máxima frecuencia se encuentre entre los 90 y 92 centimetros cúbicos, la distribución es aproximadamente simétrica y se observa que esta variable presenta una gran variabilidad. En la tabla (3.5) se presentan estadísticos descriptivos relacionados a la variable precipitaciones máximas.

Figura 3.1: Histograma para la variable precipitaciones máximas



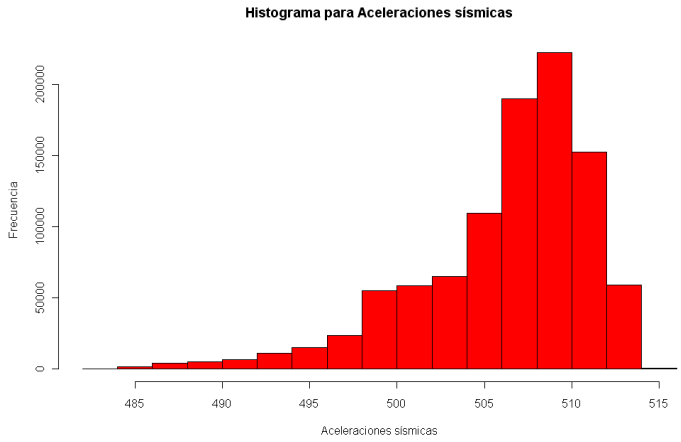
En la figura (3.2) se muestra el histograma para la variable aceleraciones

Tabla 3.5: Estadísticos descriptivos para Precipitaciones máximas

Media	95.06809
Mediana	94.229
Desvación típica	8.30093
Máximo	115.743
Mínimo	74.658
Coeficiente de asimetria de Fisher	0.4007967
Coeficiente de curtosis de Fisher	-0.04417935

sísmicas, se observa que la máxima frecuencia se encuentra entre los 508 y 510, la distribución es sesgada a la izquierda, además se observa que esta variable tiene poca variabilidad. En la tabla (3.6) se presentan estadísticos descriptivos relacionados a la variable aceleraciones sísmicas.

Figura 3.2: Histograma para la variable aceleraciones sísmicas



En la figura (3.3) se muestra el histograma para la pendiente del terreno, se observa que la máxima frecuencia se encuentra entre los 0 y 5 grados, la distribución es sesgada a la derecha y se observa que esta variable presenta poca variabilidad. En la tabla (3.7) se presentan estadísticos descriptivos relacionados a la variable pendiente del terreno.

En la figura (3.4) se muestra el histograma para la distancia a carretera, se observa que la máxima frecuencia se encuentra entre los 0 y 200 kilometros,

Tabla 3.6: Estadísticos descriptivos para Aceleraciones sísmicas

Media	506.2403
Mediana	507.4822
Desvación tipica	4.953141
Máximo	514.0879
Mínimo	483.6721
Coeficiente de asimetria de Fisher	-1.267091
Coeficiente de curtosis de Fisher	1.757101

Figura 3.3: Histograma para la variable pendiente del terreno

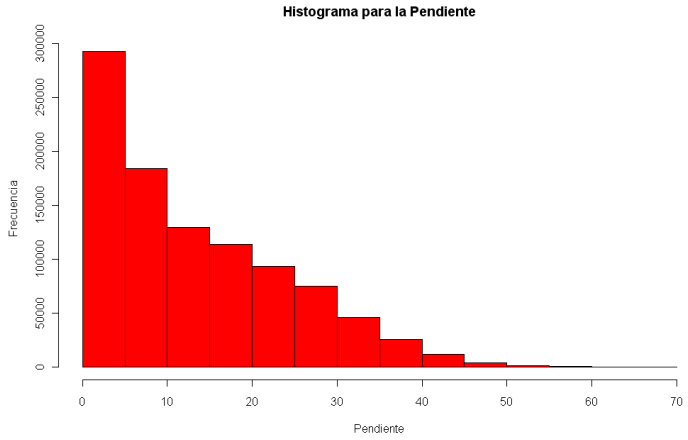


Tabla 3.7: Estadísticos descriptivos para la Pendiente del terreno

Media	13.38699
Mediana	10.45
Desvacion tipica	10.83551
Máximo	66
Mínimo	0
Coeficiente de asimetria de Fisher	0.8514425
Coeficiente de curtosis de Fisher	0.00978272

la distribución es sesgada a la derecha y se observa que en esta variable los datos estan muy concentrados. En la tabla (3.8) se presentan estadísticos descriptivos relacionados a la variable distancia a carretera.

Figura 3.4: Histograma para la variable distancia a carretera

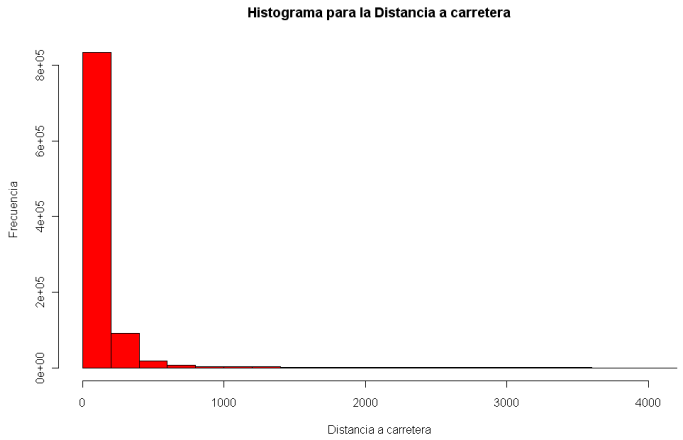


Tabla 3.8: Estadísticos descriptivos para la Distancia a carretera

Media	143.3580
Mediana	58.1
Desvacion tipica	352.4196
Máximo	4123.7
Mínimo	0
Coeficiente de asimetria de Fisher	6.325203
Coeficiente de curtosis de Fisher	46.3671

En la figura (3.5) se muestra el histograma para la distancia a falla geológica, se observa que la máxima frecuencia se encuentra entre los 0 y 200 kilometros, la distribución es sesgada a la derecha y se observa que en esta variable los datos se encuentran muy dispersos. En la tabla (3.9) se presentan estadísticos descriptivos relacionados a la variable distancia a falla geológica.

En la figura (3.6) se pueden observar el gráfico de cajas multiples de todas las variables cuantitativas, se observa que la dispersion de cada una de las variables es distinta, siendo la variable distancia a falla geológica la que presenta una mayor dispersión. Ademas se observa la presencia de una gran cantidad de outliers en las variables distancias a carretera y distancia a falla geológica.

Figura 3.5: Histograma para la variable distancia a falla geológica

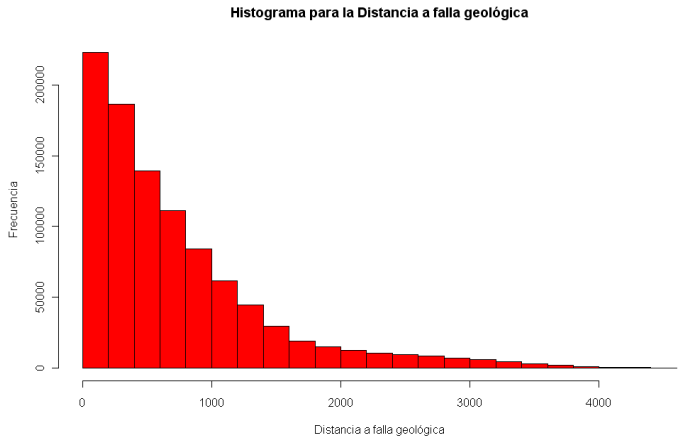


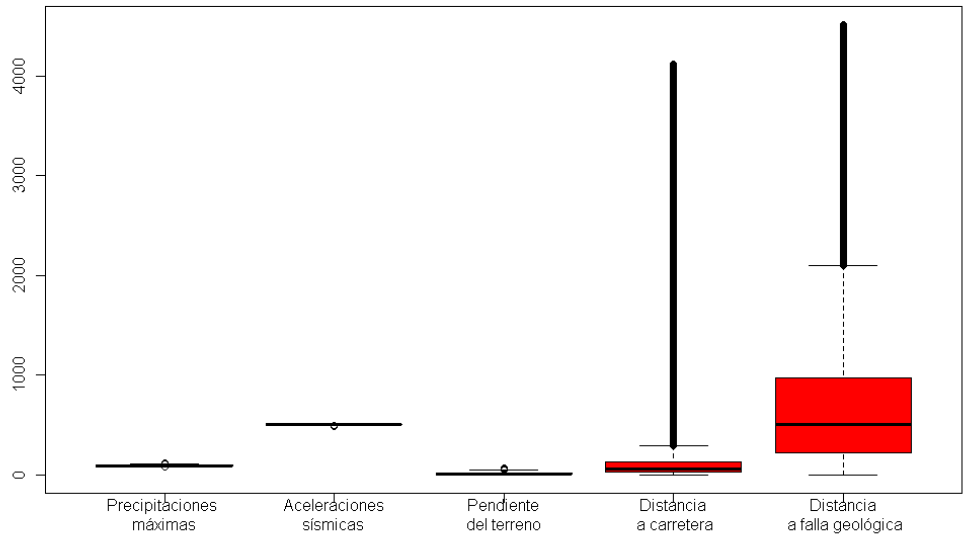
Tabla 3.9: Estadísticos descriptivos para la Distancia a falla geológica

Media	715.5796
Mediana	508.2
Desvación tipica	696.4832
Máximo	4515.7
Mínimo	0
Coeficiente de asimetria de Fisher	1.759303
Coeficiente de curtosis de Fisher	3.375295

3.2.3. ANÁLISIS ESTADÍSTICO BIDIMENSIONAL

En este apartado se presenta un analisis para determinar la relacion existente entre las dos variables explicativas de tipo ordinal, a saber geomorfologia y geologia (Xi) y la variable respuesta Ocurrencia a deslizamiento (Y), en primer lugar obtenemos las respectivas tablas de contingencia y posteriormente realizamos la prueba χ^2 para tablas de contingencia. El objetivo principal de este analisis es estudiar la Hipótesis nula (H_0): (Xi, Y) son independientes versus la Hipótesis alternativa (H_1):(Xi, Y) no son independientes (es decir que estan asociadas). Tambien se realizo un análisis para determinar problemas de multicolinealidad en las variables explicativas.

Figura 3.6: Gráfico de cajas para las variables cuantitativas



En la tabla (3.10) contiene las frecuencias cruzadas entre la Geomorfología y la Ocurrencia a deslizamiento. Así mismo, en la tabla (3.11) se proporcionan los resultados de aplicar la prueba χ^2 de Pearson de independencia.

Tabla 3.10: Tabla cruzada entre Geomorfología y Ocurrencia a deslizamiento

		Ocurrencia a deslizamiento		
		No ocurrió	Ocurrió	Total
3	Recuento	774584	3454	778038
	%	99.5	0.5	100
5	Recuento	162615	1338	163953
	%	99.18	0.82	100
Total	Recuento	937199	4792	941991
	%	99.5	0.5	100

La hipotesis de independencia es rechazada claramente (ver tabla 3.11), ya que el valor de probabilidad o p-valor de 7.976e-14 indica que es bastante improbable que los datos muestrales produzcan una H_0 verdadera. Por otra

Tabla 3.11: Prueba chi-cuadrado de Pearson para Geomorfologia y Ocurrencia a Deslizamiento

Valor	Grados de libertad	p-valor
55.8115	1	0.00000000000007976

parte si establecemos como nivel de significancia $\alpha = 0{,}05$, el valor teórico del estadístico de prueba es $\chi^2_{1,0,95} = 3{,}84$ y por lo tanto $\chi^2_{obs} = 55{,}8115 > 3{,}84$ nos lleva a tomar la decisión de rechazar la hipótesis nula de independencia y aceptar que las variables estan asociadas.

En la tabla (3.12) contiene las frecuencias cruzadas entre la Geología y la Ocurrencia a deslizamiento. Asi mismo, en la tabla (3.13) se proporcionan los resultados de aplicar la prueba χ^2 de Pearson de independencia.

Tabla 3.12: Tabla cruzada entre Geología y Ocurrencia a deslizamiento

		Ocurrencia a deslizamiento		
		No ocurrió	Ocurrió	Total
2	Recuento	119547	768	120315
	%	99.3	0.7	100
3	Recuento	110736	374	111110
	%	99.6	0.4	100
4	Recuento	658724	3217	661941
	%	99.5	0.5	100
5	Recuento	45771	433	46204
	%	99.0	1.0	100
Total	Recuento	934778	4792	939570
	%	99.4	0.6	100

Tabla 3.13: Prueba chi-cuadrado de Pearson para Geología y Ocurrencia a Deslizamiento

Valor	Grados de libertad	p-valor
62.3439	3	0.0000000000001855

La hipótesis de independencia es rechazada claramente (ver Tabla 3.13), ya que el valor de probabilidad o p-valor de 1.855e-13 indica que es bastante improbable que los datos muestrales produzcan una H_0 verdadera. Por otra

parte si establecemos como nivel de significancia $\alpha = 0{,}05$, el valor teorico del estadístico de prueba es $\chi^2_{3,0,95} = 7{,}81$ y por lo tanto $\chi^2_{obs} = 62{,}3439 > 7{,}81$ nos lleva a tomar la decision de rechazar la hipotesis nula de independencia y aceptar que las variables estan asociadas.

A continuación analizaremos las relaciones lineales entre las variables con el objetivo de determinar posible redundancia en los datos. En la tabla (3.14) se observa las correlaciones entre cada una de las variables explicativas.

Tabla 3.14: Matriz de correlaciones

	x1	x2	x3	x4	x5	x6	x7
x1	1.000	0.154	0.069	0.354	0.218	-0.064	0.047
x2	0.154	1.000	-0.295	0.075	-0.190	-0.335	-0.013
x3	0.069	-0.295	1.000	0.255	0.257	0.011	0.024
x4	0.354	0.075	0.255	1.000	0.032	-0.083	0.008
x5	0.218	-0.190	0.257	0.032	1.000	-0.097	-0.111
x6	-0.064	-0.335	0.011	-0.083	-0.097	1.000	0.285
x7	0.047	-0.013	0.024	0.008	-0.111	0.285	1.000

Como se puede observar en la tabla (3.14) las variables presentan bajas correlaciones lineales, siendo la mayor correlación entre la variable geomorfología y aceleraciones sísmicas con un valor de 0.354. Como otra medida adicional para medir el grado en que las variables estan asociadas linealmente empleamos el determinante de la matriz X'X donde la matriz X es la denominada matriz de diseño con la primera columna de unos. Si el determinante de esta matriz es proximo a cero tendremos variables que son combinaciones lineales de otras, al realizar el calculo obtenemos el valor de 1.724242e+63 con lo cual podemos asumir que el conjunto de variables explicativas son aproximadamente independientes.

3.2.4. PARTICIÓN DE LA MUESTRA

La muestra total se dividió en tres submuestras; una, utilizada para la estimación de los parámetros de la red neuronal, la segunda se utilizo para seleccionar el modelo con mayor capacidad de generalización y la última se

utilizo para determinar de forma totalmente objetiva el rendimiento final de la red.

Lo deseable cuando se realiza un modelo de redes neuronales para clasificación binaria es que los patrones correspondientes a los dos niveles de la variable respuesta deben estar en una misma proporción, para que la red neuronal no se especializa en una determinada clase de patrones.

De la muestra total que tiene un total de 939407 observaciones se observo que solo en 4792 de estos casos la variable respuesta presenta ocurrencia a deslizamiento. Dado que en la muestra presenta una gran desproporción de casos donde no hubo deslizamientos frente a donde si hubo deslizamiento, se procedio a equilibrar los niveles de la variable explicativa. El procedimiento consistio en elegir aleatoriamente 4792 observaciones donde no ocurrieron deslizamientos junto con las observaciones donde si ocurrio deslizamiento con un total de 9584 observaciones y repartir estos en igual proporcion entre cada una de las submuestras. Luego estas observaciones se reparten aleatoriamente entre los tres grupos en la proporción 60 %. 20 %, 20 %. En la tabla 3.15 se puede observar como estan distribuidos las observaciones en cada una de las submuestras.

Tabla 3.15: Datos de muestra para estimación, validación y test

	Estimación	Validación	Test	Total
Número de casos	5752	1916	1916	9584
Porcentaje	60 %	20 %	20 %	100 %

3.2.5. PREPOCESAMIENTO A LOS DATOS

Se realizo una estandarización a las variables de entrada, esto con los siguientes beneficios:

- Uniformizar el rango de variación de las variables y por tanto su importancia.
- Mejorar la convergencia del método de estimación de los parámetros del modelo de RNA.

Capítulo 4

PRESENTACIÓN DE RESULTADOS

4.1. INTRODUCCIÓN

En este capítulo se hará un estudio del número de capas ocultas necesarias para lo que es la red neuronal destinada a resolver el problema de determinar la susceptibilidad a deslizamiento de tierra en una determinada zona geográfica del AMSS. Una vez que se determinó el número de capas ocultas óptimo y basádose en que se verificó un desempeño aceptable se procedió a generar el mapa de susceptibilidad para el AMSS, obteniendose resultados acordes con la realidad.

4.2. RENDIMIENTO DENTRO Y FUERA DE LA MUESTRA

Para el proceso de estimación y validación de la red neuronal se desarrolló un paquete en R (la metodología para desarrollar paquetes en R se

comenta en el Anexo B),denominado neurogen (ver Anexo C) cuya finalidad es la de estimar un modelo de redes neuronales para problemas de clasificación binaria.

Además dado que la información de la base de datos de ocurrencia a deslizamiento resulto ser muy grande para su manejo con programas tradicionales de manejos de datos como Excel, se decidió usar la base de datos relacional postgresql la cual esta específicamente diseñada para manejar grandes volumenes de datos, en el Anexo D se encuentra detallada la información almacenada en la base de datos la cual posteriormente fue utilizada en los análisis.

Se tuvo la necesidad de recurrir a un paquete en R para manejo de datos en PostgreSQL, después de ensayar varias alternativos se recurrio a utilizar el paquete RPgSQL el cual al estar escrito en lenguaje C pudo manejar perfectamente el gran volumen de información. Para generar la submuestras correspondientes a los conjuntos de entrenamiento, validación y test se recurrió a implementar un script en R. El script que género los conjuntos de entrenamiento, validación y test esta en el Anexo E.

Para determinar la arquitectura óptima se uso el método de ensayo y error, empezando con arquitecturas con pocas neuronas e incrementando progresivamente el número de neuronas en la capa oculta. En la tabla (4.1) se muestra como es el ajuste de los modelos en el conjunto de entrenamiento.

En la tabla (4.1) se observa que el modelo que tiene 17 neuronas en la capa oculta tiene el mejor rendimiento de todos los modelos con un porcentaje de aciertos en torno al 85 por ciento.

No obstante el resultado anterior, es necesario medir la capacidad de generalización de estas arquitecturas, esto se realiza a trávez del conjunto de validación el cual consiste de observaciones no tomadas en cuenta en el proceso de estimación, al incrementar paulatinamente los parámetros del modelo se llegará a un punto en que el incremento de otra neurona en la capa oculta provocará una perdida en la capacidad de la generalización de la red neuronal es en este momento en el cual dejamos de incrementar los parámetros y nos quedamos con la arquitectura que posea la mejor capacidad de generalización, en la tabla (4.2) se presenta el rendimiento de los modelos

Tabla 4.1: Rendimiento de los modelos en el conjunto de entrenamiento

Rendimiento dentro de la muestra			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
2	0.6815	0.7208	0.7011
3	0.7072	0.7340	0.7206
4	0.7357	0.7493	0.7425
5	0.7757	0.7444	0.7601
6	0.8352	0.7295	0.7823
7	0.8460	0.7246	0.7853
8	0.8227	0.7656	0.7942
9	0.8244	0.7775	0.8009
10	0.8755	0.7886	0.8321
11	0.8508	0.7879	0.8194
12	0.8585	0.7876	0.8230
13	0.8689	0.8126	0.8408
14	0.8738	0.8008	0.8373
15	0.8863	0.8171	0.8517
16	0.8689	0.7803	0.8246
17	0.8839	0.8248	0.8543
18	0.8727	0.8206	0.8467
19	0.8769	0.7914	0.8341

en el conjunto de validación.

Como se observa el modelo con 17 neuronas en la capa oculta es el que tiene una mayor capacidad de generalización por lo que es el modelo que utilizaremos para construir el mapa de susceptibilidad a deslizamientos de tierra.

En la tabla (4.3) se presenta un resumen de este modelo, incluyendo información de la capacidad predictiva del modelo en el conjunto de test(una confirmación adicional de la capacidad predictiva del modelo) y en todo el conjunto de datos. Se observa que el porcentaje de clasificación correcto en todo el conjunto de datos es del 80 por ciento y la sensibilidad del modelo es mayor que la especificidad lo que indica una mayor capacidad del modelo para identificar las zonas con mayor riesgo lo cual es deseable a la otra alternativa

Tabla 4.2: Rendimiento de los modelos en el conjunto de validación

Rendimiento fuera de la muestra			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
2	0.6994	0.7046	0.7020
3	0.6973	0.7171	0.7072
4	0.7035	0.7359	0.7197
5	0.7630	0.7578	0.7604
6	0.8132	0.7276	0.7704
7	0.8184	0.7223	0.7704
8	0.8121	0.7610	0.7865
9	0.8058	0.7683	0.7871
10	0.8549	0.7714	0.8132
11	0.8017	0.7568	0.7792
12	0.8215	0.7797	0.8006
13	0.8225	0.7787	0.8006
14	0.8257	0.7777	0.8017
15	0.8559	0.7860	0.8210
16	0.8278	0.7756	0.8017
17	0.8539	0.8027	0.8283
18	0.8382	0.7662	0.8022
19	0.8382	0.7599	0.7991

(identificar las zonas no susceptibles a deslizamientos de tierra, es decir las zonas seguras) ya que con esto se privilegia el hecho de poder salvar vidas humanas.

Como base de comparación se estimo un modelo de regresión logística. En la tabla (4.4) se presenta un resumen de este modelo en los diferentes conjuntos de datos.

Se observa en la tabla (4.4) que el modelo de regresión logística tiende a clasificar mejor las zonas menos susceptibles a deslizamientos de tierra, es decir la especificidad del modelo es mayor que la sensibilidad, además el porcentaje de aciertos del modelo de regresión logistica no pasa del 68 por ciento una cifra que no iguala en redimiento a el peor de los modelos de redes neuronales anteriormente estimados.

Tabla 4.3: Rendimiento del modelo con 17 neuronas en la capa oculta

Rendimiento dentro de la muestra			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
17	0.8839	0.8248	0.8543
Rendimiento fuera de la muestra			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
17	0.8539	0.8027	0.8283
Rendimiento en el conjunto de test			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
17	0.8591	0.7902	0.8246
Rendimiento con todo el conjunto de datos			
Número de neuronas	Sensibilidad	Especificidad	Porcentaje de aciertos
17	0.8729	0.8009	0.8012

Tabla 4.4: Rendimiento del modelo de regresión logística

Rendimiento dentro de la muestra		
Sensibilidad	Especificidad	Porcentaje de aciertos
0.6602	0.6818	0.6710
Rendimiento fuera de la muestra		
Sensibilidad	Especificidad	Porcentaje de aciertos
0.6649	0.6847	0.6748
Rendimiento en el conjunto de test		
Sensibilidad	Especificidad	Porcentaje de aciertos
0.6382	0.6878	0.663
Rendimiento con todo el conjunto de datos		
Sensibilidad	Especificidad	Porcentaje de aciertos
0.660	0.7006	0.683

Al comparar el modelo de regresión logistica con el mejor modelo de redes neuronales se observa que el modelo de redes neuronales tiene una ganancia del 18, 15 y 16 por ciento con respecto al modelo de regresión logística en los conjuntos de entrenamiento, validación y test respectivamente. Con el conjunto total de datos la ganancia es de alrededor del 12 por ciento lo cual representa una diferencia de 112728 observaciones que se encuentran clasificadas correctamente en el modelo de redes neuronales frente al modelo de regresión logística. Por lo tanto los resultados obtenidos con el modelo de

redes neuronales con 17 neuronas en la capa oculta resultan ser aceptables y seran utilizados para generar el mapa de susceptibilidad a deslizamientos de tierra.

4.3. MAPA DE SUSCEPTIBILIDAD A DESLIZAMIENTO DE TIERRA

Para generar el mapa de susceptibilidad a deslizamiento de tierra primero se tomo en cuenta los datos que se excluyeron del análisis y se le asigno el valor de 0 a la correspondiente probabilidad de ocurrencia a deslizamiento de tierra, luego se cruzo la información de las coordenadas geográficas con la probabilidad a ocurrencia a deslizamiento de tierra utilizando el script escrito en R del Anexo F.

Posteriormente se realizó la discretización de la probabilidad a la ocurrencia a deslizamiento de tierra utilizando la codificación mostrada en la tabla (4.5) esto permitio además darle color al mapa de susceptibilidad a deslizamientos de tierra.

Tabla 4.5: Discretización de la probabilidad a deslizamiento de tierra para levantar el mapa de susceptibilidad a deslizamientos de tierra en el AMSS

Rango de valores	Significado	Color en el mapa
[0,0.25)	Susceptibilidad baja	Verde
[0.25,0.50)	Susceptibilidad media	Marrón
[0.50,0.75)	Susceptibilidad alta	Rojo
[0.75,1]	Susceptibilidad muy alta	Café

Una vez hecho esto se vacío la información en el software de sistemas de información geográfico ILWIS, lo cual generó el mapa de susceptibilidad mostrado en el Anexo G.

En el mapa se observan varias zonas que son susceptibles a deslizamientos de tierra como por ejemplo la grieta en el volcán de San Salvador la cual se pudieron identificar con un alto grado de vulnerabilidad, esto nos indica

que los resultados no estan alejados de la realidad por lo que los resultados de la modelación son aceptables y se pretende analizarlos posteriormente con las autoridades del SNET para valorar su posterior utilidad en otras investigaciones.

Capítulo 5

CONCLUSIONES Y RECOMENDACIONES

- El modelo con 17 neuronas en la capa oculta permite relacionar la variable dependiente ocurrencia a deslizamiento y las variables indepedientes geomorfología, pendiente, geología, precipitaciones, aceleraciones sísmicas, distancia a la red vial y distancia a una falla geológica.
- El modelo con 17 neuronas en la capa oculta representa una mejora de alrededor del 12 % con respecto al modelo de regresión logística.
- La integración del R con el manejador de bases de datos PostgreSQL permitio manejar la gran cantidad de información ya que la mayoria de software estadístico existente no soporta grandes bases de datos.
- La técnica de las redes neuronales es una herramienta con un alto potencial en problemas de clasificación dado que permite relaciones no lineales muy generales, las cuales podrían ser difíciles de modelar usando otras técnicas.
- Dado el elevado costo computacional de los modelos de redes neuronales, se recomienda investigar la implementación de estos en un ambiente paralelo-distribuido.
- No se pudo realizar un análisis de sensibilidad del modelo de redes neuronales para evaluar su significancía estadística por medio del método

bootstrap debido a limitaciones computacionales.

- El mapa de susceptibilidad a deslizamientos de tierra permitió identificar varias zonas catalogadas como de alto riesgo, lo cual es una prueba adicional de la capacidad predictiva del modelo de redes neuronales.
- La integración de la estadística con los sistemas de información geográfico y el manejo de grandes bases de datos constituye un nuevo paradigma en el análisis de datos por lo que se recomienda desarrollar esfuerzos para estudiar esta nueva área de interes.
- Los resultados obtenidos en la presente investigación pueden servir de base para futuras investigaciones relacionadas a la generación de mapas de susceptibilidad en El Salvador.

Bibliografía

[1] Anónimo. Ai faq/neural nets. Sitio web, 2008. `http://www.faqs.org/faqs/ai-faq/neural-nets/`.

[2] BONIFACIO MARTÍN DEL BRÍO Y ALFREDO SANZ MOLINA. *Redes Neuronales y Sistemas Borrosos*. Editorial RA-MA, 2001.

[3] Daniel Peña. *Análisis de Datos Multivariantes*. Mc Graw Hill, 2002.

[4] Douglas C. Montgomery. *Introducción al Análisis de Regresión Lineal*. Compañia Editorial Continental, 2002.

[5] ENRIQUE ARMANDO CASTELLANOS ABELLA. *Multi-scale landslide risk assessment in Cuba*. University of Ultrecht, 2008.

[6] Jean Dickinson Gibbons. *Non Parametric Statistical Inference*. Marcel Dekker, Inc, 1992.

[7] J.P. Marques de Sá. *Applied Statistics Using SPSS, STATISTICA, MATLAB and R*. 2007.

[8] PAUL D. McNELIS. *Neural Networks in Finance*. Editorial Elsevier, 2005.

[9] Pedro Isasi Viñuela. *Redes de Neuronas Artificiales Un Enfoque Práctico*. Pearson Prentice Hall, 2004.

[10] Stuart Russell. *Inteligencia Artificial Un enfoque moderno*. Pearson Prentice Hall, 1996.

[11] Valluru B. Rao. *C++ Neural Networks and Fuzzy Logic*. IDG Books, 1995.

Anexos

Anexo A

LISTADO DE VARIABLES Y SU CODIFICACIÓN

En la tabla (A.1) esta la codificación de las variables e información como el tipo de variable, el máximo y mínimo, así como la escala en la cual esta medidad la variable en el caso de ser una variable cuantitativa.

Tabla A.1: Listado de variables y su codificación

Nombre	Código	Tipo/Explicación	Mínimo	Máximo	Unidades de medición
Geomorfología	X1	Ordinal con valores 0,3 y 5	0	5	
Geología	X2	Ordinal con valores 0,1,2,...,6,7	0	7	
Precipitaciones máximas	X3	Cuantitativa	74.658	115.743	Milimetros cubícos
Aceleraciones sismicas	X4	Cuantitativa	483.67	514.08	
Pendiente del terreno	X5	Cuantitativa	0	66	Grados
Distancia a carretera	X6	Cuantitativa	0	4123.7	Metros
Distancia a falla geológica	X7	Cuantitativa	0	4515.7	Metros
Ocurrencia a deslizamiento	Y	Dicotómica, 0 y 1, 0 no ocurrio deslizamiento, 1 si ocurrio deslizamiento	0	1	

Anexo B

PROCEDIMIENTO DE DESARROLLO DE UN PAQUETE EN R

B.1. POR QUÉ CREAR UN PAQUETE EN R

Existen tres buenas razones para crear un paquete en R:

1. Crear un paquete en R obliga a documentar el código y proveer ejemplos para asegurar que el código esta funcionando correctamente.
2. Si el objetivo es divulgar la investigación, este es un camino ideal para asegurar que otras personas tengan acceso a tu trabajo.
3. Como una forma de organizar tu trabajo al mantener un conjunto de funciones en R en un paquete no te debes de preocupar por buscarlas, ni de preocuparte por recordar que es lo que hacen o como se invocan, solo debes de cargar al paquete y ver su documentación.

B.2. REQUERIMIENTOS DEL SISTEMA OPERATIVO

Antes de explicar el procedimiento de creación de un paquete en R es necesario tener instalado además del R otras herramientas como compiladores, utilidades de programación y librerías para formatear texto. En Linux estas herramientas vienen incluidas por defecto, pero Windows carece de estos componentes. Esto quiere decir que estamos obligados a descargar los equivalentes de estas herramientas si queremos desarrollar un paquete en Windows. A continuación comentaremos el proceso de instalación y configuración de estas herramientas.

Los componentes que se necesitan son los siguientes:

1. Un conjunto de utilidades de Unix (Que Ripley y Murdoch llaman "Rtools")
2. El conjunto de compiladores GNU (Solo si tu paquete contiene código C/C++ /Fortran)
3. Perl (Un lenguaje de programación usado por el R para instalar y verificar un paquete)
4. El Microsoft html help compiler
5. Una version de Tex (MikTex)

Nota: Las siguientes herramientas solo son validas para Windows XP y 2000, ya que versiones anteriores de estos sistemas operativos han reportado problemas en su instalación.

Pasos de instalación y configuración de los componentes:

1. Descargar Rtools.
   Digite en el navegador la siguiente dirección web: `http://www.murdochsutherland.com/`

`Rtools/tools.zip`, guardar el archivo comprimido a un directorio, por ejemplo C:\\Rtools y luego descomprimir el archivo.

2. Descargar e instalar minGW.
   "minGW" significa Minimalist GNU for Windows. Este software incluye los siguientes compiladores: gcc (compilador de C), g++ (compilador de C++) , g77 (compilador de Fortran) y otras herramientas. En el siguiente sitio web `http://www.mingw.org/download.shtml` se obtiene información detallada sobre como instalar esta herramienta.

3. Descargar e instalar Perl.
   Seleccionar la versión de Perl para Windows del siguiente sitio web: `http://www.activestate.com/Products/ActivePerl/` usar la instalación por defecto.

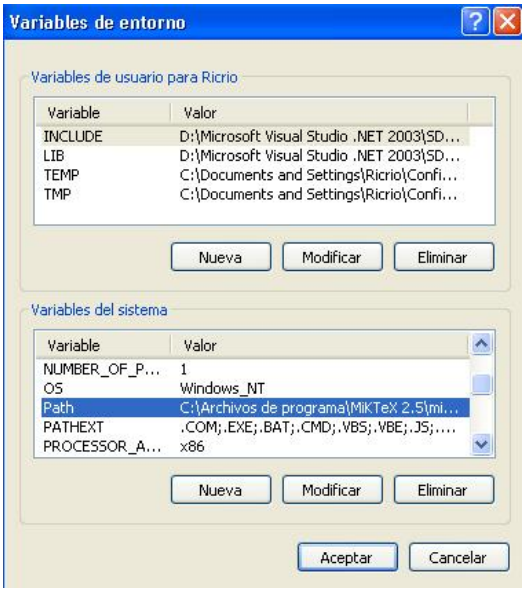
4. Descargar e instalar el Microsoft html help compiler.
   Este software es usado por R para crear archivos de extensión chm. Estos archivos son usados para realizar archivos de ayuda como los que tienen los programas de Windows. Este software puede descargarse del siguiente sitio web: `http://www.murdoch-sutherland.com/Rtools/htmlhelp.exe` una vez ha sido descargado, simplemente hacer doble clic en el instalador y seleccionar las opciones por defecto.

5. Descargar e instalar MikTex.
   Descargar la versión más reciente MikTex del siguiente sitio web: `http://www.miktex.org/` sugerimos descargar la versión básica de MikTex ya que es completamente funcional para nuestros propositos, una vez descargado instalarlo con las opciones que trae por defecto. No obstante que se dan las direcciones web para bajar las herramientas requeridas estas serán proporcionadas en el CD que se anexa al documento final.

6. Modificar la variable de entorno PATH.
   Unix usa una ruta o lista de búsqueda (como hace R) para localizar los archivos a ejecutar desde su prompt. En Windows no existen programas que usan este mecanismo. Por esta razón la mayoría de los usuarios de Windows no se han tenido que preocupar por la variable de entorno PATH. Sin embargo, para que trabajen estas herramientas y los scripts de perl: install, check, y build, es necesario configurar correctamente esta variable.

Para configurar la variable PATH en Windows debes de dar clic derecho al icono Mi Pc ("My Computer" en la versión en ingles de Windows) en el escritorio. En la ventana que aparece se seleccionan las siguientes opciones, propiedades -> opciones avanzadas -> variables de entorno, para finalmente obtener la ventana del entorno de variables que se muestra a continuación.

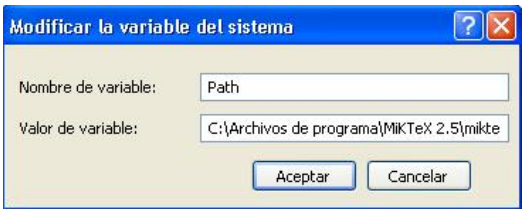
Figura B.1: Ventana del entorno de variables



Para cambiar la variable de entorno se selecciona en la subventana "Variables del sistema" ver figura (B.1) el elemento Path y luego presionar en el boton Edit (Editar en español). En la caja de texto que aparece hay que introducir al inicio los directorios donde se instalaron: MixTex, Rtools, minGW, perl, R y el html help compiler. Los directorios deben estar separados por el carácter ";" debes ser muy cuidadoso al escribir los directorios y no sobrescribir la configuración inicial. En la figura (B.2) se muestra la ventana para modificar la variable de entorno PATH.

Antes que nada hay que hacer un comentario del valor de la variable PATH, las utilidades provenientes de los entornos Unix: MikTex, Perl, minGW, R y Rtools, usan el directorio bin para almacenar los archivos ejecutables. El subdirectorio bin de estas aplicaciones debe estar en el

Figura B.2: Ventana del entorno de variables



Path, la única excepción la constituye el Microsoft html compiler.

Ahora estamos listos para probar si la variable PATH ha sido configurada correctamente. En el menú Inicio en Ejecutar digitamos cmd esto nos abre una consola de MS-DOS, una vez dentro de la consola invocaremos al programa Rterm (la versión de línea de comandos de R). Digitamos R en la consola, si el mensaje que recibimos es "R no se reconoce como un comando interno o externo ...", quiere decir que no se ha configurado correctamente la variable de entorno PATH. En otro caso veras un mensaje de bienvenida y el familiar prompt del R ">" se mostrara en la pantalla.

Después de esto verificaremos que minGW, Perl, MikTex y Rtools estan instaladas. Dentro de la consola o símbolo del sistema digitamos los siguientes comandos de forma separada.

gcc –help (El Compilador de C)
perl –help
tex –help
dos2unix –help (Una de las Rtools)

Para cada uno de estos comandos deberías de ver una lista de opciones. Si alguno de estos programas fallan verificar cuidadosamente la variable PATH examinando donde están localizados los siguientes archivos: gcc.exe, perl.exe,Tex.exe y dos2unix.exe.

El paquete R provee un conjunto de herramientas que nos ayudaran en la administración de paquetes. Algunas de estas herramientas se detallan a continuación:

INSTALL instala el paquete especificado

REMOVE remueve (desinstala) el paquete especificado
BUILD Construye el paquete especificado
CHECK Verifica el paquete especificado
R2txt Convierte formato Rd a texto
Rdconv Convierte formato Rd a otros formatos

Estas herramientas son invocadas por R desde la consola de MS-DOS o desde la consola de Linux. La sintaxis para ejecutar estas herramientas es R CMD nombreherramienta opciones. Las dos herramientas build y check son quizás las más importantes. La primera construye el paquete, es decir transforma el contenido del paquete a un formato que puede ser distribuido. La segunda verifica el paquete. El proceso de verificación se concentra en proveer a los desarrolladores de paquetes con información esencial respecto a cualquier problema que se pueda dar en un paquete. Esta herramienta debería ser ejecutada después de realizar todas las modificaciones al paquete. Las herramientas R2txt y Rdconv proveen opciones de conversión para el sistema de documentación de funciones, el cual se comentara en la siguiente sección.

B.3. DESARROLLO DE UN PAQUETE EN R

El procedimiento para desarrollar un paquete en R esta bien documentado en el manual "Writing R Extensions"[1] (WRE) del sitio web del R. Sin embargo, este material esta diseñado como una referencia y incluye muchas características que por lo general no son necesarias al construir un paquete en R. Por esta razón hemos tomado lo principal de este procedimiento porque la idea es proveer al lector de suficiente información como para colocar la construcción de un paquete en R en un contexto que tenga significado, sin abrumarlo al mismo tiempo con muchos detalles. El procedimiento para la construcción de un paquete en R se detalla en los siguientes pasos:

[1] `http://cran.r-project.org/doc/manuals/R-exts.html`

1. **Análisis previo del problema**

   Antes de proceder a crear el paquete debemos definir claramente el problema a resolver para el caso que nos hemos planteado consiste en crear un paquete en R que calcule el número de fibonacci el cual se obtiene mediante la función recursiva dada por la ecuación (B.1).

$$f(n) = \begin{cases} 1 & \text{si } n = 1 \\ 1 & \text{si } n = 2 \\ f(n-2) + f(n-1) & \text{si } n > 1 \end{cases} \tag{B.1}$$

   O sea, que los dos primeros números de la sucesión son 1, y luego cada número es la suma de los dos anteriores. Una vez que hemos planteado el problema debemos determinar cuantas funciones contendrá el paquete y especificar sus entradas y salidas. Para el problema que se esta tratando tendremos una sola función cuya entrada y salida será un número entero que se corresponden respectivamente con el valor de n y $f(n)$.

2. **Crear la estructura del paquete**
   Un paquete en R consiste de un directorio con una serie de subdirectorios y archivos, a continuación utilizando el Explorador de Windows crearemos una carpeta que se llame fibonacci la cual tendrá lo siguiente:

   Un archivo cuyo nombre debe ser DESCRIPTION. Dos subdirectorios con los nombres siguientes:

   - man <<< contendrá la documentación
   - R <<< contendrá las funciones en R

   Debemos mencionar que los directorios man y R son obligatorios. Entre los directorios que son opcionales se encuentran:

   - data <<< contendrá archivos que contienen datos (para cargar con la función load)
   - src <<< contendrá funciones escritas en lenguaje C o Fortran
   - demo <<< contendrá scripts en R (para correr con la función demo)

- exec <<< contendrá archivos ejecutables que el paquete necesita (experimental)

Dentro de la carpeta R incluiremos un archivo especial zzz.R este archivo imprimirá un mensaje que nos dirá que el paquete ha sido cargado. El contenido de este archivo es la instrucción:

cat("Paquete fibonacci cargado \n")

Otra forma para crear esta estructura es usar la función package.skeleton, ver el archivo de ayuda del R (con el comando help) para obtener más información. Otra alternativa es instalar el paquete RpackGen que se puede obtener en la dirección web:

`http://www.biostat.harvard.edu/$\sim$carey/index2.html`

3. **Editar el archivo DESCRIPTION**
El archivo DESCRIPTION contiene información básica acerca del paquete. Este archivo contiene información como 'Package', 'Version', 'License', 'Description', 'Title', 'Author' y 'Maintainer' las cuales corresponden al nombre del paquete, numero de versión, tipo de licencia, descripción del paquete, titulo, autor y la persona responsable del mantenimiento del programa. Además de esta información se puede incluir otros campos, para más información ver WRE sección 1.1.1. Un ejemplo de archivo DESCRIPTION se muestra a continuación, la notación <> significa texto que se debe sustituir por un valor apropiado.

```
Package: < nombre del paquete>
Version: < numero con formato n.n { inicia con 0.0>
Title: < corto titulo, debe ser tan descriptivo como sea
posible>
Author: < tu nombre, seguido por tu dirección de email
entre brackets >
Maintainer: < el nombre de la persona que le da mantenimiento
al programa>
Description: < entre 5 y 10 lineas que describan el paquete>
License: < el tipo de licencia del paquete,
ver WRE sección 1.1.1 >
```

El contenido del archivo DESCRIPTION para el paquete fibonacci es el siguiente:

```
Package: fibonacci
Type: Package
Title: Fibonacci
Version: 1.0
Date: 2006-12-21
Author: Ricardo Salvador Ríos Márquez
Maintainer: Ricardo Salvador Ríos Márquez <ricrio@gmail.com>
Description: Paquete para calcular los números de fibonacci.
License: GPL
```

4. **Escritura de las funciones en R**
   Como dijimos anteriormente el paquete fibonacci consistirá de una función que calcula el número de fibonacci y cuyo nombre es fib, el código de esta función es el siguiente:

```
fib <- function(n = 1)
{
if (n < 2)
fn <- 1
else
fn <- fib(n - 1) + fib(n - 2)
return(fn)
}
```

5. **Crear la documentación**
   Este es uno de los pasos más difíciles, pero es muy importante. Si la documentación es mala, nadie va usar el paquete.

   Para cada función en el directorio R, se debe crear el correspondiente archivo de documentación (con extensión rd), en el directorio man. Esto significa, si tenemos fun.r en el directorio R, se debe crear el archivo fun.rd en el directorio man.

   R usa un lenguaje de marcado dentro de los archivos rd. La utilidad de instalación de paquetes traslada los archivos rd a formato html, texto plano, y LaTeX. El lenguaje de marcado que utiliza R para la documentación se parece a LaTeX.

   Creando una plantilla
   La función de R prompt facilita la construcción de la documentación,

esta función creará una plantilla para una función que este cargada en el actual espacio de trabajo. Para utilizar la función prompt se debe primero cargar la función a documentar con la función source, por ejemplo, source(fun.r) y usar el comando prompt(fun, file=fun.rd) este creará el archivo de plantilla con el nombre fun.rd en el directorio actual de trabajo, el cual se debe mover al directorio man del paquete y editarlo con respecto a las características de la función.

Formato de archivos rd
Los archivos rd están en un lenguaje de marcado consistente de una serie de comandos seguidos por sus argumentos. Cada comando es de la forma:

```
\comando{argumento}
```

Por ejemplo,

```
\title{Este es mi título}
```

Algunos de estos comandos o partes son obligatorios, la mayoría son opcionales. El archivo de plantilla que crea la función prompt contiene los comandos obligatorios. Para obtener más información de estos comandos ver WRE sección 2.

Depurando los archivos de documentación
Los mensajes de error generados por R para los archivos de documentación pueden ser difíciles de interpretar. Por esta razón, se debe tener cuidado cuando se escribe la documentación. Existen dos ideas para resolver esto:

1) Crear el archivo rd gradualmente. Esto significa que se debe crear paulatinamente cada archivo y depurarlo continuamente.
2) Trabajar en un archivo de documentación a la vez.

Es una mala idea usar la utilidad "R CMD check" para verificar desde el principio los archivos de documentación, es mejor usar primero el comando "R CMD Rd2txt". Si por ejemplo tengo el archivo de documentación fun.rd, usar el siguiente comando en el símbolo de sistema de Windows.

```
R CMD Rd2txt fun.rd
```

El archivo será traducido y desplegado en la consola. Si esto no funciona, se debe usar el comando:

```
R CMD Rdconv -t=html -o=fun.html fun.rd
```

Esto produce una versión html del archivo fun.rd, luego usar un navegador para desplegar este archivo. Inspeccionar la salida de estos comandos minuciosamente y observar si las secciones del archivo rd que se han definido son mostradas.

El contenido del archivo de documentación para la función fib del paquete fibonacci es el siguiente:

```
\name{fib}
\alias{fib}
\title{Cálculo del número de fibonacci}
\description{Calcula el número de fibonacci de un entero. Esta función
calcula el número de fibonacci recursivamente.
}
\usage{fib(n)}
\arguments{
\item{n}{Un valor entero para el cual se calculará el número de
fibonacci}}
\details{
Retorna el número de fibonacci de un número n, El cual esta definido de la
siguiente manera:\cr
fib(1) = 1\cr
fib(2) = 1\cr
fib(n) = fib(n-1) + fib(n-2)\cr
Si el valor de n es menor que 1 entonces fib retorna 1\cr
}
\value{Retorna un valor entero.}
\references{ \url{www.geocities.com/ricardo_rios_sv/paquete_fibonacci} }
\author{ Ricardo Salvador Ríos Márquez <ricrio@gmail.com> }
\examples{
fib(7)
```

```
apply(as.array(1:7), 1, fib)
}
\keyword{ arith }
```

6. **Verificar el paquete**
   En teoría, verificar un paquete es sencillo. Se debe abrir la consola de MS-DOS de Windows y dirigirse al directorio que contiene el paquete, una vez ahí ejecutamos la siguiente instrucción:

   ```
   R CMD CHECK <nombre-del-paquete>
   ```

   Por ejemplo, R CMD CHECK correrá un script en perl, el cual creará los archivos de documentación en formato texto, LaTeX y formato html, verificará inconsistencias y varios errores, correrá ejemplos (los cuales se especifican con el comando "examples" en los archivos rd), y finalmente construirá un manual en formato dvi.

   R CMD CHECK construirá el paquete en un directorio llamado <package>.Rdcheck el cual será creado en el mismo nivel de directorio en el cual R CMD fue invocado.Finalmente, debemos asegurarnos que el tiempo para correr los ejemplos es menor que 2 minutos. Esto puede ser verificado observando el archivo <package>-Ex.Rout que se encuentra dentro del directorio Rdcheck.

7. **Construir el paquete**
   Para construir el paquete usaremos el comando R CMD BUILD <package>. Este crea un archivo en formato tar comprimido, el cual puede servir para su posterior instalación con el comando R CMD INSTALL y constituye uno de los métodos para distribuir paquetes a otros usuarios. Por ejemplo, las siguientes instrucciones que son ejecutadas en la consola de MSDOS de Windows y dentro del directorio donde se encuentra el paquete fibonacci instalan este paquete:

   ```
   R CMD BUILD fibonacci
   R CMD INSTALL fibonacci_1.0.tar.gz
   ```

   Usar R CMD BUILD –help y R CMD INSTALL –help para obtener más información acerca de cómo usar el constructor y el instalador de paquetes.

Si lo que queremos es construir una versión precompilada del paquete, para instalar desde el menú "Instalar paquete(s) a partir de archivos zip locales ..." de la versión de R en Windows, debemos primero crear un directorio que se llame por ejemplo localRlib en el directorio donde se encuentra el paquete y después ejecutar los siguientes comandos desde la consola de Windows:

```
R CMD BUILD fibonacci
R CMD INSTALL -l localRlib fibonacci_1.0.tar.gz
cd localRlib
zip -r fibonacci_1.0.zip fibonacci
```

Estos pasos crean el archivo fibonacci_1.0.zip dentro de la carpeta localRlib, el cual se puede instalar desde la interfaz grafica de usuario que proporciona R en Windows, este mecanismo de instalación de paquetes constituye un excelente mecanismo para distribuir el paquete a otros usuarios. Todo los pasos comentados anteriormente son validos también en Linux. Para obtener más detalles sobre el proceso de construcción de un paquete ver WRE.

Anexo C

EL PAQUETE NEUROGEN

C.1. INTRODUCCIÓN

Neurogen es una aplicación informática desarrollada en el entorno estadístico R el cual representa el estado del arte en computación estadística de propósito general. Esta aplicación informática permite estimar un modelo neuronal MLP con variable respuesta dicotómica, para lograr esto hace uso de un algoritmo genético elitista, el cual busca el mejor vector de pesos o coeficientes de la red.

Esta aplicación fue desarrollada para proveer al usuario de un mayor control sobre el proceso de estimación de una red neuronal, la mayoría del software comercial en redes neuronales no proporcionan los pesos de la red y no indican cual es el modelo que utilizan, por lo que son poco aconsejables para el presente estudio.

Al optar implementar neurogen en R se obtiene una gran facilidad de desarrollo ya que este entorno posee una vasta biblioteca de funciones, sin embargo, tiene el inconveniente de que al ser código interpretado más lento que el código compilado, se pierde velocidad. Siendo los algoritmos genéticos métodos de optimización que convergen lentamente se decidió desarrollar neurogen utilizando la interfaz de programación al lenguaje C que tiene el

R, logrando con ello la generación de código nativo y sobre todo altamente eficiente y veloz.

El sistema operativo para el que se diseño neurogen es Linux, sin embargo, dado que R es independiente de la plataforma se pudo lograr su portación a Windows, el cual es el sistema operativo más ampliamente usado por la comunidad universitaria.

C.2. POR QUÉ DESARROLLAR NEUROGEN?

En el proceso de desarrollo de la tesis se hizo una búsqueda del software disponible sobre redes neuronales que mejor se adaptara a los propósitos de la investigación. En el transcurso de esta búsqueda se observo que la mayoría del software que implementa el paradigma de redes neuronales con algoritmos genéticos son comerciales, lo que ocasiona que no sean muy accesibles y proporcionan poca flexibilidad, también no proporcionan explícitamente los parámetros del modelo, solo se limitan a usar la validación cruzada y no implementan otros métodos de validación como las técnicas de remuestreo, al estar implementados estos programas en Windows no se obtienen los beneficios de otras plataformas más robustas como los sistemas Linux y Unix, además de otras razones y por lo tanto no son adecuados para propósitos como los planteados en los objetivos de la presente investigación.

Por lo anteriormente mencionado se decidió desarrollar un paquete en R sobre redes neuronales con una buena base teórica y enfocado a resolver problemas de clasificación binaria, el cual se pondrá a disposición de investigadores, docentes y estudiantes con la esperanza que con un mínimo conocimiento de la herramienta estadística R puedan sacarle el máximo provecho para resolver los problemas que ellos enfrentan, así como para difundir y desarrollar estas nuevas metodologías.

C.3. CARACTERISTICAS

C.3.1. FUNCIONES QUE COMPONEN NEUROGEN

El paquete neurogen esta constituido por una serie de funciones cada una de las cuales realiza una tarea especifica con el objetivo de estimar un modelo de redes con variable respuesta dicotómica.

Son 4 las funciones que componen a neurogen, las cuales se muestran en la tabla (C.1) .

El código fuente de neurogen se puede descargar en la siguiente dirección web `http://ricardorios.net/neurogen/neurogen.zip`.

C.3.2. DISEÑO PROCEDIMENTAL

Como se menciona en el apartado anterior neurogen esta formado por una serie de funciones las cuales se llaman unas a otras, esta forma de diseñar programas caracteriza al paradigma de programación procedimental. El entorno de programación R maneja este paradigma. La programación procedimental se plantea un problema como un todo y se divide en segmentos más sencillos o de menor complejidad. Una vez terminados estos segmentos, se procede a unificarlos. La programación procedimental constituye el paradigma de programación más usado a la hora de construir paquetes en R, esto se debe en parte a su sencillez y facilidad de aprendizaje; por lo que constituye una excelente elección a la hora de implementar un paquete en R.

C.3.3. PORTABILIDAD

"La portabilidad de un software es cuan dependiente es del sistema operativo en el que corre. A mayor portabilidad mayor independencia con el

Tabla C.1: Funciones que componen neurogen

Función	Descripción
classnetfun	Función utilitaria que a partir de un vector de parámetros calcula los siguientes valores: el valor de la función de error (entropía), la respuesta de la red neuronal, las derivadas parciales (derivadas analíticas) de la red neuronal y el error de clasificación.
gen	Función utilitaria usada por la función classnet y que busca el mejor vector de coeficientes de la red neuronal utilizando un algoritmo genético elitista que emplea los operadores de selección, cruza y mutación. Esta función internamente llama a una rutina compilada en lenguaje C, la cual realiza todo el cálculo.
classnet	Función que permite estimar una red neuronal con un numero de capas fijo y que busca clasificar datos donde la variable respuesta es dicotómica. A esta función se le proporcionan ciertos parámetros relacionados al modelo que se desea ajustar y a partir de estos devuelve una lista conteniendo los siguientes valores: porcentaje de falsos positivos y negativos dentro y fuera de la muestra, el valor de la función de error (entropía), la respuesta de la red dentro y fuera de la muestra, el conjunto de coeficientes de mejor ajuste, el numero de capas de la red neuronal y el tipo de preprocesamiento.
predictclassnet	Función usada para calcular la respuesta que da un modelo neuronal previamente estimado con la función classnet, sobre un conjunto nuevo de observaciones.

sistema operativo."[1] Dado que R se ejecuta en los sistemas operativos Linux, Windows y Mac supondríamos que el paquete neurogen podría correr en estas plataformas, sin embargo, dado que nuestro paquete usa código compilado esta portabilidad depende también de la posibilidad de tener un compilador adecuado para cada uno de estos sistemas operativos. En Linux y Windows se logro crear el paquete, ya que para cada uno de los sistemas operativos existe una versión del compilador gcc el cual nos permitió crear el código

[1] `http://es.wikipedia.org/wiki/Portabilidad`

compilado, sin embargo, se desconoce si puede correr en el sistema operativo Mac ya que este sistema operativo no es muy difundido y por tal razón no se tuvo acceso al mismo.

C.4. MANUAL DE USUARIO DE LA APLICACIÓN

El manual de usuario, es un manual acerca de cómo trabajar con neurogen. Este manual de usuario comprende temas como los requerimientos para su instalación, la instalación de la aplicación y un ejemplo de cómo utilizar el paquete.

C.4.1. REQUERIMIENTOS

Antes de instalar neurogen, asegúrese de que el equipo tenga instalado el software R versión 2.0.1 o superior. Este software puede ser descargado en la siguiente dirección web `http://www.r-project.org/`.

C.4.2. INSTALACIÓN

Empleando un navegador descargar neurogen, desde la siguiente dirección web dependiendo de el sistema operativo:

- Para Windows: `http://ricardorios.net/neurogen_1.0.zip`
- Para Linux: `http://ricardorios.net/neurogen_1.0.tar.gz`

Esto guarda el archivo neurogen_1.0.zip o neurogen_1.0.tar.gz dependiendo de que se haya elegido descargar la versión para Windows o Linux.

Una vez se haya descargado, procederemos a instalarlo:

- En Windows
  - Ejecutar el programa Rgui que se encuentra en el escritorio y usar la opción "Instalar paquete(s) a partir de archivos zip locales ..." del menú paquetes.
  - Seleccionar el archivo neurogen_1.0.zip que anteriormente se ha descargado y clic en Abrir.
- En Linux
  - Loguearse con el usuario root o cualquier usuario con privilegios para escribir en el sistema de directorios.
  - Desde la terminal entrar al directorio donde hayas descargado el archivo neurogen_1.0.tar.gz y ejecutar el comando "R CMD INSTALL neurogen_1.0.tar.gz".

C.4.3. UN EJEMPLO DE CLASIFICACION SUPERVISADA USANDO NEUROGEN

Esta sección describe una típica aplicación de neurogen a un problema de identificación de cancer de piel. El objetivo de este análisis es estimar un modelo de redes neuronales que determine si un paciente tiene cancer sin necesidad de realizarle la biopsia. Debemos mencionar que esta no es una aplicación médica real y que solo se hace con propósitos ilustrativos.

DATOS

Los datos usados en este análisis son tomados del software NeuroShell el cual es un software comercial para pronostico basado en redes neuronales. Estos datos están constituidos por 3 variables explicativas y 1 variable respuesta, las cuales se detallan a continuación:

- Tipo de piel: Valor entero entre 1 y 3.
- Examen físico: Valor numérico entre 0 y 1 que representa un valor de probabilidad de que el paciente tenga cancer de piel. Un examen físico asigna el valor a cada paciente.

- Prueba de Sangre: Valor entero entre 0 y 10 que representa una prueba de sangre en la cual se observa la presencia de células cancerosas.
- Biopsia (Variable respuesta) : El diagnostico maligno = 1 benigno = 0.

El total de observaciones son 50 (36 observaciones con diagnostico maligno y 14 observaciones con diagnostico benigno).

CONFIGURACION DEL ANÁLISIS

Como primer paso en el análisis debemos realizar la carga del paquete neurogen en el entorno de programación estadística R. Esto lo conseguimos digitando en R la siguiente instrucción:

```
library(neurogen)
```

En el paquete neurogen se ha incluido los datos de cancer de piel, que se mencionaron en la sección anterior. Para tener acceso a estos datos debemos digitar la siguiente instrucción:

```
load(cancer)
```

Esta instrucción crea una variable tipo arreglo con el nombre cancer. Una vez que hemos cargado los datos que nos servirán para entrenar la red neuronal, debemos de pasar al siguiente paso entrenar la red neuronal.

ENTRENAMIENTO DE LA RED NEURONAL

Una vez entrenada la red neuronal con la función classnet y almacenado su valor de retorno en una variable, podemos con esta variable realizar predicciones con valores que no han sido previamente mostrados durante el entrenamiento de la red neuronal. Para realizar esto usaremos la función predictclassnet cuya lista de parámetros en el orden en que deberían ser escritos esta especificado en la tabla (C.2).

Tabla C.2: Argumentos o parámetros de la función classnet

Argumento	Descripción
assetx	Un arreglo que contiene una observación en cada fila y las variables en cada columna.
coldep	El numero de la columna correspondiente a la variable dependiente en assetx.
percent	Porcentaje de datos para la estimación dentro de la muestra (si es 1 se toman todas las observaciones en el proceso de entrenamiento).
errpercent	El menor valor del error de clasificación permitido (si se encuentra un error de clasificación menor a errpercent, la funcion classnet retorna).
errweight	Una ponderación que se le da a los falsos positivos (si ha errweight se le asigna 0.5 significa que se le da la misma ponderación a los falsos positivos y negativos).
limit	Punto de corte.
info	Un arreglo numérico con longitud igual al numero de capas de la red y conteniendo la cantidad de neuronas en cada una de las capas de la red.
gendum	Un numero entero conteniendo el tipo de optimización (1 = Algoritmos genéticos).
maxgen	Máximo numero de generaciones.
helge	El numero entero que nos indica el tipo de preprocesamiento (0 = sin preprocesamiento, 1 = escalamiento en el intervalo $[0, 1]$ y 2 = normalización).
visual	Un valor lógico que activa o desactiva la visualización del proceso de entrenamiento.
beta0init	Pesos o coeficientes de la red (opcional).

Las siguientes instrucciones nos permiten estimar un modelo neuronal para los datos de cancer de piel.

```
layers <- array(data=c(ncol(cancer)-1,3,1),dim=c(1,3))
net <- classnet(cancer,ncol(cancer),1,0.05,0.5,0.5,layers,1,100,2,TRUE)
```

La primera instrucción define un arreglo numérico con la arquitectura de la red neuronal para este caso particular la arquitectura es 3-3-1, es decir 3 neuronas en la capa de entrada, 3 neuronas en la capa oculta y 1 neurona en la capa de salida. La segunda instrucción asigna a la variable net el valor de retorno de la función classnet la cual es llamada con una serie de parametros. El valor que contendrá net será de un objeto tipo lista conteniendo los siguientes siete elementos:

1. NETRES: El porcentaje de falsos positivos y falsos negativos dentro y fuera de la muestra.
2. LIKELIHOOD: La función de error (entropía).
3. PROB_IN: La respuesta de la red dentro de la muestra.
4. PROB_OUT: La respuesta de la red fuera de la muestra.
5. beta: El mejor conjunto de pesos o coeficientes de la red neuronal.
6. info: La arquitectura de la red neuronal.
7. helge: Información del preprocesamiento.

PREDICCION DE NUEVAS OBSERVACIONES

Una vez entrenada la red neuronal con la función classnet y almacenado su valor de retorno en una variable, podemos con esta variable realizar predicciones con valores que no han sido previamente mostrados durante el entrenamiento de la red neuronal. Para realizar esto usaremos la función predictclassnet cuya lista de parámetros en el orden en que deberían ser escritos se especifica en la tabla (C.3).

Con las siguientes instrucciones ejemplificamos el uso de la función predictclassnet.

```
data <- array(data=c(2,0.5,2), dim=c(1,3))
predictclassnet(data, net$info, net$helge, net$beta)
```

Tabla C.3: Argumentos o parámetros de la función predictclassnet

Argumento	Descripción
assetx	Un arreglo numérico que contiene una observación en cada fila y solo las variable de entrada o explicativas en las columnas.
info	Un arreglo numérico con longitud igual al numero de capas de la red y conteniendo la cantidad de neuronas en cada una de las capas de la red.
helge	El numero entero que nos indica el tipo de preprocesamiento (0 = sin preprocesamiento, 1 = escalamiento en el intervalo $[0, 1]$ y 2 = normalización).
beta	Pesos o coeficientes de la red.

Con la primera instrucción creamos un arreglo numérico conteniendo una observación que se corresponde a las variables explicativas del problema de cáncer de piel anteriormente mencionado. La segunda instrucción nos da la respuesta de la red neuronal previamente estimada en el paso anterior ante esta nueva observación. Para un ejemplo de la salida que proporciona la ejecución de los comandos anteriormente mostrados ver anexo. Para ver la ayuda del paquete neurogen se debe ejecutar el comando help(neurogen).

Anexo D

BASE DE DATOS EN POSTGRESQL

Tabla D.1: Base de datos en PostgreSQL que maneja la información de la susceptibilidad a deslizamiento de tierra en el AMSS

Tabla	datos		
Campo	Tipo	Descripción	Atributos
id	integer	Correlativo	Llave primaria
y	smallint	Ocurrencia a deslizamiento	
x1	smallint	Geomorfología	
x2	smallint	Geología	
x3	double precision	Precipitaciones máximas	
x4	double precision	Aceleraciones sismicas	
x5	double precision	Pendiente del terreno	
x6	double precision	Distancia a carretera	
x7	double precision	Distancia a falla geológica	

Anexo E

SCRIPT EN R PARA GENERAR LOS CONJUNTOS DE ENTRENAMIENTO, VALIDACIÓN Y TEST

```
library(RPgSQL)

db.connect(host='localhost', port='5432', dbname='datos', user='ricrio', password='xxxx')
db.execute('select id from datos where y = 0', clear=F)
datos0 <- db.fetch.result()
db.clear.result()
db.execute('select id from datos where y = 1', clear=F)
datos1 <- db.fetch.result()
db.clear.result()

temp1 <- sample(datos1$id, 2876)
temp0 <- sample(datos0$id, 2876)

db.write.table(temp1, name='id_1_estimacion', no.clobber=T)
db.write.table(temp0, name='id_0_estimacion', no.clobber=T)


db.execute('select id from datos where y = 0
except select data from id_0_estimacion', clear=F)
datos0 <- db.fetch.result()
```

```
db.clear.result()
db.execute('select id from datos where y = 1
except select data from id_1_estimacion', clear=F)
datos1 <- db.fetch.result()
db.clear.result()


temp1 <- sample(datos1$id, 958)
temp0 <- sample(datos0$id, 958)

db.write.table(temp1, name='id_1_validacion', no.clobber=T)
db.write.table(temp0, name='id_0_validacion', no.clobber=T)

db.execute('select id from datos where y = 0
except (select data from id_0_estimacion
union select data from id_0_validacion)', clear=F)
datos0 <- db.fetch.result()
db.clear.result()
db.execute('select id from datos where y = 1
except (select data from id_1_estimacion
union select data from id_1_validacion)', clear=F)
datos1 <- db.fetch.result()
db.clear.result()

temp1 <- sample(datos1$id, 958)
temp0 <- sample(datos0$id, 958)

db.write.table(temp1, name='id_1_test', no.clobber=T)
db.write.table(temp0, name='id_0_test', no.clobber=T)

rm(list=ls())
```

Anexo F

SCRIPT EN R PARA CRUZAR LA INFORMACIÓN DE LAS COORDENADAS GEOGRÁFICAS CON LA PROBABILIDAD A OCURRENCIA A DESLIZAMIENTO DE TIERRA

```
library(RPgSQL)

db.connect(host='localhost', port='5432', dbname='datos', user='ricrio', password='xxxx')

db.execute('SELECT y, x1, x2, x3, x4, x5, x6, x7 from data_estimacion ', clear=F)

datos <- db.fetch.result()
```

```
db.clear.result()


media1 <- mean(datos$x1)
sd1 <- sd(datos$x1)


media2 <- mean(datos$x2)
sd2 <- sd(datos$x2)


media3 <- mean(datos$x3)
sd3 <- sd(datos$x3)


media4 <- mean(datos$x4)
sd4 <- sd(datos$x4)

media5 <- mean(datos$x5)
sd5 <- sd(datos$x5)


media6 <- mean(datos$x6)
sd6 <- sd(datos$x6)


media7 <- mean(datos$x7)
sd7 <- sd(datos$x7)


rm(datos)



db.execute('SELECT id, y, x1, x2, x3, x4, x5, x6, x7, xcoor, ycoor from data_sin_deterministicos ', clear=F)

datos <- db.fetch.result()

db.clear.result()



datos$x1 = (datos$x1 -  media1 ) / sd1

datos$x2 = (datos$x2 - media2 ) / sd2

datos$x3 = (datos$x3 - media3 ) / sd3

datos$x4 = (datos$x4 - media4 ) / sd4

datos$x5 = (datos$x5 - media5 ) / sd5

datos$x6 = (datos$x6 - media6 ) / sd6

datos$x7 = (datos$x7 - media7 ) / sd7
```

```
input <- cbind(datos$x1, datos$x2, datos$x3, datos$x4, datos$x5, datos$x6, datos$x7)

input <- as.array(input)

output <- datos$y

info <- array(data=c(ncol(input),17,1), dim=c(1,3))

helge <- list( helge= 0 )


betitas <- read.csv(file='betas.csv', header=FALSE)

betitas2 <- as.array(betitas$V1)


output_estimado <- predictclassnet(input, info, helge, betitas2)

output_estimado2 <-  ifelse(output_estimado$PROB>0.5,1,0)

ftable(output, output_estimado2)

prob <- output_estimado$PROB

length(prob) <- nrow(datos)


datos$probabilidad <- prob



db.execute('SELECT id, y, x1, x2, x3, x4, x5, x6, x7, xcoor, ycoor from deterministicos ', clear=F)

datos2 <- db.fetch.result()

db.clear.result()

prob <- seq(0,0 ,length=nrow(datos2))

length(prob) <- nrow(datos2)

datos2$probabilidad <- prob
```

Anexo G

MAPA DE SUSCEPTIBILIDAD A DESLIZAMIENTOS DE TIERRA GENERADO CON ILWIS

Mapa de Susceptibilidad a Deslizamientos usando Redes Neuronales