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

TRABAJO DE GRADUACIÓN:

"REDES NEURONALES APLICADAS A LA DEMANDA INTERMITENTE DEL NÚMERO DE PARTE QA065423-01 PARA EL TIPO DE AVION A320 EN TACA".

PRESENTADO POR:

BR. RICARDO SALVADOR RÍOS MÁRQUEZ

PARA OPTAR AL GRADO DE:

LICENCIADO EN ESTADÍSTICA

ASESOR:

MSc. ROLANDO LEMUS GÓMEZ

ASESOR ADJUNTO:

MSc. OSCAR HERNAN LEMUS GÓMEZ

CIUDAD UNIVERSITARIA, OCTUBRE DE 2007

UNIVERSIDAD DE EL SALVADOR

RECTORA :

Dra. María Isabel Rodríguez

SECRETARÍA GENERAL :

Licda. Alicia Margarita Rivas de Recinos

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA

DECANO EN FUNCIONES :

MSc. José Héctor Elías Díaz

SECRETARÍA :

Licda. Martha Noemí Martínez de Rosales

ESCUELA MATEMÁTICA

DIRECTOR :

Lic. Mauricio Hernán Lovo Córdova

TRABAJO DE GRADUACIÓN APROBADO POR:

COORDINADOR :
Lic. Mauricio Hernán Lovo Córdova

ASESOR :
MSc. Rolando Lemus Gómez

ASESOR ADJUNTO :
MSc. Oscar Hernán Lemus Gómez

AGRADECIMIENTOS

Doy gracias a Dios todo poderoso que me ha permitido cumplir satisfactoriamente todos los requisitos que exige la carrera, dándome sabiduría cuando más la necesitaba. Por otra parte agradecer a mi familia, mi padre, Angel Salvador; mi madre Santa Elizabeth y mis dos hermanos Carlos y Miguel que me dieron su apoyo sin restricciones y sobre todo mucho animo. Agradezco de antemano a mis asesores por darme la confianza para desarrollar este proyecto y a todas las personas que me ayudaron en los foros de internet.

Ricardo Salvador Ríos Márquez.

INDICE DE CONTENIDOS

RESUMEN

El objeto de estudio del presente documento es el desarrollo de un modelo de redes neuronales que permita evaluar la relación entre la demanda del número de parte QA065423-01, la información de uso del avión, datos referentes al último cambio de pieza y la experiencia del mecánico, el cual se desarrollo en cuatro capítulos.

El primero comprende el planteamiento del problema, los antecedentes y justificación, los objetivos y la metodología.

En el segundo capítulo se expone el fundamento teórico de la investigación empezando con la teoría de redes neuronales enfocada a resolver problemas estadísticos de clasificación binaria. Luego se expone el modelo de suavizado exponencial el cual es uno de los métodos más utilizados para pronosticar la demanda intermitente en los inventarios.

En el tercer capítulo se presentan las etapas previas que se tuvieron que realizar para la aplicación del modelo de redes neuronales al problema de la demanda intermitente del número de parte QA065423-01.

El capítulo cuatro describe la aplicación de la teoría de redes neuronales al problema de la demanda intermitente del número de parte QA065423-01, se tomaron los datos que van de febrero de 1999 a diciembre del 2006 para estimar y validar un modelo de redes neuronales para la demanda del número de parte QA065423-01, es decir determinar si se cambia o no el número de parte QA065423-01. Una vez estimado el modelo de redes neuronales se procedió a implementar una herramienta informática que permitiera pronosticar la demanda total del número de parte QA065423-01, para probar esta herramienta se utilizaron las observaciones que van de enero a febrero del 2007, los resultados que se obtuvieron fueron satisfactorios con un error de pronóstico del 8.33 por ciento.

En el último capítulo se presentan algunas conclusiones y recomendaciones de lo aprendido sobre las redes neuronales y principalmente de su aplicación al problema del pronóstico de la demanda intermitente del número de parte QA065423-01.

INTRODUCCIÓN

La empresa TACA se enfrenta a un ambiente adverso de competencia y de altos precios del petróleo, para mantenerse en el mercado de las aerolíneas, cada vez, más competitivo, por eso es necesario que incremente de forma permanente su productividad. Dentro de las muchas actividades que la empresa TACA podría desarrollar para mejorar su productividad se encuentra, la optimización en la gestión de los inventarios de las piezas de aviones, estos inventarios son indispensables en el proceso de mantenimiento de aviones que se realiza en AEROMAN, la cual es una base de mantenimiento de aviones que esta entre los primeros 11 talleres aeronáuticos que forman parte de la red de talleres reparadores a nivel mundial de AIRBUS, la compañía fabricante de estos aviones.

La idea entonces es lograr satisfacer la demanda de piezas de aviones, con óptimos niveles de servicio, pero con niveles de inventario que no generen sobrecostos en su operación. Dentro de las estrategias para lograr tomar estas decisiones en los inventarios es hacer uso de modelos estadísticos de pronósticos como los modelos de suavizado exponencial, el problema con este enfoque es que la demanda que presentan estos números de partes o piezas es intermitente, es decir, tiene bajo consumo, presenta una gran cantidad de ceros y valores diferentes de cero dispuestos en forma aleatoria, lo cual dificulta el pronóstico ya que los modelos estadísticos ignoran el rol especial de los valores cero, además, de otras características claves de los datos.

El tema de la demanda intermitente ha sido poco investigado, por lo que el estudio de métodos alternativos que puedan proporcionar mejores pronósticos es de gran importancia.

Los modelos conocidos como redes neuronales han representado un nuevo paradigma de solución de problemas, este nuevo enfoque es inspirado en las redes neuronales biológicas, las redes neuronales han mostrado mayor efectividad para resolver problemas que los enfoques tradicionales no han solucionado o lo han hecho de manera parcial. Las redes neuronales permiten resolver problemas que no se comprenden del todo, gracias a que son capaces de aprender, lo cual les permite encontrar una solución al problema mediante

ejemplos. Otra característica de las redes neuronales es su capacidad de descubrir relaciones no lineales en los datos, las cuales son muy difíciles de identificar por parte de los expertos.

En la presente investigación se pretende desarrollar un modelo de redes neuronales que permita prever la demanda intermitente para un número de parte o pieza de avión, los resultados que obtengamos permitirán concluir si es factible o no utilizar estos modelos al problema de la demanda intermitente en el número de parte elegido. La razón de elegir estos modelos es que conjeturamos que al no ser incorporadas las relaciones no-lineales al pronóstico por parte de los modelos tradicionales, la posible inclusión de estas al pronóstico podría generar previsiones de la demanda más certeras.

Los resultados que se obtengan en la presente investigación, serán muy beneficiosos, tanto en forma práctica y metodológica, ya que se intenta solucionar un problema real cuyo abordaje por las técnicas tradicionales proporciona pobres resultados y, además, se propone una nueva metodología de solución de problemas que ha sido poco abordada en nuestra Universidad.

Utilizando el software MATLAB (Matrix Laboratory) se simulará el modelo neuronal propuesto y se analizará la información. La herramienta MATLAB es ampliamente usada en el desarrollo de modelos neuronales, además, de ser un entorno que proporciona módulos predefinidos de redes neuronales.

CAPITULO I: INVESTIGACION PRELIMINAR

1.1 INTRODUCCION

En la etapa preliminar, se planteó el problema de investigación, se investigaron los antecedentes y la justificación del problema en estudio, se determinó, la metodología, el tiempo, los recursos a utilizar y la institución que nos proporcionaría la base de datos. Se definieron los objetivos a alcanzar, finalizando con la presentación del documento que constituye el anteproyecto.

1.2 PLANTEAMIENTO DEL PROBLEMA

En un mundo caracterizado en la actualidad por un alza en los precios del petróleo, el cual impacta de manera significativa en las aerolíneas alrededor del mundo, esta crisis conlleva a que las empresas de este sector realicen profundos cambios que les permitan enfrentar con mejor pie este ambiente adverso. En este sentido, TACA no escapa a realizar ajustes en su manera de operar, que ciertamente contribuirán a mantener su lugar de excelencia dentro de las aerolíneas internacionales.

Un elemento clave para reducir los costos en las aerolíneas es un manejo adecuado del inventario de las partes o piezas de los aviones, cada día estas piezas de aviones son demandadas por lo que pronosticar esta demanda es crucial para administrar el inventario ya que forma la base para planificar los niveles de inventario. La demanda de piezas de aviones ocurre en el proceso de mantenimiento en los aviones, el cual inspecciona cada mes una cantidad determinada de aviones lo que genera la demanda de diferentes piezas. La demanda total de un número de parte en un mes estará compuesta por la suma de las demandas individuales de los aviones en este periodo de tiempo.

Los números de piezas de los aviones presentan baja demanda, pero estas piezas son críticas para mantener operando un avión y la falta de estas piezas puede llevar a altos costos (los aviones que se mantienen en tierra, representan grandes pérdidas para las compañías de aviación). Dado el alto costo de los materiales utilizados en el proceso de mantenimiento de

los aviones y su alto nivel de sofisticación mantener grandes inventarios producen grandes riesgos y costos asociados al sobrestock. Los administradores de estas partes han tratado con demanda intermitente, la cual tiende a ser aleatoria en el tiempo y en cantidad, teniendo una larga proporción de valores ceros, este fenómeno ha recibido limitado estudio dentro de la industria de la aviación.

El pronóstico de la demanda intermitente hasta ahora, ha sido pobre. En buena parte esto se debe a que los métodos tradicionales de pronóstico tales como el suavizado exponencial y promedios móviles, trabajan bien cuando los datos son normales, pero no dan resultados acertados con datos intermitentes, que contienen una gran cantidad de ceros, lo cual es ignorado por estas técnicas, así como otras características claves de los datos.

Por estas razones se hace necesario investigar nuevas metodologías que permitan pronosticar adecuadamente la demanda intermitente. Es así como se decidió, desarrollar el presente trabajo el cual pretende establecer un modelo de redes neuronales cuyo base es la demanda intermitente del número de parte QA065423-01 (filtro del aire acondicionado) estudiando la relación de dependencia de variables tales como: edad del avión, ciclos de vuelo, tiempo de vuelo, ciclos de vuelo antes del último cambio de pieza, tiempo de vuelo transcurrido después del último cambio de pieza, número de meses transcurridos después del último cambio de pieza, y años de experiencia del mecánico.

1.3 ANTECEDENTES Y JUSTIFICACION.

En la presente investigación que se quiere desarrollar no existen estudios con respecto a la aplicación de los modelos neuronales al problema de la demanda intermitente en las piezas de los aviones en TACA, pero se han aplicado esta técnica en el contexto de la demanda intermitente de un distribuidor de componentes electrónicos en Monterrey, el resultado de este estudio mostró que las redes neuronales pueden modelar adecuadamente el problema de la demanda intermitente.[1] Otra de las técnicas poco tradicionales empleadas en este problema, ha sido el uso de la técnica conocida como máquinas de soporte vectorial (SVM) [2].

[1] http://www.geocities.com/ricardo_rios_sv/documentos_tesis/lumpy.pdf
[2] http://www.geocities.com/ricardo_rios_sv/documentos_tesis/svm.pdf

"Este método consiste en adoptar un punto de vista distinto del habitual: en lugar de buscar una reducción del espacio de los datos (como lo hace el análisis de componentes principales) y resolver el problema en ese espacio de dimensión menor, esta técnica busca un espacio de dimensión mayor donde los puntos pueden separarse de forma lineal"[3].

Entre los modelos mas utilizados para pronosticar la demanda intermitente se encuentran: el suavizado exponencial simple y el método de Croston, además, se han aplicado los modelos de Cox y de Wilcox, en la práctica los métodos estándar para pronosticar la demanda intermitente son el suavizado exponencial y el método de Croston. Estos métodos, sin embargo, al abordar el problema de la demanda intermitente en los números de partes de los aviones dan resultados poco satisfactorios y con amplios márgenes de error.

El presente trabajo tiene una gran importancia práctica ante lo poco estudiado del problema de la demanda intermitente, puede permitir dar nuevos conocimientos para abordar esta problemática, la cual al ser tratada con técnicas convencionales proporciona resultados poco confiables y dificulta el manejo de los inventarios.

En las últimas décadas del siglo XX una rama de la inteligencia artificial denominada Redes Neuronales Artificiales ha generado un gran número de aplicaciones y de mejoría a algunas ya existentes especialmente en el área Estadística.

Aproximadamente desde los primeros años de la década de los 70's, las redes neuronales comenzaron a desarrollarse, pero luego tuvieron una etapa de adormecimiento en vista que los algoritmos que regían el comportamiento de dichas redes tenían bastantes limitaciones. No fue sino hasta 1982, que se da un redescubrimiento que permitió la formulación de algoritmos para redes neuronales mucho mas generales y poderosos, de tal forma que la aplicabilidad de dichas técnicas en las áreas de Estadística y las ramas que dependen de ella, ha tenido una gran acogida y se encuentra reemplazando métodos tradicionales ocupados por años, marcando un periodo de auge de esta técnica.

[3] Peña Daniel, Análisis de Datos Multivariantes, Mc Graw Hill, S.A. 2002. Pág 452.

Uno de los modelos neuronales que más se utiliza para la resolución de problemas prácticos en el área estadística son las redes neuronales perceptron multicapa (MLP).

Debido a la poca investigación sobre redes neuronales en la Universidad de El Salvador, la investigación a desarrollar pretende dar a conocer esta técnica enfocada a resolver problemas estadísticos de clasificación. Entre las razones más importantes que motivaron su estudio, podemos plantear las siguientes.

Innovación

Aunque existe una extensa bibliografía relacionada con el modelo MLP (Multilayer Perceptron), los algoritmos para encontrar las estimaciones de los parámetros, que son tratados en la mayoría de estas obras se relacionan con técnicas que usan el gradiente negativo de la función de coste o error de la red. Estas técnicas a pesar de su rápida convergencia tienen el problema de que las soluciones que se obtienen dependen mucho de la inicialización de los parámetros de la red, si la inicialización es buena el proceso de minimización podrá tener una mayor probabilidad de encontrar el mínimo global de la función de error, sin embargo, si la inicialización no se hace lo mejor posible entonces se corre el riesgo de obtener una solución subóptima. Esta forma de buscar los parámetros de la red, requiere reestimar muchas veces el modelo y realizar muchos ajustes y por lo general es muy difícil de aplicar cuando la superficie del error es altamente no lineal, con muchos mínimos locales y puntos de silla.

La forma en que estimaremos los parámetros de la red lo realizaremos con una técnica de optimización estocástica denominada algoritmos genéticos, esta metodología al igual que las redes neuronales esta inspirada en la biología y ha superado las técnicas basadas en el gradiente descendente y ha sido incorporada en la mayoría de software sobre redes neuronales. La bibliografía acerca de este método de estimación de redes neuronales es muy escasa y poco accesible, por lo que consideramos que es un gran aporte para la comunidad universitaria ávida de nuevos conocimientos.

Aplicación

Las redes neuronales tratan de resolver de forma eficiente problemas que pueden encuadrarse dentro de dos amplios grupos: reconocimiento y generalización.

En los problemas de reconocimiento se entrena una red neuronal con inputs como sonidos, números, letras y se procede en la fase de prueba a presentar estos mismos patrones pero con ruido. Este es uno de los campos mas fructíferos en el desarrollo de redes neuronales.

En los problemas de generalización la red neuronal se entrena con unas variables de entrada y la prueba se realiza con otros casos diferentes, problemas típicos de generalización son los de clasificación y predicción.

Por lo anteriormente mencionado, la lista de problemas en los que se han aplicado con éxito redes neuronales es muy extenso y además crece constantemente, dar un listado representativo de estas aplicaciones es poco práctico. Sin embargo, nos dimos a la tarea de recopilar algunas de estas que sean aplicables a nuestro contexto socio-económico.

Medicina: Es un campo en que estos modelos han dado excelentes resultados, entre algunas de muchas aplicaciones podemos mencionar las siguientes: detección de picos epilépticos en los electroencefalogramas, clasificación de células cancerosas, clasificación de señales electrocardiográficas y electroencefalográficas, predicción de la reacción de pacientes ante determinados tratamientos, detección de arritmias, predicción de mortandad de pacientes, identificación de cáncer de pecho a partir de mamografías, clasificación de imágenes médicas, etc.

Economía: Sin duda alguna, las aplicaciones más difundidas en este campo han sido en el área financiera y de empresa. Algunos ejemplos son: Su uso en modelos de inversión, en el mundo del marketing, como instrumento para vender productos a clientes potenciales, en entidades financieras, para predecir el riesgo potencial de créditos, análisis de información contable, predicción de indicadores económicos, etc. Entre las aplicaciones que son útiles para la estadística podemos mencionar las siguientes.

Problema de los valores pérdidos: El ministerio de seguridad social en el Reino Unido utiliza una red neuronal para rellenar a posteriori el 70% de los 45,000 campos en blanco de su encuesta anual de recursos familiares. Dicha encuesta, cuenta con una muestra de 25,000 entrevistas, estaría llena de sesgos si los campos dejados en blanco por los entrevistadores se incluyeran en el análisis de los resultados. Una aplicación futura de las redes neuronales es la de rellenar los valores pérdidos también en el censo del Reino Unido.

Series de Tiempo: Es una de las aplicaciones en las cuales ha tenido mucho auge, su éxito se debe a la capacidad que tienen estos modelos de identificar patrones de comportamiento, en especial no lineales, esto les permite en estudio de series temporales poder detectar estas dinámicas, además, tienen la capacidad de reconocer y modelar comportamientos atípicos, entre ellos outliers o cambios en el nivel de las series de tiempo. Por último podemos mencionar que estos modelos no se tienen que preocupar de problemas de multicolinearidad, ni de ninguna asunción sobre los datos.

Desarrollo del área estadística

La estadística comprende un conjunto de métodos que sirven para recoger, organizar, resumir y analizar datos, así como para extraer conclusiones y tomar decisiones basadas en tal análisis. Las redes neuronales tienen muchos más aspectos en común con la estadística que con cualquier otra rama del saber, antes de puntualizar la ventaja de utilizar estos métodos en la investigación estadística, mencionaremos primero como se relacionan los modelos neuronales con la estadística.

Desde un punto de vista estadístico, muchos de los problemas que se intentan resolver con las redes neuronales entran en la categoría de los denominados **problemas mal planteados**, en los que el espacio de trabajo es tan amplio y los datos disponibles tan escasos, que resulta difícil encontrar la red neuronal que los ajusta correctamente.

"Hablando en términos estadísticos las redes neuronales son estimadores no paramétricos, que realizan estimaciones denominadas de modelo libre"[4]. Por ejemplo, la regresión lineal es un estimador paramétrico, pues se impone al problema un determinado modelo de partida (la línea recta). A diferencia de los modelos paramétricos, las redes neuronales como el MLP, seria un estimador de modelo libre, pues no se impone ninguna forma funcional de partida. Podemos objetar que las redes neuronales poseen pesos o parámetros y por lo tanto son paramétricas, sin embargo, aunque podemos saber el valor de cada peso, no sabemos la influencia exacta o efecto marginal de cada peso en el resultado final.

La desventaja de este enfoque es que la red neuronal posee inherentemente una gran varianza, por lo que para realizar un ajuste óptimo el número de ejemplos debería tender a infinito, esta es una condición indispensable al trabajar con estos modelos ya que necesitan una gran cantidad de datos.

Pese a que estadísticamente no poseen propiedades deseables, algunos modelos neuronales como el MLP realiza un tipo de regresión multidimensional y no lineal, en la que no es necesaria la suposición de una determinada forma funcional a priori, ni ningún otro supuesto; "en este sentido se suele observar en su aplicación práctica que el MLP y otros modelos neuronales superan con frecuencia a la regresión y otras técnicas lineales convencionales para espacios de dimensión alta (mayor que tres), aspecto que estudios rigurosos han corroborado"[5].

Pero en concreto, ¿qué ventaja aportaría el conocimiento de las redes neuronales al área estadística?. En primer lugar, los métodos neuronales, son relativamente fáciles de emplear y la interpretación de sus resultados resulta alcanzable a usuarios no necesariamente expertos en el tema. Otra ventaja, es que estos modelos no se acoplan a una determinada suposición de la distribución de los datos, el enfoque es por lo tanto "dejar hablar a los datos". Además, las redes neuronales, gracias a su posibilidad de entrenamiento en línea, pueden utilizarse en

[4] Vapnik, V.M. Guest Editorial: Vapnik-Chervonenkis Learning Theory and It's Applications. IEEE Transaction on Neural Networks, 10,5, pp. 985-7, 1999.
[5] Principe, J.C., Euliano, N. R., Lefebvre, W. C. Neural and Adaptative Systems. Fundamentals Through Simulations. John Wiley 2000.

aplicaciones de control industrial, en la que los datos van llegando uno tras otro, tarea imposible para una herramienta estadística en la que se requiere la presencia de todos los datos simultáneamente desde el principio.

Pese a su éxito, las redes neuronales presentan ciertos inconvenientes. Aunque las redes neuronales pueden muchas veces superar a muchas técnicas estadísticas, no se interpreta fácilmente su funcionamiento, por lo que es desaconsejable su uso cuando el propósito de la investigación es la explicación del fenómeno. Además, estos modelos son incapaces de contrastar estadísticamente si los inputs utilizados son significativos, por lo tanto, las redes neuronales no poseen propiedades paramétricas deseables para contrastar hipótesis equivalentes a las pruebas t o F. Sin embargo podemos llevar a cabo análisis de sensibilidad del modelo. 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, pero sin duda con unos resultados mucho menos robustos que sus contrapartes parametricas.

Pese a sus ventajas e inconvenientes consideramos que las técnicas estadísticas y neuronales están muy relacionadas, ningún campo es a priori mejor que otro (en algunos casos ambos son indistinguibles), y que lo razonable es aplicar este método cuando mejorar la predicción o clasificación, pero no la explicación, sean los puntos principales de la investigación.

Por último, siendo la estadística una de las principales herramientas que se utiliza en redes neuronales, consideramos que los profesionales en estadística son los que disponen de una mejor formación para ser expertos profesionales en este campo y a su vez posibilitar su desarrollo.

Importancia del proyecto

Se tuvo conocimiento del problema de la demanda intermitente en las piezas de los aviones mediante conversaciones que se tuvieron con el responsable de la administración y planeación del inventario de piezas de aviones de AEROMAN. En las platicas se mencionó

que para la empresa es importante investigar este tema porqué les puede permitir reducir los costos e inversiones del mantenimiento de aviones. Luego se realizaron visitas a AEROMAN con el objetivo de tener una mejor comprensión del problema y de su abordaje como tema de tesis.

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

La Investigación: en este caso, se esta realizando un proyecto basado en la estadística, matemáticas y computación, para apoyar las ciencias administrativas (gestión de inventarios).

La Proyección Social: la empresa TACA como parte de su actividad genera una gran cantidad de empleos tanto directos como indirectos, lo cual representa un gran aporte para el país, por lo que cualquier acción tendiente a mejorar el servicio que presta, es de beneficio para nuestra sociedad.

Y la promoción de la carrera: como se menciona en la descripción de la Licenciatura en Estadística un aspecto fundamental que justifica la necesidad de crear y desarrollar una carrera en el área estadística es su aplicabilidad en el ámbito empresarial. Consideramos que el presente trabajo puede fomentar el uso de la metodología estadística en este ámbito de la actividad del país y fomentar el conocimiento de la Licenciatura en Estadística.

Factibilidad de la investigación

Dado que se cuenta con recursos técnicos tales como amplia bibliografía, equipo adecuado, software disponible en windows para simular el modelo, conocimiento de técnicas de programación orientada a objetos en varios lenguajes de programación, herramientas de apoyo como Internet y listas de correo sobre el tema, etc; los recursos económicos que implica el desarrollo del proyecto son mínimos debido a la existencia del recurso técnico apropiado. Todo esto junto con el plan de trabajo propuesto nos permitirá asegurar que es factible llevar a cabo este proyecto según lo planificado.

1.4 OBJETIVOS.

1.4.1 GENERALES

1. El propósito de esta investigación es aplicar los conocimientos estadísticos adquiridos a la base de datos del sistema de control de inventario de partes de aviones de AEROMAN para desarrollar un modelo de redes neuronales que nos permita evaluar la relación entre la demanda del número de parte QA065423-01 (variable respuesta o dependiente) y la información del uso del avión, datos referentes al último de cambio de pieza, y la experiencia del mecánico (variables de entrada o explicativas)
2. Potenciar en la Universidad de El Salvador el estudio de la teoría de Redes Neuronales.

1.4.2 ESPECIFICOS

1. Dar un aporte a la carrera Licenciatura en Estadística, por medio del estudio teórico sobre redes neuronales, con énfasis en el área de problemas estadísticos de clasificación binaria.

2. Desarrollar una aplicación informática sobre redes neuronales en el área de interés y que sirva para el desarrollo y difusión de estos métodos.

3. Estimar y evaluar un modelo neuronal con la aplicación informática para obtener la mejor estimación de probabilidad de que el número de parte QA065423-01 sea reemplazado con base en los datos del sistema de control de inventario de partes de aviones de AEROMAN.

4. Proporcionar a la empresa TACA información que permita determinar la factibilidad de utilizar redes neuronales para pronosticar la demanda total del número de parte QA065423-01.

1.5 METODOLOGÍA

La metodología a emplear deberá partir 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 las 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

Para nuestra investigación se deberá primero realizar una selección de las variables de interés en los datos referidos al número de parte QA065423-01 de la base de datos del sistema de control de inventario de partes de aviones de AEROMAN. 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 explicativas que hemos considerado para nuestra investigación se pueden clasificar en tres categorías:

1. Variables del uso del avión.
2. Variables referentes al último cambio de pieza.
3. Experiencia del mecánico.

1) Variables del uso del avión.

Estas variables se relacionan con el uso que se le da al avión, y se detallan a continuación:

Edad del avión: Representa el número de meses que tiene el avión.

Ciclos de vuelo del avión: Un avión realiza un ciclo cuando despega o aterriza, por lo tanto esta variable nos indica la suma de despegues y aterrizajes que ha realizado el avión.

Tiempo de vuelo del avión: Esta variable nos indica el total de horas que ha volado el avión.

2) Variables referentes al último cambio de pieza.

Las condiciones relacionadas al último cambio de pieza tienen una gran influencia en la decisión del cambio de pieza del filtro del avión, ya que esta es una parte del avión que tiende a deteriorarse a medida que tenga más uso, estas variables son:

Ciclos de vuelo transcurridos desde el último cambio de pieza: Representa los ciclos de vuelo que el avión ha realizado desde que se le hizo el último cambio de pieza.

Tiempo de vuelo transcurrido desde el último cambio de pieza: Esta variable nos indica el total de horas que el avión ha volado desde que se hizo el último cambio de pieza.

Días transcurridos desde el último cambio de pieza: Representa el número de días transcurridos desde que se le efectuó al avión el último cambio de pieza.

3) Experiencia del mecánico.

Las decisiones que toman los mecánicos con respecto al cambio de una pieza difieren de un mecánico a otro, parte de la incapacidad de predecir la demanda intermitente puede deberse a este hecho, ya que las decisiones de cambio de piezas posiblemente tenderán a ser más homogéneas ha medida que los mecánicos posean más años de experiencia y por eso hemos considerado la inclusión de la variable experiencia del mecánico la cual mide el número de años que ha estado laborando el mecánico que realiza la inspección del área del avión en la cual se encuentra localizada el número de parte QA065423-01.

Considerando las variables anteriormente descritas y en base a que nuestro horizonte de pronóstico es de un mes podemos formular la función de la demanda del número de parte QA065423-01 en un avión determinado al que se le va a ser revisión en el mes t como:

$$D(t) = f(U_{t-1}, C_{t-1}, E_t)$$

Donde t: mes a pronosticar

D(t): demanda a pronosticar

con valores 1 -> se cambia la pieza

0 -> no se cambia la pieza

U_{t-1} : variables de uso del avión un mes antes

C_{t-1} : variables de cambio de pieza un mes antes

E_t : experiencia del mecanico que va inspeccionar el avión en el mes a pronosticar.

Debemos mencionar que la variable experiencia del mecánico se tiene con un mes de anticipación debido a la planificación que se realiza de las tareas de mantenimiento en los aviones.

1.5.2 ELECCIÓN DE LOS CONJUNTOS DE APRENDIZAJE Y PRUEBA

Una vez que hemos obtenido los datos con los cuales vamos a realizar nuestro análisis, procederemos a dividirlos en dos grupos: uno de aprendizaje y uno de prueba. De esta forma, se procederá al entrenamiento del MLP solamente con los ejemplos del conjunto de aprendizaje, mientras que se comprobará la capacidad predictiva o de generalización de la red con los datos del conjunto de prueba. En caso de ser un número pequeño quizas haya que aplicar alguna técnica especifica de entrenamiento para pocos patrones.

1.5.3 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 usamos la función de activación sigmoidea la cual trabaja para valores entre 0 y 1, debemos transformar los datos de tal manera que estén comprendidos dentro de este rango y además que no saturen la función de activación.

1.5.4 PROCESO DE ENTRENAMIENTO

En esta fase se obtendrán 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, encontrar las soluciones por medio de métodos analíticos es imposible en este problema ya que se

desconoce a priori la forma de la función de error por que esta se construye a medida que se modifican los pesos.

"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 cuando en el ajuste de unos datos a un polinomio empleamos uno de grado demasiado elevado: sobran grados de libertad y aunque ajusten perfectamente los casos que nos proporcionan (conjunto de aprendizaje), nos apartamos de la tónica general y fallamos ante nuevos casos (conjunto de prueba). Por otro lado, si el número de neuronas no es suficiente no obtendremos un error aceptable ni siquiera para los datos que queremos ajustar. Para decidir el número de neuronas ocultas recurriremos a la siguiente técnica"[6] :

- **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 prueba.

Aunque existe software que nos permite realizar este proceso debemos tomar en cuenta que "los algoritmos genéticos son métodos no algorítmicos y ellos se adaptan al problema"[7], por consecuencia los software diseñados para tal efecto son de propósito general y pueden no adaptarse a las particularidades que pueda tener nuestro problema de estudio, por lo tanto su implementación es necesaria y la realizaremos en MATLAB.

[6] Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes Neuronales y sistemas borrosos , Editorial RA-MA, Universidad de Zaragoza 2001, Págs 217-218.
[7] http://tolstoy.newcastle.edu.au/R/help/98b/0474.htm

1.5.5 EVALUACIÓN DE RESULTADOS

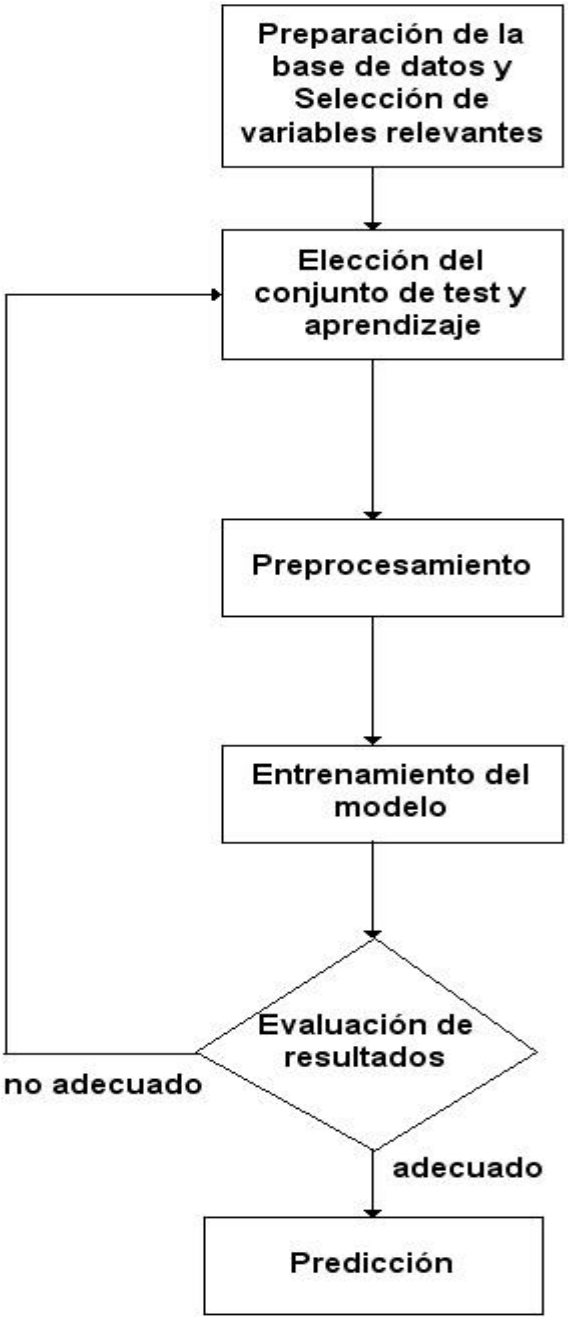
Finalizada la fase de entrenamiento y almacenadas las mejores estimaciones de los parámetros, ya estaremos en disposición de aplicar el modelo sobre casos nuevos (no empleados en el entrenamiento) para medir su eficacia de forma completamente objetiva. Si se comprueba que se siguen obteniendo resultados dentro del margen de error deseado, se procederá a implementar una herramienta informática que en base a la mejor estimación de los parámetros del modelo nos permita determinar previsiones mensuales de la demanda total del número de parte QA065423-01, es decir, la demanda total es una aplicación lineal de las demandas individuales de los aviones que ingresan en un mes para revisión. Podemos formular la función de la demanda total como:

$$\begin{aligned} DT(t) &= f(D_1(t), D_2(t), ..., D_n(t)) \\ &= D_1(t) + D_2(t) + ... + D_n(t) \\ &= \sum_{i=1}^{n} D_i(t) \end{aligned}$$

Donde t : mes a pronosticar
DT(t): demanda total a pronosticar
$D_i(t)$: demanda en el avión iésimo que pasa a revisión en el mes t
n : número de aviones que pasan a revisión en el mes t

En el caso de no obtener resultados aceptables, habrá que revisar alguna de las fases anteriores: puede que los conjuntos de aprendizaje y de prueba no sean representativos, que las variables no hayan sido preprocesadas o escaladas adecuadamente, que la reducción de la dimensión de las variables provocó que se perdiera información. Puede suceder, también, que el MLP no sea el modelo más apropiado o que su resolución no sea viable. El proceso de la metodología que vamos a seguir en la investigación se puede esquematizar de la forma siguiente.

Figura 1.1: Metodología propuesta para el trabajo de investigación



CAPITULO II. 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. Por último, se explican el modelo de suavizado exponencial el cual es uno de los modelos estadísticos más utilizados para pronosticar la demanda intermitente en los inventarios.

2.2 TEORÍA DE REDES NEURONALES

2.2.1 HISTORIA DE LAS REDES NEURONALES[8]

2.2.1.1 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 gente 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

[8] http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/c14.html

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.

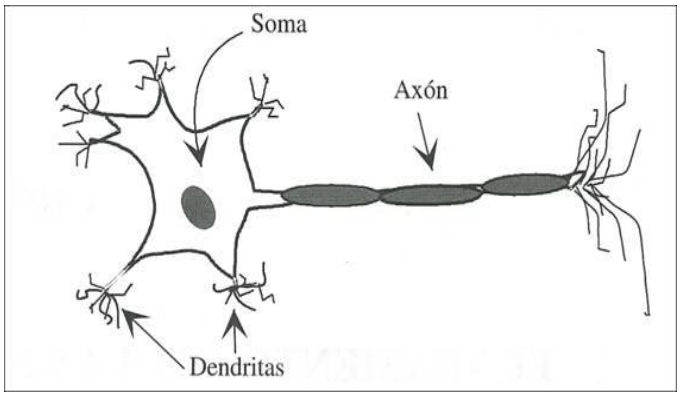
Dentro de la IA, se trabajó en dos enfoques distintos. Por un lado, se desarrolló lo que se conoce como el enfoque simbólico. Este enfoque asienta sus bases en la manipulación de símbolos en vez del mero cálculo numérico, tradicional de la computación algorítmica. La realidad se plasma por medio de una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

2.2.1.2 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

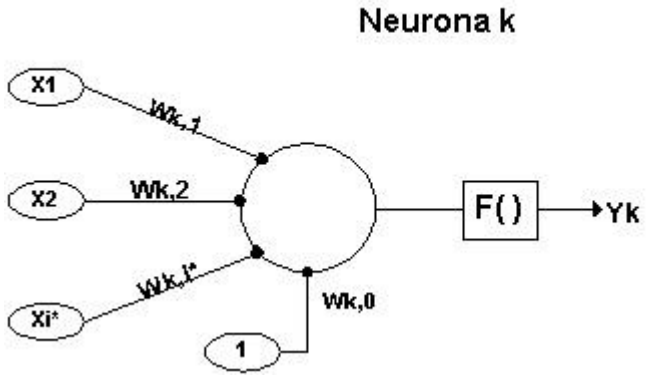
recoger información de las experiencias y utilizarlas para actuar ante situaciones futuras. Íntimamente relacionada con el aprendizaje esta la generalización, que podría definirse como la capacidad para abstraer la información útil, más allá de los casos particulares. De esta manera, la red neuronal es capaz de responder ante casos desconocidos.

2.2.2 CONCEPTOS BÁSICOS SOBRE REDES NEURONALES

2.2.2.1 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 siguiente figura 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:

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

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 un 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, que es un número, se modifica durante el entrenamiento de la red neuronal, y es aquí, por tanto, 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.

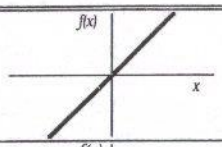
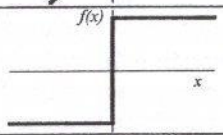
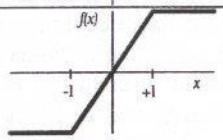
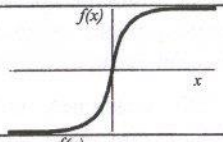
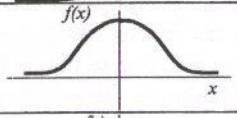
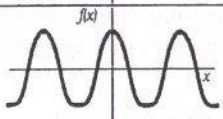
Una regla de propagación. Con esas entradas y los pesos sinápticos, se suele hacer algún tipo de operación para obtener el valor del potencial postsináptico (valor que es función de las entradas y los pesos y que es el que se utiliza en último término para realizar el procesamiento) Una de las operaciones 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:

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

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.

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	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

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 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. Mas adelante, comentaremos las funciones de activación más usadas para predicción.

2.2.2.2 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. ¿Qué 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 especifico. Las más comunes son las siguientes:

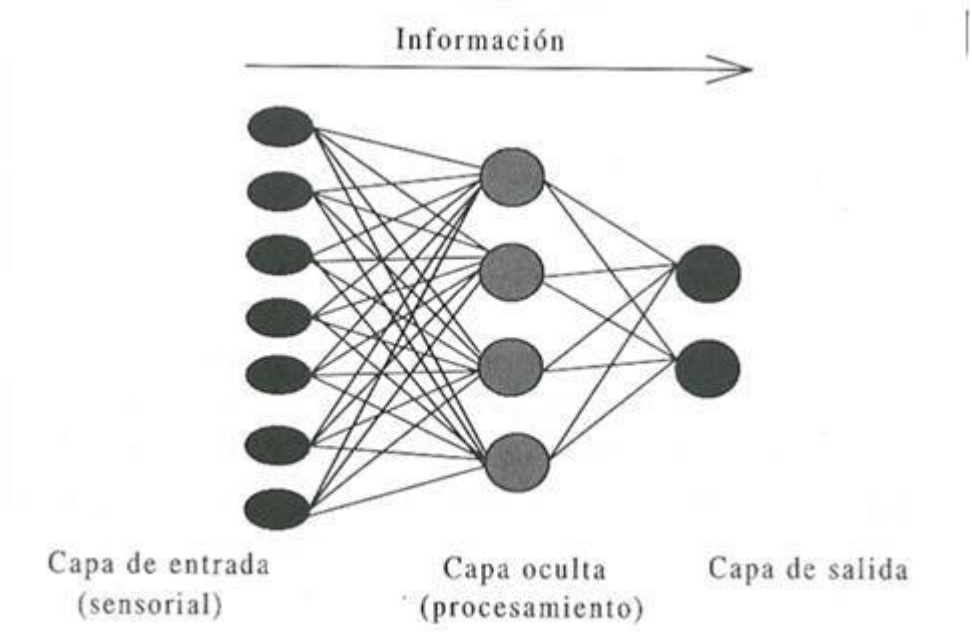
Capa de entrada: las neuronas de la capa de entrada, reciben los datos que se proporcionan a la red para que los procese.

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 siguiente figura ejemplifica lo anteriormente dicho:

Figura 2.4: 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 log-sigmoidea, en cambio en los métodos estadísticos el procesamiento de la información es secuencial. Mientras que este enfoque podría parecer esotérico, esto representa un camino muy eficiente para modelar procesos estadísticos no lineales" [9].

2.2.2.3 TIPOS DE REDES NEURONALES

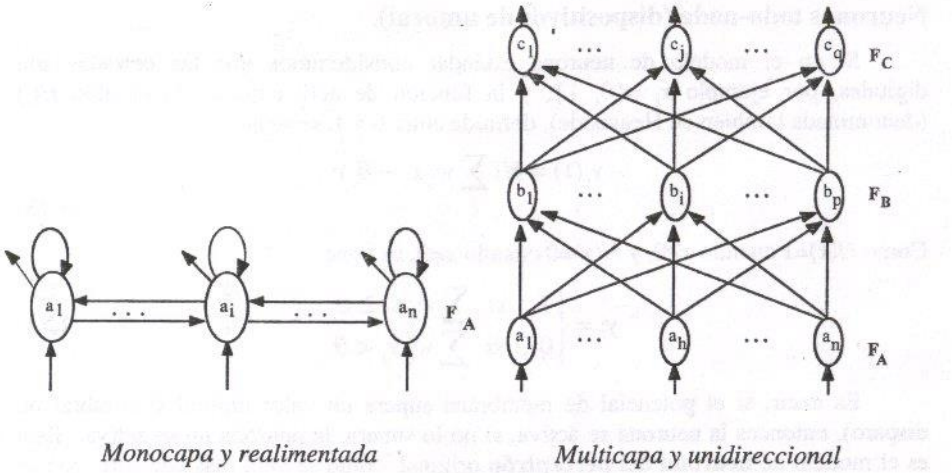
Según el criterio que escojamos para clasificar las redes neuronales tendremos una clasificación u otra. Lo más común es usar la arquitectura y el tipo de aprendizaje como criterios de clasificación.

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

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 se puede trazar un camino de una neurona a sí misma la arquitectura presenta ciclos. Este tipo de redes se denominan recurrentes o realimentados (recurrent).

Figura 2.5: 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.

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

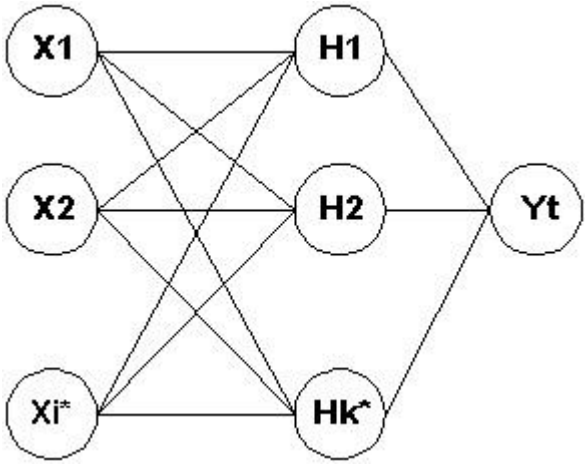
reconocer regularidades en el conjunto de entradas, es decir, estimar una función densidad de probabilidad p(x) que describe la distribución de patrones x en el espacio de entrada R^n.

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 2.6 se muestra la arquitectura neuronal más frecuentemente usada en problemas de pronóstico.

Figura 2.6: Red neuronal unidireccional multicapa



La red presentada en la figura 2.6 tiene i* variables de entrada Xi para i=1 hasta i=i*, k* neuronas ocultas Hk 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:

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

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.

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

Donde L es por lo general:

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

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

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.

2.2.2.4 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"[10].

Las siguientes ecuaciones describen esta red:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i^*} w_{k,i} x_{i,t}$$
$$N_{k,t} = L(n_{k,t})$$
$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

Donde $L(n_{k,t})$ 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,...,i*, y t=1,...,T con el vector de coeficientes o conjunto de pesos de la capa de entrada observadas $w_{k,i}$ i=1,...,i*, además, el termino constante $w_{k,0}$ forma la variable $n_{k,t}$, esta variable es transformada por la función log-sigmoidea, 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,...,k*, y agregándole el termino constante γ_0, se construye la salida de la red $\hat{y}_t$ 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:

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

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

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

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

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

Donde $T(n_{k,t})$ es la función de activación 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 log-sigmoidea, 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 es dada por el siguiente sistema:

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

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

$$= \int_{-\infty}^{n_{k,t}} \sqrt{\frac{1}{2\pi} e^{-0.5 n_{k,t}^2}}$$

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

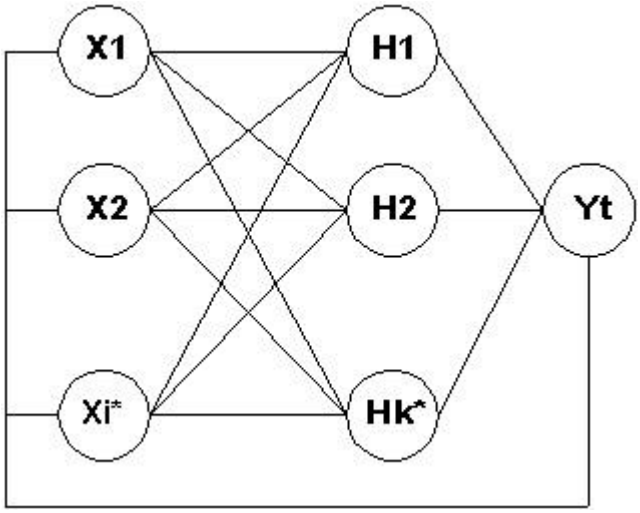
2.2.2.5 CONEXIONES CON SALTOS[11]

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.

[11] Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005, Pág 30-32.

La figura 2.7 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.

Figura 2.7: 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 $\{\beta_i\}$ i=1,...,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.

2.2.2.6 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 .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.

2.2.2.7 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.

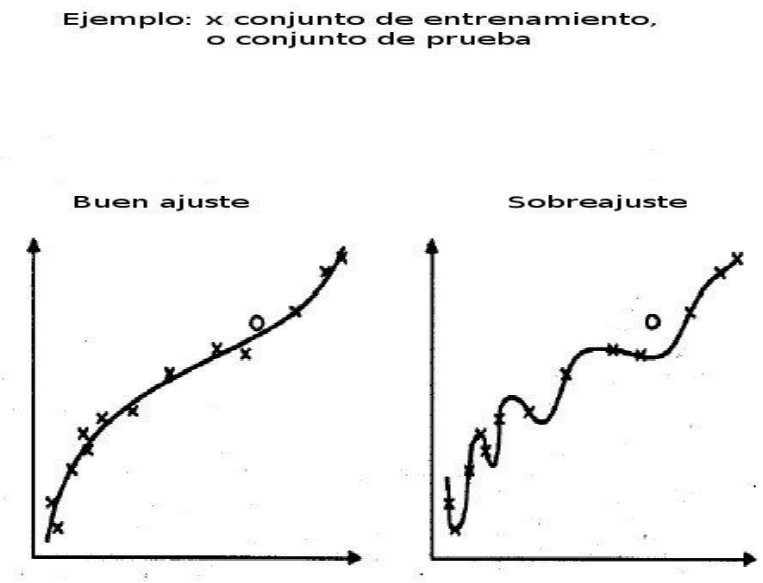
- La información requerida para lograr un modo adecuado de generalización no debe estar oculta por otros rasgos o propiedades de los datos de entrenamiento. En este sentido, en muchos casos es necesario un proceso previo de extracción de rasgos.

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.

2.2.2.8 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 2.8 se ilustra el problema del sobreajuste .

Figura 2.8: 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 se debe evitar.

2.2.3 ESTIMACIÓN DE UNA RED NEURONAL[12]

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

[12] Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005.

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.

2.2.3.1 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[13], 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 log-sigmoidea) o –1 (para la función de activación tangente hiperbólica) , si no se escalan los datos posiblemente habrá una gran pérdida 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 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

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

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.

2.2.3.2 EL PROBLEMA DE LA ESTIMACIÓN NOLINEAL

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 óptimas, 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 2.9 ilustra el problema de encontrar óptimos globales o mínimos globales en una superficie altamente no lineal.

Figura 2.9: Pesos y función de error

Como muestra la figura 2.9, 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 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 $\Omega = \{w_{k,i}, \gamma_k\}$ de una red con una capa oculta, nosotros minimizaremos la función de pérdida ψ, definida como la suma de los cuadrados de las diferencias entre la salida actual observada y, y $\hat{y}$ la salida pronosticada de la red.

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

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 pérdida la entropía que viene dada por:

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

Utilizando esta función de pérdida 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 evolucionaria estocástica comúnmente conocida como algoritmos genéticos, la cual inicia con una población de p supuestas inicializaciones, $[\Omega_{01}, \Omega_{02}, ..., \Omega_{0P}]$, 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.

2.2.3.3 DIFERENCIAS ENTRE LOS ALGORITMOS GENÉTICOS Y LOS METODOS TRADICIONALES DE OPTIMIZACIÓN[14]

- 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 probabilísticos, mientras las otras técnicas usan operadores determinísticos.

- Las técnicas evolutivas no necesitan conocimientos específicos sobre el problema que intentan resolver.

[14] Apuntes del Curso Regional de Biomatemáticas y TIC aplicados a la Enseñanza de las Matemáticas El Salvador – Febrero 2006.

2.2.3.4 VENTAJAS DE LAS TÉCNICAS EVOLUTIVAS[15]

- 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.

2.2.3.5 BÚSQUEDA ESTOCÁSTICA EVOLUCIONARIA: ALGORITMOS GENÉTICOS

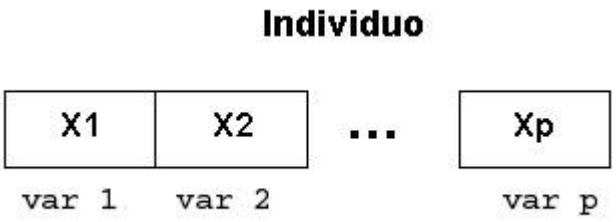
Los algoritmos genéticos (AG) son métodos sistemáticos para la resolución de problemas de búsqueda y optimización basados en los mismos métodos de la evolución biológica: Selección basada en la población, reproducción sexual y mutación. La principal ventaja de los algoritmos genéticos es reducir la probabilidad de obtener soluciones óptimas 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

[15] Apuntes del Curso Regional de Biomatemáticas y TIC aplicados a la Enseñanza de las Matemáticas El Salvador – Febrero 2006.

una *cadena*, es decir, una secuencia de símbolos (números) en la figura 2.10 se observa un cromosoma con p variables, para nuestro caso las variables representaran los pesos de la red.

Figura 2.10: Ejemplo de un cromosoma o individuo



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 $px1$.

Figura 2.11: Creación de una población de vectores aleatorios

$$\begin{pmatrix}\Omega_1\\ \Omega_2\\ \Omega_3\\ .\\ .\\ .\\ \Omega_p\end{pmatrix}_1 \begin{pmatrix}\Omega_1\\ \Omega_2\\ \Omega_3\\ .\\ .\\ .\\ \Omega_p\end{pmatrix}_2 \cdots \begin{pmatrix}\Omega_1\\ \Omega_2\\ \Omega_3\\ .\\ .\\ .\\ \Omega_p\end{pmatrix}_{N^*}$$

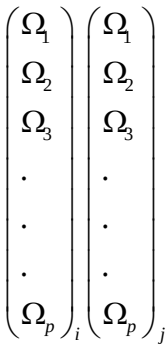
2. Selección

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 pérdida ψ la cual hemos definido con anterioridad, estos cuatro vectores de coeficientes agrupados en dos

parejas, le calculamos la función de pérdida. 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.

Figura 2.12: Selección de dos vectores aleatorios



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 fijada $p > 0$. 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 $G = 1, 2, ..., G^*$, es dada por la probabilidad:

$$\tilde{p}_r = 0.15 + \frac{0.33}{G}$$

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 $r1$ y $r2$ 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:

$$\tilde{\Omega}_{i,p} = \begin{cases} \Omega_{i,p} + s\left[1 - r_2^{(1-G/G^*)^b}\right] & \text{si } r_1 > 0.5 \\ \Omega_{i,p} - s\left[1 - r_2^{(1-G/G^*)^b}\right] & \text{si } r_1 \leq 0.5 \end{cases}$$

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 $G \to G^*$. Así la probabilidad de que un elemento mute evoluciona a través del tiempo.

Algunos investigadores como Fogarty[16], 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 actitud. 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 actitud 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

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

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 pérdida ψ, El valor de G^* podría ser muy grande , pero sin embargo, podría pasar 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 cruce, 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 significancia y plausibilidad de los resultados.

2.2.4.1 CRITERIO DENTRO Y FUERA DE LA MUESTRA[17]

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

[17] NeuroShell Classifier, Archivos de ayuda del NeuroShell Classifier, versión 2.0, 2005.

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 estimación. 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 médico, 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 verdaderos-negativos 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 falsos-negativos 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.

Sensitividad 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. Sensitividad 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.

Sensitividad 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 sensitividad 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 sensitividad 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)

2.2.4.2 EL MÉTODO BOOTSTRAP 0.632[18]

Desgraciadamente, muchas veces los problemas del mundo real no tienen un número 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 "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"[19] aparece en la tabla 2.1.

[18] Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005, Pág 101.

[19] 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 $1-\left[1-\left(\frac{1}{n}\right)\right]^n \approx 1-e^{-1}=0.632$.

Tabla 2.1: Prueba "Bootstrap 0.632" para el sesgo fuera de la muestra.

Definición	Operación
Obtener el error cuadrático medio para todo el conjunto de datos	$MSSE^0 = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \hat{y}_i\right]^2$
Extraer una muestra de longitud n con reemplazamiento	z_1
Estimación de los coeficientes del modelos	Ω^1
Obtener los datos omitidos del conjunto total de datos	$\tilde{z}$
Pronosticar fuera de la muestra con los coeficientes Ω^1	$\hat{\tilde{z}}_1 = \hat{\tilde{z}}_1\left(\Omega^1\right)$
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	$\text{MSSE}' = \frac{1}{B}\sum_{b=1}^{B} MSSE^b$
Calcular el sesgo ajustado	$\bar{\omega}^{(0.632)} = 0.632\left[MSSE^0 - MSSE'\right]$
Calcular el error estimado ajustado	$MSSE^{(0.632)} = 0.368 * MSSE^0 + 0.632 * MSSE'$

2.2.4.3 CRITERIO INTERPRETATIVO Y SIGNIFICANCÍA DE LOS RESULTADOS[20]

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

[20] Paul D. McNelis, Neural Networks in Finance, Editorial Elsevier, 2005.

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? 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 dado por:

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

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 una función N log sigmoidea tiene la siguiente propiedad:

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

Los modelos lineales implican derivadas parciales que son independientes de los valores de x. 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 x.

Diferencias Finitas

El camino más común usado para calcular las derivadas son los métodos de diferencia finita. Dada la función de una red neuronal, $y = f(x)$, $x = \left(x_1, x_2, ..., x_i, ..., x_{i^*}\right)$ un camino para aproximar $f_{i,t}$ es a través de la formula de diferencias finitas.

$$\frac{\partial y}{\partial x_i} = \frac{f\left(x_1, ..., x_i + h_i, ..., x_{i^*}\right) - f\left(x_1, ..., x_i, ..., x_{i^*}\right)}{h_i}$$

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

La segunda derivada parcial es calculada de la siguiente manera:

$$\frac{\partial^2 y}{\partial x_i \partial x_j} = \frac{1}{h_j h_i}\left[\begin{array}{l}\left\{f\left(x_1, ..., x_i + h_i, x_j + h_j, ..., x_{i^*}\right) - f\left(x_1, ..., x_i, ..., x_j + h_j, ..., x_{i^*}\right)\right\} \\ -\left\{f\left(x_1, ..., x_i + h_i, x_j, ..., x_{i^*}\right) - f\left(x_1, ..., x_i, ..., x_j, ..., x_{i^*}\right)\right\}\end{array}\right]$$

Mientras que la derivada parcial directa de segundo orden es dada por:

$$\frac{\partial^2 y}{\partial x^2{}_i} = \frac{1}{h^2{}_i}\left[\begin{array}{l}\left\{f\left(x_1, ..., x_i + h_i, x_j, ..., x_{i^*}\right) - 2f\left(...x_i, ..., x_j, ..., x_{i^*}\right)\right\} \\ +\left(x_1, ..., x_i - h_i, x_j, ..., x_{i^*}\right)\end{array}\right]$$

El valor de h_i se establece como $h_i = \max(\varepsilon x_i, \varepsilon)$, 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 $\left[y^{b1}, x \right]$ 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 $\hat{e}^{bi}$ 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.

2.2.5 IMPLEMENTANDO EL MODELO[21]

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

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

con el mismo conjunto de pesos en los tres criterios. Pero en general nosotros deberíamos estar satisfechos si el modelo cumple los tres criterios.

2.2.5.1 ALGUNOS ASPECTOS A TENER EN CUENTA PARA EL USO DE REDES NEURONALES[22]

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.
- 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.
- Tareas críticas o potencialmente peligrosas, cuya resolución deba ser siempre perfectamente predecible y explicable. A veces no resulta fácil interpretar la

[22] Bonifacio Martín del Brío y Alfredo Sanz Molina, Redes Neuronales y sistemas borrosos , Editorial RA-MA, Universidad de Zaragoza 2001, Pág 211.

operación de la red neuronal o predecir con total fidelidad el resultado que pueda proporcionar en todos los casos posibles.

2.3 TEORIA ESTADISTICA

2.3.1 SUAVIZADO EXPONENCIAL[23]

En estadística, el suavizado exponencial se refiere a un particular tipo de promedio móvil, la cual suele ser usada para realizar pronósticos. El suavizado exponencial es comúnmente aplicado a mercados financieros y datos económicos, pero este puede ser usado con cualquier conjunto discreto de medidas repetidas. Los datos originales son a menudo representados por $\{ x_t \}$, y la salida del el suavizado exponencial es comúnmente escrita como $\{ s_t \}$ la cual puede ser considerada como la mejor estimación del siguiente valor de x.

PROMEDIO MOVIL SIMPLE

Intuitivamente, la mejor manera de suavizar una serie de observaciones es calcular un simple promedio móvil. El estadístico de suavizado s_t es entonces solo la media de las ultimas k observaciones.

$$s_t = \frac{1}{\mathrm{k}} \sum_{n=0}^{k-1} x_{t-n} = \frac{\mathrm{x_t} + x_{t-1} + \ldots + x_{t-k+1}}{k}$$
$$= \mathrm{s_{t-1}} + \frac{x_t - x_{t-k}}{k}$$

Donde k>1 y su elección es arbitraria. Un pequeño valor de k tendrá menos efectos en el suavizado y más sensible a cambios recientes en los datos. Una desventaja de esta técnica es que no puede ser usada en los primeros k-1 términos de la serie de datos.

PROMEDIO MOVIL PONDERADO

Un ligeramente más complicado método para suavizar una serie de datos $\{ x_t \}$ es calcular un promedio móvil ponderado seleccionando un conjunto de factores de ponderación,

[23] http://en.wikipedia.org/wiki/Exponential_Smoothing

$\{w_1, w_2, ..., w_k\}$ tales que $\sum_{n=1}^{k} w_n = 1$

y entonces usar estos pesos para calcular la serie suavizada $\{s_t\}$ de la siguiente forma:

$$s_t = \sum_{n=1}^{k} w_n x_{t+1-n} = w_1 x_t + w_2 x_{t-1} + ... + w_k x_{t-k+1}$$

En la practica los factores de ponderación son a menudo seleccionados para dar más importancia a los más recientes términos en la serie y menos peso a los datos mas antiguos. Note que esta técnica tiene la misma desventaja que el promedio móvil, ya que no puede ser calculada hasta que haya por lo menos k observaciones y esto podría implicar un calculo más complejo en cada paso del proceso de suavizado.

SUAVIZADO EXPONENCIAL

La más simple forma de suavizado exponencial es dada por la formulas:

$$\begin{aligned} s_0 &= x_0 \\ s_t &= \alpha x_t + (1-\alpha) s_{t-1} \\ &= s_{t-1} + \alpha (x_t - s_{t-1}) \end{aligned}$$

Donde α es el factor de suavizado, y $0 < \alpha < 1$. En otras palabras el estadístico de suavizado s_t es un simple promedio ponderado de la ultima observación x_t y el anterior estadístico de suavizado s_{t-1}. El suavizado exponencial simple es fácilmente aplicable, y produce un estadístico de suavizado con apenas solo dos observaciones.

Valores de α cercanos a uno tendrán un menor efecto de suavizado y son menos sensibles a recientes cambios. No existe un método formal para seleccionar α. Alternativamente, alguna técnica estadística puede ser usada para optimizar el valor de α. Por ejemplo, el método de mínimos cuadrados podría ser usado para determinar el valor de α donde la suma de $(s_{n-1} - x_n)^2$ es minimizada. La forma simple del suavizado exponencial también suele ser conocida como “Suavizado exponencial de Brown” y como “Promedio movil ponderado exponencial”.

POR QUÉ ES EXPONENCIAL?

Haciendo una directa sustitución de la ecuación que define el suavizado exponencial simple se descubre lo siguiente:

$$\begin{aligned} s_t &= \alpha \mathrm{x}_\mathrm{t} + (1-\alpha) s_{t-1} \\ &= \alpha \mathrm{x}_\mathrm{t} + \alpha(1-\alpha) x_{t-1} + (1-\alpha)^2 s_{t-2} \\ &= \alpha \left[\mathrm{x}_\mathrm{t} + (1-\alpha) x_{t-1} + (1-\alpha)^2 x_{t-2} + (1-\alpha)^3 x_{t-3} + \ldots \right] + (1-\alpha)^t x_0 \end{aligned}$$

En otras palabras cuando el tiempo pasa el estadístico de suavizado s_t se hace un promedio ponderado de un mayor y mayor número de la pasada observación x_{t-n}, y los factores asignados a las observaciones anteriores son en general proporcionales a los términos de la progresión geométrica:

$$\{1, (1-\alpha), (1-\alpha)^2, (1-\alpha)^3, \ldots\}$$

Una progresión geométrica es la versión discreta de una función exponencial, de este hecho es donde se origina el nombre de suavizado exponencial.

CAPITULO III: DISEÑO DE LA INVESTIGACIÓN

3.1 INTRODUCCION

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 las variables relevantes, la elección de los conjuntos de aprendizaje y prueba con fines de validación, e información sobre la implementación de las herramientas informáticas desarrolladas en la investigación.

3.2 PREPARACIÓN DE LA BASE DE DATOS

Para poder llevar a cabo la preparación de la base de datos se solicitó el apoyo del Gerente de Control de Inventarios de AEROMAN, quien proporcionó ficheros en Excel con datos históricos de la demanda del número de parte QA065423-01 y la información relacionada con las variables de interés. Sin embargo, esta información es confidencial por lo tanto no puede ser publicada.

La base de datos obtenida comprende 141 observaciones pertenecientes al período que va de Marzo de 1999 a Febrero del 2007. Se asocia a la base de datos, un libro de códigos en el que se detallan los nombres de las variables utilizadas, su codificación, su tipo, su rango de valores, su significado, así como la escala que se utiliza (ver ANEXO 1); para ver la descripción de las variables referirse a la sección 1.5.1.

Se debe mencionar que las variables referentes al último cambio de pieza mencionadas en la sección 1.5.1, se tuvieron que obtener a partir de las variables del uso del avión. Para lograr calcular estas variables se utilizo el Lenguaje de Consulta Estructurado (Structured Query Language) o mejor conocido como SQL, el cual permite recuperar información de interés de una base de datos, de una forma sencilla. Se procedió por tanto a migrar la información al sistema gestor de base de datos conocido como PostgreSQL, para ello se recurrió a crear una tabla cuyos campos se correspondieran a los códigos de las variables definidos en el ANEXO 1, y esta constituida por la siguiente instrucción SQL:

```
CREATE TABLE data
(
   x1 real,
   x2 real,
   x3 real,
   x4 real,
   x5 real,
   x6 integer,
   x7 smallint,
   y smallint,
   f  date,
   t varchar(6),
   id serial  primary key
);
```

Una vez creada la tabla que contendrá la información, se exportó el archivo de Excel de la base de datos a formato csv, y se cargo la información a la tabla con el siguiente comando de PostgreSQL:

```
\COPY data FROM data.csv USING DELIMITERS ‘,’ ;
```

Ya teniendo la información almacenada en la base de datos, se procedió a realizar el cálculo de la variable “Ciclos de vuelo transcurridos desde el último cambio de pieza” (codificada como X4) , basándonos en la información contenida en la variable “Ciclos de vuelo del avión” (codificada como X2), para lograr el cálculo de la variable X4 se debe tener en cuenta el identificador del avión (codificada como T) y la fecha de la observación (codificada como F), ya que esta información nos proporciona un ordenamiento de las observaciones con el

cual posteriormente podemos sacar la información que nos interesa. Además de lo anteriormente mencionado, para el cálculo de la variable X4 se debe tomar en cuenta los siguientes aspectos, los cuales fueron previamente analizados para realizar este cálculo.

En primer lugar, si la observación corresponde a la primera revisión del avión entonces el valor de la variable X4 es igual a cero.

Cuando la observación corresponde a un registro (el cual no se corresponde con la primera revisión del avión) que no ha presentado cambios de la pieza en revisiones anteriores, entonces el valor de la variable X4 es igual al valor de la variable X2 menos el valor de la variable X2 en la primera revisión del avión. El aspecto anterior puede calcularse de igual forma, considerando el hecho de que la primera vez que se reviso el avión no hubo cambio de pieza.

Por último, si la observación corresponde a un registro (el cual no se corresponde con la primera revisión del avión) que ha presentado cambios de la pieza en revisiones anteriores, entonces el valor de la variable X4 es igual al valor de la variable X2 menos el valor que tiene la variable X2 la ultima vez que se reviso el avión y presentó cambio de pieza.

Para facilitarnos el cálculo de la variable X4, nos auxiliamos de PL/pgSQL el cual es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL. Este lenguaje permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones, dando mucho más control que las sentencias SQL estándar y posibilitando realizar operaciones complejas sobre datos. Una discusión profunda sobre el lenguaje PL/pgSQL queda fuera del alcance de la presente tesis, sin embargo, para el lector interesado sugerimos la lectura de la siguiente pagina web:

http://www.postgresql.org/docs/8.2/interactive/plpgsql.html

Para realizar el calculo de la variable X4, primero creamos una función en PL/pgSQL que tomara como argumento el identificador del avión y que devolviera los ciclos volados la primera vez que se reviso el avión. Tal función se denominó “primerDatoX2” y esta

constituida por las siguientes instrucciones (se han puesto números de línea para identificar las partes clave del código de la función):

```
1  CREATE FUNCTION primerDatoX2(varchar) RETURNS real AS '
2  DECLARE
3  tail ALIAS FOR $1;
4  dato real;
5  BEGIN
6  SELECT  INTO dato x2 FROM data WHERE T = tail ORDER BY f ASC LIMIT 1;
7  return dato;
8  END;
9 ' LANGUAGE 'plpgsql';
```

En la línea 1 se define el nombre de la función , sus parámetros y el tipo de valor que retornara la función. La línea 2 es una palabra reservada del lenguaje la cual nos dice que las líneas siguientes contendrán declaraciones de variables. En la línea 3 creamos un alias para referirnos al primer parámetro de la función (Los parámetros pasados a la función son nombrados con los identificadores $1, $2, etc) . En la línea 4 declaramos la variable “dato” de tipo real (variable numérica con 6 dígitos decimales de precisión) que almacenará el resultado de los ciclos volados la primera vez que se revisó un avión determinado. Las líneas que van de la 5 a la 8 constituyen un bloque de instrucciones. La línea 6 es una instrucción especial que permite almacenar el resultado de la consulta “El valor de los ciclos volados la primera vez que se reviso el avión” en la variable “dato”, en la línea 7 se retorna el valor de la variable “dato”. La línea 9 es una instrucción especial que define el lenguaje que se utiliza.

El siguiente paso consistió en crear una función en PL/pgSQL que generara la variable X4 a partir de la información contenida en la variable X2 . Esta función usa la función primerDatoX2 y se muestra a continuación.

```
1  CREATE FUNCTION generarX4(varchar, date) RETURNS real AS'
2  DECLARE
3  tail ALIAS FOR $1;
4  fecha ALIAS FOR $2;
5  dato_anterior real;
6  dato_actual real;
7  BEGIN
8  SELECT INTO dato_actual x2 FROM data2 WHERE T = tail  AND  f = fecha;
9  SELECT INTO dato_anterior x2  FROM  data2  WHERE  T = tail  AND  f < fecha
10 AND  y="1"   order by f DESC LIMIT 1;
11 IF dato_anterior IS NULL THEN
12 SELECT INTO dato_anterior primerDatoX2(tail);
13 END IF;
14 RETURN dato_actual - dato_anterior;
15 END;
```

En la línea 1 definimos el nombre de la función, los argumentos de la función los cuales serán el identificador del avión y la fecha de revisión del avión y por último tipo de valor que devolverá la función. De la línea 2 a la 6 declaramos alias para cada uno de los parámetros de la función, además declaramos dos valores tipo real dato_anterior y dato_actual. En la línea 8 a la variable dato_actual le asignamos el valor de los ciclos volados en la fecha especificada por el alias “fecha” para un avión con identificador igual al alias “tail”, el cual representa el identificador del avión. En la línea 9 a la variable dato_anterior le asignamos el valor de los ciclos volados desde el último cambio de pieza. Esto lo hacemos con la siguiente instrucción SQL:

```
SELECT x2 FROM  data2  WHERE  T = tail  AND  f < fecha  AND  y="1" order by f
DESC  LIMIT 1;
```

Una vez obtenido este valor en la línea 11 se hace una comparación de igualdad del valor almacenado en dato_anterior con NULL (Una marca especial usada para indicar que el valor de un dato es desconocido en SQL), si el valor es diferente de NULL entonces pasa directamente a la línea 14, en cambio si el valor almacenado en la variable dato_anterior es NULL quiere decir que desde la primera vez que se revisó el avión no se ha cambiado la pieza, por lo tanto para obtener los ciclos volados desde el último cambio de pieza volvemos a calcular dato_anterior como el valor de los ciclos volados la primera vez que se reviso el avión. En la línea 14 retornamos la diferencia entre la variable dato_actual y dato_anterior el cual representa los ciclos volados desde el último cambio de pieza.

A continuación se procedió a generar la variable X4 con la siguiente instrucción SQL:

```
UPDATE data SET  x4 =  generarX4(x2, fecha) ;
```

Similar procedimiento se siguió para las variables “tiempo de vuelo transcurrido desde el último cambio de pieza” y “días transcurridos desde el último cambio de pieza”, para ver la información completa de las funciones utilizadas ver ANEXO 2.

3.3 SELECCIÓN DE VARIABLES RELEVANTES

Para seleccionar las variables a emplear en la red neuronal se decidió emplear dos estrategias: por un lado, se le consulto al Gerente de Control de Inventarios de AEROMAN conocedor del tema, y por otro, se llevo a cabo un análisis de correlación lineal sobre los datos, con el objetivo de determinar posible redundancia en los datos, la matriz de correlaciones lineales de las variables de entrada se muestra en la siguiente tabla.

Tabla 3.1: Matriz de correlaciones lineales para los datos de las variables de entrada del número de parte QA065423-01.

	X1	X2	X3	X4	X5	X6	X7
X1	1	0.989	1	0.350	0.376	0.338	-0.043
X2	0.989	1	0.989	0.378	0.382	0.350	-0.038
X3	1	0.989	1	0.350	0.376	0.338	-0.043
X4	0.350	0.378	0.350	1	0.977	0.984	0.030
X5	0.376	0.382	0.376	0.977	1	0.980	0.033
X6	0.338	0.354	0.338	0.984	0.980	1	0.058
X7	-0.043	-0.038	-0.043	0.030	0.033	0.058	1

La primera variable que excluimos de la investigación fue la edad del avión (codificada como X1) ya que esta variable se obtiene como el cociente entre el tiempo de vuelo del avión (codificada como X3) y el valor de 3600 horas voladas que representa una estimación del número de horas que un avión vuela en un año. Dado que la variable edad del avión es combinación lineal de las horas voladas su inclusión solo representaría agregar una dimensión más al modelo.

Otras variables que se excluyeron de la investigación fueron los ciclos de vuelo del avión (codificada como X2) y los ciclos de vuelo transcurridos desde el último cambio de pieza (codificada como X4), ya que presentan respectivamente alta correlación lineal (igual o superior a 0.977) con la variable tiempo de vuelo del avión (codificada como X3) y tiempo de vuelo transcurrido desde el último cambio de pieza (codificada como X5), por lo que la pérdida de información al excluir estas variables será mínima, además, se mantuvieron las variables relacionadas al tiempo de vuelo en vez de las variables relacionadas a los ciclos de vuelo ya que la pieza del avión que estamos analizando se mantiene en funcionamiento a medida que el avión se mantenga volando, por lo tanto, las variables relacionadas al tiempo de vuelo reflejan mucho mejor el problema que se esta planteando que las variables relacionadas a los ciclos de vuelo.

Por lo anteriormente mencionado la red neuronal a estimar constara primero de la variable dependiente: Demanda del número de parte QA065423-01 (se cambia la pieza, no se cambia la pieza) que es una variable dicotómica; y luego las variables independientes: tiempo de vuelo del avión, días transcurridos desde el último cambio de pieza, tiempo de vuelo transcurrido desde el último cambio de pieza y experiencia del mecánico.

3.4 ELECCIÓN DE LOS CONJUNTOS DE APRENDIZAJE Y PRUEBA

Dado que la cantidad de datos no es muy grande para poder dividir los datos en dos conjuntos de aprendizaje y prueba tendremos que usar remuestreo para la validación de los modelos, por lo tanto para la estimación y validación de los parámetros a estimar usaremos los datos comprendidos entre Marzo de 1999 y Diciembre de 2006.

Los datos que van de Enero a Febrero del 2007 se han reservado con el objetivo de pronosticar la demanda total del número de parte QA065423-01 en estos meses y evaluar, de manera objetiva, el desempeño de la herramienta informática que pronostica la demanda total del número de parte QA065423-01.

3.5 PROCESO DE DESARROLLO DE LOS PROGRAMAS

Antes de presentar los resultados de la investigación se tuvo que realizar ciertos pasos para la obtención de los resultados de la investigación. En primer lugar se trasladó la información contenida en la base de datos de PostgreSQL mencionada en la sección 3.2 a formato Matlab (.mat), esto se logro instalando el correspondiente driver ODBC para PostgreSQL y configurando un DSN para poder acceder a la información desde MATLAB. Las siguientes instrucciones muestran el código de una función en MATLAB que permite guardar la información de los datos que van a servir para estimar los parámetros de la red neuronal (conjunto de entrenamiento)

```
function import_data
conn = database('demanda_intermitente', 'admin', 'admin')
curs = exec(conn, ' SELECT x3,x5,x6,x7, y  FROM data WHERE f < '01/01/2007'')
setdbprefs('DataReturnFormat','numeric')
curs = fetch(curs)
data = curs.Data
save data data
close(curs)
close(conn)
```

Esta función de MATLAB guarda el conjunto de aprendizaje en el archivo data.mat, que luego usará el programa que estima y valida el modelo de redes neuronales.

El programa que estima y valida el modelo de redes neuronales guarda los resultados de la arquitectura de la red neuronal, el conjunto de pesos o parámetros de la red y la información del preprocesamiento en el archivo main_run.mat y luego esta información será almacenada en una base de datos de Access (ver ANEXO 3), para lograr ingresar la información a la base de datos de Access desde MATLAB se configuró un DSN para esta base de datos y luego se recurrió a la siguiente función en MATLAB.

```
function export_access
load main_run
conn = database('demanda_intermitente', '', '')

colnames = {'id', 'numero_parte_id',
'capa_entrada','capa_oculta','capa_salida'}
arquitectura = [1 1 arquitectura]
insert(conn, 'arquitecturas', colnames, arquitectura)

colnames = {'id','arquitectura_id','beta'}
[junk0 junk1] = size(beta)
junk2 = 1:junk1
junk2 = junk2'
junk3 = ones(junk1,1)
beta = beta'
beta = [junk2, junk3,beta]
insert(conn, 'betas', colnames, beta)

colnames = {'id','numero_parte_id',  'desviacion'};
[junk0 junk1] = size(stdp)
junk2 = 1:junk0
junk2 = junk2'
junk3 = ones(junk0,1)
stdp = [junk2, junk3,stdp]
insert(conn, 'desviaciones', colnames, stdp)

colnames = {'id','numero_parte_id','media'};
[junk0 junk1] = size(meanp)
junk2 = 1:junk0
junk2 = junk2'
junk3 = ones(junk0,1)
meanp = [junk2, junk3,meanp]

insert(conn, 'medias', colnames, meanp)
close(conn)
```

Después de haber logrado ingresar la información de la estimación de la red neuronal, esta será posteriormente utilizada por la herramienta informática que pronostica la demanda total del número de parte mencionado en la sección 1.5.5, la cual usará esta información para realizar pronósticos de la demanda total del número de parte QA065423-01.

Para definir la forma en la que la herramienta informática realizara los pronósticos se optó por realizarlo en una base de datos de Access (ver ANEXO 4) la cual esta constituida por la tablas datos y pronóstico. La primera de estas tablas se diseño para mantener la información

de los datos de los aviones que van a entrar a revisión, mientras que la segunda tabla almacena el pronostico de la demanda total. Por último se debe mencionar que para la generación de los gráficos que muestran el comportamiento de la demanda total y la demanda total pronosticada se recurrió a utilizar la librería grafica ZedGraph "la cual provee un alto grado de flexibilidad ya que casi cualquier aspecto del grafico puede ser modificado y además es fácil de usar"[24].

[24] http://zedgraph.org/wiki/index.php?title=Main_Page

CAPITULO IV: 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 pronóstico de la demanda del número de parte QA065423-01 sobre los datos históricos de esta demanda. Una vez que se determinó el número de capas ocultas óptimo y basándose en que se verificó un desempeño aceptable se procedió a realizar una herramienta informática que permitiera pronosticar la demanda total del número de parte QA065423-01, y se realizo el pronóstico de la demanda total en el periodo que va de enero a febrero del 2007 y luego se comparo los resultados obtenidos con el suavizado exponencial simple el cual es uno de los métodos tradicionales que se usan para pronosticar la demanda total del número de parte QA065423-01.

4.2 RENDIMIENTO DENTRO Y FUERA DE LA MUESTRA

Para el proceso de estimación y validación de la red neuronal se desarrolló un programa informático en MATLAB (ver ANEXO 5), que mide el rendimiento dentro y fuera de la muestra de un modelo de redes neuronales con respuesta dicotómica. El programa informático se ejecuto en una computadora personal con un procesador Athlon a 1.8 GHz y con 256 Megabytes en memoria RAM. Se debe mencionar que se tuvieron ciertos problemas con respecto al número de iteraciones necesario para lograr un rendimiento adecuado de los modelos. Se tuvo que ensayar con los siguientes números de iteraciones: 1000, 10000 y 100000 logrando con este último una tasa de error aceptable. Otro problema que se tuvo fue la lenta convergencia del programa, a pesar de que se utilizó el compilador de MATLAB el cual permite crear código en lenguaje C optimizado y por lo tanto muy eficiente, los tiempos de convergencia para medir el rendimiento dentro y fuera de la muestra fueron superiores a 4 horas. Los tiempos de ejecución del programa informático se pueden ver en la siguiente tabla.

Tabla 4.1: Tiempos de ejecución del programa informático que mide el rendimiento dentro y fuera de la muestra del modelo de redes neuronales con variable respuesta dicotómica.

Número de neuronas en la capa oculta	Tiempo de ejecución
2	4 horas 12 minutos
3	5 horas 37 minutos
4	6 horas 21 minutos
5	9 horas 19 minutos

Para medir el rendimiento dentro y fuera de la muestra se tuvo que estimar 5 modelos de redes neuronales cada uno de los cuales con las mismas variables de entrada o explicativas que se mencionan en la sección 3.3, pero con diferentes números de neuronas en la capa oculta. La siguiente tabla da información del rendimiento dentro de la muestra de los modelos estimados.

Tabla 4.2: Porcentajes de error dentro de la muestra.

Número de neuronas en la capa oculta	Falsos positivos (%)	Falsos negativos (%)	Error (%)
2	8.33	2.27	10.6
3	6.06	3.03	9.09
4	4.55	1.52	6.07
5	3.03	1.52	4.55

En la tabla mostrada anteriormente se observa que el modelo que tiene 5 neuronas en la capa oculta tiene el mejor rendimiento de todos los modelos con un error de clasificación del 4.55 por ciento, sin embargo, es necesario medir la capacidad de generalización de estos modelos, ya que a medida que el número de parámetros de una red neuronal aumenta decrece su capacidad de generalización, por lo tanto, se necesita medir la capacidad de generalización de estos modelos o rendimiento fuera de la muestra.

Para evaluar el rendimiento fuera de la muestra se uso el método “Bootstrap 0.632” descrito en la sección 2.5.2, a continuación se muestran los resultados obtenidos.

Tabla 4.3: Media de los porcentajes de error fuera de la muestra (“Bootstrap 0.632”)

Número de neuronas en la capa oculta	Falsos positivos (%)	Falsos negativos (%)	Error (%)
2	12.15	5.98	18.13
3	11.15	5.68	16.83
4	5.70	4.08	9.78
5	4.95	10.22	15.17

La tabla 4.3 da información acerca de la media de los porcentajes del error de clasificación de cada uno de los 5 modelos considerados. Se puede observar que el modelo con 4 neuronas en la capa oculta es el que tiene mejor rendimiento fuera de la muestra con un promedio de error de clasificación del 9.78 por ciento, por lo tanto la arquitectura óptima es de 4 neuronas en la capa oculta y este es el modelo que usaremos para realizar predicciones. Un hecho interesante es que la adición de otra neurona en la capa oculta mejora el rendimiento dentro de la muestra con respecto al modelo de 4 neuronas en la capa oculta, como se evidencia en la tabla 4.2, sin embargo, esto ocasiona un comportamiento diferente en el error fuera de la muestra, ya que el modelo de 4 neuronas es el que tiene un mejor rendimiento, por lo tanto el modelo de 5 neuronas en la capa oculta esta sobreajustado, lo cual esta de acorde a lo comentado en la sección 2.3.8. De lo anteriormente expuesto los resultados obtenidos son buenos, consiguiendo una clasificación de la demanda del número de parte QA065423-01 del 93.93 por ciento dentro de la muestra y del 90.22 por ciento fuera de la muestra. Para efectos de comprobación se muestran los parámetros de los modelos estimados en el ANEXO 6.

Además, de lo anteriormente expuesto se debe mencionar ciertas limitaciones del modelo. En primer lugar los parámetros del modelo no tienen interpretación y no se pueden establecer intervalos de confianza o pruebas estadísticas las cuales son habituales en los métodos estadísticos. Ni, por último obviar el elevado costo computacional requerido en el

entrenamiento y validación de las redes neuronales, muy superior al de los modelos estadísticos.

No obstante estas limitaciones el modelo se ve fortalecido por su mayor versatilidad de uso, al no depender su aplicabilidad del cumplimiento de supuestos teóricos y por lo tanto puede usarse en un sistema informático sin intervención humana.

4.3 PRONÓSTICO DE LA DEMANDA TOTAL

Después de haber realizado un análisis de las redes neuronales en cuanto a su estructura, es decir, encontrar el número de neuronas ocultas que tengan una mejor capacidad de "generalización", se implementó el pronosticador de demanda intermitente (PRODIN) el cual es una herramienta informática diseñada para obtener de manera anticipada información sobre los volúmenes de la demanda total del numero de parte QA065423-01 la cual se define en la sección 1.5.5 del primer capitulo. Para solucionar el problema de la demanda intermitente del numero de parte QA065423-01, PRODIN toma como base la carga de trabajo, es decir el numero de aviones que se revisan en un mes determinando en cada uno de los aviones que ingresan a revisión el cambio o no cambio de la pieza en cuestión, y calcula el total de piezas demanda para ese mes. La herramienta informática PRODIN se implementó en el lenguaje de programación Visual Basic .NET (ver ANEXO 7 para ver el código fuente).

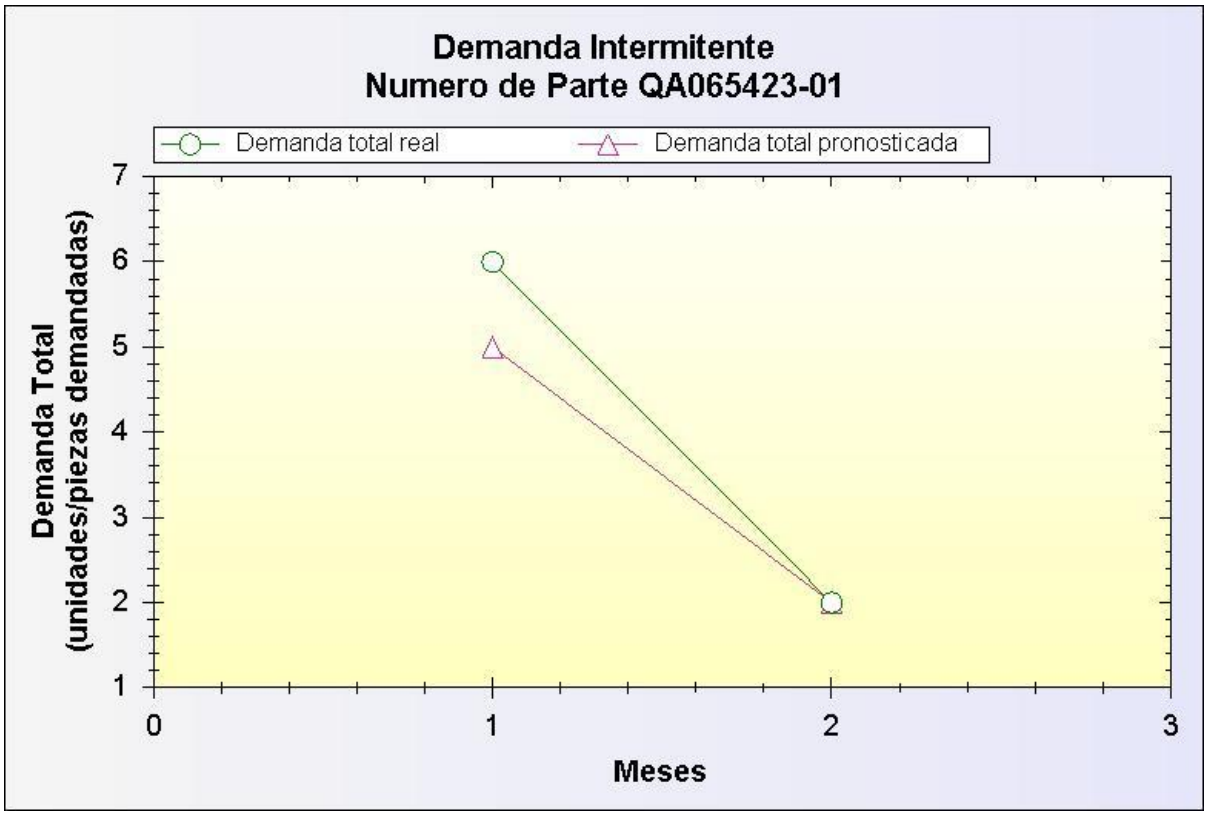
Para probar el rendimiento de PRODIN se procedió a realizar el pronóstico de la demanda total del número de parte QA065423-01 para el periodo que va de Enero a Febrero del 2007. Para establecer una medida de comparación se procedió a estimar un modelo de suavizado exponencial simple, el cual es uno de los modelos que se usan comúnmente para pronosticar la demanda total.

En las páginas siguientes se muestran los resultados obtenidos, así como la gráfica que muestra el comportamiento de la demanda total real y la demanda total pronosticada.

Tabla 4.4: Pronóstico de la demanda total del número de parte QA065423-01 para enero y febrero del 2007 mediante PRODIN.

Mes	Carga de trabajo	Demanda total observada	Demanda total predicha	Error (%)
Enero	6	6	5	16.67
Febrero	3	2	2	0
	MAPE(%)	8.33333333		

Figura 4.1: Gráfica de la demanda total y demanda total pronosticada del número de parte QA065423-01 para Enero y Febrero del 2007 mediante PRODIN.



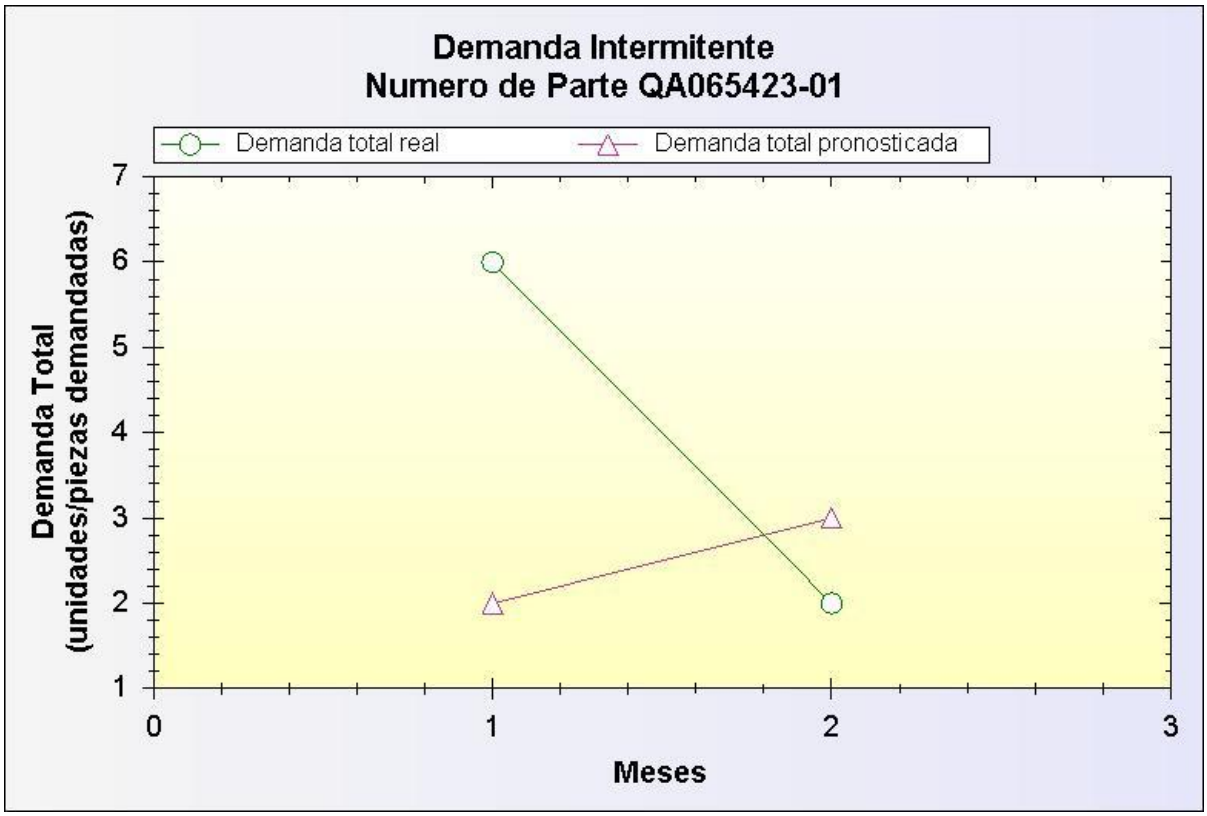
En la tabla 4.4 podemos observar que para Enero del 2007 hubo una demanda de 6 piezas de un total de 6 aviones que fueron a revisión, el pronóstico que da PRODIN es de 5 piezas lo cual es aceptable. Para Febrero del 2007 el pronóstico coincidió con la demanda observada.

Para establecer una medida de comparación se procedió a estimar un modelo de suavizado exponencial simple, el cual es uno de los modelos mas comúnmente usados para pronosticar la demanda total. A continuación se comentan los resultados obtenidos.

Tabla 4.5: Pronóstico de la demanda total del número de parte QA065423-01 para enero y febrero del 2007 mediante suavizado exponencial simple.

Mes	Carga de trabajo	Demanda total observada	Demanda total predicha	Error (%)
Enero	6	6	2	66.67
Febrero	3	2	3	50.0
	MAPE(%)	58.33333333		

Figura 4.2: Gráfica de la demanda total y demanda total pronosticada del número de parte QA065423-01 para Enero y Febrero del 2007 mediante suavizado exponencial simple.



Se observa en la tabla 4.5 que el modelo de suavizado exponencial simple presenta un mayor error cuando la demanda total se dispara (La mayor parte de las veces la demanda total del número de parte QA065423-01 oscila entre 0 y 2 piezas), es decir en Enero, este comportamiento se debe a que los modelos tradicionales de pronóstico de la demanda total, no toman en cuenta la carga de trabajo ya que para Enero del 2007 se tuvo una carga de trabajo constituida por 6 aviones que entraron a revisión, los cuales presentaron ciertas características que posibilitaron en mayor o menor medida el cambio de la pieza en estudio en cada uno de los aviones. Además, dado que el suavizado exponencial simple no requiere la especificación de estos factores, tiene una gran desventaja al tratar de hacer pronósticos cuando el comportamiento de la demanda total es poco sistemático. Por lo tanto, las predicciones realizadas por PRODIN resultan ser más fiables ya que toman en cuenta más factores.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

A continuación se presentan algunas conclusiones que pudieron inferirse de lo aprendido sobre las redes neuronales y principalmente de su aplicación al problema del pronóstico de la demanda intermitente del número de parte QA065423-01.

- El modelo de redes neuronales con 4 neuronas en la capa oculta permite relacionar la variable dependiente demanda del número de parte QA065423-01 y las variables independientes: tiempo de vuelo del avión, días transcurridos desde el último cambio de pieza, tiempo de vuelo transcurrido desde el último cambio de pieza y experiencia del mecánico.

- Existen ciertos problemas de la vida real que para su solución deben de descomponerse en partes, en cada una de estas partes se debe ensayar distintos métodos, estadísticos, matemáticos, informáticos, etc.

- En relación con la demanda total del número de parte QA065423-01 el modelo de suavizado exponencial simple con un MAPE del 58.33 por ciento, presenta errores de pronóstico muy superiores a los estimados por medio de PRODIN con un MAPE del 8.33 por ciento.

- Con estos resultados se amplia el grupo de metodologías disponibles para el pronóstico de la demanda total del número de parte QA065423-01.

- La técnica de las redes neuronales es una herramienta con un alto potencial en el pronóstico y modelación de variables dado que permite relaciones no lineales muy generales, las cuales podrían ser difíciles de ajustar usando otras técnicas.

- Se recomienda investigar el uso de los algoritmos genéticos para optimizar la arquitectura de la red neuronal, ya que actualmente es una línea de investigación que pretende sustituir al usual método de prueba y error.

- Se recomienda que se evalué la herramienta informática que pronostica la demanda total mes a mes para ver si se pueden obtener predicciones más certeras de la demanda total del número de parte QA065423-01.

- Se recomienda tomar en cuenta el elevado costo computacional de los modelos de redes neuronales, ya que puede requerir una gran inversión en recursos informáticos para una respuesta en un tiempo razonable.

- Sobre la base del modelizado matemático de la demanda del número de pieza QA065423-01 con redes neuronales, puede plantearse generalizar este modelo a otros números de piezas con la incorporación de más neuronas en la capa de salida, donde cada neurona en la capa de salida representaría la variable respuesta demanda de un número de pieza determinado.

- Los resultados obtenidos en la presente investigación pueden servir de base para futuras investigaciones relacionadas a la demanda total de otros números de piezas.

REFERENCIAS BIBLIOGRAFICAS

- PAUL D. McNELIS, Neural Networks in Finance, Editorial Elsevier, San Diego California 2005.
- BONIFACIO MARTÍN DEL BRÍO Y ALFREDO SANZ MOLINA, Redes Neuronales y Sistemas Borrosos, Editorial RA-MA, Universidad de Zaragoza 2001.
- ADEL A. GHOBBAR, CHRIS H. FRIEND, Evaluation of forecasting methods for intermittent parts demand in the field of aviation: a predictive model, Editorial Elsevier, University London, School of Engineering 2002.
- PEÑA DANIEL, Análisis de Datos Multivariantes, Mc Graw Hill, S.A. 2002.

ANEXO 1.

LISTADO DE VARIABLES Y SU CODIFICACIÓN

Nombre	Código	Tipo/Explicación	Mínimo	Máximo	Escala
Edad del avión	X1	Continua	0.7496	9.93262	Años
Ciclos de vuelo del avión	X2	Continua	1102	14966	Ciclos de vuelo
Tiempo de vuelo del avión	X3	Continua	2698.56	35757.4	Horas
Ciclos de vuelo transcurridos desde el último cambio de pieza	X4	Continua	0	4642	Ciclos de vuelo
Tiempo de vuelo transcurrido desde el último cambio de pieza	X5	Continua	0	10402.4	Horas
Días transcurridos desde el último cambio de pieza	X6	Continua	33	3420	Días
Experiencia del mecánico	X7	Ordinal, 1 a 3, 1 principiante, 2 intermedio, 3 experto	1	3	
Demanda	Y	Dicotómica, 0 a 1, 0 no se cambia la pieza, 1 se cambia la pieza	0	1	Piezas
Fecha	F	Fecha en que se reviso el avión			
Tail	T	Cadena, identificador del avión			

ANEXO 2.

FUNCIONES EN PL/pgSQL PARA GENERAR LAS VARIABLES REFERENTES AL ÚLTIMO CAMBIO DE PIEZA.

```
CREATE FUNCTION primerDatoX2(varchar) RETURNS real AS '
DECLARE
tail ALIAS FOR $1;
dato real;

BEGIN
SELECT  INTO dato x2 FROM data2 WHERE T = tail ORDER BY f ASC LIMIT 1;
return dato;
END;
' LANGUAGE 'plpgsql';


CREATE FUNCTION generarX4(varchar, date) RETURNS real AS'
DECLARE
tail ALIAS FOR $1;
fecha ALIAS FOR $2;
dato_anterior real;
dato_actual real;

BEGIN
SELECT INTO dato_actual x2 FROM data2 WHERE T = tail  AND  f = fecha;
SELECT INTO dato_anterior x2  FROM  data2  WHERE  T = tail  AND  f < fecha   AND
y="1"   order by f DESC LIMIT 1;
IF dato_anterior IS NULL THEN
SELECT INTO dato_anterior primerDatoX2(tail);
END IF;
RETURN dato_actual - dato_anterior;
```

```
END;

' LANGUAGE 'plpgsql';


CREATE FUNCTION primerDatoX3(varchar) RETURNS real AS '
DECLARE
tail ALIAS FOR $1;
dato real;

BEGIN
SELECT  INTO dato x3 FROM data2 WHERE T = tail ORDER BY f ASC LIMIT 1;
return dato;
END;

' LANGUAGE 'plpgsql';


CREATE FUNCTION generarX5(varchar, date) RETURNS real AS'
DECLARE
tail ALIAS FOR $1;
fecha ALIAS FOR $2;
dato_anterior real;
dato_actual real;

BEGIN
SELECT INTO dato_actual x3 FROM data2 WHERE T = tail  AND  f = fecha;
SELECT INTO dato_anterior x3  FROM  data2  WHERE  T = tail  AND  f < fecha   AND
y="1"   order by f DESC LIMIT 1;
IF dato_anterior IS NULL THEN
SELECT INTO dato_anterior primerDatoX3(tail);
END IF;
```

```
RETURN dato_actual - dato_anterior;
END;

' LANGUAGE 'plpgsql';


CREATE FUNCTION primerDatoX1(varchar) RETURNS real AS '
DECLARE
tail ALIAS FOR $1;
dato real;

BEGIN
SELECT  INTO dato round(x1*365) FROM data2 WHERE T = tail ORDER BY f ASC LIMIT 1;
return dato;
END;
' LANGUAGE 'plpgsql';


CREATE FUNCTION generarX6(varchar, date) RETURNS real AS'
DECLARE
tail ALIAS FOR $1;
fecha ALIAS FOR $2;
dato_anterior real;
dato_actual real;

BEGIN
SELECT INTO dato_actual round(x1*365) FROM data2 WHERE T = tail  AND  f = fecha;
SELECT INTO dato_anterior round(x1*365)  FROM  data2  WHERE  T = tail  AND  f < fecha  AND  y="1"  order by f DESC LIMIT 1;
IF dato_anterior IS NULL THEN
SELECT INTO dato_anterior primerDatoX3(tail);
```

```
RETURN dato_anterior;
ELSE
RETURN dato_actual - dato_anterior;
END IF;
END;

' LANGUAGE 'plpgsql';
```

ANEXO 3.

BASE DE DATOS EN ACCESS QUE GUARDA LA INFORMACIÓN DE LA ESTIMACIÓN DE UN MODELO DE REDES NEURONALES.

Tabla	arquitecturas	
Uso	Almacena la arquitectura de la red neuronal	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
numero_parte_id	Entero Largo	
capa_entrada	Entero	
capa_oculta	Entero	
capa_salida	Entero	

Tabla	betas	
Uso	Almacena los pesos o parámetros de la red neuronal	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
arquitectura_id	Entero Largo	
beta	Double	

Tabla	desviaciones	
Uso	Almacena las desviaciones de las variables de entrada	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
numero_parte_id	Entero Largo	
desviacion	Double	

Tabla	medias	
Uso	Almacena las medias de las variables de entrada	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
numero_parte_id	Entero Largo	
media	Double	

Tabla	numeros_partes	
Uso	Almacena la información de los números de partes	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
numero_parte	Texto con longitud de 11	

ANEXO 4.

BASE DE DATOS EN ACCESS QUE MANEJA LA INFORMACIÓN DEL PRONOSTICO DE LA DEMANDA TOTAL DEL NÚMERO DE PARTE QA065423-01.

Tabla	datos	
Uso	Almacena la información de los datos que se van a pronosticar	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
x3	Doble	
x5	Doble	
x6	Doble	
x7	Doble	
y	Doble	
fecha	Fecha Corta	

Tabla	pronostico	
Uso	Almacena la información de los pronósticos	
Campo	Tipo	Atributos
id	Autonumérico	Llave Primaria
x3	Doble	
x5	Doble	
x6	Doble	
x7	Doble	
y	Doble	
fecha	Fecha Corta	
mes	Entero largo	
anho	Entero largo	

ANEXO 5.

PROGRAMA EN MATLAB PARA MEDIR EL RENDIMIENTO DENTRO Y FUERA DE LA MUESTRA DEL MODELO DE REDES NEURONALES CON VARIABLE RESPUESTA DICOTÓMICA.

```
% Funcion main
% Uso: Función principal que mide el rendimiento dentro y fuera de la muestra de un
%      modelo de redes neuronales con variable respuesta dicotómica.
% Programada en: Matlab versión 7.0.
% Comentario: guarda en el archivo main_run.mat la siguiente información:
%           numero_parte: el número de parte.
%           meanp: las medias de cada una de las variables de entrada.
%           stdp: las desviaciones estándar de cada una de las variables de entrada.
%           arquitectura: la arquitectura de la red neuronal.
%           beta: el mejor vector de parámetros de la red neuronal.
%           RESS: rendimiento dentro de la muestra.
%           NETRES: rendimiento fuera de la muestra (bootstrap 0.632)
%           netresout_final_mean: media de los errores fuera de la muestra.
%           netresout_final_std: desviación estandar de los errores fuera de la muestra.

function main;
clear;
tic;
numero_parte = 'QA065423-01';
load data;

nneurons = 4;
[nrowdata ncoldata] = size(data);

% Etapa de preprocesamiento

data2 = data(:,1:(ncoldata-1));
[pn,meanp,stdp] = prestd(data2');
pn = pn';
[nrowpn ncolpn] = size(pn);
data = [pn data(:,ncoldata)];

% Fin de etapa de preprocesamiento

warning off;

% Etapa de Estimacion de los parametros de la red neuronal
maxgen1 = 100000;
ndraws = 100;
```

```
limit = .5;
[nrow1, ncol1] = size(data);
arquitectura = [ncol1-1 nneurons 1];

for j= 1:1,

[ netres,likelihood, beta] = ...
   classnet(data, ncol1, 1,.5, limit,[1 nneurons 0 0], 1, maxgen1, 0);

end

RESS(:,:,j) = [ netres ];

% Fin de etapa de estimacion de los parametros de la red neuronal

% Etapa de validacion de la red neuronal

for i = 1:ndraws,
   i
indexin = ceil(rand(nrow1,1) * nrow1);
indexin = sort(indexin);
for j=1:nrow1,
   if sum(ismember(indexin,j)) > 0,
      indexout(j,:) = NaN;
   else indexout(j,:) = j;
   end
end
indexout = excise(indexout);
data1in = data(indexin,:);
[rrin, ccin] = size(data1in);
dataout = data(indexout,:);
[rrout, ccout] = size(dataout);
percentin = rrin /(rrin + rrout);
mydatanew = [data1in; dataout];

[netres1 ] = classnet(mydatanew, ncol1, percentin,.5, limit,[1 nneurons 0 0], 1, maxgen1,
0,beta);

clear mydatanew data1in dataout;
NETRES(:,:,i) = netres1;

end;

for i = 1:ndraws,
netresout_final(i,:) = NETRES(2,:,i);
end
```

```
netresout_final_mean = mean(netresout_final);

netresout_final_std = std(netresout_final);

% Fin de etapa de validacion de la red neuronal


save main_run numero_parte meanp stdp arquitectura beta RESS  NETRES netresout_final_mean netresout_final_std ;
```

```
%Funcion classnet
%Uso: Realiza la estimación de un modelo de redes neuronales con respuesta binaria.
%
%Entradas: assetx: matriz de datos.
%              coldep: número de columna de la variable dependiente.
%              percent: porcentaje de datos para el error dentro de la muestra.
%              errweight: factor de error para falsos positivos y falsos negativos.
%              limit: punto de corte.
%               info: neuronas en la capa de entrada, oculta y salida.
%               gendum: tipo de optimización, 1 = algoritmos genéticos.
%               maxgen: máximo numero de generaciones para el algoritmo genético.
%               helge: funcion de escalamiento, 0 = sin escalamiento.
function [ NETRES,  ...
     LIKELIHOOD, beta] =...
   classnet(assetx,coldep,percent,errweight,limit,info,gendum,maxgen, helge, beta0init);
global NEPOCH;

global P T nlayer nneuron1 nneuron2 nneuron3;
nntwarn off;
warning off MATLAB:divideByZero
fun = 'classnetfun';
nlayer = info(1);
nneuron1 = info(2);
nneuron2 = info(3);
nneuron3 = info(4);
popsize = 100; pc = .9; pdes = 0; toler = .000001; elite = 1;
[rr cc] = size(assetx);
y = assetx(:,coldep);
if coldep == 1, x = assetx(:, 2:end,:);
else x = [assetx(:,1:coldep-1) assetx(:, coldep+1:end)];
end


[rp, cp] = size(x);
yxmat = [y x];
   [nrow ncol] = size(x);
   [nrowy ncoly] = size(y);
nrow1 = round(percent * nrow);
nrow11 = nrow1 + 1; [nrow12 nrow13] = size(x(1:nrow1,:));

yy = y(1:nrow1,:); xx = [x(1:nrow1,:)];
[nrow ncol] = size(x);


nrow1 = round(percent * nrow);
nrow11 = nrow1 + 1; [nrow12 nrow13] = size(x(1:nrow1,:));

yy = y(1:nrow1,:); xx = x(1:nrow1,:);
```

```
smin = .1;
smax = .9;
[rx, cx] = size(x);
[ry, cy] = size(y);
maxy = max(y);
miny = min(y);
maxx = max(x);
minx = min(x);

if helge == 0, PN = x; TN = y;
end


global P T;
P = PN(1:nrow1,:);
T = TN(1:nrow1,:);
[rp, cp] = size(P);
nparm = nneuron1 * cp + nneuron1 + (nneuron1+1);
nparm1 = cp + 1;
maxgen1 = maxgen + 25;
nepoch =  NEPOCH;
tp = [25, nepoch, .02, .01, 1.07, .7, .9, 1.04];
pm = 1/nparm; elite = 1;
pdes = 0; scale = .1;


if nargin > 9, beta0 = beta0init;
else beta0 = randn(1, nparm) * .1; end;

if gendum >= 1,
   beta = genf(fun,beta0,popsize,maxgen);
   else beta = beta0
end


[LIK3,sse3,A3] = feval(fun,beta);


if helge == 0, A3n = A3;
end

A3real = A3n;
A3 = A3>limit;
err3 = A3 - T;
err3a = err3>0;
err3b = err3<0;
err3ap = sum(err3a) / nrow;
```

```
err3bp = sum(err3b) / nrow;
err3p =  errweight * err3ap + (1-errweight) * err3bp;
netres_in = [err3ap err3bp err3p];



if percent < 1,
xout  = [x(nrow11:nrow,:)];
yout = y(nrow11:nrow,:);
[n1 c1] = size(yout);

T1 = y(nrow1+1:end,:);
clear global P T;
global P T;
T = TN(nrow1+1:end,:);
P = PN(nrow1+1:end,:);
[junk,junk,A31] = feval(fun,beta);

if helge == 0, A31n = A31;
   end
A31 = A31n;
A31real = A31;
A31 = A31>limit;
err31 = A31 - T1;
err31a = err31>0;
err31b = err31<0;
err31ap = sum(err31a) / n1;
err31bp = sum(err31b) / n1;
err31p =  errweight * err31ap + (1-errweight) * err31bp;
netres_out = [err31ap err31bp err31p];

else
  netres_out = ones(1,3);

end

NETRES = [netres_in; netres_out];

LIKELIHOOD = [LIK3];
```

```
%Funcion classnetfun
%Uso: Calcula la salida de la red neuronal.
%
%Entrada: beta: parámetros de la red neuronal.
%
%Salidas: LIK: valor de la función de error (entropía)
%             sse3: error cuadrático medio.
%             A3: salida de la red neuronal.
function [LIK,sse3,A3] = classnetfun(beta);
global P T  nneuron1;
[rt ct] = size(T);
rt = min([rt ct]);
[rp cp] = size(P);
x = P;
for j = 1:nneuron1,
nn1(:,j) = x * beta(1+(j-1)*cp:j*cp)' + beta(nneuron1*cp+j);
end;
for j = 1:nneuron1,
   nn2(:,j) = 1 ./(1 + exp(-nn1(:,j)));
end

yhat =   nn2(:,1:nneuron1) * beta(nneuron1*cp+nneuron1+1:nneuron1*cp+2*nneuron1)'
...
   + beta(nneuron1*cp+2*nneuron1+1);

yhat = 1 ./  (1 + exp(-yhat));


T = T;
TT = T;
y = T;
A3 = yhat;
lik = y .* log(yhat) + (1-y) .* log(1-yhat);
LIK = -sum(lik);

sse3 = (TT-yhat)' * (TT-yhat);
g = [];
xmean = mean(P);



BETA = [beta(nneuron1*cp+nneuron1+1:nneuron1*cp+2*nneuron1) ];
GAMMA = beta(1:nneuron1*cp);

GAMMA1 = reshape(GAMMA,cp,nneuron1);
```

```
for j = 1:nneuron1,
   nn1mean(:,j) = xmean * beta(1+(j-1)*cp:j*cp)' + beta(nneuron1*cp+j);
end;



for j = 1:nneuron1,
   nn2mean(:,j) = 1 ./(1 + exp(-nn1mean(:,j)));
end

nn3mean =  nn2mean(:,1:nneuron1) *
beta(nneuron1*cp+nneuron1+1:nneuron1*cp+2*nneuron1)' ...
   + beta(nneuron1*cp+2*nneuron1+1);

nn3mean = 1 ./  (1 + exp(-nn3mean));

for i = 1:cp,
   for j = 1:nneuron1,
      junk(j,i) = BETA(j)* nn2mean(j) * (1-nn2mean(j)) * GAMMA1(i,j);
   end


end
```

```
%Funcion genf
%Uso: Realiza la optimización de una función utilizando un algoritmo genético.
%
%Entradas: fun: función a optimizar.
%              beta0: parametros iniciales.
%              popsize: tamaño de población.
%              maxgen: número de generaciones.
%
%Salidas: controlg: parámetros que minimizan la función.
%           sumutilb: minimo valor de la función.
function [CONTROLB, sumutilb] = genf(fun, beta0, popsize, maxgen)


if nargin < 4; error(' must have fun, beta0, popsize maxgen');end,
global ELITE CURVAU TOLER PMSTART PDES EXPONB
global SCALE PSHUFFLE PLINEAR PMEND PCONTROL INTERVISION
global CONTROLB

if isempty(INTERVISION),    INTERVISION= 1;       end
if isempty(PMSTART),        PMSTART = .35;        end
if isempty(PCONTROL),       PCONTROL= .95;        end
if isempty(PMEND),          PMEND   = .15;        end
if isempty(PLINEAR),        PLINEAR = .33;        end
if isempty(PSHUFFLE),       PSHUFFLE= .33;        end
if isempty(SCALE),          SCALE   = 1;          end
if isempty(EXPONB),         EXPONB  = 2;          end
if isempty(PDES),           PDES    = .050;       end
if isempty(TOLER),          TOLER   = .001;       end
if isempty(CURVAU),         CURVAU  = .75;        end
if isempty(ELITE),          ELITE   = 1;          end
[junk nparm] = size(beta0);
if junk ~=1; error(' beta0 is not a column vector');end
www(1,:)= beta0;
lwww=popsize:-1:1;    lwww=lwww.^CURVAU;     swww=lwww/sum(lwww);
for i=2:popsize,                swww(i)=sum(swww(i-1:i));          end
rand('state',exp(abs(randn*randn)));
for repl = 1:maxgen,

  if (rand < PDES | repl==1)
    if repl==1
      popsize=max(popsize,2*nparm);

      www = [www(1,:); (randn(popsize-1,nparm)).* SCALE];
      for i = 1:popsize,
        sse100(i)   =  feval(fun,www(i,:));
      end
    else
```

```
   if ELITE==0;
     www=[CONTROLB;www];
   end
   www = [www(1:nparm,:); (randn(popsize-nparm,nparm)).* SCALE];
   for i = nparm+1:popsize,
    sse100(i)   =  feval(fun,www(i,:));
   end
  end
  [sse100 inds] =sort(sse100);  www=www(inds,:);
  sumutilb=sse100(1);
  CONTROLB=www(1,:);
end;

maxg=rand/EXPONB/repl;

for ii = 1:2:popsize,
  for i=1:2;              nnr=rand;
    [junk ind]=min(abs(swww-nnr));
    if (swww(ind)-nnr)<0; ind=ind+1;end,   nnrow(i)=ind;
  end

  p1  = nnrow(1);          p2         = nnrow(2);

  if rand         <= PCONTROL,       randy       = rand;
    if randy      < PSHUFFLE,   rrr         = rand(1,nparm) > .5;
      child1      =www(p1,:);   child2      = www(p2,:);
      child1(rrr) =www(p2,rrr);       child2(rrr) = www(p1,rrr);
    elseif randy < (PSHUFFLE + PLINEAR),      aaa = rand(1,nparm);
      child1 = aaa .* www(p1,:) + (1-aaa) .* www(p2,:);
      child2 = aaa .* www(p2,:) + (1-aaa) .* www(p1,:);
    else
      cutpoint  = round(rand * (nparm-2))+1 ;
      child1    = [www(p1,1:cutpoint) www(p2, cutpoint+1:nparm)];
      child2    = [www(p2,1:cutpoint) www(p1, cutpoint+1:nparm)];
    end
  else, child1  = www(p1,:);    child2      = www(p2,:);
  end


  pm      = PMEND + (PMSTART - PMEND) / repl;

  rr1     = [rand(2,nparm) <= pm] .* ([(rand(2,nparm) > 0.5)] * 2 - 1);

  s       = rand(2,nparm) .* (1-(rand(2,nparm).^maxg));

  child1  = child1 +rr1(1,:) .* s(1,:);     schild1 = feval(fun,child1);
  child2  = child2 +rr1(1,:) .* s(2,:);     schild2 = feval(fun,child2);
  nwww(ii+[0 1],:)  = [child1; child2];
```

```
  nsse(ii+[0 1])    = [schild1; schild2];
 end
 if ELITE==0
  [sse100, ind] = sort(nsse);
  sse1001=sse100(1);
  if sse100(1)<sumutilb;
    CONTROLB  = nwww(ind(1),:);
    sumutilb  = sse100(1);
  end
  www       = nwww(ind,:);
  ssemean   = mean(sse100);
 else
  [nsse ind]=sort(nsse);
  nwww=nwww(ind,:);
  nwww=[www(1,:);nwww(1:end-1,:)];
  nsse=[sse100(1),nsse(1:end-1)];
  [sse100, ind] = sort(nsse);
  CONTROLB  = nwww(ind(1),:);
  sumutilb  = sse100(1);
  www       = nwww(ind,:);
  ssemean   = mean(sse100);
  sse1001   =sse100(1);
  sumutilb =sse100(1);
 end



 if INTERVISION==1;
    disp([toc/60 repl maxgen sse1001 sumutilb ]);
 end

 save GENJUNK CONTROLB nwww maxgen popsize nsse sse100;

 if sumutilb<TOLER
  disp('success toler');
  return
 end
 clear rrr1 child1 child2 schild1 schild2 wwwc ssec ssez indc nwww
 clear ssemean nnrow p1 p2 pp randy aaa cutpoint nsse nsse1 s
end
```

```
%Funcion excise
%Uso:  Remueve filas con datos perdidos.
%   EXCISE(X) remueve filas de X conteniendo algun NaNs.
%   EXCISE(X, CODE) usa el valor de CODE en vez de NaNs.
function x=excise(x,miss_code);
if nargin==1,
  miss=isnan(x);
else
  miss=(x==miss_code);
end;

[m n]=size(x);
if n==1,
  x(miss) = [];
else
  x(any(miss'),:) = [];
end;
```

ANEXO 6.

PARÁMETROS DE LOS MODELOS DE REDES NEURONALES.

Modelo con 2 neuronas ocultas

Parámetro	Valor
$w_{1,1}$	2.29012281011784
$w_{1,2}$	-1.8611496594617
$w_{1,3}$	7.86703266299345
$w_{1,4}$	-23.0398657960857
$w_{2,1}$	-0.68328378456071
$w_{2,2}$	0.313748818462841
$w_{2,3}$	0.331543192105526
$w_{2,4}$	22.3181602351461
$w_{1,0}$	-8.72501869768793
$w_{2,0}$	-4.41633295443546
γ_1	5.48910237677857
γ_2	39.2721371348807
γ_0	-3.17302870082049

Modelo con 3 neuronas ocultas

Parámetro	Valor
$w_{1,1}$	-0.0352105293281677
$w_{1,2}$	1.17503604298584
$w_{1,3}$	-1.07294220462862
$w_{1,4}$	5.17569471714441
$w_{2,1}$	-13.1632099757062
$w_{2,2}$	1.30804932437534
$w_{2,3}$	6.75357613655888
$w_{2,4}$	9.65601902584348
$w_{3,1}$	-0.952998417500212
$w_{3,2}$	5.6340202604483
$w_{3,3}$	-0.769211657447561
$w_{3,4}$	14.6370264462726
$w_{1,0}$	-0.243625500910412
$w_{2,0}$	-8.23103798239221
$w_{3,0}$	-6.73429511369271
γ_1	-7.54153451239536
γ_2	31.5872044805018
γ_3	10.3673778599604
γ_0	2.25904932115354

Modelo con 4 neuronas ocultas

Parámetro	Valor
$w_{1,1}$	-3.96870814238585
$w_{1,2}$	-12.0019007025732
$w_{1,3}$	6.89802756163829
$w_{1,4}$	-26.2949776394079
$w_{2,1}$	0.310840522054578
$w_{2,2}$	12.5801490423833
$w_{2,3}$	-0.156311847249546
$w_{2,4}$	28.6150487360666
$w_{3,1}$	0.855541134013066
$w_{3,2}$	0.00728498846821931
$w_{3,3}$	-0.429357585477035
$w_{3,4}$	-0.974980007753189
$w_{4,1}$	-26.9617747672662
$w_{4,2}$	14.0229143478905
$w_{4,3}$	12.8359975604298
$w_{4,4}$	15.7302436039273
$w_{1,0}$	3.80306982524111
$w_{2,0}$	-10.2020514392211
$w_{3,0}$	0.410128025781856
$w_{4,0}$	-7.92752328893966
γ_1	21.5595783731863
γ_2	38.5914176340145
γ_3	16.0786883036992

γ_4	15.3435460513003
γ_0	-31.2759132955402

Modelo con 5 neuronas ocultas

Parámetro	Valor
$w_{1,1}$	1.64075712376112
$w_{1,2}$	-4.96139524440879
$w_{1,3}$	-0.28366701077631
$w_{1,4}$	13.984068985773
$w_{2,1}$	0.0226338532954697
$w_{2,2}$	-2.21103705993635
$w_{2,3}$	12.7174946380706
$w_{2,4}$	-8.14263453236582
$w_{3,1}$	1.01786280360695
$w_{3,2}$	4.91718651542136
$w_{3,3}$	-4.67464533413493
$w_{3,4}$	12.3305498205992
$w_{4,1}$	28.2035156018341
$w_{4,2}$	0.32854247790526
$w_{4,3}$	-7.47143338081725
$w_{4,4}$	-12.3196460222105
$w_{5,1}$	4.49867944673388
$w_{5,2}$	-1.57586478347431
$w_{5,3}$	-5.65559253626169
$w_{5,4}$	12.1631050204776

$w_{1,0}$	8.07855810149844
$w_{2,0}$	9.90497016814322
$w_{3,0}$	-9.84007426842669
$w_{4,0}$	15.5578903145302
$w_{5,0}$	5.15849066032335
γ_1	-16.1992798534583
γ_2	10.5613259015842
γ_3	26.3150294459059
γ_4	-14.2488517329602
γ_5	14.0818877754087
γ_0	6.18974145349867

ANEXO 7.

CODIGO FUENTE DE PRODIN.

```
'-------------------------------------------------------------------------
' Archivo       : Form1.vb
' Programada en : Visual Basic 2005
' Descripción   : Formulario principal
' Version       : 1.0
' Uso         : Pronostica la demanda total del número de parte QA065423-01


Imports System
Imports System.Data
Imports System.Data.OleDb
Imports System.IO
Imports ZedGraph




Public Class Form1
    Inherits System.Windows.Forms.Form

#Region "Private Declarations"
    Dim myForm2 As Form2
    Private oConexion1 As New OleDbConnection
    Friend WithEvents Button2 As System.Windows.Forms.Button
    Private oConexion2 As New OleDbConnection




#End Region



#Region " Código generado por el Diseñador de Windows Forms "

    Public Sub New()
        MyBase.New()

        'El Diseñador de Windows Forms requiere esta llamada.
        InitializeComponent()

        'Agregar cualquier inicialización después de la llamada a
InitializeComponent()

    End Sub

    'Form reemplaza a Dispose para limpiar la lista de componentes.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
```

```
        MyBase.Dispose(disposing)
    End Sub

    'Requerido por el Diseñador de Windows Forms
    Private components As System.ComponentModel.IContainer

    'NOTA: el Diseñador de Windows Forms requiere el siguiente
procedimiento
    'Puede modificarse utilizando el Diseñador de Windows Forms.
    'No lo modifique con el editor de código.
    Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem2 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem4 As System.Windows.Forms.MenuItem
    Friend WithEvents Button1 As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Me.MainMenu1 = New System.Windows.Forms.MainMenu(Me.components)
        Me.MenuItem1 = New System.Windows.Forms.MenuItem
        Me.MenuItem2 = New System.Windows.Forms.MenuItem
        Me.MenuItem3 = New System.Windows.Forms.MenuItem
        Me.MenuItem4 = New System.Windows.Forms.MenuItem
        Me.Button1 = New System.Windows.Forms.Button
        Me.Button2 = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'MainMenu1
        '
        Me.MainMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.MenuItem1})
        '
        'MenuItem1
        '
        Me.MenuItem1.Index = 0
        Me.MenuItem1.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.MenuItem2, Me.MenuItem3, Me.MenuItem4})
        Me.MenuItem1.Text = "&Archivo"
        '
        'MenuItem2
        '
        Me.MenuItem2.Index = 0
        Me.MenuItem2.Text = "Abrir red neuronal"
        '
        'MenuItem3
        '
        Me.MenuItem3.Enabled = False
        Me.MenuItem3.Index = 1
        Me.MenuItem3.Text = "Abrir datos a pronosticar"
        '
        'MenuItem4
        '
        Me.MenuItem4.Index = 2
        Me.MenuItem4.Text = "Salir"
        '
        'Button1
        '
```

```
        Me.Button1.Location = New System.Drawing.Point(72, 64)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(136, 40)
        Me.Button1.TabIndex = 0
        Me.Button1.Text = "Pronosticar"
        Me.Button1.Visible = False
        '
        'Button2
        '
        Me.Button2.Location = New System.Drawing.Point(72, 132)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New System.Drawing.Size(136, 36)
        Me.Button2.TabIndex = 1
        Me.Button2.Text = "Graficar"
        Me.Button2.UseVisualStyleBackColor = True
        Me.Button2.Visible = False
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(292, 266)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.Button1)
        Me.Menu = Me.MainMenu1
        Me.Name = "Form1"
        Me.Text = "PRODIN"
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub MenuItem2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem2.Click

        menuOpenClick1()

    End Sub

#Region "Menu Open1"
    ' Menu Open Click routine!!
    Public Sub menuOpenClick1()
        Dim openFile As New OpenFileDialog
        openFile.FileName = ""
        'Make sure only *.mdb files can be opened
        'by using a filter
        openFile.Filter = "Microsoft Access Application (*.mdb)|*.mdb"

        Dim res As System.Windows.Forms.DialogResult =
openFile.ShowDialog()
        If res = System.Windows.Forms.DialogResult.Cancel Then
            Return
        End If
        'Change the mouse icon and caption
        'of the form to inform the user that
        'the data is being loaded
        Me.Cursor = Cursors.WaitCursor
        Me.Text = "Cargando Datos Por favor Espere..."
```

```
        Try
            'The connection parameters
            oConexion1.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + openFile.FileName '
            oConexion1.Open()
            oConexion1.Close()

        Catch oExcep As OleDbException
            MessageBox.Show("Error al conectar con datos" & _
    ControlChars.CrLf & _
    oExcep.Message & ControlChars.CrLf & _
    oExcep.Source())
        End Try

        Me.Cursor = Cursors.Default
        Me.Text = "PRODIN"
        Me.MenuItem2.Enabled = False
        Me.MenuItem3.Enabled = True

    End Sub 'menuOpenClick

#End Region


#Region "Menu Open2"
    ' Menu Open Click routine!!
    Public Sub menuOpenClick2()
        Dim openFile As New OpenFileDialog
        openFile.FileName = ""
        'Make sure only *.mdb files can be opened
        'by using a filter
        openFile.Filter = "Microsoft Access Application (*.mdb)|*.mdb"

        Dim res As System.Windows.Forms.DialogResult =
openFile.ShowDialog()
        If res = System.Windows.Forms.DialogResult.Cancel Then
            Return
        End If
        'Change the mouse icon and caption
        'of the form to inform the user that
        'the data is being loaded
        Me.Cursor = Cursors.WaitCursor
        Me.Text = "Cargando Datos Por favor Espere..."

        Try
            'The connection parameters
            oConexion2.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + openFile.FileName '
            oConexion2.Open()
            'MessageBox.Show("Conectado !!")
            oConexion2.Close()
            'MessageBox.Show("Desconectado !!")

        Catch oExcep As OleDbException
            MessageBox.Show("Error al conectar con datos" & _
    ControlChars.CrLf & _
    oExcep.Message & ControlChars.CrLf & _
```

```
    oExcep.Source())
        End Try

        Me.Cursor = Cursors.Default
        Me.Text = "PRODIN"
        Me.Button1.Visible = True
        Me.MenuItem3.Enabled = False



    End Sub 'menuOpenClick

#End Region

#Region "Sub Forecast"
    Public Sub Forecast()

        oConexion1.Open()
        Dim oComando1 As New OleDbCommand("SELECT * FROM arquitecturas",
oConexion1)
        Dim oComando2 As New OleDbCommand("SELECT * FROM betas",
oConexion1)
        Dim oComando3 As New OleDbCommand("SELECT * FROM desviaciones",
oConexion1)
        Dim oComando4 As New OleDbCommand("SELECT * FROM medias",
oConexion1)
        Dim oComando5 As New OleDbCommand("SELECT * FROM salidas",
oConexion1)
        Dim oComando6 As New OleDbCommand("SELECT * FROM entradas",
oConexion1)

        Dim oComando7 As New OleDbCommand("SELECT count(*) FROM betas",
oConexion1)
        Dim oComando8 As New OleDbCommand("SELECT count(*) FROM
desviaciones", oConexion1)
        Dim oComando9 As New OleDbCommand("SELECT count(*) FROM medias",
oConexion1)
        Dim oComando10 As New OleDbCommand("SELECT count(*) FROM entradas",
oConexion1)



        oConexion2.Open()
        Dim oComando11 As New OleDbCommand("SELECT count(*) FROM datos",
oConexion2)
        'Dim oComando12 As New OleDbCommand("SELECT * FROM datos",
oConexion2)
        Dim oComando12 As New OleDbCommand("SELECT id,
horas_voladas,horas_voladas_ultimo_cambio, dias_ultimo_cambio,
experiencia_mecanico, demanda, fecha FROM datos", oConexion2)


        Dim oDataReader1 As OleDbDataReader
        'Dim oDataReader2 As OleDbDataReader

        Dim num_betas As Integer
        Dim num_desviaciones As Integer
```

```
Dim num_medias As Integer
Dim num_variables_entrada As Integer
Dim num_data As Integer


num_betas = oComando7.ExecuteScalar
num_desviaciones = oComando8.ExecuteScalar
num_medias = oComando9.ExecuteScalar
num_variables_entrada = oComando10.ExecuteScalar
num_data = oComando11.ExecuteScalar



Dim arquitectura(2) As Short
oDataReader1 = oComando1.ExecuteReader

Dim i As Short = 0
Dim j As Short = 0
Dim k As Short = 0


While oDataReader1.Read
    arquitectura(0) = oDataReader1.GetInt16(2)
    arquitectura(1) = oDataReader1.GetInt16(3)
    arquitectura(2) = oDataReader1.GetInt16(4)
End While

oDataReader1.Close()



Dim betas(num_betas - 1) As Double

oDataReader1 = oComando2.ExecuteReader

While oDataReader1.Read
    betas(i) = oDataReader1.GetDouble(2)
    i = i + 1
End While

i = 0
oDataReader1.Close()

Dim desviaciones(num_desviaciones - 1) As Double

oDataReader1 = oComando3.ExecuteReader

While oDataReader1.Read
    desviaciones(i) = oDataReader1.GetDouble(2)
    i = i + 1
End While

i = 0
oDataReader1.Close()

Dim medias(num_medias - 1) As Double

oDataReader1 = oComando4.ExecuteReader
```

```
While oDataReader1.Read
    medias(i) = oDataReader1.GetDouble(2)
    i = i + 1
End While

i = 0
oDataReader1.Close()

Dim variable_salida As String

oDataReader1 = oComando5.ExecuteReader

While oDataReader1.Read
    variable_salida = oDataReader1.GetString(1)
End While

i = 0
oDataReader1.Close()

Dim variables_entrada(num_variables_entrada - 1) As String

oDataReader1 = oComando6.ExecuteReader

While oDataReader1.Read
    variables_entrada(i) = oDataReader1.GetString(1)
    i = i + 1
End While

i = 0
oDataReader1.Close()



' Almacenamos el id

Dim input(num_variables_entrada - 1) As Double
Dim id(num_data - 1) As Integer
Dim demanda(num_data - 1) As Double
Dim fechas(num_data - 1) As Date
Dim output(num_data - 1) As Integer



oDataReader1 = oComando12.ExecuteReader

Dim nn1(arquitectura(1) - 1) As Double
Dim nn2(arquitectura(1) - 1) As Double
Dim producto As Double
Dim suma As Double
Dim yhat As Double

producto = 1
suma = 0
j = 0
```

```
        While oDataReader1.Read


            id(k) = oDataReader1.GetInt32(0)
            demanda(k) = oDataReader1.GetDouble(num_variables_entrada + 1)
            fechas(k) = oDataReader1.GetDateTime(num_variables_entrada + 2)

            For i = 0 To num_variables_entrada - 1
                input(i) = oDataReader1.GetDouble(i + 1)
            Next

            ' Procesamiento

            For i = 0 To num_variables_entrada - 1
                input(i) = (input(i) - medias(i)) / desviaciones(i)
            Next


            ' Calculo de la salida de la red neuronal




            For j = 0 To arquitectura(1) - 1

                For i = 0 To num_variables_entrada - 1

                    producto = input(i) * betas((1 + (j + 1 - 1) *
num_variables_entrada) - 1 + i)

                    suma = producto + suma
                Next




                suma = suma + betas((arquitectura(1) *
num_variables_entrada + j + 1) - 1)

                nn1(j) = suma

                suma = 0
                producto = 1
            Next

            suma = 0
            producto = 1


            For i = 0 To arquitectura(1) - 1
                nn2(i) = 1 / (1 + Math.Exp(-nn1(i)))
            Next

            For i = 0 To arquitectura(1) - 1


                producto = nn2(i) * betas((arquitectura(1) *
num_variables_entrada + arquitectura(1) + 1) - 1 + i)
```

```
                suma = producto + suma

            Next



            yhat = suma + betas(arquitectura(1) * num_variables_entrada + 2
* arquitectura(1) + 1 - 1)

            yhat = 1 / (1 + Math.Exp(-yhat))

            suma = 0
            producto = 1


            If yhat > 0.5 Then
                output(k) = 1

            Else
                output(k) = 0
            End If

            k = k + 1

        End While

        oDataReader1.Close()

        Dim oComando15 As New OleDbCommand("DELETE FROM pronostico",
oConexion2)
        oComando15.ExecuteScalar()



        Dim sSQL As String = "INSERT INTO pronostico
(id,demanda,demanda_pronosticada,fecha) VALUES(?,?,?,?)"

        Dim oComando13 As New OleDbCommand(sSQL, oConexion2)

        oComando13.Parameters.Add(New OleDbParameter("@id",
OleDbType.Integer))
        oComando13.Parameters.Add(New OleDbParameter("@demanda",
OleDbType.Double))
        oComando13.Parameters.Add(New
OleDbParameter("@demanda_pronosticada", OleDbType.Double))
        oComando13.Parameters.Add(New OleDbParameter("@fecha",
OleDbType.Date))



        For i = 0 To (num_data - 1)

            oComando13.Parameters("@id").Value = id(i)
            oComando13.Parameters("@demanda").Value = demanda(i)
```

```
            oComando13.Parameters("@demanda_pronosticada").Value =
output(i)
            oComando13.Parameters("@fecha").Value = fechas(i)
            oComando13.ExecuteNonQuery()

        Next


        Dim oComando16 As New OleDbCommand("UPDATE pronostico SET mes =
Month(fecha), anho = Year(fecha) ", oConexion2)

        oComando16.ExecuteScalar()


        Dim oComando17 As New OleDbCommand("SELECT sum(demanda),
sum(demanda_pronosticada),mes,anho FROM pronostico GROUP BY mes, anho ORDER
BY anho,mes ASC ", oConexion2)


        oDataReader1 = oComando17.ExecuteReader

        k = 0

        Dim list1 As New PointPairList()
        Dim list3 As New PointPairList()



        While oDataReader1.Read
            k = k + 1
            list1.Add(k, oDataReader1.GetDouble(0))
            list3.Add(k, oDataReader1.GetDouble(1))



        End While


        oDataReader1.Close()

        Dim x As New Xml.Serialization.XmlSerializer(list1.GetType)
        Dim y As New Xml.Serialization.XmlSerializer(list3.GetType)

        Dim stream1 As Stream = New FileStream("x.xml", FileMode.Create,
FileAccess.Write, FileShare.Read)
        Dim stream2 As Stream = New FileStream("y.xml", FileMode.Create,
FileAccess.Write, FileShare.Read)



        x.Serialize(stream1, list1)
        stream1.Close()

        x.Serialize(stream2, list3)
        stream2.Close()
```

```
        oConexion1.Close()
        oConexion2.Close()

        MessageBox.Show("Pronostico Generado !!")

        Me.Button1.Visible = False


        Me.Button2.Location = Me.Button1.Location
        Me.Button2.Visible = True


    End Sub


#End Region



    Private Sub MenuItem4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem4.Click
        Application.Exit()


    End Sub

    Private Sub MenuItem3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem3.Click
        menuOpenClick2()

    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem1.Click

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Forecast()

    End Sub



    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        myForm2 = New Form2()
        myForm2.Show()
```

```
    End Sub
End Class

'-------------------------------------------------------------------------
' Archivo     : Form2.vb
' Descripcion : Reporte
' Version     : 1.0



Imports ZedGraph
Imports System.IO
Imports System.Data


Public Class Form2

    Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        CreateGraph(zg1)


    End Sub
#Region "Sub CreateGraph"
    ' Call this method from the Form_Load method, passing your
ZedGraphControl
    Private Sub CreateGraph(ByVal zgc As ZedGraphControl)
        Dim myPane As GraphPane = zgc.GraphPane

        ' Set the titles and axis labels
        myPane.Title.Text = "Demanda Intermitente" & Chr(10) & "Numero de
Parte QA065423-01"
        myPane.XAxis.Title.Text = "Meses"
        myPane.YAxis.Title.Text = "Demanda Total" & Chr(10) &
"(unidades/piezas demandadas)"

        Dim stream1 As Stream = New FileStream("x.xml", FileMode.Open)


        Dim list1 As New PointPairList()
        Dim mySerializer As New
Xml.Serialization.XmlSerializer(GetType(PointPairList))
        list1 = CType( _
        mySerializer.Deserialize(stream1), PointPairList)
        stream1.Close()


        Dim curve As LineItem = myPane.AddCurve("Demanda total real",
list1, Color.Green, SymbolType.Circle)
        curve.Line.Width = 1.5F
        curve.Line.Fill = New Fill(Color.White, Color.FromArgb(60, 190,
50), 90.0F)
        curve.Line.IsSmooth = True
        curve.Line.SmoothTension = 0.6F
        curve.Symbol.Fill = New Fill(Color.White)
        curve.Symbol.Size = 10
```

```
        Dim stream2 As Stream = New FileStream("y.xml", FileMode.Open)

        Dim list3 As New PointPairList()
        list3 = CType( _
        mySerializer.Deserialize(stream2), PointPairList)
        stream2.Close()


        curve = myPane.AddCurve("Demanda total pronosticada", list3,
Color.FromArgb(200, 55, 135), SymbolType.Triangle)
        curve.Line.Width = 1.5F
        curve.Line.Fill = New Fill(Color.White, Color.FromArgb(160, 230,
145, 205), 90.0F)
        curve.Symbol.Fill = New Fill(Color.White)
        curve.Symbol.Size = 10

         myPane.Fill = New Fill(Color.WhiteSmoke, Color.Lavender, 0.0F)
         myPane.Chart.Fill = New Fill(Color.FromArgb(255, 255, 245), _
              Color.FromArgb(255, 255, 190), 90.0F)


        myPane.BarSettings.ClusterScaleWidth = 10

        ' Bars are stacked
        myPane.BarSettings.Type = BarType.Stack

        ' Enable the X and Y axis grids
        myPane.XAxis.MajorGrid.IsVisible = True
        myPane.YAxis.MajorGrid.IsVisible = True

        ' Manually set the scale maximums according to user preference
         myPane.XAxis.Scale.Max = 13



        myPane.YAxis.Scale.Max = 7


         ' Add a BoxObj to show a colored band behind the graph data
        Dim box As New BoxObj(0, 110, 1200, 10, _
           Color.Empty, Color.FromArgb(225, 245, 225))
        box.Location.CoordinateFrame = CoordType.AxisXYScale
        ' Align the left-top of the box to (0, 110)
        box.Location.AlignH = AlignH.Left
        box.Location.AlignV = AlignV.Top
        ' place the box behind the axis items, so the grid is drawn on top
of it
        'box.ZOrder = ZOrder.E_BehindAxis
        myPane.GraphObjList.Add(box)

         ' Calculate the Axis Scale Ranges
        zgc.AxisChange()
    End Sub


#End Region
```

```
    ' On resize action, resize the ZedGraphControl to fill most of the
Form, with a small
    ' margin around the outside
    Private Sub Form1_Resize(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Resize
        SetSize()
    End Sub

    Private Sub SetSize()
        Dim loc As New Point(10, 10)
        zg1.Location = loc
        ' Leave a small margin around the outside of the control
        Dim size As New Size(Me.ClientRectangle.Width - 20,
Me.ClientRectangle.Height - 20)
        zg1.Size = size
    End Sub

    'Display customized tooltips when the mouse hovers over a point
    Private Function MyPointValueEvent(ByVal control As ZedGraphControl, _
            ByVal pane As GraphPane, ByVal curve As CurveItem, _
            ByVal iPt As Integer) As String Handles zg1.PointValueEvent

        ' Get the PointPair that is under the mouse
        Dim pt As PointPair = curve(iPt)

        Return curve.Label.Text + " is " + pt.Y.ToString("f2") + " units at
" + pt.X.ToString("f1") + " days"
    End Function

    ' Customize the context menu by adding a new item to the end of the
menu
    Private Sub MyContextMenuBuilder(ByVal control As ZedGraphControl, _
               ByVal menu As ContextMenuStrip, ByVal mousePt As Point, _
               ByVal objState As ZedGraphControl.ContextMenuObjectState) _
            Handles zg1.ContextMenuBuilder
        Dim item As New ToolStripMenuItem
        item.Name = "add-beta"
        item.Tag = "add-beta"
        item.Text = "Add a new Beta Point"
        AddHandler item.Click, AddressOf Me.AddBetaPoint

        menu.Items.Add(item)
    End Sub

    ' Handle the "Add New Beta Point" context menu item.  This finds the
curve with
    ' the CurveItem.Label = "Beta", and adds a new point to it.
    Private Sub AddBetaPoint(ByVal sender As Object, ByVal args As
EventArgs)
        ' Get a reference to the "Beta" curve PointPairList
        Dim x As Double, y As Double

        Dim ip As IPointListEdit = zg1.GraphPane.CurveList("Beta").Points

        If (Not IsNothing(ip)) Then
            x = ip.Count * 5.0
            y = Math.Sin(ip.Count * Math.PI / 15.0) * 16.0 * 13.5
            ip.Add(x, y)
```

```
            zg1.AxisChange()
            zg1.Refresh()
        End If
    End Sub

    Private Sub zg1_ZoomEvent(ByVal control As ZedGraphControl, ByVal
oldState As ZoomState, _
            ByVal newState As ZoomState) Handles zg1.ZoomEvent
        'Here we get notification everytime the user zooms
    End Sub
End Class
```