ISSN 0188-7297

INSTITUTO MEXICANO DEL TRANSPORTE

SCT
SECRETARÍA DE COMUNICACIONES Y TRANSPORTES

Certificación ISO 9001:2008 ‡

RED NEURONAL *FEEDFORWARD* COMO ESTIMADOR DE PATRONES DE CORRIENTES EN EL INTERIOR DEL PUERTO DE MANZANILLO SUJETO A LA ACCIÓN DE TSUNAMIS

Juan Pedro Vásquez López

Publicación Técnica No. 406
Sanfandila, Qro, 2014

SECRETARÍA DE COMUNICACIONES Y TRANSPORTES

INSTITUTO MEXICANO DEL TRANSPORTE

RED NEURONAL *FEEDFORWARD* COMO ESTIMADOR DE PATRONES DE CORRIENTES EN EL INTERIOR DEL PUERTO DE MANZANILLO SUJETO A LA ACCIÓN DE TSUNAMIS

Publicación Técnica No.406
Sanfandila, Qro, 2014

Esta investigación fue realizada en la Coordinación de Ingeniería Portuaria Y Sistemas Geoespaciales del Instituto Mexicano del Transporte, por el M. en C. Juan Pedro Vásquez López.

Se agradece la colaboración del Ing. José Miguel Montoya, Jefe de la División de Ingeniería de Puertos y Costas del IMT, quien proporcionó los datos para este estudio.

Contenido

Resumen		iv
Abstract		vi
Resumen Ejecutivo		viii
Introducción.		1
Capítulo 1.	Metodología	5
Capítulo 2.	Datos	11
Capítulo 3.	Estrategias de cómputo	15
Capítulo 4.	Búsqueda de la RNAf óptima	25
Capítulo 5.	Resultados	31
Capítulo 6.	Conclusiones	39
Bibliografía		41
Anexo 1	Estado del arte	45
Anexo 2	Coordenadas UTM de los puntos de modelación	49

Resumen

En el presente estudio se elaboró un breve estado del arte que guardan las redes neuronales artificiales *feedforward* en el campo de la ingeniería portuaria e hidráulica marítima, en lo relativo a los Tsunamis y,específicamente, a los efectos que se originan en los puertos.

Se exploró la posibilidad de aplicar esta novedosa metodología para conseguir un estimador de patrones de corrientes litorales en el interior del Puerto de Manzanillo en Colima, sujeto a la acción de tsunamis.

Se logró el diseño, codificación y optimización de dos redes neuronales artificiales que aprendieron una el patrón de rapidez máxima y otra el patrón de rapidez media. Con un error porcentual absoluto medio menor al 19% en ambos casos.

Abstract

Present study begins with a brief state of the art keept by*feedforward* artificial neural networks in the field of maritime port and hydraulic engineering, with regard to Tsunamis and specifically to the effects that originate into ports.

Also we explored the possibility of applying this new methodology to get an estimate of patterns of coastlines currents in the interior of the port of Manzanillo in Colima, subject to the action of tsunamis.

Achieving the design, coding and optimization of two artificial neural networks, one learned the maximum speed pattern and another for the pattern of average speed. With one absolute percentage error means less than 19% in both cases.

Resumen ejecutivo

El presente trabajo comenzó por hacer una breve investigación bibliográfica en el campo de la ingeniería portuaria e hidráulica marítima en lo relativo a la aplicación de redes neuronales artificiales. Hallando que, si bien se reportan muy ilustrativas aplicaciones de esta novedosa metodología de la inteligencia artificial a diversos retos ingenieriles del ramo, nada se halló para tsunamis en cuanto sus efectos en el interior de los puertos. Este trabajo promete,entonces, ser de los pioneros en tan particular tema.

Las redes neuronales artificiales (RNA) son algoritmos computacionales que han ganado popularidad en la ciencia y tecnología, donde los métodos matemáticos y numéricos tradicionales hallaban serias dificultades en problemas que involucraban muchas variables y de naturaleza no linealy,en el más extremo de los casos, cuando no existía un modelo previo que describiera las relaciones funcionales entre dichas variables.

Estas RNA son sistemas de procesamiento cuyo funcionamiento simula el de un cerebro biológico real. Tienen la capacidad de procesar en paralelo grandes cantidades de información, aun cuando ésta sea parcial y difusa. Pueden aprender y memorizar información muy variada y formalizarla y, desde luego, hacer predicciones a partir de los datos con los que ha sido entrenada.

Así, proveen de una poderosa herramienta de interpolación no lineal y multidimensional. Por lo que se les ha empleado principalmente en identificación y predicción de patrones.

Las RNA de topología *feedforward* se caracterizan por ejecutar el procesamiento de datos en una sola dirección. Distinguiéndose tres capas principales de unidades independientes de cómputo llamadas neuronas: la capa de entrada, donde se reciben los datos a procesar; la capa intermedia, donde se da el procesamiento propiamente dicho y la capa de salida. Las redes neuronales artificiales tipo *feedforwad* (RNAf) son relativamente fáciles de programar, por lo que son muy populares entre la comunidad científica.

Los datos proporcionados para el presente estudio proceden de un estudio previo realizado en la misma división del IMT, con un modelo hidráulico a escala. Contándose con un conjunto de 21 casos de elevación inicial de tsunami local y sus respectivas corrientes litorales generadas en 423 puntos de modelación en el interior del Puerto de Manzanillo, Col.

Ante tan incipiente cantidad de datos y elevada complejidad, se decidió tomar los 21 casos de elevación inicial de tsunami local pero considerar la rapidez máxima y la rapidez media de las corrientes litorales en sólo 50 puntos de modelación en el interior del puerto y proponer sendas RNAf.

Para hallar la RNAf óptima que aprenda los patrones de rapidez máxima y media, hace falta tomar en cuenta todas las posibles combinaciones de parámetros propios de la red neuronal (como número de neuronas en la capa

intermedia, tasa de aprendizaje, etc.). Enfrentándose a, cuando menos, 2^{14} ejecuciones de algoritmo. Para completarlas todas y hallar la combinación que haga que la respuesta de la RNAf tenga el mínimo error porcentual absoluto medio (MAPE) respecto de los datos de validación, o sea, la RNAf óptima para cada patrón propuesto, se implementaron estrategias de cómputo paralelo y de registro y seguimiento de la ejecución de algoritmos largos en una base de datos relacional.

La RNAf óptima, en ambos casos de patrones, tiene cinco neuronas de entrada cuyas variables son referentes al punto de modelación, sin variables sintéticas adicionales, a saber: número del punto de modelación, elevación inicial, latitud, longitud y profundidad del punto de modelación. Con una sola variable de salida: la rapidez máxima en cada punto de modelación para una RNAf y la rapidez media para la otra. Ambas también con 14 neuronas en la capa intermedia, es decir, 2.8 veces el número de neuronas en la capa de entrada. En ambos casos la mejor convergencia se logró al tomar 21 casos dato para entrenamiento y 1 para predicción y validación de la respuesta de la red.

Hallándose que el aprendizaje del patrón para ambas RNAf se dio con una precisión de la predicción y un error porcentual absoluto medio menor al 19%.

Futuros desarrollos pueden buscar abarcar la complejidad completa del problema o incluir la evolución temporal del tsunami; lo que evidentemente requerirá de un mayor número de casos dato.

Finalmente, ese trabajo tomó ventaja de la plataforma estadística R que, respecto de los lenguajes de programación tradicionales, mostró prestaciones muy superiores y simplicidad de codificación.

Introducción

Los tsunamis son series de ondas oceánicas largas, las cuales se caracterizan por tener una longitud de onda mucho mayor que su altura y que se forman al ocurrir alguna alteración de corta duración, pero de gran extensión, en su superficie libre. Tienen diversas causas, pudiéndose clasificar en naturales, como los sismos grandes pero con poca profundidad focal; los deslizamientos del suelo submarino y las erupciones volcánicas e impactos de meteoritos; o bien, generadas por el hombre, como son las explosiones nucleares.

En aguas profundas la longitud de estas ondas alcanzan cientos de metros y su amplitud es del orden de apenas un metro. Pero su rapidez alcanza cientos de kilómetros por hora. A medida que el tsunami se acerca a la costa, su velocidad disminuye, pero ya que es un solitón, su disipación es relativamente poca, y por conservación energética, su altura se ve incrementada. En algunos casos alcanzando decenas de metros [5].

Por lo que la actividad comercial, industrial y turística que se desarrolla en zonas costeras, particularmente en los puertos, se ve amenazada por tan grande peligro. De ahí el capital interés de su estudio y modelación. La altura que alcanzan los tsunamis en la costa depende de las características de sus olas en mar abierto, pero también de las características de la zona en tierra donde llega, por lo que se deben considerar: la resonancia, difracción y refracción, junto con la pendiente y la configuración batimétrica de la costa.

En México, los estados de Colima, Guerrero, Oaxaca y Michoacán son los que han presentado mayor incidencia de tsunamis locales a lo largo de la historia reciente [7].

El recuento documental de afectaciones por tsunamis en el país comprende casos a lo largo de la Costa Occidental desde el año 1787 al 2003. Teniéndose verificados 60 tsunamis en 250 años.

Se distinguen dos tipos: los de origen lejano, como por ejemplo los recientes tsunamis originados en Sumatra en el año 2004, Chile 2010 y Japón 2011; cuyas consecuencias incluyen alturas de ola de un máximo de 2.5 m. Y, por otra parte, los tsunamis de origen cercano, que han llegado a tener alturas de ola de 5 y 10 metros; como los que se registraron en las costas de Oaxaca en 1787 y en el estado de Guerrero en 1925.

Los tsunamis ocurridos en México han tenido una mediana de 2m de altura de ola, pero algunos casos extremos llegan a superar la barrera de los 10 m. y,por lo

general, son generados por sismos grandes, mayores de 7.0, en particular, de mediana 7.7 grados en la escala sísmica de Richter.

Por otra parte, la ciencia y la tecnología se están enfrentando a problemas de complejidad cada vez más profunda y que se dan cada vez con mayor frecuencia. Entonces se crean nuevos desarrollos teóricos y tecnologías computacionales que coadyuven en la solución de tales retos.

Recientemente, los métodos computacionales de inteligencia artificial han ganado terreno en donde los métodos matemáticos y numéricos tradicionales hallaban serias dificultades para resolver problemas que involucraban muchas variables y eran de naturaleza no lineal, o estaban centrados en el procesamiento de una enorme cantidad y variedad de datos y, en el caso más extremo, donde se carecía de un modelo matemático previo que describiera la esencia del problema mismo [16].

Uno de dichos métodos son las redes neuronales artificiales (RNA). Son sistemas de procesamiento de datos e información modelados en similitud del funcionamiento de un cerebro biológico real.

Su mayor mérito es la habilidad de procesar grandes volúmenes de información, parcial y difusa, cuya relación funcional no es del todo clara. Pues, por ejemplo, experiencias tan cotidianas para el cerebro humano como el reconocimiento de voz o de grafías, son epítomes de la complejidad para la programación tradicional de sistemas informáticos. A la vez que soluciones notables logradas mediante este novedoso método.

Una RNA es capaz de aprender y ajustarse a su ambiente informático exterior. El entrenamiento de dicha red es el proceso de aprendizaje, el cual se lleva a cabo mediante el procesamiento repetido de ejemplos específicos conocidos y validados. Dándole la capacidad de aprender y memorizar grandes cantidades de información muy variada y formalizarla. Adicionalmente, es capaz de hacer predicciones a partir de los datos con los que ha sido entrenada. Tal como el cerebro biológico aprende a base de experiencias repetidas y hace inferencias a partir de ellas.

Internamente, las RNA se componen de algoritmos que representan unidades simples de cómputo llamadas neuronas, interconectadas en una topología dada, y el masivo procesamiento paralelo de información que llevan a cabo emula el funcionamiento de una red neuronal biológica real, dando lugar a procesos de aprendizaje y veloz síntesis de información. Sus salidas tienen las ventajas de no estar sujetas a un modelo matemático específico, ser tolerantes a fallas localizadas de los datos de entrada y no necesitan de condiciones externas a los datos mismos [8, 14].

Así, son capaces de desarrollar relaciones funcionales entre datos y proveer una poderosa herramienta de interpolación no lineal y multidimensional; por lo que se

les ha empleado principalmente en la resolución de problemas que involucran tareas de predicción e identificación de patrones. Sus aplicaciones se extienden a diversos campos de la actividad humana, como son: Econometría, Electrónica, Física, Medicina, Automatización, por citar algunos [17].

En el campo de la ingeniería marítima y de costas, se reconoce que las RNA se han aplicado exitosamente en problemas que involucran la estimación o predicción de parámetros ambientales, cargas estructurales y comportamiento de estructuras; llevando a cabo tareas de aproximación de funciones, optimización, modelado de sistemas y aprendizaje de patrones. También en predicción de niveles de marea, olas costeras, olas de viento, predicción de la licuefacción del suelo marino inducido por el oleaje, predicción de tormentas y morfología de costas [6, 11].

Finalmente, se debe hacer énfasis en el trabajo realizado por la Dra. Barbara Zanuttigh, de la Universidad de Boloña. Como extensión del desarrollo de la famosa herramienta para el cálculo de *overtopping* de estructuras *CLASH-project*, implementó una solución de RNA para la predicción de coeficientes de reflexión de olas para una gran variedad de estructuras de protección. Entrenó y validó la red con una gigantesca base de datos con 6×10^3 casos registrados. Logrando que la red haga predicciones con una asombrosa precisión de apenas 0.04 de rmse [19].

1 Metodología

Ante tan prometedor panorama, se planteó la necesidad de revisar el estado del arte que guardan las redes neuronales artificiales *feedforward* en el campo de la ingeniería portuaria e hidráulica marítima, en lo relativo a los Tsunamis y específicamente a los efectos que se originan en los puertos. Además, explorar la posibilidad de aplicar esta novedosa metodología de las RNA en conseguir un estimador de patrones de corrientes litorales en el interior de un puerto sujeto a la acción de un tsunami, en este caso el del Puerto de Manzanillo, en el estado mexicano de Colima.

1.1 Las RNA tipo *feedforward*

Profundizando en el ámbito de las RNA, si se utiliza una de especial arquitectura llamada*feedforward*, el Dr. George Cybenko [4] demostró que con una configuración de dos capas y un número suficiente de neuronas, se puede aproximar cualquier función continua a un grado de precisión arbitrario. Por lo tanto, no sorprende ver a lo largo de la literatura que el tipo de red neuronal artificial más utilizado sea el RNAf.

Es de notarse que, si bien se han buscado en la inteligencia artificial nuevos enfoques para lidiar con los problemas de predicción e identificación de patrones, los profesores Warner y Misra [18] han demostrado también que hay bastante similitud en los fundamentos sobre los que operan el análisis generalizado de regresión y las RNA del tipo multicapa f*eedforward* (RNAf).

Una RNAf se caracteriza por ser un conjunto de neuronas que reciben información multivariable, la procesan y dan una respuesta que puede ser multivariable también. En la arquitectura *feedforward* la topología del arreglo de neuronas y sus interconexiones hace fluir la información de forma unidireccional para que nunca pueda pasar más de una vez a través de una neurona antes de generarse la respuesta de salida. Tal como muestra la figura 1.1.

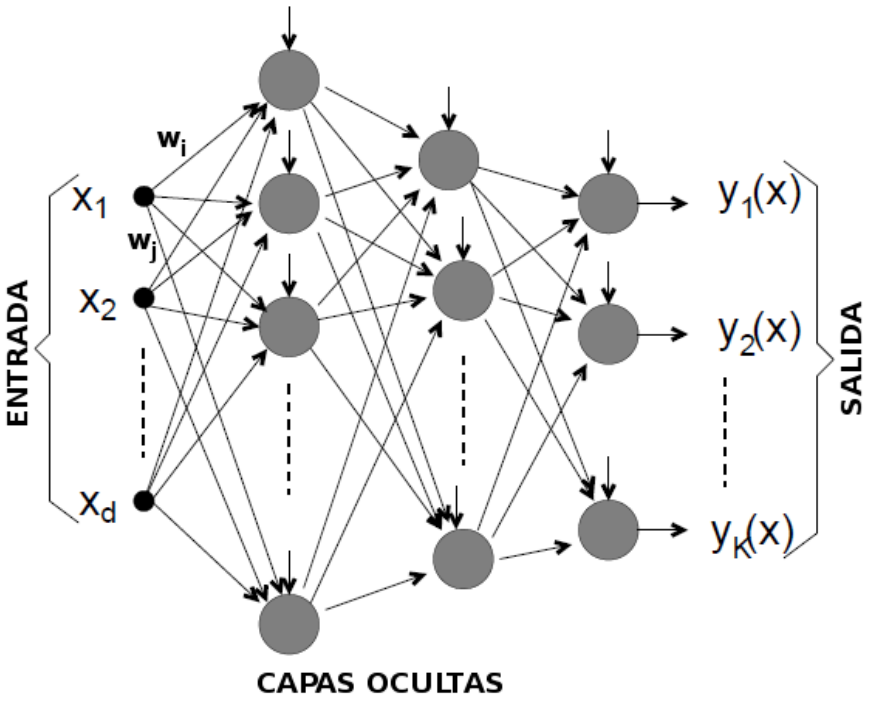


Figura 1.1 Diagrama de la estructura típica de una red neuronal artificial tipo *feedforward* (RNAf).

A muy grandes rasgos, el trabajo de la red es ingresar los datos a través de las neuronas de la capa de entrada, y al pasar la información a la siguiente capa, llamada oculta, cada una de las neuronas receptoras recibe la suma ponderada de todas las entradas conectadas a ella. Pues cada conexión entre neuronas representa un peso de conexión. Lo que matemáticamente se puede resumir como:

$$\sum_j w_{ij} x_j$$

Donde w_{ij} son los pesos de cada conexión de las x_j neuronas a la neurona receptora. Pero, al igual que sucede con las neuronas biológicas reales, a menos que el estímulo tenga cierta magnitud, la información se procesa, de lo contrario se ignora. En las RNAf se logra mediante un valor umbral m_i, y a menos que la suma ponderada supere un valor dado, se considera la salida efectiva de la información procesada. Nuevamente, se puede expresar como:

$$\sum_j w_{ij} x_j - m_i$$

El cerebro biológico aprende mediante la modificación de la intensidad de las conexiones sinápticas mismas. En los algoritmos se emula añadiendo también una función de activación. Se han considerado muchas a lo largo de la literatura; sin embargo, la más famosa, junto con la tangente hiperbólica, es la función sigmoidal que tiene la forma:

$$g: \mathbb{R} \to (0,1) \mid g(entrada) = \frac{1}{1 + e^{-entrada}}$$

Y sus propiedades se muestran en la figura 1.2.

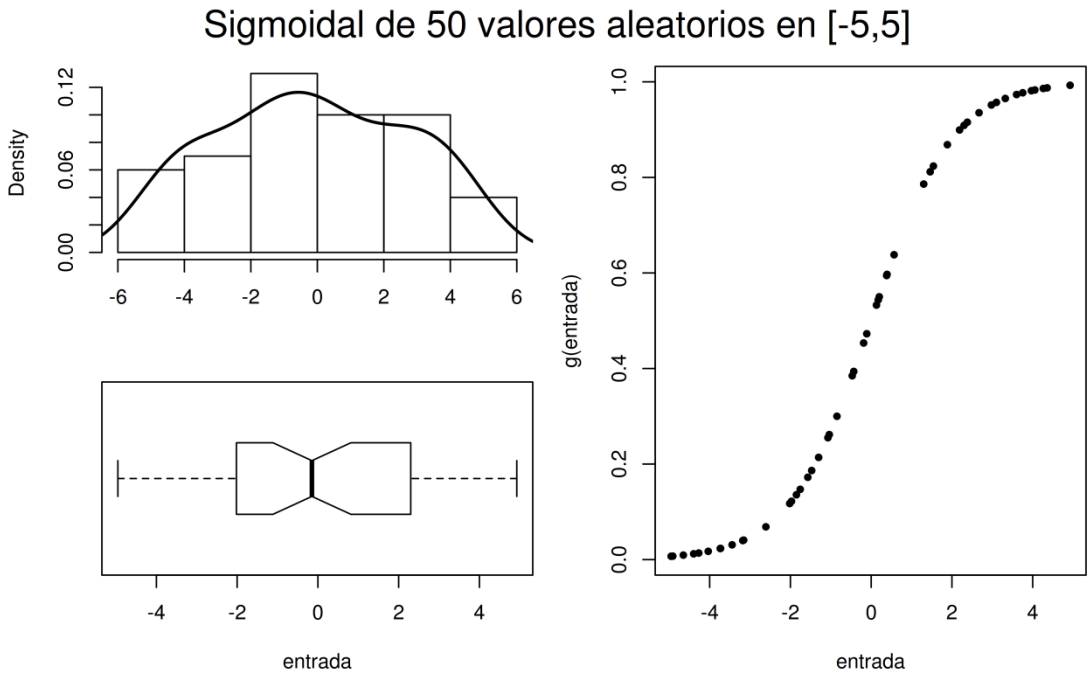


Figura 1.2. Sigmoidal como función de activación.

El proceso de aprendizaje de la neurona puede ser del tipo conocido como **entrenamiento supervisado**. Esencialmente es un ciclo donde se cuenta con un conjunto de valores objetivo el cual sirve de patrón de comparación para la salida que arroja la RNAf. La diferencia entre éstas dos se utiliza para modificar, en la próxima iteración, los valores peso en las interconexiones de las neuronas de la red, a fin de minimizar dicha diferencia (minimizar el error).

Diversos algoritmos se utilizan para el aprendizaje de las RNAf. Sin embargo, el más popular es el de propagación reversa (*backpropagation algorithm*), y se fundamenta en una estrategia simple para reducir el error cuadrático medio, similar al método generalizado de regresión.

1.2 La plataforma estadística R

Históricamente programar una RNAf representaba un gran esfuerzo de codificación de complejas y largas recetas en lenguaje C/C++ o, aún peor, en el eficiente pero críptico FORTRAN [12].

Afortunadamente, en la actualidad se cuentan con robustas plataformas de lenguaje de más alto nivel y generación que permiten una codificación más breve, eficiente, estructurada y legible. Como lo es el software de análisis estadístico R, de código abierto y licencia libre [15].

Cuenta, además, con diversas extensiones, provistas a manera de paquetes adicionales de fácil instalación al cuerpo principal del programa, que proveen de funciones y métodos específicos de diversas áreas del conocimiento. En cuanto Inteligencia Artificial, AMORE es un paquete para diseño, programación y entrenamiento de RNAf para esta plataforma [10].

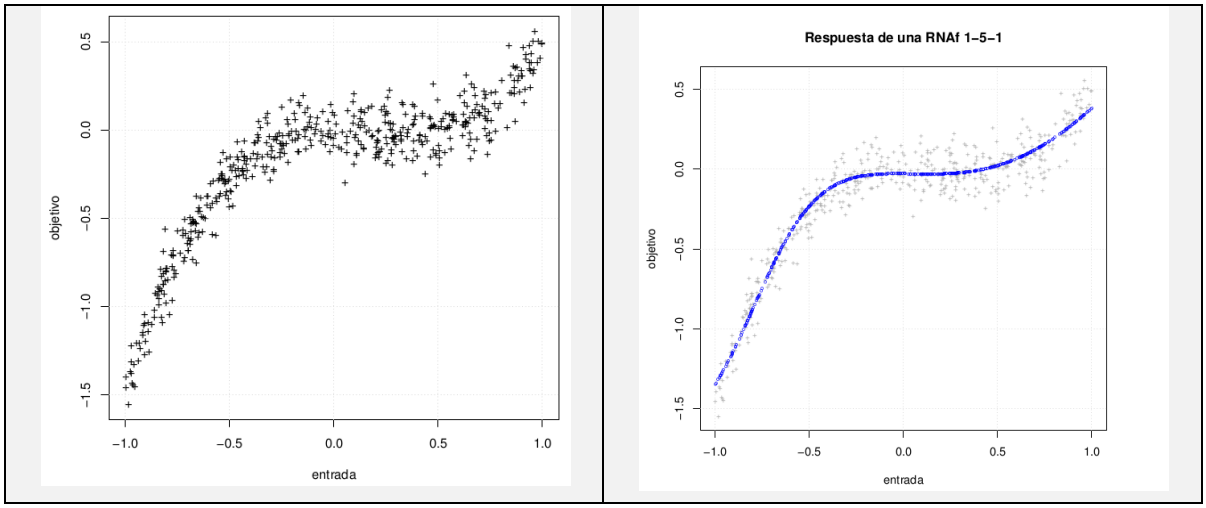
1.3 Ejemplo de una RNAf simple

Por ejemplo, una RNAf de topología 1-3-1, es decir, de 1 sola neurona de entrada, 3 neuronas en la única capa oculta y 1 neurona de salida; puede aprender con bastante facilidad (apenas 500 iteraciones) un patrón polinómico de grado 3 sumado a ruido gaussiano del 10 %y en unas cuantas líneas de código, como muestra la siguiente sesión de R:

Algoritmo 1.1 Sesión en R: ejemplo de una RNAf simple

```
# INICIO DE SESIÓN EN R
set.seed(12) # semilla para consistencia de aleatorios
require(AMORE) # paquete para RNAf
# población de datos: 5000 números entre -1 y 1
universo <- seq(-1, 1, length.out = 5000)
# datos de entrada: una muestra aleatoria de 500 números
entrada <- matrix(sample(x = universo, size = 500, replace =
FALSE),
ncol = 1)
# patrón de aprendizaje: una función cúbica con ruido normal
objetivo <- entrada^3 - 0.5 * entrada^2 + 0.1 *
rnorm(length(entrada),
mean = 0, sd = 1)
# gráfica objetivo ~ entrada
plot(entrada, objetivo, pch = "+")
```

```
grid()
# armado de la RNAf
RNAf <- newff(n.neurons = c(1, 5, 1), learning.rate.global = 0.02,
   momentum.global = 0.05, error.criterium = "LMLS", Stao = NA,
   hidden.layer = "sigmoid", output.layer = "purelin", method =
"ADAPTgdwm")
# entrenamiento usando sólo 80% de los datos
entrada.frac <- entrada[1:400, ]
objetivo.frac <- objetivo[1:400, ]
# y 500 repeticiones con reporte de 5 líneas
resultado <- train(RNAf, entrada.frac, objetivo.frac,
error.criterium = "LMLS",
report = TRUE, show.step = 500, n.shows = 5)
# simulación con todos los datos
y <- sim.MLPnet(resultado$net, entrada)
# gráfica original objetivo ~ entrada (puntos grises)
titulo = "Respuesta de una RNAf 1-5-1"
plot(entrada, objetivo, pch = "+", cex = 0.7, col = "darkgrey",
main = titulo)
# respuesta de la red entrenada (puntos azules)
points(entrada, y, col = "blue", pch = "o", cex = 0.5)
grid()
graphviz.MLPnet(net = resultado$net, filename = "RNAf-
ejemplo.dot") #diagrama
# FIN DE SESIÓN EN R
```



Como se puede observar en esta última gráfica de la sesión, los puntos indicados con '+' en color gris forman el patrón que real que debe aprender la red, la función polinómica con ruido añadido. En círculos azules se indican los puntos dados como respuesta de la RNAf.

Es evidente que la RNAf aprendió efectivamente el patrón, a pesar de haber sido entrenada con sólo una fracción de los datos y, además, con ruido añadido a los mismos.

Los pesos y conexiones de la red se muestran en el diagrama generado con la última línea de código de la sesión R, en la figura 1.3.

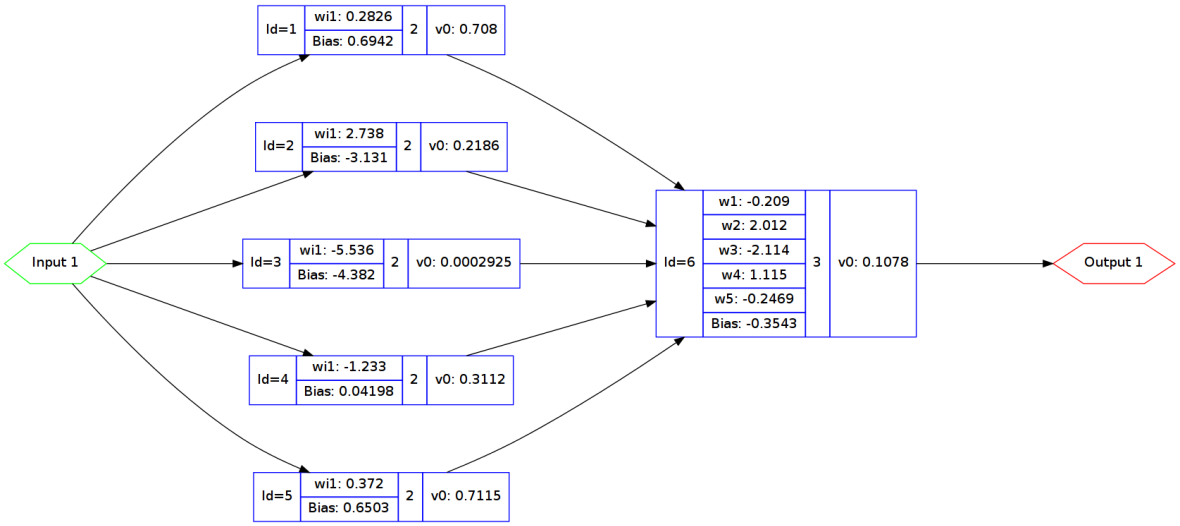


Figura 1.3 Estructura de la RNAf de ejemplo.

2 Datos

Las velocidades de corrientes litorales en el Puerto de Manzanillo (bajo el efecto de tsunamis) que se tienen como datos para el presente trabajo, proceden de los resultados aportados por un estudio previo. Dicha investigación fue realizada por personal del Instituto Mexicano del Transporte en su División de Ingeniería de Puertos y Costas, mediante un modelo hidráulico a escala 1:150 [13]. Los puntos de modelación (enumerados en detalle en el Anexo 2) y algunas fotos del modelo físico pueden observarse en la figura 2.1 y 2.2 respectivamente.

Figura 2.1 Puntos modelados del interior del Puerto de Manzanillo, Col. ('+' rojas) en el estudio IMT-VE-12/13.

Es importante recalcar que para los fines del presente trabajo, dichos datos se consideran debidamente calibrados y validados.

(a) (b) (c)

Figura 2.2 Modelo hidráulico utilizado en el estudio IMT-VE-12/13. (a) Generador de olas, (b) modelo a escala y (c) sensores.

Sus respectivos archivos tienen la siguiente nomenclatura: *corr_NNNN.dat*, donde *NNNN* es la elevación inicial de la superficie al momento de originarse el tsunami (η_0) dada en milímetros. Por lo que en la siguiente sesión R se cuentan y listan todos los casos dato y con la elevación inicial en metros, como se resume en la tabla 2.1.

Algoritmo 2.1. Conteo y listado de casos dato de elevaciones iniciales.

```
archisV <- list.files(path = "datos/", pattern = "corr_")
ETHA0 <- matrix(nrow = length(archisV), ncol = 1)
for (i in 1:length(archisV)) {
   ETHA0[i, 1] <- as.numeric(substr(archisV[i], start = 6, stop =
   9))/100
}
# t(ETHA0)[1,]
```

Tabla 2.1. Casos dato de elevaciones iniciales.

Caso	1	2	3	4	5	6	7	8	9
η_0	1.07	1.08	1.24	1.42	1.46	1.60	1.78	2.49	2.58

Caso	10	11	12	13	14	15	16	17	18
η_0	2.66	2.84	3.02	3.20	3.37	3.44	3.55	4.30	6.02

Caso	19	20	21
η_0	8.60	12.05	15.50

La estructura interna de cada uno de los archivos es una matriz de números reales. Como se expresa en la ecuación siguiente:

$$\eta_0: \begin{matrix} v_{x_{11}} & \cdots & v_{x_{1k}} \\ \vdots & \ddots & \vdots \\ v_{x_{n1}} & \cdots & v_{x_{nk}} \\ v_{y_{11}} & \cdots & v_{y_{1k}} \\ \vdots & \ddots & \vdots \\ v_{y_{n1}} & \cdots & v_{y_{nk}} \end{matrix} \mid \begin{cases} n \in [t_i = 1, \cdots, t_f = 59] \\ k \in [1, \cdots, 423] \end{cases}$$

Donde, dada una elevación inicial η_0 por cada archivo, hay *n* pasos de tiempo (59 en total) y *k* puntos de modelación (en total 423, ver fig. 2.1). La mitad de los renglones corresponde a la componente *x* de la velocidad en cada paso y en cada punto; así la segunda mitad corresponde a las de la componente *y*.

La lectura de un archivo se ejemplifica en la siguiente sesión R:

Algoritmo 2.2. Lectura de un archivo de datos típico.

```
# LEE UN ARCHIVO DE DATOS TÍPICO
d <- read.table("datos/corr_0107.dat")
dim(d) # IMPRIME EL NÚMERO DE RENGLONES Y COLUMNAS

## [1] 118 423

# DONDE LOS PRIMEROS RENGLONES SON PASOS DE TIEMPO DE LA
# COMPONENTE X
vx <- d[1:(dim(d)[1]/2), ]
# Y LOS RESTANTES RENGLONES SON PASOS DE TIEMPO DE LA
# COMPONENTE Y
vy <- d[(1 + dim(d)[1]/2):dim(d)[1], ]
dim(vx)

## [1] 59 423

dim(vy)

## [1] 59 423
```

3 Estrategias de cómputo

Así entonces, se tienen apenas 21 elevaciones iniciales para 59 pasos de tiempo por cada uno de los 423 puntos de modelación. Significando que en tan sólo 21 casos ejemplo una RNAf debiera aprender un patrón de casi 215 elementos; evidentemente se debe concluir que resultan insuficientes [4, 14, 16, 17, 19].

3.1 Limitando la complejidad de los patrones

Por lo anteriormente explicado se decidió reducir la complejidad del estudio de manera que la baja cantidad de casos experimentados pueda servir para entrenar una RNAf satisfactoriamente, adoptando las siguientes medidas:

1. La complejidad proveniente de los pasos de tiempo se puede evitar considerando funciones de agregación sobre ellos, como por ejemplo:

    a) el valor máximo y

    b) el promedio de la rapidez (magnitud de la velocidad) sobre cada punto de modelación.

2. Se pueden considerar sólo los primeros 50 del total de puntos de modelación.

Mismo que son mostrados en la figura 3.1.

Figura 3.1 Puntos de modelación efectivos para el presente estudio en el interior del puertode Manzanillo, Col.

3.2 Definición de variables de entrada y salida

En ausencia de relaciones funcionales ya conocidas, o de variables sintéticas adicionales, proponen como variables de entrada a la RNAf todo lo que se conoce de cada punto de modelación:

1. Elevación inicial: (ver Tabla 2.1).
2. Número de punto de modelación: k = [1; :::; 50] (ver capítulo 2).
3. Coordenada x (longitud) del punto de modelación.
4. Coordenada y (latitud) del punto de modelación.
5. Coordenada z (profundidad en positivo) del punto de modelación.

Y respecto de las variables de salida de la RNAf, como se mencionó anteriormente, hay un par de funciones de agregación interesantes, por lo que se tendrán dos RNAf distintas, a saber:

1. RNAf para modelar la rapidez máxima en los puntos modelados.
2. RNAf para modelar el promedio de la rapidez en los puntos modelados.

3.3 Normalización de los datos

Cuando se trabaja con datos continuos y de rangos muy variados como variables de entrada es recomendable normalizarlos antes de procesarlos en una RNAf, de lo contrario, el algoritmo tiene pocas posibilidades de converger [14, 12]. Empleándose para tal propósito la fórmula de la ecuación siguiente:

$$\phi\big(y(x)\big) = \frac{y - \bar{y}}{\sigma_y}$$

Donde *y* es el valor original a normalizar, $\bar{y}$ es el promedio de las mediciones quese quieren normalizar y $\sigma_y$su desviación estándar.

El efecto sobre la función *y(x)*es únicamente sobre su rango, pues la forma misma de la función no es afectada, si bien se pierde la dimensión de la función y queda en unidades absolutas (u. a.), como se muestra en la figura 3.2.

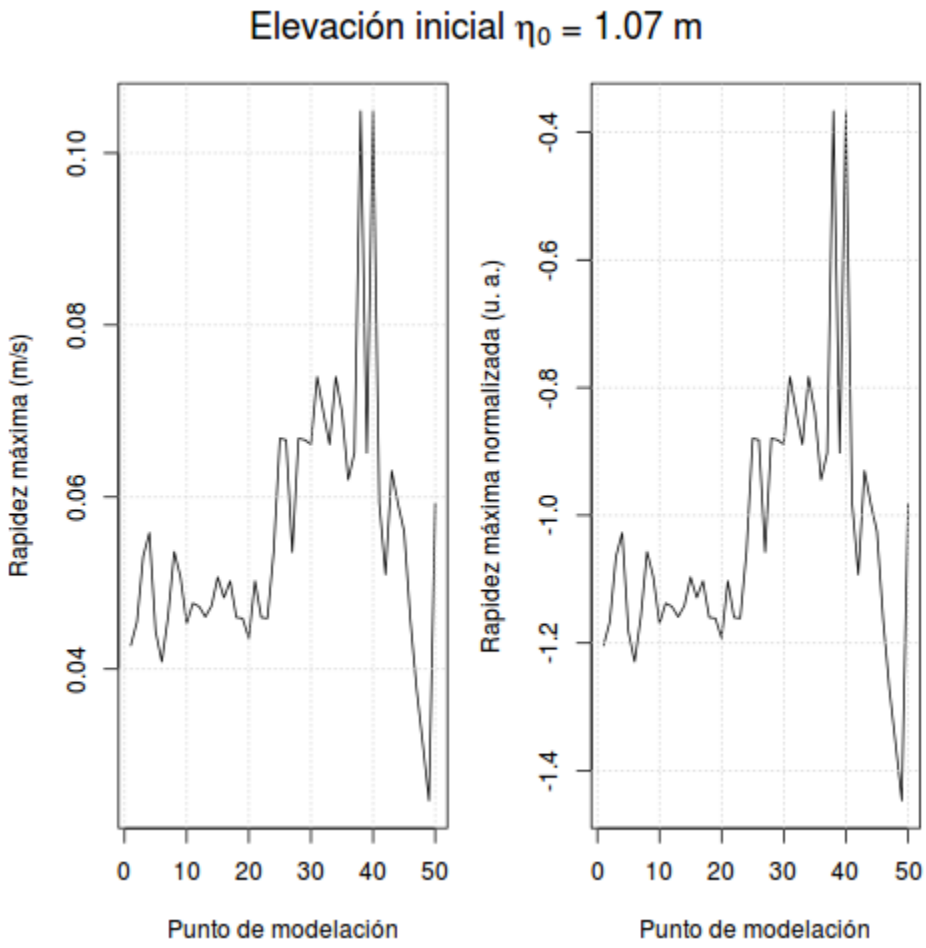


Figura 3.2 Efecto de la normalización sobre una porción de los datos.

Este procedimiento se aplicará a todas las variables de entrada de las RNAf que se programen. Dándose posteriormente una desnormalización a la variable de salida de la RNAf para recuperar su rango y dimensiones.

Como se puede observar en la primera sesión R en el algoritmo 1.1, hay parámetros propios de la RNAf cuyos valores óptimos deben hallarse para cada problema en particular, éstos son:

1. Las variables de entrada,

2. las variables de salida.

3. el número de neuronas intermedias, al menos el doble de las de entrada según la literatura (ver fig. 1.1),

4. la tasa de aprendizaje de la red (sensibilidad al patrón),

5. el *momentum* global (para evitar los mínimos locales y converger al mínimo global),

6. número de pasos de entrenamiento y

7. la fracción de los datos que servirán para entrenamiento y el resto para validación de la predicción.

Debiéndose encontrar la combinación que resulte en la RNAf óptima, es decir, en la que tenga el mínimo error porcentual absoluto medio (MAPE) que se expresa como:

$$MAPE = \frac{100}{N}\sum^{N}\left|\frac{RNAf - observados}{observados}\right|$$

es decir, el promedio del valor absoluto del cociente de la diferencia de los valores predichos por la red *RNAf* y los datos de validación *observados* entre los valores *observados*.

3.4 Cómputo paralelo y registro de avance

Con tener apenas 4 opciones por cada uno de los parámetros de la red que se deben optimizar al aplicarse sobre el problema, aún con la complejidad reducida, se tienen casi 2^{14} ejecuciones del algoritmo para cada una de las RNAf.

Si *a priori* se considerara que cada iteración pudiera tomar en promedio unos 10 segundos en ejecutarse, una corrida completa fácilmente pudiera superar las 45hrs. Lo que requiere de un enfoque de cómputo paralelo simple que aproveche todo el poder que ofrecen los actuales procesadores multinúcleo de 64-bit (como el Intel® Core i5 donde se llevaron a cabo las corridas). La plataforma estadística R, cuenta con los paquetes *doMc* y *foreach* para tal propósito [1, 2], cuando se tienen, como en esta ocasión, iteraciones independientes del ciclo de búsqueda de la combinación óptima de parámetros de la red.

Por otra parte, también, para tan largas corridas, se debe contar con un sistema de registro de resultados y seguimiento del algoritmo, para que ante una eventualidad como un corte de energía eléctrica (tan comunes en los países subdesarrollados), se pueda reanudar la corrida donde se interrumpió. Por este motivo, se elaboró una base de datos relacional en MySQL® para el registro del avance de la corrida, enlazada a R mediante el paquete RMySQL [9].

3.5 Código de pre-proceso

Todas las estrategias de cómputo previamente descritas, se aplican en las siguientes sesiones R, que pueden considerarse formalmente como la primera parte del código definitivo.

El pre-proceso comprende tres etapas:

1. Lectura de datos, normalización y disminución de la complejidad. La salida es el archivo que alimentará las entradas de la RNAf.

2. Creación de una base de datos relacional para el registro y seguimiento del algoritmo que buscará la RNAf óptima.

3. Población de la base de datos con todas las posibles combinaciones de parámetros de la red neuronal propuestos.

Algoritmo 3.1Pre-proceso con estrategias de cómputo implementadas(1ª Parte).

```
# FUNCIÓN PARA NORMALIZAR ----
norma <- function(original) {
   normalizado <- (original - mean(original))/sd(original)
   return(normalizado)
}
numPuntos <- 50 #reduce complejidad del problema
# CARGA DE DATOS ----
archisV <- list.files(path = "../datos/", pattern = "corr_")
# LISTADO DE ELEVACIONES INICIALES ----
ETHA0 <- matrix(nrow = length(archisV), ncol = 1)
for (i in 1:length(archisV)) {
   ETHA0[i, 1] <- as.numeric(substr(archisV[i], start = 6, stop =
      9))/100
}
# MATRICES VACÍAS ----
rMax <- matrix(nrow = numPuntos, ncol = dim(ETHA0)[1])
rMedia <- matrix(nrow = numPuntos, ncol = dim(ETHA0)[1])
e0 <- matrix(nrow = numPuntos, ncol = dim(ETHA0)[1])
# LEE LAS COORDENADAS DE LOS PUNTOS DE MODELAC. ----
cc <- read.table("../datos/TranspuestasXYZ.dat")
cc <- cc[1:numPuntos, ] #sólo los puntos efectivos
# CICLO PRINCIPAL QUE PROCESA CADA ARCHIVO DE DATOS ----
for (i in 1:dim(ETHA0)[1]) {
   a <- paste("../datos/", archisV[i], sep = "")
   d <- read.table(a)
   d <- d[, 1:numPuntos] #sólo los puntos efectivos
   vx <- d[1:59, ] #componente x
   vy <- d[60:118, ] #componente y
   v <- sqrt(vx * vx + vy * vy) #rapidez
   # RAPIDEZ EN CADA PUNTO (RENGLON) POR CADA EXPERIMENTO
   # (COLUMNA) ----
   rMedia[, i] <- matrix(sapply(v, mean), ncol = 1)
   rMax[, i] <- matrix(sapply(v, max), ncol = 1)
   e0[, i] <- matrix(rep(ETHA0[i], numPuntos), ncol = 1)
   if (i <= (dim(ETHA0)[1] - 1)) {
      # repite lectura de coordenadas
```

```
        cc0 <- read.table("../datos/TranspuestasXYZ.dat")
        cc0 <- cc0[1:numPuntos, ]
        cc <- rbind(cc, cc0)
        }
    }
# TRANSFORMACIÓN MATRICES A VECTORES ----
rapMax <- matrix(as.vector(rMax), ncol = 1)
rapMedia <- matrix(as.vector(rMedia), ncol = 1)
names(cc) <- c("lon", "lat", "z")
puntos <- matrix(rep(1:numPuntos, dim(ETHA0)[1]), ncol = 1)
elevac0 <- matrix(as.vector(e0), ncol = 1)
# JUNTA VECTORES ----
P <- cbind(puntos, elevac0, cc, rapMax, rapMedia)
P <- P[order(P[, 2], P[, 1]), ] #ordena por elevac. inic. y núm.
pto. modelac.
normP <- P #copia la matriz P
for (j in 1:dim(P)[2]) {
    # normaliza cada columna
    normP[, j] <- matrix(norma(P[, j]), ncol = 1)
}
# ESCRIBE ARCHIVOS PARA ALIMENTAR A LA RNAf ----
write.table(P, "../datos/P.txt", row.names = FALSE) #sin
normalizar
write.table(normP,   "../datos/normP.txt",   row.names   =   FALSE)
#NORMALIZADO
```

Ahora ya se tiene el archivo que ha de alimentar a la RNAf llamado *normP.txt*. Contiene 1050 renglones (50 puntos de modelación por 21 casos de elevación inicial) y 7 columnas:

1. el número de punto de modelación;
2. la elevación inicial (repetida para los 50 puntos en cada ocasión);
3. longitud,
4. latitud,
5. la profundidad (*z*) del punto de modelación;
6. rapidez máxima,
7. y rapidez media.

Para la siguiente etapa es necesario contar con la base de datos relacional y su respectiva tabla creada, planeada para hacer en ella el registro y seguimiento de la rutina que buscará la RNAf óptima. Como se describe en el algoritmo siguiente:

Algoritmo 3.2 Creación de la base de datos en MySQL para seguimiento del algoritmo.

```
# CREA Y SELECCIONA LA BASE DE DATOS
CREATE DATABASE corridas ; USE corridas ;
# CREA LA ÚNICA TABLA DE LA BD
CREATE TABLE ` combinaciones ` (
`nCP` int (11) DEFAULT NULL,
` colsP ` varchar (50) DEFAULT NULL,
` colsQ ` varchar (50) DEFAULT NULL,
` hidden ` int (11) DEFAULT NULL,
` l r ` double DEFAULT NULL,
`mg` double DEFAULT NULL,
` ss ` double DEFAULT NULL,
` picked ` int (11) DEFAULT NULL,
`done ` int (11) DEFAULT NULL )
ENGINE=InnoDB DEFAULT CHARSET= latin1 ;
# CREA EL USUARIO CON CONTRASEÑA Y DA ACCESO A LA BD EN EL SERVIDOR
ACTUAL
CREATE USER ' userRNAf '@' localhost ' IDENTIFIED BY ' imt123 ' ;
GRANT ALL ON corridas . * TO ' userRNAf '@' localhost ' ;
```

La siguiente parte del pre-proceso es, pues, la generación de todas las combinaciones posibles de los parámetros de la red. Las variantes propuestas, es decir, los conjuntos dominio, se enumeran a continuación:

Fracción de los 21 casos de elevación inicial que se usan para entrenamiento:

- 2 casos para entrenamiento, 19 para para validación de la RNAf.
- 7 casos para entrenamiento, 14 para para validación de la RNAf.
- 20 casos para entrenamiento, 1 para para validación de la RNAf.

Columnas de entrada:

- Número de punto de modelación y elevación inicial.
- Número de punto de modelación, elevación inicial y profundidad del punto de modelación.
- Número de punto de modelación, elevación inicial, longitud, latitud y profundidad del punto de modelación.

Columnas de salida:

- Sólo la objetivo: rapidez máxima o media, según el caso.
- La columna objetivo y el número de punto de modelación.
- La columna objetivo, el número de punto de modelación y la elevación inicial.

Tasa de aprendizaje: desde 2×10^{-2} hasta 2×10^{-5} en 4 pasos.

Momentum global: desde 5×10^{-2} hasta 5×10^{-5} en 4 pasos.

Pasos de entrenamiento: desde 100 hasta 9000 en 4 pasos.

Factores para multiplicar el número de neuronas de la capa de entrada y determinar el número de neuronas en la capa oculta: desde 2.0 hasta 3.25 en 4 pasos.

Y su implementación se describe en el algoritmo siguiente.

Algoritmo 3.3 Pre-proceso con estrategias de cómputo implementadas (2ª parte)

```
# LIBRERIAS Y REGISTRO DE NÚCLEOS ----
set.seed(12) #fija la semilla de aleatorios
library(RMySQL) #para conectarse con la BD
library(doMC) #para cómputo paralelo
registerDoMC(cores = 4) #bueno para Core-i5
# VALORES INICIALES ----
tope <- 21 #número de casos de elevación inicial
numPuntos <- 50 #puntos de modelación reducidos
colObj <- 6 #rapMax o 7 para rapMedia
n <- 4 #divisiones de parámetros propios de la RNAf
archiDatos <- "../datos/normP.txt"
# CONJUNTOS DOMINIO ----
numCasosP <- rev(c(2, 7, 20)) #casos de aprendizaje

colsP <- rev(list(c(1, 2), c(1, 2, 5), c(1, 2, 3, 4, 5)))
#columnas de entrada
colsQ <- rev(list(c(colObj), c(colObj, 1), c(colObj, 1, 2),
c(colObj,
2, 5))) #cols. de salida
LR <- c(0.02, 0.002, 2e-04, 2e-05) #tasas de aprendizaje
MG <- c(0.05, 0.005, 5e-04, 5e-05) #momenta
SS <- c(100, 500, 1000, 9000) #pasos de entrenamiento
FF <- c(2, 2.25, 2.5, 2.75, 3, 3.25) #factores p/núm. neuronas
capa oculta
picked <- 0 #bandera: combinación en proceso
done <- 0 #bandera: combinación ya procesada
# CONEXION CON BD EN MYSQL ----
```

```
base <- "corridas"
usuario <- "userRNAf"
passw <- "imt123"
bd <- dbConnect(MySQL(), user = usuario, password = passw, dbname
= base,
host = "localhost")
dbSendQuery(bd, "TRUNCATE combinaciones") #borra registros pasados

# CICLOS PARALELIZADOS ----
for (nCP in numCasosP) {
   for (cP in colsP) {
      for (cQ in colsQ) {
         for (factor in FF) {
            foreach(k = 1:n) %dopar% {
            for (j in 1:n) {
               foreach(i = 1:n) %dopar% {
               bd <- dbConnect(MySQL(), user = usuario,
                  password = passw, dbname = base, host =
                  "localhost")
               lr = LR[k]
               mg = MG[j]
               ss = SS[i]
               orden <- paste("INSERT INTO combinaciones VALUES
                  (",
                  nCP, ",'", paste(as.character(cP), collapse =
                     "-"),
                  "','", "'", paste(as.character(cQ), collapse =
                     "-"),
                  "',", (floor(length(cP) * factor + 1)),
                  ",", lr, ",", mg, ",", ss, ",", picked,
                  ",", done, ")", sep = "")
               dbSendQuery(bd, orden)

                  }
               }
            }
         }
      }
   }
}
dbDisconnect(bd)
```

4 Búsqueda de la RNAf óptima

Una vez poblada la base de datos con las posibles combinaciones de parámetros de la red neuronal y teniéndose el archivo de datos de entrada completado, se procede a la búsqueda de la RNAf que converge con menor MAPE o RMSE, es decir, la RNAf en sí.

Algoritmo 4.1 Búsqueda de la RNAf óptima

```
# COMENTAR LÍNEAS EN CASO DE REINICIAR TRAS APAGÓN ----
file.remove("status.txt") #borra resultados anteriores
x <- cbind("colObj", "nCP", "colsP", "colsQ", "hidden", "lr",
"mg", "ss", "mape", "sigma", "lmls", "tp", "tt")
write.table(x, file = "status.txt", row.names = FALSE, col.names =
FALSE,
sep = "|")
# LIBRERIAS, VALORES INICIALES Y REGISTRO DE NÚCLEOS ----
t0 <- proc.time() #inicia cronómetro
set.seed(12) #semilla para aleatorios
library(AMORE) #para RNAf
library(RMySQL) #para conexión con BD
library(doMC) #cómputo paralelo
registerDoMC(cores = 4) #bueno para Core-i5
tope <- 21 #casos de elevación inicial
numPuntos <- 50 #complejidad reducida
colObj <- 6 #rapMax o 7 para rapMedia
archiDatos <- "../datos/normP.txt"
# VALORES DE CONEXION CON BD EN MYSQL ----
base <- "corridas"
usuario <- "userRNAf"
passw <- "imt123"
# CICLOS RESTANTES ----
bd <- dbConnect(MySQL(), user = usuario, password = passw, dbname
= base,
host = "localhost")
Q0 <- "UPDATE combinaciones SET picked=0 WHERE picked=1 AND
done=0"
dbGetQuery(bd, Q0) #repara interrupción
Q1 <- "SELECT * FROM combinaciones WHERE picked=0 AND done=0"
combinacs <- dbGetQuery(bd, Q1) #faltan por procesar
dbDisconnect(bd)

# CICLO PARALELIZADO ----
foreach(k = 1:dim(combinacs)[1], .inorder = FALSE, .combine =
rbind) %dopar%{
   set.seed(12) #homologa semilla aleatoria
```

```
    caso <- combinacs[k, ] #cada una de las combinaciones
    bd <- dbConnect(MySQL(), user = usuario, password = passw,
       dbname = base, host = "localhost")
    marcaPicked <- paste("UPDATE combinaciones SET picked = 1 WHERE
       nCP=",
       caso$nCP, " AND colsP='", caso$colsP, "' AND colsQ='",
       caso$colsQ, "' AND hidden=", caso$hidden, " AND lr=",
       caso$lr, " AND mg=", caso$mg, " AND ss=", caso$ss,
       sep = "")
    dbGetQuery(bd, marcaPicked) #asegura para procesar
    t1 <- proc.time() #inicia cronómetro local del caso
    P1 <- read.table(archiDatos, header = TRUE) #lee los datos
    nCP <- caso$nCP #fracción de casos para entrenamiento

    # LEE LAS COLUMNAS DE ENTRADA ----
    A <- as.character(caso$colsP)
    B <- strsplit(A, "-")[[1]]
    D <- paste(B, collapse = ",")
    E <- paste("c(", D, ")")
    cP <- eval(parse(text = as.character(E)))

    # LEE LAS COLUMNAS DE SALIDA ----
    A <- as.character(caso$colsQ)
    B <- strsplit(A, "-")[[1]]
    D <- paste(B, collapse = ",")
    E <- paste("c(", D, ")")
    cQ <- eval(parse(text = as.character(E)))
    hidden = caso$hidden #núm. neuronas capa oculta
    lr = caso$lr #tasa de aprendizaje
    mg = caso$mg #momentum
    ss = caso$ss #pasos de entrenamiento

    # MATRIZ DE ENTRADA DE DATOS A LA RNAf ----
    P <- as.matrix(P1[1:(nCP * numPuntos), cP])
    # PATRON DE ENTRENAMIENTO ----
    Q <- as.matrix(P1[1:(nCP * numPuntos), cQ])

    # armado de la RNAf ----
    net.start <- newff(n.neurons = c(dim(P)[2], hidden,
       length(cQ)), learning.rate.global = lr, momentum.global =
       mg,
       error.criterium = "LMLS", Stao = NA, hidden.layer =
       "tansig",
       output.layer = "purelin", method = "ADAPTgdwm")

    # entrenamiento con fracción de datos ----
    result <- train(net.start, P, Q, error.criterium = "LMLS",
    report = TRUE, show.step = ss, n.shows = 3)

    # simulación con el resto de datos ----
    sim <- sim.MLPnet(result$net, as.matrix(P1[(numPuntos *
```

```
        nCP + 1):(numPuntos * tope), cP]))

    # valores observados (resto de datos) ----
    obs <- as.matrix(P1[(numPuntos * nCP + 1):(numPuntos *
        tope), cQ])

    # modelo lineal simulados vs observados ----
    fit <- lm(sim[, 1] ~ obs[, 1])

    # cálculo del MAPE ----
    mape <- mean(abs((sim[, 1] - obs[, 1])/obs[, 1]))

    tf <- round((proc.time() - t1)[[3]], 4) #tiempo de cómputo RNAf
    tff <- round((proc.time() - t0)[[3]], 4) #t total del programa

    # MATRIZ DE RESULTADOS
    params <- as.character(cbind(colObj, nCP,
        paste(as.character(cP),
        collapse = "-"), paste(as.character(cQ), collapse = "-"),
        hidden, lr, mg, ss, mape, round(summary(fit)$sigma,
        4), round(result$Merror[3], 4), tf, tff))

    # REGISTRO DE RESULTADOS AL ARCHIVO status.txt
    write.table(matrix(params, nrow = 1), file = "status.txt",
        append = TRUE, row.names = FALSE, col.names = FALSE,
        sep = "|", quote = F)
    marcaDone = paste("UPDATE combinaciones SET done = 1 WHERE
        nCP=",
        caso$nCP, " AND colsP='", caso$colsP, "' AND colsQ='",
        caso$colsQ, "' AND hidden=", caso$hidden, " AND lr=",
        caso$lr, " AND mg=", caso$mg, " AND ss=", caso$ss,
        " AND picked=", 1, sep = "")
    dbGetQuery(bd, marcaDone) #marca la combinación de 'procesada'

    # DESCONECTA LA BD ----
    dbDisconnect(bd)
}
```

Una vez ejecutadas todas las corridas programadas en la sección anterior, se registran los resultados en el archivo *status.txt*, uno para cada RNAf. Su estructura consta de:

- 13825 renglones correspondientes a todas las posibles combinaciones de parámetros y casos dato.
- 13 columnas separadas entre sí por el símbolo pipe '|':
    1. Columna objetivo: correspondiente al número de columna del archivo *normP.txt* que simula la RNAf, a saber, 6 para rapidez máxima y 7 para rapidez media.
    2. nCP: la fracción de casos que sirven para entrenamiento.

3. colsP: las columnas de entrada a la RNAf.
4. colsQ: las columnas de salida de la RNAf.
5. hidden: número de neuronas en la capa oculta de la RNAf.
6. lr : tasa de aprendizaje.
7. mg: momentum global.
8. ss: número de pasos de entrenamiento.
9. mape: error MAPE entre la predicción de la RNAf y los casos reservados para validación.
10. sigma: error RMSE entre la predicción de la RNAf y los casos reservados para validación.
11. lmls: error de convergencia del entrenamiento de la red.
12. tp: tiempo de cómputo ocupado por la combinación particular.
13. tt: tiempo total del programa transcurrido hasta el momento.

4.1 RNAf óptima

Basta entonces con hacer una ordenación de los datos contenidos en el archivo *status.txt* de acuerdo a la columna 9 (MAPE) y se podrá recuperar la RNAf óptima para cada caso, la rapidez máxima o la rapidez media para cada punto de modelación.

Dicha red neuronal artificial se puede reconstruir mediante un algoritmo parecido al que hizo la búsqueda (las modificaciones son obvias). Haciendo énfasis en que, hay que adaptarlo para ambos casos de RNAf: rapidez máxima y rapidez media.

Algoritmo 4.2 RNAf óptima para aprendizaje del patrón de rapidez máxima.

```
set.seed(12) #homologa la semilla de aleatorios
# MODIFICAR RUTA PARA CADA RNAf
d <- read.table("rapidezMaxima/status.txt", header = TRUE, sep =
"|")
d1 <- d[order(d$mape, d$ss), ] #ordena por menor MAPE
dd <- head(d1)[1, ] #toma el mínimo
write.table(dd, "rapidezMaxima/min.txt", col.names = TRUE,
row.names = FALSE)
# LIBRERIAS ----
colObj <- 6 #6 para rapMax o 7 para rapMedia
tope <- 21 #casos de elevac. inic.
numPuntos <- 50 #ptos. de modelac.
archiDatos <- "datos/normP.txt" #datos normalizados
archiParam <- "rapidezMaxima/min.txt" #params. de RNAf optima
require(AMORE) #para modelar RNAf
pmin <- read.table(archiParam, header = TRUE) #vector de params.
lr <- pmin$lr #tasa de aprendizaje
mg <- pmin$mg #momentum global
ss <- pmin$ss #núm. de pasos de entrenam.
nCP <- pmin$nCP #fracc. de datos de entrenam.

# LEE LAS COLUMNAS DE ENTRADA ----
```

```
A <- as.character(pmin$colsP)
B <- strsplit(A, "-")[[1]]
D <- paste(B, collapse = ",")
E <- paste("c(", D, ")")
cP <- eval(parse(text = as.character(E)))
# LEE LAS COLUMNAS DE SALIDA ----
A <- as.character(pmin$colsQ)
B <- strsplit(A, "-")[[1]]
D <- paste(B, collapse = ",")
E <- paste("c(", D, ")")
cQ <- eval(parse(text = as.character(E)))
P1 <- read.table(archiDatos, header = TRUE) #archivo de datos
P <- as.matrix(P1[1:(nCP * numPuntos), cP]) #matriz de entrada
Q <- as.matrix(P1[1:(nCP * numPuntos), cQ]) #patrón de entenam.
# armado de la RNAf ----
net.start <- newff(n.neurons = c(dim(P)[2], pmin$hidden,
length(cQ)),
    learning.rate.global = lr, momentum.global = mg,
error.criterium = "LMLS",
    Stao = NA, hidden.layer = "tansig", output.layer = "purelin",
    method = "ADAPTgdwm")
# entrenamiento ----
result <- train(net.start, P, Q, error.criterium = "LMLS", report
    = TRUE,
    show.step = ss, n.shows = 3)
# simulación ----
sim <- sim.MLPnet(result$net, as.matrix(P1[(nCP * numPuntos +
    1):(numPuntos * tope), cP]))
# fracción de datos para validación ----
obs <- as.matrix(P1[(nCP * numPuntos + 1):(numPuntos * tope),cQ])
# modelo lin. simulados vs observados ----
fit <- lm(sim[, 1] ~ obs[, 1])
mape <- mean(abs((sim[, 1] - obs[, 1])/obs[, 1])) #MAPE
rmse <- round(sqrt(mean(resid(fit)^2)), 4) #RMSE
```

5 Resultados

La RNAf óptima para aprendizaje del patrón de rapidez máxima en los 50 puntos de modelación en el interior del puerto de Manzanillo, Col. tiene las siguientes características:

5.1 RNAf estimador del patrón de rapidez máxima

Se describe a continuación la topología de la RNAf estimador del patrón de rapidez máxima. Como entrada tiene las 5 variables conocidas del punto de modelación:

1. Número del punto de modelación,
2. elevación inicial (η_0),
3. latitud,
4. longitud,
5. y profundidad *z* del punto de modelación.

Y una sola variable de salida: la rapidez máxima. Como se ilustra en el diagrama de la RNAf óptima en la figura 5.1

Sus parámetros se enumeran a continuación:

- Los casos de entrenamiento son 20, y 1 de validación de respuesta de la red neuronal.
- Cuenta con 14 neuronas en la capa oculta.
- La tasa de aprendizaje es: 2×10^{-4}.
- El *momentum* global: 0.05.
- Se requirieron de 9000 pasos de entrenamiento en los 20 casos primerosdesignados.

El error MAPE entre los valores observados y los predichos por la RNAf, es de apenas 17.6% (RMSE de 0.48). Con una distribución normal en los residuales, se reconoce que la cantidad variables de entrada en la red es suficiente y que aprende el patrón satisfactoriamente. Como atestiguan los resultados del modelo lineal entre valores observados y simulados en la tabla 5.1, así como las gráficas e histograma de la figura 5.2 y la gráfica del patrón de rapidez máxima observado junto con el predicho por la red con bandas de confianza de la figura 5.3.

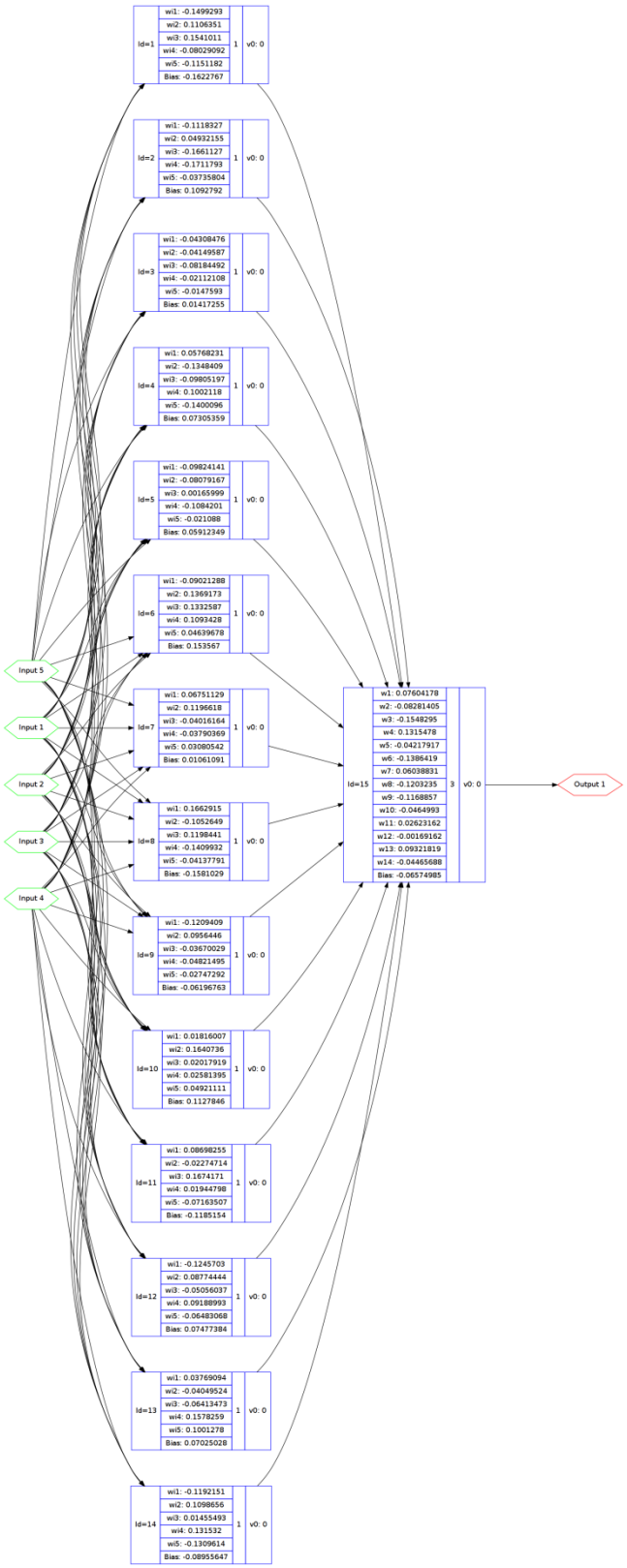


Figura 5.1 Diagrama de la RNAf óptima para el patrón de rapidez máxima.

Tabla 5.1 Regresión lineal entre la predicción de la RNAf y los datos de validación en el patrón de rapidez máxima.

```
Call:
lm(formula = sim[, 1] ~ obs[, 1])

Residuals:
    Min      1Q  Median      3Q     Max
-1.2449 -0.3283 -0.0088  0.2510  1.3799

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.23460    0.17793   1.319    0.194
obs[, 1]     0.87262    0.07399  11.793 8.74e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4911 on 48 degrees of freedom
Multiple R-squared:  0.7434,Adjusted R-squared:  0.7381
F-statistic: 139.1 on 1 and 48 DF,  p-value: 8.737e-16
```

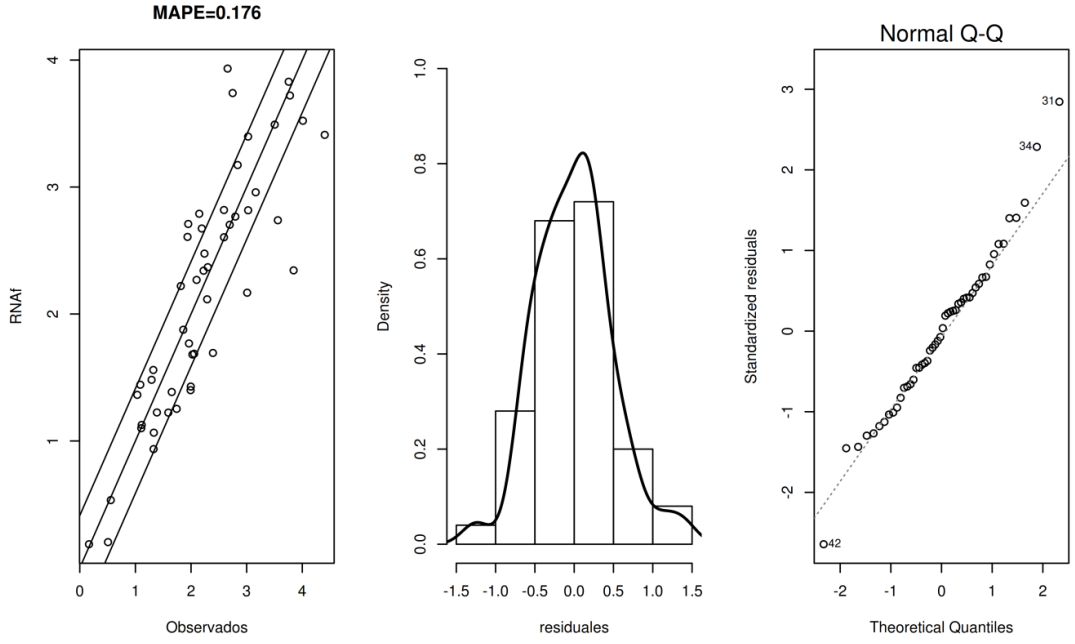


Figura 5.2 Gráfica del modelo lineal entre la predicción de la RNAf y los datos de validación en el patrón de rapidez máxima. Junto con el histograma y el gráfico normal q-q que muestran que los residuales tienen una distribución normal.

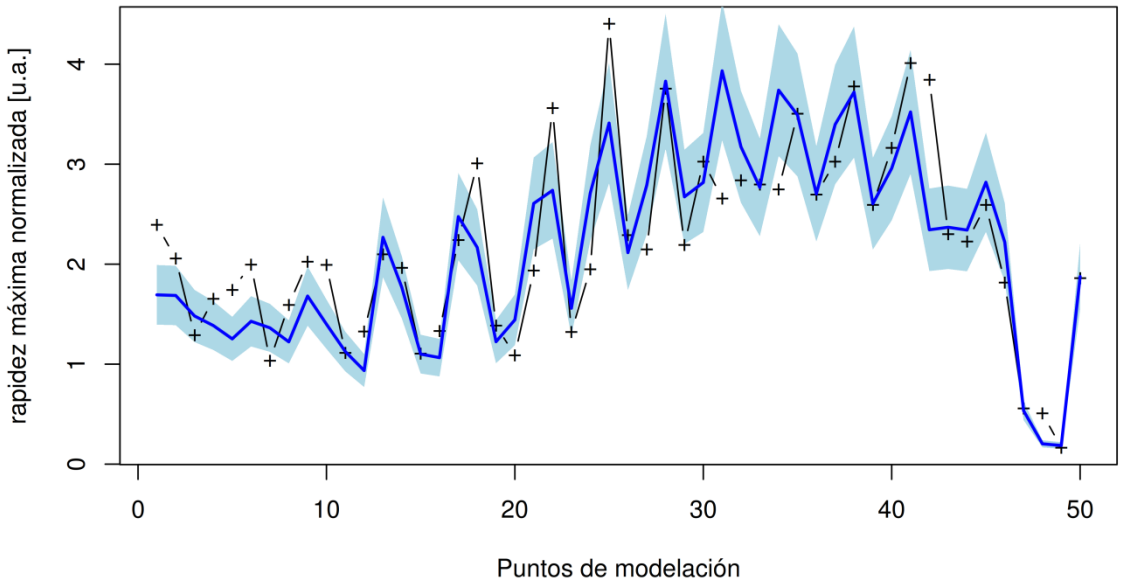


Figura 5.3 Patrón de rapidez máxima observado ('+' negras) junto con el predicho por la red (línea azul) con bandas de confianza (sombra azul claro).

5.2 RNAf estimador del patrón de rapidez media

Por otra parte, la RNAf óptima para aprendizaje del patrón de rapidez media en los puntos de modelación en el interior del Puerto de Manzanillo, Col. tiene las siguientes características:

Como entrada tiene las 5 variables conocidas del punto de modelación:

1. Número del punto de modelación,
2. elevación inicial (η_0),
3. latitud,
4. longitud,
5. y profundidad *z* del punto de modelación.

Y una sola variable de salida: la rapidez media.

Como se ilustra en el diagrama de la RNAf óptima en la figura 5.4.

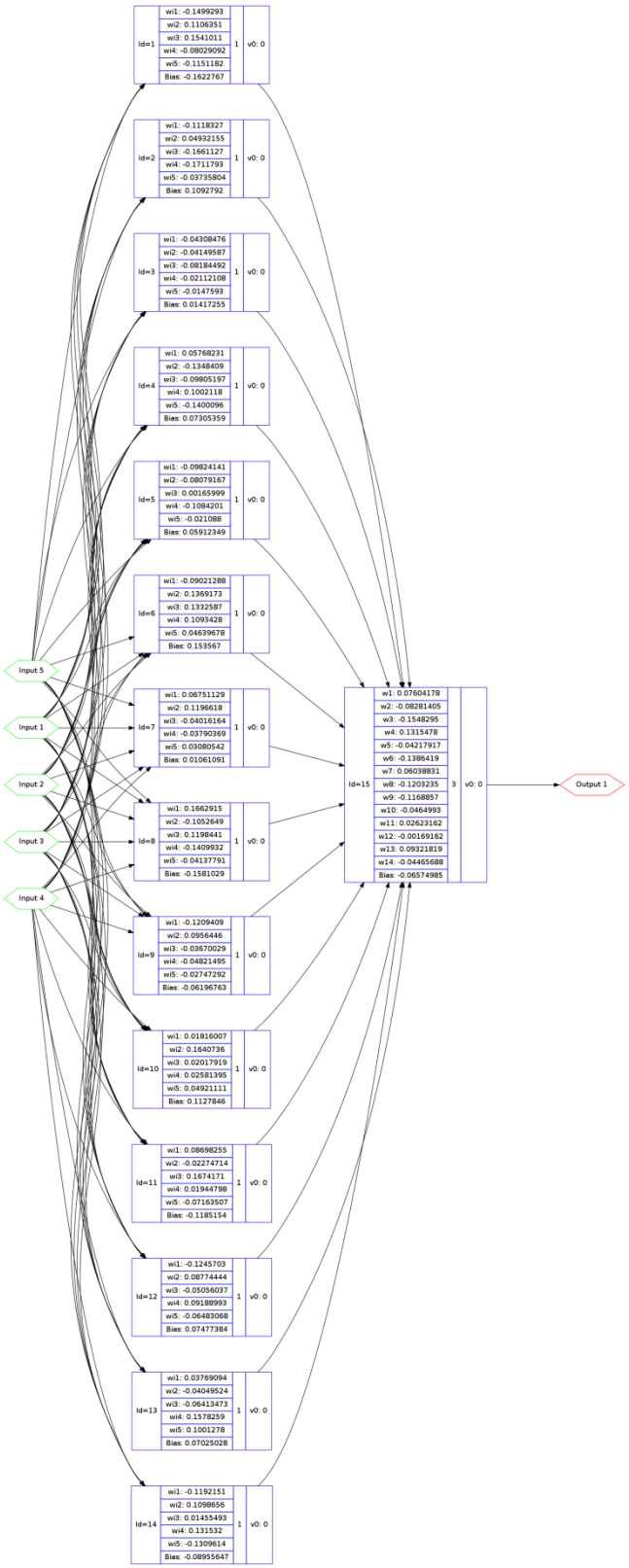


Figura 5.4 Diagrama RNAf óptima para patrón de rapidez media.

Sus parámetros se enumeran a continuación:

- Los casos de entrenamiento son 20 y 1 de validación de respuesta de la red neuronal.
- Cuenta con 14 neuronas en la capa oculta.
- La tasa de aprendizaje es: 2×10^{-3}.
- El *momentum* global: 5×10^{-5}.
- Se requirieron tan sólo de 500 pasos de entrenamiento en los 20 casos primeros designados.

El error MAPE entre los valores observados y los predichos por la RNAf, es de apenas 18.7% (RMSE de 0.39). Con una distribución normal en los residuales, se reconoce que la cantidad variables de entrada en la red es suficiente y que aprende el patrón satisfactoriamente. Como atestiguan los resultados del modelo lineal entre valores observados y simulados en la tabla 5.2, así como las gráficas e histograma de la figura 5.5 y la gráfica del patrón de rapidez media observado junto con el predicho por la red con bandas de confianza de la figura 5.6.

Tabla 5.2 Regresión lineal entre la predicción de la RNAf y los datos de validación en el patrón de rapidez media.

```
Call:
lm(formula = sim[, 1] ~ obs[, 1])

Residuals:
     Min       1Q   Median       3Q      Max
-0.72542 -0.31442  0.02915  0.22389  1.06892

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.65255    0.17647   3.698 0.000558 ***
obs[, 1]     0.68575    0.06992   9.808 4.76e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4012 on 48 degrees of freedom
Multiple R-squared:  0.6671, Adjusted R-squared:  0.6602
F-statistic:  96.2 on 1 and 48 DF,  p-value: 4.76e-13
```

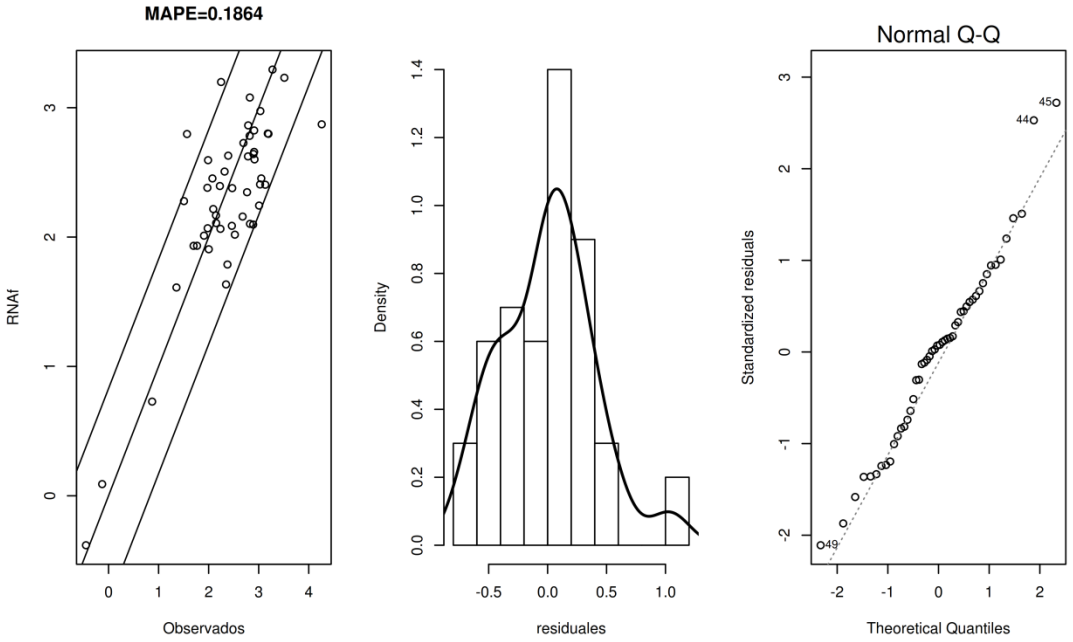


Figura 5.5 Gráfica del modelo lineal entre la predicción de la RNAf y los datos de validación en el patrón de rapidez media. Junto con el histograma y el gráfico normal q-q que muestran que los residuales tienen una distribución normal.

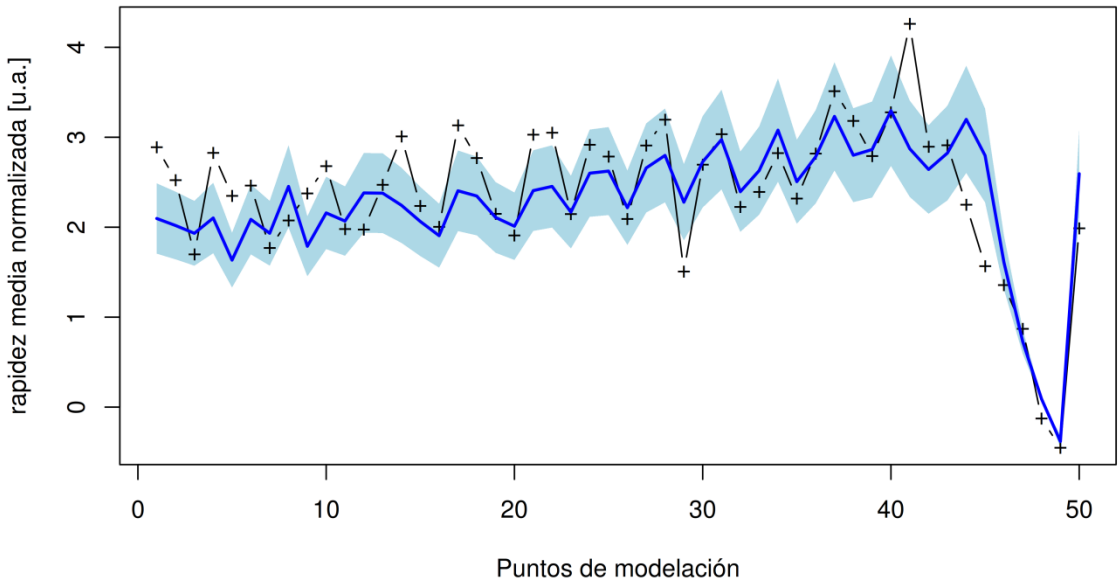


Figura 5.6 Patrón de rapidez media observado (‘+’ negras) junto con el predicho por la red (línea azul) con bandas de confianza (sombra azul claro).

6 Conclusiones

El presente trabajo muestra que es posible construir dos redes neuronales artificiales tipo *feedforward* como estimadores de corrientes en el interior del Puerto de Manzanillo, Col. bajo la acción de tsunamis, uno para el patrón de rapidez máxima y otro para el patrón de rapidez media.

Bastaron, en ambos casos, considerar 5 variables de entrada, todas referentes al punto de modelación, y una única variable de salida, la rapidez máxima o media según el caso de la red construida. La aparente distribución normal de los residuales del modelo lineal, entre la respuesta de la red y los datos de validación, es un buen indicador de que las redes neuronales artificiales programadas aprendieron en su totalidad los patrones a los que fueron aplicadas.

Con incipientes 21 casos dato, se halló que para ambos estimadores, la red óptima se tiene utilizando 20 casos para entrenamiento y 1 para validación de la respuesta de la red neuronal. En ambos casos se logró una respuesta de la red con un error porcentual absoluto medio menor al 19 %.

Ambas redes neuronales óptimas utilizaron una capa intermedia con 2.8 veces más neuronas que en su capa de entrada; tal como indica la literatura. Respecto de las tasas de aprendizaje son similares en ambos casos. Sin embargo, el que el *momentum* global sea 10^3 veces menor en el caso del patrón de rapidez media, puede indicar la ausencia de mínimos locales, que es congruente con el hecho de que se requirieron casi 10^2 veces menos pasos de entrenamiento también.

Futuros desarrollos pueden involucrar técnicas que acorten aún más el error de la predicción de la red neuronal artificial, abarquen la complejidad completa del problema o incluyan la evolución temporal del fenómeno del tsunami entrando al puerto. Para lo que evidentemente se requerirá de un mayor número de casos dato.

Por otra parte, este trabajo tomó ventaja de la plataforma R, que respecto de los lenguajes de programación tradicionales, mostró prestaciones superiores y simplicidad de codificación.

Bibliografía

[1] ANALYTICS, REVOLUTION. 2013. doMC: Foreach parallel adaptor for the multicore package. R package version 1.3.0.

[2] ANALYTICS, REVOLUTION, & WESTON, STEVE. 2014. foreach: Foreach looping construct for R. R package version 1.4.2.

[3] BIVAND, ROGER, KEITT, TIM, & ROWLINGSON, BARRY. 2014. rgdal: Bindings for the Geospatial Data Abstraction Library. R package version 0.8-16.

[4] CYBENKO, GEORGE. 1989. Approximation by Superpositions of a Sigmoidal Function. Mathematics of control, signals, and systems, **2**, 303–314.

[5] DEAN, R.G., & DALRYMPLE, R.A. 1991. Water Wave Mechanics for Engineers and Scientists. Advanced series on ocean engineering. World Scientific.

[6] DEO, M. C. 2010. Artificial neural networks in coastal and ocean engineering. Indian journal of geomarine science, **39**(4).

[7] DÉVORA, ANTONIO J. SÁNCHEZ, & SANZ, SALVADOR F. FARRERAS. 1993. Catálogo de Tsunamis (Maremotos) en la Costa Occidental de México. World Data Center A for Solid Earth Geophysics Publication SE-50.

[8] HAYKIN, SIMON S. 1994. Neural Networks: A comprehensive foundation. Macmillan College Publishing Company.

[9] JAMES, DAVID A., & DEBROY, SAIKAT. 2012. RMySQL: R interface to the MySQL database. R package version 0.9-3.

[10] LIMAS, MANUEL CASTEJÓN, MERÉ, JOAQUÍN B. ORDIERES, MARCOS, ANA GONZÁLEZ, ASCACIBAR, FRANCISCO JAVIER MARTÍNEZ DE PISÓN, ESPINOZA, ALPHA V. PERNÍA, & ELÍAS, FERNANDO ALBA. 2010. AMORE: A MORE flexible neural network package. R package version 0.2-12.

[11] MANDAL, SUBHENDU, PATIL, SANJAY G., MANHUNATHA, Y. R., & HEGDE, A. V. 2008. Application of Neural Networks in Coastal Engineering – an Overview. The 12th international conference of international association for computer methods and advances in geomechanics, Oct.

[12] MASTERS, TIMOTHY. 1993. Practical Neural Network Recipes in C++. Academic Press.

[13] MONTOYA, JOSÉ MIGUEL, FLORES ÁLVAREZ, JUAN ESTÉBAN, CASAS VALENCIA, CINDY, et al. 2013. Estudio en modelo hidráulico para determinar el patrón de velocidades originado por tsunamis locales en el Puerto de Manzanillo, Col. Rep Tec. VE-12/13. Instituto Mexicano del Transporte.

[14] ORR, GENEVIEVE. 2007 (Apr.). Neural Networks (lecture notes). http://www.willamette.edu/_gorr/classes/cs449/intro.html.

[15] R CORE TEAM. 2013. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

[16] TURBAN, E., ARONSON, J.E., & LIANG, T.P. 2005. Decision support systems and intelligent systems. Pearson/Prentice Hall.

[17] VASQUEZ-LOPEZ, JUAN P. 2013 (Apr.). Las redes neuronales artificiales feedforward como identificadoras de regresiones espurias entre caminatas aleatorias. Tesis de grado de Maestría en Ciencias en Cómputo Aplicado.

[18] WARNER, BRAD, & MISRA, MANAVENDRA. 1996. Understanding Neural Networks as Statistical Tools. The american statistician, **50**(4), 284–293.

[19] ZANUTTIGH, BARBARA, FORMENTIN, SARA MIZAR, & BRIGANTI, RICCARDO. 2013. A neural network for the prediction of wave reflection from coastal and harbor structures. Coastal engineering, **80**(0), 49–67.

Anexo 1.Estado del arte

Libros

Dean & Dalrymple [5] Los doctores Dean y Dalrymple, de las Universidades de Florida y Delaware respectivamente, escriben este libro como una introducción a la teoría clásica de olas con un enfoque teórico riguroso. Haciéndolo una referencia clásica de la ingeniería hidráulica marítima.

Dévora & Sanz [7] Los Maestros Antonio Sánchez Dévora y Salvador Farreras arman este catálogo histórico de los tsunamis ocurridos en México. Aportando datos útiles para hacer algunas estadísticas básicas como fecha, intensidad sísmica, máximo de altura de ola registrado y localidad de origen. Dando cuenta de más de 60 tsunamis considerables y documentados entre los años 1787 a 2003.

Haykin [8] Éste es uno de los libros de texto de referencia obligada y más popular en los trabajos relacionados con redes neuronales artificiales e inteligencia artificial. Abarca tópicos con ejemplos concretos como: proceso de aprendizaje, perceptrones de una y multicapa; además del procesamiento temporal con redes tipo *feedforward*.

Masters [12] El Dr. Timothy Masters se especializa en estadística matemática ycómputo numérico. Este libro es la referencia obligada para el programadorexperto que quiere “reinventar la rueda” al aplicar técnicas de algoritmos deredes neuronales artificiales en la solución de problemas, pero con controldetallado de la totalidad del código.

Turban et al. [16] El Dr. Efraim Turban de la Universidad de California en Berkeley y su equipo presentan un extenso panorama de casos reales de CómputoAplicado, particularmente de sistemas expertos e inteligencia artificial. Acompañadosde breves fundamentos teóricos y guías para la implementación de estas nuevastecnologías. De especial interés son los capítulos dedicados a inteligenciaartificial y redes neuronales artificiales.

Artículos

Cybenko [4] El Dr. George Cybenko, del Centro de Investigación y Desarrollo de Supercómputo de la Universidad de Illinois EEUU, demuestra con formalismo topológico que una combinación lineal finita de funciones sigmoidales, como las usadas como funciones de activación en las redes neuronales artificiales, poder aproximar uniformemente cualquier función de n variables reales. Y como corolario: una red de dos capas puede aproximar cualquier región de decisión en precisión arbitraria.

Deo [6] El Profesor Deo, del Indian Institute of Technology de Bombay, hace en su artículo una introducción básica de los fundamentos teóricos que describen el funcionamiento de una red neuronal artificial; procediendo con un recuento de algunas de las aplicaciones que se han dado a estas herramientas de cómputo en el campo de la Ingeniería de costas y marítima. Destacándose la predicción e interpolación de parámetros de oleaje, cálculo de cargas en estructuras de protección marítima y respuesta de las mismas. Advierte de la necesidad de profundizar en la calibración y optimización del proceso de aprendizaje de la red para tener resultados coherentes y efectivos. Mencionando la ventaja de involucrar otras tecnologías además de la sola red, como la lógica difusa y los algoritmos genéticos.

Mandal et al. [11] En este artículo el Dr. Mandal junto con los investigadores del National Institute of Technology Karnataka en India: Dr. Sanjai Patil, Dr. Manhunatha y Dr. Hedge, hacen un recuento del conocimiento básico e introductorio de las bondades y del funcionamiento de una red neuronal *feedforward* y de algunas de las aplicaciones que se han llevado a cabo en el campo de la Ingeniería de Costas. Resaltando: la predicción de parámetros de oleaje, predicción de mareas y daños a estructuras costeras. Donde los resultados predichos por las redes neuronales artificiales superan en precisión a los modelos tradicionales empleados para la solución de esos problemas.

Warner & Misra [18] Los Profesores Brad Warner de la United States Air Force Academy y Manavendra Misra de la Colorado School of Mines, hacen un detallado análisis teórico del funcionamiento de las redes neuronales f*eedforward*. Comparándolo con el método estadístico de regresión multivariable, demostrando las fuertes similitudes en sus fundamentos y un desempeño muy similar entre ambas metodologías. Dicho parecido se hace más cercano cuando ambas se aplican a tareas de predicción e identificación de patrones.

Zanuttigh et al. [19] El trabajo que presentan es una extensión de la famosa herramienta para el cálculo de *overtopping* del proyecto CLASH. Este artículo detalla la implementación de una red neuronal artificial para la predicción de coeficientes de reflexión de olas para una gran variedad de estructuras de protección. La red se entrenó y validó con una enorme base de datos con 6×10^3 casos registrados. Lograron una alta precisión de la red, con un rmse de apenas 0.04.

Misceláneos

Analytics [1] Documentación del paquete R que permite que tareas básicas de cómputo paralelo sean llevadas a cabo desde la plataforma R, permitiendo el uso intensivo de todos los núcleos de los nuevos procesadores de 64-bit multinúcleo como el Intel ®Core i5 y Core i7, o su competidor AMD®.

Analytics & Weston [2] Documentación del paquete R que paraleliza tareas de cómputo que se puedan reducir a iteraciones independientes de un mismo ciclo.

Bivand et al. [3] Paquete que permite la interacción de R con las utilerías gdaldel sistema operativo. Para poder, así, ejecutar el catálogo de funciones, como porejemplo, el de conversión de coordenadas entre diferentes modelos cartográficos.

James & DebRoy [9] Paquete que conecta bases de datos relacionales MySQL®con la plataforma estadística R, para acceso y escritura de datos.

Limas et al. [10] Paquete creado para simulaciones que involucren redes neuronalesartificiales *feedforward* en R.

Montoya et al. [13] Estudio que aportó los datos sobre velocidades de corrienteslitorales en el interior del Puerto de Manzanillo en presencia de tsunamis,producto de un modelo hidráulico.

Orr [14] La Dra. Orr tiene un curso de redes neuronales artificiales desde 1999en la Universidad Willamette en Óregon, EEUU. Cuenta con amplio materialdidáctico disponible en línea, simulaciones y ejercicios. Abarca desde los tópicosbásicos, pasando por todos los detalles de la construcción de algoritmos quecomponen una red neuronal artificial, hasta los procedimientos para la optimización del funcionamiento de las redes, como por ejemplo, estrategias para evitar el*overfitting* y la no-convergencia.

R Core Team [15] Reconocido software estadístico de licencia gratuita y códigoabierto. Cuenta con diferentes extensiones que facilitan la programación y elanálisis en diversas áreas, entre ellas, inteligencia artificial y redes neuronalesartificiales.

Vasquez-Lopez [17] En esta tesis de grado el autor aplica una RNAf que disciernesatisfactoriamente las asociaciones ilegítimas que se dan entre caminatas aleatorias.Un tema de gran relevancia para el Cómputo Aplicado, pero también para laEconometría. Los algoritmos empleados se basan en la plataforma estadísticaR y el paquete AMORE.

Anexo 2

Coordenadas UTM de los 423 puntos de modelación.

```
## lon     lat     z
## 1 573199 2107250 15.274
## 2 573199 2107200 16.929
## 3 573199 2107150 17.300
## 4 573199 2107100 16.497
## 5 573249 2107250 12.082
## 6 573249 2107200 16.381
## 7 573249 2107150 17.185
## 8 573249 2107100 14.683
## 9 573299 2107250 12.549
## 10 573299 2107200 16.211
## 11 573299 2107150 16.390
## 12 573299 2107100 13.133
## 13 573349 2107250 15.034
## 14 573349 2107200 16.615
## 15 573349 2107150 16.448
## 16 573349 2107100 11.345
## 17 573399 2107250 14.897
## 18 573399 2107200 16.433
## 19 573399 2107150 16.294
## 20 573399 2107100 10.260
## 21 573449 2107250 14.943
## 22 573449 2107200 16.406
## 23 573449 2107150 15.459
## 24 573499 2107250 16.189
## 25 573499 2107200 16.591
## 26 573499 2107150 15.324
## 27 573549 2107250 16.330
## 28 573549 2107200 16.774
## 29 573549 2107150 15.219
## 30 573599 2107250 16.686
## 31 573599 2107200 17.114
## 32 573599 2107150 15.263
## 33 573649 2107250 15.642
## 34 573649 2107200 16.635
## 35 573649 2107150 14.799
## 36 573699 2107250 16.433
## 37 573699 2107200 16.993
## 38 573699 2107150 13.338
## 39 573749 2107250 16.777
## 40 573749 2107200 16.961
## 41 573749 2107150 16.562
## 42 573749 2107100 12.041
## 43 573799 2107250 16.743
## 44 573799 2107200 16.697
## 45 573799 2107150 16.577
## 46 573799 2107100 16.120
## 47 573799 2107050 15.277
## 48 573799 2107000 14.968
## 49 573799 2106950 14.937
## 50 573849 2107250 16.647
## 51 573849 2107200 17.007
## 52 573849 2107150 16.333
## 53 573849 2107100 15.748
## 54 573849 2107050 15.181
## 55 573849 2107000 15.043
## 56 573849 2106950 14.982
## 57 573849 2106900 14.998
## 58 573849 2106850 14.427
## 59 573849 2106800 11.490
## 60 573899 2107300 14.477
## 61 573899 2107250 16.986
## 62 573899 2107200 16.971
## 63 573899 2107150 16.315
## 64 573899 2107100 15.532
## 65 573899 2107050 15.145
## 66 573899 2107000 15.015
## 67 573899 2106950 15.043
## 68 573899 2106900 14.866
## 69 573899 2106850 14.693
## 70 573899 2106800 14.759
## 71 573949 2107300 13.002
## 72 573949 2107250 16.800
## 73 573949 2107200 17.168
## 74 573949 2107150 16.308
## 75 573949 2107100 15.359
## 76 573949 2107050 15.139
## 77 573949 2107000 14.652
## 78 573949 2106950 14.707
## 79 573949 2106900 14.843
## 80 573949 2106850 14.882
## 81 573949 2106800 14.426
## 82 573999 2107300 15.603
## 83 573999 2107250 16.765
```

```
## 84 573999 2107200 17.065
## 85 573999 2107150 16.462
## 86 573999 2107100 15.563
## 87 573999 2107050 15.196
## 88 573999 2107000 15.042
## 89 573999 2106950 14.725
## 90 573999 2106900 14.783
## 91 573999 2106850 15.161
## 92 573999 2106800 15.297
## 93 574049 2107300 16.626
## 94 574049 2107250 17.328
## 95 574049 2107200 16.771
## 96 574049 2107150 15.949
## 97 574049 2107100 15.515
## 98 574049 2107050 15.312
## 99 574049 2107000 15.142
## 100 574049 2106950 15.056
## 101 574049 2106900 15.272
## 102 574049 2106850 15.648
## 103 574049 2106800 16.068
## 104 574099 2107350 16.231
## 105 574099 2107300 16.369
## 106 574099 2107250 17.073
## 107 574099 2107200 16.722
## 108 574099 2107150 15.358
## 109 574099 2107100 15.379
## 110 574099 2107050 15.301
## 111 574099 2107000 15.221
## 112 574099 2106950 15.162
## 113 574099 2106900 15.433
## 114 574099 2106850 15.783
## 115 574099 2106800 15.851
## 116 574149 2107400 14.192
## 117 574149 2107350 16.554
## 118 574149 2107300 16.880
## 119 574149 2107250 17.365
## 120 574149 2107200 16.887
## 121 574149 2107150 15.544
## 122 574149 2107100 15.303
## 123 574149 2107050 15.203
## 124 574149 2107000 15.316
## 125 574149 2106950 15.359
## 126 574149 2106900 15.508
## 127 574149 2106850 15.560
## 128 574149 2106800 15.696
## 129 574199 2107450 13.859
## 130 574199 2107400 16.322
## 131 574199 2107350 16.487
## 132 574199 2107300 17.253
## 133 574199 2107250 17.304
## 134 574199 2107200 16.777
## 135 574199 2107150 15.473
## 136 574199 2107100 15.268
## 137 574199 2107050 15.302
## 138 574199 2107000 15.075
## 139 574199 2106950 15.152
## 140 574199 2106900 15.503
## 141 574199 2106850 15.939
## 142 574199 2106800 15.600
## 143 574249 2107550 10.128
## 144 574249 2107500 13.633
## 145 574249 2107450 16.242
## 146 574249 2107400 16.343
## 147 574249 2107350 16.789
## 148 574249 2107300 17.259
## 149 574249 2107250 17.088
## 150 574249 2107200 16.877
## 151 574249 2107150 15.500
## 152 574249 2107100 15.375
## 153 574249 2107050 15.190
## 154 574249 2107000 15.143
## 155 574249 2106950 15.294
## 156 574249 2106900 15.387
## 157 574249 2106850 15.852
## 158 574249 2106800 11.541
## 159 574299 2107600 8.808
## 160 574299 2107550 11.686
## 161 574299 2107500 16.407
## 162 574299 2107450 16.346
## 163 574299 2107400 16.299
## 164 574299 2107350 16.806
## 165 574299 2107300 16.611
## 166 574299 2107250 16.401
## 167 574299 2107200 15.752
## 168 574299 2107150 15.538
## 169 574299 2107100 15.509
## 170 574299 2107050 15.458
## 171 574299 2107000 15.346
## 172 574299 2106950 15.504
## 173 574299 2106900 15.571
## 174 574299 2106850 15.026
## 175 574349 2107650 8.119
## 176 574349 2107600 13.701
## 177 574349 2107550 16.094
## 178 574349 2107500 16.533
## 179 574349 2107450 16.436
## 180 574349 2107400 16.674
## 181 574349 2107350 16.722
## 182 574349 2107300 16.326
## 183 574349 2107250 16.474
## 184 574349 2107200 15.427
## 185 574349 2107150 15.133
## 186 574349 2107100 15.488
## 187 574349 2107050 15.589
## 188 574349 2107000 15.349
## 189 574349 2106950 15.467
## 190 574349 2106900 15.162
## 191 574399 2107700 12.746
## 192 574399 2107650 14.623
## 193 574399 2107600 16.515
## 194 574399 2107550 16.551
## 195 574399 2107500 16.678
## 196 574399 2107450 16.911
## 197 574399 2107400 16.899
```

```
## 198 574399 2107350 16.492
## 199 574399 2107300 16.514
## 200 574399 2107250 15.070
## 201 574399 2107200 14.946
## 202 574399 2107150 15.156
## 203 574399 2107100 15.640
## 204 574399 2107050 15.528
## 205 574399 2107000 15.251
## 206 574399 2106950 14.892
## 207 574399 2106900 14.632
## 208 574449 2107850 12.594
## 209 574449 2107800 14.560
## 210 574449 2107750 14.864
## 211 574449 2107700 15.922
## 212 574449 2107650 16.364
## 213 574449 2107600 16.540
## 214 574449 2107550 16.689
## 215 574449 2107500 16.885
## 216 574449 2107450 16.630
## 217 574449 2107400 16.356
## 218 574449 2107350 16.339
## 219 574449 2107300 16.351
## 220 574449 2107250 15.237
## 221 574449 2107200 15.037
## 222 574449 2107150 15.046
## 223 574449 2107100 15.270
## 224 574449 2107050 15.376
## 225 574449 2107000 14.711
## 226 574449 2106950 13.821
## 227 574499 2107950 12.906
## 228 574499 2107900 14.597
## 229 574499 2107850 16.144
## 230 574499 2107800 16.051
## 231 574499 2107750 16.140
## 232 574499 2107700 16.301
## 233 574499 2107650 16.819
## 234 574499 2107600 16.789
## 235 574499 2107550 16.568
## 236 574499 2107500 16.662
## 237 574499 2107450 16.211
## 238 574499 2107400 15.983
## 239 574499 2107350 16.069
## 240 574499 2107300 15.082
## 241 574499 2107250 15.286
## 242 574499 2107200 14.825
## 243 574499 2107150 13.571
## 244 574499 2107100 14.324
## 245 574549 2108100 14.368
## 246 574549 2108050 16.272
## 247 574549 2108000 16.287
## 248 574549 2107950 16.184
## 249 574549 2107900 16.149
## 250 574549 2107850 16.529
## 251 574549 2107800 16.454
## 252 574549 2107750 16.478
## 253 574549 2107700 16.265
## 254 574549 2107650 16.720
## 255 574549 2107600 16.603
## 256 574549 2107550 16.299
## 257 574549 2107500 16.141
## 258 574549 2107450 16.221
## 259 574549 2107400 15.616
## 260 574549 2107350 14.646
## 261 574549 2107300 15.320
## 262 574549 2107250 14.241
## 263 574299 2108750 11.665
## 264 574299 2108700 11.524
## 265 574299 2108650 10.793
## 266 574349 2108750 11.887
## 267 574349 2108700 12.083
## 268 574349 2108650 11.973
## 269 574349 2108600 11.757
## 270 574349 2108550 12.901
## 271 574349 2108500 8.753
## 272 574349 2108450 8.223
## 273 574399 2108750 11.982
## 274 574399 2108700 12.652
## 275 574399 2108650 15.510
## 276 574399 2108600 15.326
## 277 574399 2108550 15.286
## 278 574399 2108500 15.483
## 279 574399 2108450 15.548
## 280 574449 2108750 15.499
## 281 574449 2108700 15.749
## 282 574449 2108650 15.979
## 283 574449 2108600 15.918
## 284 574449 2108550 15.750
## 285 574449 2108500 15.641
## 286 574449 2108450 15.955
## 287 574449 2108400 15.501
## 288 574499 2108750 15.812
## 289 574499 2108700 15.769
## 290 574499 2108650 15.927
## 291 574499 2108600 16.058
## 292 574499 2108550 16.347
## 293 574499 2108500 16.310
## 294 574499 2108450 16.182
## 295 574499 2108400 16.196
## 296 574499 2108350 16.230
## 297 574549 2108750 16.054
## 298 574549 2108700 16.012
## 299 574549 2108650 16.119
## 300 574549 2108600 16.195
## 301 574549 2108550 16.765
## 302 574549 2108500 16.650
## 303 574549 2108450 16.668
## 304 574549 2108400 16.545
## 305 574549 2108350 16.295
## 306 574549 2108300 16.368
## 307 574599 2108750 16.266
## 308 574599 2108700 16.220
## 309 574599 2108650 16.529
```

```
## 310 574599 2108600 16.362
## 311 574599 2108550 16.432
## 312 574599 2108500 16.625
## 313 574599 2108450 16.625
## 314 574599 2108400 16.142
## 315 574599 2108350 16.456
## 316 574599 2108300 16.643
## 317 574599 2108250 16.574
## 318 574599 2108200 16.755
## 319 574599 2108150 16.378
## 320 574599 2108100 16.286
## 321 574599 2108050 16.425
## 322 574599 2108000 16.765
## 323 574599 2107950 16.509
## 324 574599 2107900 16.315
## 325 574599 2107850 16.449
## 326 574599 2107800 16.537
## 327 574599 2107750 16.477
## 328 574599 2107700 16.646
## 329 574599 2107650 16.110
## 330 574599 2107600 15.705
## 331 574599 2107550 15.788
## 332 574599 2107500 15.920
## 333 574599 2107450 12.022
## 334 574649 2108750 16.065
## 335 574649 2108700 16.350
## 336 574649 2108650 16.666
## 337 574649 2108600 16.853
## 338 574649 2108550 16.407
## 339 574649 2108500 16.625
## 340 574649 2108450 16.526
## 341 574649 2108400 16.323
## 342 574649 2108350 16.611
## 343 574649 2108300 16.933
## 344 574649 2108250 16.552
## 345 574649 2108200 16.680
## 346 574649 2108150 16.419
## 347 574649 2108100 16.687
## 348 574649 2108050 16.607
## 349 574649 2108000 16.816
## 350 574649 2107950 16.791
## 351 574649 2107900 16.307
## 352 574649 2107850 16.260
## 353 574649 2107800 16.314
## 354 574649 2107750 16.053
## 355 574649 2107700 15.955
## 356 574649 2107650 15.719
## 357 574649 2107600 15.036
## 358 574699 2108750 16.061
## 359 574699 2108700 16.078
## 360 574699 2108650 16.110
## 361 574699 2108600 16.279
## 362 574699 2108550 16.227
## 363 574699 2108500 16.547
## 364 574699 2108450 16.147
## 365 574699 2108400 16.396
## 366 574699 2108350 16.479
## 367 574699 2108300 16.676
## 368 574699 2108250 16.864
## 369 574699 2108200 16.516
## 370 574699 2108150 16.549
## 371 574699 2108100 16.465
## 372 574699 2108050 16.414
## 373 574699 2108000 16.545
## 374 574699 2107950 16.444
## 375 574699 2107900 16.110
## 376 574699 2107850 15.749
## 377 574699 2107800 14.470
## 378 574749 2108750 15.988
## 379 574749 2108700 16.021
## 380 574749 2108650 16.237
## 381 574749 2108600 16.290
## 382 574749 2108550 16.271
## 383 574749 2108500 16.369
## 384 574749 2108450 16.615
## 385 574749 2108400 16.746
## 386 574749 2108350 16.390
## 387 574749 2108300 16.697
## 388 574749 2108250 16.517
## 389 574749 2108200 16.327
## 390 574749 2108150 16.580
## 391 574749 2108100 16.766
## 392 574749 2108050 16.695
## 393 574749 2108000 16.403
## 394 574749 2107950 9.743
## 395 574799 2108750 15.782
## 396 574799 2108700 15.757
## 397 574799 2108650 16.424
## 398 574799 2108600 16.391
## 399 574799 2108550 16.188
## 400 574799 2108500 16.429
## 401 574799 2108450 16.640
## 402 574799 2108400 16.639
## 403 574799 2108350 16.696
## 404 574799 2108300 16.764
## 405 574799 2108250 16.533
## 406 574799 2108200 16.412
## 407 574799 2108150 15.876
## 408 574849 2108750 15.855
## 409 574849 2108700 15.896
## 410 574849 2108650 16.050
## 411 574849 2108600 16.268
## 412 574849 2108550 16.398
## 413 574849 2108500 16.248
## 414 574849 2108450 16.461
## 415 574849 2108400 16.383
## 416 574849 2108350 16.442
## 417 574849 2108300 15.234
## 418 574899 2108750 16.048
## 419 574899 2108700 16.653
## 420 574899 2108650 16.417
## 421 574899 2108600 16.686
## 422 574899 2108550 16.534
## 423 574899 2108500 16.139
```

INSTITUTO MEXICANO DEL TRANSPORTE

SCT
SECRETARÍA DE COMUNICACIONES Y TRANSPORTES

Carretera Querétaro-Galindo km 12+000
CP 76700, Sanfandila
Pedro Escobedo, Querétaro, México
Tel +52 (442) 216 9777 ext. 2610
Fax +52 (442) 216 9671

publicaciones@imt.mx

http://www.imt.mx/