UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MODELADO DE TRÁFICO VEHICULAR MEDIANTE REDES NEURONALES RECURRENTES

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN INGENIERÍA
(COMPUTACIÓN)

P R E S E N T A :

BEATRIZ PERALTA CORTÉS

DIRECTOR DE TESIS: DR. LUIS A. ÁLVAREZ ICAZA LONGORIA

México, D.F. agosto 2008.

A todas las personas que han permanecido cerca
y han sido testigos de esta aventura:
mis padres, mis hermanos y
mis amigos.

Agradecimientos

Agradezco el apoyo del personal académico y administrativo que labora en el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas de la UNAM, en especial a los miembros del Departamento de Ingeniería de Sistemas Computacionales y Automatización (DISCA) por brindarme su apoyo académico y moral para llevar a buen termino esta tesis.

También le doy las gracias al personal académico del Instituto de Investigaciones Sobre la Universidad y la Educación de la UNAM por su comprensión y apoyo para hacer posible esta maestría. En especial, al Dr. Enrique Ruiz-Velasco y a Armando Torres.

De manera especial agradezco al Dr. Luis Álvarez Icaza Longoria quien me guío en la realización de mi tesis como asesor. Le doy las gracias porque siempre tuvo la paciencia y la comprensión necesaria para responder mis dudas, cuestionamientos e inquietudes emanadas durante la realización de esta tesis.

Hago una mención muy especial al Dr. Héctor Benítez Pérez quien además de brindarme su amistad, me motivo a realizar esta maestría y me brindo un espacio en DISCA. Le agradezco de todo corazón su paciencia, su confianza y su tiempo, pero sobre que es un buen amigo.

Asimismo, expreso mi gratitud a las doctoras María Elena Lárraga y Angélica Del Rocío Lozano por la lectura, los comentarios, sus observaciones y las correcciones a este trabajo de investigación.

A mis padres, no tengo palabras para agradecer que me hayan dado la vida, la libertad y la confianza que han depositado en mi. Los amos, muchas gracias.

A mis hermanos: Salvador y Fidel que sin quererlo has sido un aliciente para llevar a cabo esta maestría y muchos otros proyectos, gracias desde el fondo de mi corazón.

Magali gracias por ser la fortaleza y el ejemplo de mujer trabajadora, luchona, capaz de reconocer que a veces es necesario retroceder para tomar impulso y continuar. Gracias porque siempre has estado cerca, me has escuchado y has sido cómplice de este y muchos otros proyectos y locuras, gracias mil.

Carlitos muchas gracias por tu amistad, por estar cerca en los momentos más difíciles, por escucharme y brindarme tu comprensión y apoyo siempre. Gracias por ser un hermano y confidente, te quiero mucho.

Miguel muchas gracias por tu amistad, comprensión y apoyo. Te agradezco de todo corazón el tiempo que me dedicaste y el impulso que me diste para finalizar esta tesis.

Gracias a Diana, Fernando, María Luisa, Paty Uriostegui, Alondra, Rebeca, Gisell y a todos los amigos y familiares por estar siempre conmigo, acompañandome no sólo en este proceso y de quienes he recibido palabras de aliento que me motivan e impulsan a continuar cuando lo he necesitado.

A mis compañeros Gaby, Carlos y Moisés muchas gracias por su paciencia, comprensión, pero sobre todo por ser mis cómplices en esto.

Para terminar, doy gracias a papá Dios por iluminar mi camino y permitirme concliur este sueño.

Índice general

Capítulo 1

Introducción

Aunque las autopistas fueron concebidas para proporcionar movilidad ilimitada a los usuarios. Actualmente, esta movilidad se ha restringido seriamente debido a que el número de vehículos que transita por ellas se ha incrementado [Papageorgiou et al., 2002]. Esto ha motivado que se realicen trabajos para modelar el tráfico vehicular utilizando diferentes técnicas, con el propósito de auxiliar en el diseño de sistemas de control para mejorar la eficiencia en el uso de estas autopistas. Sin embargo, a pesar del extenso trabajo realizado, el modelado de tráfico vehicular sigue presentando problemas abiertos.

El tráfico en autopistas normalmente se mide mediante detectores de paso sencillos o dobles colocados en el pavimento, que arrojan datos de la acupación o flujo vehicular y sobre la longitud efectiva de los vehículos. Estas mediciones se pueden convertir en cantidades macroscópicas como la densidad y la velocidad vehicular [Muñoz et al., 2003].

Una forma de modelar el tráfico consiste en definir celdas que representen fragmentos de longitud conocida de la autopista. Con ello se puede analizar el comportamiento del tráfico, la influencia de rampas de entrada y salida, la densidad y el flujo de las diferentes celdas y observar así cambios discretos en secciones de interés de la carretera [Daganzo, 1994].

Existen propuestas que utilizan herramientas de sistemas adaptables para llevar a cabo el modelado de tráfico vehicular. En particular, resulta de interés encontrar descripciones autoajustables en función de mediciones del flujo, la acupación o la velocidad, que describan las relaciones entre estas variables. La mayor parte de los modelos existentes se ajustan de manera empírica y son difíciles de calibrar.

Las redes neuronales recurrentes son útiles para modelar fenómenos dinámicos no lineales [Fausett, 1994, Hassoon, 1995, Haykin, 1999, Nelles, 2001] y no han sido aplicadas para representar el tráfico vehicular. Esta tesis plantea explorar su uso para representar relaciones flujo-densidad.

1.1. Hipótesis

- En esta tesis se considera que el tráfico vehicular se comporta según un modelo macróscopico donde las variables importantes son la densidad, la velocidad y el flujo vehicular.

- Se parte del supuesto de que el problema global de asignación de tráfico en una red de autopistas ha sido resuelto previamente mediante la aplicación de herramientas matemáticas apropiadas, en particular aquellas relacionadas con la programación dinámica y la teoría de colas [1] .

- Como en la operación de cualquier política global de asignación de tráfico en una red de autopistas se presentan perturbaciones que devían el estado de la red de su condición óptima, se requieren algoritmos de control de acceso local que regulen los valores de la densidad y el flujo en la vecindad de una rampa de acceso alrededor de los valores óptimos.

- La relación flujo-densidad tiene una dinámica interna no lineal y su comportamiento es muy importante en las estrategias locales de control.

- El conocimiento de esta representación flujo-densidad es útil en aplicaciones que persiguen control local en rampas de acceso.

- Las redes neuronales recurrentes son adecuadas para representar fenómenos dinámicos no lineales, en particular son útiles para representar la relación entre flujo y densidad.

[1] No se encontró en la literatura revisada [Kleinrock, 1975, Di Fabbraro et al., 1995, Zhang and Recker, 1999, Ziliaskopoulos, 2000, Chang and Li, 2002], ni al entrevistarse con expertos en tráfico, que la teoría de colas se utilizara con frecuencia en la solución de los problemas globales de asignación de tráfico.

1.2. Objetivo

- En esta tesis se propone aplicar una red neuronal recurrente al modelado de relaciones flujo-densidad en una sección finita de una autopista.

- El trabajo plantea comparar esta forma de modelado con datos de tráfico obtenidos de algunas ciudades que poseen sistemas de medición en línea de tráfico vehicular y verificar la calidad de la aproximación con datos reales de tráfico.

1.3. Alcances

El modelo de redes neuronales recurrentes para representar la relación flujo-densidad se empleará en un grupo de secciones de una autopista que cuente con los sensores apropiados para proporcionar la información de entrada a la red neuronal. No se plantea aplicar éste modelo en una red de autopistas.

El trabajo no considera el manejo local de colas que puedan ocurrir al aplicar una estrategia local de regulación. Se supone que el manejo de estas colas ha sido resuelto con una estrategia global de asignación de tráfico [Zhang and Recker, 1999, Hegyi et al., 2002].

1.4. Relevancia

La congestión recurrente en las autopistas tiene importantes impactos sociales, económicos y ambientales. Mejorar el uso de las vías de transporte requiere contar con modelos dinámicos de tráfico vehicular. Con este trabajo se pretende obtener descripciones auto-ajustables en función de mediciones de flujo, acupación o velocidad que ajuste datos reales de tráfico vehicular. Es de especial interés que estos modelos sean auto-ajustables para evitar la necesidad de calibrarlos frecuentemente. En este sentido el trabajo de tesis contribuye al desarrollo de herramientas analíticas para auxiliar en el diseño y control de autopistas y vías rápidas.

1.5. Metas

Contar con un modelo de red neuronal recurrente que sea capaz de representar la relación flujo-densidad del tráfico vehicular y realizar una serie de simulaciones numéricas que sirvan para comparar los resultados obtenidos con datos reales de tráfico.

1.6. Metodología

Se atacará el problema de modelado de una sección de autopista a través de una red neuronal recurrente y se considerará que los datos se ajustarán en función de mediciones de acupación para obtener como salida funciones de flujo o velocidad. Además, se deberán comparar las salidas de las redes neuronales con datos de tráfico obtenidos de campo, para ello se realizarán las siguientes actividades:

- Se estudiará el modelo de transmisión por celdas y algunos modelos derivados de él como medio para representar el comportamiento macroscópico del tráfico vehicular.
- Se hará una revisión de los modelos de redes neuronales para determinar una topología apropiada.
- Se representará los datos reales de un grupo de secciones de una autopista en la topología seleccionada.
- Se probará por simulación la calidad de la aproximación de la red neuronal recurrente a los datos reales y se propondrán los ajustes necesarios en la topología seleccionada.

1.7. Organización de la tesis

Esta tesis se desarrolla de la siguiente manera: el capítulo 2 describe los conceptos básicos de tráfico vehicular. El capítulo 3 contiene las descripción de las redes neuronales recurrentes y sus algoritmos de aprendizaje. El capítulo 4 describe el desarrollo del modelado del tráfico vehicular mediante redes neuronales recurrentes, las observaciones y los resultados obtenidos. Por último el capítulo 5 incluye las conclusiones y el trabajo futuro.

Capítulo 2

Tráfico vehicular

2.1. Introducción

Las autopistas fueron concebidas originalmente para proporcionar movilidad ilimitada a los usuarios que viajan en automóviles. Desafortunadamente con el paso del tiempo, la cantidad de vehículos que transita por ellas ha ido incrementándose, a tal grado que actualmente las autopistas se ven afectadas por el congestionamiento de automóviles, el cual forma colas que puede alcanzar muchos kilómetros. Debido a esto se han realizado numerosos trabajos de investigación para mejorar la movilidad.

Aunque los modelos de asignación de tráfico dinámicos involucran niveles de planeación en grandes redes, típicamente organizan los niveles de tráfico a muchos destinos, sus modelos están basados en relaciones de flujo simple que no son perfectamente consistentes con las leyes de conservación del tráfico [Daganzo, 1994].

Los modelos de operación de tráfico pueden ser micróscopicos, macroscópicos o mesoscópicos. La simulación micróscopica supone que el comportamiento individual de un vehículo está en función de las condiciones del tráfico en su ambiente. Estos modelos toman en cuenta el comportamiento individual de un vehículo y su trayectoria de tal manera que el modelado se puede realizar, por ejemplo, con autómatas celulares, cuya cualidad esencial es que tienen como unidad de simulación los vehículos individuales. Aunque las simulaciones micróscopicas usualmente mantienen la ruta hacia el destino de cada vehículo, sus suposiciones son difíciles de validar por el comportamiento de los humanos en el tráfico real.

El modelo macróscopico, por otro lado, se basa en el comportamiento de un conjunto de vehículos, el cual es más fácil de observar y validar, dependiendo de las

condiciones del tráfico en su ambiente. La mayoría de estos modelos se basan en la teoría hidrodinámica del flujo de tráfico y generalmente no distinguen sus componentes de flujo por el destino, cada flujo se trata como uno solo por simplicidad. Cuando el flujo alcanza una bifurcación en la carretera o algún otro punto donde hay que escoger una ruta, el modelo usualmente especifica proporciones fijas de distribución o fija los flujos de salida. En realidad, los flujos dependen de los destinos de los vehículos que alcanzan las bifurcaciones en cuestión [Daganzo, 1994].

Los modelos macroscópicos utilizan secciones de autopistas que van de los 100 a los 10,000 metros e intentan modelar densidad (ρ) y velocidad (v) de manera agregada. Existen dos clasificaciones: de primero y segundo orden. En el grupo de modelos de primer orden se estudia la conservación de la densidad (ρ) y la relación entre velocidad-densidad (v-ρ). En el grupo de segundo orden, la conservación de densidad (ρ) y la dinámica de la velocidad v.

Los primeros trabajos sobre soluciones a modelos hidrodinámicos se desarrollaron por Vaughan, Hurdle y Hauer [Vaught et al., 1984] quienes formularon el problema para una red simple, la cual consiste de una serie de secciones conectadas. La solución a las ecuaciones diferenciales resultantes usa el método clásico de características para el modelo hidrodinámico del flujo de tráfico. Newell propone un método de solución para el modelo hidrodinámico de una sección intentando resolver el problema para múltiples secciones [Newell, 1993]. El método usado en este trabajo predice la variación del flujo de tráfico al final de la sección, sin evaluar el comportamiento en puntos intermedios [Daganzo, 1994].

Los modelos mesoscópicos tratan de estar entre los dos modelos anteriores generalmente se refieren a tipos de vehículos y rutas de origen destino.

Entre los modelos macroscópicos se encuentra el modelo de transmisión por celdas, el cual se tomó como base para este trabajo.

2.2. Modelo de Transmisión por Celdas

En esta sección se describe un modelo de tráfico sobre una carretera en donde los vehículos entran por un lado y lo dejan por el otro, es decir, la carretera no tiene ninguna entrada o salida intermedia. Dicha carretera está dividida en secciones homogéneas llamadas *celdas*, numeradas consecutivamente desde $i = 1$ hasta I. La longitud de las celdas, L_i, está basada en la distancia que recorre un vehículo a velocidad de flujo libre en un intervalo de tiempo determinado por una señal de reloj. Además, se considera a la carretera como equivalente a una de un solo

carril donde el número de vehículos n_i que contiene por unidad de longitud proporciona la densidad vehicular promedio en la sección como $k_i = n_i/L_i$. La velocidad de flujo libre se considera constante, pues se ha observado empíricamente que la velocidad media-espacial del tráfico en una autopista permanece relativamente constante, independiente del flujo, al menos que éste exceda la capacidad máxima [Daganzo, 1994].

De esta forma, todos los vehículos que viajan a velocidad v en una celda se supone que avanzan a la siguiente celda en un intervalo de tiempo t constante. Así, no es necesario conocer en cuál de las celdas están localizados. Entonces, el sistema evoluciona bajo la siguiente ecuación:

$$n_{i+1}(t+1) = n_i(t) \tag{2.1}$$

donde $n_i(t)$ es el número de vehículos en la celda i al tiempo t. La recursión se mantiene para todo el flujo, al menos que el tráfico sea lento al presentarse congestionamiento. Esto sucede generalmente durante las horas pico, generando retrasos que se atribuyen a las colas formadas por embotellamientos, donde el flujo temporal excede la máxima capacidad, más que cualquier dependencia entre el flujo y la velocidad.

Para incorporar las colas formadas por embotellamientos se define como $N_i(t)$ el máximo número de vehículos que puede contener la celda i en el tiempo t y como $Q_i(t)$ al máximo número de vehículos que pueden fluir en la celda i en un intervalo de tiempo t a $t+1$. La primer constante es el producto de la longitud de la celda y su "densidad de embotellamiento", mientras que la segunda es el mínimo de la "capacidad de flujo" de la celda $i-1$ a la i. A esto se le conoce como "capacidad" de la celda i.

De acuerdo a con las definición anterior, la cantidad de vehículos que puede fluir de la celda $i-1$ a i en un intervalo de tiempo de t a $t+1$, $y_i(t)$, es la menor de las siguientes cantidades:

- $n_{i-1}(t)$, el número de vehículos en la celda $i-1$ al tiempo t,
- $Q_i(t)$, la capacidad del flujo dentro de i para el intervalo t, y
- $N_i(t) - n_i(t)$, la cantidad de espacio vacío en la celda i al tiempo t.

La última cantidad asegura que la densidad vehicular sobre cualquier celda de la autopista permanece por debajo de la densidad de embotellamiento. Este modelo

se basa en la conservación de vehículos, así la ocupanción de la celda i al tiempo $t+1$ es igual a su ocupanción al tiempo t, más el flujo de entrada menos el flujo de salida; es decir:

$$n_i(t+1) = n_i(t) + y_i(t) - y_{i+1}(t) \tag{2.2}$$

donde los flujos están relacionados con las condiciones actuales al tiempo t como se indica a continuación:

$$y_i(t) = min\{n_{i-1}(t), Q_i(t), N_i(t) - n_i(t)\} \tag{2.3}$$

Esta ecuación actualiza la ocupanción de las celdas en cada intervalo de tiempo.

Se deben establecen condiciones de frontera a la entrada y salida del conjunto de celdas de manera que la ecuación (2.3) se tiene que modificar para tomar en cuenta las siguientes consideraciones. Las celdas de salida o destino del tráfico pueden tener un tamaño infinito ($N_{i+1} = \infty$), mientras que una celda fuente u origen tiene un número infinito de vehículos ($n_\infty(0) = \infty$) que entran a una celda vacía "0" de tamaño infinito, $N_0(t) = \infty$. La capacidad del flujo de entrada $Q_0(t)$ a la primera celda es igual al flujo de entrada del link deseado para el intervalo de tiempo $t+1$ y actúa como un dispositivo que lanza el flujo a la tasa deseada.

El número de vehículos que entra a la celda no tiene relación directa con el número de vehículos que pueden salir. Sólo las condiciones actuales afectan el flujo de entrada a la celda. Aunque las primeras dos restricciones ($y_i(t) \leq n_{i-1}(t)$ y $y_i(t) \leq Q_i(t)$) son razonables, se puede argumentar que si una celda empieza a vaciarse rápidamente entonces la tercera restricción, la ocupanción, $y_i(t) \leq N_i(t) - n_i(t)$ puede ser innecesaria.

2.2.1. Equivalencia con el modelo hidrodinámico

Las ecuaciones (2.2) y (2.3) son aproximaciones discretas del modelo hidrodinámico de Lighthill, Whitham, Richards (1995) con una relación densidad-flujo $(k-q)$ en la forma de un triángulo isósceles [Daganzo, 1994], como se muestra en la figura 2.1

Este modelo, además de considerar la velocidad de flujo libre v que representa una onda que viaja en el sentido y la rapidez de los vehículos en condiciones libres de congestionamiento, toma en cuenta la velocidad conocida como onda de retroceso, (w), la cual tiende a desplazarse en el sentido contrario al flujo del tráfico.

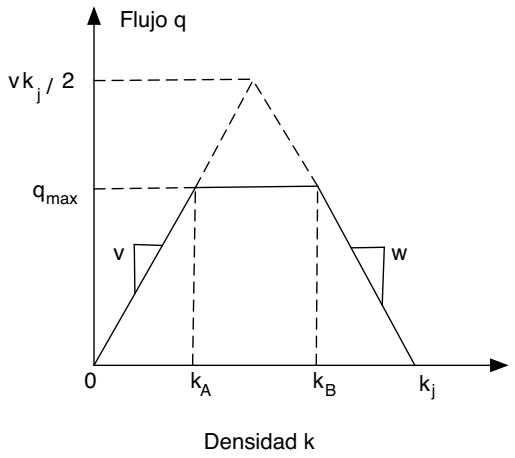


Figura 2.1: Diagrama relación Densidad-Flujo

Esta relación se expresa como:

$$q = min\{vk, q_{max}, w(k_j - k)\}, \text{ para } 0 \leq k \leq k_j \tag{2.4}$$

donde v es la velocidad de flujo libre (y la velocidad de todos los movimientos de onda hacia atrás) k_j es la densidad de embotellamiento y $q_{max} \leq k_j v/2$ es el flujo máximo. Si la ecuación (2.4) se remplaza en la ecuación de conservación:

$$\partial q(x,t)/\partial x = -\partial k(x,t)/\partial t, \tag{2.5}$$

se obtiene:

$$\partial(min\{vk(x,t), q_{max}, w(k_j - k(x,t))\})/\partial x = -\partial k(x,t)/\partial t \tag{2.6}$$

Se desea mostrar que la ecuación (2.6) es equivalente a la ecuación (2.3). Observe que para una autopista homogénea, las características de las celdas son independientes de i y t : $N_i(t) = N$ y $Q_i(t) = Q$. Para mostrar la equivalencia de las aproximaciones discretas y continuas, se define una señal de reloj igual a dt y se escoge una unidad de distancia tal que $vdt = 1$. Entonces la longitud de las celdas es 1, v es también 1 y la siguiente equivalencia se mantiene: $x = 1$, $k_i = N$, $q_{max} = Q$

y $k(x,t) = n_i(t)$. Con esta convención para las variables en los paréntesis de la ecuación 2.6 esta es equivalente a

$$min\{n_i(t), Q, N - n_i(t)\}, \tag{2.7}$$

la cual coincide con la definición de $y_{i+1}(t)$ de la ecuación (2.3), excepto para el subíndice de n en el último término que es $i+1$ en lugar de i. Esto, sin embargo, no afecta al menos que la densidad no sea continua. Podemos indicar que la variable en los paréntesis de $y_{i+1}(t)$, como un resultado del lado izquierdo de la ecuación 2.5 es aproximado por

$$y_{i+1} - y_i(t). \tag{2.8}$$

mientras que el lado derecho de la ecuación (2.5) por

$$-[n_i(t+1) - n_i(t)], \tag{2.9}$$

y la igualdad de estas dos cantidades justifica la recursión de la ecuación (2.2).

Otra forma de verificar la consistencia de la ecuación (2.3) con la ecuación (2.6) toma en cuenta el flujo en la celda i, entonces $y_i(t)$ se puede especificar como una función del flujo que se envía desde la celda $i-1$ y la ecuación (2.3) queda como:

$$y_i(t) = min\{n_{i-1}, Q_i(t), N_{i-1}(t) - n_{i-1}(t)\}. \tag{2.10}$$

Alternativamente, $y_i(t)$ se especifica como una función de la ocupanción en la celda que recibe el flujo . Mientras que estas y otras especificaciones son consistentes con la ecuación (2.6) cualquier variación pequeña de la densidad no es equivalente en su comportamiento durante la fase de discontinuidades.

La consistencia de la ecuación (2.3) con la ecuación (2.6) no garantiza un comportamiento razonable a través de discontinuidades. Con esta expresión se puede predecir el flujo nulo que se produce cuando la demanda llega a la densidad de embotellamiento. Así, el modelo indica que la cola de vehículos no puede avanzar inmediatamente después que se ha quitado un obstáculo.

Como las discontinuidades son comunes y permanecen espontáneamente cuando baja la densidad del tráfico, los efectos del modelo permanecen y deben ser examinados. La ecuación (2.3) puede replicar el comportamiento del modelo continuo a través de discontinuidades al iterar la ecuación (2.1), entonces automáticamente varía la densidad y se produce una onda de choque. Está propiedad es muy útil para cálculos automáticos.

2.2.2. Formación de colas de tráfico

En una autopista, los espacios son susceptibles de ser ocupados durante un proceso de congestionamiento, ya que se van llenando hacia atrás hasta que llegará el momento en que todas las celdas estarán ocupadas completamente. Sin embargo, si el flujo de entrada $Q_e(t)$ sigue mandando vehículos en esta situación, entonces comenzará a existir una acumulación de los mismos antes de poder entrar a la primera de las celdas del modelo, esto comenzará a formar una cola de automóviles que requieren entrar al sistema pero no pueden hacerlo. Dicha acumulación varía en el tiempo según

$$c(t+1) = c(t) + [Q_e(t+1) - y_1(t)] \tag{2.11}$$

de tal manera que cuando el flujo de entrada $Q_e(t)$ es mayor que el flujo calculado $y_1(t)$ que puede entrar a la primera celda, entonces la cola de vehículos $c(t)$ aumenta; si, por el contrario, el flujo Q_eT decrece hasta ser menor que $y_1(t)$, entonces $c(t)$ también decrecerá. Esta última condición puede llevar a $c(t)$ a un valor negativo, lo cual no ocurre en la realidad, por lo que es necesario imponer la condición: Si $c(t) \leq 0$ entonces $c(t) = 0$

Para tomar en cuenta los vehículos que requieren entrar al sistema, es necesario modificar nuevamente el cálculo del flujo a la entrada entonces se tiene

$$y_1(t) = min\{c(t) + Q_e(t+1), Q_i(t+1), [N_i(t) - n_i(t)]\} \tag{2.12}$$

Con lo cual, si no existe el espacio para vaciar una parte de la cola, ésta crecerá aún más; si por el contrario, es posible dar cabida a una fracción de esos vehículos acumulados, entonces se incorporarán al contingente de los que circulan por el sistema.

En este capítulo se describieron los conceptos básicos de tráfico vehicular, lo cuales se emplearan para describir el problema fundamental de este trabajo de tesis.

Capítulo 3

Redes Neuronales Artificiales

3.1. Introducción

Una Red Neuronal Artificial es un sistema de procesamiento de información que está formado por elementos no-lineales llamados neuronas. Son modelos matemáticos aplicables para resolver una amplia variedad de problemas, tales como clasificación de patrones, reconocimiento y síntesis de lenguaje, funciones de aproximación, modelado de sistemas no lineales y de control, compresión de imágenes, memoria asociativa, agrupamientos, etc. [Caudill and Butler, 1992, Hassoon, 1995, Hilera and Martínez, 1995, Luo and Unbehauen, 1997].

Las Redes Neuronales Artificiales intentan reproducir de forma simplificada el funcionamiento de un cerebro [Corchado et al., 2000]. No intentan reproducir todo el cerebro humano, sino que se centran en mecanismos de resolución de problemas individuales. Por ejemplo, tratan de imitar el proceso de almacenar información en patrones y utilizar tal información para resolver cierto tipo de problemas.

Las Redes Neuronales Artificales proporcionan un esquema de procesamiento alternativo basado en la operación de un número determinado de neuronas que se conectan entre sí. Un atributo significativo de las neuronas es su habilidad para aprender, interactuando con el medio en el que se encuentran o con información de entrada. La información de entrada se puede procesar en un solo sentido o realimentarse formando uno o más ciclos de entrada. A las redes que forman ciclos se le denomina Redes Neuronales Recurrentes o simplemente redes recurrentes.

Las redes recurrentes no se deben confundir con las redes de propagación hacia atrás, donde sólo el error se propaga hacia las capas anteriores. Algunos ejem-

plos de redes recurrentes son la red de Hopfield y las redes de memoria asociativa [Caudill and Butler, 1992].

En este capítulo se presenta la descripción de las redes neuronales recurrentes, que se utilizarán más adelante para modelar el tráfico vehicular en una autopista.

3.2. Redes Neuronales Recurrentes

Las redes neuronales recurrentes son redes neuronales con uno o más ciclos de realimentación. Es decir, cada patrón de entrada pasa a través de la red más de una vez antes de que se genere una salida. La realimentación puede ser local o global.

La realimentación global en un perceptrón multicapa puede tomar una gran variedad de formas, por ejemplo la salida de las neuronas en la capa de salida puede realimentar la entrada de las neuronas en la capa de entrada o la salida de las neuronas de la capa oculta pueden realimentar a las neuronas de la capa de entrada. Si el perceptrón tiene más de una capa oculta, entonces la realimentación puede tomar muchas formas.

Básicamente, las redes recurrentes tienen dos funcionalidades: a) ser memorias asociativas y b) realizar mapeos entrada-salida.

Por definición, el espacio de entrada en una red se mapea en un espacio de salida. Para este tipo de aplicaciones, la red recurrente responde temporalmente a la aplicación externa de una señal de entrada, pero además se considera que con la aplicación de la realimentación adquiere una representación de estados y un manejo dinámico que la hace útil para diversas aplicaciones como predicciones no lineales y modelado, procesamiento de lenguaje, plantas de control y diagnóstico [Haykin, 1999].

Las redes recurrentes requieren más conexiones y memoria que la redes normales de propagación hacia atrás, debido a que se requiere guardar el resultado de la actividad de la red de cada neurona recurrente en cada paso. De hecho, si una secuencia de entrada consiste de N pasos se deben mantener N copias del nivel de la actividad de cada neurona. Las redes recurrentes actúan como la cinta de una película, la cual mantiene el registro de cada escena, es decir, la acción a través de la secuencia y las neuronas recurrentes proporcionan un contexto natural para cada escena.

Las redes recurrentes operan de la siguiente manera. Al iniciar, t=1, la red recibe el patrón de entrada, mientras que las capas media y de salida tienen un nivel de

actividad estándar, generalmente constante. El patrón de entrada se procesa en la red de manera semejante al perceptrón multicapa y se genera una salida, la cual se almacena para cálculos futuros del error y la actualización de los pesos.

En el segundo tiempo, t=2, se presenta el segundo patrón de la secuencia de entrada y se propaga a través de la red. Sin embargo, en este tiempo, la capa oculta tiene una entrada adicional, esta entrada es el resultado de la actividad de la red en el tiempo t=1. Esta actividad, combinada con el estímulo de entrada, genera la salida de la red para este tiempo y el resultado se almacena para calcular el error y modificar los pesos. Este proceso se repite para toda la secuencia de entrada y con cada patrón de entrada se genera un nuevo patrón de salida que se almacena con la actividad de la red.

Cuando se ha presentado toda la secuencia de entrada, se calcula cualquiera de los errores y se genera el cambio de pesos. En ese instante, también se calcula el cambio de los pesos para cada patrón de entrada. El peso final cambia para esta secuencia y es la suma de estos N pesos combinados. Los pesos no se cambian en este instante, sino hasta que todos los patrones de la secuencia en el conjunto de entrenamiento hayan sido procesados similarmente y se haya calculado la suma acumulada del cambio de los pesos para cada patrón de la secuencia que actualmente se está usando para modificar los pesos. Este proceso se llama *procedimiento de entrenamiento en lote* porque se procesan todos los patrones antes de cambiar los pesos y se agregan como un grupo.

En la figura 3.1 se muestra una red recurrente con tres neuronas en la capa de entrada, dos neuronas en la capa media u oculta, una neurona de salida y una neurona marcada con R, que representa la entrada que realimenta a la red. La actualización de los pesos se hace usando el algoritmo de retropropagación, *back-propagation*. El conjunto de entrenamiento para cada ciclo del algoritmo está dado por $\{x(n), d(n)\}_{n=1}^{N}$ donde N es el número total de elementos en el conjunto de entrenamiento. A continuación se presenta el algoritmo de la red recurrente.

Suponemos que aplicamos un vector de entrada $\mathbf{x}$ de dimensión tres, que consiste de los componetes (x_1, x_2, x_3) como se muestra en la figura 3.1. El estado inicial de las neuronas de la capa media y de salida está predeterminado por algun valor constante, por ejemplo 0.3.

1. *Inicialización*. Se supone que no hay información con mayor prioridad, entonces se establecen los pesos y los umbrales con una distribución uniforme.

2. *Presentar el conjunto de entrenamiento*. Se presenta a la red el conjunto de

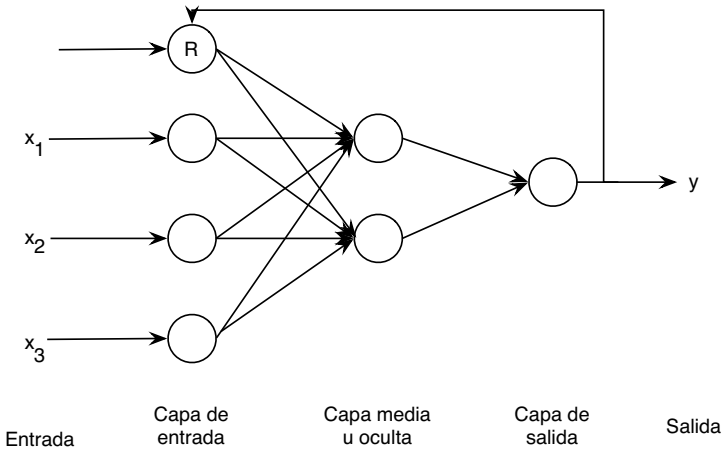


Figura 3.1: Red recurrente con tres neuronas en la capa de entrada, dos neuronas en la capa media y una neurona de salida. La neurona etiquetada con R representa la entrada que realimenta a la red.

entrenamiento y para cada elemento del conjunto se realiza la secuencia hacia adelante y hacia atrás descrita por los pasos 3 y 4 respectivamente.

3. *Cálculo hacia adelante*. Dado un elemento de entrenamiento denotado por ($\mathbf{x}$(n),$\mathbf{d}$(n)), donde el vector $\mathbf{x}$(n) es la entrada que se aplica a los nodos sensoriales de la capa de entrada y el vector $\mathbf{d}$(n) es la salida deseada que se presenta en la capa de salida. Se calcula el campo local inducido y la función de activación para procesar los datos hacia delante capa por capa.

   El campo local inducido es $v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n)$ donde m es el número de entradas aplicadas a la neurona j. Se aplica la función de activación y se obtiene la salida de la neurona j dada por $y_j(n) = \varphi_j(v_j(n))$.

   Para la neurona j en la capa l, el campo local inducido $v_j^{(l)}(n)$ se calcula de la siguiente manera:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)} y_i^{(l-1)}(n) \tag{3.1}$$

   donde $y_i^{(l-1)}(n)$ es la función de salida de la neurona i en la capa previa $l-1$ a la iteración n y $w_{ji}^{(l)}$ es el peso sináptico de la neurona j en la capa l que alimenta la neurona i en la capa $l-1$. Para $i = 0$, se tiene que $y_0^{(l-1)}(n) = +1$ y

$w_{j0}^{(l)}(n) = b_j^{(l)}(n)$ es el sesgo aplicado a la neurona j en la capa l. Se supone que se usa la función φ_j por ejemplo, sigmoide. La señal de salida de la neurona j en la capa l es:

$$y_j^{(l)} = \varphi_j(v_j(n)) \tag{3.2}$$

si la neurona j está en la capa oculta (*i.e.*, $l = 1$), se tiene

$$y_j^{(0)}(n) = x_j(n) \tag{3.3}$$

donde $x_j(n)$ es el j-ésimo elemento del vector de entrada $\mathbf{x}$(n). Si la neurona j está en la capa de salida (*i.e.*, $l = L$) se establece

$$y_j^{(L)} = o_j(n) \tag{3.4}$$

se calcula el error

$$e_j(n) = d_j(n) - o_j(n) \tag{3.5}$$

donde $d_j(n)$ es el j-esimo elemento del vector de la salida deseada $\mathbf{d}$(n.)

4. *Cálculo hacia atrás.* Calcular el gradiente local δ de la red definido por

$$\delta_j^{(l)}(n) \begin{cases} e_j^{(L)}(n)\varphi_j^{'}(v_j^{(L)}(n)) \text{ para la neurona } j \text{ en la capa de salida } L \\ \\ \varphi_j^{'}(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) \text{ para la neurona j en la capa oculta l} \end{cases} \tag{3.6}$$

donde $\varphi_j^{'}(\cdot)$ denota la derivada con respecto al argumento. Para ajustar los pesos sinápticos de la red en la capa l empleando la regla delta generalizada se tiene:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n) y_i^{(l-1)}(n) \tag{3.7}$$

donde η es el parámetro de la tasa de aprendizaje y α es la contante del momento.

5. *Iteración.* Repetir los pasos 3 hacia adelante y 4 hacia atrás presentando nuevos elementos del conjunto de entrenamiento hasta que la red alcance un criterio de error dado.

   El orden para presentar el conjunto de entrenamiento puede ser aleatorio para cada época. El momento y la tasa de aprendizaje se ajustan decrementándolos en la medida que el número de iteraciones se incrementa.

3.3. Arquitectura de las redes recurrentes

Las redes neuronales recurrentes pueden tomar diferentes formas o arquitecturas. En esta sección se describen cuatro arquitecturas, cada una de ellas incorpora un perceptrón multicapa o parte de él y explora la capacidad del mapeo no lineal del perceptrón multicapa.

3.3.1. Modelo recurrente entrada-salida

El modelo recurrente entrada salida, *Input-Output Recurrent Model*, tiene una sola entrada que se aplica a una memoria con una línea de retraso de q unidades y sigue la naturaleza del perceptrón multicapa. La figura 3.2 muestra la arquitectura general de una red recurrente.

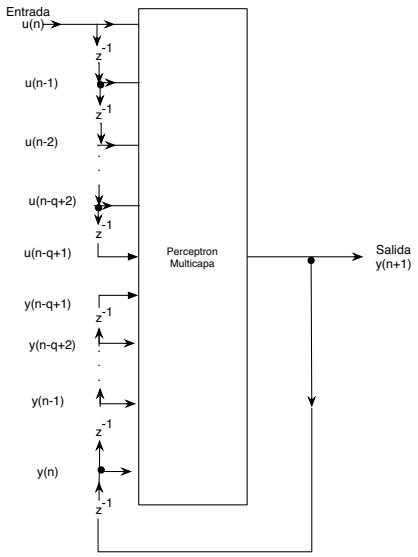


Figura 3.2: Arquitectura general de una red recurrente

Este modelo tiene una salida que realimenta la entrada vía otra memoria con línea de retraso también de q unidades. El contenido de estas dos memorias con líneas de retraso se usa para alimentar la capa de entrada del perceptrón multicapa. Los valores actuales del modelo de entrada se denotan con $u(n)$, y el valor de salida se denotan con $y(+1)$; que son las salidas de la red adelantadas una unidad de

tiempo. El vector aplicado a la capa de entrada del perceptrón multicapa consiste de una ventana de datos formada de la siguiente manera:

- Valores de entrada presentes y pasados, $u(n), u(n-1), \ldots, u(n-q+1)$, que representan la entrada exógena que origina la salida de la red.
- Valores de retraso de la salida, $y(n), y(n-1), \ldots, y(n-q+1)$, los cuales se usan para realimentar a la red.

La red recurrente de la figura 3.2 se conoce como modelo de entradas exógenas con autoregresión no lineal, *nonlinear autoregressive with exogenous input model, NARX*. El comportamiento dinámico del modelo NARX se describe por

$$y(n+1) = F(y(n), \ldots, y(n-q+1), u(n), \ldots, u(n-q+1)) \tag{3.8}$$

donde F es una función de sus argumentos. En la figura 3.2 se asume que las dos memorias con líneas de retraso son de tamaño q y generalmente son diferentes

3.3.2. Modelo Espacio - Estado

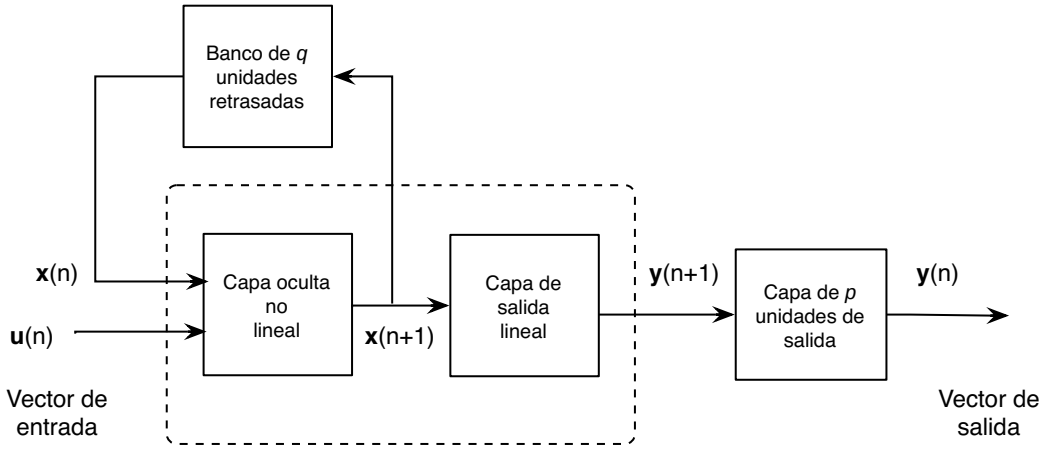


Figura 3.3: Modelo espacio-estado.

La figura 3.3 muestra el diagrama de bloques de la red recurrente, llamada modelo de espacio-estado (*state-space model*). Las neuronas ocultas definen el estado de la red. La salida de la capa oculta realimenta la capa de entrada vía un banco de unidades retrasadas. La capa de entrada consiste de la concatenación de los nodos realimentados y los nodos origen. La red se conecta al ambiente externo vía los nodos origen. El número de unidades de retraso usadas para alimentar la salida de la capa oculta a la capa de entrada determina el orden del modelo, el vector $\mathbf{u}$(n) de $m \times 1$ denota el vector de entrada y el vector $\mathbf{x}$(n) de $q \times 1$ denota la salida de la capa oculta al tiempo n, el funcionamiento dinámico del modelo de la figura 3.3 está dado por las ecuaciones acopladas:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)) \tag{3.9}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) \tag{3.10}$$

donde $\mathbf{f}(\cdot,\cdot)$ es una función no lineal que caracteriza la capa oculta, y $\mathbf{C}$ es la matriz de pesos sinápticos que caracteriza la capa de salida. La capa oculta es no lineal, pero la capa de salida es lineal.

La red recurrente de la figura 3.3 considera algunas arquitecturas como casos especiales. Por ejemplo, la red recurrente simple (SRN *simple recurrent network*) de la figura 3.4 descrita por Elman en 1990. Esta red tiene una arquitectura similar a la figura 3.3 excepto que la capa de salida puede ser no lineal y se omite el banco de unidades de retraso en la salida.

La red de Elman contiene conexiones recurrentes de las neuronas ocultas a la capa de unidades de contexto, la cual consiste de unidades retrasadas que almacenan la salida de las neuronas ocultas por un paso de tiempo, y entonces alimentan el banco de la capa de entrada. Las neuronas ocultas tienen un registro de la primera activación, la cual habilita la red para llevar a cabo el aprendizaje que se extiende durante todos los tiempos. Las neuronas ocultas también alimentan a las neuronas de salida, las cuales reportan la respuesta de la red al estímulo aplicado.

Debido a la naturaleza de la realimentación hacia las neuronas ocultas, estas neuronas pueden continuar reciclando información a través de la red sobre múltiples pasos del tiempo, y además descubrir representaciones abstractas en el tiempo. La red recurrente simple es como una cinta de registro de los datos pasados [Haykin, 1999].

La noción de estado juega un papel muy importante en la formulación matemática de un sistema dinámico. El estado de un sistema dinámico se define como

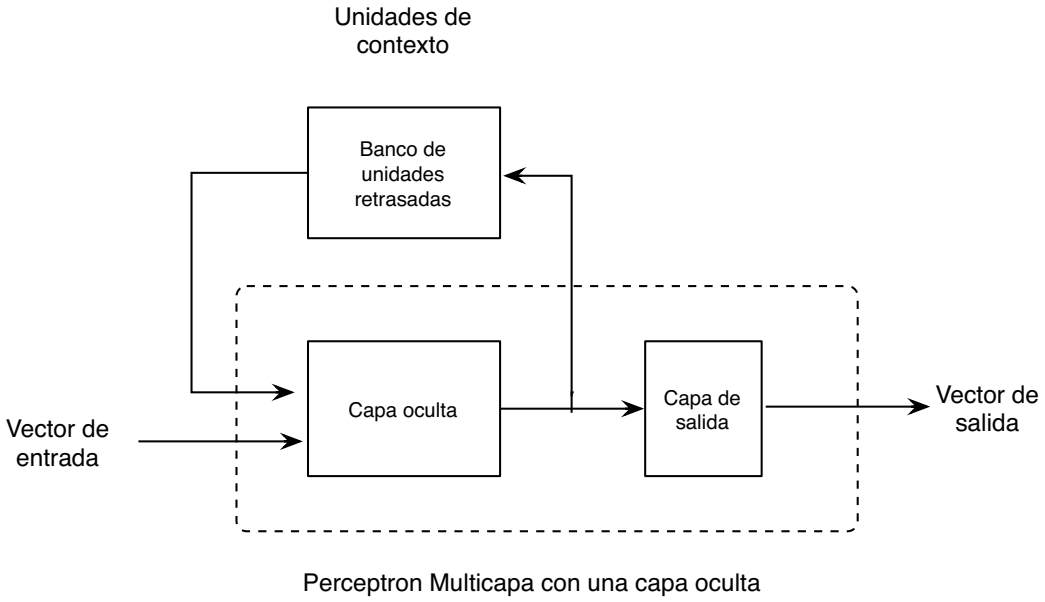


Figura 3.4: Red recurrente simple (SRN).

un conjunto de cantidades que resumen toda la información necesaria sobre el funcionamiento pasado del sistema para describir el desempeño futuro, excepto para efectos externos que surgen de la entrada aplicada [Haykin, 1999]. El vector $\mathbf{x}$(n) de $q \times 1$ denota el estado de un sistema de tiempo discreto no lineal, el vector $\mathbf{u}$(n) de $m \times 1$ denota la entrada aplicada al sistema, y el vector $\mathbf{y}$(n) de $p \times 1$ denota la salida correspondiente del sistema. En términos matemáticos, el funcionamiento dinámico del sistema, suponiendo que está libre de ruido, se describe por el siguiente par de ecuaciones no lineales.

$$\mathbf{x}(n+1) = \varphi(\mathbf{W}_a\mathbf{x}(n) + \mathbf{W}_b\mathbf{u}(n)) \tag{3.11}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) \tag{3.12}$$

donde $\mathbf{W}_a$ es una matriz de $q \times q$, $\mathbf{W}_b$ es una matriz de $q \times (m-1)$, y $\mathbf{C}$ es una matriz de $p \times q$ y $\varphi : \mathbb{R}^q \to \mathbb{R}^q$ es un mapeo diagonal descrito por

$$\varphi : \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{bmatrix} \rightarrow \begin{bmatrix} \varphi(x_1) \\ \varphi(x_2) \\ \vdots \\ \varphi(x_q) \end{bmatrix} \tag{3.13}$$

con memoria y $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ los espacios $\mathbb{R}^m$, $\mathbb{R}^q$ y $\mathbb{R}^p$ son llamados *espacio de entrada*, *espacio de estado*, y *espacio de salida*, respectivamente. La dimensión del espacio de estado q, es el orden del sistema. Así, el modelo espacio-estado de la figura 3.3 es un modelo recurrente de m-entradas, p-salidas de orden q. La ecuación (3.11) es la ecuación de proceso del modelo y la ecuación (3.12) es la ecuación de salida. La ecuación de proceso es una forma especial de la ecuación (3.9).

La red recurrente de la figura 3.3 basada en el uso del perceptrón multicapa estático y dos memorias con línea de retraso, proporciona un método para implementar un sistema no lineal con realimentación lineal descrito por la ecuaciones (3.11), (3.12) y (3.13). Observe que en la figura 3.3 sólo las neuronas en el perceptrón multicapa que realimentan su salida a la capa de entrada vía retrasos son responsables de definir el estado de la red recurrente. Esta afirmación excluye a la neuronas en la capa de salida para la definición de estado.

Para la interpretación de las matrices $\mathbf{W}_a$, $\mathbf{W}_b$ y $\mathbf{C}$, y la función no lineal $\varphi(\cdot)$ se tiene que:

- La matriz $\mathbf{W}_a$ representa los pesos sinápticos de las q neuronas en la capa oculta que están conectadas a los nodos realimentados en la capa de entrada. La matriz $\mathbf{W}_b$ representa los pesos sinápticos de las neuronas ocultas que están conectadas a los nodos origen en la capa de entrada. Esto supone que el término de sesgo de las neuronas ocultas se absorbe en la matriz de pesos $\mathbf{W}_b$.

- La matriz $\mathbf{C}$ representa los pesos sinápticos de las p neuronas lineales en la capa de salida que se conectan a las neuronas ocultas. Esto supone que el término de sesgo de las neuronas de salida se absorbe en la matriz de pesos $\mathbf{C}$.

- La función no lineal $\varphi(\cdot)$ representa la función de activación de una neurona oculta. La función de activación típicamente toma la forma de una función tangente hiperbólica:

$$\varphi(x) = tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.14}$$

o la función logística :

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{3.15}$$

Una propiedad importante de las redes recurrentes descrita por el modelo espacio-estado de las ecuaciones (3.11) y (3.12) es que puede aproximarse a una amplia clase de de sistemas dinámicos no lineales. Sin embargo, las aproximaciones son solo válidas en subconjuntos compactos del espacio estado y para intervalos de tiempo finito [Haykin, 1999].

3.3.3. Perceptrón multicapa recurrente

La arquitectura del perceptrón multicapa recurrente, *recurrent multilayer perceptron* (RMLP) puede tener una o más capas ocultas, como el perceptrón multicapa estático. Cada cálculo en las capas del RMLP tiene realimentación alrededor de ellas, como se puede ver en la figura 3.5 para el caso de un RMLP con dos capas ocultas.

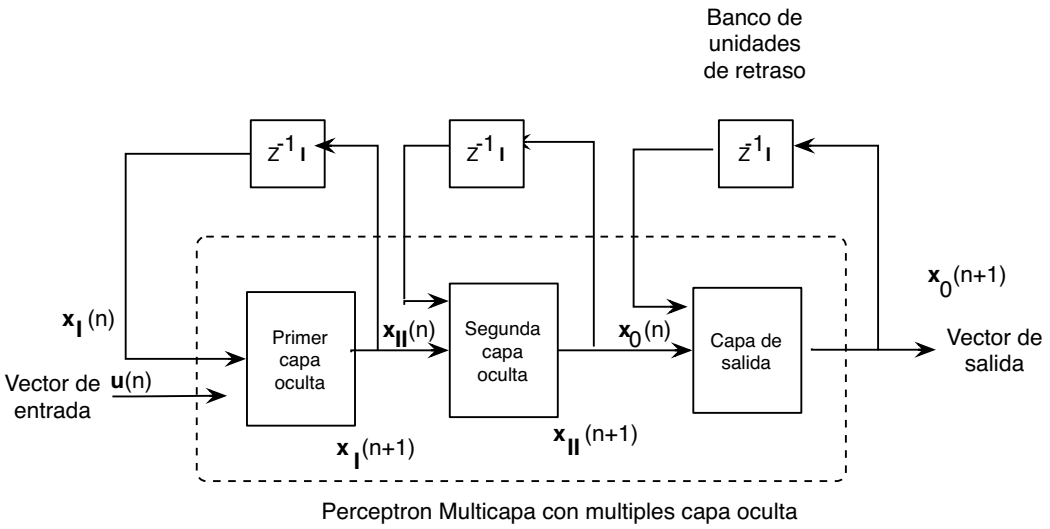


Figura 3.5: Perceptrón multicapa recurrente.

El vector $\mathbf{x}_I(n)$ denota la salida de la primera capa oculta, $\mathbf{x}_{II}(n)$ denota la salida de la segunda capa oculta y así sucesivamente, y el vector $\mathbf{x}_0(n)$ denota la salida de

la capa de salida. En general, el comportamiento dinámico del RMLP responde al vector de entrada $\mathbf{u}(n)$ y se describe por el siguiente sistema de ecuaciones:

$$\begin{aligned} \mathbf{x}_I(n+1) &= \varphi_I(\mathbf{x}_I(n), \mathbf{u}(n)) \\ \mathbf{x}_{II}(n+1) &= \varphi_{II}(\mathbf{x}_{II}(n), \mathbf{x}_I(n+1)) \\ &\vdots \\ \mathbf{x}_0(n+1) &= \varphi_0(\mathbf{x}_0(n), \mathbf{x}_K(n+1)) \end{aligned}$$

donde $\varphi_I(\cdot,\cdot), \varphi_{II}(\cdot,\cdot), \ldots, \varphi_0(\cdot,\cdot)$ denotan las funciones de activación que caracteriza la primera capa oculta, la segunda capa, ..., y la capa de salida del RMLP, respectivamente y K denota el número de capas ocultas en la red.

El RMLP incluye la red de Elman de la figura 3.4 y el modelo espacio-estado de la figura 3.3 desde la capa de salida del RMLP o cualquiera de sus capas ocultas que no tenga restricciones de la forma de la función de activación.

3.3.4. Red de segundo orden

En la descripción del modelo espacio-estado de la figura 3.3 se usa el término de orden para referirse al número de neuronas ocultas cuya salida realimenta la capa de entrada vía un banco de unidades retrasadas.

En otro contexto el término orden se emplea para referirse a la forma en la cual se define el campo local inducido de una neurona. Por ejemplo, en un perceptrón multicapa el campo local inducido, v_k, de una neurona está definido por

$$v_k = \sum_j w_{a,kj} x_j + \sum_j w_{b,ki} u_i \tag{3.16}$$

donde x_j es la señal realimentada derivada de la neurona oculta j y u_i es la señal de entrada aplicada a la neurona i en la capa de entrada; las w's representan los pesos sinápticos de la red. La ecuación (3.16) describe una *neurona de primer orden*. Si el campo local inducido v_k se combina usando multiplicaciones como se muestra a continuación

$$v_k = \sum_i \sum_j w_{kij} x_i u_j \tag{3.17}$$

entonces se dice que es una *neurona de segundo orden*. La neurona de segundo orden k tiene sólo un peso, w_{kji}, que se conecta con la entrada del nodo i y j.

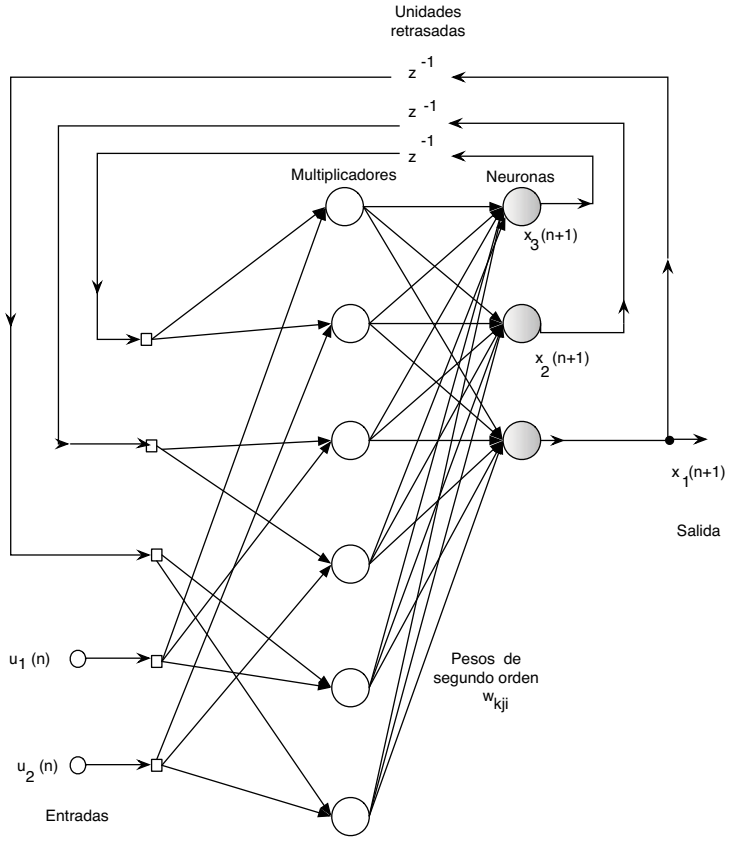


Figura 3.6: Red recurrente de segundo orden, con conexiones de sesgo a las neuronas, las cuales se omiten para simplificar la representación. La red tiene 2 neuronas de entrada y 3 neuronas de estado, de manera que se necesitan $3\times2 = 6$ multiplicadores.

Las neuronas de segundo orden constituyen la base de las redes recurrentes de segundo orden, en la figura 3.6 se puede ver un ejemplo de ellas. La red recibe como entrada una secuencia de tiempo ordenada y la evolución de su comportamiento se define por las siguientes ecuaciones:

$$v_k(n) = b_k + \sum_i \sum_j w_{kij} x_i(n) u_j(n) \tag{3.18}$$

y

$$\begin{aligned} x_k(n+1) &= \varphi(v_k(n)) \\ &= \frac{1}{1+exp(-v_k(n))} \end{aligned}$$

donde $v_k(n)$ es el campo local inducido de la neurona k, b_k es el sesgo asociado, $x_k(n)$ es el estado (salida) de la neurona k, $u_j(n)$ es la entrada aplicada a la neurona j, y w_{kij} es el peso de la neurona k de segundo orden.

Una característica única de las redes recurrentes de segundo orden en la figura 3.6 es que el producto de $x_j(n)u_j(n)$ representa el par {estado, entrada} y los pesos positivos w_{kij} representan el estado de transición, {estado, entrada} $\rightarrow$ {siguiente estado}, mientras que los pesos negativos representan la ausencia de transición. El estado de transición se describe como

$$\delta(x_i, u_j) = x_k \tag{3.19}$$

Las redes neuronales recurrentes de segundo orden se usan para representar el aprendizaje de autómatas de estado finito deterministas *deterministic finite-state automata (DFA)*; un DFA es un dispositivo de procesamiento de información con un número de estados finito.

En esta sección se describió la arquitectura de las redes recurrentes haciendo énfasis en el uso de realimentación global. Sin embargo, también es posible tener arquitecturas de redes recurrentes sólo con realimentación local.

3.4. Algoritmos de entrenamiento

Las redes neuronales llevan a cabo un proceso mediante el cual modifican y actualizan los pesos entre las conexiones de las neuronas, de acuerdo a una regla establecida. A este proceso se le conoce como fase de entrenamiento. El entrenamiento es visto como una búsqueda en un espacio parametrizado multidimensional llamado de pesos, para dar una solución y optimizar gradualmente la regla de aprendizaje [Hassoon, 1995].

Después de un cierto periodo de tiempo, la red ya no modifica los pesos entre sus conexiones, es decir, la fase de entrenamiento ha concluido y ahora la red es capaz de proporcionar una repuesta ante un dato de entrada nuevo. A esta fase se le llama fase de prueba.

De la misma manera que existen dos modo de entrenamiento para el perceptrón multicapa estático: el modo de lote (*batch*) y el modo secuencial[1]. En las redes recurrentes se tiene dos modos de entrenamiento: entrenamiento por época (*Epochwise training*) y entrenamiento continuo (*Continuous training*) [Haykin, 1999].

- Entrenamiento por época. En este contexto una época corresponde a un elemento del conjunto de entrenamiento. Para una época, la red recurrente empieza ejecutándose en un estado inicial hasta alcanzar un nuevo estado en el cual el entrenamiento se detiene; la red borra sus valores y establece un nuevo estado inicial para la siguiente época. El estado inicial no tiene que ser el mismo para cada época de entrenamiento, sin embargo, es importante que el estado inicial para una nueva época sea diferente del estado alcanzado en la época anterior. Este modo de entrenamiento se emplea para simular la operación de una máquina de estado finito en donde se tienen diferentes estados inciales y finales.

- Entrenamiento continuo. Este modo de entrenamiento se aplica a situaciones en donde es importante conservar el resultado de la red. En este entrenamiento la red aprende mientras procesa la señal de entrada y el proceso de aprendizaje no se detiene. Este entrenamiento se emplea por ejemplo en el procesamiento de lenguaje.

Para una topología definida mediante la cual se identificará una función $f(\cdot)$ determinada, es necesario establecer un método mediante el cual sintonizar los párametros de la red $\hat{\mathrm{f}}(\cdot)$. La elección del método de entrenamiento depende del uso que se le dará a la red, a) como una red recurrente o b) como una red estática. Esto último es importante, porque los métodos desarrollados para sistemas recurrentes son concebidos tomando en cuenta esta propiedad, lo que implica que su entrenamiento sea más complejo que en el caso de redes estáticas.

3.4.1. Algoritmos para sistemas estáticos

Un sistema estático se representa mediante una función sin memoria

$$y = f(x, \overline{\theta}) \tag{3.20}$$

[1]El modo de lote calcula la sensibilidad de la red para todo el conjunto de entrenamiento antes de ajustar los parámetros libres de la red. Por otro lado, en el modo secuencial, los parámetros se ajustan después de procesar cada elemento del conjunto de entrenamiento

donde $x \in \mathbb{R}^n$ son las entradas de la función y $\theta \in \mathbb{R}^{n\theta}$ los párametros. El objetivo consiste en minimizar una *función de costo* $J(\cdot)$ que depende del error entre la función real y la estimada

$$E = f(\cdot) - \hat{\mathrm{f}}(\cdot) \tag{3.21}$$

con base en el cálculo de los parámetros θ tal que optimicen la función o en dado caso que sean subóptimos. De la definición de la función de costo depende del método de optimización que se emplee, así como del conocimiento que se tiene para empezar la búsqueda.

3.4.2. Métodos basados en el gradiente

Los métodos basados en el gradiente tienen como objetivo ajustar el valor de los párametros θ en cada paso del entrenamiento. Para ello se basan en la información de la topología local de la función de costo, en función de los párametros para moverse hacia el mínimo local con información del gradiente.

Dentro de las técnicas de optimización local para funciones no lineales, los métodos basados en el gradiente son los más comunes [Nelles, 2001]. Esto se debe a que son relativamente fáciles de usar y a su velocidad de convergencia, pero tienen la desventaja que los resultados sólo se pueden garantizar de forma local. Es decir, generalmente es difícil determinar si el valor de los párametros $\hat{\bar{\theta}}$ origina un mínimo local o global de la función de costo $J(\cdot)$, la cual puede ser tomada como el error cuadrático instantáneo.

$$J(k) = \frac{1}{2}(\hat{\mathrm{f}}(k) - f(k))^2 \tag{3.22}$$

En general, el gradiente se entiende como la derivada parcial

$$\mathbf{g} = \frac{\partial \mathbf{J}(\hat{\bar{\theta}})}{\partial \hat{\bar{\theta}}} \tag{3.23}$$

donde el objetivo de los métodos basados en el gradiente consiste en determinar la forma de variar los parámetros $\hat{\bar{\theta}}$ en cada paso del tiempo k, de forma proporcional a un paso η_θ en la dirección p dada por el gradiente g rotado y escalado por una matriz de dirección R, es decir, calcular para el tiempo $k+1$ el valor de $\hat{\bar{\theta}}$ mediante

$$\hat{\bar{\theta}}(k+1) = \hat{\bar{\theta}}(k) - \eta_\theta \mathbf{p}(k) \tag{3.24}$$

$$\mathbf{p}(k) = R(k)\mathbf{g} = \mathbf{R}(k)\frac{\partial \mathbf{J}(\hat{\bar{\theta}}(k))}{\partial \hat{\bar{\theta}}} \tag{3.25}$$

con esto se busca reducir el valor de la función de costo en cada paso de entrenamiento. En este tipo de métodos, se supone de manera general que la función de costo es convexa con respecto a los párametros. Sin embargo, cuando el sistema es no lineal generalmente esto no se puede garantizar y puede existir un gran número de mínimos locales. Si el algoritmo de búsqueda inicia cerca de uno de estos mínimos, suele ser difícil que encuentre un mínimo con mejor desempeño, en caso de existir, y se dice que queda atrapado en un mínimo local.

La diferencia entre los métodos basados en el gradiente está dada por la elección de la matriz $R(k)$ y el paso de entrenamiento η_θ. Este tipo de métodos generalmente son sensibles al ruido, así como a la magnitud de la ganancia de adaptación η_θ. A continuación se describen algunos de estos métodos.

1. Descenso en la mayor inclinación (*steepest descent).* Este método se establece con la forma más sencilla de la ecuación (3.24) porque la matriz de dirección, $R(k)$, es la matriz identidad $R(k) = I$ y η_θ es una constante.

   Las ventajas del descenso en la mayor inclinación se reflejan en su desempeño en las primeras iteraciones, además es relativamente fácil de implementar y comprender. No obstante, tiene la desventaja de que converge lentamente después de transcurrir varias iteraciones; lo que implica que no se encuentre el óptimo local, a menos que se considere un número infinito de pasos de entrenamiento. Generalmente no se busca el valor del óptimo, sino un valor subóptimo que satisfaga un determinado criterio de aproximación, es decir, una cota del error.

2. Método Newton. En este método, la matriz de dirección, $R(k)$, se considera como la inversa del Hessiano

$$R(k) = H^{-1}(k) = (\frac{\partial^2 \mathbf{g}}{\partial \theta^2})^{-1} \tag{3.26}$$

   La ventaja del método de Newton consiste en tener una buena velocidad de convergencia, especialmente al iniciar cerca del óptimo. Sin embargo sus desventajas son notorias porque requiere calcular la matriz Hessiana, lo cual puede ser difícil, ya que requiere que esta matriz sea positiva definida para converger y en general, computacionalmente no es conveniente porque es necesario calcular la inversa de una matriz. Para sortear estos obstáculos se usan formas aproximadas del Hessiano tales como:

$$\hat{\mathrm{H}}(k) = \hat{\mathrm{H}}(k+1) + Q(k+1) \tag{3.27}$$

o mediante métodos basados en la fórmula de Broyden-Fletcher-Goldfarb-Shanno, tomando como valor inicial el Hessiano $H(0) = P$,

$$\hat{\mathrm{H}}(k) = (P - \Gamma(k+1))H^{-1}(k+1)(P - \Gamma(k+1))^T + \Theta(k+1) \quad (3.28)$$

$$\Gamma(k) = \frac{\Delta\theta(k)\Delta\mathbf{g}^T(k)}{\Delta\theta^T(k)\Delta\mathbf{g}(k)} \quad (3.29)$$

$$\Theta(k) = \frac{\Delta\theta(k)\Delta\theta^T(k)}{\Delta\theta^T(k)\Delta\mathbf{g}(k)} \quad (3.30)$$

$$\Delta\theta(k) = \theta(k+1) - \theta(k) \quad (3.31)$$

$$\Delta\mathbf{g}(k) = \mathbf{g}(k+1) - \mathbf{g}(k) \quad (3.32)$$

3. Métodos conjugados. Este tipo de métodos emplean aproximaciones de los métodos anteriores. A pesar de que los resultados no son tan precisos, computacionalmente son más convenientes, ya que la complejidad de los cálculos disminuye, lo que permite resolver problemas con una mayor cantidad de parámetros.

   Los métodos conjugados se pueden describir como una actualización de parámetros dada por

$$\theta(k) = \theta(k-1) - \eta_\theta \mathbf{p}(k-1) \quad (3.33)$$

   donde

$$\mathbf{p}(k) = \mathbf{g}(k) - \beta(k)\mathbf{p}(k-1) \quad (3.34)$$

   y el término β está dado por

$$\beta(k) = \frac{\mathbf{g}^T(k)\mathbf{g}(k)}{\mathbf{g}^T(k-1)\mathbf{g}(k-1)} \quad (3.35)$$

   Entre las ventajas de estas aproximaciones se tiene que no se requiere calcular el Hessiano, esto permite que se aplique a problemas con un gran número de parámetros, además tiene una buena velocidad de convergencia y los requerimientos computacionales son bajos comparados con los métodos de Newton.

3.4.3. Métodos basados en mínimos cuadrados

Los métodos basados en mínimos cuadrados sobresalen por no hacer mayores suposiciones sobre la topología de la función de costo, salvo suponer que es una función suave. Comúnmente, la función de costo más usada es la suma de los errores al cuadrado ponderados

$$J(\theta) = \sum_{i_1}^{N} q_i (f_i - \hat{\mathrm{f}}_i)^2 \tag{3.36}$$

donde q_i es la ponderación de cada error. Típicamente $q_i = 1$. También se puede escribir como

$$J = \Phi^T \Phi \tag{3.37}$$

donde $\Phi = [\Phi(1,\theta)\Phi(2,\theta)\ldots\Phi(N,\theta)]^T$ es la matriz que contiene los errores de los datos generados por el modelo dados por $\hat{\mathrm{f}}$ con respecto a los datos obtenidos f_i. En general para aplicar este método se supone que el modelo $\hat{\mathrm{f}}$ es lineal con respecto a sus párametros, es decir

$$\hat{\mathrm{y}} = \hat{\mathrm{f}}(\mathbf{x}, \overline{\theta}) = \phi(\mathbf{x})\overline{\theta} \tag{3.38}$$

Con esta suposición, se puede determinar que el valor de los parámetros que minimizan la función de costo 3.36 es

$$\hat{\overline{\theta}} = (\phi^T \phi)^{-1} \phi^T f \tag{3.39}$$

El cálculo de la matriz $(\phi^T\phi)^{-1}$ se puede llevar a singularidades, entonces suele emplearse el método recursivo para mínimos cuadrados, el cual considera incluso su uso en sistemas en tiempo real mediante la expresión de adaptación de párametros

$$\hat{\overline{\theta}}(k) = \hat{\overline{\theta}}(k-1) + \overline{\gamma}(k)E(k) \tag{3.40}$$

donde

$$E(k) = y(k) - \hat{\mathrm{y}} = y(k) - \phi^T(k)\hat{\overline{\theta}}(k-1) \tag{3.41}$$

$$\gamma(k) = \frac{P(k-1)\phi(k)}{\phi^T(k)P(k-1)\theta(k) + 1} \tag{3.42}$$

$$P(k) = (I - \gamma(k)\phi^T(k))P(k-1) \tag{3.43}$$

Por lo general, se inicia el valor de P con $P(0) = \alpha I$ donde $\alpha >> 1$.

3.4.4. Algoritmos para sistemas recurrentes

Los algoritmos para sistemas recurrentes consideran los dos modos de entrenamiento: entrenamiento por época y entrenamiento continuo. En un sistema recurrente la salida final depende de los valores anteriores tanto de la salida como de la entrada. A continuación se describen los siguientes algoritmos para redes recurrentes:

- El algoritmo de retropropagación a través del tiempo, *back-propagation-through-time algorithm (BPTT)*, está basado en el perceptrón multicapa. Se puede implementar en el modo de entrenamiento por época o por modo continuo (tiempo-real) o combinado.
- El algoritmo de aprendizaje recurrente de tiempo real, se deriva del modelo espacio-estado descrito por la ecuación (3.11) y (3.12). El aprendizaje en tiempo real (continuo) se basa en el gradiente descendiente que utiliza una cantidad mínima de la información disponible, es decir, la estimación instantánea del gradiente de la función de costo con respecto al vector de parámetros que tiene que ser ajustado.

Estos algoritmos tienen muchas características en común. Primero, ambos se basan en el método del gradiente descendiente por medio del cual el valor de la función de costo, basado en el criterio de error cuadrado, se minimiza con respecto a los pesos sinápticos de la red. Segundo, ambos son relativamente simples de implementar, sin embargo, pueden convergen lentamente. Tercero, ambos están relacionados por la representación de una gráfica de flujo del algoritmo de retropropagación a través del tiempo [Haykin, 1999].

A continuación se presentan algunas heurísticas que se deben considerar antes de describir los algoritmos mencionados para mejorar el entrenamiento de las redes recurrentes que involucran el uso del método del gradiente descendiente.

- El orden lexicográfico de los datos de entrenamiento se pueden usar, con las cadenas menores de símbolos que se presentan a la red.
- El entrenamiento puede empezar con datos de entrenamiento pequeños, y se puede ir incrementando como avanza el proceso de entrenamiento.
- Los pesos sinápticos de la red se pueden actualizar solo si el error absoluto de los datos de entrenamiento que empiezan a procesarse por la red es más grande que algún criterio preescrito

- El uso de pesos que decae durante el entrenamiento es recomendado.

La primera heurística es de particular interés. Si es implementable, proporciona un procedimiento para aliviar el desvanecimiento del problema del gradiente que se incrementa en el entrenamiento de la red por medio del método del gradiente descendiente.

3.4.5. Algoritmo de retropropagación a través del tiempo

El algoritmo de retropropagación a través del tiempo, *backpropagation-through-time*, es una extensión del algoritmo de retropropagación estándar. Este algoritmo se puede desplegar de una operación temporal sobre una red alimentada hacia adelante y una topología que crece en una capa en cualquier tiempo.

Sea $\mathcal{N}$ una red recurrente que requiere aprender una tarea temporal, empezando en un tiempo n_0 hasta el tiempo n. Dada $\mathcal{N}^*$ que denota la red con alimentación hacia adelante (*feedforward*) que resulta de la operación temporal de la red recurrente $\mathcal{N}$. La red $\mathcal{N}^*$ resultante se relaciona con la red original $\mathcal{N}$ como sigue:

- En cada paso del intervalo $(n_0, n]$, la red $\mathcal{N}^*$ tiene una capa que contiene K neuronas, donde K es el número de neuronas que contiene la red $\mathcal{N}$.
- En cualquier capa de la red $\mathcal{N}^*$ hay copias de neuronas en la red $\mathcal{N}$.
- Para cada paso $l \in [n_0, n]$, las conexiones sinápticas de la neurona i en la capa l a la neurona j en la capa $l+1$ de la red $\mathcal{N}^*$ son una copia de los conexiones sinápticas para la neurona i a la neurona j en la red $\mathcal{N}$.

La aplicación de este procedimiento deja básicamente dos implementaciones diferentes del algoritmo de retroalimentación a través del tiempo, las cuales dependen del algoritmo de entrenamiento que se utilice ya sea continuo o por épocas.

3.4.6. Algoritmo de propagación hacia atrás truncado

Para usar el algoritmo de propagación hacia atrás en modo de tiempo real, se usa un valor instantáneo de la suma de los errores cuadrados, normalmente,

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in \mathcal{A}} e_j^2(n)$$

como función de costo para ser minimizada. Como en el modo secuencial del algoritmo de aprendizaje de retropropagación estándar, se usa el gradiente negativo de la función de costo $\mathcal{E}(n)$ para calcular el ajuste apropiado de los pesos sinápticos de la red en cada instante de tiempo n. Los ajustes son hechos sobre la base continua, mientras se ejecuta la red. Sin embargo, el orden se hace en una manera viable computacionalmente, sólo se guardan la historia relevante de los datos de entrada y el estado de la red para un número fijo de pasos de tiempo, llamada profundidad truncada *truncation depth*, que se denota por h. Sólo la información mayor al tiempo h se considera irrelevante y se ignora. Si no se trunca el cálculo y se permite regresar al tiempo de inicio, el tiempo de cálculo y los requisitos almacenados pueden crecer linealmente con el tiempo mientras la red se ejecuta, eventualmente alcanza un punto donde el proceso de aprendizaje completo llega a ser impráctico.

La segunda forma del algoritmo se llama algoritmo de retropropagación en el tiempo truncado *truncated back-propagation through-time (BPTT(h))*. El gradiente local para la neurona j se define ahora por

$$\delta_j(l) = \frac{\partial \mathcal{E}(l)}{\partial v_j(l)} \text{ para toda } j \in A \text{ y } n - h < l \leq n \tag{3.44}$$

el cual deja la fórmula:

$$\delta_j(l) \begin{cases} \varphi'(v_j(l))e_j(l) \text{ para } l = n \\ \\ \varphi'(v_j(l)) \sum_{k \in \mathcal{A}} w_{kj}(l)\delta_k(l+1) \text{ para } n - h < l < n \end{cases} \tag{3.45}$$

Una vez que se calcula la propagación hacia atrás al tiempo $n - h + 1$, se ajustan los pesos sinápticos w_{ji} de la neurona j:

$$\Delta w_{ji}(n) = \eta \sum_{l=n-h+1}^{n} \delta_j(l)x_i(l-1) \tag{3.46}$$

donde η y $x_i(l-1)$ son como se definió previamente. Se observa que el uso de $w_{kj}(l)$ en la ecuación (3.45) requiere que la historia de los valores de los pesos se mantenga. El uso de w_{kj} en la ecuación se puede justificar sólo si el parámetro de la tasa de aprendizaje η es suficientemente pequeño para asegurar que el valor de los pesos no cambia significativamente de un paso de tiempo al siguiente.

Comparando la ecuación 3.45 con la ecuación **??**, se puede ver que, a diferencia del algoritmo de épocas BPTT, la señal de error sólo se aplica en el cálculo del tiempo actual n. Esto explica la razón para no mantener el registro de los valores

pasados de la respuesta deseada. El algoritmo de retropropagación truncado a través del tiempo realiza el cálculo de los primeros pasos similarmente al algoritmo de retropropagación estocástico para el cálculo de las neuronas ocultas en el perceptrón multicapa.

3.4.7. Aprendizaje Recurrente en Tiempo Real

En esta sección se describe otro algoritmo de aprendizaje conocido como aprendizaje recurrente en tiempo real *real-time recurrent learning (RTRL)*. En este algoritmo se ajustan los pesos sinápticos de la red recurrente, conectada completamente, en tiempo real; es decir, el ajuste de los pesos se lleva a cabo mientras la red está procesando la señal recibida. La figura 3.7 muestra esta red recurrente, la cual está formada por q neuronas con m entradas externas y tiene dos capas distintas: una capa de entrada realimentada y concatenada y una otra capa de nodos de procesamiento, correspondiente a las conexiones sinápticas de la red que se procesan hacia adelante y hacia atrás. La descripción espacio-estado de la red se define por las ecuaciones (3.11) y (3.12). A continuación se presenta la ecuación de proceso (3.11) en su forma expandida:

$$\mathbf{x}(n+1) = \begin{bmatrix} \varphi(\mathbf{w}_1^T \xi(n)) \\ \vdots \\ \varphi(\mathbf{w}_j^T \xi(n)) \\ \vdots \\ \varphi(\mathbf{w}_q^T \xi(n)) \end{bmatrix} \tag{3.47}$$

además se supone que todas las neuronas tienen una función de activación común $\varphi(\cdot)$. El vector $\mathbf{w}_j$ de $(q+m+1) \times 1$ es el vector de pesos sinápticos de la neurona j en la red recurrente, definido como:

$$\mathbf{w}_j = \begin{bmatrix} \mathbf{w}_{a,j} \\ \mathbf{w}_{b,j} \end{bmatrix} \quad j = 1,2, \ldots, q \tag{3.48}$$

donde $\mathbf{w}_{a,j}$ y $\mathbf{w}_{b,j}$ son las j-ésimas columnas de las matrices de pesos transpuestas $\mathbf{W}_a^T$ y $\mathbf{W}_b^T$, respectivamente. El vector $\xi(n)$ de $(q+m+1) \times 1$ se define por:

$$\xi(n) = \begin{bmatrix} \mathbf{x}(n) \\ \mathbf{u}(n) \end{bmatrix} \tag{3.49}$$

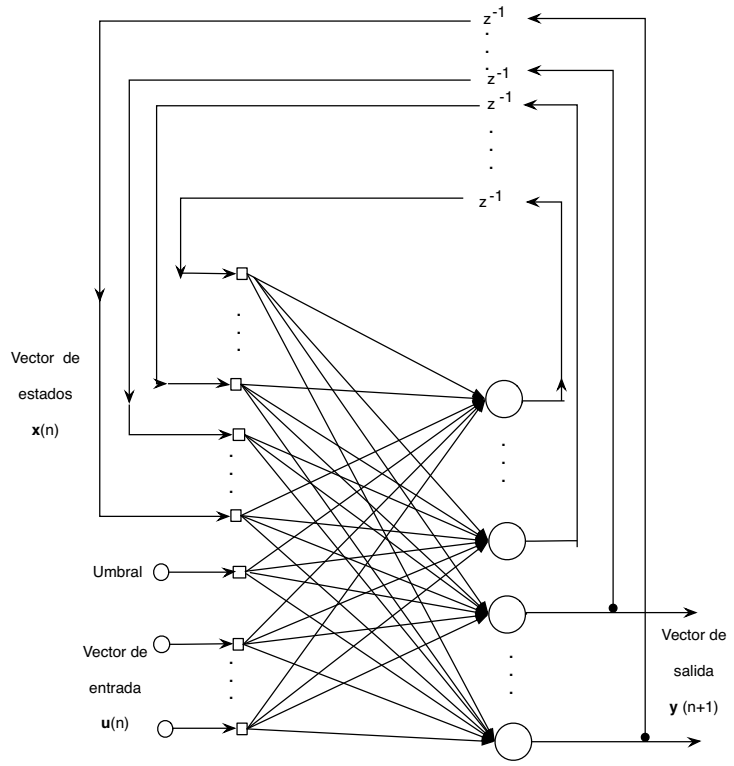


Figura 3.7: Red recurrente conectada completamente para la formulación del algoritmo RTTL.

donde $\mathbf{x}(n)$ es el vector de estado de $q \times 1$ y $\mathbf{u}(n)$ de $(m+1) \times 1$ el vector de entrada. El primer elemento de $\mathbf{u}(n)$ es $+1$ y, en forma correspondiente, el primer elemento de $\mathbf{w}_{b,j}$ es igual a el sesgo b_j aplicado a la neurona j.

Para simplificar la representación se introducen tres nuevas matrices $\mathbf{\Lambda}_j(n), \mathbf{U}_j(n)$ y $\mathbf{\Phi}(n)$, descritas como sigue:

1. $\mathbf{\Lambda}_j(n)$ es una matriz de $q \times (q + m + 1)$ definida como la derivada parcial del vector de estados $\mathbf{x}(n)$ con respecto al vector de pesos $\mathbf{w}_j$:

$$\mathbf{\Lambda}_j(n) = \frac{\partial \mathbf{x}(n)}{\partial \mathbf{w}_j}, \ j = 1,2, \ldots, q \tag{3.50}$$

2. $\mathbf{U}_j(n)$ es una matriz de $q \times (q + m + 1)$ donde todos los renglones son cero,

excepto para el j-ésimo renglón que es igual a la transpuesta del vector $\xi(n)$:

$$\mathbf{U}_j(n) = \begin{bmatrix} 0 \\ \xi^T(n) \\ 0 \end{bmatrix} \leftarrow j\text{ésimo rengón, } j = 1,2, \ldots, q \tag{3.51}$$

3. $\mathbf{\Phi}(n)$ es la matriz diagonal de $q \times q$ en donde el k-ésimo elemento de la diagonal es la derivada parcial de la función de activación con respecto a sus argumentos, evaluada en $\mathbf{w}_j^T \xi(n)$:

$$\mathbf{\Phi}(n) = diag\left(\varphi'(\mathbf{w}_1^T\xi(n)), \ldots, \varphi'(\mathbf{w}_j^T\xi(n)), \ldots, \varphi'(\mathbf{w}_q^T\xi(n))\right) \tag{3.52}$$

Con estas definiciones, se puede diferenciar la ecuación (3.48) con respecto a $\mathbf{w}_j$, usando la regla de la cadena, se obtiene la ecuación recursiva:

$$\mathbf{\Lambda}_j(n) = \mathbf{\Phi}(n)[\mathbf{W}_a(n)\mathbf{\Lambda}_j(n) + \mathbf{U}_j(n)],\ j = 1,2, \ldots, q \tag{3.53}$$

Esta ecuación recursiva describe la dinámica del estado no lineal (*i.e.* la evolución del estado) del proceso de aprendizaje recurrente en tiempo real.

Para completar la descripción del proceso de aprendizaje, se necesita relacionar la matriz $\Lambda_j(n)$ con el gradiente de la superficie de error con respecto a $\mathbf{w}_j$. Entonces se usa la ecuación de medida (3.12) para definir el vector de error de $p \times r$

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n) = \mathbf{d}(n) - \mathbf{C}\mathbf{x}(n) \tag{3.54}$$

Después se define la suma instantánea del error cuadrado al tiempo n en términos de $\mathbf{e}(n)$ por

$$\mathcal{E}(n) = \frac{1}{2}\mathbf{e}^T(n)\mathbf{e}(n) \tag{3.55}$$

Como el objetivo del proceso de aprendizaje consiste en minimizar la función de costo obtenida por la suma de $\mathcal{E}(n)$ sobre el tiempo n, se tiene:

$$\mathcal{E}_{total} = \sum_n \mathcal{E}(n) \tag{3.56}$$

Ahora se puede usar el método del gradiente descendiente por pasos, el cual

requiere conocer la matriz gradiente escrita como:

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{E}_{total} &= \frac{\partial \mathcal{E}_{total}}{\partial \mathbf{W}} \\ &= \sum_{n} \frac{\partial \mathcal{E}(n)}{\partial \mathbf{W}} \\ &= \sum_{n} \nabla_{\mathbf{w}}\mathcal{E}(n)\end{aligned}$$

donde $\nabla_{\mathbf{w}}\mathcal{E}(n)$ es el gradiente de $\mathcal{E}(n)$ con respecto a la matriz de pesos $\mathbf{W} = \{\mathbf{w}_k\}$. Se puede continuar derivando la ecuación actualizando los pesos sinápticos de la red recurrente sin involucrar aproximaciones. Sin embargo, es posible desarrollar un algoritmo de aprendizaje que se use en el entrenamiento de la red recurrente en *tiempo real*, utilizando una estimación instantánea del gradiente, llamado $\nabla_{\mathbf{w}}\mathcal{E}(n)$.

Tomando la ecuación (3.55) como la función de costo para minimizar, se deriva con respecto al vector de pesos $\mathbf{w}_j$ y se obtiene:

$$\frac{\partial \mathcal{E}(n)}{\partial \mathbf{w}_j} = \left(\frac{\partial \mathbf{e}(n)}{\partial \mathbf{w}_j}\right)\mathbf{e}(n) \tag{3.57}$$

$$= -\mathbf{C}\left(\frac{\partial \mathbf{x}(n)}{\partial \mathbf{w}_j}\right)\mathbf{e}(n) \tag{3.58}$$

$$= -\mathbf{C}\mathbf{\Lambda}_j(n)\mathbf{e}(n),\ j = 1,2,\ldots,q \tag{3.59}$$

Además, se determina el ajuste que se aplica al vector de pesos sinápticos $\mathbf{w}_j(n)$ de la neurona j dado por:

$$\begin{aligned}\mathbf{\Delta w}_j(n) &= -\eta\frac{\partial \mathcal{E}(n)}{\partial \mathbf{w}_j} \\ &= \eta\mathbf{C}\mathbf{\Lambda}_j(n)\mathbf{e}(n),\ j = 1,2,\ldots,q\end{aligned}$$

donde η es el parámetro de la tasa de aprendizaje y $\mathbf{\Lambda}$ es gobernada por la ecuación (3.53).

Como sólo permanecen los términos en que se especifican las condiciones iniciales del proceso de aprendizaje, se propone el conjunto

$$\mathbf{\Lambda}_j(0) = \mathbf{0} \text{ Para toda } j \tag{3.60}$$

esto implica que inicialmente la red recurrente está en un estado constante.

Cuadro 3.1: Resumen del algoritmo de aprendizaje recurrente de tiempo real

Parámetros
m = dimensión del espacio de entrada
q = dimensión del espacio de estados
p = dimensión del espacio de salidas
$\mathbf{w}_j$ = vector de pesos sinápticos de la neurona j, $j = 1, 2, \ldots, q$.

inicialización
1. Establecer los pesos sinápticos del algoritmo con valores pequeños seleccionados por una distribución uniforme.
2. Establecer el valor inicial del vector de estados $\mathbf{x}(0) = \mathbf{0}$

Cálculos
Calcular para $n = 0, 1, 2, \ldots,$
$\mathbf{\Lambda}_j(n+1) = \mathbf{\Phi}(n)[\mathbf{W}_a(n)\mathbf{\Lambda}_j(n) + \mathbf{U}_j(n)]$
$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{C}x(n)$
$\nabla\mathbf{w}_j(n) = \eta\mathbf{C}\Lambda_j(n)\mathbf{e}(n)$
Las definiciones de $\mathbf{x}(n), \mathbf{\Lambda}_j(n), \mathbf{U}_j(n)$ y $\mathbf{\Phi}(n)$ están dadas por la ecuación (3.48), (3.50), (3.51) y (3.52), respectivamente.

La tabla 3.1 presenta un resumen del algoritmo de aprendizaje recurrente en tiempo real. Se formula el algoritmo como se describió anteriormente aplicándose a una función arbitraria $\varphi(\cdot)$ con respecto al argumento.

Para el caso especial de una sigmoidal no lineal en la forma de una función tangente hiperbólica, se tiene

$$\begin{aligned} x_j(n+1) &= \varphi(v_j(n)) \\ &= tanh(v_j(n)) \end{aligned}$$

y

$$\begin{aligned} \varphi'(v_j(n)) &= \frac{\partial \varphi(v_j(n))}{\partial v_j(n)} \\ &= sech^2(v_j(n)) \\ &= 1 - x_j^2(n+1) \end{aligned}$$

donde $v_j(n)$ es el campo local inducido de la neurona j y $x_j(n+1)$.

Al aplicar el algoritmo de aprendizaje recurrente de tiempo real usando el gradiente instantáneo $\nabla_{\mathbf{w}}\mathcal{E}(n)$ significa que se describen las derivadas para un tiempo no real basado en el gradiente verdadero $\nabla_{\mathbf{w}}\mathcal{E}_{total}$. Esta derivada es análoga a la obtenida por el algoritmo de retropropagación estándar usado en el entrenamiento ordinario del perceptrón multicapa, donde el cambio de los pesos se hace después de presentar cada patrón de entrenamiento. Mientras que en el algoritmo de aprendizaje recurrente en tiempo real esto no garantiza que siga precisamente el gradiente negativo de la función $\mathcal{E}_{total}(\mathbf{W})$ con respecto a la matriz de tiempo $\mathbf{W}$. Así que las diferencias entre las versiones de tiempo real y no real son pequeñas; estas dos versiones llegan a ser casi idénticas cuando el parámetro de la tasa de aprendizaje η va reduciendo. La ventaja potencial de esta derivada en el desarrollo del gradiente verdadero es en que al observar la trayectoria, obtenida graficando $\mathcal{E}(n)$ contra los elementos de la matriz de pesos $\mathbf{W}(n)$, esta puede depender del cambio de los pesos producidos por el algoritmo, el cual puede ser visto como otra fuente de retroalimentación y por siguiente una causa de instabilidad en el sistema. Se puede evitar este efecto usando una tasa de aprendizaje η suficientemente pequeña para hacer una escala de tiempo del cambio de los pesos mucho más pequeña el tiempo de escala de la operación de la red [Haykin, 1999]

3.4.8. Teacher Forcing

Una estrategia que frecuentemente se usa en el filtrado adaptable para el entrenamiento de redes recurrentes es *Teacher Forcing*. Teacher Forcing también se le conoce como método de la ecuación-error, *equation-error method*. Durante el entrenamiento de la red, Teacher Forcing remplaza la salida actual de una neurona por la respuesta deseada correspondiente (*i.e.* señal de origen) principalmente en el cálculo de subsecuencias, dondequiera que la respuesta deseada esté disponible. Aunque Teacher Forcing se describe bajo el algoritmo de RTRL, este se aplica a cualquier otro algoritmo de aprendizaje. Sin embargo, la neurona en cuestión debe alimentar la salida que regresa a la red.

Los beneficios de incluir Teacher Forcing son:

- *Teacher Forcing puede dar un entrenamiento rápido.* La razón de este mejoramiento se debe a una cantidad que forza el aprendizaje como para suponer que la red aprende correctamente todo desde que empieza la tarea a la que pertenecen las neuronas donde se aplica el algoritmo.

- *Teacher Forcing puede servir como un mecanismo correctivo durante el entrenamiento.* Por ejemplo, si los pesos sinápticos de la red tienen valores correctos y de algún modo la red está operando actualmente en una región errónea del espacio de estado, entonces, ajustar los pesos sinápticos sería una estrategia errónea en esta situación.

Un algoritmo de aprendizaje basado en el gradiente que usa Teacher Forcing en la optimizacion de una función de costo es contraparte no forzada (unforced counterpart). Teacher Forcing y las vesiones no forzadas del algoritmo pueden dar diferentes soluciones, al menos pertinentes a las señal de error cero, en tal caso el aprendizaje no es necesario.

Estos dos últimos algoritmos sufren de los mismos inconvenientes que sus símiles para sistemas estáticos: suponen que la función de costo es convexa y suave con respecto a los parámetros y son relativamente sensibles al ruido y al valor de la ganancia de adaptación. En caso de que no se pueda garantizar la convexidad global, se supone de que el algoritmo comenza la búsqueda en la vecindad de un buen mínimo local, por lo que en el caso de funciones de costo relativamente complejas es necesario contar con un algoritmo de inicialización de parámetros que lo sitúe cerca de algún mínimo, idealmente el *mínimo global*.

En este capítulo se describieron las redes neuronales recurrentes, las cuales además de proporcionar un esquema de procesamiento alternativo basado en la operación de un número determinado de neuronas que se conectan entre sí, procesan la información de entrada formando ciclos, es decir, se realimenta con la información de salida. Esta característica es importante para el modelado del tráfico vehicular. En particular, el uso de las redes neuronales recurrentes como se mostrará en el siguiente capítulo.

Capítulo 4

Modelado de tráfico vehicular mediante redes neuronales recurrentes

4.1. Trabajo relacionado

En esta sección se describe brevemente el trabajo que se ha realizado para modelado del tráfico vehicular en una parte de la carretera I-210 West del sur de California. En los diferentes trabajos para modelar el tráfico en vías rápidas, el tráfico se mide normalmente mediante detectores de paso sencillos o dobles colocados en el pavimento. Estos detectores arrojan datos de la ocupanción o volumen de tráfico junto con datos sobre la longitud efectiva de los vehículos. Estas mediciones se pueden convertir en cantidades macroscópicas como densidad y velocidad.

Una forma de modelar el tráfico consiste en definir celdas que representen fragmentos de longitud conocida de la autopista. Con ello se puede analizar el comportamiento del tráfico, la influencia de rampas de entrada y salida, la densidad y el flujo de las diferentes celdas y observar así cambios discretos en secciones de interés de la carretera [Daganzo, 1994].

Como se describió en la sección 2.2.1 en una autopista existen dos tipos de flujo, al primero de ellos se le llama *flujo libre*, esto quiere decir que los vehículos circulan libremente en la autopista y el segundo es el *flujo congestionado* que se presenta cuando los vehículos no pueden avanzar a una velocidad libre ya que existen otros vehículos que se los impiden.

Para el caso de un segmento lineal de una autopista sin rampas de entrada o salida, la conservación de los vehículos se puede escribir como:

$$\rho_i(k+1) = \rho_i(k)\frac{T_S}{l_i}(q_i(k) - q_{i+1}(k)) \tag{4.1}$$

Donde k es el índice de tiempo, T_S es un intervalo de tiempo discreto, l_i es el tamaño de la i-ésima celda, q_i(k) es la tasa de flujo (medida en vehículos por unidad de tiempo) de la i-ésima celda durante el intervalo $[k, k+1)$. q_i(k) se calcula tomando el mínimo de 2 cantidades:

$$q_i(k) = min(S_{i-1}(k), R_i(k)) \tag{4.2}$$

donde

$$S_{i-1}(k) = min(\upsilon\rho_{i-1}(k), Q_{M,i-1}) \tag{4.3}$$

es el flujo máximo que puede ser previsto por la celda $i-1$ bajo condiciones de flujo libre en el intervalo $[k, k-1)$, y

$$R_i(k) = min(Q_{M,i}, \omega(\rho_J - \rho_i(k))) \tag{4.4}$$

es el flujo máximo que puede ser recibido por la i-ésima celda bajo condiciones de flujo congestionado en el mismo intervalo de tiempo. Por otra parte, la densidad

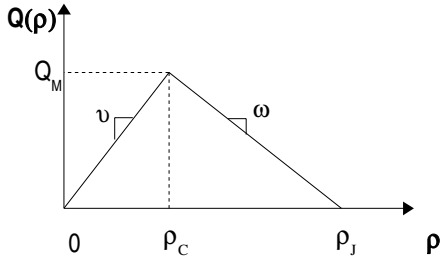


Figura 4.1: Flujo en función de la densidad

se define como el número de vehículos por milla. Así, Q_M es el flujo máximo que ocurre con densidad ρ_C y ρ_J indica la densidad máxima (estancamiento total). La figura 4.1 muestra el comportamiento del flujo en función de la densidad vehicular.

La velocidad de flujo libre v es la velocidad promedio a la cual los vehículos viajan a través de la carretera bajo condiciones de no-congestionamiento; si un bloque de automóviles se mueve de una celda a otra a velocidad v se le puede llamar *efecto de transporte hacia adelante*. Por otro lado, ω es la velocidad promedio en la que las ondas de congestionamiento se propagan en dirección contraria al tráfico a través de la carretera en condiciones de congestionamiento completo, a la forma en la que estas ondas avanzan en sentido opuesto al trafico a través de las diferentes celdas se le llama *efecto de transporte hacia atrás*.

Al considerar un tramo con rampas de entrada y salida, la complejidad del sistema crece. Existen modelos que permiten considerar este tipo de sistemas, algunos de ellos son el modelo de transmisión de celdas (cell transmission model CTM) [Daganzo, 1994, Daganzo, 1995] y el modelo de modo conmutado (switching-mode model SMM) [Muñoz et al., 2003]. Sin embargo el tratamiento matemático que hay que hacer en ambos es extenso.

4.2. Estimación de la densidad y flujo del tráfico vehicular mediante redes neuronales recurrentes

A pesar de los trabajos que se han realizado en esta área, aún no existe una técnica que modele fielmente la densidad y el flujo del tráfico vehicular, es por ello que se emplean las redes neuronales recurrentes para este fin.

La meta consiste en utilizar una red neuronal recurrente con la topología del perceptrón multicapa para modelar el comportamiento del tráfico en una autopista utilizando datos de tráfico obtenidos durante un día.

Para llevar a cabo el modelado del tráfico vehicular se cuenta con un conjunto de datos disponible para las fases de entrenamiento y prueba. Estos datos son mediciones que se tomaron cada 5 segundos a partir de las 5:00 am hasta las 12:00 horas del 25 de Abril de 2001 en el tramo representado en la figura 4.2 de la carretera I-210 West del sur de California, de aproximadamente 2 millas de largo, 4 carriles y tres estaciones con detectores pricipales etiquetados Myrtle (ML 34.05), Huntington (ML 33.05), Santa Anita (ML 32.20), además estaciones con detectores adicionales en las rampas. El flujo de la rampa de entrada en la celda i se denota por r_i y f_j representa el flujo sobre la rampa de salida en la celda j. En la figura 4.2, cada conjunto de círculos representa un detector de paso, de manera que se tiene el

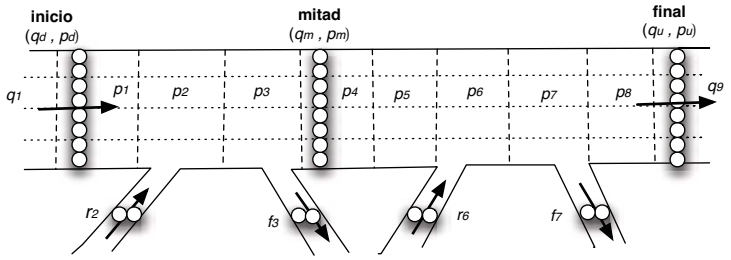


Figura 4.2: Segmento de autopista a analizar y selección de celdas

número de vehículos (nc) que entran a las celdas 1, 4 y 9, así como la densidad de automóviles que hay en estas; también se tiene la cantidad de vehículos que pasan por cada una de las rampas de entrada y salida y sus respectivas densidades.

La densidad ρ está definida como el número de vehículos por milla y la velocidad v se define como millas por segundo de esta manera el flujo q es igual al número de vehículos por segundo.

Para relacionar las medidas y las cantidades se hacen algunas suposiciones con respecto a $q_u, \rho_u, q_m, \rho_m, q_d, \rho_d$ y las medidas de flujo de cada entrada y salida de la rampa. Para el flujo y la densidad usadas en este modelo: (1) ρ_u es la densidad en la primera celda, es decir, $\rho_u = \rho_1$; (2) similarmente, $\rho_d = \rho_8$; (3) la densidad media ρ_m es la medida de ρ_5 entonces la estación media ML (Huntington) está en la celda 5; (4) r_i o f_j es el flujo de la rampa de entrada o salida correspondiente a la celda i ó j, respectivamente. [Muñoz et al., 2003].

Los datos usados en este estudio fueron obtenidos por el sistema de medición de desempeño (PeMS[1]). Cada detector proporciona medidas de volumen ($veh/tiempo$) y porcentaje de ocupanción.

Una condición necesaria para la estabilidad numérica es que los vehículos viajando a velocidad máxima no crucen múltiples celdas en un intervalo de tiempo, es decir, $vT_S \leq l_i$, $i = 1, 2, \ldots, n$. Además, se tienen las siguientes longitudes de las celdas [0.0880.3750.3750.1920.0880.2760.2760.246] millas. Tomando en cuenta lo anterior se determina un intervalo de tiempo $T_S = 5$ segundos.

Para contrarrestar el ruido en los datos PeMS de 30 seg, se utilizó un filtro pasa bajas de primer orden Butterworth con una frecuencia de corte de $0.01T_s^{-1}$ Hz que

[1]Performance Measurement System [PeMS]

se aplicó a los datos usando una técnica de filtrado de fase-cero hacia delante y hacia atrás. Una dificultad que se presenta al seleccionar la sección de prueba considera que es posible que no todos los detectores de pasos funcionen apropiadamente en el mismo tiempo. En el caso donde los detectores no funcionaban correctamente, los datos se corrigieron usando información de los sensores vecinos. La interpolación, el filtrado y el conjunto de correlación de datos se usan como entradas.

Algunos de los parámetros de las celdas usados en la simulación con el modelo conmutado, que se toman como base para este trabajo son: (v = 63 mph, Q_M = 8000 veh/h, ρ_J = 688 veh/mi) los cuales se evaluaron de las 5 am a las 12 pm usando medidas de densidad principalmente en lugar de densidades de celdas, con valores nominales para v, w, y ρ_J. v, Q_M y ρ_J se ajustaron subsecuentemente para mejorar la evaluación empírica y las medidas de flujos principales. w = 14.26 mph y ρ_c = 127 veh/mi se calcularon en función de v, Q_M, y ρ_J suponiendo que todos los parámetros deben satisfacer el triángulo del diagrama fundamental de la figura 4.1 [Muñoz et al., 2003].

4.3. Arquitectura de la red neuronal recurrente

En este trabajo se propone utilizar los valores de los datos de entrada y salida para estimar los valores del centro. De forma que se tienen las siguientes medidas: número de coches nc_d y el flujo q_d atrás (*down*), el número de coches nc_u y flujo q_u adelante (*up*), el flujo q_{r_2} de la rampa de entrada r_2, el flujo q_{f_3} de la rampa de salida f_3, el flujo q_{r_6} de la rampa de entrada r_6 y el flujo q_{f_7} de la rampa de salida r_7 como entradas para calcular las salidas, de esta manera se consideran que se tienen 8 variables como datos de entrada. Las salidas son flujo q_m y densidad nc_m en el punto intermedio (*medium*). Esto permitirá estimar el flujo y la densidad soportado por dicho segmento de autopista. En las gráficas 4.3 y 4.4 se puede observar el comportamiento del flujo vehicular y la densidad que van a estimar.

Para modelar el tráfico vehicular se supone que el flujo y densidad hacia atrás y hacia adelante (q_u, ρ_u, q_d, ρ_d) son conocidos, así como los datos del flujo de la rampa, mientras que la densidad media, ρ_m y el flujo q_m, se consideran como "faltantes" y es necesario estimarlas. La propuesta de esta prueba es determinar si usando una red recurrente se pueden reproducir exactamente los valores de ρ_m y q_m.

Para decidir las entradas con las que se alimentó al perceptrón multicapa recurrente, se tomaron en cuenta los efectos de transporte (tanto para adelante como para atrás) de las celdas; la condición de que T_S debe ser mayor al tiempo que los

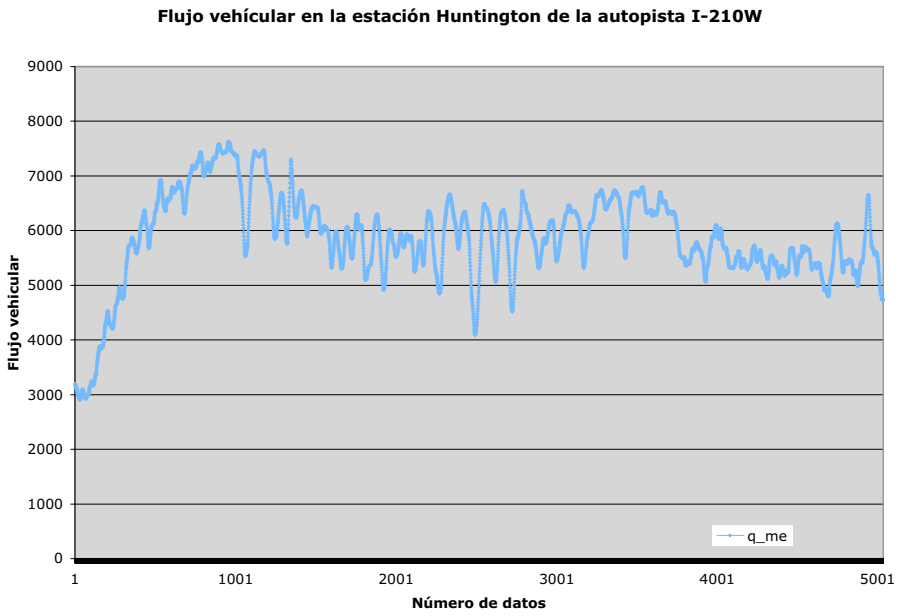


Figura 4.3: Flujo vehicular que se observó cada 5 segundos a partir de las 5:00 am hasta las 12:00 horas del 25 de Abril de 2001.

vehículos en movimiento tardan en atravesar múltiples celdas facilitó este trabajo.

Como se propone usar una red recurrente entonces además de las 8 variables de entrada, se tienen dos entradas extras que realimentan la red neuronal y que corresponden a la salida estimada de la red. De esta forma, se consideran diez neuronas en la capa de entrada ($I = 10$) y dos neuronas en la capa de salida ($O = 2$).

Como se comentó anteriormente, el conjunto de datos experimentales contiene un total de 5041 mediciones realizadas entre las 5:00 y las 12:00 hrs. de un día de tráfico, esto implica que el tamaño de la muestra (S) para la red neuronal sea 5041 datos.

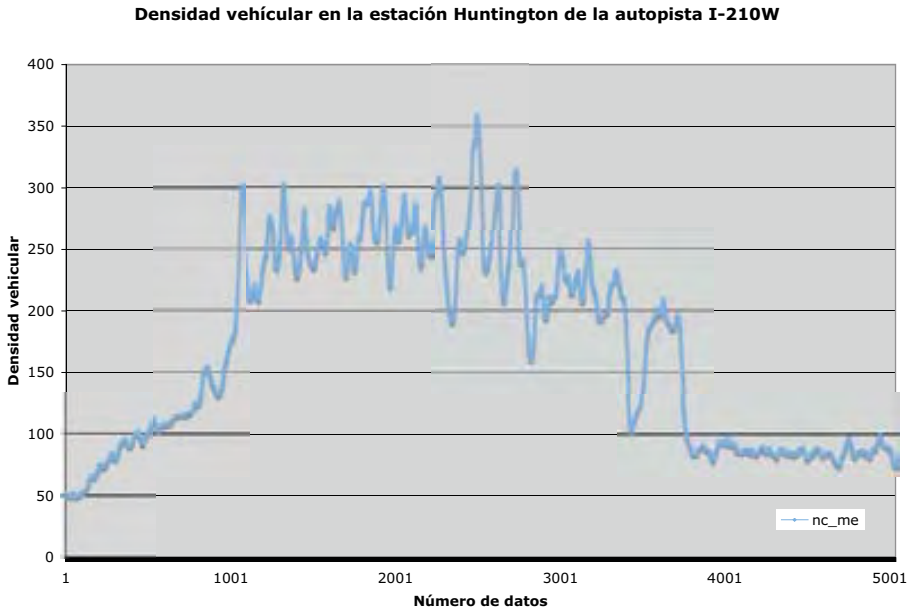


Figura 4.4: Densidad vehicular que se observó cada 5 segundos a partir de las 5:00 am hasta las 12:00 horas del 25 de Abril de 2001.

Utilizando la heurística que enuncia que "El número de conexiones debe ser una tercera parte del tamaño de la muestra", se obtiene que:

$$
\begin{aligned}
C = \lceil \frac{S}{3} \rceil &= 1680 \\
H &= \frac{C - O}{I + O + 1} \\
&= \frac{1680 - 2}{10 + 2 + 1} \\
&= 129,07
\end{aligned}
$$

El número de neuronas en la capa oculta (H) debe ser 129, tras una serie de experimentos se observó que este valor da una buena aproximación. Sin embargo, se obtienen mejores resultados con 200 neuronas en la capa oculta, por lo que se cambió a este valor.

En la figura 4.5 se puede observar la topología del perceptrón multicapa recurrente que se utiliza en estas tesis.

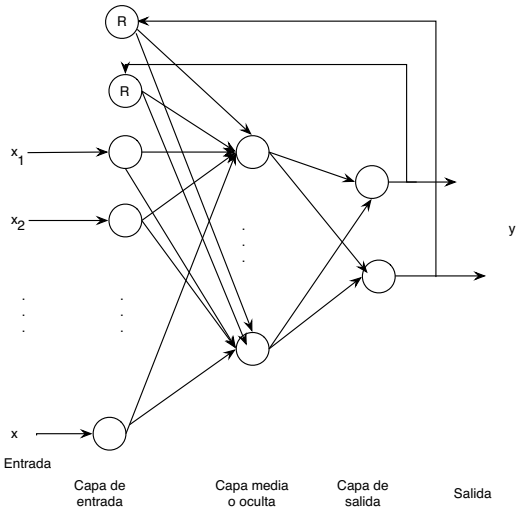


Figura 4.5: La arquitectura del perceptrón multicapa recurrente

También experimentando, se observó que se obtenían mejores resultados utilizando la función sigmoidal ó exponencial (4.5) que la tangente hiperbólica (4.6), por lo que se utlizó dicha función como función umbral para todas las pruebas.

$$y = \frac{1}{1 + e^{-x}} \tag{4.5}$$

$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.6}$$

Los pesos de la red neuronal se inicializaron con valores aleatorios entre -0.1 y 0.1 usando el generador de números aleatorios.

4.4. Resultados

En esta sección se presenta el resultado de entrenar a la red neuronal recurrente con muestras de diferentes tamaños utilizando los siguientes valores:

- 10 variables de entrada
- 2 variables de salida
- 0.1 como valor del momento
- 0.5 como tasa de aprendizaje
- 2000 iteraciones

Para que los datos estén un intervalo de [0,1] se buscó el valor máximo de los datos de entrada, el cual es 8456.07108 y este valor se utilizó para normalizar el conjunto de entrenamiento y prueba.

Tras una serie de experimentos se observó que entrenando la red neuronal con 129 neuronas en la capa oculta se obtiene una buena estimación para la variable del flujo, la cual se puede observar en la figura 4.6. Sin embargo, el resultado para la variable densidad difícilmente se ajusta a la salida deseada como se puede ver en la gráfica 4.7.

En la gráfica de densidad (figura 4.7) se muestra que para valores normalizados menores a 0.015 aproximadamente, la red es capaz de ajustarse y mostrar un comportamiento semejante a los valores reales, pero para valores mayores a 0.015 la red no alcanza los valores deseados, no obstante la diferencia entre estas curvas es aproximadamente de 0.02. Esto se debe a que en el tráfico vehicular existen dos comportamientos, como se ha mencionado en el capítulo 2, cuando hay flujo libre y congestionado, lo que demuestra que la red puede reflejar un comportamiento semejante al real para un solo tipo de flujo. Es decir, la red puede estimar los datos de manera aceptable cuando la densidad en la autopista no es crítica.

Por otro lado la figura 4.8 representa el error que se obtiene de la red neuronal después de 2000 ejecuciones. Este error, $\mathcal{E}(n)$, se toma cada 100 iteraciones y representa el error del algoritmo de aprendizaje (ver sección 3.4). En esta gráfica se puede observar que en las primeras 800 ejecuciones la red va disminuyendo el error pero llega un momento en que este valor se incrementa, es por ello que en los experimentos posteriores se reduce el número de iteraciones.

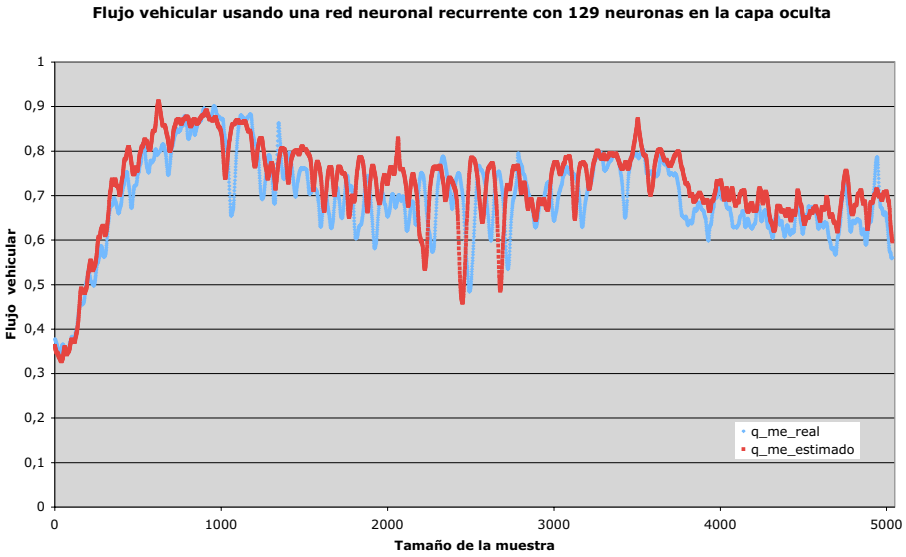


Figura 4.6: Comparación de los valores calculados por la red recurrente y los valores reales para la variable flujo.

Para corroborar que el comportamiento de la red es diferente cuando la densidad de la autopista es crítica se realiza el siguiente experimento. Tomando en cuenta el comportamiento anterior, se divide la muestra de entrenamiento en tres partes: 1) la primera parte comprende los datos del inicio de la gráfica 2) la segunda es la parte central, es decir con densidad crítica y 3) por último los datos finales.

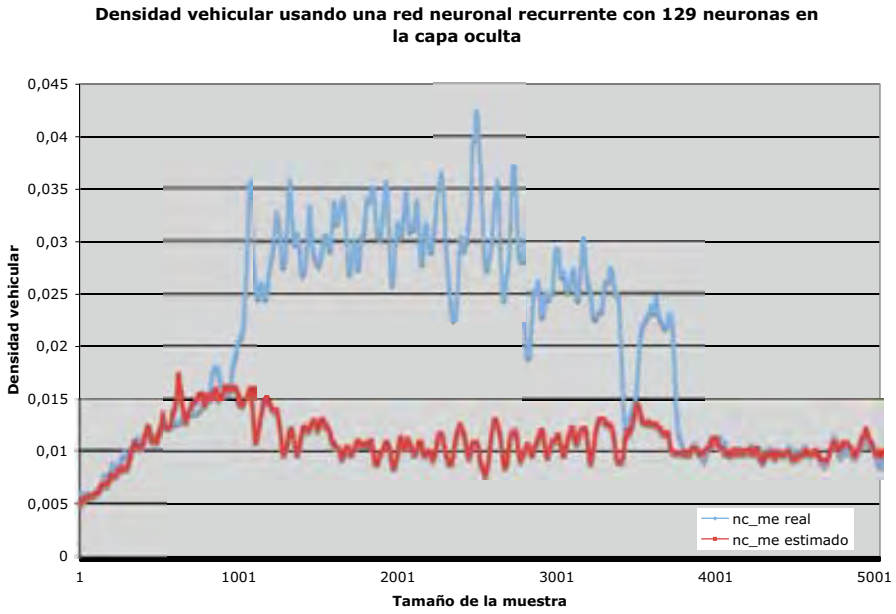


Figura 4.7: Comparación de los valores calculados por la red recurrente y los valores reales para la variable de densidad.

Para este experimento se tomaron en cuenta los mismos parámetros que en el caso anterior. Se usaron 129 neuronas en la capa oculta, 2000 iteraciones y sólo cambio el tamaño de la muestra.

Para la variable flujo (ver figura 4.9) se puede observar que el resultado estimado por la red neuronal se aproxima de manera aceptable a los valores deseados.

La estimación hecha por la red neuronal para variable densidad se presenta en la gráfica 4.10. En esta gráfica se puede observar como la red neuronal intenta reproducir un comportamiento semejante al de los datos deseados. Sin embargo, como en el caso anterior aún no se ajusta de manera aceptable. Con densidades muy bajas no hay problema, el resultado demuestra que estima muy bien los valores

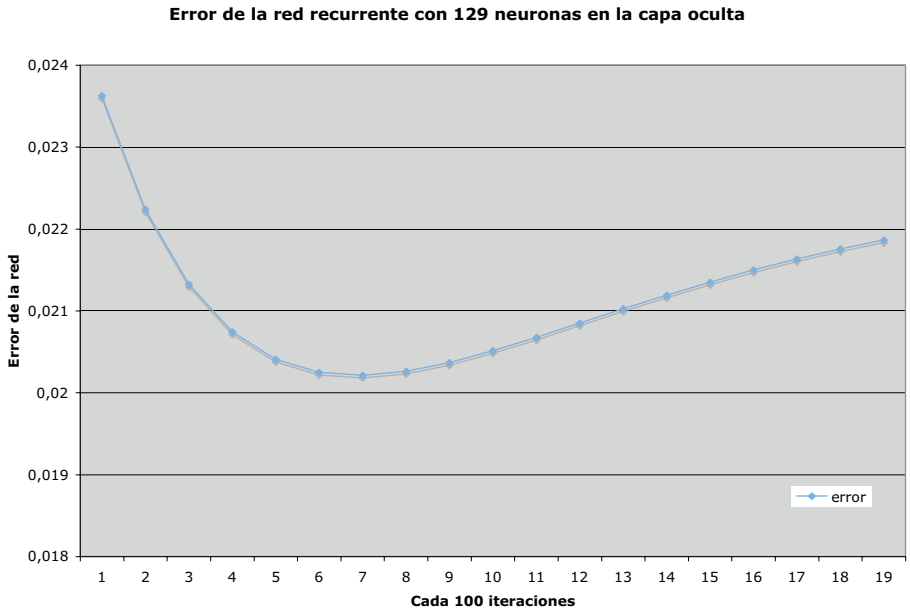


Figura 4.8: El error que se obtiene durante la fase de entrenamiento usando el algoritmo de retropropagación recurrente.

reales, pero cuando se incrementa la densidad la red muestra un comportamiento alejado de los valores esperados.

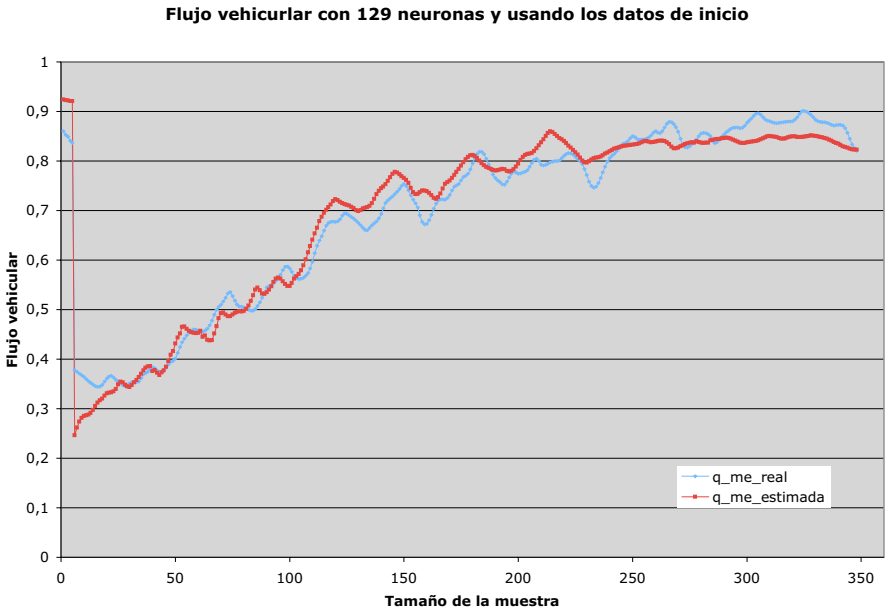


Figura 4.9: Resultado de la red recurrente para la variable flujo usando la primera muestra.

Con respecto al error del algoritmo de aprendizaje (figura 4.11) se puede ver que en las primeras 500 iteraciones el valor tiende a cero y después empieza a incrementarse.

La segunda parte de este experimento considera los datos donde la densidad es crítica. Para ello se seleccionaron 920 datos y se utilizaron los mismos parámetros: 129 neuronas y 2000 iteraciones, además se presenta el resultado del error de aprendizaje cada 100 ciclos.

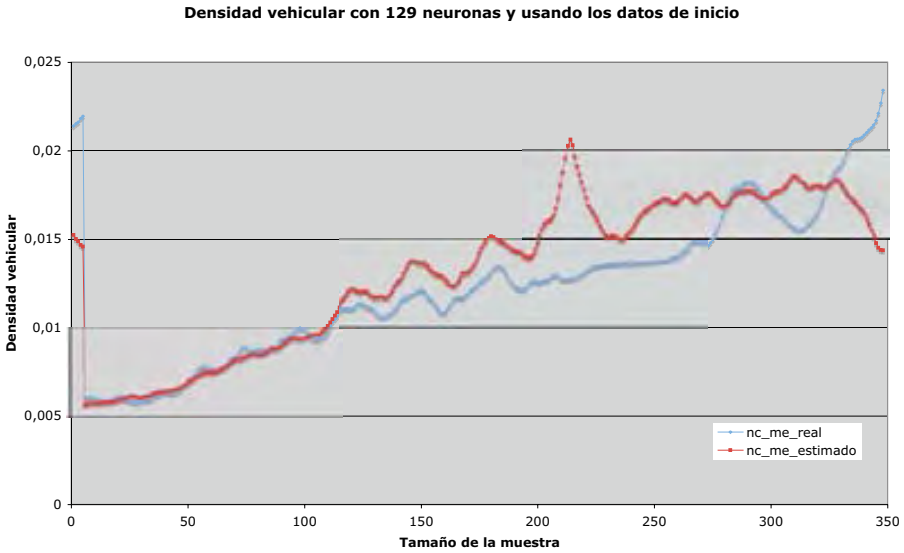


Figura 4.10: Resultado de la red recurrente para la variable densidad usando la primera muestra.

El resultado estimado por la red neuronal para la variable de flujo se muestra en la gráfica 4.12. Como se puede ver, la red trata de reproducir el comportamiento de flujo de manera semejante a los valores reales.

Para la variable densidad (ver gráfica 4.13) se observa que la red reproduce mejores resultados que en el experimento anterior, en donde la red se entrenó y probó con todo el conjunto de entrenamiento, de manera que la red es capaz de estimar los datos reales de forma más aceptable. Con esto se confirma que para los datos con un comportamiento semejante la red puede seguir un patrón parecido al real. Lo mismo sucedió con la gráfica 4.12 de flujo que también se aproxima a las datos reales.

Con respecto al error del algoritmo de aprendizaje de la red (figura 4.14), en las primeras 500 iteraciones tiende a cero y después empieza a incrementarse, de alguna

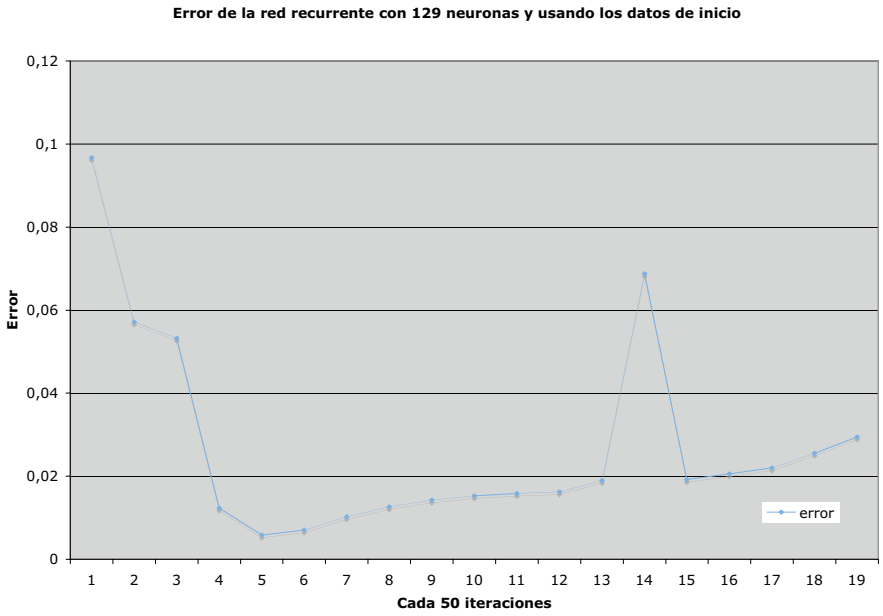


Figura 4.11: Error del entrenamiento de la red recurrente para la primera muestra.

manera tiene el mismo comportamiento que en el experimento anterior.

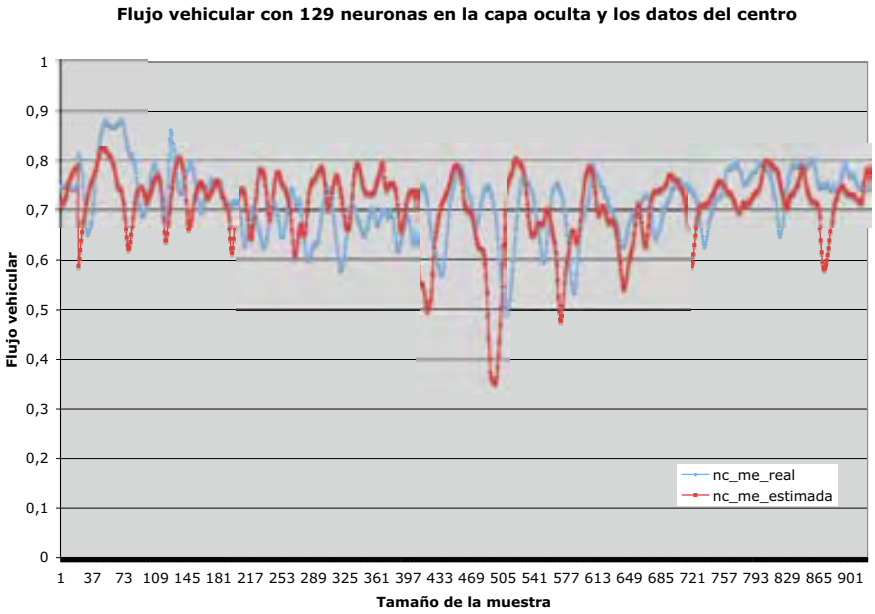


Figura 4.12: Resultado de la red recurrente para la variable del flujo con la segunda muestra (los datos del centro).

Para confirmar que la red reproduce valores semejantes a los reales dependiendo del conjunto de entrenamiento, se realiza el experimento usando los últimos datos del conjunto de muestra. Este experimento se realiza con 2000 iteraciones, 129 neuronas y 435 datos en el conjunto de entrenamiento.

A continuación se presentan los resultados obtenidos después de entrenar la red neuronal con el conjunto de entrenamiento que contiene los datos finales de muestra total.

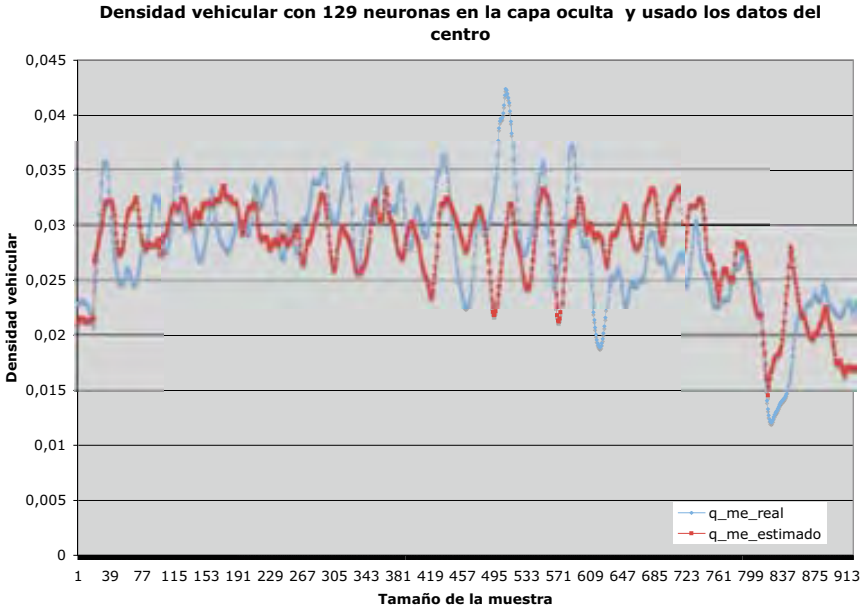


Figura 4.13: Resultado de la red recurrente para la variable densidad con la segunda muestra (los datos del centro).

Se puede observar en las figuras 4.15 y 4.16 la estimación realizada por la red neuronal para las variables flujo y densidad. En estás gráficas se aprecia que la aproximación hecha por la red para el flujo (figura 4.15) es bastante aceptable al reproducir de manera muy semejante los datos reales. Mientras que para la variable densidad (figura 4.16), a pesar de reproducir un comportamiento parecido a los datos reales sucede lo mismo que con los datos de inicio. Es decir, para valores mayores, en este caso de 0.013 aproximadamente, la red no alcanza a reproducir eso comportamiento.

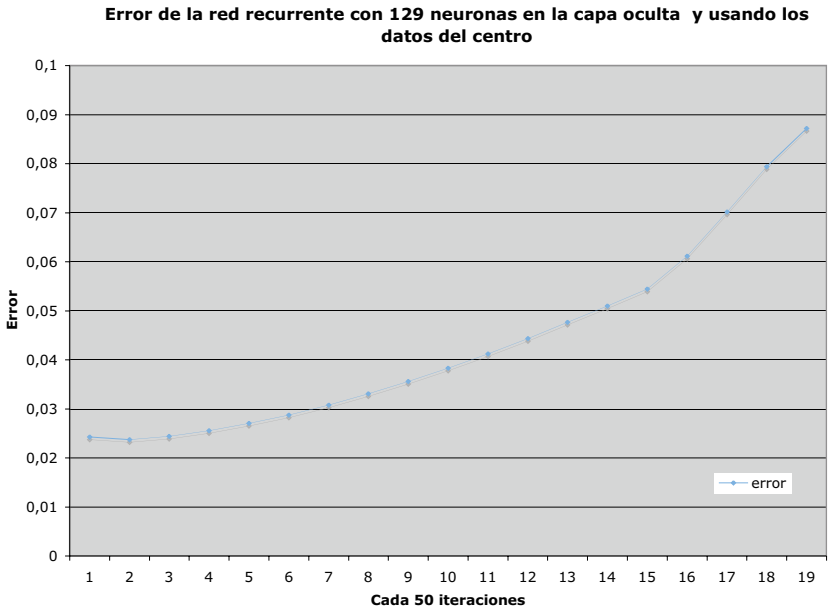


Figura 4.14: Error del entrenamiento de la red recurrente con la segunda muestra (los datos del centro).

Con respecto al error del algoritmo de aprendizaje (figura 4.17) se puede ver que en las primeras 900 o 1000 iteraciones el error tiende a cero y después oscila incrementándose y decrementándose.

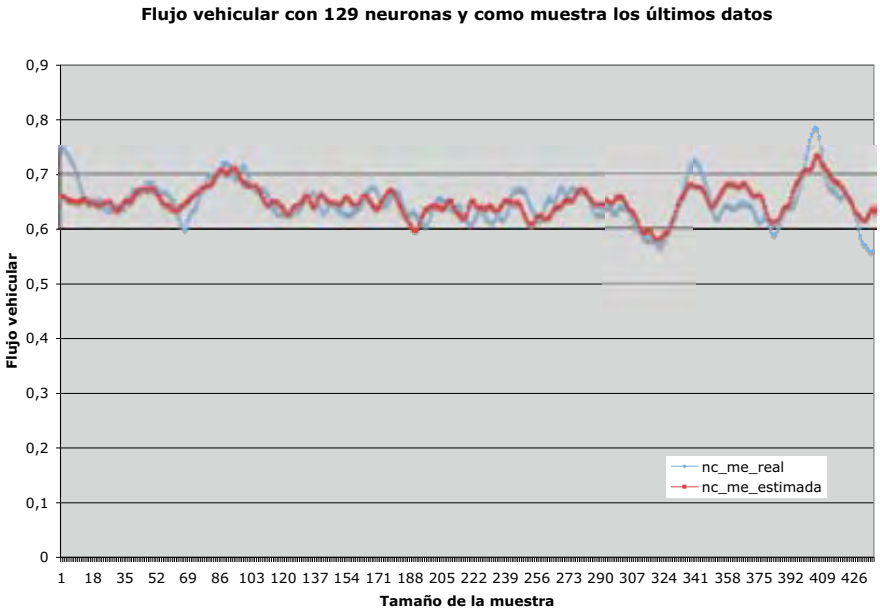


Figura 4.15: Resultado de la red recurrente para la variable del flujo con la última muestra (los últimos datos).

Considerando el comportamiento de la red neuronal en los experimentos anteriores, donde se obtienen mejores resultados al separar el conjunto de muestra, se pensó en usar un umbral para verificar si la red es capaz de reproducir los dos comportamientos; ya que se observó que separándolos la red produce una mejor estimación para diferentes densidades.

Para este experimento se tomó una muestra del 66 % de todo el conjunto de datos, el cual se empleó en la etapa de entrenamiento y el otro 34 % se usó en la etapa de prueba. Además se utilizó un umbral de 0.015 y se realizaron 2000 iteraciones con 129 neuronas en la capa oculta.

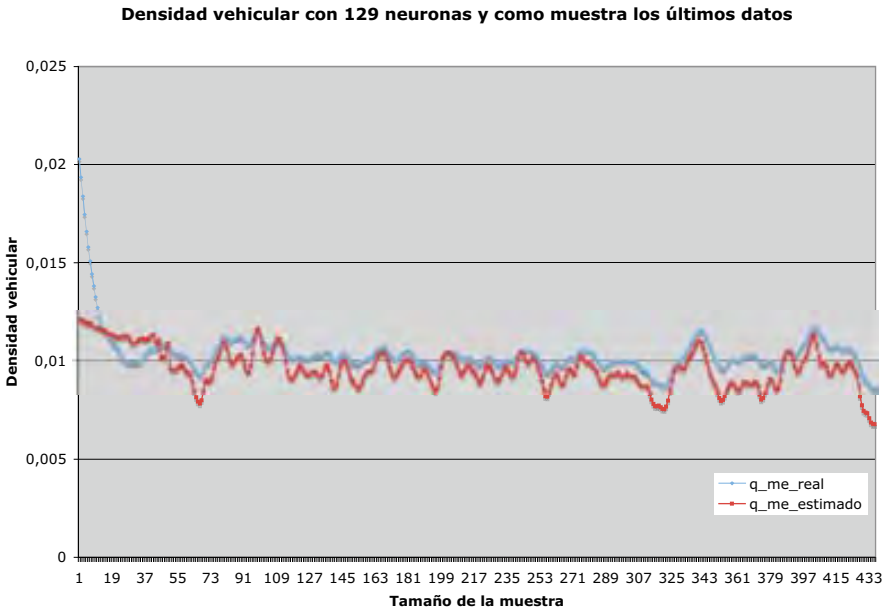


Figura 4.16: Resultado de la red recurrente para la variable densidad con la última muestra (los últimos datos).

Después de entrenar y validar los resultados obtenidos por la red se comparan estos valores con los valores reales (ver figuras 4.3 y 4.4) de los datos de entrada. Esta comparación se muestra en las figuras 4.18 y 4.19.

La figura 4.18 muestra la comparación de la variable densidad con sus valores estimados. Se puede observar que el resultado mejora considerablemente con respecto al primer experimento (ver gráfica 4.7). En este caso la red logra reproducir los dos comportamiento del tráfico vehicular: flujo libre y flujo congestionado.

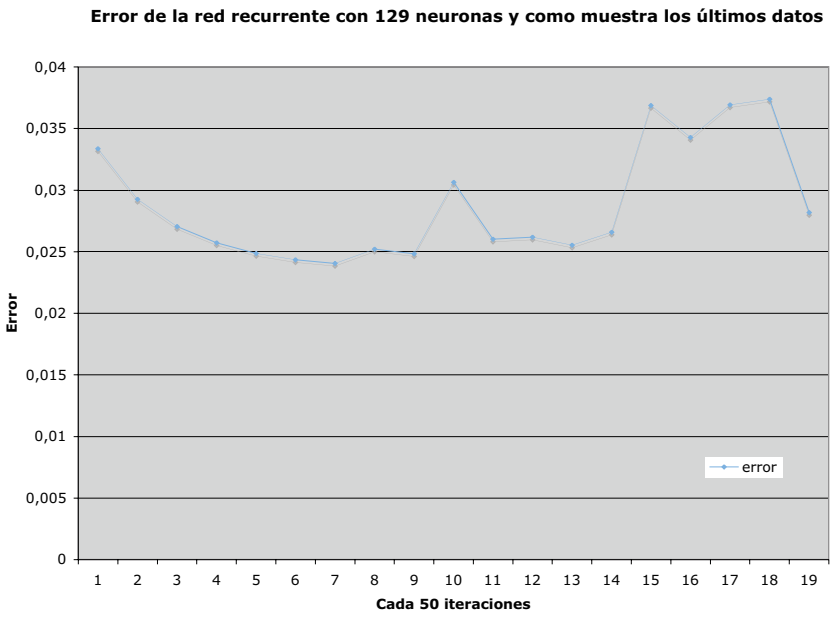


Figura 4.17: Error del entrenamiento de la red recurrente para la última muestra.

Para la variable flujo (gráfica 4.19) se tiene que el resultado de la red estima de manera semejante los datos reales. Sin embargo, se obtiene una mejor aproximación en el primer experimento (gráfica 4.6), en donde la estimación se ajusta de manera más fiel a los datos reales.

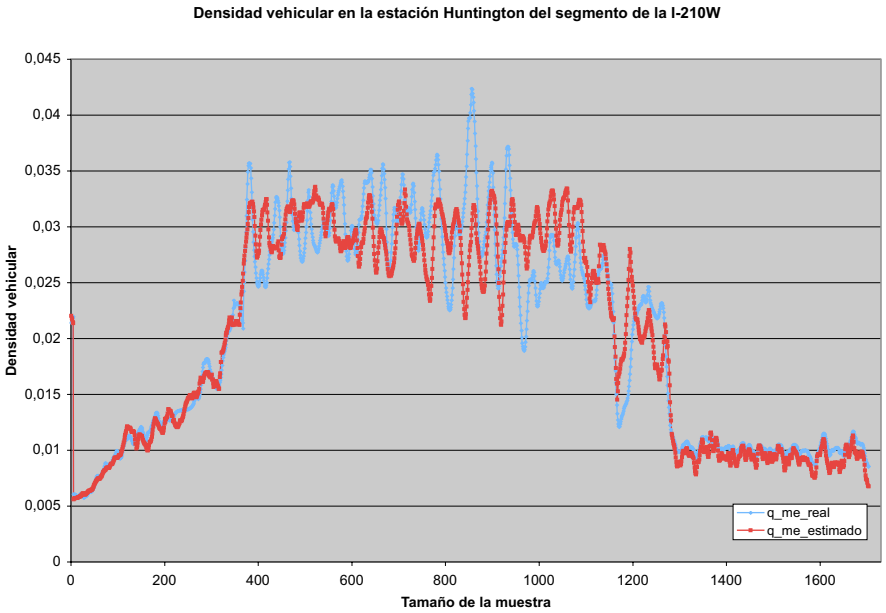


Figura 4.18: Comparación de los valores calculados por la red recurrente y los valores reales para la variable densidad usando un umbral.

En este experimento, también se tomó el error del algoritmo de aprendizaje cada 100 iteraciones, el cual se ve en la figura 4.20. Como se puede observar, el error tiende rápidamente a cero en las primeras seis iteraciones, después se incrementa lentamente, oscila incrementándose y decrementándose. Sin embargo, los resultados para las dos variables son más satisfactorios que en los primeros experimentos.

Buscando mejorar el resultado y basándose en las observaciones de los experimentos anteriores, se llevó a cabo la siguiente prueba. Se tomó todo el conjunto de entrenamiento, se buscó el máximo de cada una de las entradas y se normalizó cada entrada con el máximo obtenido. Es decir, si en las corridas anteriores se normalizó con el máximo de todo el conjunto de prueba, ahora se hace por cada variable de

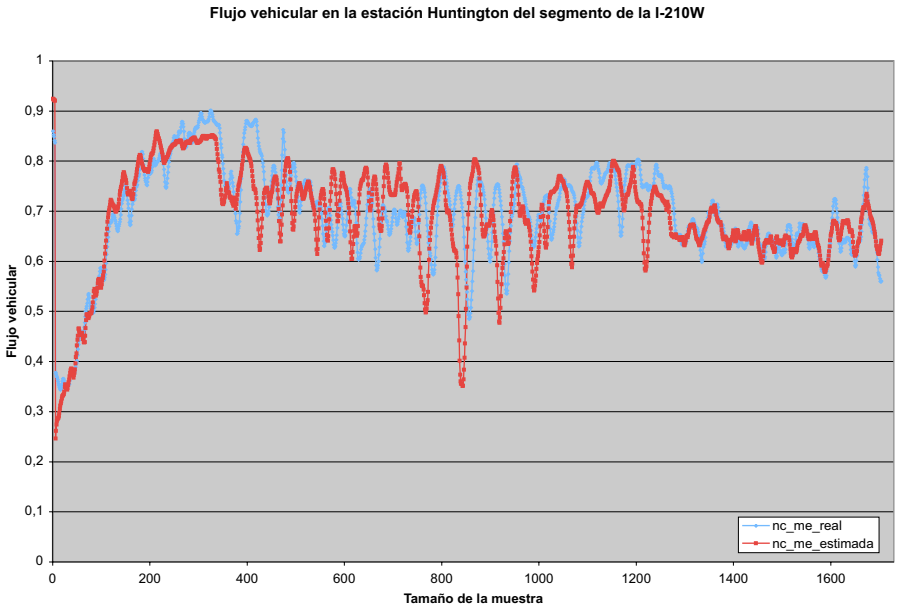


Figura 4.19: Comparación de los valores calculados por la red recurrente y los valores reales para la variable de flujo usando un umbral.

entrada, de forma que se tienen 8 máximos y con ellos se normaliza cada variable. Además, como se observó que el comportamiento del error tiende rápidamente a cero, en estas pruebas sólo se hicieron 500 iteraciones, no se considera ningún valor como umbral y se realizaron corridas por separado para cada variable.

El resultado de la prueba para la variable densidad se presenta en la gráfica 4.21. La figura 4.21 a) muestra la comparación de los datos reales con los estimados por la red, de manera que se aprecia una aproximación muy cercana entre estos datos. El error del algoritmo de aprendizaje se gráfica cada 50 iteraciones, también se puede observar como tiende a cero y despúes de 350 iteraciones empieza a incrementarse.

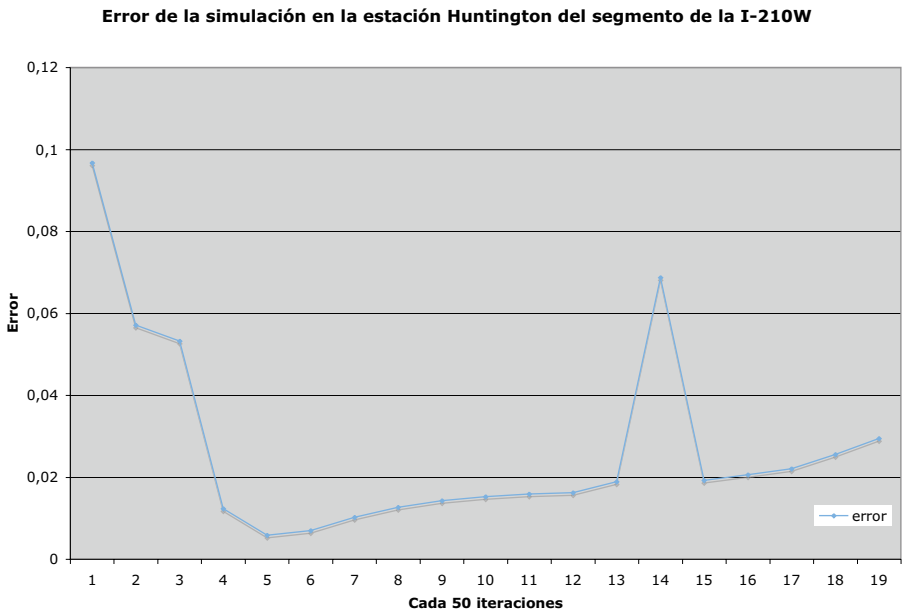


Figura 4.20: El error que se obtiene durante la fase de entrenamiento usando el algoritmo de retropropagación recurrente usando un umbral.

Con las mismas observaciones que se realizaron para la prueba de la variable densidad se llevaron a cabo para la variable de flujo. El resultado de esta estimación se puede ver en la gráfica 4.22. En la figura 4.22 a) se puede observar la aproximación de la red neuronal que intenta reproducir los valores reales, sin embargo, no se obtiene una estimación aceptable. Por otra parte, el error (figura 4.22 b) tiende a cero hasta la iteración 250 y después se incrementa.

En el último experimento se tiene un resultado aceptable para la variable densidad, mientras que para la variable de flujo el resultado trata de ajustarse a los valores reales, pero tiene oscilaciones muy grandes que se alejan de los valores reales y no proporciona una buena estimación.

Otro experimento que se realizó consiste en utilizar dos funciones gausianas para clasificar las entradas del flujo libre y flujo congestionado. Estas gausinas se aplicaron a los datos de densidad. La primera gausiana cubre el intervalo [0,150] con una media de 75 y una desviación estándar de 20 y la segunda gausiana cubre un intervalo de [150, 375] con una media de 204 y desviación estándar de 25. Estas gausianas se aplicaron a los datos de densidad. Una vez clasificados los datos se presentaron a la red neuronal, la cual se normalizó utilizando el máximo de cada entrada, se hicieron 500 iteraciones y se utilizo una sóla variable de salida. Los resultados que se obtuvieron se presentan en las graficas 4.23 y 4.24.

En la grafica 4.23 se presenta el resultado de la red neuronal utilizando 115 neuronas en la capa oculta, una sóla variable de salida y desnormalizando los datos con el máximo de todos los datos del conjunto de entrenamiento. Como se puede observar, el resultado de la densidad estimado por la red neuronal se ajusta de manera más aceptable a los datos originales. Con respecto a la gráfica del error, se puede ver que tiende a cero.

Por otro lado, en la gráfica 4.24 se puede observar el resultado de la variable flujo usando 200 neuronas y denormalizó la salida con el máximo de todos los datos del conjunto de entrenamiento. En este experimento se obtuvieron mejores resultados con 200 neuronas en la capa oculta, no obstante, se aprecia que el uso de las gausinas si influye en el resultado. Sin embargo, para fines de esta tesis se considera que el resultado del primer experimento, el cual se presenta en la figura 4.6 se brinda una mejor estimación que ésta.

Por lo tanto, con estos experimentos se observa que se tienen mejores resultados cuando se entrena la red con una sola variable de salida. Una red recurrente con 129 neuronas en la capa oculta y normalizando el conjunto de entrenamiento entre el máximo de todos los elementos (8456.07108) obtiene una estimación aceptable para la variable del flujo (figura 4.6). Mientras que usando un umbral y una red con 200 neuronas en la capa oculta y normalizando el conjunto de entrenamiento entre el máximo de cada entrada puede reproducir el comportamiento de densidad tanto para flujo libre como congestionado de manera aceptable (figura 4.21). Por otra parte, si se separa el flujo congestionado del flujo libre como se hizo en los experimentos de las gráficas 4.10, 4.13 y 4.16 y la densidad (ver las gráficas 4.9, 4.12 y 4.15) se puede pude concluir que la red hace una estimación aceptable para las dos variables, normalizando los datos entre el máximo (8456.07108) y usando 129 neuronas. Separando los datos del flujo libre del congestionado usando dos gausianas se obtiene una mejor estimación para la densidada (ver grafica 4.23).

En este capítulo se presentó el resultado de los experimentos realizados para modelar el tráfico vehicular en una sección de la carretara I-210W. En estos experimentos se estimó el flujo y la densidad del tráfico en la sección media de un segmento de autopista dado, a partir de los datos de entrada y salida. Los resultados fueron favorables con respecto a los valores reales.

En cada experimento se analizó el resultado para las variables flujo, densidad y el error generado durante la etapa de entrenamiento. Además se varió el tamaño del conjunto de entrenamiento y el valor con el que se normalizan los datos de entrada para trabajar en el intervalo [0,1].

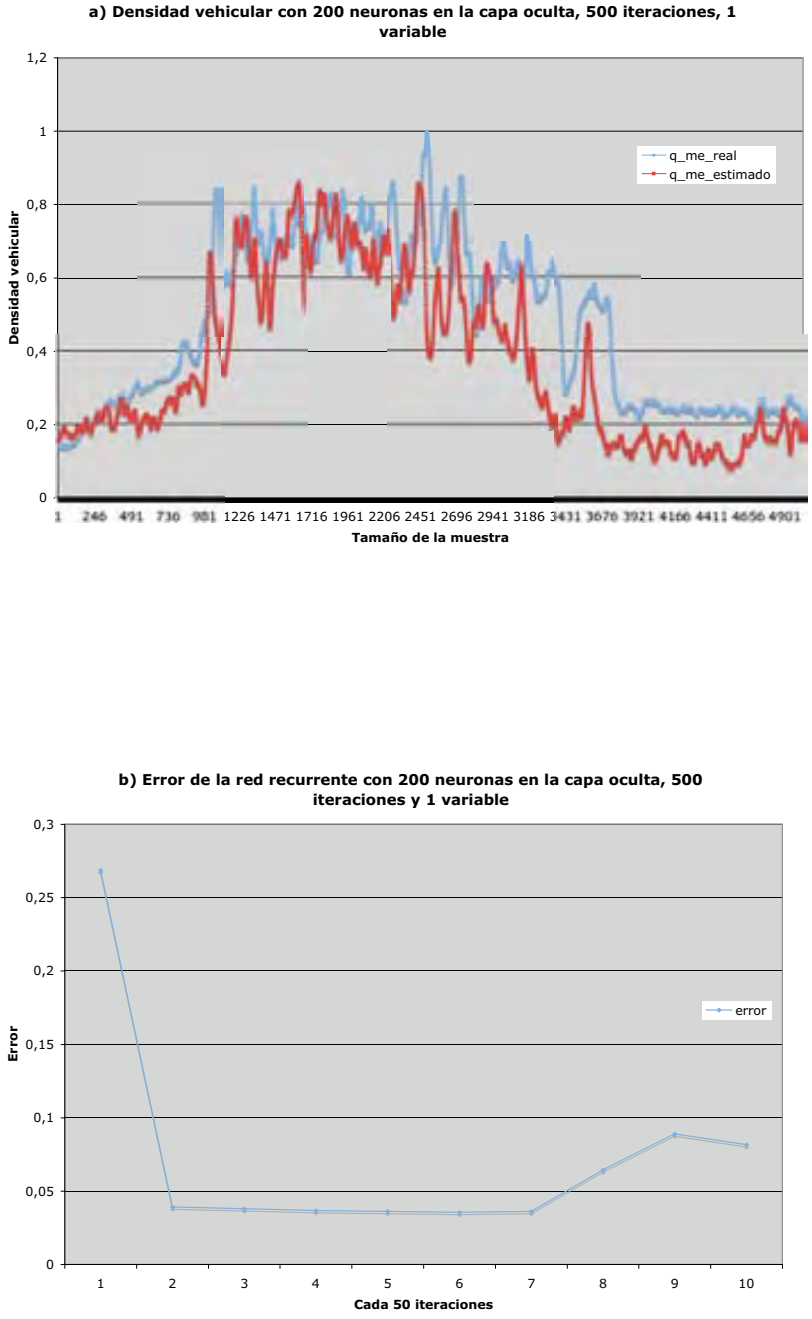


Figura 4.21: Comparación de los valores calculados por la red recurrente con los valores reales para la variable densidad sin umbral. b) El error del algoritmo de aprendizaje cada 50 iteraciones.

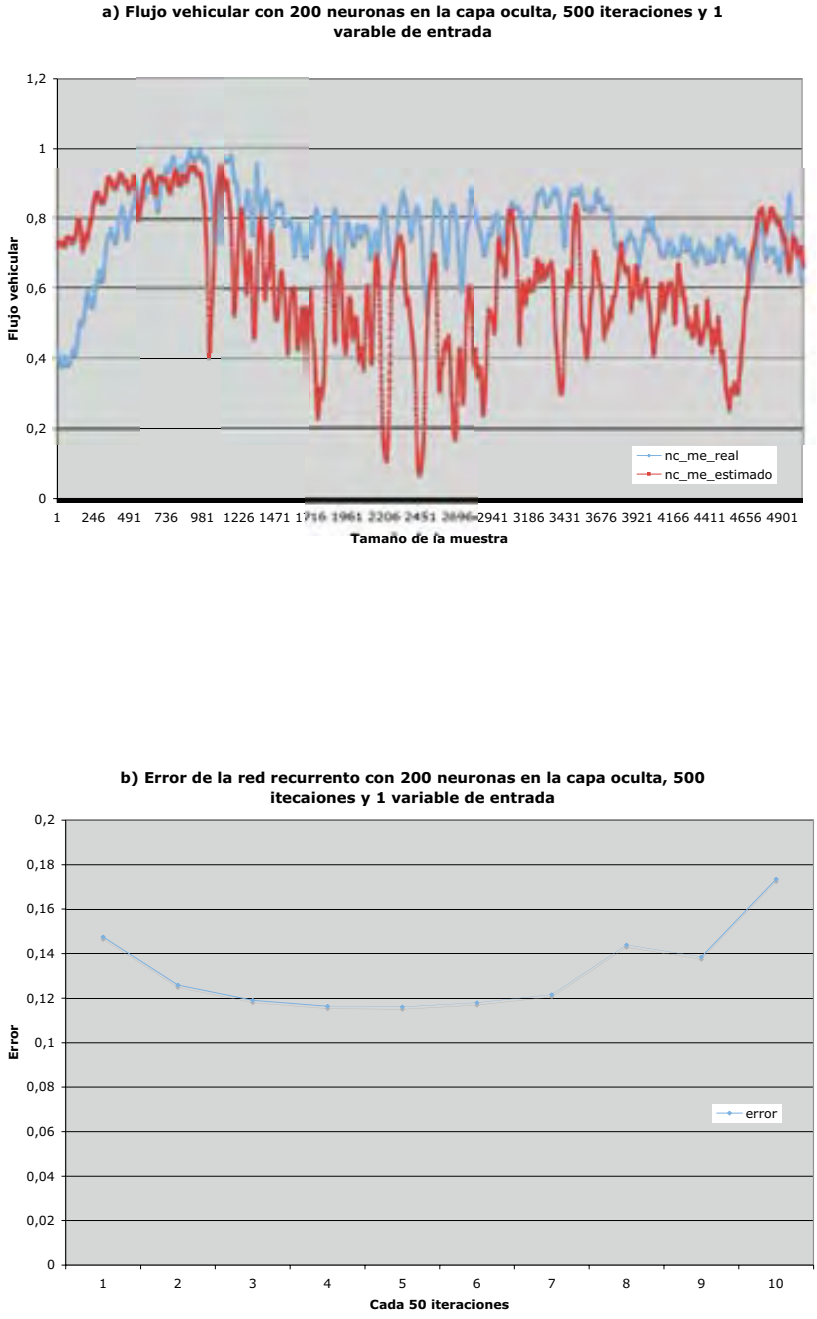


Figura 4.22: a) Comparación de los valores calculados por la red recurrente con los valores reales para la variable flujo sin umbral. b) Error del algoritmo de aprendizaje cada 50 iteraciones.

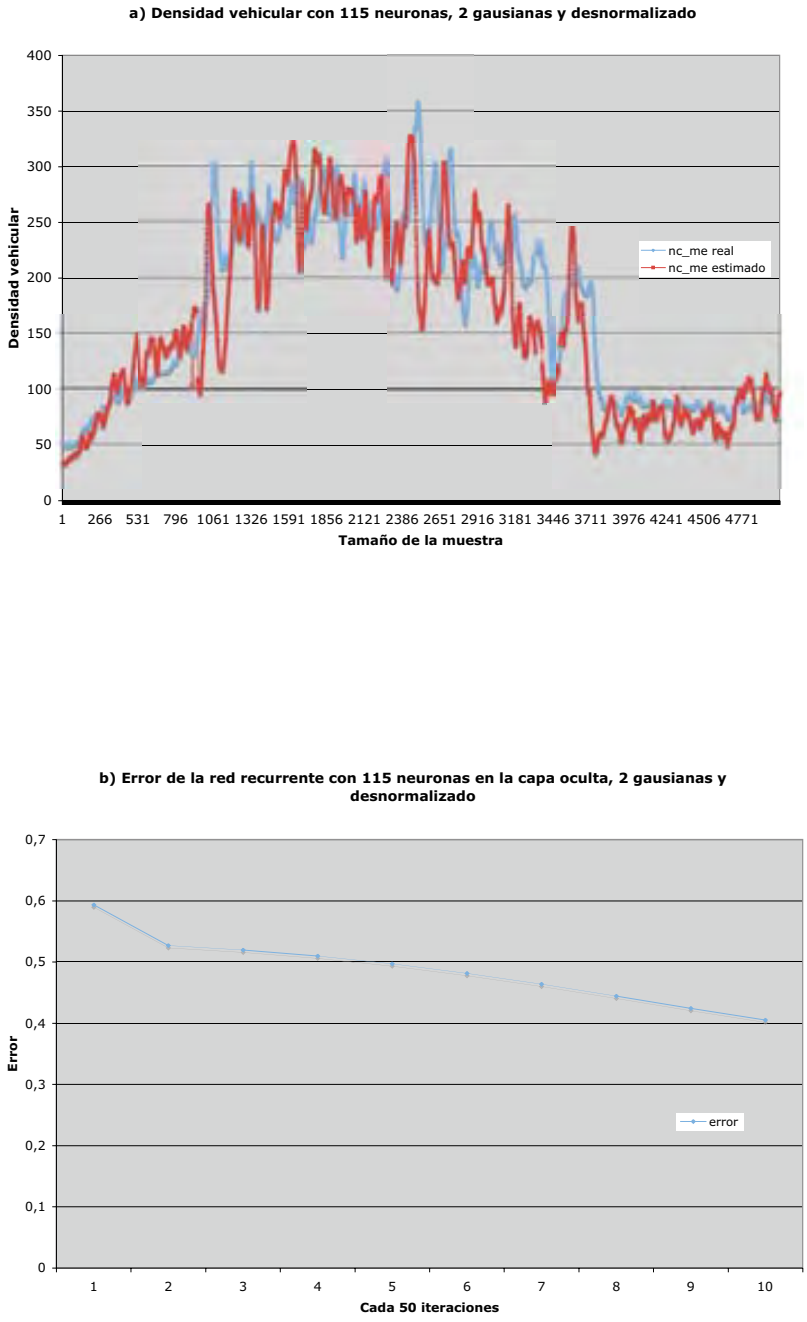
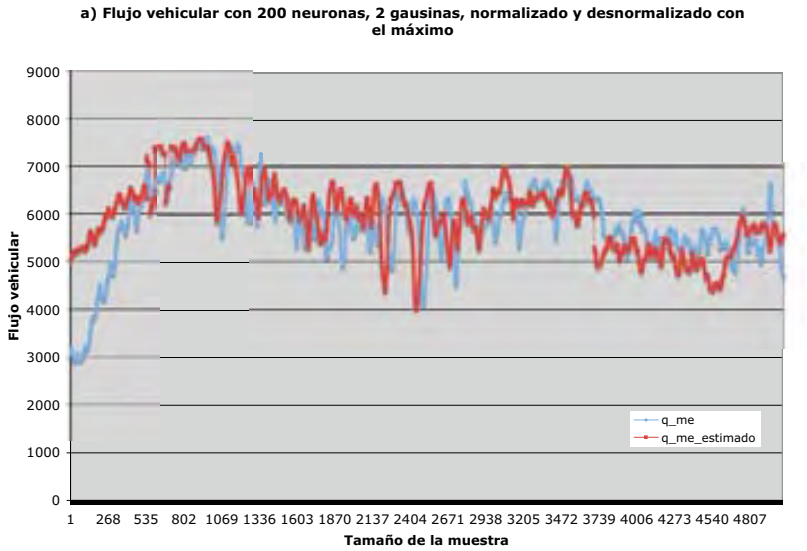


Figura 4.23: Densidad utilizando 115 neuronas en la capa oculta, una variable de salida y desnomalizando los datos con el máximo.



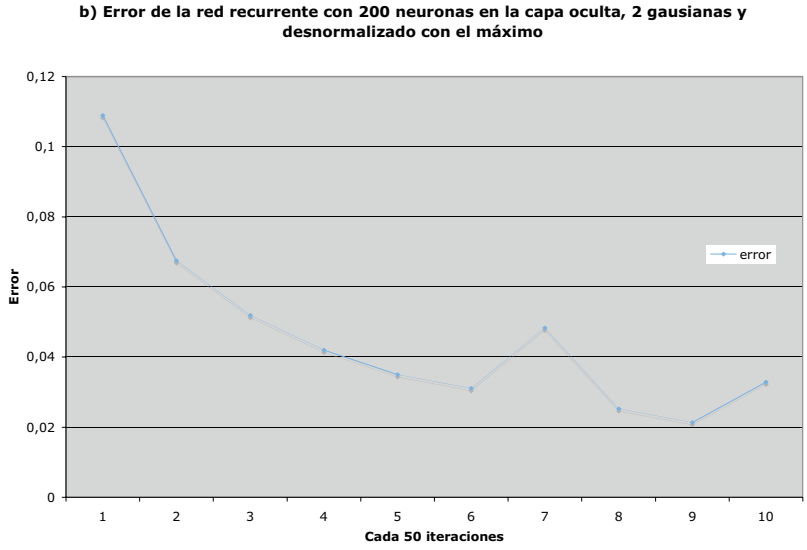


Figura 4.24: Flujo utilizando 200 neuronas en la capa oculta, una variable de salida y desnomalizando los datos con el máximo.

Capítulo 5

Conclusiones

En esta tesis se utilizaron las redes neuronales recurrentes para modelar la relación flujo-densidad en el tráfico vehicular. Este tipo de modelos no había sido empleado para representar funciones de estas variables de tráfico, se partió de la hipótesis de que las redes neuronales recurrentes poseen características que las hacen eficientes en la representación compacta de fenómenos dinámicos no lineales [Haykin, 1999, Fausett, 1994, Hassoon, 1995, Nelles, 2001].

Como resultado, se concluyó que las redes neuronales recurrentes son un modelo viable para el modelado de la relación flujo-densidad debido a su capacidad para almacenar información en estados interiores.

Se comparó esta forma de modelado con datos de tráfico medidos el 25 de abril de 2001 en la carretera I-210 West del sur de California a partir de las 5:00 am a 12:00 pm.

Se observó que la red recurrente estima satisfactoriamente la densidad vehicular del conjunto de prueba si se separa la estimación de los regímenes de tráfico libre y tráfico congestionado. El error del algoritmo de aprendizaje convergió rápidamente a cero.

También se observó que la red recurrente es capaz de reproducir el comportamiento del flujo libre y congestionado, empleando una red neuronal con una sola variable de salida y normalizando el conjunto de entrenamiento y usando un umbral apropiado para separar la estimación de los regímenes del flujo.

La forma en que se normalizaron los datos fue un factor importante para el desempeño de la red. Se observó que normalizando el conjunto de entrenamiento

con un máximo general, la red reproduce una salida satisfactoria para la variable flujo. Para reproducir la densidad fue necesario normalizar las variables de entrada con respecto al valor máximo de densidad en cada entrada.

Como trabajo futuro se puede explorar la posibilidad de reducir el tamaño de la red neuronal recurrente resultante y aplicar este modelo conjuntamente con algoritmos locales de control en rampas de acceso de autopistas.

Bibliografía

[Caudill and Butler, 1992] Caudill, M. and Butler, C. (1992). *Understanding Neural Networks: Computer Explorations*, volume 2: Advanced Networks. The MIT Press, London, England.

[Chang and Li, 2002] Chang, T. and Li, Z. (2002). Optimization of mainline traffic via an adaptive coordinated ramp metering control model with dynamic od estimation. *Transportation Research Part C*, 8.

[Corchado et al., 2000] Corchado, J. M., Díaz, F., Borrajo, L., and Fernández, F. (2000). *Redes Neuronales Artificiales, Un enfoque práctico*. Servicio de Publicacións de Univerdidade de Vigo, España.

[Daganzo, 1994] Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research B*, 28 B(4). Pág. 269 - 287.

[Daganzo, 1995] Daganzo, C. F. (1995). The cell transmission model, part ii: Network traffic. *Transportation Research B*, 29(2). Pág. 79 - 93.

[Di Fabbraro et al., 1995] Di Fabbraro, A., DiÑoto, D., Parisini, T., Recagno, V., Sacone, V., and Zoppoli, R. (1995). Neural networks for minimization of travelling time on freeway system. *Transportation system: Theory and application of advanced technology*, 2.

[Fausett, 1994] Fausett, L. (1994). *Fundamentals of neural networks Architectures, algorithms, and applications*. Prentice-Hall, United States of America.

[Hassoon, 1995] Hassoon, M. H. (1995). *Fundamentals of Artificial neural network*. The MIT Press, London England.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks A Comprehensive Fundations*. Prentice Hall, United States of America.

[Hegyi et al., 2002] Hegyi, A., De Shutter, B., Hellendoorn, H., and van den Boom, T. (2002). Optimal coordination of ramp metering and variable speed control.- an mpc approach. *Proceedings of the American Control Conference*.

[Hilera and Martínez, 1995] Hilera, J. R. and Martínez, V. J. (1995). *Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones*. RA-MA, Madrid, España.

[Kleinrock, 1975] Kleinrock, L. (1975). *Queueing system*, volume I. Jhon Wiley & Son, Canada. Pág. 155 - 178.

[Luo and Unbehauen, 1997] Luo, F. and Unbehauen, R. (1997). *Applied Neural Networks for Signal Processing*. Cambridge University Press, United States of America.

[Muñoz et al., 2003] Muñoz, L., Sun, X., Horowitz, R., and Alvarez (2003). Traffic density estimation with the cell transmission model. *Proceedings of the American Control Conference*, pages 3750 – 3755.

[Nelles, 2001] Nelles, O. (2001). *Nonlinear System Identifation. From classical Approaches to Neural Networks and Fuzzy Models*. Springer-Verlay Berling Heidelberng, Germany.

[Newell, 1993] Newell, G. F. (1993). A simplified theory of kinematic waves. i: general theory; ii:queuing at freeway bottlenecks; iii: Multi-destination flows. *Transportation Research 27B*, pages 281–314.

[Papageorgiou et al., 2002] Papageorgiou, M., Fellow, IEEE, and Kotsialos, A. (2002). Freeway ramp metering: An overview. *IEEE Transactions on Intelligent Transportation Systems*, 3(4). Pág. 271 - 281.

[Vaught et al., 1984] Vaught, R., Hurdle, V. F., and Hauer, E. (1984). *A traffic flow model with time time-dependent O-D patterns. In J. Vollmuller and R. Hamerslag (Eds.)*. VNU Science Press. Proceedings of the 9th International Symposium on Transportation and Traffic Theory, Utrech. Pág. 155 - 178.

[Zhang and Recker, 1999] Zhang, M. and Recker, W. (1999). On optimal freeway ramp control policies for congested traffic corridors. *Transportation Research Part B*, 33.

[Ziliaskopoulos, 2000] Ziliaskopoulos, A. (2000). A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation Science*, 34(1).