Universidad Michoacana De San Nicolás De Hidalgo

Facultad de Ciencias Físico Matemáticas
"Mat. Luis Manuel Rivera Gutiérrez "

Predicción de la posición de las fuentes de reproducción de mosquitos con el uso de dispositivos electrónicos e inteligencia artificial

T E S I S

PRESENTADA EN CUMPLIMIENTO DE LOS REQUISITOS PARA EL GRADO DE:

Licenciado En Ciencias Físico Matemáticas

PRESENTA:

Eduardo Mellado Villaseñor

DIRECTOR DE TESIS:
Dr. José Antonio González Cervera

Morelia Mich. junio de 2018

para mi familia

Índice general

Agradecimientos

A lo largo de esta tesis he conocido personas increíbles que han hecho que el proyecto que estoy desarrollando sea posible realizarlo. A estas personas los admiro mucho ya que me han enseñado el camino del bien, por así decirlo, ellos me instruyeron para seguir el camino de un investigador. Quiero darle las más sinceras gracias a cada una de las personas que voy a mencionar enseguida.

La primera persona que menciono se llama José Antonio González Cervera mejor conocido por su apellido González por la comunidad estudiantil. González es el asesor de mi tesis. Él me guió durante todo el trabajo y durante todo ese tiempo, ayudó a darme cuenta de errores cometidos, de decirme cuales eran las tareas importantes, es decir, aquellas en las que tenía que enfocarme con el fin de no perder tiempo en otras que no tenían importancia en ese momento, también me introdujo al hacer y desarrollar ciencia. Actualmente existen otros proyectos que yo espero en un futuro cercano se puedan realizar.

La siguiente persona se llama Francisco Siddhartha quien ha seguido de cerca el trabajo y al que también le he pedido ayuda a visualizar el entorno de trabajo. Siddhartha es una persona inspiradora.

A Venecia Chávez Medina por su colaboración dentro de este trabajo de tesis, ya que contribuyo a la simulación numérica lo cual es fundamental en este trabajo.

A mis amigos y futuros colegas que siempre están disponibles. Tales como Jesús Ortiz que es un excelente amigo y compañero de trabajo. A fieles y cercanos compañeros como Miguel Angel Ceja, Gabriela Sánchez, Jorge Antonio, Itzayana Guzman, Misha López y también aquellos que no menciono, ya que cada uno de ellos saben lo valioso que son para mí.

Quiero agradecer al grupo de física computacional conocido como fiscom del Instituto de Física y Matemáticas. Ellos a lo largo del proyecto me hicieron preguntas que me sirvieron para darme cuenta de importantes aspectos pequeños a los

que yo pasa desapercibido y así poder refinar mejor mi trabajo. En especial quiero agradecer a Mauricio Valencia por sus asesorías y consejos acerca del proyecto.

Hay un grupo especial al que les agradezco incondicionalmente su apoyo, ese grupo es mi familia. Mis papás Juventino y Blanca por ayudarme a sacar mi carrera y por darme un gran ejemplo de profesión. Mi hermano mayor Gabriel que ya es un Doctor en Física Óptica y por ser un pilar e inspirarme a terminar mi carrera, aparte él siempre estuvo disponible para apoyarme en lo que yo necesitaba. A mi hermano Juventino por su apoyo, su parecencia y orientarme a darle un mejor uso tanto laboral como persona. A mi hermana Yesenia por sus valiosos consejos.

Morelia MEX., 21 de junio de 2018

Resumen

El uso de dispositivos electrónicos y de modelos teóricos llevados a la aplicación real fue parte del desarrollo del trabajo de esta tesis. El principal problema que se resolvió, fue encontrar la ubicación de la fuente de reproducción de mosquitos. Se creo y diseño un dispositivo electrónico donde su principal tarea es medir variables climatológicas tales como la humedad, temperatura, precipitación, velocidad y dirección del viento, además del conteo de mosquitos. Estas variables climatológicas se relacionan con la existencia de los mosquitos, es decir, debido a estudios de científicos biólogos los mosquitos aparecen más cuando la precipitación es baja, cuando hay mucha humedad, cuando no hay corrientes de viento bastante fuertes y cuando la temperatura se encuentra entre los (18 a 25) °C . Otro punto en particular es los mosquitos aparecen a ciertas horas del día, en sí, los mosquitos frecuentan las horas entre las 5pm a 8pm ya que por lo general a esas horas las variables climatológicas tienden a decender. Con esta información fue posible utilizar de una manera más adecuada el detector de mosquitos. Debido a la insuficiencia de detectores se decidió utilizar un modelo matemático (en particular la ecuación de reacción-difusión) con el fin de aproximar el comportamiento de los mosquitos; los datos generados por dicha ecuación fueron moldeados y estructurados de tal forma que la red neuronal logrará ajustar todos los datos y así poder allegar a las ubicaciones de las fuentes de reproducción de mosquitos. Finalmente, los resultados mostrados para las aproximaciones de las fuentes de reproducción, están dentro del orden de un 75 % de efectividad. Por último, sabiendo que las herramientas utilizadas en este trabajo dan buenos resultados, se espera tener resultados parecidos para los datos reales generados por todos los detectores construidos en un futuro, además de que esta aplicación pueda beneficiar a comunidades tonto urbanas como rulares para la prevención de enfermedades peligrosas que portan lo mosquitos.

Palabras clave: Mosquitos, portadores de enfermedades peligrosas, inteligencia artificial, dispositivos electrónicos, caja contadora de mosquitos, redes neuronales, ecuación de reacción-difusión, modelo matemático.

Abstract

The use of electronic devices and theoretical models brought to the real application was part of the development of the work of this thesis. The main problem that was solved was finding the location of the mosquito breeding source. An electronic device was created and designed, where its main task is to measure weather variables such as humidity, temperature, precipitation, wind speed and direction, as well as mosquito counts. These climatological variables are related to the existence of mosquitoes, that is, due to studies of biologists, mosquitoes appear more when precipitation is low, when there is a lot of humidity, when there are no strong wind currents and when the temperature is between (18 to 25) °C. Another point in particular is the mosquitoes appear at certain times of the day, in itself, mosquitoes frequent the hours between 5pm to 8pm since usually at that time the climatic variables tend to decender. With this information it was possible to use the mosquito detector more appropriately. Due to the insufficiency of the detectors it was decided to use a mathematical model (in particular the reaction-diffusion equation) in order to approximate the behavior of the mosquitoes; the data generated by this equation were molded and structured in such a way that the neural network will manage to adjust all the data and thus be able to find the locations of the sources of mosquito reproduction. Finally, the results shown for the approximations of the reproduction sources are within the order of 75 % of effectiveness. Finally, knowing that the tools used in this work give good results, it is expected to have similar results for the real data generated by all the detectors built in the future, besides that this application can benefit urban silly communities as rulares for the prevention of dangerous diseases that carry mosquitoes.

Capítulo 1

Introducción

La abundancia en el número de mosquitos es un problema que se conoce desde hace muchos años. Últimamente, atrae la atención de mucha gente, entre ellos científicos y biólogos ya que estos insectos suelen transmitir enfermedades peligrosas que puede llevar a la muerte de animales o de seres humanos, si es que no se llevan tratamientos adecuados.

Existe una gran variedad de especies de mosquitos en todo el mundo. En particular en México las especies existentes es el Mosquito Anopheles de la familia Culicidae que transmite la enfermedad conocida como malaria. El mosquito conocido como Culex pipiens que es un mosquito común y por lo general se encuentra en las zonas urbanas. Otro mosquito es conocido como mosquito culex y este en particular son portadores de importantes enfermedades, como el Virus del Nilo Occidental, filariasis, encefalitis virales y la malaria aviar. Recientemente se ha hablado de un mosquito portador de virus del dengue y de la fiebre amarilla, así como de otras enfermedades, como la chikunguña y la fiebre de Zika, este mosquito es conocido como Mosquito Aedes Aegypti y prácticamente existe desde la parte norte de México hasta la parte sur de Argentina.

El estudiar los hábitos de los mosquitos requiere de diferentes técnicas de análisis, los biólogos se han dado cuenta de como atraerlos para llevarlos a una trampa y ser eliminados, también se han inventado químicos para asfixiar a los mosquitos y conducirlos a una muerte segura. Sin embargo, esto causa daños al ambiente, incluyendo a seres humano y/o animales.

Es evidente que algunos métodos para eliminar a mosquitos no son muy eficientes y que el uso de alguno de ellos no nos da información como para conocer su comportamiento, en otras palabras, es bueno conocer en donde se están re-

produciendo o en donde hay abundancia de mosquitos. De acuerdo a estudios de científicos biólogos, la mayor cantidad de mosquitos por lo general se encuentran en lugares húmedos y con cierta temperatura.

Una de las técnicas para contar mosquitos es crear una trampa que contenga CO_2 [1]. El cual se puede generar de diferentes formas. La forma de capturar mosquitos depende del sistema creado, se puede utilizar cosas como telas adesivas, tubos en que solo tenga una sola dirección, es decir, si el mosquito entra por el tubo este ya no pueda salir, también se puede usar un ventilador en que el mosquito pase a través del ventilador y como la dirección del viento que se genera se dirige hacía la parte interna de la trampa esto evita que el mosquito se escape [2].

Las trampas descritas anteriormente presentan ciertas ventajas y desventajas, pues de usar alguna de ellas, resulta que hay mas desventajas que ventajas. La primera es que para contar la cantidad de mosquitos se tiene que hacer de manera manual y esto en algún momento se dejaría de hacer debido a que el físico experimental sufriría frustración o algo por el estilo. La segunda es que cada vez que se tenga que recolectar el conteo de mosquitos, se tiene que limpiar todas las trampas, para evitar futuros errores de conteo. La tercera es que se tiene que poner la trampa en un lugar estratégico para que no vaya a ser destruida por algún agente externo. La cuarta es el costo que implicada cada trampa, es decir, su mantenimiento es demasiado caro. La quinta es que solo se tiene información acerca del número y posiblemente de la especie de los mosquitos. Por último la única ventaja es que se tiene el conteo de mosquitos. Así que, contar mosquitos no es una tarea nada fácil y si se haría de este modo, tarde o temprano fracasaría dicha estudio. Debido a que existe la posibilidad de no hacer una instalación adecuada de la trampa.

En consecuencia a lo anterior, la única información que se tiene es la cantidad de mosquitos, sin embargo, no se podría decir por que los mosquitos aparecen en la región donde se instalaron las trampas, también sería prácticamente imposible saber cual es la fuente de reproducción solo con esa información. Esto nos sugiere la idea de recabar información diferente respecto al conteo de mosquitos.

De acuerdo a investigaciones anteriores hechas por investigadores en el área, los mosquitos aparecen en ciertas regiones que depende del habitad [3]. Debido a que no es tan factible construir trampas manuales, es necesario construir una *trampa digital* que permita hacer tareas complicadas como recolectar información acerca de las variables climatologías como también el conteo de mosquitos, en el que se requiera hacer un mantenimiento mínimo. En el capítulo 2 se da a más detalle los pasos que se siguieron para construir una trampa digital y en el mismo capítulo se renombra como *caja contadora de mosquitos*. La trampa digital ofrece

muchas ventajas ya que puede ser diseñada para que sea lo más autónoma posible, es decir, un sistema de auto-evaluación que pueda enviar reportes de fallas o de estabilidad. En capítulo 2, se habla una poco acerca de este diseño.

El estudio de mosquitos se ha hecho en diferentes partes del mundo, tales como en Japón, Corea del sur, Australia y Estados Unidos. La forma en la que se hace este tipo de análisis es utilizando un algoritmo de la rama conocida como Inteligencia Artificial llamado Redes Neuronales Aritificales. Este algoritmo esta hecho en base a la estructura de las neuronas del cerebro humano y/o animales. En sí, el algoritmo es básicamente la forma mas simple de una neurona biología, lo que la convierte en una neurona artificial, y una de las diferencias entre la biológica y artificial es la manera en la que aprende, es decir, la neurona artificial no aprende cosas solo hace ajustes basado en modelos matemáticos. En el capítulo 3, se desarrolla la estructura básica de un conjunto de neuronas artificiales que son conocidas como *redes neuronales*. Una de las aplicaciones de las redes neuronales se trata de modelar la abundancia de mosquitos en regiones urbanas [4]. Desarrollado en Corea del Sur, en el que se modelan las variables climatologícas y la cantidad de los mosquitos en ciertas épocas del años. El objetivo principal es predecir la abundancia de estos respecto a el habitad y su cambio del clima para así poder tomar medidas preventivas para evitar que los mosquitos transmitan enfermedades peligrosas. Otra aplicación respecto a las redes neuronales es predecir la población de Culex (una familia de mosquitos) bajo ciertos datos meteorológicos [5]. Realizado en el norte de Australia en la que se estudia por series de tiempo capturados en 6 zonas diferentes.

Como se ha descrito anteriormente, el trabajo en esta tesis consta en predecir la ubicación de fuentes de reproducción de mosquitos y esto se convierte en el problema de interés a resolver. Lo cual, en el capítulo 4 se inicia modelando el problema de interés de este trabajo para finalmente mostrar los resultados, cabe mencionar que los datos generados dentro de este modelo matemáticos fueron hechos por Venecia Chavés Medina. Finalmente el capítulo 5 se muestran las conclusiones y futuros avances.

Capítulo 2

Modelo captura de mosquitos

Actualmente existen herramientas computacionales que permiten cincelar dicho proyecto. Como ejemplo de la utilidad de estas herramientas es que se construyeron dos modelos virtuales: El primer modelo se describe en el [Anexo A] y ahí se explica el motivo por el cual no se utilizó este diseño; el segundo modelo es totalmente diferente al primero. Una de las cosas principales que cambiaron fue su forma, es decir, el primer prototipo tiene parecido a un prisma rectangular mientras que el segundo prototipo cuenta con una base o cimiento. Por otro lado ambas cajas siguen conteniendo los mismos componentes como; los sensores, el panel solar y el sistema de captura. En el resto de este capítulo se describen con mejor detalle.

2.1. Visualización virtual

Una de las grandes ventajas de tener un modelo virtual orientado a modelos de construcción es que es posible detectar errores de diseño antes de ir directamente a construirlo de manera física. Los modelos descritos anteriormente se realizaron con herramientas computacionales de diseño como Google SketchUp, AutoCad, ProgeCad, TurboCad, entre otros. Estas sirven para modelar objetos en 3D y cada uno de ellos cuenta con cierta complejidad de uso.

La herramienta que se utilizó para hacer los modelos virtuales descritos es **Google SketchUp**. Ya que esta herramienta es la más fácil de utilizar en comparación con los otros ya mencionados. Entonces, SketchUp es bastante útil para modelar proyectos en 3D, ya que nos permite visualizar mejor el entorno del proyecto dise-

ñado. Una de las ventajas de usar este programa es que se puede navegar dentro del cuerpo del proyecto, lo que nos permite analizar la ubicación de los componentes electrónicos como los sensores de medición y así poder ubicarlos para obtener el mejor espacio entre los objetos. Otra de las ventajas es que una vez finalizado el proyecto es posible imprimirlo usando una impresora 3D y de esta manera, poder refinar y detectar posibles errores de construcción. No obstante, para este propósito no se imprimió en 3D ya que solo se necesitaba tener una visualización del modelo a construir.

Antes de empezar a diseñar, se necesitaba saber que componentes electrónicos va a contener la caja contadora de mosquitos, con el fin de saber la distribución de éstos dentro de la caja. Después, es importante la ubicación de los detectores o cajas contadoras de mosquitos y analizando varias posibilidades, es conveniente poner las cajas en las azoteas ya que no es un lugar donde estorben y puedan ser dañadas, pues lo ideal sería ponerla en un lugar donde haya jardín o humedad, en dado caso de que se decidiera colocarlas en estos sitios, se tendría que poner protección como una jaula para protegerla de animales o incluso de personas o niños. Las cajas se pondrían en las azoteas de los edificios de la Universidad Michoacana de San Nicolás de Hidalgo, esto implica que no en todos los edificios se tendrá un fácil acceso a la luz eléctrica lo que obliga a que las cajas sean autosustentables. Como su nombre lo dice requiere el uso de paneles solares y sus respectivos componentes.

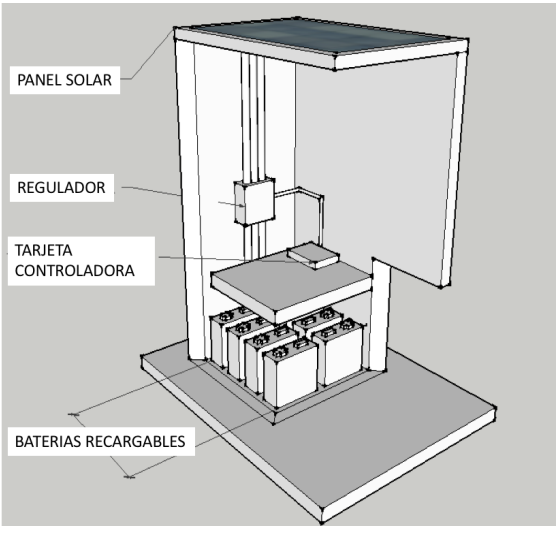
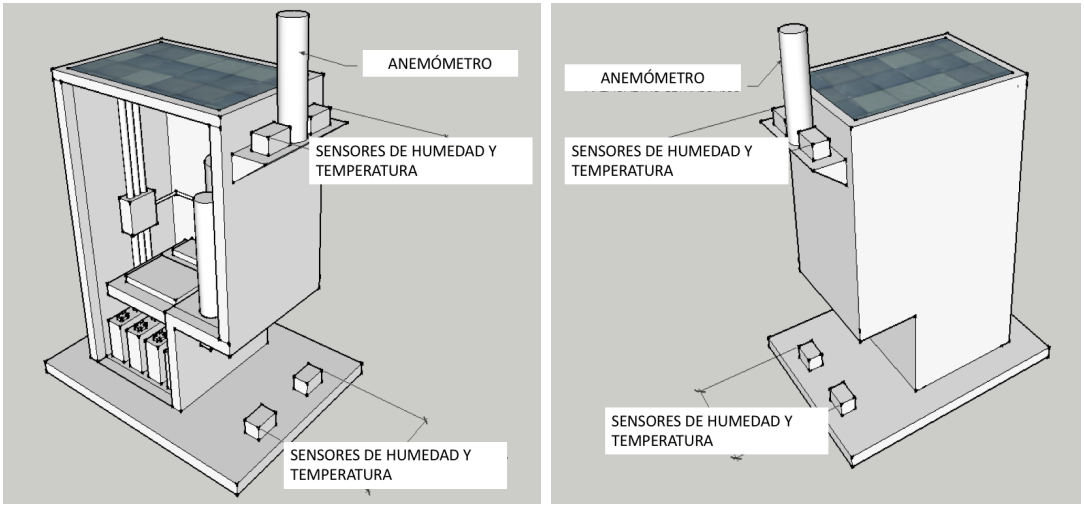


Figura 2.1: Vista externa, el panel solar se ubica en la parte superior de la caja mientras que las baterías se colocan en la parte baja. En medio de ellos colocamos el regulador.

En un principio la caja contendrá un único panel solar como se muestra en la Figura. 2.1, además se muestran componentes como un regulador de carga y la ubicación de las baterías. En la misma Figura 2.1 se pueden ver varias baterías aunque no se necesiten todas, en sí solo es para indicar la posición de ellas.

Los mosquitos se puede ver en ciertos lugares y estos aparecen dependiendo del clima [3]. Se sabe que todas las especies de animales existentes, viven en ciertos lugares debido al confort térmico de cada especie. Por ejemplo un oso polar no podría vivir en un lugar cálido, un pez no podría vivir fuera del agua, y así sucesivamente. Adicionalmente, las investigaciones que se han hecho acerca de los mosquitos [4] demuestra que dependen de algunas variables climatológicas tales como la humedad, precipitación, temperatura, velocidad y dirección del viento. Midiendo estas cantidades se puede estudiar las correlaciones que existen entre ellas y la cantidad de mosquitos, así mismo una de las ventajas es que existen sensores electrónicos que las miden. En el capitulo 4 se explica a más detalle por que se necesitan estos sensores.

Hoy en día, existen sensores independientes que son capaces de medir una o más variables climatológicas, se el caso de las variables como la temperatura y la humedad. En las Figuras 2.2a y 2.2b se observa la ubicación de sensores independientes con dos perspectivas diferentes. Siendo así, estos sensores se trataron de tal manera que no estuvieran cerca de las tarjetas controladoras del sistema o del regulador ya que éstos generan calor y por lo tanto alterarían los datos reales.



(a) Vista externa, perspectiva izquierda. (b) Vista externa, componentes de la caja.

Figura 2.2: Se muestran distintas perspectivas de la posición de los sensores.

Por último y más importante de la caja es el sistema de captura de mosquitos. La posición de este se encuentra en la parte frontal de la caja, ver Figura. 2.3, y este tiene una pequeña salida hacía el exterior. Una forma de atraer a los mosquitos es utilizar luz ultravioleta [5], lo que permite utilizarla para aumentar la probabilidad de captura, se diseño una matriz de leds con luz ultravioleta y se colocó cerca de la entrada de captura de mosquitos. En la parte inferior, justo debajo de la matrices de leds se encuentran los sensores de humedad y temperatura, estos sensores son los mismos que se encuentran junto al anemómetro, Figura 2.2, ya que el objetivo de poner dos sensores en dos lugares distintos de la caja es comparar los datos obtenidos respecto a la captura de los mosquitos, en otras palabras, si se no observa alguna diferencia correspondiente a la ubicación de los sensores, para futuras cajas se eliminarían alguno de los sensores de medición debido a que no son necesarios instalarlos en las próximas cajas a construir.

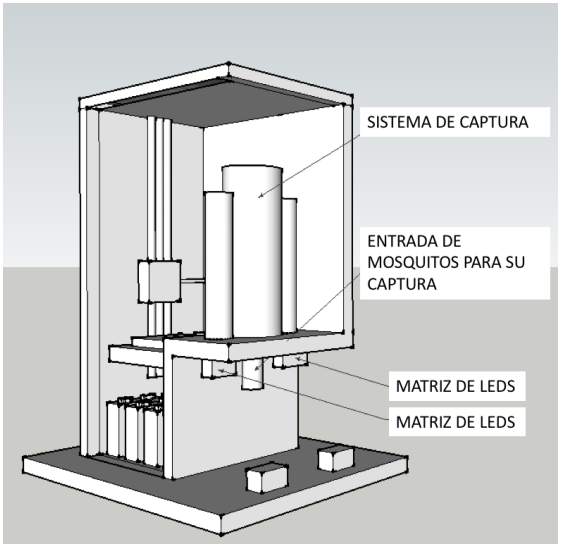


Figura 2.3: Vista externa, sistema captura de mosquitos.

Hasta ahora solo se ha hablado de como es que se desearía construir la caja contadora de mosquitos ya que solo en las figuras anteriores, se muestran dibujos esquemáticos de los objetos y componentes de la caja contadora de mosquitos. En la sección 2.2 se describe como es que se estructuró cada sensor y en la sección 2.3 mostramos los resultados finales.

2.2. Manejo y uso de sensores

Se desean construir circuitos para utilizarlos dentro de un entorno científico, en este caso se utilizarán para la medición de las variables climatológicas y un conteo de mosquitos ya que el objetivo es encontrar la correlación entre la población y las condiciones. Los componentes electrónicos principales a utilizar son las tarjetas programables y sensores de medición, además una de las ventajas es que son sencillos de utilizar ya que cualquier persona es capaz de utilizarlos aún no sabiendo nada de electrónica.

2.2.1. Tarjetas programables

Actualmente, existen demasiados componentes electrónicos. Uno de ellos es la tarjeta programable tal como el Arduino, Rasberry Pi, BeagleBone, Udoo; todas esas tarjetas tienen el mismo propósito, la diferencia entre ellas varía en sus distintos componentes tales como la memora RAM (Random-Access Memory), almacenamiento de información, cantidad de procesamiento de programas y velocidad de procesamiento. Otra de las diferencias notables es que ocupan tan poco espacio que son muy cómodas de usar. Con estas se pueden hacer una gran cantidad de proyectos, por ejemplo un dispositivo que sirva para la captura de los mosquitos.

Las tarjetas que se van a utilizar son el Arduino y la Rasberry Pi. Se eligieron estas ya que son las mas populares del mundo, además existen muchos tutoriales, libros, cursos de robótica y más.

El Arduino básicamente es una tarjeta madre micro-controladora. Un micro-controlador es una simple computadora que puede correr un programa a la vez y este lo corre una y otra vez. Esto lo hace muy simple de utilizar. Una Rasberry Pi es una computadora con un propósito general, es decir, cuenta con un sistema operativo llamado Raspbian bajo la distribución Linux y con la habilidad de correr múltiples programas. Esto lo hace mas complicado de utilizar que Arduino. Ambas tienen algunas diferencias entre si, respecto a como programarlas, es decir, cada una de ellas utiliza su propia interfaz, su propio lenguaje de programación. Aunque la tarjeta Rasberry Pi puede utilizar cualquier lenguaje gracias al sistema operativo Linux pero siempre y cuando ese lenguaje a utilizar tenga las correspondientes librerías. Como podemos notar ambas tarjetas son parcialmente diferentes.

Hasta ahora tenemos una idea mas clara de las tarjetas programables, pero uno de los principales motivos por el cual se eligieron estas tarjetas es que se

pueden fusionar. Ya que, Rasberry Pi lanzó una librería que se puede comunicar con Arduino. Así nació RaspDuino, lo cual es fabuloso e increíblemente útil, ya que se puede dejar que Arduino haga las tareas sencillas como hacer mediciones de sensores y posteriormente ser enviados y por otro lado dejar que la Rasberry Pi se encargue de los cálculos intensos.

Para construir la caja contadora de mosquitos necesitamos algo que se encargue de hacer mediciones. Arduino será el encargado de medir las variables climatológicas como; temperatura, humedad, precipitación, velocidad y dirección del viento.

Una vez que hemos hablado de los diseños y de las tarjetas programables a utilizar, es momento de configurar los sensores correspondientes que medirán dichas variables climatológicas.

2.2.2. Sensor de temperatura y humedad

En el presente, existen sensores que son capaces de medir dos cosas a la vez. En este caso existe un sensor conocido como DHT-22 que es un dispositivo capaz de medir la humedad y temperatura. Internamente, este sensor tiene dos componentes; un sensor de humedad capacitivo [5] y un termistor [6].

Un sensor de humedad capacitivo contiene un dieléctrico en el cual, el aire cambia su permitividad eléctrica con respecto a la humedad del ambiente, cabe decir que, el aire penetra en el campo eléctrico que hay entre las placas del sensor consiguiendo una variación en el dieléctrico. Mientras que un termistor es un elemento de detección de temperatura compuesto por material semiconductor sinterizado que presenta un gran cambio en la resistencia en proporción a un cambio pequeño en la temperatura. En general, los termistores tienen coeficientes de temperatura negativos, lo que significa que la resistencia del termistor disminuye a medida que aumenta la temperatura. La forma en la que el Arduino interpreta su funcionamiento es a través de señales digitales calibradas [6].

Para conectar el sensor DHT-22 al Arduino se hace la siguiente configuración. En la Figura 2.4 vemos a los dos dispositivos unido por ciertas conexiones. El sensor tiene 4 pines; el pin 1 requiere una fuente de alimentación de 3 a 5V que esta puede ser proporcionada por el mismo Arduino, el pin 2 es la salida de datos que es conectado con el puerto 2 del Arduino y para proteger ambos dispositivos se coloca una resistencia de 10kΩ; el pin 3 es nulo; el pin 4 es la tierra del sensor y este se conecta a la tierra del Arduino. Además este sensor tiene una velocidad de actualización bastante lenta, entonces solo se podrá enviar información a la Rasberry Pi cada 2 segundos, mientras este activo.

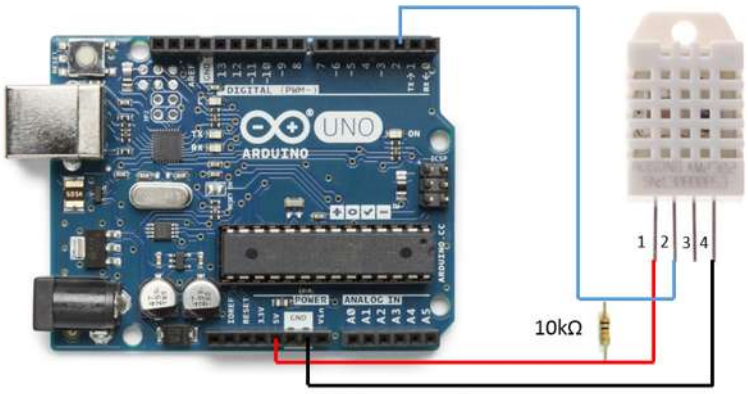


Figura 2.4: Conexiones guiadas por color donde el rojo indica la alimentación del sensor, el color negro la tierra y el color azul indica la lectura de datos, estas uniones indician la conexión entre el Arduino y el sensor DHT-22.

2.2.3. Sensor de precipitación

La precipitación es cualquier forma de hidrometeoro que cae de la atmósfera y llega a la superficie terrestre. Este fenómeno incluye lluvia, llovizna, nieve, aguanieve, granizo, neblina. Sin embargo, de todos los fenómenos mencionados solo se desea saber si llueve o no. Ya que se pretende ver cuales son los efectos relacionados con la abundancia de los mosquitos.

Entre los múltiples módulos que se pueden encontrar en el mercado, aparece el **YL-83**, que es capaz de detectar gotas de agua, o sea lluvia. Este módulo se instalará en el caja contadora de mosquitos ya que puede ser utilizado para tomar medidas preventivas respecto al funcionamiento de la caja.

Este módulo consiste en una serie de pistas conductoras impresas sobre una placa de baquelita. La separación entre las pistas es muy pequeña. Lo que este módulo hace es crear un corto circuito cada vez que las pistas se mojan. El agua hace que se cree un camino de baja resistencia entre las pistas con polaridad positiva y las pistas conectadas a tierra (GND). La corriente que fluye a través de estas pistas se ve limitada por resistencias de 10KΩ en cada conductor, lo que impide que el corto circuito que se genera cuando se moja la placa llegue a estropear el micro controlador.

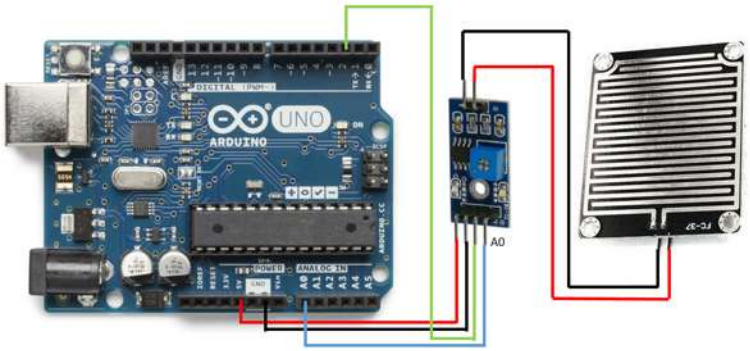


Figura 2.5: Conexiones guiadas por colores, donde el rojo indica alimentación, el negro tierra, el verde la lectura de datos de manera digital y el azul la lectura de datos de manera análoga y estas uniones indican la conexión entre el Arduino, el sensor FC-37 ubicado en el centro de la imagen y el módulo YL-83 que se encuentra en el lado derecho de la imagen.

La detección de lluvia consta de dos partes, el sensor **YL-83** mencionado anteriormente y el sensor **FC-37**, Figura 2.5. Este último sensor, cuenta con un circuito que posee las resistencias limitadoras de corriente y es el encargado de alimentar el módulo YL-83. Posee un amplificador operacional, específicamente el circuito integrado LM392. Este es el encargado de amplificar el pequeño diferencial de voltaje que se general cuando una gota de agua cae sobre las pistas del módulo. Aquí es donde se genera la señal de salida que puede ser del tipo analógica o digital. La señal digital oscilará entre los valores HIGH y LOW dependiendo de si hay agua o no sobre las pistas de la placa YL-83. La salida analógica entregará un nivel de voltaje que variará dependiendo de la cantidad de agua que haya sobre el módulo.

Para utilizar el módulo basta con conectar el circuito de control a los pines que se indican en la Figura 2.5.

2.2.4. Sensor para medir la velocidad de viento

En el tiempo actual, se conocen distintas formas para medir la velocidad del viento ya que existen varios dispositivos. A estos dispositivos se les conocen como anemómetros. Hay desde analógicos, digitales o ultrasónicos; todos ellos varían por su complejidad tanto en construcción como en tecnología.

Para fines prácticos se construyó nuestro propio dispositivo para medir la velocidad del viento. Este sensor consta de solo tres objetos principales estos son; un cilindro de 3/4" de pulgada, tres aspas circulares y un motor de corriente directa (DC).

La forma en la que se construyó es muy simple. Primero se fabricó una base cilíndrica de madera. En las orillas de la base se colocaron 3 aspas y al centro de la base se introdujo el eje central de motor. Después se soldó un cable color negro a la parte negativa del motor y análogamente el cable color rojo a la parte positiva. Por último, el motor se introdujo a la base circular del cilindro de 3/4" de pulgada y se fijo el motor con un pegamento bastante adhesivo conocido como plasti-acero.

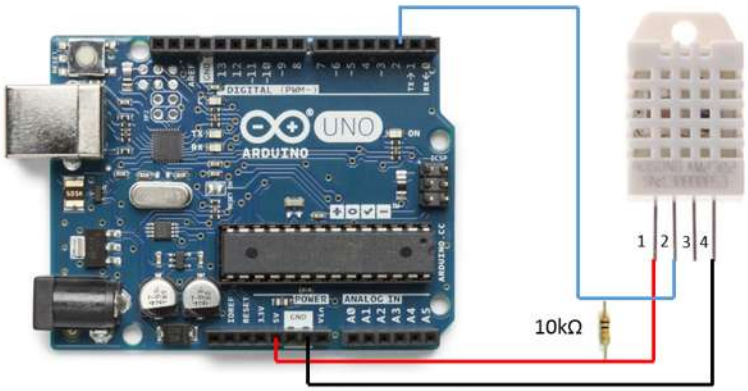


Figura 2.6: Conexiones guidas por colores que indician la conexión entre el Arduino y el sensor DHT-22.

Se tiene un motor de corriente directa (DC) y cuando se gira el eje del motor, se puede observar con la ayuda de un multímetro que este genera valores conocido como voltaje inverso. A esto se le conoce como un generador eléctrico, que transforma energía mecánica de rotación en energía eléctrica y se le puede llamar una máquina generatriz de fem, correctamente llamado alternador.

Cuando el anemómetro esta funcionando genera valores. Sin embargo, no se sabe que significan esos datos por que no es posible saber cual es la velocidad de viento. Para calibrar el anemómetro, este se pegó a la parte del techo de un carro lo cual no es una buena idea ya que por efecto de Bernoulli [10] el aire en el techo va más rápido del que no golpea a el. Si se deseara evitar esto, lo que se tendría que hacer es sacar el anemómetro lo suficientemente lejos del coche, pero ponerlo en el techo es mas fácil y se cree que el error no sería tan grande debido a que no se conduce a altas velocidades.

El siguiente paso es medir con un multímetro el voltaje en la escala de mili-volts e ir avanzando en distintas velocidades fijas con el coche hasta lograr estabilizar las medidas. Muy importante es que ese día no haya viento pero como esto es casi imposible,entonces se decidió hacer varias medidas. Así se que se generó una tabla donde se registró la tensión marcada por el multímetro, ver tabla 2.1. Los datos registrados partieron dentro del intervalo (20 - 90) km/h aumentando de 10 en 10 km/h. Este proceso se repitió cinco veces.

Tabla 2.1: Anemómetro, medidas resultantes del motor eléctrico para cinco medidas .

Velocidad	Medida 1	Medida 2	Medida 3	medida 4	Medida 5
Km/h	**mV**	**mV**	**mV**	**mV**	**mV**
20	82	86	85	82	81
30	125	124	129	124	127
40	190	186	194	197	196
50	250	244	246	250	256
60	320	320	319	320	315
70	390	383	397	390	393
80	430	434	424	433	427
90	480	473	478	486	482

Como no es posible saber la medida exacta en mV correspondiente a Km/h respectivamente, debido al efecto Bernoulli, al no mantener la velocidad de manera constante del vehículo y a los cambios repentinos del viento tanto en dirección como en velocidad. Entonces, lo que se hace es calcular la *media* de todos los procesos y así se considera la media como *valor real*. La media se calcula como

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}. \tag{2.1}$$

La ecuación (2.1), ayuda a calcular el valor que se considerara como valor real, esto debido a las cinco mediciones hechas en la tabla 2.1. Así mismo, al realizar una o varias medidas, es posible calcular el error absoluto de la forma

$$E_a = |\bar{x} - x|, \tag{2.2}$$

por lo que entre más medidas se hagan, esto implica que el error absoluto disminuye. El error absoluto es un indicador de la imprecisión que tiene una de-

terminada media. En la tabla 2.2 se puede visualizar la media de cada velocidad correspondiente y una medida extra, en el que se calcula dicho error.

Tabla 2.2: Anemómetro, medidas resultantes del motor eléctrico.

Velocidad	Media	Medida 6	Error absoluto
Km/h	**mV**	**mV**	**mV**
20	83.2	81	2.2
30	125.8	122	3.8
40	192.6	190	2.6
50	249.2	246	3.2
60	318.8	316	2.8
70	390.6	397	6.4
80	429.6	429	0.6
90	479.8	483	3.2

Posterior a esto si se gráfican los valores km/h vs mV se puede observar que que la relación entre ellos es casi lineal. Así para poder describir este comportamiento se puede hacer una regresión lineal para este conjunto de datos. Básicamente la idea consiste en obtener una ecuación de la forma

$$y = mx + b \tag{2.3}$$

que mejor se ajuste a los datos que se tengan. La ecuación 2.3 representa la ecuación de la recta. Para esto se necesita encontrar el valor m que es la pendiente y el valor de b que es el valor que corta en el eje de las y. Entonces para encontrar a m y b se tiene lo siguiente

$$m = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i - n \sum_{i=1}^{n} (x_i y_i)}{\left(\sum_{i=1}^{n} x_i\right)^2 - n \sum_{i=1}^{n} x_i^2} \tag{2.4}$$

$$b = \bar{y} - m\bar{x} \tag{2.5}$$

donde n es el número de datos disponibles y $\bar{x}$ e $\bar{y}$ son el promedio y se calculan utilizando la forma de la ecuación (2.1).

Para determinar que tan bueno es nuestro ajuste calculamos el coeficiente de correlación como

$$R = \left(\frac{\sigma_{xy}}{\sigma_x \sigma_y} \right)^2 \tag{2.6}$$

donde $\sigma_x \sigma_y$ representan las desviaciones típicas y σ_{xy} representa la covarianza, estas se calculan por

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i)^2}{n} - \bar{x}^2}, \tag{2.7}$$

$$\sigma_y = \sqrt{\frac{\sum_{i=1}^{n} (y_i)^2}{n} - \bar{y}^2}, \tag{2.8}$$

$$\sigma_{xy} = \frac{\sum_{i=1}^{n} (x_i y_i)}{n} - \bar{x}\bar{y}. \tag{2.9}$$

Se creó un código en Python para hacer estos cálculos, dando como resultado a $m = 5.5726$, $b = -35.6190$ y $R = 0.9606$. En la Figura 2.7 podemos visualizar los puntos respectivos y la respectiva ecuación de la recta que mejor ajusta al conjunto de puntos.

$$y = 5.5726x - 35.6190 \tag{2.10}$$

Por último basta con conectar el anemómetro al Arduino. Para esto se conecta el polo positivo del motor a una de las entradas análogas del Arduino y el polo negativo del motor a la tierra del Arduino.

Un aspecto importante de esta calibración es que solo funciona para la forma en la que se construyó el anemómetro, es decir, importan demasiado sus componentes y las medidas que dependen tanto de la altura, la base y el modelo del motor que se utilizó ya que si se cambia algunos de estos componentes la ecuación (2.10) de la recta obtenida ya no serviría y se tendría que volver a calibrar.

Entonces para probar el funcionamiento del anemómetro se realizo la misma actividad que cuando se calibró, por lo que se colocó en el techo del coche nuevamente y se viajó a distintas velocidades hasta comprobar su funcionamiento.

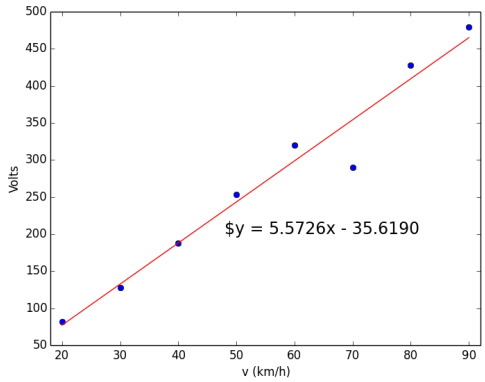


Figura 2.7: Gráfica de los puntos registrados para la calibración y su respectiva regresión lineal.

2.2.5. Sensor para dirección del viento

Si se tuviera un anemómetro resultaría mas fácil medir la dirección del viento debido a su tecnología. Pero es posible construir un dispositivo que indique hacía donde se dirige.

Para construir el dispositivo que nos indicara la dirección del viento. Se utilizaran 8 reed switch, dos baleros, un tubo cilíndrico de 3/4"de pulgada y un guiador.

Un reed switch es un sensor usado para detectar el campo magnético. Estos en español son conocidos como interruptores de láminas.

El reed switch es un tipo de componente de interruptor de línea que realiza el control mediante señales magnéticas. Estos están construidos comúnmente por un tubo de cristal sellado que contiene dos láminas elásticas. Por lo general, las láminas se encuentran separadas, además están hechas de materiales especiales. Sin embargo, cuando una sustancia magnética se aproxima al tubo de cristal, las dos laminas se magnetizan lo que implica que estas se atraen entre sí. Como resultado, al estar juntas las laminas se conectan para dejar pasar el flujo de corriente. Después de que la fuerza magnética desaparezca, las laminas se separan y por lo tanto ya no haya flujo de corriente.

Los baleros son rodamientos que se utilizan en muchos lugares. Internamente contiene balines circulares y estos están en medio de dos superficies circulares una interna y otra externa. Entonces es posible fijar el cilindro interno del balero para que el externo se mueva y de manera análoga si se fija el cilindro externo el interno gira. Por esto es que se decidió usar un balero ya que su función principal en sí

es que se puede utilizar como un tornillo de giro infinito tanto para la izquierda como para la derecha. Como no se sabe hacía donde se dirige el viento y si este cambia de dirección constantemente, el balero resuelve este problema.

Para fabricar este dispositivo primero se tomaron dos baleros y estos se insertan en los extremos dentro del cilindro de 3/4"de pulgada quedando así el cilindro externo del balero fijo. Como el cilindro interno del balero quedó para hacer un giro infinito allí es donde se inserta un guiador. En una de las puntas del guiador tiene un imán y del otro extremo tiene un simple cuadro metálico que este será empujado por el viento. Tanto el imán como el cuadro metálico están alineados entre sí. Los reed switches son posicionados en los cuatro puntos cardinales.

La forma en que se configura este dispositivo es la siguiente: Se nombra a cada reed switch con N de Norte, NE de Noreste, E de Este, SE de Sureste, S de sur, SO de Suroeste, O de Oeste y NO de Noroeste. Entonces cuando el imán se acerque a cualquier de los puntos cardinales la computadora registrara la ubicación en la que se propagó.

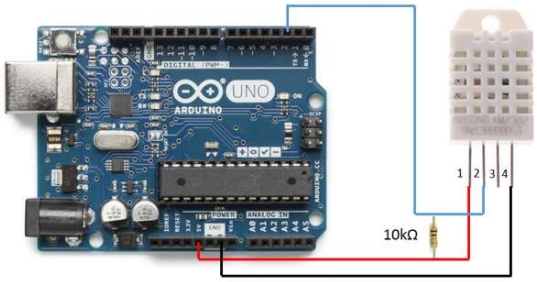


Figura 2.8: Conexiones guidas por colores que indician la conexión entre el Arduino y el sensor DHT-22.

2.2.6. Construcción de un sistema de captura

Existen varias formas por las que un mosquito se encuentra rondando cerca de los seres vivos y por lo general los mosquitos hembra tienden a picar para succionar su sangre con el objetivo de reproducirse. Una de estas formas se debe a la respiración, es decir, cada vez que los seres vivos inhalan oxigeno y posterior a ello exhalan CO_2. Otra forma es por la temperatura que emiten los seres vivos, los mosquitos tienen visión infrarroja.

Generalmente, la forma de atraer a los mosquitos es utilizando gas CO_2 y llevarlos directo a una trampa. Se tienen pensado dos sistemas de captura; atracción

por luz ultravioleta y el uso de dicho gas. Aunque solo se pretende usar el método de atracción por la matriz de leds.

En la Figura 2.3 se puede ver la ubicación de la matriz leds. Con el uso del Arduino y los leds se puede configurar una serie de encendido y apagado de leds. La forma en la que se conectaron los leds fue de forma paralela y para cada linea paralela tiene conectado 3 leds adicionales en serie. La forma del encendido de leds es similar al comportamiento de un vúmetro de luces, ya que este enciende luces de acuerdo a las señales recibidas [8].

Ya se ha hablado de la atracción de los mosquitos, ahora hablaremos de la construcción de un tipo aspiradora, cada pie cúbico de aire al nivel del mar pesa un poco más de 30 gramos. Primero hay que recordar que se mencionó *aire al nivel del mar*, pues a medida de que se asciende el aire es menos denso. El aire es compresible y el gran peso del aire superior oprime el aire al nivel del mar en un volumen relativamente pequeño. A una altura de 7620 metros, con mucho menos aire oprimiendo desde arriba, un cuarto de metro cúbico de aire pesa menos de medio kilogramo. Entonces, una columna de aire de 1 x 1 pulgadas cuadradas y tan alta como el cielo (aproximadamente 100km) pesa menos de 7 kilogramos, lo cual se explica por que se habla de presión atmosférica al nivel del mar.

Esta presión actúa sobre nosotros exactamente como la presión del agua que actúa sobre un buzo de aguas profundas. Cuando se ensancha la cavidad del pecho, el aire entra para llenar el espacio. Para sacar el aire se contrae el pecho, creando una presión interna un poco más alta que la presión del aire exterior, y así se saca el aire atrapado. Cualquier presión más baja que la presión de aire atmosférico es llamada vacío. Sin embargo, existen dos tipos de vacío; suave y alto, es decir, vacío suave es aquel que se refiere a presiones un poco debajo de la presión de aire atmosférico y vacío alto se refiere a la presión cercana al cero absoluto.

Incidentalmente, lo que a veces se pasa desapercibido es que la presión no es el estado normal o promedio del universo, el vacío sí lo es. Existe más vacío en el espacio interplanetario e interestelar que presión en la tierra. Por consiguiente, el vacío o casi completo vacío, es más natural que un espacio lleno con tierra, ladrillo, aire o gente.

Sin embargo, el vacío es raro en la tierra y puede por consiguiente utilizarse en muchas formas interesantes. Una de éstas, la aspiradora, que absorbe el aire atmosférico por un tubo que transporta el polvo y la suciedad, junto con él. El aire es aspirado y llevado a través de una tela porosa, aquí las partículas de polvo o suciedad son muy grandes para atravesar los poros de la tela.

Para construir una aspiradora se necesita de un motor como parte fundamental y un tubo, como ya se ha mencionado. Entonces el objetivo del motor es que no absorbe si no mas bien expulsa aire de la aspiradora y crea un el vacío en interior del aparato. Ese vacío es lo que produce el efecto de succión, físicamente el vacío se crea con el aire que hay alrededor que en este caso es el aire que esta afuera de la aspiradora entonces cuanto mas rápido expulse el aire dentro de la estructura más rápido volverá a entrar y a llenar ese vacío. Pero la potencia de la bomba de aire no es lo único que afecta la eficacia de una aspiradora para absorber el polvo, el tamaño de la entrada de aire también marca la diferencia, es decir, cuanto mas pequeña sea la abertura mas fuerte será la succión. Esto último se debe al efecto Bernoulli, en la Figura 2.9 se muestra el comportamiento del aire a través de un tubo que cuenta con diferentes áreas.

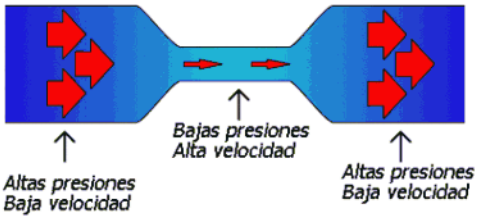


Figura 2.9: Efecto Bernoulli debido al flujo de aire a través de un tubo con reducción de área.

2.3. Construcción final

Lo único que falta es unir todos los componentes descritos anteriormente

Es de suma importancia tener a disposición un dispositivo que sea capaz de hacer mediciones respecto a las variables del clima y de la cantidad de mosquitos, ya que es una forma de obtener datos para los cuales serán analizados posteriormente con el uso de la inteligencia artificial. En el siguiente capitulo se describe a detalle los recursos necesarios para poder analizar los dichos datos capturados por los dispositivos electrónicos. Ya que con esta herramienta nos permitirá poder hacer predicciones de fuentes de mosquitos.

Capítulo 3

Aprendizaje Computacional

La capacidad de aprender se considera uno de los atributos distintivos del ser humano y de los animales, y ha sido una de las principales áreas de investigación de la Inteligencia Artificial desde sus inicios.

En los últimos años se ha visto un crecimiento acelerado en la capacidad de generación y almacenamiento de información, debido a la creciente automatización de procesos y a los avances desarrollado en cualquier área que requiera del uso de una computadora. Esto ha llevado a desarrollar una gran cantidad de herramientas y técnicas que tienen que ver con el análisis de información. En este aspecto, el desarrollo en el área de aprendizaje ha sido fundamental. El área de aprendizaje en general trata de construir programas que mejoren su desempeño automáticamente con la experiencia.

Desde el comienzo de las computadoras se cuestionó si serían capaces de aprender. El darles la capacidad de aprendizaje a las máquinas abre una amplia gama de nuevas aplicaciones. El entender como pueden aprender las máquinas nos puede ayudar a entender las capacidades y limitaciones humanas de aprendizaje, siempre y cuando se desarrollen mas mecanismos en función al comportamiento humano, como las prótesis robóticas [7] que pueden ser controlados por los humanos.

El aprendizaje humano en general es muy diverso e incluye, la adquisición de conocimiento, desarrollo de habilidades a través de instrucción y práctica, organización de conocimiento, descubrimiento de hechos, entre otros.

De la misma forma el aprendizaje computacional o aprendizaje maquina (conocido en inglés como Machine Learnig) se encarga de estudiar y modelar computacionalmente los procesos de aprendizaje en sus diversas manifestaciones.

En general, se busca construir programas que mejoren automáticamente con la experiencia, es decir, se registra cada proceso ocurrido al que se les dan ciertas prioridades para que así los procesos sean mejorados. Aunque no se tienen programas que aprendan tan bien como los humanos, actualmente existen algoritmos que han probado ser muy efectivos para ciertas tareas tales como las industrias de coches, la bolsa de valores y la física computacional, entre otros.

Actualmente existe una gran cantidad de métodos de aprendizaje tales como: árboles de decisión con el uso de reglas de clasificación, métodos de clasificación, redes y regresiones no-lineales, modelos gráficos de dependencias probabilísticas, redes neuronales, algoritmos genéticos y muchos otros más.

Algunos de los ejemplos que usan los algoritmos de aprendizaje son sistemas de reconocimiento de voz (utilizado por los bancos), manejo de vehículos autónomos (utilizado por los carros Telsa). Por otro lado, existe una gran cantidad de aplicaciones reales relacionadas con descubrimiento de conocimiento en base de datos que utilizan algún algoritmo de aprendizaje. Estas aplicaciones se encuentran en áreas tales como

- Astronomía: clustering y clasificación de cuerpos celestes.
- Biología molecular: predicción de substancias cancerígenas, genoma humano.
- Aspectos climatológicos: predicción de tormentas.
- Medicina: caracterización y predicción de enfermedades.
- Física: análisis de ondas gravitacionales.
- Mecatrónica: aprendizaje de tareas en robótica.

En este trabajo se estudia el algoritmo de aprendizaje perteneciente al tipo de Métodos de clasificación y regresiones no-lineales. Este método es conocido como **Redes Neuronales Artificiales** (RNAs) e implementa un algoritmo de entrenamiento llamado **retropropagación** (en inglés es conocido como **Backpropagation algorithm**).

Aunque muchos tipos de algoritmos han sido propuestos en los últimos años, el algoritmo de retropropagación es el más destacado en cuestión de clasificación. En las siguientes secciones se describirá a detalle el funcionamiento de dicho método y al final se darán algunos ejemplos con la finalidad de comprender el funcionamiento de las Redes Neuronales Artificiales.

3.1. Redes Neuronales

Las RNAs [6] son un paradigma computacional, cuya estructura busca emular el aprendizaje como ocurre en las redes neuronales biológicas. El cerebro humano corresponde a un sistema altamente complejo, no lineal y paralelo. En términos sencillos lo anterior equivale a decir que puede realizar operaciones simultáneamente a diferencia de las computadoras comunes que son de tipo secuencial, es decir, realizan una operación a la vez. De esta manera las redes neuronales son capaces de realizar tareas como reconocimiento de patrones, regulación de energías eólicas y solares, además de aplicaciones en el área de la domótica[1] y recientemente han sorprendido en el ámbito de las artes visuales con imágenes generadas a partir de las redes neuronales de la compañía Google.

Las investigaciones realizadas en torno al estudio de RNAs, están motivadas en modelar la información del procesamiento de la información en sistemas nerviosos biológicos. Especialmente por la forma del funcionamiento del cerebro humano.

3.1.1. Forma de aprendizaje

Las primeras personas en introducir el concepto de redes neuronales fueron Warren McCulloch y Walter Pitts (1943) [15]. Ellos crearon un modelo informático para redes neuronales basado en las matemáticas y algoritmos denominados como lógica de umbral. Este modelo señaló el camino para que la investigación de redes neuronales se dividiera en dos enfoques distintos. El primero, centrado en los procesos biológicos en el cerebro y el otro en la aplicación de redes neuronales para la inteligencia artificial.

Antes de que McCulloch y Pitts introdujeran este concepto, en el año 1940 el psicólogo Donald Hebb creo una hipótesis de aprendizaje basado en el mecanismo de plasticidad neuronal que ahora se conoce como aprendizaje de Hebb [8]. Este tipo de aprendizaje se empezó a aplicar en los modelos computacionales con el uso de la maquina de Turing y posteriormente, las primeras maquinas computacionales en ser utilizadas fueron las calculadoras que simulaban una red o aprendizaje de Hebb.

[1]Se llama *domótica* a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

3.1.2. Redes neuronales como un modelo del cerebro humano

Desde que se empezó a estudiar la anatomía y estructura del tejido nervioso, los investigadores trataron de conocer la forma en cómo este tejido y los órganos que constituye, especialmente el cerebro que procesa la información que reciben de los órganos receptores, para dar una respuesta adecuada a sus estímulos.

Aunque aún se está lejos de comprender completamente el funcionamiento y la estructura del sistema nervioso, se conoce con detalle la estructura de la neurona, como elemento básico del tejido nervioso y la forma cómo se estructura la corteza cerebral.

La neurona, como toda célula, consta de una membrana exterior conocida como cuerpo o soma, que la limita y le sirve de órgano de intercambio con el medio exterior. Éste medio consta del citoplasma y el núcleo, el primero es el cuerpo principal de la célula donde radica el grueso de sus funciones y el segundo contiene el material genético de la célula, ver la figura 3.1.

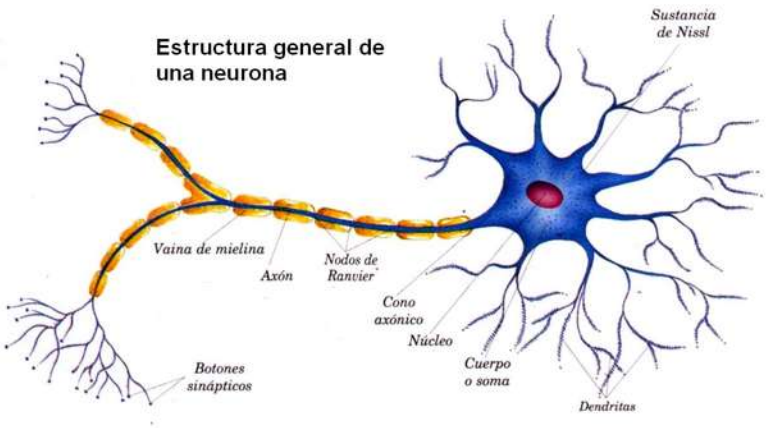


Figura 3.1: Esquema de una neurona biológica.

El citoplasma presenta unos alargamientos llamados dendritas, que son órganos de recepción. Ellos finalizan con un gran número de fibras que son conductores y éstas llevan la señal o impulso nervioso de los receptores hacía otras neuronas o incluso hacia la misma neurona. Estas fibras terminan en un pequeño corpúsculo llamado sinapsis, que constituye un relevador bioquímico y que sirve para transferir la señal de una neurona a otra.

Existen dos clases de sinapsis: actuadoras e inhibidoras. Las actuadoras son

aquellas que favorecen el disparo de la neurona receptora y las inhibidoras dificultan este proceso.

Cuando se presenta un cierto desbalance entre las sinapsis actuadoras y las inhibidoras activas, la neurona dispara un impulso de salida, que constituye la respuesta de la neurona. Este impulso nervioso de salida es conducido por una prolongación cilíndrica alargada (hasta de varios decímetros de largo) de la neurona, que se llama cilindro eje o **axón**, que en su extremo se divide en varias fibras para comunicarse con otras neuronas o con órganos efectores o motores como glándulas o músculos.

El citoplasma de las neuronas forma la masa gris de los centros nerviosos y el conjunto de cilindros ejes forma la masa blanca de aquéllos.

3.1.3. Topología de una red neuronal

Las redes neuronales artificiales tienen la forma sobre-simplificada de las neuronas biológicas aunque las artificiales son mas sencillas, es decir, no posee tantos elementos a los ya mencionados.

En la parte superior de la figura 3.2 se muestra una neurona biológica y en la inferior una artificial. Se sabe que en las neuronas biológicas la entrada de información es recibida a través de las dendritas, después es enviada al núcleo y recibe una estimulación para que posteriormente pase a través del cuello del axón y finalmente obtener un resultado. De manera similar una neurona artificial, recibe entradas que son representadas como $x_1, x_2, ..., x_n$ y que son procesadas dentro de un cuerpo y por último se obtiene una o varias salida.

En las neuronas biológicas, una vez que se procesó información de una neurona, ésta pasa por otra neurona y después por otra y así hasta obtener un resultado deseado, por lo que al final se forma una red neuronal biológica. La misma idea se aplica a la red artificial donde, si se hace un zoom en la parte del cuerpo lo que esta pasando son tres cosas

1. Cuando entra una señal, se multiplica por un valor de peso asignado a esta entrada en particular. Es decir, si una neurona tiene tres entradas, entonces tiene tres pesos que se pueden ajustar individualmente.

2. Las señales de entrada modificadas se suman a un solo valor.

3. Finalmente, el resultado del cálculo de la neurona se convierte en una señal

de salida. Esto se hace alimentando el resultado a una función de activación (también llamada función de transferencia).

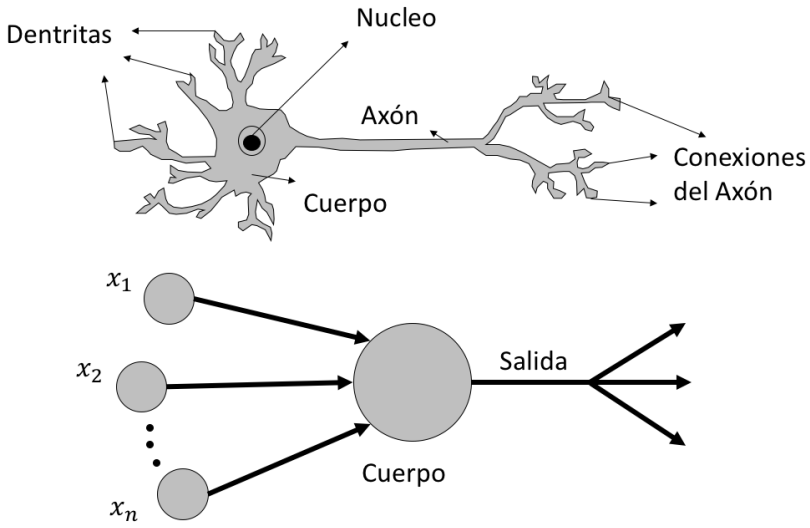


Figura 3.2: Esquema de una neurona artificial (abajo) comparada con una biológica (arriba).

En la figura 3.3 se puede ver una red de neuronas artificiales formada por dos neuronas de entradas, una sola capa oculta[2] y una neurona de salida. Cada neurona artificial esta conectada por dos entradas y después por una salida.

Lo anterior, no significa que las redes neuronales artificiales estén restringidas, es decir, la red neuronal puede tener n entradas, k capas ocultas con l neuronas y m salidas.

En la figura 3.3 se observa que las entradas, las neuronas de la capa oculta y las salidas están representadas con círculos, además están dirigidas por una flecha que representa los pesos w de la red neuronal.

3.1.4. El Perceptrón

Frank Rosenblat (1958) creó el perceptrón que es la red neuronal más simple posible; un modelo computacional de una sola neurona. Un perceptrón se compone

[2]Las capas ocultas se refieren al conjunto de neuronas artificiales que se tenga, en este caso la figura 3.3 se observa que se tiene una sola capa oculta con tres neuronas artificiales

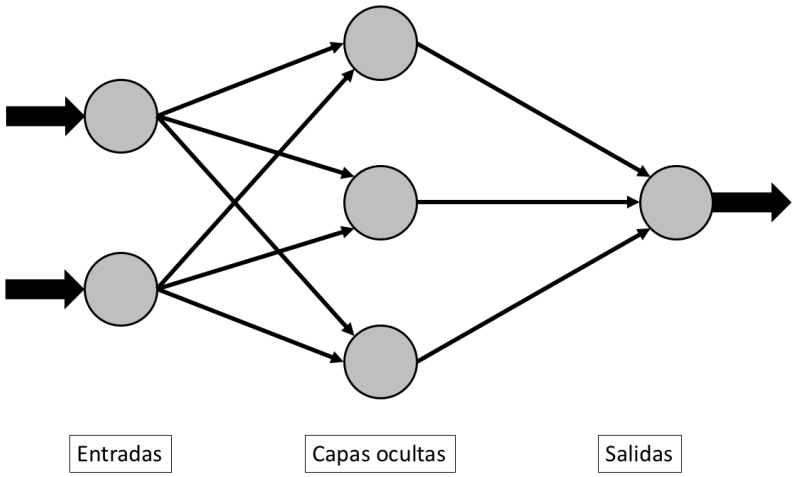


Figura 3.3: Esquema de una red neuronal artificial.

de una o más entradas, un procesador y una única salida. Además es el primer algoritmo de aprendizaje que se inventó. Ver figura 3.4.

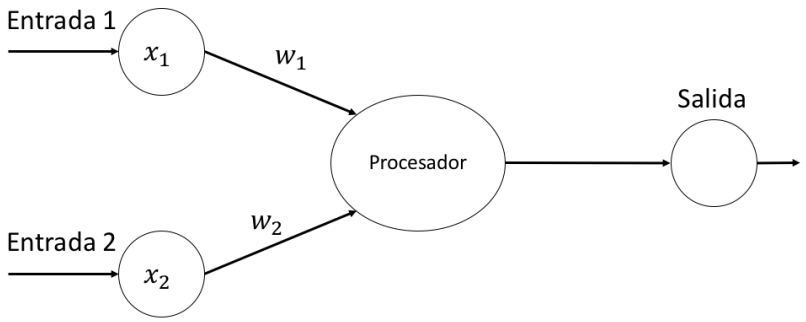


Figura 3.4: Esquema de un perceptrón simple, con dos entradas y dos pesos.

Un perceptrón sigue un modelo conocido como **propagación hacía adelante** (en inglés es llamado como **feed-forward**) donde la red es recorrida desde las entradas que son guiadas por pesos conectados a las capas ocultas y para finalizar en la salidas.

El perceptrón simple puede ser procesado para dos entradas como en la figura 3.4. Las entradas son llamadas x_1 y x_2, esto sería el primer paso en el que se recibe las entradas, entonces veamos un ejemplo con números. Sean las entradas:

$$\text{Entrada } 1 : x_1 = 12$$
$$\text{Entrada } 2 : x_2 = 4$$

El paso número dos es hacer que cada entrada que se envía a la neurona primero debe multiplicarse por algún valor (a menudo un número entre -1 y 1). Al crear un perceptrón, generalmente se comienza asignándole pesos aleatorios. Aquí, asignemos valores a los siguientes pesos

$$\text{Peso } 1 : w_1 = 0.5$$
$$\text{Peso } 2 : w_2 = -1$$

Tomamos cada entrada y la multiplicamos por su peso

$$x_1 * w_1 \Rightarrow 12 * 0.5 = 6$$
$$x_2 * w_2 \Rightarrow 4 * -1 = -4$$

Después generamos la suma como en el paso tres

$$\sum_{i=1}^{2} x_i w_i = 6 + (-4) = 2$$

La salida de un perceptrón se genera pasando esa suma a través de una función de activación $f(x)$. La forma más básica de una función de activación es una función binaria simple que toma sólo dos resultados posibles

$$f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \tag{3.1}$$

Evaluando el resultado anterior en esta función, se tiene como salida 1.

$$f(\sum_{x=1}^{2} x_i w_i) = f(2) = 1 \tag{3.2}$$

Un ejemplo para el que se puede usar una salida binaria simple es al momento de tomar una decisión para encender un LED o no, es decir, el LED conectado en un circuito puede recibir la señal emitida: si el perceptrón devuelve un valor positivo como salida, significa que el LED se enciende; si no, permanece apagado. Otro ejemplo sería avanzar o frenar un coche electrónico, entre otros más. Entonces se puede concluir que la función de activación le dice al algoritmo que camino debe tomar.

En `http://www.ifm.umich.mx/~fiscom/inteligencia_artificial/tesis`, se encuentra el código *perceptron_simple.py* descrito anteriormente.

Hasta ahora solo se ha hablado como propagar hacia adelante ya que el perceptrón no fue desarrollado para entrenar los pesos, además de que no podía ser aplicado para más capas ocultas, por ende unos años mas tarde se desarrolló un algoritmo que resuelve este problema de entrenamiento y es conocido como **Backpropagation** o **retropropagación** que fue creado por Paul Werbos (1975) [1]. Este fue un avance clave ya que el algoritmo de retropragación resuelve eficazmente el problema de redes neuronales que son de múltiples capas.

En la siguiente sección se explica a fondo tal algoritmo, ya que para obtener lo que se quiere (aprender un conjunto de patrones) se requiere los ajustar los pesos.

3.2. Redes Neuronales Artificiales

El algoritmo de entrenamiento retropropagación, surge el caso especial en el que las redes tienen mas de una capa oculta. En esta sección se explicará y ejemplificará la construcción de una red neuronal utilizando el algoritmo de aprendizaje. Para esto se pretende dar un conjunto de datos para que sean propagados a través de la red neuronal, hacer una comparación de resultados respecto a la clasificación obtenida, posteriormente hacer un respectivo ajuste de pesos y luego repetir con otro conjunto de datos, además se darán formulas explícitas para actualizar los pesos y mostrar cómo pueden ser calculados usando operaciones algebráicas lineales.

3.2.1. Red neuronal generalizada

En la sección 3.1.3 se habló tanto de las redes neuronales biológicas como las redes neuronales artificiales y en particular de un perceptrón. En la figura 3.5, podemos ver la estructura general de una red neuronal artificial.

Dentro del aprendizaje maquina, la retropagación es un algoritmo de aprendizaje supervisado o de clasificación, básicamente se puede ver como una función de activación que decide si un conjunto de datos de entrada pertenece a un clase especifica o no. Una de las cosas interesantes de este algoritmo es que permite un aprendizaje en tiempo real ya que el conjunto de elementos son entrenados un paso a la vez, lo que permite actualizar valores de entrada como sea requerido.

Cuando se combinan múltiples neuronas, cada neurona de salida funciona independientemente de todas las demás por lo tanto, aprender cada producto se puede considerar de forma aislada.

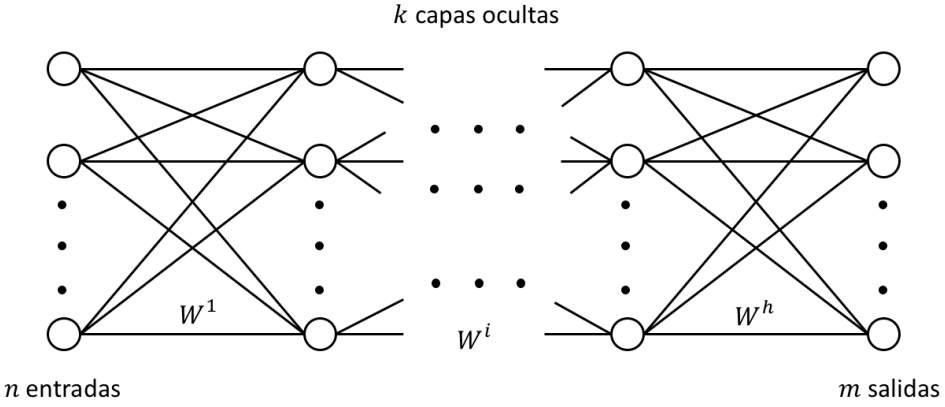


Figura 3.5: Estructura de una red neuronal generalizada, con n entradas, k capas ocultas y m salidas, se muestran también los pesos de las neuronas representados por W^h.

Para simplificar un poco más las cosas y tener un mejor entendimiento de la red neuronal artificial, se tomará $k = 1$, esto es una sola capa oculta como se muestra en la figura 3.6, en al que se tienen l neuronas en la capa oculta, n entradas, m salidas y dos conjunto de datos que corresponde a los pesos, en particular W^1 que se ubica entre las entradas y las neuronas de la capa oculta y W^2 que está entre las neuronas de la capa oculta y las salidas.

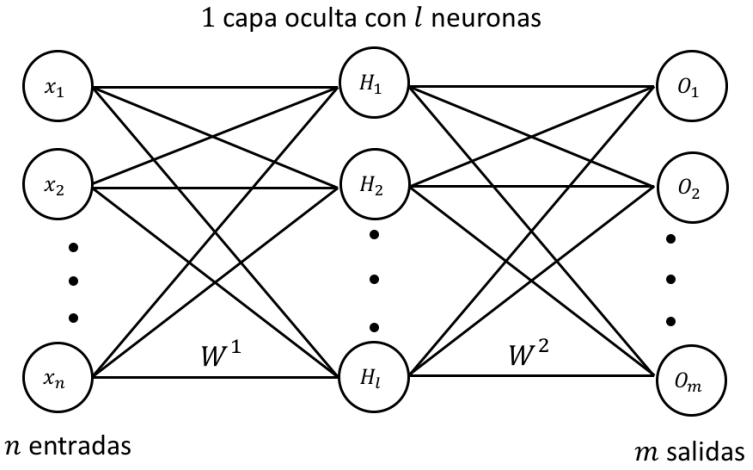


Figura 3.6: Estructura de una red neuronal, con n entradas, $k = 1$ capas ocultas con l neuronas y m salidas, además de sus respectivos pesos W^1 y W^2.

Se tiene un conjunto D de entradas con n elementos para esta red neuronal. Las entradas son datos de información que serán propagados en dicha red. Se puede representar a este conjunto de datos como un arreglo $\vec{x_i}$ para entrada i-ésima e $\vec{y_i}$ como la i-ésima salida, entonces

$$D = x_i, y_i \qquad i = 1, ..., n \tag{3.3}$$

como en el caso del perceptrón, en esta red neuronal los pesos W^1 son números generados aleatoriamente, por lo general entre 1 y -1. Para fines de este trabajo se elegirá una cantidad l neuronas. Para este caso ($k = 1$), a partir de ahora la notación correspondiente a los pesos W^1 se escribirán como w_{nl}.

Si se elige una solo neurona de la capa oculta H_1, se regresaría al caso del perceptrón, lo que significa que las dimensiones de la matriz serían del tamaño w_{n1} y se resolvería la ecuación 3.4 para $l = 1$, lo mismo sería para $l = 2$ con los pesos correspondientes w_{n2} con neurona H_2, al igual que para $l = 3$ con neurona H_3 y así sucesivamente hasta llegar a la neurona H_l. Es claro tener en cuenta que ahora H es un arreglo de datos con l conjunto de elementos.

$$
\begin{aligned}
H_1 &= \sum_{i=1}^{n} x_i w_{i1} \\
H_2 &= \sum_{i=1}^{n} x_i w_{i2} \\
&\vdots \\
H_l &= \sum_{i=1}^{n} x_i w_{il}
\end{aligned} \tag{3.4}
$$

Hasta ahora la ecuación (3.4) calcula el valor de cada neurona l, entonces como en el caso del perceptrón, los valores de cada neurona se evalúan en una función de activación y esta prácticamente le dice al algoritmo que camino tomar.

Para este tipo de red neuronal que se esta analizando existen una gran cantidad de funciones de activación.

Una de las funciones de activación más popular es la función sigmoidea, ésta es una función real que se define como $f_c : \mathbb{R} \to (0, 1)$ con la expresión

$$f_c = \frac{1}{1 + e^{-cx}}, \tag{3.5}$$

con c contante que puede ser seleccionada de manera arbitraria.

En la figura 3.7 se puede ver la función sigmoidea y la manera en la que cambia de acuerdo a como varía el valor constante c. Como se puede observar los valores que se eligieron son para $c = 1$, $c = 2$, $c = 3$. Además un caso particular de la función sigmoidea es cuando el valor c es bastante grande, es decir, cuando $c \to \infty$ la función sigmoidea converge a la función escalón, ver la ecuación (3.1).

A fin de simplificar todos los valores de c, para caracterizar expresiones futuras para $c = 1, 2, 3, ...$ lo que se hace es tomar a $c = 1$. Entonces la notación de la ecuación (3.5) para $f_1(x)$ será $f(x)$.

Otra de la funciones de activación es la función rampa, mostrada en figura 3.8b y esta definida como

$$ramp(x) = \begin{cases} -1 & \text{si } x \leq -1 \\ x & \text{si } -1 < x < 1 \\ 1 & \text{si } 1 \geq x \end{cases} \tag{3.6}$$

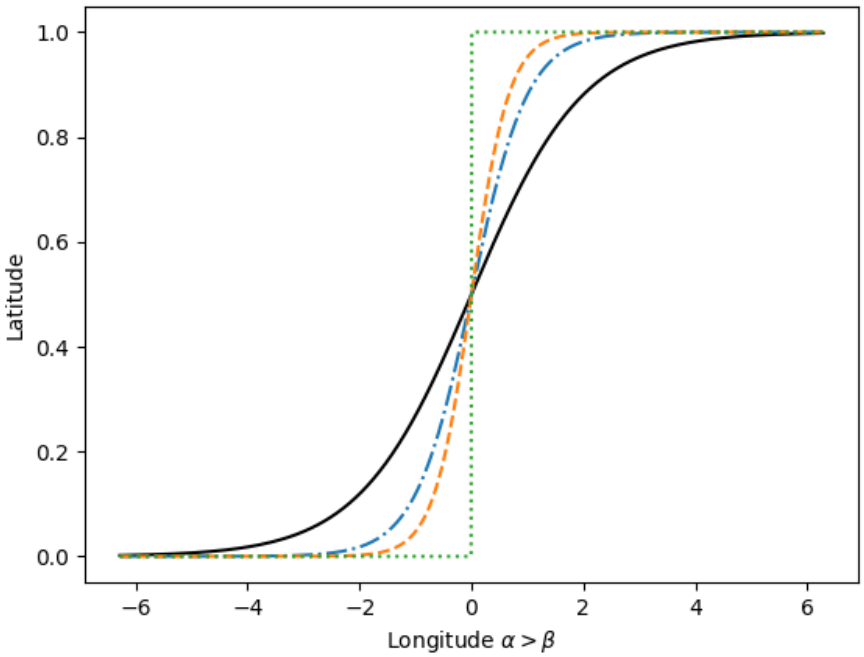
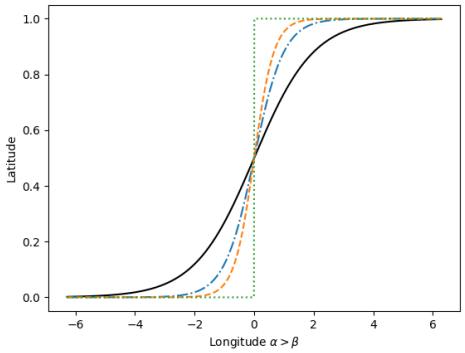


Figura 3.7: Se muestran cuatro funciones sigmoideas, la función con la línea negra uniforme corresponde $c = 1$, la línea "$\cdot - \cdot$" de color azul a $c = 2$, la línea "- - -" de color naranja a $c = 3$ y la línea "$\cdot \cdot \cdot$" de color verde a $c \to \infty$.

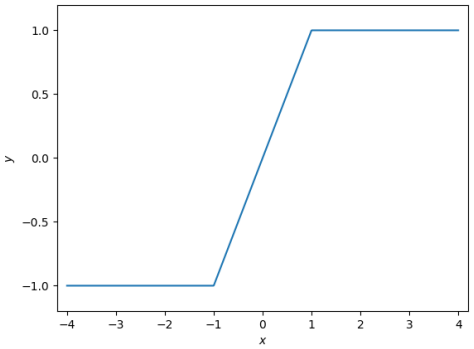
La función tangente hiperbólica es otra de las funciones de activación, ver figura 3.8c. Esta es una función real con dominio en $[-\infty, \infty]$ e imagen en $(-1, 1)$. Además esta función se puede expresar de varias maneras pero siendo la mas común

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.7}$$

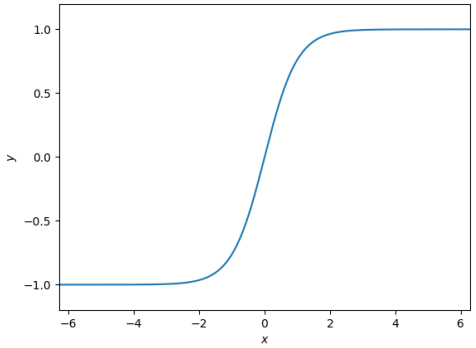
Aunque existen más funciones de activación diferentes a las mostradas anteriormente, las ecuaciones (3.5), (3.6) y (3.7), son de las más populares que se utilizan en las redes neuronales artificiales. Estas son funciones de activación continuas, es decir, el límite existe en todo punto de la función a diferencia de la función de activación que se utiliza para el perceptrón, que es una función de activación discontinua en el punto cero.



(a) Función de activación sigmoidea.



(b) Función de activación rampa.



(c) Función de activación tangente hiperbólica.

Figura 3.8: Se muestran diferentes funciones de activación.

Retomando la ecuación (3.4), el siguiente paso es evaluar el valor H_l dentro de la función de activación, como se hizo en el perceptrón. Entonces, sea la función de activación que se haya elegido quedaría de la forma

$$H_l = \mathbf{F}\left(\sum_{i=1}^{n} x_i w_{il}\right). \tag{3.8}$$

Es necesario calcular el valor de todas las salidas O_m, la forma de hacerlo es haciendo operaciones, como las que se hicieron para calcular los H_l. En la figura 3.6, del lado derecho de la imagen se ve la conexión entre las l neuronas y las m salidas que estas se encuentran conectadas por los pesos W^2, que a partir de ahora se escribirán como $\tilde{w}_{lm}$. El valor de los pesos $\tilde{w}_{lm}$ son generados de manera aleatoria entre 1 y -1. Como en la ecuación (3.8) ya se calculó el valor de las H_l neuronas, ahora para encontrar el valor de las salidas, simplemente se sigue propagando hacía adelante y esta vez utilizando a los pesos $\tilde{w}_{lm}$ correspondientes, así que

$$O_m = \sum_{l=1}^{n} H_l \tilde{w}_{lm}, \tag{3.9}$$

y evaluando dentro de la función de activación **G** (puede ser la misma que se utilizó en la ecuación (3.8), **F**), quedaría

$$O_m = \mathbf{G}\left(\sum_{l=1}^{n} H_l \tilde{w}_{lm}\right). \tag{3.10}$$

Nuevamente O es un arreglo de datos con m conjuntos de elementos que representan las salidas.

Se ha finalizado la descripción del funcionamiento de la propagación hacía adelante y se han generado O_m salidas. Ahora se introduce un valor deseado T para cada salida lo que significa que se tiene un número de m de valores deseados, o sea T_m . Como en el caso del perceptrón se genero una sola salida y esta generó un valor de salida igual a **uno** debido a la función de activación. Ahora supongamos que el problema requiere que la salida sea igual a **cero** (esto represento el valor deseado), por lo tanto, designa un error generado en la salida y es necesario corregirlo.

Al suponer que se cuentan con salidas, por ejemplo, si se tienen 3 salidas esto implica que se tiene 3 valores deseados para cada ejemplo de esta red neuronal. Ahora si se cuenta con un número m de salidas y m valores deseados respectivamente, por ende, surge un error de cada neurona y se calcula de la siguiente manera

$$E_m = \frac{1}{2} \|T_m - O_m\|^2 . \tag{3.11}$$

La ecuación (3.11) representa el error local de cada l neurona. Entonces una vez calculado, la función de error de cada neurona lo que procede ahora es sumar todos los errores, es decir, recoger todos los errores cuadráticos tal como

$$E^G = \sum_{i=1}^{m} E_i \tag{3.12}$$

En la figura 3.9 del lado derecho se representa cómo se calculan las ecuaciones (3.11) y (3.12)para los errores tanto local como global, respectivamente. Esto después de haber calculado la ecuación (3.10).

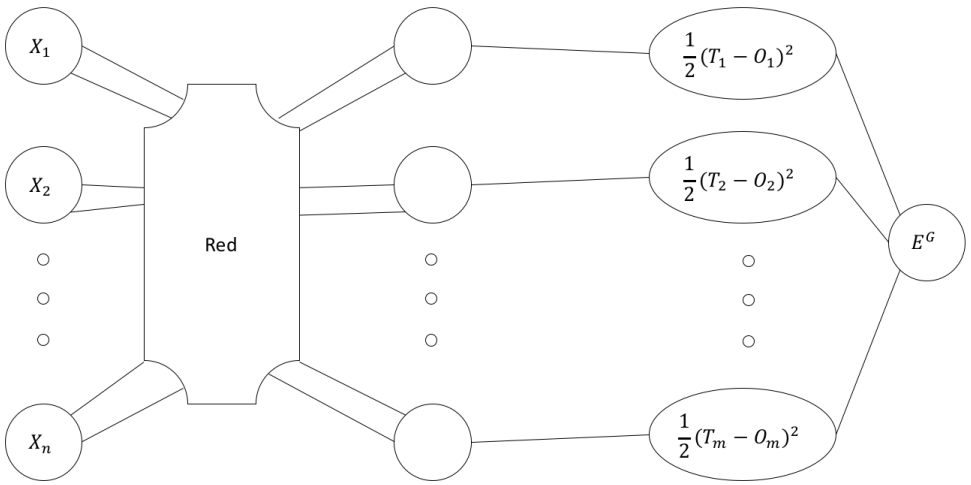


Figura 3.9: Red neuronal donde se muestra el cálculo del error local y global E^G, donde O_m representa las salidas y T_m representa los valores esperados.

Hasta ahora solo se ha hecho la propagación hacía adelante, pues el cálculo esta hecho para $k = 1$ capas ocultas con el fin de simplificar las cuentas. Pero, si se tuvieran k capas ocultas, la ecuación (3.8) cambiaría un poco la notación, es decir, para $k = 1$ con l_1 neuronas se escribe como $H^1_{l_1}$, para $k = 2$ con l_2 neuronas se tendría $H^2_{l_2}$ y así sucesivamente hasta las k capas ocultas, en general se remplazaría dicha ecuación por $H^k_{l_q}$ con l_q neuronas. La cantidad q de cada l_q neuronas pueden o no ser iguales, dicho de otro modo, $l_1 = 10$ neuronas, $l_2 = 50$ neuronas y así de manera consecutiva, no hay alguna restricción. Sin embargo, una vez definido la cantidad de capas ocultas y de neuronas, implica que se ha construido una única estructura de una red neuronal, y si se cambian los valores de k capas ocultas o l neuronas, denota otra única estructura de una red neuronal, lo cual tiene una consecuencia al momento de calcular los resultado, en otros términos, los resultados pueden mejorar o empeorar, estos argumentos se discuten en la capítulo 5.

Se ha calculado la *propagación hacía adelante* que genera salidas O_m y además se ha calculado la *función error* de cada salida, además del error global de la red neuronal artificial, el siguiente paso es que la red neuronal sea capaz de "aprender" un conjunto de datos, esto se muestra en la siguiente sección en la que se describe el algoritmo de aprendizaje conocido como *retropropagación*.

3.2.2. Algoritmo de aprendizaje

Existen algoritmos de aprendizaje supervisados y no supervisados, estos hace una diferencia importante cuando se quiere optimizar el aprendizaje de una computadora o de algún sistema que lo requiera por ejemplo los textos predecibles de nuestros teléfonos celulares.

Entrenamiento supervisado en el que se introducen a la red conjunto de datos en las unidades de entrada y una vez que se propaga esta información a las unidades de salida, se comparan con los objetivos propuestos, a través de una persona que patrocine sus datos, mide el error y utiliza un algoritmo para corregir los pesos de manera que este error se minimice.

Entrenamiento no supervisado en el que al introducir un conjunto de datos, el resultado numérico o la clasificación son desconocidos, por lo que el sistema debe descubrir características estadísticamente mas sobresalientes de la población de datos de entrada para poder adaptar la red.

En este caso, al introducir un arreglo de valores deseados T_m significa que estamos supervisado el algoritmo y que éste se modificara conforme T_m lo indique.

Lo ideal es que al calcular la función de error, ver ecuación (3.11), esta devuelva un error lo suficientemente pequeño para cada salida O_m.

El objetivo principal del algoritmo retropropagación, es minimizar E de la función error utilizando un proceso iterativo del descenso gradiente. En la red neuronal, los pesos W^1 y W^2 son inicializados con valores aleatorios, además son los únicos que son capaces de ser modificados para que el error cuadrático E sea lo más pequeño posible.

Prácticamente, la retropropagación funciona mediante la composición de funciones, una red neuronal en general tiene un total de h pesos como $W^1, W^2, ..., W^h$ entonces el descenso gradiente se calcula de la forma

$$\vec{\nabla} E = \left(\frac{\partial E}{\partial W^1}, \frac{\partial E}{\partial W^2}, ..., \frac{\partial E}{\partial W^h} \right), \tag{3.13}$$

para esta red neuronal se utilizo $k = 1$ capa ocultas, esto significa que solo tenemos $h = 2$ pesos, entonces, el gradiente total calculado es de la siguiente forma

$$\vec{\nabla} E = \left(\frac{\partial E}{\partial W^1}, \frac{\partial E}{\partial W^2} \right) = \left(\frac{\partial E}{\partial w_{nl}}, \frac{\partial E}{\partial \tilde{w}_{lm}} \right). \tag{3.14}$$

Si se considera los pesos entre la capa oculta y la salida se tendría entonces l x m variables y para cada una de estas direcciones tomando el sentido opuesto. En el que cada peso w_{nl} y $\tilde{w}_{lm}$ puede ser actualizado usando un incremento, donde Δw_{lm} es igual a $w_{nl}^{n+1} - \tilde{w}_{lm}^{n}$ (donde el nuevo peso es adaptado mientras que el viejo esta sin adaptar), entonces quedado así de la siguiente forma

$$\Delta w_{lm} = -\eta \frac{\partial E}{\partial w_{lm}}, \tag{3.15}$$

la variable η introducida en la ecuación (3.15) representa la constante de aprendizaje, es decir, es el *parámetro de proporcionalidad* que define el tamaño de paso de cada iteración en la dirección negativa del gradiente, ésta constante es considerada para que se cumpla la igualdad. Si η es muy grande significa que la red neuronal no aprenderá muy bien pero lograra terminar su proceso computacional bastante rápido y de manera análoga, si η es muy pequeño significa que aprenderá

bastante bien y que obtendrá un error de entrenamiento muy pequeño pero con una consecuencia computacional, el proceso será muy lento.

Una vez que se tiene este método calculado, este gradiente descrito puede ajustar valores de los pesos de la red iterativamente. De esta manera se espera encontrar el mínimo de la función error E, de la forma $\vec{\nabla} E = \vec{0}$.

Las ecuaciones (3.13) y (3.15) son las ecuaciones más importantes del algoritmo **retropagación**, y éstas están escritas de manera general. En la siguiente sección se mostrarán los pasos del algoritmo retropropagación.

3.2.3. Pasos del algoritmo

Las redes neuronales artificiales tienen cuatro pasos para completar un entrenamiento y estos se pueden dividir como

1. Propagación hacía adelante,
2. Retropropagación para las salidas,
3. Retropropagación para las capas ocultas,
4. Actualización de pesos.

Este algoritmo es detenido cuando el valor de la función error E sea lo suficientemente pequeño. En seguida se describirán a detalle los pasos del algoritmo dividido en partes.

Primer paso: Propagación hacía adelante.

Este paso ya se ha descrito en la sección 3.2.1, ahí se describe a detalle como se calcula la propagación hacía adelante. En resumidas cuentas, primero se tiene que aplicar la ecuación (3.8) para determinar el valor de neuronas H_l y no olvidar evaluar en alguna de las funciones de activación que se haya elegido. Posteriormente se aplica la ecuación (3.10) para determinar un dicho valor y después evaluar en alguna de las funciones de activación para obtener el valor de las salidas O_m.

Segundo paso: retropropagación para las salidas.

El algoritmo retropropagación consta de corregir los pesos y la forma que funciona es recorriendo de atrás hacía adelante, es decir, empieza corrigiendo los pesos por la parte de las salidas O_m. En el diagrama de la figura 3.10 se puede apreciar tal corrección en el que se toma la j-ésima salida. En el mismo diagrama lo que se encuentra dentro del recuadro, representa el calculo de los errores.

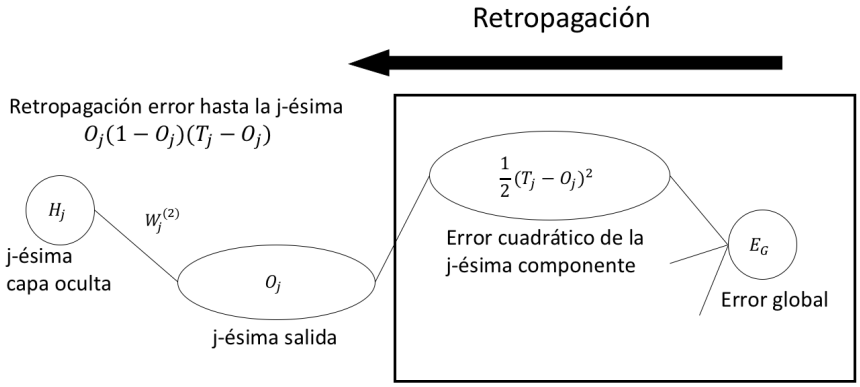


Figura 3.10: Retropropagación aplicando a las salidas.

Primero se empieza buscando el primer conjunto de derivadas parciales $\partial E/\partial \tilde{w}_{lm}$, esto es debido a la ecuación (3.14). En la figura 3.10 se puede ver el camino que toma el método retropropagación.

Al calcular la derivada de E respecto a los pesos $\tilde{w}_{lm}$, no hay que dejar de pasar desapercibido varios aspectos: se ve que

$$\frac{\partial E}{\partial \tilde{w}_{lm}} = \frac{\partial}{\partial \tilde{w}_{lm}} \left(\sum_{k=1}^{m} \frac{1}{2}(T_k - O_k)^2 \right), \tag{3.16}$$

al calcular la derivada parcial, se observa que T_l no depende de $\tilde{w}_{jk}$, la ecuación anterior resulta como

$$\frac{\partial}{\partial \tilde{w}_{lm}} \left(\sum_{k=1}^{m} \frac{1}{2}(T_k - O_k)^2 \right) = -\sum_{k=1}^{m} (T_k - O_k) \frac{\partial O_k}{\partial \tilde{w}_{lm}}, \tag{3.17}$$

después se calcula la derivada parcial $\partial O_k/\partial \tilde{w}_{lm}$ que aparece del lado derecho

de la ecuación y remplazando O_k por la ecuación (3.10) se obtiene

$$\begin{aligned}\frac{\partial}{\partial \tilde{w}_{lm}} O_k &= \frac{\partial}{\partial \tilde{w}_{lm}} F\left(\sum_{n=1}^{m} H_n \tilde{w}_{lm}\right) \\ &\rightarrow \quad x_k = \sum_{i=1}^{n} H_i \tilde{w}_{ik} \\ &= \frac{\partial F(x_k)}{\partial x_k} \frac{\partial x_k}{\partial \tilde{w_{lm}}} \\ &= F'(x_k) \frac{\partial}{\partial \tilde{w_{lm}}} \sum_{i=1}^{n} H_i \tilde{w_{ik}}, \end{aligned} \tag{3.18}$$

en el resultado de la ecuación se observa que aparece un termino en la derivada que es constante, es decir, H_i es constante ya que no depende de los pesos $\tilde{w}_{lm}$, luego reestructurando el resultado de la ecuación y resolviendo la parcial para el cual surgen deltas de Kronecker δ las cuales por definición son eliminadas por las sumaorias. Esto resulta de la siguiente forma

$$\frac{\partial}{\partial \tilde{w_{lm}}} \sum_{i=1}^{n} H_i \tilde{w_{ik}} = \sum_{i=1}^{n} H_i \delta_{il} \delta_{km} = H_l \delta_{km}, \tag{3.19}$$

entonces, sustituyendo la ecuación (3.19) dentro de la ecuación (3.18) la derivada parcial resulta como

$$\frac{\partial}{\partial \tilde{w}_{lm}} O_k = F'(x_k) H_l \delta_{km} \tag{3.20}$$

y por último, remplazando en la derivada parcial del error E, ver ecuación (3.17), queda como resultado

$$\begin{aligned}\frac{\partial E}{\partial \tilde{w}_{lm}} &= -\sum_{k=1}^{m} (T_k - O_k) F'(x_k) \delta_{km} H_l \\ &= -(T_m - O_m) F'(x_m) H_l \end{aligned} \tag{3.21}$$

donde $F'(x_k)$ representa alguna de las funciones de activación descritas anteriormente. Se pueden recopilar mediante inspección simple todos los términos

multiplicativos que definen el error de propagación inversa $\delta_j^{(2)}$ y este depende la función de activación que se haya elegido, por ejemplo, si se eligiera la función sigmoidea (ver ecuación (3.5)), la derivada resultaría de la forma

$$\frac{d}{dx}f(x) = \frac{d}{dx}\frac{1}{1+e^{-x}} = \frac{e^{-x}}{(1+e^{-x})^2} = (1-f(x))f(x) \tag{3.22}$$

ahora remplazando $f(x)$ por $F'(x_m)$ para la función de activación elegida con argumentos x_m y tomando el hecho que $F'(x_m) = O_m$ resulta

$$F'\left(\sum_{n=1}^{m} H_n \tilde{w}_{nl}\right) = (1-O_m)O_m \tag{3.23}$$

sustituyendo la derivada anterior en la ecuación (3.21) podemos definir un $\tilde{\delta}_j$ y este se define como

$$\delta_j^{(2)} = -(T_m - O_m)(1-O_m)O_m, \tag{3.24}$$

por lo que la derivada parcial para la función de activación sigmoidea se escribe como

$$\frac{\partial E}{\partial \tilde{w}_{lm}} = \left[(O_m - T_m)(1-O_m)O_m\right] H_l = \delta_j^{(2)} H_l. \tag{3.25}$$

Tercera parte: Retropropagación para las capas ocultas.

Como en el caso anterior, ahora se quieren calcular las derivadas parciales $\partial E/\partial w_{nl}$. Cada neurona l de la capa oculta es conectado con cada unidad m de las salidas que se conectan con los pesos $\tilde{w}_{lm}$.

Primero se empieza buscando el primer conjunto de derivadas parciales $\partial E/\partial w_{nl}$, esto es debido a la ecuación (3.14), pero ahora aplicado a

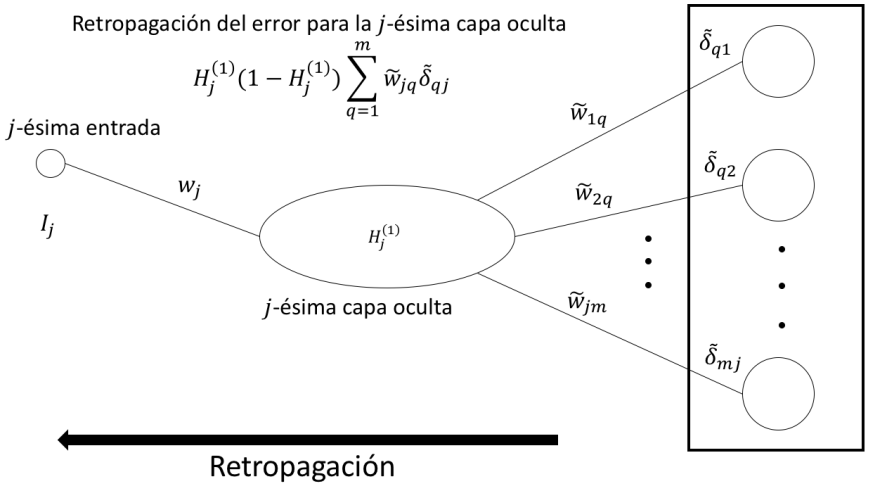


Figura 3.11: Retropropagación aplicando a las capas ocultas.

$$\begin{aligned}\frac{\partial E}{\partial w_{nl}} &= \frac{\partial}{\partial w_{nl}}\left(\sum_{q=1}^{n}\frac{1}{2}(T_q-O_q)^2\right) \\ &= -\sum_{q=1}^{n}(T_q-O_q)\frac{\partial}{\partial w_{nl}}O_q,\end{aligned} \tag{3.26}$$

se calcula la derivada parcial $\partial O_q/\partial w_{nl}$ y remplazando a O_q por la ecuación (3.10) se obtiene

$$\begin{aligned}\frac{\partial}{\partial w_{nl}}O_q &= \frac{\partial}{\partial w_{nl}}F\left(\sum_{r=1}^{n}H_r\tilde{w}_{rq}\right) \\ &\rightarrow \quad x_r=\sum_{r=1}^{n}H_r\tilde{w}_{rq} \\ &= \frac{\partial F(x_r)}{\partial x_r}\frac{\partial x_r}{\partial w_{nl}} \\ &= F'(x_r)\left(\sum_{r=1}^{n}\tilde{w}_{rq}\frac{\partial H_r}{\partial w_{nl}}\right),\end{aligned} \tag{3.27}$$

se puede apreciar que aparece un termino en la derivada que es constante,

es decir, $\tilde{w}_{rq}$ es constante ya que no depende de los pesos w_{nl}, además podemos escribir a $F'(x_r)$ que representa la función de activación en general. Realizando un proceso similar al anterior para $\partial H_r/\partial w_{nl}$, se tiene

$$\begin{aligned}
\frac{\partial H_r}{\partial w_{nl}} &= \frac{\partial}{\partial w_{nl}} F\left(\sum_{j=1}^{n} I_j w_{jr}\right) \\
&\rightarrow \quad x_j = \sum_{j=1}^{n} I_j w_{jr} \\
&= \frac{\partial F(x_j)}{\partial x_j}\frac{\partial x_j}{\partial w_{nl}} \\
&= \sum_{j=1}^{n} F'(x_j) I_j \frac{\partial F(w_{jr})}{\partial w_{nl}} \\
&= \sum_{j=1}^{n} F'(x_j) I_j \delta_{jn}\delta_{rl}, \qquad (3.28)
\end{aligned}$$

como en la ecuación (3.28) aparecen unas deltas de kronecker, es posible quitar la sumatoria junto con la delta correspondiente, es decir, esto resulta como

$$\sum_{j=1}^{n} F'(x_j) I_j \delta_{jn}\delta_{rl} = F'(x_n) I_n \delta_{rl} \qquad (3.29)$$

y sustituyendo la ecuación (3.29) dentro de la ecuación (3.28) tambiés se puede eliminar la otra δ restante, entonces

$$\begin{aligned}
F'(x_r)\left(\sum_{r=1}^{n} \tilde{w}_{rq}\frac{\partial H_r}{\partial w_{nl}}\right) &= \sum_{r=1}^{n} F'(x_r)\tilde{w}_{rq}F'(x_n)I_n\delta_{rl} \\
&= F'(x_l)\tilde{w}_{lq}F'(x_n)I_n, \qquad (3.30)
\end{aligned}$$

Por último la ecuación (3.30) se remplaza dentro de la ecuación (3.27) y por tanto la $\partial E/\partial w_{nl}$ resulta como

$$\begin{aligned}\frac{\partial E}{\partial w_{nl}} &= -\sum_{q=1}^{n}(T_q - O_q)F'(x_l\tilde{w}_{lq})F'(x_n)I_n \\ &= I_n F'(x_n)\sum_{q=1}^{n}(T_q - O_q)F'(x_l)\tilde{w}_{lq}, \end{aligned} \tag{3.31}$$

Como se puede notar en la ecuación (3.31) aparecen unas funciones de activación, como una de esas funciones de activación ya se había calculado en la tercera parte, entonces usando la misma función de activación(finción sigmoidea) para este caso la ecuación (3.31) anteriror resulta como

$$\frac{\partial E}{\partial w_{nl}} = -I_n H_j(1 - H_j)\sum_{q=1}^{n}(T_q - O_q)O_q(1 - O_q)\tilde{w} \tag{3.32}$$

El error de la retropropagación puede ser calculado de la misma forma para algún numero k de las capas ocultas y la expresión para la derivada parcial de E mantiene la misma forma analítica.

Cuarta parte: Actualización de los pesos.

Después que se hayan calculado todas las derivadas parciales de red, lo siguiente es actualizar los pesos en la dirección negativa del gradiente. Antes se había mencionado una constante de entrenamiento η que define el tamaño de paso de la corrección. La corrección de los pesos esta dado por

$$\Delta\tilde{w}_{ij} = \eta H_i^{(1)}\delta_j^{(2)} \qquad \text{for } i = 1, ..., k;\ j = 1, ..., l \tag{3.33}$$

y

$$\Delta w_{ij} = \eta I_i \delta_j^{(1)} \qquad \text{for } i = 1, ..., m;\ j = 1, ..., k \tag{3.34}$$

Es muy importante hacer correcciones a los pesos después de haber aplicado el error de la retropropagación que se calculó para todas las unidades dentro de la red.

3.3. Ejemplos de una red neuronal artificial

Es esta sección se tratará toda la información recolectada anteriormente, se dará un ejemplo simbólico, uno numérico y por último, se ejemplificará de manera parcial un problema de reconocimiento de imagenes para reconocer números.

3.3.1. Ejemplo simbólico

Ahora se aplicaran todos los conceptos descritos con anterioridad, por ejemplo se analizará una red neuronal sencilla de manera simbólica. En este ejemplo se tendrán dos entradas, tres capas ocultas y dos salidas. Se aplicarán los cuatro pasos del algoritmo descritos anteriormente.

La red neuronal a entrenar consta de dos entradas x_1 y x_2 y según los pasos descritos con anterioridad lo primero que se hace es aplicar la *propagación hacía adelante*.

Consta en multiplicar las entradas por los pesos que se encuentran entre las capas ocultas. La función de activación elegida para este caso es la función Sigmoidea, entonces el valor total de H_1, H_2 y H_3, es

$$\begin{aligned} H_1 &= (x_1 * w_{11}) + (x_2 * w_{21}) \rightarrow \quad H_1 = Sig(H_1), \\ H_2 &= (x_1 * w_{12}) + (x_2 * w_{22}) \rightarrow \quad H_2 = Sig(H_2), \\ H_3 &= (x_1 * w_{13}) + (x_2 * w_{23}) \rightarrow \quad H_3 = Sig(H_3), \end{aligned}$$

lo mismo se hace para las salidas, por lo que

$$\begin{aligned} O_1 &= (H_1 * \tilde{w}_{11}) + (H_2 * \tilde{w}_{21}) + (H_3 * \tilde{w}_{31}) \rightarrow \quad O_1 = Sig(O_1), \\ O_2 &= (H_1 * \tilde{w}_{12}) + (H_2 * \tilde{w}_{22}) + (H_3 * \tilde{w}_{32}) \rightarrow \quad O_2 = Sig(O_2), \end{aligned}$$

En el siguiente paso es aplicar la *retropropagación* para las salidas, como lo indica la ecuación (3.24). Una vez que se han obtenido las salidas ahora se calcula el error, además se deben tener en cuenta a los valores deseados $T = (T_1, T_2)$, entonces

$$\delta_{O_1} = O_1(1 - O_1)(T_1 - O_1)$$
$$\delta_{O_2} = O_2(1 - O_2)(T_2 - O_2)$$

Después aplicamos el *retropropagación* a las neuronas de las capas ocultas, aplicando el tercer caso y debido a la ecuación (**??**). Dentro de esta ecuación se nota que se están tomando en cuenta los pesos viejos. Por lo que

$$\delta_{H_1} = H_1(1 - H_1)(\delta_{O_1}\tilde{w}_{11} + \delta_{O_2}\tilde{w}_{12})$$
$$\delta_{H_2} = H_2(1 - H_2)(\delta_{O_1}\tilde{w}_{21} + \delta_{O_2}\tilde{w}_{22})$$
$$\delta_{H_3} = H_3(1 - H_3)(\delta_{O_1}\tilde{w}_{31} + \delta_{O_2}\tilde{w}_{32})$$

En el último paso nos queda actualizar los pesos como lo indican las ecuaciones (3.33) y (3.34) y estas quedan de la forma

$$\tilde{w}_{11} = \tilde{w}_{11} + \eta\delta_{O_1}O_1 \qquad \tilde{w}_{12} = \tilde{w}_{12} + \eta\delta_{O_2}O_1$$
$$\tilde{w}_{21} = \tilde{w}_{21} + \eta\delta_{O_1}O_2 \qquad \tilde{w}_{22} = \tilde{w}_{22} + \eta\delta_{O_2}O_2$$
$$\tilde{w}_{31} = \tilde{w}_{31} + \eta\delta_{O_1}O_3 \qquad \tilde{w}_{32} = \tilde{w}_{32} + \eta\delta_{O_2}O_3$$

$$w_{11} = w_{11} + \eta\delta_{H_1}x_1 \qquad w_{21} = w_{21} + \eta\delta_{H_1}x_2$$
$$w_{12} = w_{12} + \eta\delta_{H_2}x_1 \qquad w_{22} = w_{22} + \eta\delta_{H_2}x_2$$
$$w_{13} = w_{13} + \eta\delta_{H_3}x_1 \qquad w_{23} = w_{23} + \eta\delta_{H_3}x_2$$

Con η como la constante de aprendizaje que por lo regular es igual a uno para casos no tan extremos, es decir, no es uno cuando se tiene una gran cantidad de datos.

3.3.2. Ejemplo numérico con valores reales

En la siguiente figura 3.12 vemos un red sencilla que contiene dos entradas, una capa oculta con dos neuronas, una salida y además tiene un valor deseado

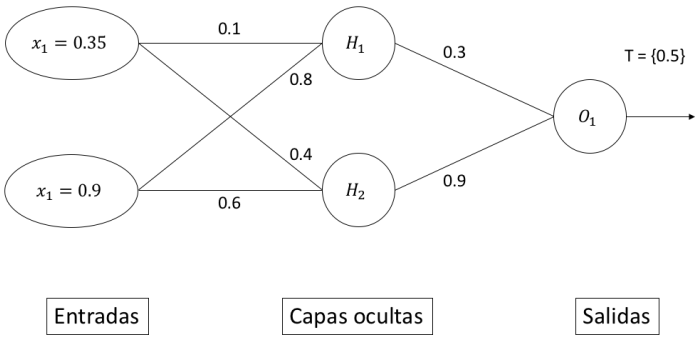


Figura 3.12: Ejemplo numérico de una red neuronal con dos entradas, dos capas ocultas y una salida.

$T = 0.5$. Entonces aplicando los pasos descritos anteriormente y asumiendo que tiene la función de activación Sigmoidea, se tiene

1. Feed-forward.
   $H_1 = (0.35 * 0.1) + (0.9 * 0.8) = 0.775$.
   Salida $H_1 = Sig(H_1) = 0.68$
   $H_2 = (0.9 * 0.6) + (0.35 * 0.4) = 0.68$.
   Salida $H_2 = Sig(H_2) = 0.6637$
   $O_1 = (0.3 * 0.68) + (0.9 * 0.6637) = 0.80133$.
   Salida $O_1 = Sig(O_1) = 0.69$

2. Backpropagation para las salidas.
   Error de $O_1 = \delta_{O_1} = O_1(1 - O_1)(T_1 - O_1) = 0.69(1 - 0.69)(0.5 - 0.69)$

3. Backpropagation para las capas ocultas.
   Error de $H_1 = \delta_{H_1} = \delta_{O_1} x w^{(1)} = -0.00537837$
   Error de $H_2 = \delta_{H_2} = \delta_{O_1} x w^{(2)} = -0.01613511$

4. Actualización de pesos.

Para $\tilde{w}_{lm}$, con $\eta = 1$	Para w_{nl}, con $\eta = 1$
$\tilde{w}_{11} = \tilde{w}_{11} + \eta\delta_{O_1} O_1 = 0.2724$	$w_{11} = w_{11} + \eta\delta_{O_2} x_1 = 0.0982$
$\tilde{w}_{21} = \tilde{w}_{21} + \eta\delta_{O_1} O_2 = 0.8729$	$w_{21} = w_{21} + \eta\delta_{O_2} x_2 = 0.7952$
	$w_{12} = w_{12} + \eta\delta_{O_2} x_1 = 0.3944$
	$w_{22} = w_{22} + \eta\delta_{O_2} x_2 = 0.5855$

Como ultimo paso es calcular los errores, el error local viejo de las salidas es de $E_L = -0.19$. Si se volviera a ejecutar usando los nuevos pesos ahora el error disminuiría a $E_L = -0.18205$. Mientras que el error global estaría dado por $E_G = 0.01810$.

3.3.3. Ejemplo reconocimiento de patrones de una imagen

Un problema en el que se puede aplicar el algoritmo retropagación es en el de reconocimiento de patrones, ya sea un conjunto de datos que represente ubicaciones, movimientos, manejo de brazos robóticos, imágenes, entre otros.

En este caso, se usara para el reconocimiento de imágenes. Supongamos que se tiene un conjunto de imágenes en blanco y negro, cada una de ellas tiene grabado un número del 0 al 9, lo que significa que se tiene un total de 10 imágenes. Además el tamaño de cada imagen es de 5x7 píxeles.

Primero se analiza como es que se va a estructurar el problema, lo único que se sabe hasta ahora es que se tienen 10 imágenes y cada una de ellas contiene grabado el número 0 hasta 9. El tamaño total de cada imagen es 35 píxeles y ademas son a blanco y negro. En la figura 3.13 vemos algunas de las imágenes que se desean aprender para después reconocerlas. Se puede observar que están bien definidas, es decir,cada píxel o es blanco u oscuro. Por lo que solo se tienen dos opciones, entonces se puede representar al color blanco con un cero y al color negro con un uno, de esta manera se tiene una forma de representar a cada imagen.

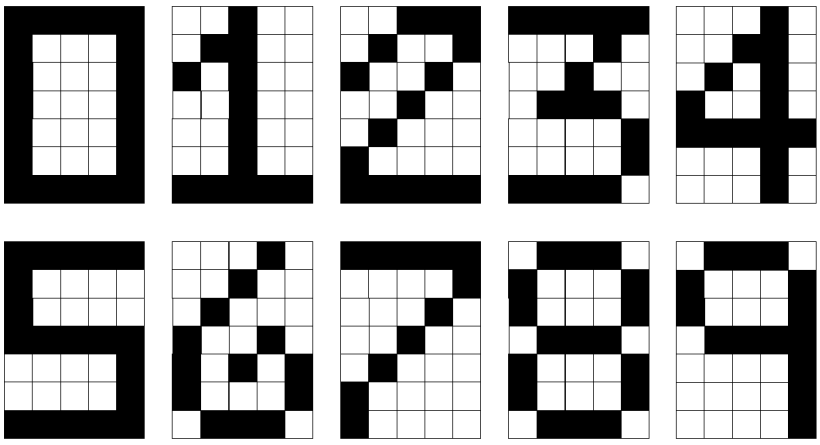


Figura 3.13: Ejemplo de una configuración de patrones de números.

Para comprender mejor lo dicho anteriormente, tomemos una imagen mas pequeña de (2 x 2) con un total de 4 píxeles, ver la figura 3.14. Como se desea aprender este patrón significa que tenemos que introducir los datos como entrada a una red neuronal. Se observa que el primer píxel es de color blanco, entonces a ese píxel se representa por un cero, después el siguiente píxel es de color negro y se representa por un uno, en seguida se tiene un píxel de color blanco que le corresponde un valor cero y por último se tiene un péxel negro que corresponde al valor uno. Por lo tanto, se tiene el conjunto de datos como entrada de la siguiente forma $x_1 = [0, 1, 0, 1]$. Además se observa que esta simple red neuronal tiene una capa culta con tres neuronas y dos salidas. Cada salida le corresponde un valor esperado, significa que esta imagen pequeña de (2 x 2) píxeles le corresponda el patrón asignado, o sea $T = [0, 1]$.

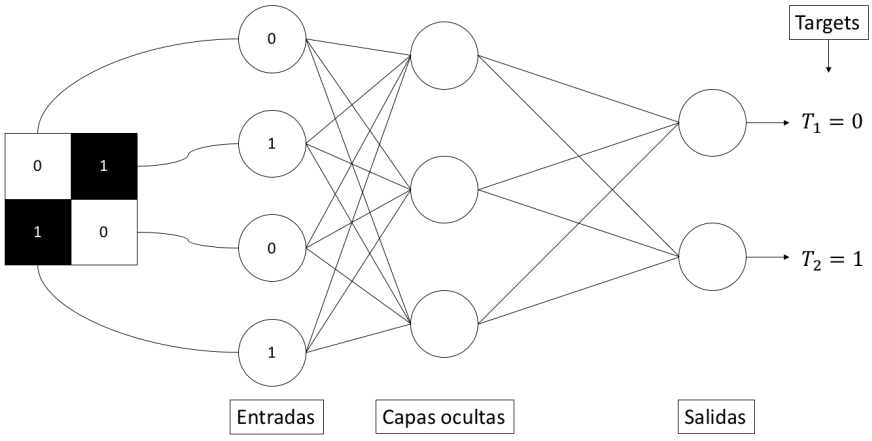


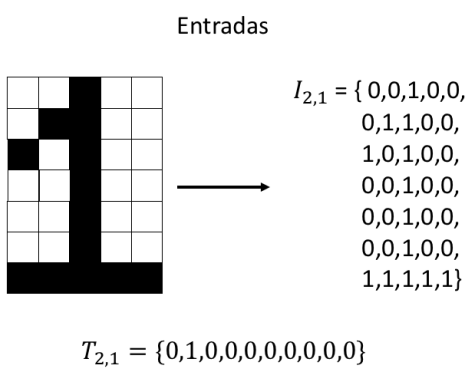
Figura 3.14: Ejemplo de una imagen de (2 x 2) píxeles donde se registran los datos en forma de espiral y estos serán procesados por una red neuronal

Una de las reglas sería empezar por la parte superior de la imagen e ir registrando los datos posterior a ello se pasa a la segunda fila y así sucesivamente, esto se conoce como registro por filas. Otra regla sería, empezar hacer conteo por columnas o haciendo una espiral (ver figura 3.14), en fin, mientras se mantenga una regla establecida, la red neuronal funcionara. Esto es solo para no tener errores futuros al momento de querer reconocer una imagen.

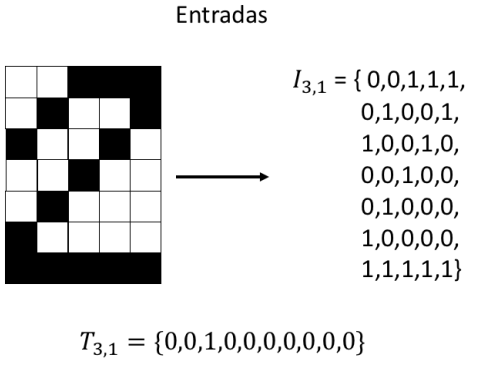
Una vez descrito la forma de estructurar los datos, ahora se toma una imagen de la figura 3.13, por ejemplo sea la imagen que representa al uno (ver figura 3.17a) y al dos (ver figura 3.17b). Como en el ejemplo anterior pero ahora se tienen más píxeles, se construye un conjunto de datos correspondientes a cada imagen. Así, se pude ir construyendo los demás patrones de cada imagen por lo que al final se

tendría un total de 10 conjuntos de datos y estos pueden ser representados por una matriz I, donde en la primera fila se encuentran los datos que representan al cero en la segunda fila a los datos que representan al uno y así sucesivamente hasta llegar al número nueve. Esto implica que se tiene una matriz de dimensiones (10 x 35) y se ve de la forma

$$I_{10,35} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 1 & \dots & 1 \end{bmatrix}$$



(a) Datos para la imagen uno, con la forma de representar los valores de entrada y su respectivo valor deseado $T_{2,1}$.



(b) Datos para la imagen uno, con la forma de representar los valores de entrada y su respectivo valor deseado $T_{3,1}$.

Figura 3.15: Se muestran dos conjuntos de patrones, del lado izquierdo se encuentra el número 1 y del lado derecho el número 2, ambos con su respectivo patrón de entrada y valor esperado respectivamente.

Es importante tener en cuenta que los valores deseados T correspondientes a cada imagen sean diferentes, es decir, el valor esperado de la imagen uno es diferente al valor esperado de la imagen dos, ver en la parte inferior de la figura 3.17. Lo mismo para los otros números. Entonces la matriz de valores esperados con dimensiones (10 x 10) se puede ver de la forma

$$T_{10,10} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & \dots & T_{1m} \\ T_{21} & T_{22} & T_{23} & \dots & T_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T_{l1} & T_{l2} & T_{l3} & \dots & T_{lm} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Hasta ahora se ha construido toda la estructura, sin embargo el problema no esta completo, es decir, si observamos las imágenes de la figura 3.13 se observa que hay una imagen por cada número, lo cual en la vida real hay muchas maneras de escribir por ejemplo el número cero, en la figura 3.16 se puede apreciar dos formas de un representar a un cero. Se observa que ambos ceros tiene diferentes patrones, es decir, un cero es diferente al otro. Aunque los humanos podemos determinar que ambas imágenes son un cero, para una computadora no lo es.

Si se deseara agregar ambos ceros a nuestro conjunto de datos del este ejemplo, se tendría que asignarle un valor esperado. Como ambas imágenes representan al número cero por lo tanto se le asignaría el mismo valor esperado, al igual si se tuvieran n imágenes seguirían compartiendo el mismo valor esperado. Entonces el tamaño de la matrices correspondientes serían $I = I_{11,35}$ y $T_{11,10}$. Como no es el caso de que se tengan más de una imagen que represente al cero o a cualquier otro valor por lo tanto se trabaja con los datos disponibles desde un principio.

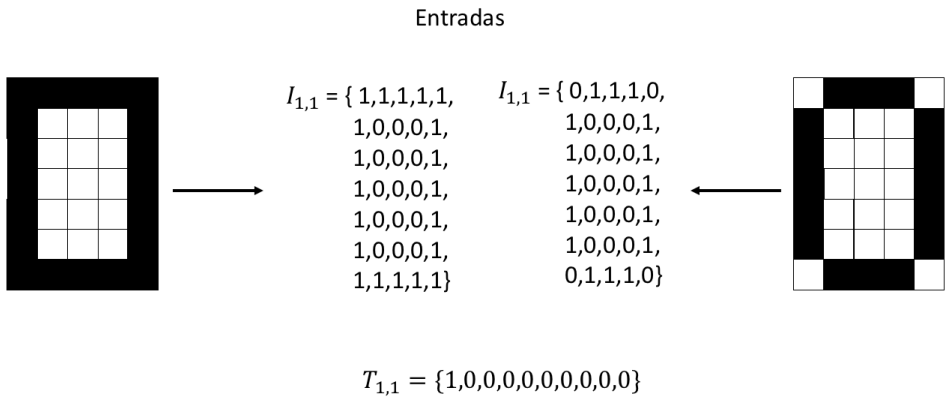


Figura 3.16: Dos imágenes que representan al número cero con un conjunto de datos diferentes pero con el mismo valor esperado T.

El siguiente paso es determinar el valor de los pesos w_{nl} y $\tilde{w}_{lm}$. Para hacer esto, lo primero que se debe hacer es determinar cuantas neuronas son las que vamos a trabajar y como no hay un regla para determinar el numero de neuronas sean las adecuadas por lo tanto para este problema se eligieran un total de 20 neuronas. Además los valores de las matrices serán números aleatorias dentro del intervalo $[-0.5, 0.5]$.

La forma de calcular la red neuronal es la misma que los ejemplos anteriores, se entrena un patrón a la vez, que de hecho esa es la forma correcta para entrenar una red aunque lo idea es que se entrene con los patrones de manera aleatoria.

Es importante no entrenar un solo patrón muchas veces y luego pasar al otro patrón distinto ya que no es conveniente hacer esto, debido a que la red neuronal ajustará los pesos para ese patrón y luego, al pasar al otro patrón y ser entrenarlo demasiadas veces, pues ahora ajustará los pesos para este patrón y en consecuencia perderá información para el primer patrón entrenado.

El entrenamiento se debe detener cuando el error global sea lo suficientemente pequeño, y junto a ella se revisa la efectividad de reconocimiento (mejor conocido como validación de la red neuronal), que consta de evaluar todas las imágenes para cada iteración. Esto ayuda a saber si la red es capas de reconocer a las mismas imágenes con las que esta entrenando. Una vez conseguido un error pequeño y una efectividad aproximadamente de un 100 %, se puede considerar que la red ya esta entrenada pero aún no esta lista para ser usada, ya que queda un último paso y es el de validar utilizando un patrón desconocido para red neuronal. En este caso se utilizó el número cero posicionado de lado derecho de la figura 3.16.

En la tabla 3.1 se muestran los datos de entrenamiento como el número de veces que se entreno la red, el error global de cada iteración de entrenamiento y la efectividad de la red. Se puede apreciar que el error global es muy grande en la primer entrenamiento lo cual es algo que se esperaba ya que las matrices de pesos son inicializadas de manera aleatoria, esto también implica que también arroje una efectividad baja. En el segundo entrenamiento resulto ser bastante bueno para la efectividad, sin embargo el error global es grande. Se puede observar que el error disminuye mientras se sigue entrenando la red neuronal y que la efectividad sigue aumentando. En el entrenamiento 6 se tuvo una efectividad del 80 % y luego en el siguiente entrenamiento la efectividad bajo a un 70 %, esto sucede ya que los pesos w_{nl} y w_{lm} siempre se ajustan. Lo más importante de la red es que el error este disminuyendo secuencialmente.

Cuando se llega al entrenamiento número 18 se obtiene una efectividad del 100 % y el error global es 4 veces menos respecto al primero error global obtenido en el primer entrenamiento. Si se deja entrenando la red neuronal, se observa que el error disminuye bastante. Lo ideal sería que el error global este muy cercano al cero pero esto no garantiza que la efectividad sea siempre de un 100 % así que es bueno saber que esta pasado mientras se esta entrenando la red neuronal.

El siguiente paso es saber si la red neuronal es capas de predecir un patrón desconocido. Utilizando el conjunto de imágenes que desconoce por completo al red neuronal, ver figura 3.18. Se muestran distintos imágenes que representan a los números de una forma diferente.

Evaluando dentro de la red neuronal se obtiene resultados que se pueden ver en

Tabla 3.1: Valores medidos del entrenamiento de una red neuronal artificial.

Entrenamiento	Error global	Efectividad
1	4.36318111668	10.0 %
2	4.13919530686	40.0 %
3	3.99095099704	50.0 %
4	3.85956242331	80.0 %
6	3.63419763736	80.0 %
7	3.53407399007	70.0 %
13	2.97903787796	70.0 %
14	2.89147849538	80.0 %
15	2.80637207221	90.0 %
17	2.64481290233	90.0 %
18	2.56867610718	100.0 %
28	1.94968739826	100.0 %
37	1.53574559867	100.0 %
38	1.49573939222	100.0 %
50	1.08938582442	100.0 %

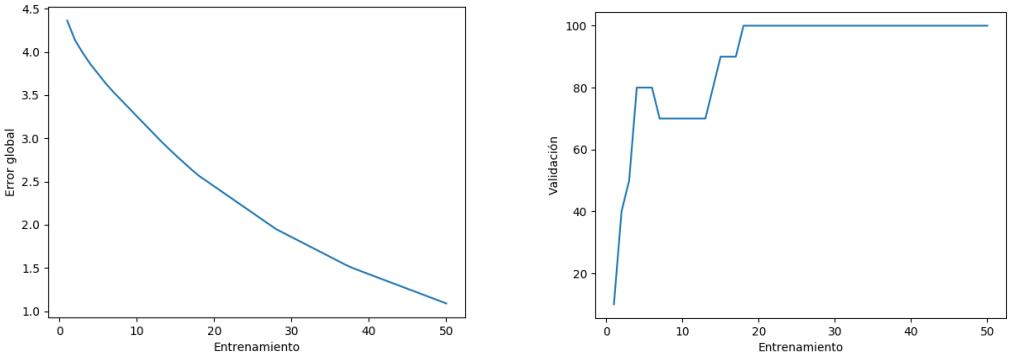
la tabla 3.2. Se observa que cuando se le preguntó a la red neuronal el patrón cero, este logró reconocerlo, al igual que al uno, dos, tres, cuatro, seis, siete y nueve. Si se comparan las imágenes con las que se entrenó y las que se uso para predecir se aprecia que son muy similares. Las imágenes que no logro reconocer fue el cinco y el ocho. El cinco resulto ser el ocho y el ocho resulto ser el cero, si se hace una comparación a simple vista el cinco es muy similar al ocho y el ocho es muy similar al cero por eso es que fueron predichas de esta manera. Además de que el cero y el ocho son patrones demasiado similares similares.

Estos resultados dependen de como fue que se realizó dicha predicción, es decir, al momento de clasificar la información. Prácticamente lo que se hace es calcular la norma y encontrar el error mas pequeño al cual se le asigna dicha predicción.

Esto se calcula de la siguiente manera

$$P = \|O - T_i\|^2, \tag{3.35}$$

se evalúa la salida O y se compara con todos los otros valores registrados T_i, después se calcula el error y el mas pequeño se asigna dicha posición de predicción.



(a) Gráfica del error global. (b) Gráfica de validación.

Figura 3.17: Se muestran dos gráficas correspondientes a los resultados acerca del entrenamiento de reconocimiento de números.

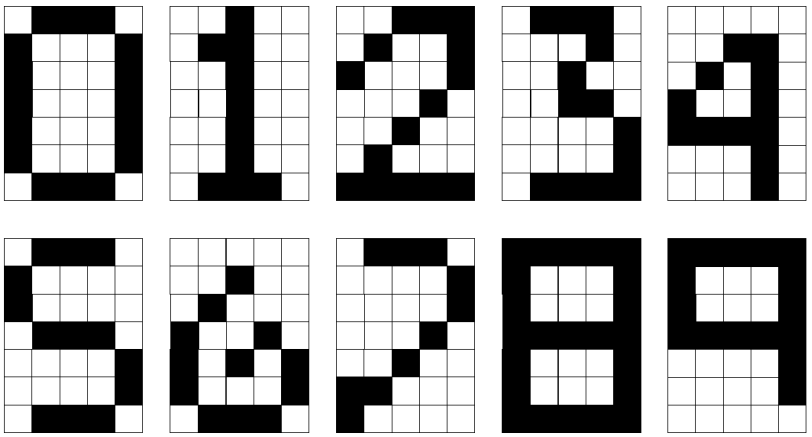


Figura 3.18: Forma de predicción para algunos patrones

Para fines prácticos este ejemplo sirve para construir una red neuronal que puede ser usada en muchos problemas, en los que requieran precisión de datos y estos se necesiten ser **clasificados**.

Tabla 3.2: Valores predichos por la red neuronal artificial.

Patrón	Valor reconocido
0	0
1	1
2	2
3	3
4	4
5	8
6	6
7	7
8	0
9	9

3.4. Discusión

Vemos que las redes neuronales son buenas para resolver problemas en los que es necesario clasificar información. Aunque existen muchas otras redes neuronales para el propósito de éste trabajo fue el método mejor acoplado. Una de las ventajas de usar estos modelos es que se tienen más resultados teóricos y una mayor cantidad de aplicaciones reales, lo cual refleja cierto proceso de maduración del área.

Sin embargo, hasta ahora solo se ha comentado el funcionamiento de una red neuronal, además se han explicado ciertos ejemplos que permiten entender el funcionamiento de una red neuronal simple hasta un conjunto de redes neuronales (ejemplo reconocimiento de patrones, sección (3.3.3)).

En el capitulo 4 se aplicara el uso de las redes neuronales para la detección de focos de reproducción de los mosquitos. Se utilizará el mismo procedimiento que en el ejemplo de reconocimiento de patrones, a diferencia de que los conjuntos son datos diferentes, es decir, no solo unos o ceros.

Capítulo 4

Aplicación a la propagación de mosquitos

En la actualidad, se ha estado hablando de insectos que transmiten efermedades infecciosas tales como la fiebre amarilla, dengue, malaria, chucunguya y zika[9], algunas de estas puede llegar a causar hasta la muerte. Éstos insectos pertenecen a la familia Culicidae y son conocidos como mosquitos o zancudos. Además, existen una gran cantidad de géneros alrededor del mundo.

La forma en la mosquitos nacen, consta de cuatro fases distintas: huevo, larva, pulpa y adulto. Dependiendo del género, los mosquitos pueden tardarse en nacer desde quince días hasta un mes dependiendo de la temperatura del ambiente [10]. Por ejemplo si la temperatura es baja significa que se desarrollan lentamente o de lo contrario aumenta su desarrollo cuando la temperatura ambiente es aproximadamente 25°C. Para temperaturas mayores los mosquitos suelen no desarrollarse.

El lugar donde se desarrollan los mosquitos por lo general es donde hay agua estancada y para cumplir sus cuatro fases de nacimiento se necesita por lo menos de una semana. Así es posible encontrar mosquitos en pantanos, canales, charcos, riberas de ríos, costas, agujeros de árboles y hasta en las plantas, en general los mosquitos pueden nacer en algún tipo de recipiente siempre y cuando el huevo este al aire libre. No es necesario que haya una gran cantidad de agua, a veces basta con 1cm de agua para completar la fase larvaria [11].

Es común ver mosquitos cuando la precipitación está en modo lluvia, lo que significa que hay humedad y se generan charcos. Cuando casi no suelen verse mosquitos es a temperaturas bajas o en periodos de sequía, esto significa que los huevos quedan inactivos [11]. Hay géneros de mosquitos que depositan sus huevos

en lugares donde haya un posibilidad de inundación en alguna hábitat, como los ya mencionados anteriormente.

La alimentación de los mosquitos varía dependiendo las condiciones. Los mosquitos hembra, poseen una pieza bucal en forma de aguja preparada para perforar la piel de los mamíferos o incluso hasta aves, reptiles y anfibios para succionar la sangre. Así las hembras toman las proteínas proporcionadas por la sangre e inician el ciclo gonotrófico que significa poner huevos. Mientras que para los mosquitos macho, su alimentación consiste en néctar, savia y jugos de frutas, generalmente ricos en azúcares. También hay géneros que no se alimentan de sangre, más bien sus larvas son depredadores de otras larvas de mosquitos [12].

En secciones posteriores, se describirá el problema principal que se va a resolver en esta tesis: se utilizará un modelo matemático para hacer una simulación del comportamiento de los mosquitos simulando los posibles datos obtenidos por las cajas para posteriormente ser usados por herramientas de inteligencia artificial, se analizaran los datos para obtener las posiciones de las fuentes de mosquitos.

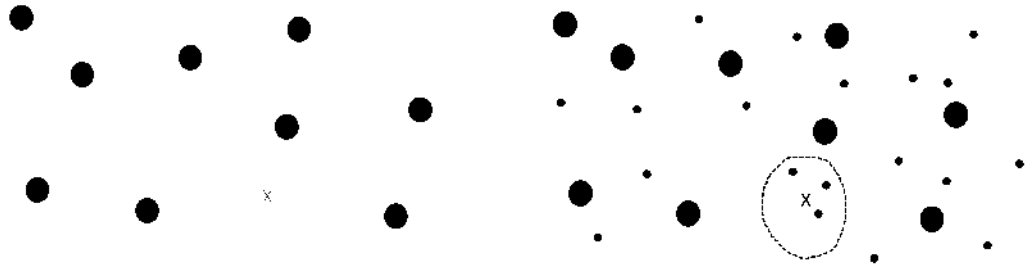
4.1. Modelo del problema

Sabemos que los mosquitos se desarrollan dependiendo las condiciones del clima. Esto significa que de alguna manera es importante tener registro del comportamiento climatológico. Para esto en el capítulo 2 se construyó una caja contadora de mosquitos que se encarga de conseguir datos de las variables climatológicas y por supuesto el número de mosquitos capturados. Esta caja tiene la capacidad de obtener datos como la velocidad y dirección de viento, la precipitación, temperatura, humedad y la cantidad de mosquitos capturados debido a los sensores que posee.

Los datos de las variables climatológicas y el conteo de los mosquitos serán estudiados con el objetivo de predecir la fuente de reproducción de los mosquitos.

En la figura 4.1a se muestra una imagen que tiene varias x grandes y un circulo negro. Representa una región donde los círculos negros significan las fuentes de reproducción de los mosquitos y la equis representa la caja que se encargará de hacer las mediciones. Conforme los mosquitos se reproduzcan, estos irán en busca de comida o néctares para su supervivencia y reproducción como se muestra en la figura 4.1b y son representados por círculos negros más pequeños. La caja contadora de mosquitos cuenta con una región de captura, es decir, si un mosquito entra a la región de captura, este mosquito será capturado para después generar

un registro perteneciente al conteo de mosquitos.



(a) Región sin propagar mosquitos. (b) Región propagada por mosquitos.

Figura 4.1: Se muestran dos figuras a) y b), donde los círculos negros representan las fuentes de reproducción y las X representan las cajas contadoras de mosquitos. En la figura de la derecha a), los círculos grandes representan las fuente de reproducción de mosquitos y los círculos pequeños representan a los mosquitos.

Después de un tiempo se generará un archivo de datos reales respecto al conteo y registro de variables climatológicas y estos datos serán ingresados a una red neuronal para que estos sean ajustados por una dicha red. Sin embargo, utilizar una sola caja contadora de mosquitos no es suficiente como para poder predecir la ubicación de las fuentes de reproducción de los mosquitos. Por lo que es necesario construir más cajas contadoras de mosquitos y distribuirlas dentro de la región que se va analizar. En la figura 4.2 se puede observar la distribución de varias cajas contadoras de mosquitos y fuentes de reproducción dentro de dicha región.

El problema consta en encontrar un patrón de reconocimiento que permita a la red neuronal ser capaz de localizar la fuente de mosquitos. La forma en la que se utilizaran los datos dentro de la red neuronal será ingresando los valores capturados por las cajas contadoras de mosquitos respecto al registro de las variables climatológicas y la cantidad de mosquitos. Recordando la construcción de una red neuronal como la que se describió en el capitulo 3, en la figura 4.3 se muestra esquemáticamente la forma de ingresar datos para entrenar la red neuronal. En total se tendrían seis datos como entrada y se tendrá un conjunto de datos de entrada por cada caja contadora que se haya construido.

Cada caja contadora de mosquitos registrara un patrón característico de la ubicación donde se encuentra tal caja. Por ejemplo, en la figura 4.2 se tienen un total de siete cajas contadoras de mosquitos que están distribuidas dentro de una región, así, para identificar cada caja, se construyen subregiones, en este caso se puede considerar una caja por cada subregión, pero no esta restringido a que solo haya una sola caja, ya que puede haber más cajas contadoras de mosquitos. En la

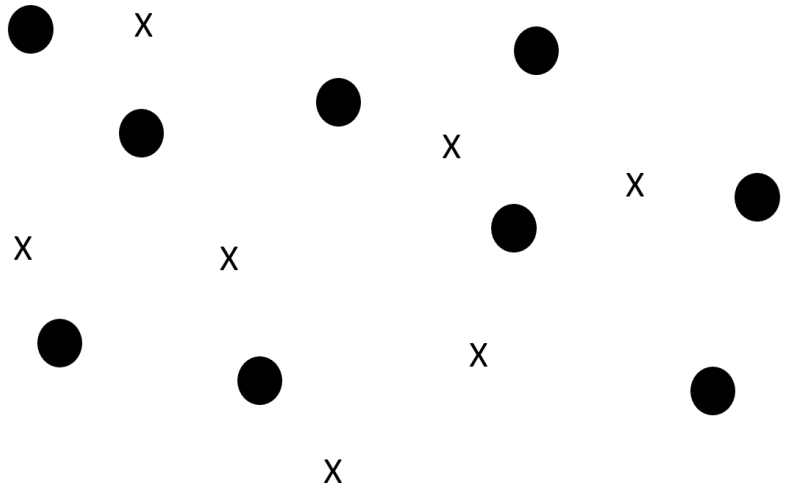


Figura 4.2: Región donde hay fuentes de reproducción (representados por círculos negros) y cajas contadoras de mosquitos (representados por una X).

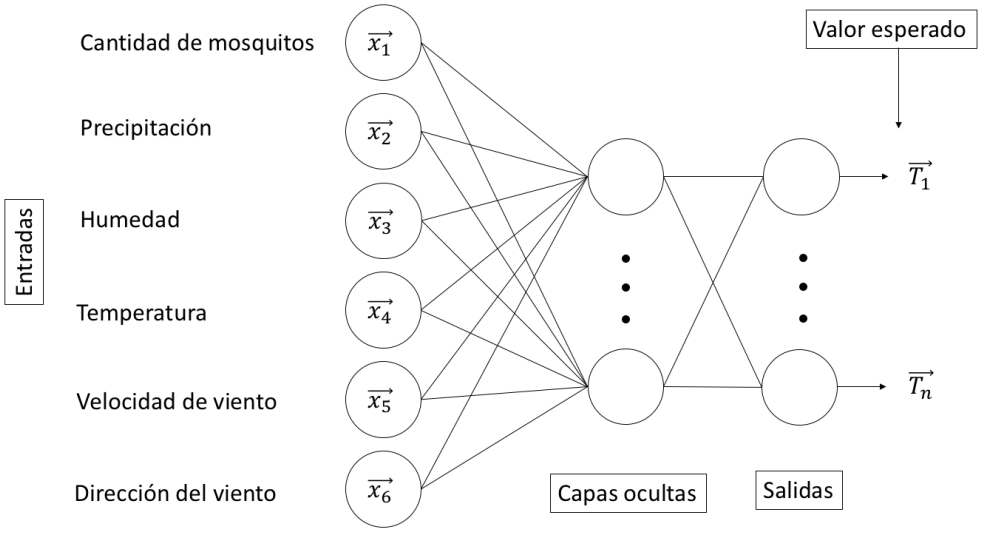


Figura 4.3: Estructura de una red neuronal en la que se tienen 6 datos como entradas.

figura 4.4 se puede apreciar cinco subregiones y cada una de ellas contiene fuentes de reproducción y cajas contadoras de mosquitos. En la subregión uno contiene una caja y dos fuentes, en la subregión dos contiene una caja y una fuente, en la subregión tres contiene dos cajas y dos fuentes, en la subregión cuatro contiene

dos cajas y tres fuentes y en la subregión cinco contiene una caja contadora de mosquitos y una fuente de reproducción de mosquitos respectivamente.

A todas las subregiones se les asigna un identificador para cada caja contadora de mosquitos, para las subregiones que contienen dos cajas, estas comparten el mismo identificador y como solo se tienen cinco regiones esto implica que el valor esperado de cada caja solo tenga cinco elementos. Entonces, como se tienen siete cajas significa que se tienen siete registros de datos respecto a las variables climatológicas y el conteo de mosquitos. Siendo así, el valor esperado de la única caja que se encuentra dentro de la subregión uno corresponde a $T_1 = \{1, 0, 0, 0, 0\}$, como en la subregión dos hay una sola caja significa que el valor esperado correspondiente a la subregión dos corresponde a $T_2 = \{0, 1, 0, 0, 0\}$, para la subregión tres hay un total de dos cajas y su valor esperado para cada caja es el mismo, es decir, $T_3 = T_4 = \{0, 0, 1, 0, 0\}$, para la subregión cuatro hay dos cajas y su valore esperado corresponde a $T_5 = T_6 = \{0, 0, 0, 1, 0\}$ y por último para la subregión cinco hay una sola caja y el valor esperado corresponde es $T_7 = \{0, 0, 0, 0, 1\}$.

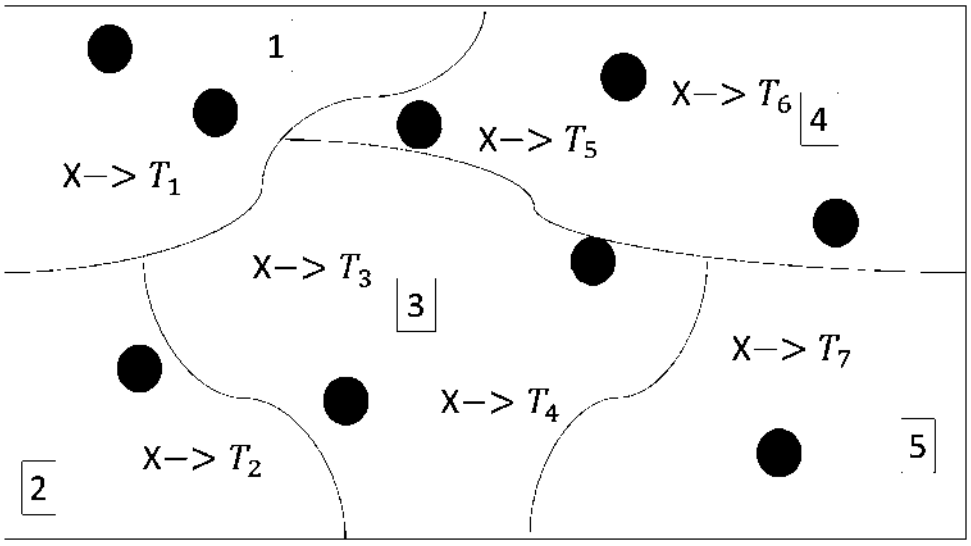


Figura 4.4: Subregiones de la ubicación de las cajas contadoras de mosquitos y las fuentes de reproducción de mosquitos.

En el ejemplo anterior se muestra para un caso particular, pero si se tuvieran N cajas contadoras de mosquitos y M fuentes de reproducción, se aplicaría la misma idea de asignarle un identificador a las cajas y dividir la región en subregiones con el fin de estructurar este problema.

En un principio se planeó construir 50 cajas y distribuirlas en los espacios de la Universidad Michoacana de San Nicolás de Hidlago y así crear varías regiones

dentro del campus. No obstante, solo se construyó una sola caja contadora de mosquitos la cual funciona como prototipo tal y como se describió el capítulo 2. En este caso, para lograr que el modelo funcione (ver figura 4.3), se necesita algo que simule el comportamiento de los mosquitos, es decir, modelarlo de una manera cuantitativa el comportamiento de dicha población. Para esto es necesario contar con un modelo matemático para este sistema.

4.2. Modelo matemático

La ecuaciones diferenciales son aquellas que son capaces de modelar un gran número de sistemas tanto físicos como biológicos, tales como movimiento de fluidos, comportamientos de campos electromagnéticos, comportamiento de población, por mencionar algunos. Así, se puede tener un conjunto de ecuaciones que sirven para describir cierto sistema. En particular las ecuaciones diferenciales parciales (EDPs) asociadas con sistemas que evolucionan suelen, clasificarse en tres categorías: parabólicas, hiperbólicas y elípticas. Por ejemplo, las ecuaciones que modelan los fenómenos vibratorios, como la ecuación de onda, suelen ser hiperbólicas mientras que las ecuaciones que modelan difusión son parabólicas. Existen ecuaciones que sirven para modelar fenómenos tanto químicos como biológicos ya que modelan la aparición de ondas viajeras en sistemas que sufren la influencia de procesos de reacción y difusión.

El problema descrito en la sección 4.1, requiere un tipo de propagación o evolución, entonces el termino *ecuación de evolución* se refiere, en general a una EDP que involucra variables independientes tanto temporales como espaciales de la forma

$$\frac{\partial u(\vec{r},t)}{\partial t} = K\left[u(\vec{r},t)\right], \tag{4.1}$$

en donde el lado izquierdo es simplemente la derivada temporal de primer orden de la variable dependiente u y el lado derecho involucra operadores que actuan sobre u los cuales puede ser lineales o no y en ocasiones , depende de t, $\vec{r}$ y u.

4.2.1. Ecuaciones de tipo reacción-difusión

Difusión. Una de las ecuaciones más importantes que caracterizan las EDPs parabólicas y de evolución es la Ecuación de Difusión que describe la densidad de

fluctuaciones en un material que éste bajo una proceso difusivo en un punto en el espacio y todo tiempo[4]. Dicha ecuación se escribe como

$$\frac{\partial u(\vec{r},t)}{\partial t} = -\nabla \cdot \vec{J}, \tag{4.2}$$

donde $\vec{J}$ es conocido como el flujo difusivo de la densidad del material y es proporcional al gradiente de la densidad, tal como

$$\vec{J} = -D(u(\vec{t},t),\vec{r})\vec{\nabla}u(\vec{r},t), \tag{4.3}$$

siendo D el coeficiente de difusión de u que en general depende de u, $\vec{r}$ y t y el signo menos transporta de densidades altas a bajas, por lo tanto la ecuación de difusión se escribe

$$\frac{\partial u(\vec{r},t)}{\partial t} = \vec{\nabla} \cdot [(D(u(\vec{t},t),\vec{r})\vec{\nabla}u(\vec{r},t))]. \tag{4.4}$$

Si el coeficiente de difusión no depende de la densidad ni de la posición, es decir, D es constante. La ecuación (4.4) se reduce

$$\frac{\partial u(\vec{r},t)}{\partial t} = D\nabla^2 u(\vec{r},t). \tag{4.5}$$

Esta ecuación define un problema de valores iniciales, es decir, si contamos con información de cómo es u a un cierto tiempo inicial t_0 para todo el dominio espacial, entonces la ecuación describe como $u(\vec{r},t)$ evoluciona en el tiempo.

Reacción. Otra manera en la que el material que se esté modelando puede cambiar, es bajo la influencia de procesos de reacción. Esto se refiere a crecimiento, interacción con su entorno o cambios de estado. Si $u(\vec{r},t)$ es la densidad de población de una clase de elementos a un tiempo t, esta población se modela como

$$\frac{\partial u(\vec{r},t)}{\partial t} = f(u(\vec{r},t)). \tag{4.6}$$

con $f(u)$ el término de reacción que puede referirse a crecimiento exponencial, $f(u) = \omega u$, o crecimiento logístico, $f(u) = \omega u(1 - u/k)$, donde ω es la constante de crecimiento de la población y k es la capacidad de carga del entorno, es decir,

la densidad máxima de población que el entorno puede albergar. Si se suman los procesos anteriores y se añade alguna condición de frontera, se obtiene un sistema de reacción-difusión que tiene la forma de una ecuación parabólica con una fuente.

$$\frac{\partial u}{\partial t} = -\nabla \cdot \vec{J} + f(u, \vec{r}, t). \tag{4.7}$$

Un problema de reacción-difusión se puede plantear como un problema de valores iniciales, $u(0, x)$ y valores en la frontera $u(t, 0)$ y $u(t, L)$ cuyo dominio temporal es abierto, $t \geqslant 0$. Un esquema muestra la idea anterior, ver figura 4.5.

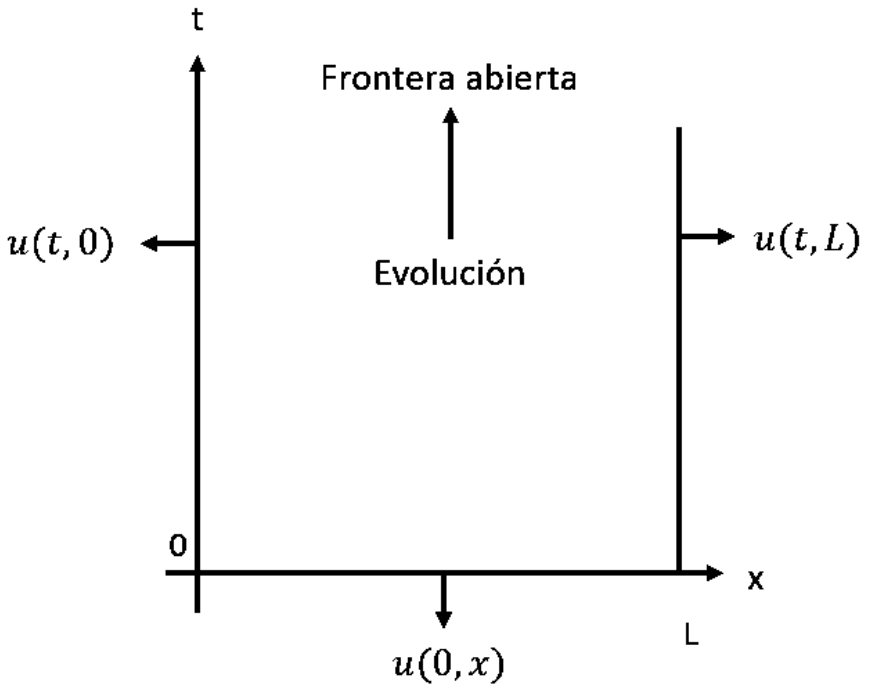


Figura 4.5: Dominio de la solución de un problema de reacción-difusión en un dimensión espacial: $V = [0, L] \subset \mathbb{R}$.

Por último, al hacer pasos algebraicos dentro de las ecuaciones (4.2), (4.3), (4.4) y (4.7). De este modo, la ecuación de *reacción-difusión* es finalmente

$$\frac{\partial u(\vec{r}, t)}{\partial t} = \nabla \cdot (D(u(\vec{r}, t), \vec{r}) \nabla u(\vec{r}, t)) + f(u(\vec{r}, t)). \tag{4.8}$$

En la siguiente sección se muestra la aplicación de la ecuación reacción-difusión, dirigida hacía la propagación de mosquitos que fue construida bajo una cierta región y con condiciones iniciales y de frontera.

4.3. Aplicación de la ecuación reacción-difusión

La ecuación de reacción-difusión, permite simular la propagación de mosquitos bajo ciertas condiciones. En términos generales se toma una región como la que se ve en la figura 4.5. Se pone un punto sobre el espacio y posteriormente se propaga la ecuación (4.8). Luego, en otro lado del espacio se coloca un medidor que contará los los valores propagados en dicho punto. En la figura 4.6 se muestra un esquema de la forma en la que se propaga tal ecuación, aquí el circulo negro representa la fuente de reproducción y la equis el medidor de dicho valor.

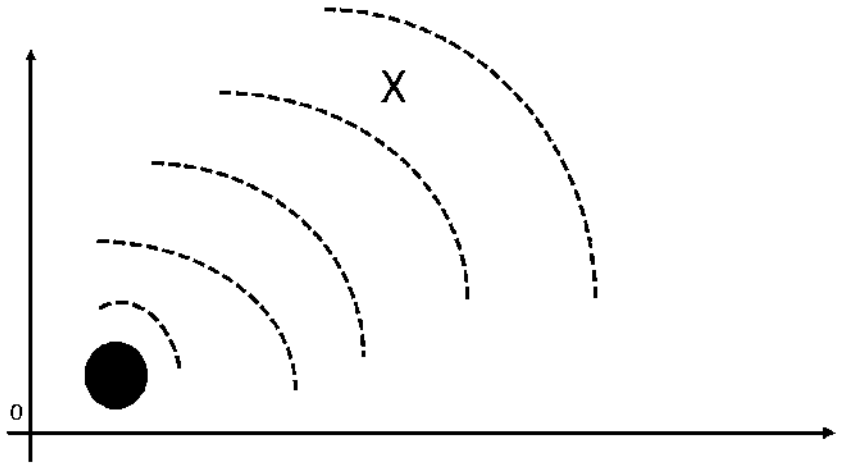


Figura 4.6: Dominio de la solución de un problema de reacción-difusión en un dimensión espacial con una captura de datos.

Esta misma idea se aplicó para más medidores. Dentro del dominio se encuentran distribuidos 9 medidores y éstos representan las cajas contadoras de mosquitos. Luego, se colocó una fuente de reproducción, es decir, se resolvió la ecuación de reacción-difusión en ese punto durante un tiempo t. Consiguiendo así datos para cada caja. Nuevamente se repitió el mismo proceso, solo que ahora se cambio la posición de la fuente de reproducción, y así sucesivamente hasta conseguir 81

simulaciones y con un conjunto de datos para cada caja. En la figura 4.7 se muestra esquemáticamente la distribuidas de las fuentes de reproducción y de las cajas dentro de dicha región. Cada uno de estos círculos negros indica la posición de las fuentes de reproducción para después aplicar la ecuación de reacción-difusión.

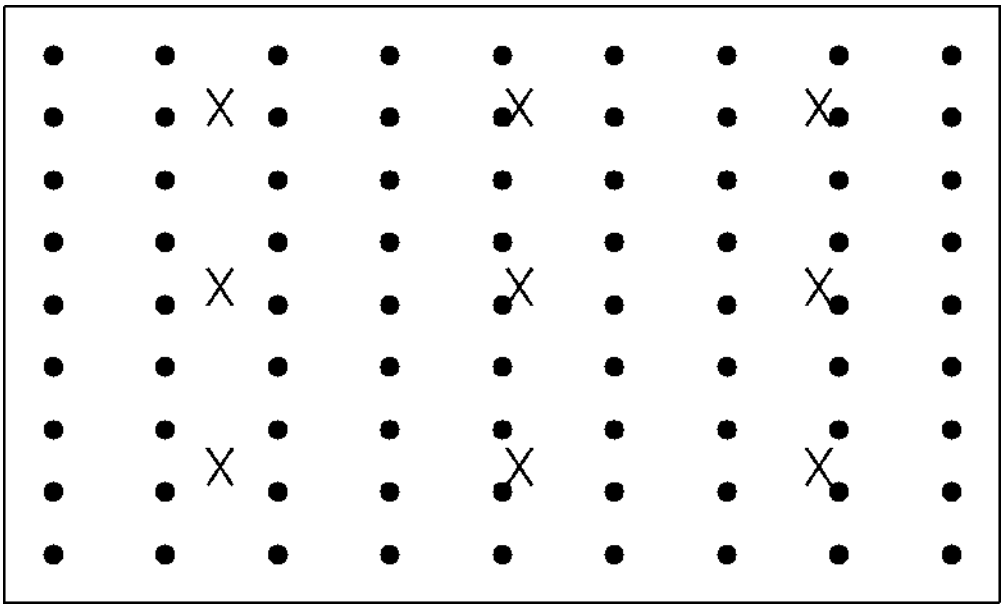


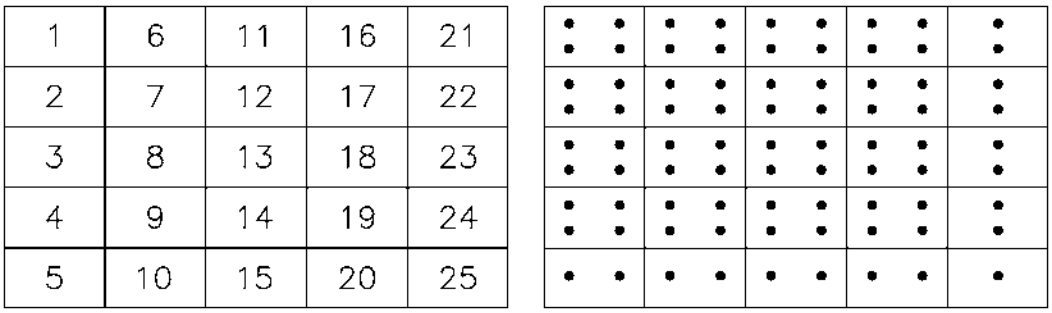
Figura 4.7: Los círculos negros representan las fuentes de reproducción en la que se utilizó la ecuación reacción difusión, además de las cajas contadoras de mosquitos.

Una vez que se tiene el conjunto de datos, el siguiente paso es asignarles un identificador T y una distribución de subregiones, como se había comentado anteriormente en la sección 4.1.

Estructurar este problema, nos llevo a crear tres figuras que ayudan a distinguir la forma en que se organizan los datos los cuales serán ingresados a la red neuronal.

La figura 4.8a indica el número de subregiones dividas, siendo así un total de 25 subregiones. Esto no significa que tengan que ser estrictamente 25 subregiones, pueden ser más o menos subregiones. Recordando que esto ayuda, para poner un identificador T para la red neuronal.

La forma en la que se representó el conjunto de datos de cada una de las fuentes es poniendo un círculo tal y como se muetsra en la figura 4.8b, que indica el acomodo cada una de las fuentes dentro de las 25 subregiones. Se puede observar que 16 subregiones tienen 4 fuentes, 8 subregiones tiene 2 fuentes y por ultimo una subregion tiene una sola fuente de reproducción respectivamente; esta estructura se debe a que solo existen 81 fuentes, entonces al tratar de simetrizar las fuentes



(a) Subregiones creadas. (b) Fuentes en cada subregión.

Figura 4.8: Se muestran dos figuras a) y b), que representan la estructura de de subregiones y la posición de la fuentes de reproducción.

dentro de todas las subregiones con la misma cantidad, no es posible ya que no se no se cuenta con 100 fuentes de reproducción, sin embargo esto no afecta al aprendizaje de la red neuronal, tampoco significa que esta distribución de fuentes tenga que ser así, también puede ser diferente el acomodo de ellas.

1	10	19	28	37	46	55	64	73
2	11	20	29	38	47	56	65	74
3	12	21	30	39	48	57	66	75
4	13	22	31	40	49	58	67	76
5	14	23	32	41	50	59	68	77
6	15	24	33	42	51	60	69	78
7	16	25	34	43	52	61	70	79
8	17	26	35	44	53	62	71	80
9	18	27	36	45	54	63	72	81

Figura 4.9: Valores asignados a cada fuente de reproducción.

Asignarle un identificador o valor esperado T, ayuda bastante al momento de ingresar los datos dentro de la red neuronal. En la figura 4.9 se muestra la forma en la que se les asigna un valor esperado a cada una de las fuentes y en la tabla

Fuentes	Valor esperado (identificador)
$1, 2, 10, 11$	$T_1 = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$3, 4, 12, 13$	$T_2 = \{0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$5, 6, 14, 15$	$T_3 = \{0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$7, 8, 16, 17$	$T_4 = \{0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$9, 18$	$T_5 = \{0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$19, 20, 28, 29$	$T_6 = \{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$21, 22, 30, 31$	$T_7 = \{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$23, 24, 32, 33$	$T_8 = \{0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$25, 26, 34, 35$	$T_9 = \{0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$27, 36$	$T_{10} = \{0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$37, 38, 46, 47$	$T_{11} = \{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$39, 40, 48, 49$	$T_{12} = \{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$41, 42, 50, 51$	$T_{13} = \{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0\}$
$43, 44, 52, 53$	$T_{14} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0\}$
$45, 54$	$T_{15} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0\}$
$55, 56, 64, 65$	$T_{16} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0\}$
$57, 58, 66, 67$	$T_{17} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0\}$
$59, 60, 68, 69$	$T_{18} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0\}$
$61, 62, 70, 71$	$T_{19} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0\}$
$63, 72$	$T_{20} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0\}$
$73, 74$	$T_{21} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0\}$
$75, 76$	$T_{22} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0\}$
$77, 78$	$T_{23} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0\}$
$79, 80$	$T_{24} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0\}$
81	$T_{25} = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$

Tabla 4.1: Tabla de valores esperados ligados a cada una de las subregiones en las que se encuentra un cierto número de fuentes de reproducción.

4.1 se muestra los valores esperados de cada una de las fuentes de reproducción. Recordemos que cada fuente de reproducción cuenta con un conjunto de datos registrados por las cajas contadores de mosquitos, por lo general los valores esperados T se reducen a una matriz identidad, pero en este caso se decidió escribir cada el elemento correspondiente de cada fuente con el fin de aclarar mejor esta estructura, sin embargo, la matriz T no tiene las dimensiones de una matriz cuadrada, más bien sus dimensiones corresponden al numero de fuentes y al numero

de subregiones que se tenga, es decir, $T_{81,25}$ con 81 filas correspondientes a las fuentes de reproducción y 25 columnas que corresponden a las subregiones que representan las salidas de la red neuronal, tal y como se describió en la sección 4.1.

Ya se ha estructurado toda la información disponible de una manera que se pueda entender la red neuronal, prácticamente es el mismo proceso que se hizo en el ejemplo de la sección 3.3.3 del capitulo 3. En la siguiente sección se mostraran los resultados.

Nota

Parte del contenido de esta sección (sección 4.2) fue tomado de la tesis hecha por Venecia Chávez Medina que concierne al tema de *Solución numérica de la ecuación reacción-difusión aplicada a tumores cerebrales*. Realizada en el año 2017 y dirigida por él Dr. Francisco Shidartha Guzmán Murillo.

4.4. Desenlace del problema

Una vez definido una estructura general de los datos, es momento de aplicar la red neuronal artificial descrita en el capitulo 3. Esta se puede utilizar directamente con la información dada en la sección anterior.

En general, para entrenar una red neuronal, primero se debe de elegir la cantidad de datos para entrenar, después se elige la cantidad de datos para predecir. Debido a la estructura del problema, se utilizó la mayor cantidad de datos posibles para el entrenamiento, esto equivale a 56 fuentes de reproducción debido al número de subregiones disponibles, de otro modo, se decidió quitar una fuente de reproducción por cada subregion por lo tanto se tienen un total de 25 fuentes que serán utilizadas para predecir la ubicación una vez que se haya entrenado la red neuronal.

La forma en la que se va a entrenar la red neuronal es creando ciertas configuraciones. Estas se catalogan por 3 categorías de entrenamiento: La primera categoría se construyó de manera supervisada, es decir, se eligieron cuales son las fuentes que serán utilizadas para entrenar y cuales para predecir. La segunda categoría cuenta con 5 configuraciones, estas se basan en revolver cada subregión, de ahí tomar al menos una fuente para predecir y el resto para entrenar, de igual

manera se tienen 56 fuentes para entrenar y 25 para predecir. La tercer categoría consiste en mostrar 5 configuraciones hechas de manera aleatoria, en otros términos, simplemente se revolvieron las 81 fuentes y de ahí se tomaron las primeras 56 para entrenar y el resto para predecir.

Se tiene un total de 11 configuraciones, el siguiente paso es saber cuantas veces se va a entrenar la red neuronal, debido a que en un principio se sabe que por cada entrenamiento realizado, el error global irá disminuyendo, entonces se decidió entrenar la red neuronal unas 4000 mil veces, pero esto no significa que estemos limitados a entrenar esa cantidad, pueden ser más o menos entrenamientos. La idea de hacer muchos entrenamientos es que durante su proceso puede surgir algo conocido como sobre-entrenamiento de la red neuronal, esto se tratará más adelante.

En este caso, conforme el entrenamiento evolucionaba, para cada entrenamiento, se hacía una evaluación respecto a la validación y a la predicción, en otras palabras, se le preguntaba a la red neuronal cuantas fuentes reconocía, para aquellas en las que conocía (conjunto de entrenamiento) y las que no conocía (conjunto de predicción). Con esto, permite ver como va evolucionando la red neuronal.

En la figura 4.10 se muestra la configuración de la primera categoría. Se puede observar, que se quitó una fuente de reproducción de cada subregión, como la categoría uno se hizo de manera supervisada, se decidió quitar una fuente de reproducción con la condición de que tiene que ser vecina a las otras subregiones. En este caso, la única fuente de reproducción que se encuentra dentro de la subregión 25 será tomada para predecir.

La parte mas importante de la red neuronal artificial es ver el comportamiento del error global. En este y otros algoritmos numéricos lo ideal es que el error global sea lo mas pequeño posible. Como se dijo en el capitulo 3, el error tendrá que ir disminuyendo siempre conforme el número de entrenamientos avance. En la figura 4.11 se observa el error global de la red neuronal para la configuración de la primera categoría. También se aprecia que le tomó aproximadamente 500 entrenamientos para que la red tenga un error bastante pequeño y a partir de ahí se nota que sigue siendo muy pequeño conforme el entrenamiento se prolonga, dado esto se puede decir que la red neuronal esta entrenada y lista para reconocer patrones similares a los entrenados. Sin embargo, en la misma figura se puede observar que el error no siempre es pequeño, ya que tiene saltos en los que el error se incrementa, uno muy notorio es durante el entrenamiento 3500-4000 en el error se aleja mucho del cero, esto se puede deber a que el gradiente se estancó en un mínimo local hasta que avanzo al siente paso (dado por la constante de aprendizaje) para después salir del mínimo local y continuar en la dirección negativa del gradiente, además

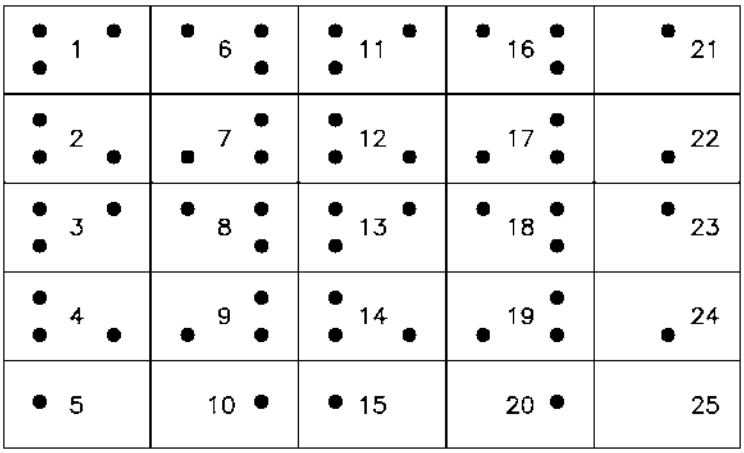


Figura 4.10: Configuración supervisada donde los círculos negros representan las fuentes para entrenar y el resto para predecir.

el error más pequeño obtenido depende de los límites computacionales, es decir, una computadora no existe el cero absoluto. Otro problema es que no es posible visualizarlo ya que el gradiente se esta calculando en muchas dimensiones.

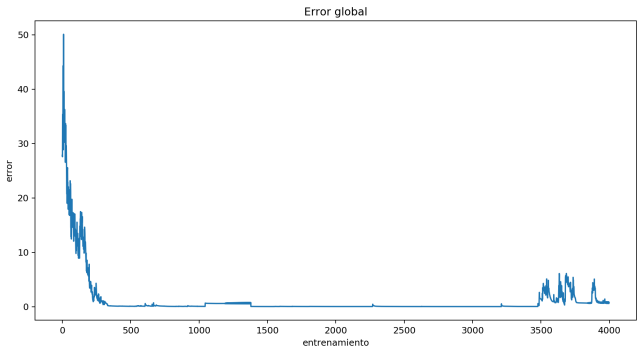


Figura 4.11: Se puede ver el error global de la red neuronal.

Es un hecho que si a la red neuronal se le preguntan cosas que conoce es muy probable que acierte. El la figura 4.12 se muestra una gráfica porcentual que permite validar el conjunto de entradas. Entonces conforme el entrenamiento avance, la red neuronal será capaz de reconocer las fuentes de entrenamiento hasta que llegue el momento de acertar en un 100 % sobre los valores de entrenamiento, es decir, es 100 % significa que reconoció las 56 fuentes de entrenamiento. Esto ocurre aproximadamente durante el entrenamiento número 500 y viendo la gráfica del error global se observa que en ese entrenamiento el error es bastante pequeño.

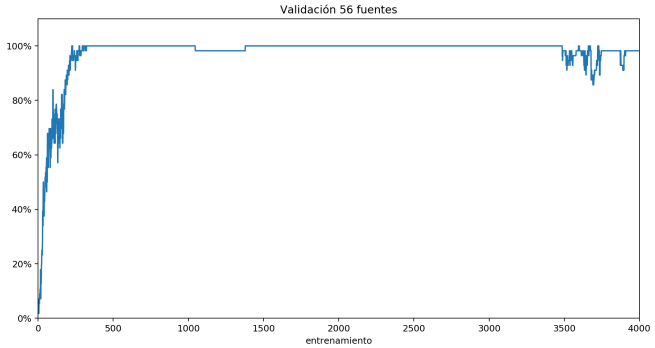


Figura 4.12: Se muestra la validación de la valores de entrenamiento.

Por último resta mostrar la efectividad de la valores distintos a los entrenados. Entonces, para las 25 fuentes restantes que dejaron para predecir, en la Figura 4.13 se muestra una gráfica porcentual respecto a la predicción de valores. De igual manera que en la calculó del error global y la validación de valores, también se evalúa la predicción conforme el entrenamiento avance. Sin embargo, la predicción de valores nunca llega a ser un 100 % por distinta razones y una de ellas es el valor desconocido de la región 25 a la cual no se tiene ninguna información para la cual reconocer tal región, así aunque el error global sea tan pequeño o que se acierte en un 100 % de validación de entrenamiento, la predicción nunca será totalmente satisfactoria. El *porcentaje de eficiencia* es aquel que arrojó en cada entrenamiento, es decir, es el porcentaje reconocido de las 25 fuentes desconocidas para la red neuronal, entonce dicho porcentaje máximo obtenido es de un 76 % y en promedio se tiene un 63.25 % a partir de que la red neuronal presenta una estabilidad y esta ocurre aproximadamente en el entrenamiento número 500. Esto es un porcentaje razonable y valido para nuestros objetivos que es encontrar la fuente de reproducción.

Es importante saber que es lo que paso con el resto del porcentaje de eficiencia que no logró predecir la red neuronal, a este se le llamó como *porcentaje ineficiente*. Como la eficiencia de predicción esta basada en el número total de aciertos, aquellos que no fueron acertados no significa que no sea válidos en su totalidad, ya que es probable que la fuente de reproducción no acertada este muy cerca a la subregión correspondiente.

La red neuronal siempre va a determinar una fuente de reproducción, en este caso es la más parecida respecto a las fuentes ajustadas por la red neuronal. En la figura 4.14 se muestra con un círculo verde a los cuales la red neuronal acertó, aquellos que aparecen con un círculo rojo son a los que se aproximo en un radio

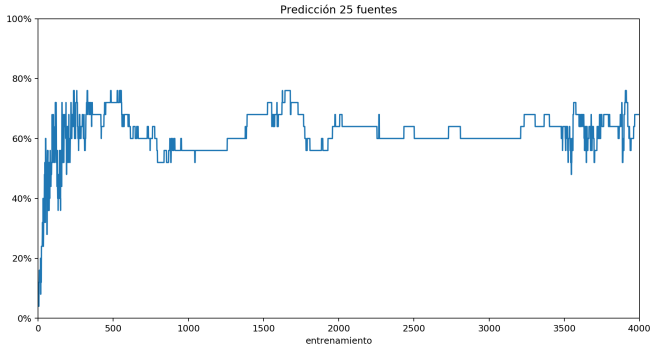


Figura 4.13: Se ve la predicción de valores desconocidos para red neuronal.

de una subregión y por último los círculos azules son aquellos que no se acerco en un radio de dos subregiones. Esto ocurrió, en el entrenamiento 4000 de la red neuronal y en ese momento el porcentaje de eficiencia es de un 72 %. Analizando la misma figura, se observa que la fuente de reproducción de la subregión número 25 esta entre los valores no reconocidos, lo cual es algo que ya se esperaba debido a la falta de información de entrenamiento de dicha subregión, el valor reconocido de la subregión 25 es la subregión 14. Asimismo la fuente de reproducción de la subregión 21 corresponde a la subregión 23.

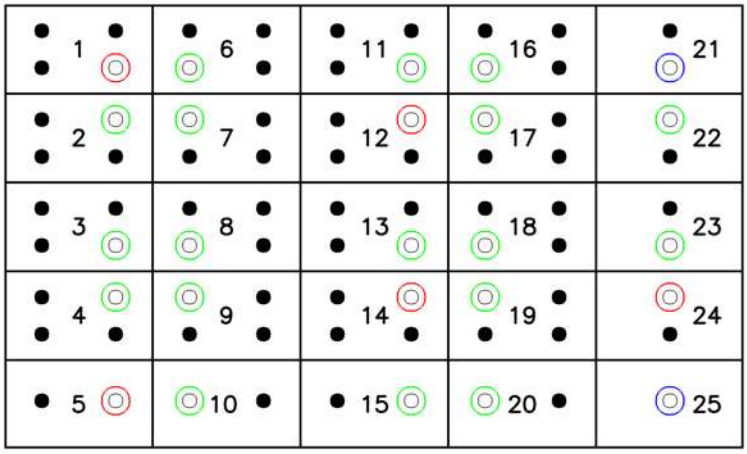
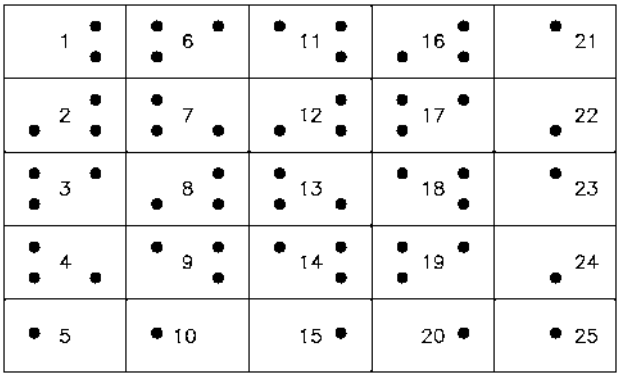


Figura 4.14: Fuentes que fueron acertadas por la red neuronal, el círculo verde significa una predicción correcta, el círculo rojo una aproximación de un radio de una subregión a la predicción correcta y el círculo azul significa valor ni siquiera aproximado en un radio de dos subregiones.

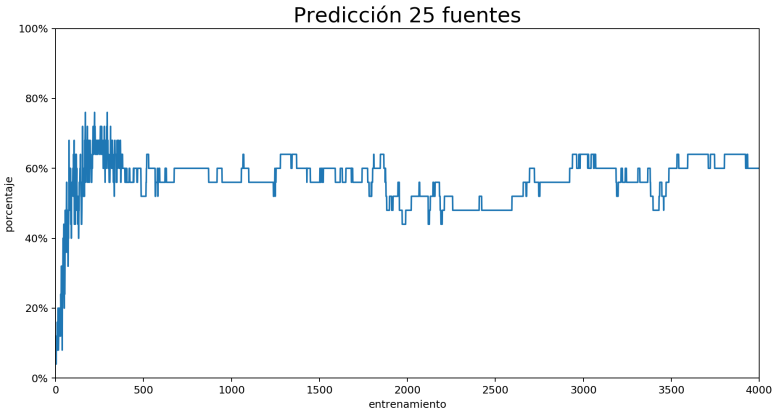
Enseguida se muestra los resultados de las categorías dos y tres. En esta ocasión

se omite el resultado mostrado en al figura 4.14 ya que la intención es mostrar lo que pasaba con el porcentaje ineficiente. En las siguientes figuras se muestra los resultados restantes de las 11 configuraciones hechas. En las figuras 4.15 a 4.19 se muestran los resultados de la categoría dos y de las figuras 4.20 a 4.24 se indican los resultados de la categoría tres. Si se observan ambas categorías, en la parte de la gráfica del error global se puede notar una cierta discrepancia de la categoría dos a la tres, es decir, pareciera que a la red neuronal le "cuesta" ajustar los pesos de la categoría tres que a la dos, por decirlo así. Se contempla que el error global de la categoría dos presenta estabilidad a partir del entrenamiento número 500, mientras que en la categoría tres revela estabilidad a partir del entrenamiento número 1500. Un punto a observar de dicha gráfica es cuando hay discrepancia en cierto intervalo, también, lo presentan las gráficas de validación y de predicción dentro del mismo intervalo. Otro punto que surge de gráfica del error es que su comportamiento es algo similar a la gráfica de validación. La gráfica de predicción no es tan fácil predecir su comportamiento, a diferencia de la gráfica del error global en la que se sabe que el error tiende a cero conforme el entrenamiento progrese, en tal caso la gráfica de predicción solo nos permite saber que tanto puede determinar cosas desconocidas, al avistar todas las gráficas de predicción de la categoría dos se puede notar que en promedio el porcentaje de eficiencia es de un 72 % y para la categoría tres el promedio es de un 64 % respectivamente.

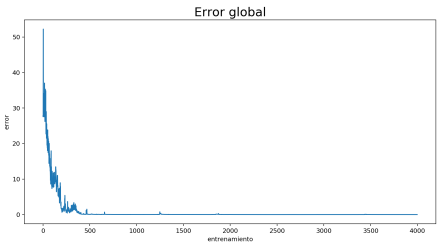
El no limitarnos en crear configuraciones permite conocer desde distintos puntos de vista que no siempre resultarán una mejoría de resultados, el crear tres categorías con diferente reglas fijas de construcción nos permite determinar cuales son las reglas adecuadas para dicho entrenamiento. En base a esto, significa que el campo es demasiado amplio, es decir, cambiar la estructura de la red neuronal, por ejemplo en aumentar $l+1$ neuronas para una capa oculta o incluso incrementar el número de capas ocultas, es evidente que se tendrían mejores o peores resultados, el siguiente capítulo se describe algunas conclusiones y mejoras al sistema que conciernen al trabajo futuro de esta tesis.



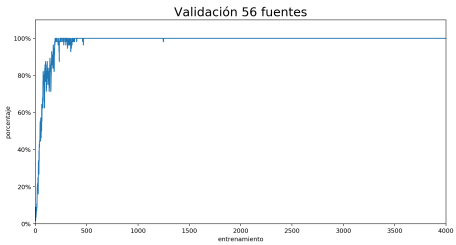
(a) Primer conjunto de datos entrenados supervisados.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 84.0 %.

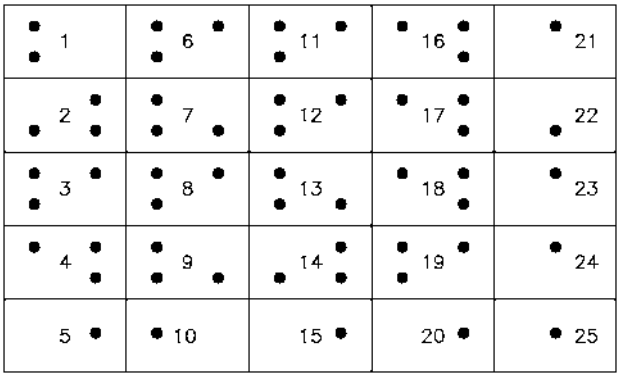


(c) Gráfica del error global igual a 0.8621.

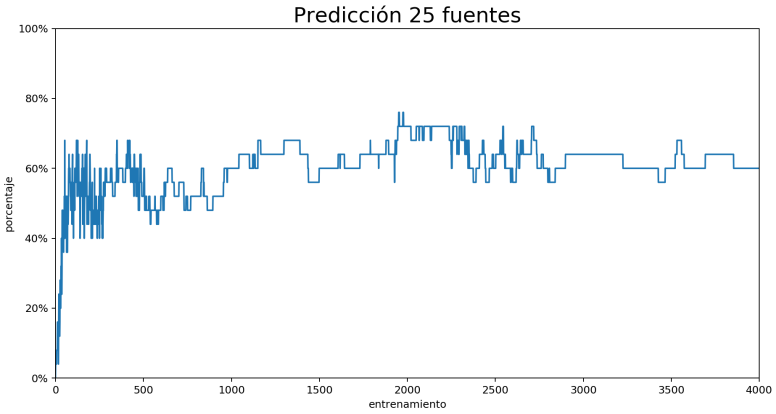


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 98.21 %.

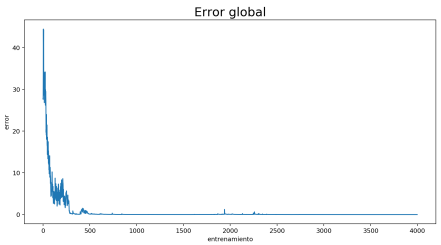
Figura 4.15: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento supervisado, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



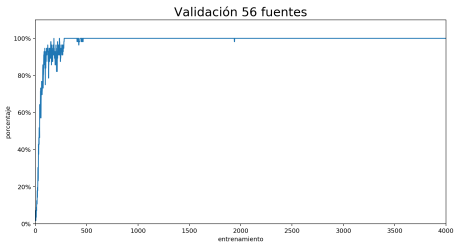
(a) Segundo conjunto de datos entrenados supervisados.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 68.0 %.

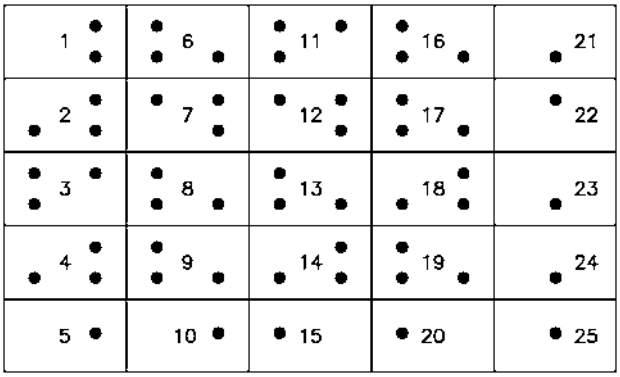


(c) Gráfica del error global igual a 0.0154.

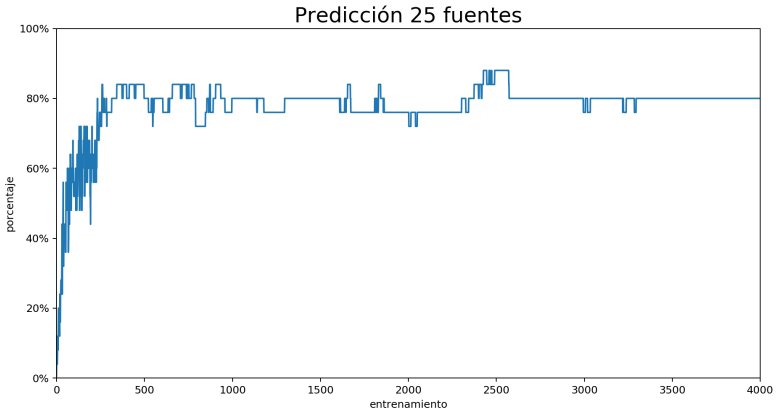


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

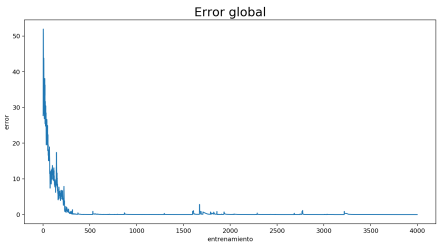
Figura 4.16: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento supervisado, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



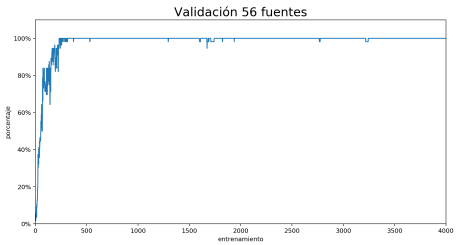
(a) Tercer conjunto de datos entrenados supervisados.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 72.0 %.

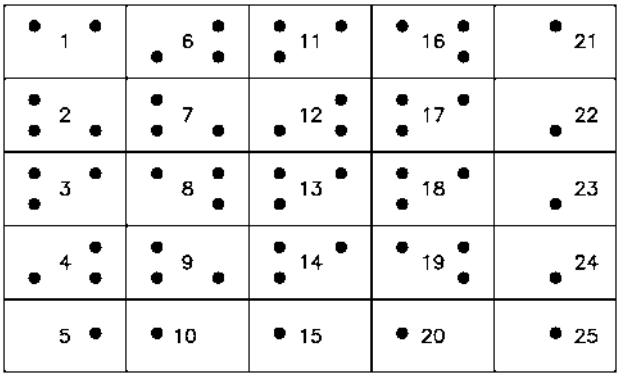


(c) Gráfica del error global igual a 0.0097.

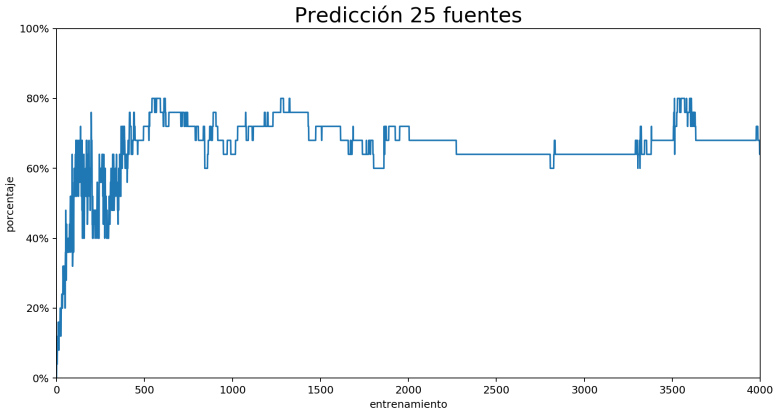


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

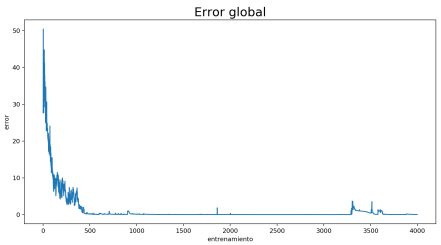
Figura 4.17: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento supervisado, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



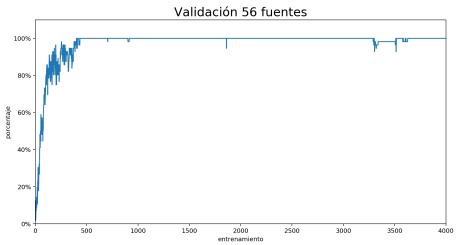
(a) Cuarto conjunto de datos entrenados supervisados.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 76.0 %.

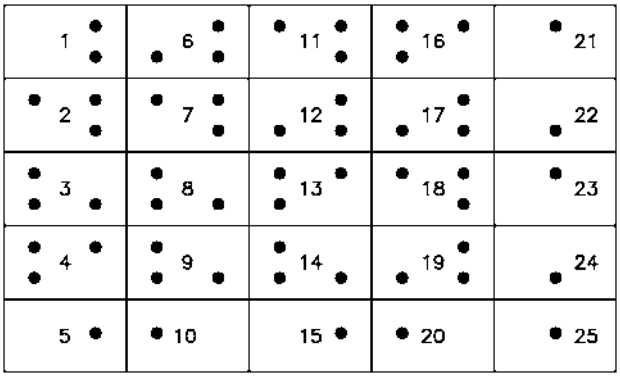


(c) Gráfica del error global igual a 0.0108.

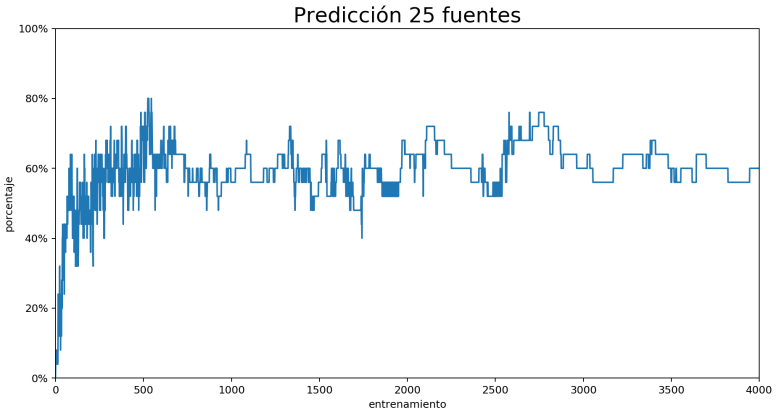


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

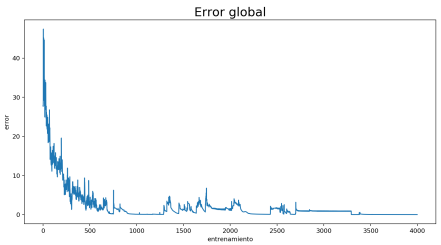
Figura 4.18: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento supervisado, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



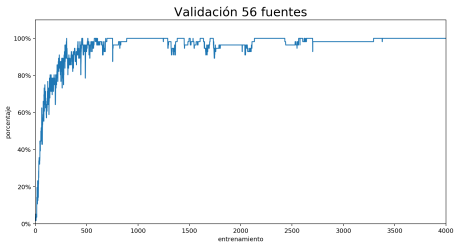
(a) Quinto conjunto de datos entrenados supervisados.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 60.0 %.

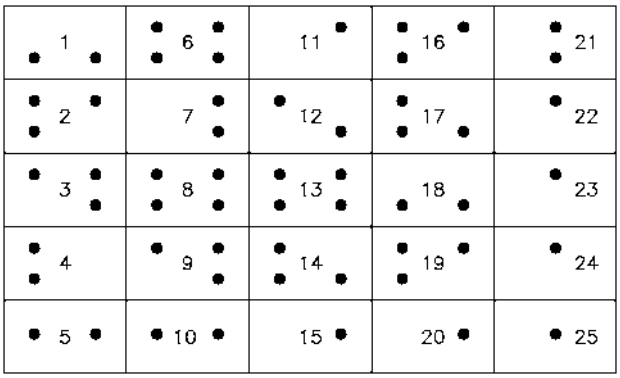


(c) Gráfica del error global igual a 0.1297.

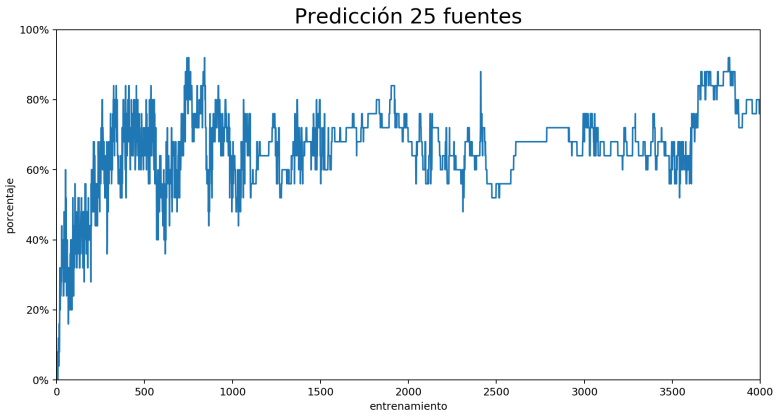


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

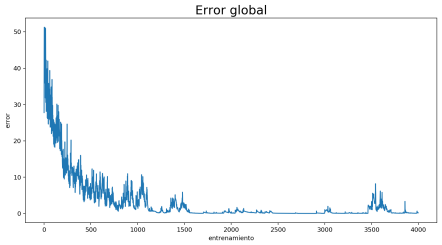
Figura 4.19: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento supervisado, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



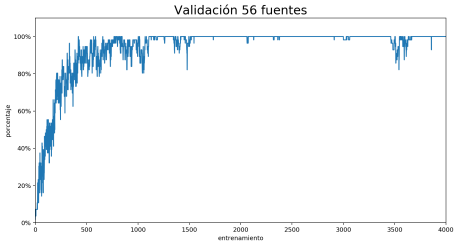
(a) Primer conjunto de datos entrenados de forma aleatoria.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 60.0 %.

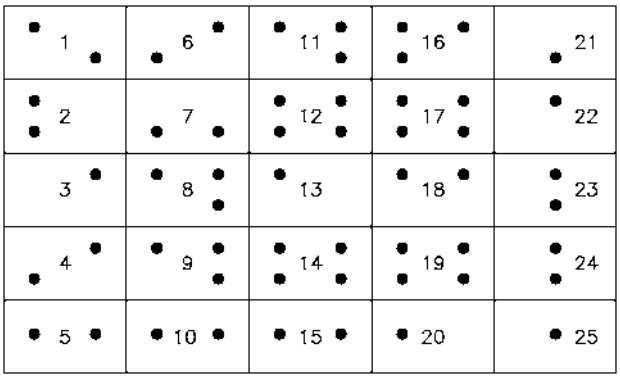


(c) Gráfica del error global igual a 0.0081.

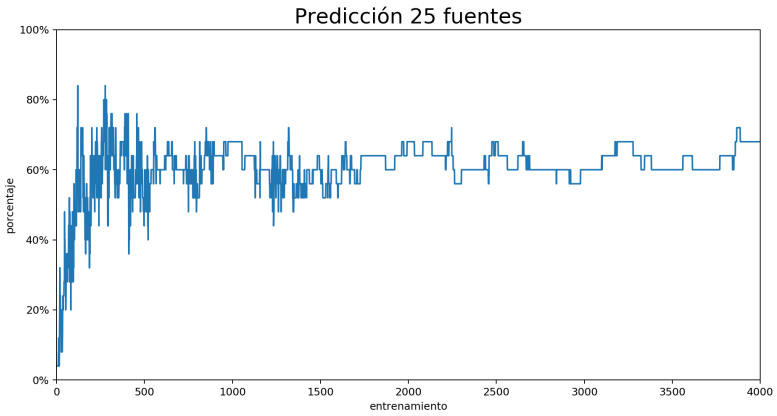


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

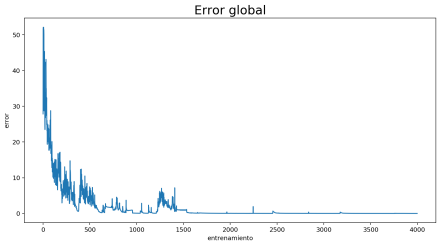
Figura 4.20: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento aleatorio, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



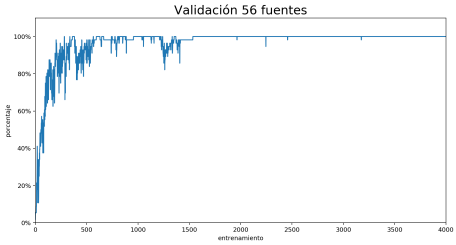
(a) Segundo conjunto de datos entrenados de forma aleatoria.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 60.0 %.

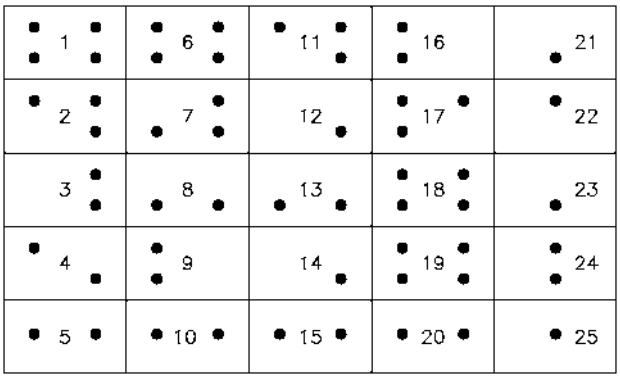


(c) Gráfica del error global igual a 0.0041.

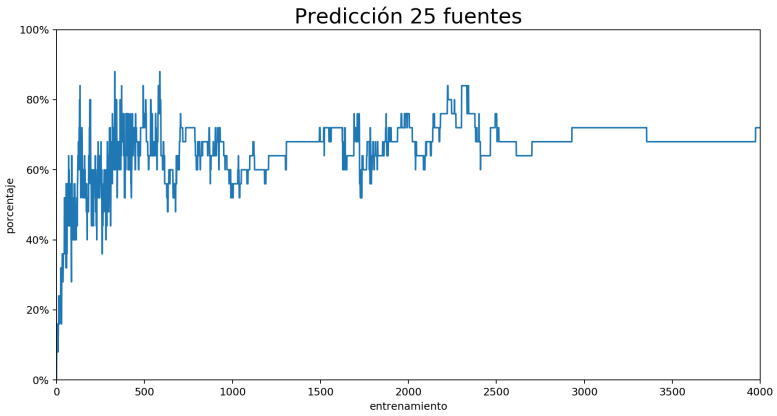


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

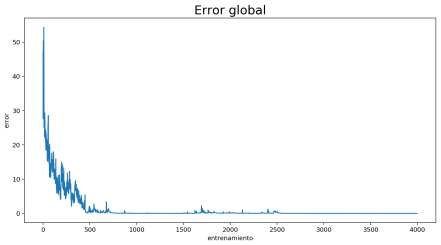
Figura 4.21: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento aleatorio, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



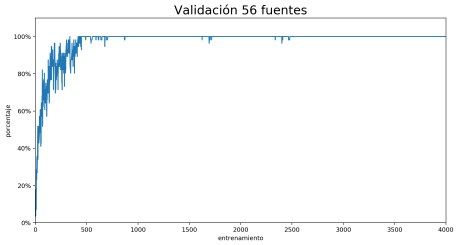
(a) Tercer conjunto de datos entrenados de forma aleatoria.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 80.0 %.

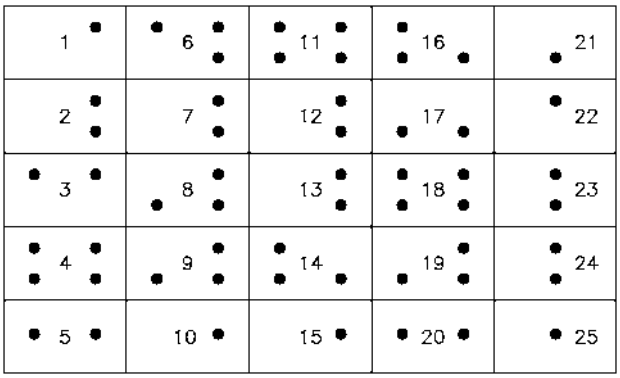


(c) Gráfica del error global igual a 0.0103.

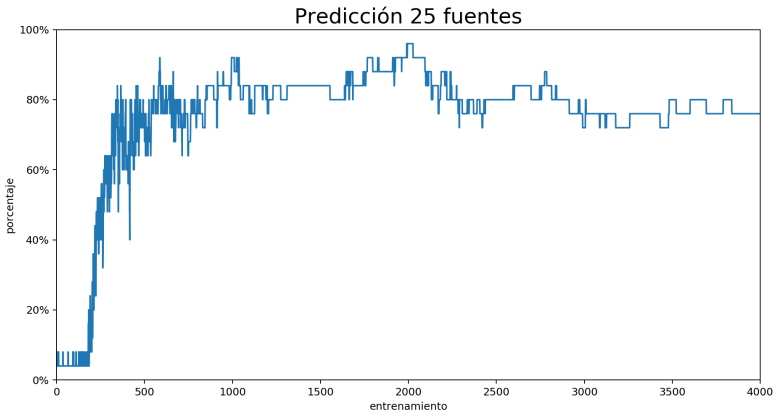


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

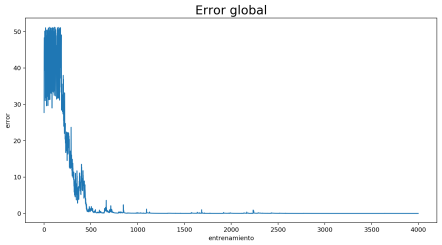
Figura 4.22: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento aleatorio, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



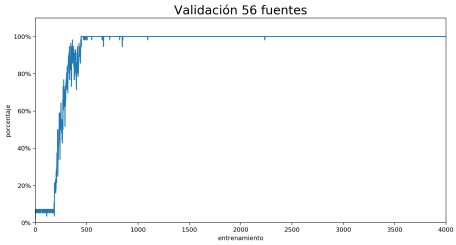
(a) Cuarto conjunto de datos entrenados de forma aleatoria.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 64.0 %.

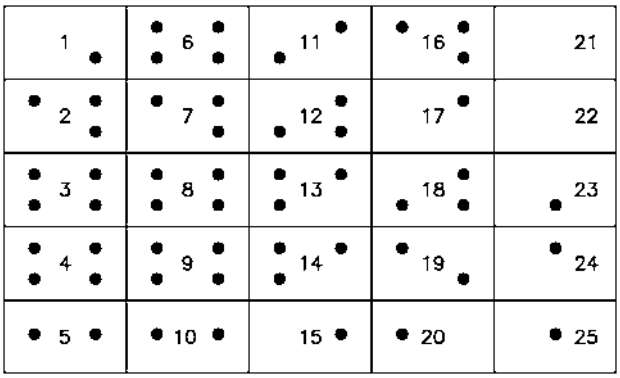


(c) Gráfica del error global igual a 0.0425.

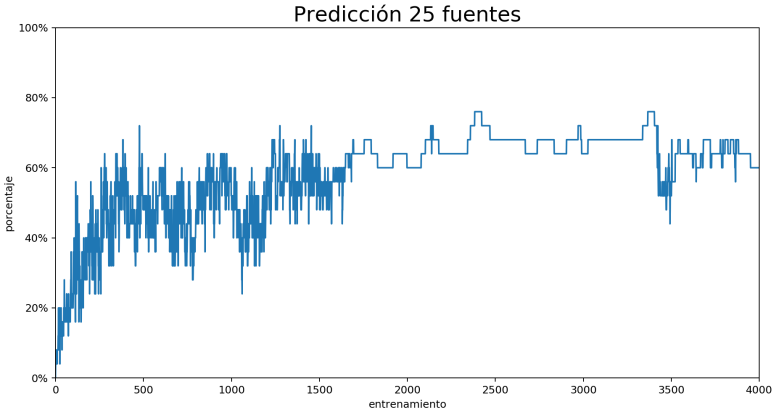


(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

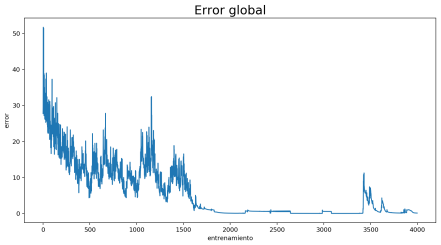
Figura 4.23: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento aleatorio, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.



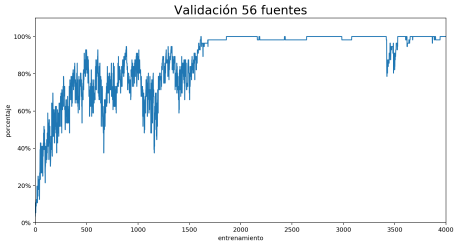
(a) Quinto conjunto de datos entrenados de forma aleatoria.



(b) Predicción de los 25 valores desconocidos, porcentaje de efectividad es de un 60.0 %.



(c) Gráfica del error global igual a 0.0147.



(d) Validación de las 56 fuentes de reproducción de entrenamiento, porcentaje de validación es de un 100.0 %.

Figura 4.24: Se muestran 4 figuras, donde el inciso a) muestra la configuración del entrenamiento aleatorio, en el inciso b) una gráfica de predicción, en el inciso c) una gráfica del error global e inciso d) una gráfica de validación.

Capítulo 5

Conclusiones

El trabajo presentado esta basado en encontrar la fuente de reproducción de mosquitos utilizando un dispositivo construido para medir el número de mosquitos y las variables climatológicas, junto a ello se utilizó inteligencia artificial para buscar los patrones en los datos. A falta de datos obtenidos por las cajas, se simuló el comportamiento de propagación de los mosquitos usando la ecuación de *reacción-difusión* los cuales fueron usados por la red neuronal para estudiar el comportamiento de dichos patrones.

El estudiar el comportamiento de los mosquitos representa un sistema complejo. El uso de de herramienta para las mediciones climatologías y conteo de mosquitos ofrece una buena ventaja ya que si se cuentan con los datos, el objetivo se convierte en encontrar una correlación basada en la comparación de la datos medidos por la caja contadora de mosquitos (los datos reales) y con los realizados en este trabajo, es decir, los datos generados por el modelo matemático.

La ventaja de utilizar el modelo matemático propuesto es que se conoce como surgieron los datos y así poder dar explicaciones de resultados. Como en este caso no se tienen ningún dato real, el modelo matemático tuvo la función de aproximar información a la real y la red neuronal proceso dicha información para poder encontrar la fuente de reproducción.

En el capitulo anterior, se creo un conjunto de configuraciones de encontrar la mejor forma de entrenar. La categoría uno arrojo buenos resultado, pero por obvias razones no es la mejor forma de entrenar a una red neuronal ya que su configuración fue previamente seleccionada. Entre la categoría dos y tres, la que arrojo mejores resultados es la categoría dos debido a su forma de entrenar. Al revisar todos los resultados de dicha categoría se puede observar que aprende

mucho más rápido. Por lo tanto, es posible hacer un promedio de los pesos W de la categoría dos. Esto significa que los resultados mejoren y la red neuronal sea capaz de aumentar el porcentaje de predicción.

Una vez promediado los pesos de la categoría número dos para el entrenamiento número 4000 la predicción arrojo un porcentaje de predicción de un 80 %, es decir, mejoró en predecir dos subregiones. Esto es una mejora importante y suficiente como para determinar las ubicaciones de fuentes de reproducción. Se puede decir que entre más configuraciones se entrenen y posteriormente se haga dicho promedio, el porcentaje de predicción aumentará pero nunca llegara a ser un 100 % al menos que se hagan un número infinito de configuraciones distintas.

En sí, lo que hace la red neuronal es aprenderse el comportamiento de las ecuación *reacción-difusión* bajo ciertas condiciones iniciales y de condiciones de frontera dentro de una región.

Una vez que se tienen resultados favorables es siguiente paso es introducirlos en el mundo real. Vimos que la ecuación de *reacción-difusión* nos ayuda aproximar o simular valores que son posibles que sucedan. Además, recordado la distribución de las fuentes de reproducción dentro de las 25 subregiones, ver figuras 4.8a, 4.8b y 4.9. Nos permite interpretar la configuración de cada subregión como las zonas en donde se ubiquen las cajas contadoras de mosquitos, es decir, las subregiones que tienen 4 fuentes de reproducción se pueden interpretar como regiones húmedas, aquellas en las que posiblemente hay un río o jardines. Aquellas subregiones que poseen 2 fuentes de reproducción se pueden interpretar como regiones semihúmedas o referirse a cajas que se encuentran cercas de subregiones húmedas. Como en este caso, dentro de la subregión 25 hay una sola fuente de reproducción, esta se puede identificar como una zona seca. En dado caso de que se decidierá hacer un cambio dentro de la configuración de las subregiones, se tendría que tomar en cuenta las zonas húmedas por la razón de que los mosquitos tienden a reproducirse en estos sitios.

En trabajos futuros, se pretende mejorar el diseño de este sistema respecto a la caja contadora de mosquitos y al análisis de datos. Por ejemplo, se agregarán algunos sensores a la caja tales como: sensores de comunicación con el fin de recolectar la información sin necesidad de tener que visitar la caja, además de tener comunicación entre las cajas la que nos permita tener una red. Otro aspecto, importante a mejorar es el análisis de datos en tiempo real, como todas las cajas estarán conectadas entre si, pues toda la información será enviada a una caja principal y esta a su vez enviará la información a un servidor en la que se podrá visualizar en una página web la evolución respecto a la captura de mosquitos. Adicionalmente, se construirá un algoritmo que generé reportes automáticos acerca

del estado de los mosquitos y del estado de la propia caja, así cuando haya alguna falla, se enviará una reporte para que sea solucionado lo más pronto posible.

La caja contadora de mosquitos es una herramienta útil para el objetivo del trabajo, sin embargo, se pueden hacer más cosas. Una de ellas es que es importante conocer el nivel de contaminación de la ciudad. Esto es posible, ya que se puede medir el nivel de contaminación de una cierta región utilizando sensores láser y como en la caja cuenta con suficiente espació, pues es posible que sea instalado y esta información puede ser enviada a los departamentos correspondientes para que ellos tomen medidas respecto al problema de la contaminación.

Apéndices

A.1. Primer prototipo

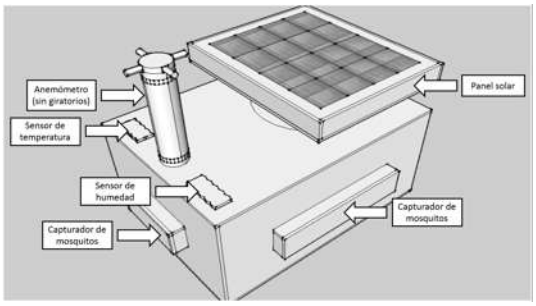
El primer modelo no resulto como se esperaba, aunque pareciera que es un buen modelo, sin embargo al momento de construirlo las estructuras no eran las adecuadas, la ubicación de los sensores, la forma en que colocó el panel solar no permitía configurar sus componentes de una manera fácil, la falta de espacio, la ventilación y lo principal la captura de mosquitos no era la más eficiente.

Aquí se muestran distintas imágenes en distintos ángulos del primer modelo diseñado, figura (A.1).

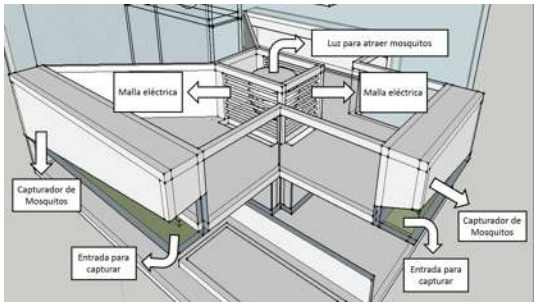
En seguida menciono todos errores de cada imagen y por lo que decidió no construir esta caja contadora de mosquitos.

En la figura (A.1a) se muestra el primer modelo vista externa, sin embargo aparenta ser una caja sencilla en la que soporta todos los componentes.

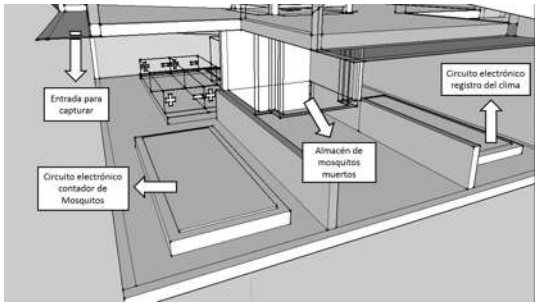
En la figura (A.1b) se muestra el sistema de captura en el que cuenta con 3 entradas para mosquitos. En cada entrada se colocó un sensor de movimiento con el fin de detectar el paso de un mosquito. Para evitar el paso de algún insecto mas grande que un mosquito se agregó una malla con espacio suficiente para la entrada de los mosquitos. Además para atraer a los mosquitos se puso unos leds en el que destellaba y atenuaba la luz ultra violeta de tal forma que esta sea de una intensidad menor a la luz que se encuentra en la malla electrificada. Así en centro de la caja se tenia una luz ultravioleta de mayor intensidad en esta parte los mosquitos morirían a tocar la malla electrificada y así deberían caer en un deposito. Sin embargo, en la forma que se pensó no era la mas adecuada. El primer error detectado fue que los mosquitos no alcanzaba a detectar la luz que estaba a la entrada, por lo que esto resultarían fallidas todas las configuraciones internas. Además, para contar a los mosquitos no fue posible encontrar una forma



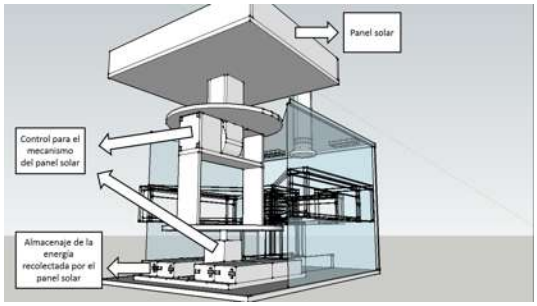
(a) Vista externa, componentes de la caja.



(b) Vista interna, sistema de captura.



(c) Vista interna, colectores y control de la caja.



(d) Vista interna, sistema del panel solar.

Figura A.1: Se muestran distintas imágenes del primer prototipo hecho.

de contar las descargas eléctricas por medio del sistema Arduino.

En la figura (A.1c) se muestra la ubicación del deposito de insectos muertos que caen cuando son electrificados. Aún lado del deposito se muestra las computadoras que controlan todo funcionamiento de la caja contadora de mosquitos. En la parte trasera del deposito están ubicadas las baterías que alimentaría a las tarjetas programables.

En un principio se había pensado construir un sistema en el que un panel solar que se mueva respecto a la posición del sol. En la figura (A1.d) se muestra el sistema de como es que se pretendía construirlo. Sin embargo, cuando se empezó a construir nos dimos cuenta que no funciona para nada ya que no cuenta con los soportes necesarios para poder moverse en los dos grados de libertad que poseía. Lamentablemente, lo primero que se construyó de la caja contadora de mosquitos fue este sistema, como no funcionaba lo propuesto se modifico hasta que se logro hacerlo funcionar. Efectivamente el panel solar seguía la luz del sol pero se tubo otro problema, el consumo de energía era ápice, casi no se obtenía la energía extra como para instalar esta estructura.

Es evidente que gracias al primer sistema propuesta para una caja que cuenta mosquitos, nos sirvió de experiencia para saber que no todo lo que se dibuja en la SketchUp funcionara. Sin embargo, al ser un prototipo es bastante útil utilizar la herramienta.

A.2. Algoritmos

El **Algoritmo 1** representa una serie de pasos que utilizo para el procesamiento de datos. Este fue programado en Fortran 90 y Python a la vez.

Algoritmo 1 Bosquejo del código que se desarrollo este trabajo.

1. Obtención de datos proporcionados por la caja contadora de mosquitos.
    1.1. Información capturada por Python y procesados para su uso.
    1.2. Python genera archivos de texto.
2. Programación de una Red Neuronal en Fortran.
    2.1. Se toman los archivos de texto para ser computables por Fortran.
    2.2. Se entrena la Red Neuronal.
    2.3. Se genera archivos salida de texto computados por la Red Neuronal.
3. Python procesa los archivos salida de texto para su análisis.
    3.1. Generación de gráficas y respectivos componentes.
4. Almacenamiento de información dentro de una base de datos.
5. Envió de respuesta a cada caja con mensaje de éxito.

El **Algoritmo 2** representa una serie de pasos que utilizo para capturar datos de la caja contadora de mosquitos. Este fue programado en Python.

Algoritmo 2 Bosquejo del código que se desarrollo para la caja contadora de mosquitos.

1. Configuración de sensores.
    1.1 Configurando los tiempos de operación de cada sensor.
2. Información capturada y envidad a través de otros sensores.
    2.1 Encriptación de información para ser enviada a la base central.
3. Espera de una respuesta satisfactoria por parte de la base central.

Bibliografía

[1] Sugar-fermenting yeast as an organic source of carbon dioxide to attract the malaria mosquito Anopheles gambiae. Renate C SmallegangeEmail author, Wolfgang H Schmied, Karel J van Roey, Niels O Verhulst, Jeroen Spitzen, Wolfgang R Mukabana and Willem Takken. 2010.

[2] Señales físico químicas involucradas en la búsqueda de hospederos y en la inducción de picadura por mosquitos.José Luis Torres-Estrada, M en C, Dr en C,(1) Mario H Rodríguez, MC, Dr en Filosofía.(2).

[3] Weather variabilty impacts on oviposition dynamics of the southern house mosquitos at intermediate time scales. Bull Entornol Res "011, 101(6):633-641.

[4] Application of an artificial neural network (ANN) model for predicting mosquito abundaces in urban areas. Keun Young Lee, Namilo Chung, Suntae Hwang. 2015.

[5] Predicting Cilex pipiens/restuans population dynamics by interval lagged weather data. Karin Lebi, Katharina Brugger and Franz Rubel. 2013.

[6] Miller, K. D. in Models of Neural Networks, III (eds. Domany, E., van Hemmen, J. L. & Schulten, K.) 55–78 (Springer, New York, 1996).

[7] Joseph T. Belter, MS, BS, Jacob L. Segil, Aaron M. Dollar, PhD, SM, BS. Mechanical design and performance specifications of anthropomorphic prosthetic hands.

[8] Hebb, D. O. The Organization of Behavior: A Neuropsychological Theory (Wiley, New York, 1949).

[9] Mosquitoes of Michigan -Their Biology and Control. Michigan Mosquito Control Organization. 2013.

[10] The Deadliest Animal in the World. Gates, Bill.

[11] Phylogenetic analysis and temporal diversification of mosquitoes. BMC Evolutionary Biology. Reidenbach, K.R.; Cook, S.; Bertone, M.A.; Harbach, R.E.; Wiegmann, B.M.; Besansky, N.J. (2009).

[12] Society, National Geographic. "Mosquitoes, Mosquito Pictures, Mosquito Facts – National Geographic". National Geographic. Retrieved 2016-03-20.

[13] The carnivores, Toxorhynchites . Wing Beats. Jones, C.; Schreiber, E. (1994).

[14] Solución numérica de la ecuación reacción-difusión aplicada a tumores cerebrales, Venecia Chávez Medina, 2017, pag.2-7.

[15] A logical calculus of the ideas immanent in nervous activity. Warren McCulloch.