Mapas Autoorganizativos y el Problema del Agente Viajero, Backpropagation y Reconocimiento de Digitos

Jorge Guevara-Díaz[1]

[1] Maestria en Ciencia de la Computación, Universidad Nacional de Trujillo

jorge.jorjasso@gmail.com

Resumen

En el presente trabajo hago un analisis de dos redes neuronales : Los Mapas Autoorganizativos de Kohonen y su aplicacion en el Problema del Agente Viajero y la BackPropagation y su aplicacion el el reconocimiento de digitos, se presenta ademas un analisis de los algoritmos involucrados

1. Introducción

¿Como es que podemos reconocer una cara conocida en una fotografía?. ¿Como asociamos objetos, hechos que nunca hemos visto antes, con objetos ya conocidos? ¿Podemos hacer que la computadora haga lo mismo?. La solución de diversos problemas de la vida diaria y de la industria a veces requieren que un computador tenga muchas de estas capacidades mostrando un comportamiento inteligente por así decirlo. la capacidad de procesamiento de los computadores actuales es varias veces mas rápida que las células que conforman nuestro sistema neurobiológico , pero aun así no hemos logrado tener computadores que tengan software demasiado inteligente que les permita pensar , tomar desiciones, hacer un reconocimiento general de objetos y hechos en un tiempo razonable, quizás el hecho de que la la arquitectura de las computadoras convencionales es totalmente diferente a la arquitectura del cerebro donde las células del los sistemas neurobiológicos están masivamente interconectadas y muestran un alto grado de paralelismo , esto seria la causa de que el cerebro pueda hacer el reconocimiento de tramas complejas en un tiempo demasiado pequeño

Sin embargo se pueden construir modelos computacionales inspirados en la fisiología de nuestro cerebro que ayuden en la resolución de determinados problemas.

El presente documento detalla dos modelos computacionales inspirados en las redes neuronales cerebrales, obviamente no modelan a detalle el comportamiento biológico, que en sí es muy complejo, mas bien se tratan de modelos computacionales que han mostrado ser muy efectivos en la solución de diversos problemas. Estos modelos computacionales basados en los sistemas neuronales de nuestro cerebro , reciben el nombre de redes neuronales artificiales y han mostrado ser muy útiles en diversos problemas , donde los algoritmos tradicionales no pueden encontrar solución, o la solución de estos problemas requiere una complejidad en tiempo demasiado grande para ser tratado

El documento esta organizado de la siguiente manera. En la sección 2 se describe una breve introducción a las redes neuronales artificiales partiendo del modelo biológico de una red neuronal, la sección 3 describe los mapas autoorganizativos, en la sección 4. Se describe la aplicación de los mapas autoorganizativos en la solución del problema del agente viajero, en la sección 5 se describe la red neuronal backpropagation, en la sección 6 se muestra una aplicación de la red neuronal de backpropagation en el reconocimiento de dígitos, y finalmente en la sección 7 se encuentran las conclusiones del presente trabajo

2. Redes Neuronales Artificiales

Las neuronas son células nerviosas y constituyen uno de los elementos fundamentales del sistema nervioso y una de sus funciones es el transmisión del flujo nervioso. Entre sus partes tenemos el axon, las dendritas, y el cuerpo de la célula , el axon está rodeado de una membrana que recibe le nombre de vaina de mielina, y los nodos de Ranvier que son como puntos de corte para la vaina de mielina a través del axon, la unión sináptica se llama a la unión del axon de las neuronas con otras neuronas

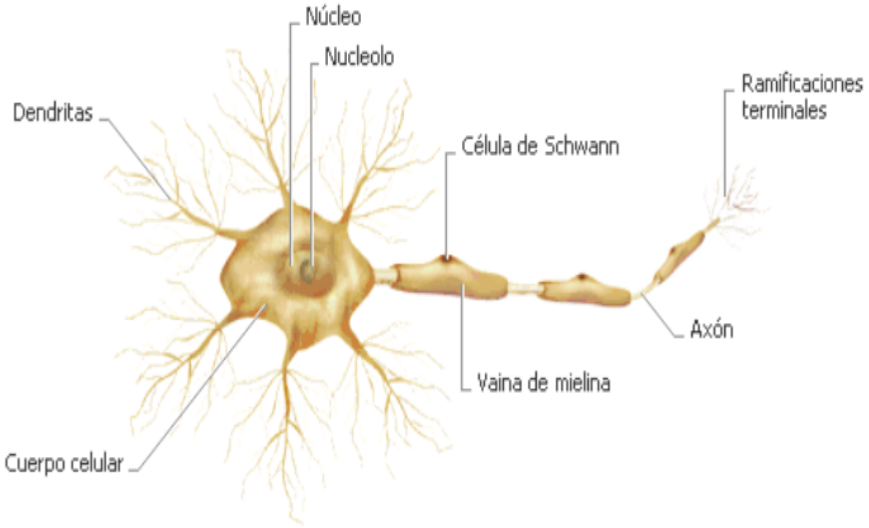

Figura 1: Una red neuronal

En las neuronas tiene una membrana celular que separa el fluido intersticial del plasma celular, esta membrana se comporta de una manera permeable para ciertas especies iónicas y mantiene una diferencia de potencial entre el fluido que está al interior de la célula y el fluido que está fuera de la célula, entre las especies iónicas están los iones de sodio, de potasio, de cloruro y diversos iones orgánicos

Lo que pasa es interesante pues hay una difusión de iones por la membrana , pero los iones que son demasiado grandes no se pueden difundir es decir no pueden salir de la célula, entonces la carga negativa de estos iones no permite que algunos iones de cloro Cl^- que también son negativos puedan entrar a la célula , en consecuencia habrá más iones Cl^- fuera de la célula que dentro de ella, luego también se da el hecho de que va a ver mayor concentración de iones de potasio K^+ dentro de la célula y una concentración mas alta de iones de sodio fuera de la célula. El resultado se tiene como mas iones de sodio y cloro fuera de la célula, y más iones orgánicos y de potasio dentro de ella, produciendo una diferencia de potencial de unos 70 a 100 mV, siendo el potencial negativo el potencial del fluido intra celular este potencial se llama $potencial\ de\ reposo$, las entradas provenientes de otras neuronas pueden reducir o aumentar la diferencia de potencial que existe, si el caso es que se reduce la diferencia de potencial es que esta ocurriendo un proceso exitatorio y por consiguiente una despolarización, esta despolarización resultante en el montículo de axon hace que la membrana celular altere su permeabilidad para los iones de sodio Na^+, el resultado es que hay un fuerte flujo de entrada de los iones de

sodio Na^+ en la célula y despolarizan aún mas a la célula, todo este proceso desencadena al potencial de acción que luego será transmitido por el axon de la célula nerviosa que es el resultado de una serie de despolarizaciones que tienen lugar en los nodos de Ranvier, luego en la unión de dos neuronas vecinas ocurre una actividad llamada sinapsis que es la liberación de sustancias neurotransmisoras por parte de la célula presináptica para ser absorbidas por la célula postsináptica cuando el potencial de acción llega a la membrana presináptica, todo esto puede provocar entrada de iones positivos que tiendan a despolarizar la neurona convirtiéndose en elementos exitatorios, o de iones negativos que tiendan a polarizar a la neurona convirtiéndose en elementos inhibitorios que van a provocar el la célula postsináptica se genere o no un potencial de acción

Para construir el modelo computacional de estas redes se puede establecer una cierta analogía Se puede decir que una Red Neuronal Artificial RNA , es una coleccion de procesadores paralelos conectados entre sí en forma de un grafo dirigido, organizados de tal modo que la estructura de la red sea adecuada para el problema que se esté considerando [Fre91] se puede ver cada unidad de procesamiento como un nodo, y las conexiones que hay entre unidades de procesamiento como arcos. y estas conexiones pueden ser exitatorias o inhibitorias

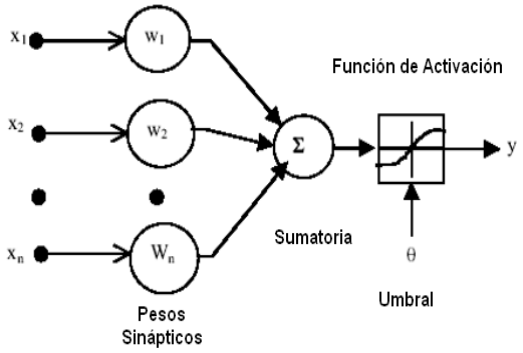

Figura 2: Modelo computacional de una red neuronal

Una neurona puede verse como una unidad de procesamiento que recibe entradas de neuronas vecinas y tiene una única salida que a la vez puede ser entrada a otra unidad de procesamiento o neurona, cada conexión entre neuronas tiene asociada una magnitud llamada peso, que puede ser visto análogamente como la intensidad de conexiones sinápticas entre neuronas biológicas, las neuronas artificiales también tienen un valor de entrada total que típicamente se calcula sumando las entradas ponderadas de todas las $neuronas_j$ hacia la $neurona_i actual$

$$EntradaNeurona_i = \sum_{j=1}^{n} x_j w_{ij} \tag{1}$$

Se puede modelar las excitaciones o inhibiciones haciendo tener un signo negativo a los pesos por ejemplo, luego que la entrada total de neuronas se tiene que encontrar un valor de activación análogamente que en el modelo biológico que se propague por el axon hacia otras neuronas. podemos escribir el valor de activación en la $neurona_i$ de esta manera:

$$valorActivacion_t = F_i(valorActivacion_i(t-1), EntradaNeurona_i(t)) \tag{2}$$

El valor de activación depende del valor de activación anterior y de la entrada neta, sin embargo en la mayoría de los casos el valor de la activación y de la entrada neta son iguales, y

se suelen usar los dos términos para referirse a lo mismo, luego calculado el valor de activación podemos calcular el valor de salida de la $neurona_i$ por medio de una función de salida

$$x_i = f_i(valorActivacion_i) \tag{3}$$

Sin embargo como el valor de activación para los casos que vamos a tratar para este documento es igual que la entrada total de la $neurona_i$ entonces:

$$x_i = f_i(EntradaNeurona_i) \tag{4}$$

La función de salida puede tomar un amplio rango de posibilidades algunas de las cuales son las siguientes:

$$f(u) == \begin{cases} 0 & \text{si } u < 0 \\ 1 & si \geq 0 \end{cases} \tag{5}$$

$$f(u) == \begin{cases} -1 & \text{si } u < -1 \\ u & -1 \leq u \leq 1 \\ 1 & si \geq 1 \end{cases} \tag{6}$$

$$f_u = \frac{1}{1 + e^{-au}}. \quad 0 \leq f(u) \leq 1 \tag{7}$$

$$f_u = \frac{e^{\alpha u - e^{-\alpha u}}}{e^{\alpha u + e^{-\alpha u}}} \tag{8}$$

3. Mapas Autoorganizativos

A continuación se hablará de un tipo de red neuronal basada en el hecho de que nuestras neuronas tienden a organizarse en determinados grupos con respecto a otros , por ejemplo neuronas cercanas entre si responden a frecuencias de sonidos similares según una sucesión ordenada de tonos, luego también se observa que en la corteza cerebral existe una correspondencia entre las zonas cerebrales y las partes del cuerpo que están controlan,por ejemplo área del lenguaje, área de percepción visual, etc muchas de estas regiones vienen ya determinadas por nuestros genes, pero muchas otras se forman en un proceso de aprendizaje, es decir las neuronas vecinas participan en un proceso determinado de aprendizaje

El modelo computacional fué descrito por kohonen [Koh81] y algunas ventajas de este tipo de redes es que son redes de aprendizaje no supervisado , pues no se necesita tener valores esperados , ni a partir de los valores obtenidos empezar ajustar los pesos , también son indiferentes al orden de entrada de los datos El modelo consta de dos capas de neuronas, una capa de entrada y una capa de competición, las activaciones de los elementos de procesamiento están dado por:

$$y_i = -r_i(y_i) + EntradaNeurona_i + \sum_{i=1}^{n} z_{ij} y_j \tag{9}$$

Esta ecuación significa que la activación de una neurona está en función de un término de perdidas $-r_i(y_i)$, más la entrada total hacia la neurona, y finalmente se le suma las interacciones laterales de las unidades , que en teoría abarcan todas las neuronas de la capa de competición ,

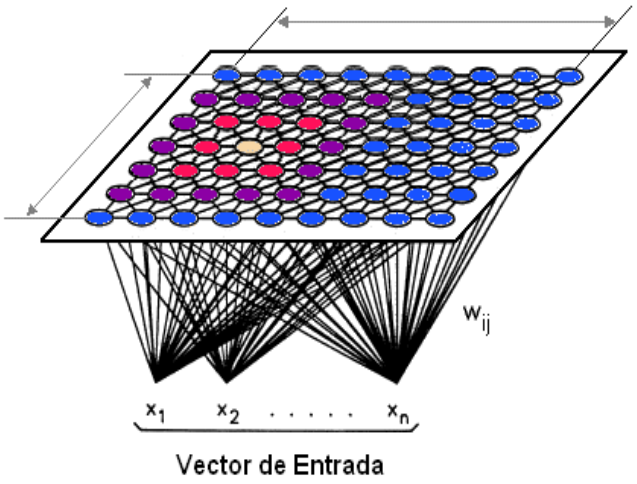

Figura 3: Red neuronal de kohonen o SOM

pero solamente mostraran actividad las neuronas que estén en un radio de vecindad cercano a la neurona ganadora, es decir las neuronas vecinas a las ganadoras compartirán la victoria con las neuronas ganadoras

3.1. Algoritmo de aprendizaje de los Mapas Autoorganizativos

Para la capa de competición se determinará una neurona ganadora, para esto se basará en la similitud que existe entre los datos de entrada y los pesos asociados a la capa de competición de la neurona, los pesos inicialmente pueden ser valores aleatorios, una medida de similitud puede ser la distancia euclidiana:

$$\|x - w_c\| = \min_i(\|x - w_i\|) \tag{10}$$

luego durante el proceso de entrenamiento , las neuronas que estén en un radio cercano a la de la neurona ganadora , presentarán actividad positiva, luego el proceso de aprendizaje , constará en ajustar los pesos de tal modo que se asemejen al vector presentado

$$w_i(t+1) == \begin{cases} w_i(t) + \alpha(t)h(\mid i - g \mid)(x - w_i(t)) & \text{si } i \epsilon R \\ 0 & si \geq 0 \end{cases} \tag{11}$$

La actualización de pesos se da para la neurona ganadora y para las neuronas vecinas en una vecindad R y que según las neuronas estén mas alejadas o mas cerca de la neurona ganadora, aprenderán en mas o menos medida, es decir la actualización de sus pesos depende de la función h. El factor α es una función que nos da la taza de aprendizaje que va disminuyendo conforme avanza el tiempo del entrenamiento, el radio R también disminuye conforme avanza el tiempo

Existen muchas funciones h de vecindad , generalmente se usa la función gaussiana

$$h(\mid i - g \mid, t) = e^{-\frac{(\mid i-g \mid)^2}{2R(t)^2}} \tag{12}$$

El proceso de aprendizaje se puede observar como sigue: las neuronas están ubicadas en un espacio con sus respectivas posiciones en el eje de coordenadas (u, v) , luego a cada neurona se

le asocia un peso inicial que puede ser un valor aleatorio (W_u, W_v) , es decir existe un espacio de posiciones y un espacio de pesos que se inician con valores aleatorios

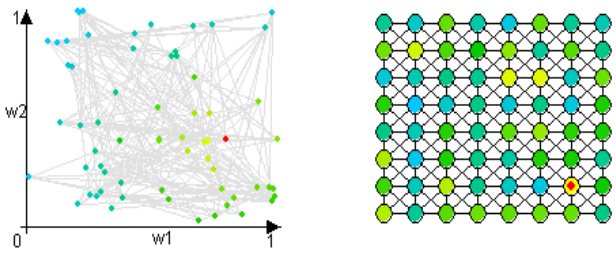

Figura 4: Espacio de el vector de pesos y espacio fisico las posiciones de las neuronas

Luego las neuronas según continua el proceso de aprendizaje éstas aprenden regulando sus respectivos pesos asemejandolos a la trama de entrada

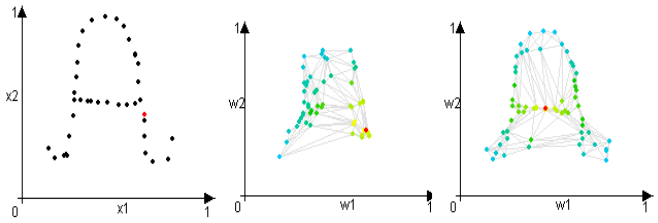

Figura 5: Vector de entrada, espacio de pesos tras algunas iteraciones, espacio de pesos final

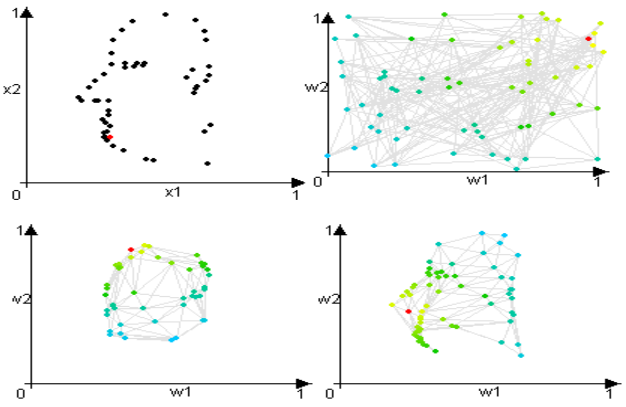

Figura 6: Vector de entrada y espacio de pesos

Al finalizar el proceso de aprendizaje el espacio de pesos tendrá una configuración parecida al vector de entrada

El algoritmo en seudocódigo sería

Algoritmo 1 Aprendizaje Kohonen

Requiere: inicializar pesos $w_{i,j}$ para cada neurona en la capa de competición
Mientras los pesos $w_{i,j}$ no varíen significativamente **Hacer**
Para cada patrón de entrada **Hacer**
presentar el patrón a la red
obtener la neurona ganadora
actualizar pesos de la neurona ganadora y de sus vecinos
Fin Para
actualizar tasa de aprendizaje y radio de vecindad
Fin Mientras

4. Aplicación de los Mapas Autoorganizativos en la solución del problema del Agente viajero

Las redes neuronales han mostrado ser muy potentes en la solución de problemas que tienen como única manera de solución algoritmos aproximados, el problema del agente viajero es un tipo de ellos, este problema consiste en hallar el ciclo de hamilton de menor coste en un grafo

El modelo de esta red neuronal para el presente trabajo es el siguiente:

Se tiene dos capas de neuronas , una capa de entrada de dos neuronas para las ciudades, cada una de estas dos neuronas recibirá las coordenadas de posición de cada ciudad , es decir la posición del nodo en el plano (coordenadas x , y) , la segunda capa tendrá inicialmente muchas neuronas. Cada neurona de la red neuronal, en la capa de competición tendrá asociado dos pesos un peso wx y uno wy que serán inicializados en forma aleatoria, luego la idea principal es que tras cierto estimulo enviado desde la capa de entrada con sus dos neuronas , este estimulo se propague por todas las neuronas de la capa de competición , y se encuentre una neurona ganadora, la cual se obtiene mediante una medida de distancia euclidiana entre la neurona ingresada y las demás neuronas de la capa de competición

$$\|x - w_c\| = \min_i(\sqrt{\sum_{k=1}^{m}(x - w_i)^2}) \tag{13}$$

Una vez obtenida la neurona ganadora aprende con sus neuronas vecinas

$$w_i(t+1) = \begin{cases} w_i(t) + \alpha(t)h(\mid i - g \mid)(x - w_i(t)) & \text{si } i \epsilon R \\ 0 & si \geq 0 \end{cases} \tag{14}$$

donde w son los pesos (en cada coordenada x e y) de la neurona, α es la taza de aprendizaje que la neurona es decir que tan rápido quiero que aprenda la neurona , la funcion h es una función que para el presente trabajo se ha considerado de dos tipos gaussiana y triangular y modelan la manera en que los vecinos de la neurona ganadora, participan del proceso de aprendizaje

Función gaussiana

$$h(\mid i - g \mid, t) = e^{-\frac{(\mid i-g \mid)^2}{2R(t)^2}} \tag{15}$$

función triangular

$$h(\mid i-g \mid, t) = \begin{cases} 0 & \text{si} \mid i-g \mid > t \\ \frac{R(t)-|i-g|}{R(t)} & \text{si} \mid i-g \mid \leq t \end{cases} \tag{16}$$

Donde R es el radio de la vecindad, también se ha tenido en cuenta. la inserción-eliminación de neuronas , es decir , si hay neuronas que nunca ganan , tras un cierto numero de iteraciones simplemente se las elimina, y si hay neuronas que tienen varios ganadores a la vez , se insertan neuronas al lado de esta. Las tasas de aprendizaje α y el radio de la vecindad R decrecen a medida que avanza el proceso de aprendizaje, hasta llegar a una situación que solo una neurona resulta ganadora y el radio de la vecindad es tan pequeño que solo participa en el proceso de aprendizaje esta neurona

Finalmente la solución al problema del agente viajero se da por la salida de las neuronas ordenadas por su peso en w_i

A continuación detallo el algoritmo implementado para el presente trabajo, se tendrá en cuenta que las ciudades y los caminos con sus distancias respectivas que interconectan las ciudades se han modelado haciendo uso de un grafo no dirigido $G(V; E)$, donde V son los vértices o nodos y E son los arcos

Algoritmo 2 Mapa Autoorganizativo para el Problema del Agente Viajero

Requiere: inicializar pesos $w_{i,j}$ para cada neurona en la capa de competición
Mientras los pesos $w_{i,j}$ no varíen significativamente **Hacer**
Para cada nodo u_i de G(V,E) **Hacer**
presentar la ciudad u_i a la red
obtener la neurona ganadora
Si la neurona ganadora , ha ganado antes para esta pasada **Entonces**
valor$\Leftarrow$ aleatorio
Si $valor < 0{,}5$ **Entonces**
crear neurona a la izquierda
asignar pesos a la neurona creada $w_{n,j} = (0{,}04 * w_{k-1,j} + (0{,}95)w_{k,j}) + (0{,}01\gamma_i)$
Si No
crear neurona a la derecha
asignar pesos a la neurona creada $w_{n,j} = (0{,}04 * w_{k+1,j} + (0{,}95)w_{k,j}) + (0{,}01\gamma_i)$
Fin Si
Fin Si
Si si la neurona no gano para tres iteraciones seguidas **Entonces**
eliminar la neurona
actualizar pesos de la neurona ganadora y de sus vecinos
Fin Si
Fin Para
actualizar tasa de aprendizaje y radio de vecindad
Fin Mientras

La solución del problema del agente viajero mediante esta red neuronal se justifica por el hecho de que este tipo de red neuronal es un mapa autoorganizativos, que inicialmente tiene valores aleatorios, pero tras presentarle un conjunto de ciudades con sus respectivas coordenadas tiende a organizarse y buscar neuronas vecinas que se le parezcan, y asociarse con ellas, traducido a nuestro problema para este trabajo será que una cierta ciudad es asociada a cierta neurona,

y los vecinos mas cercanos a esta neurona tienen asociadas ciudades que forman caminos mas cercanos entre ellos , que las ciudades que tienen neuronas que están mas lejos de otras, luego , al final formaran conjuntos de rutas pequeñas que unidas dará la solución final

Se comparó la red neuronal con un algoritmo del tipo greedy

PRUEBAS

Numero ciudades	Ruta Greedy	Ruta Kohonen G	Ruta Kohonen T
4	1160	1160	1160
5	1111.6	1111.6	1111.6
6	1285.50	1115.34	1181.98
7	1567.70	1538.50	1567.70
8	1447.71	1540.44	1659.52
9	1667.81	1688.44	1659
10	1696.86	1672.53	1647.89
11	2061.35	1876.05	1958.55
12	1427.03	1469.86	1469.86
13	2173.22	2104.41	2089.77
14	1787.16	1797.55	1773.59
15	2547.85	2370.59	2605.72

Figura 7: Pruebas realizadas

En los experimentos la red neuronal daba la mayoria de veces menores distancias, mejores rutas, a cambio de una complejidad en el tiempo mas alta con respecto a algortimo tipo Greedy

4.1. Análisis del algoritmo

Análisis del algoritmo

1 En la creación del grafo $G(V, E)$ se utilizó una matriz de adyacencia tenemos: complejidad de espacio $\theta(V^2)$, complejidad de tiempo para listar todos los vertices adyacentes a u : $\theta(V)$, complejidad de tiempo para determinar si (u, v) pertenece a E $\theta(1)$

2 En la creación de la Red neuronal se utilizó una arreglo de neuronas complejidad en inicializar pesos $\theta(n)$, donde cada vértice representa una ciudad, complejidad en accesar cada neurona $\theta(1)$

3 En la obtención de la neurona ganadora, complejidad en calculo de distancia de un vértice u hacia todas las neuronas y computar la neurona ganadora $\theta(n)$

4 Cuando se inserta y elimina neuronas, complejidad en insertar y eliminar neurona $\theta(1)$

5 Para actualizar los pesos, complejidad en actualizar pesos $\theta(n)$

6 En el algoritmo de aprendizaje de kohonen, En determinar todas neuronas ganadoras para todos los vertices u del grafo $\theta(V * n * iteraciones)$, En insertar-eliminar neuronas $\theta(iteraciones * V)$

La complejidad del algoritmo resultante es la que sigue $\theta(V * n * iteraciones)$, donde V es el numero de vertices del grafo , que en este caso representarían las ciudades, n es el numero de neuronas, $iteraciones$ son el numero de iteraciones necesarias para que el algoritmo aprenda.

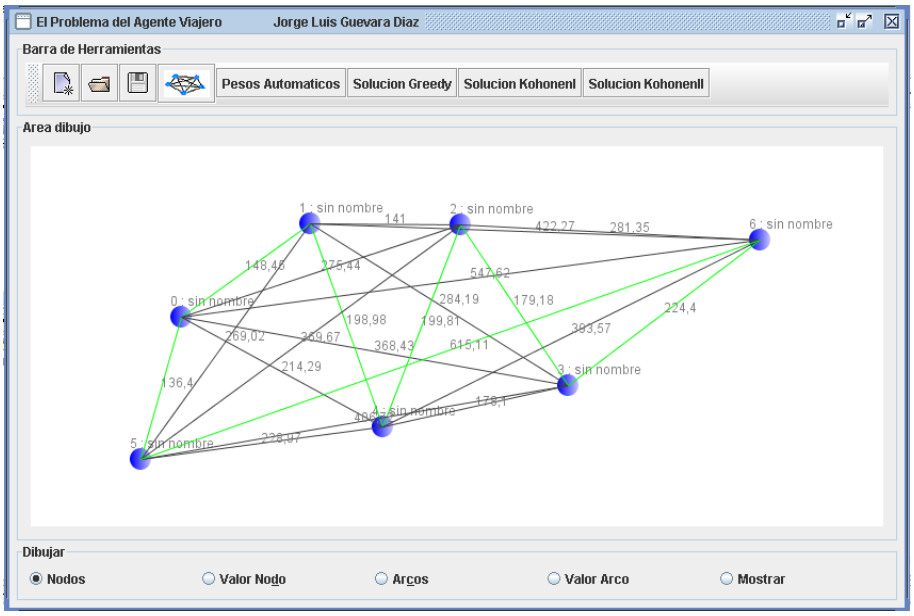

Figura 8: Salvapantalla de la implementación hecha para el presente trabajo, las lineas verdes indican la mejor ruta encontrada por el algoritmo

El resultado de este analisis es favorable frente al hecho de que una solución por fuerza bruta es del orden factorial $\theta((n-1)!)$

La solución del problema del agente viajero mediante esta red neuronal se justifica por el hecho de que este tipo de red neuronal es un mapa autoorganizativos, que inicialmente tiene valores aleatorios, pero tras presentarle un conjunto de ciudades con sus respectivas coordenadas tiende a organizarse y buscar neuronas vecinas que se le parezcan, y asociarse con ellas, traducido a nuestro problema para este trabajo será que una cierta ciudad es asociada a cierta neurona, y los vecinos mas cercanos a esta neurona tienen asociadas ciudades que forman caminos mas cercanos entre ellos , que las ciudades que tienen neuronas que están mas lejos de otras, luego , al final formaran conjuntos de rutas pequeñas que unidas dará la solución final

5. Red Neuronal Backpropagation

Es una red neuronal supervisada es decir va a requerir conocer las salidas en un determinado momento de tiempo para poder actualizar los pesos de las neuronas, esta red permite que presentando una serie de valores arbitrarios, la red tienda a reconocer el patrón tras haber aprendido de patrones anteriores, incluso si nunca antes ha visto al patrón de entrada

Esta red aprende de un conjunto de valores dados y posteriormente aprende reconocer patrones nunca antes presentados a la red, es decir asemeja los patrones de entrada, con los patrones que y aha visto anteriormente

Una de las ventajas de esta red, es que es buena en reconocer patrones que contengan cierto ruido

El funcionamiento de esta rd es como sigue , primero la red aprende conjunto de datos dados en formas como par entrada-salida, y para esto emplea un ciclo de propagación dado en dos fases , cuando se aplica el primer conjunto de datos, los datos correspondientes a la entrada se propagan por la red hacia las neuronas ocultas de esta hasta llegar a la salida , luego esta

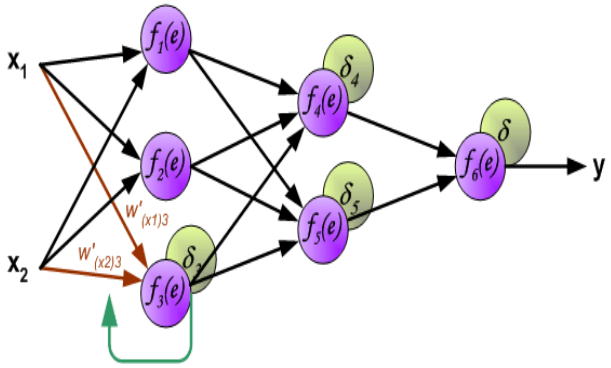

Figura 9: Arquitectura de una Red Neuronal Backpropagation

señal de salida obtenida en la fase de propagación se compara con la salida deseada y se hace un cálculo del error para cada neurona de la capa de salida, luego estos errores nos permitirán ajustar los pesos de la red neuronal, estos errores se empiezan a transmitir hacia atrás , este es el ciclo de propagación hacia atrás, permitiendo conocer que tanto contribuyó cada neurona en el error global y según esto es que ha ce la actualización de pesos

Concluido el entrenamiento de esta red , esta será capaz de reconocer correctamente datos que se le presenten que sean similares a los valores usados en la parte del entrenamiento de la red, esto es por que las neuronas después de haber concluido el proceso de entrenamiento, serán activadas dada una señal de entrada, siendo estas activaciones exitatorias o inhibitorias. La red neuronal Backpropagation tiene la característica de reconocer datos que no ha visto en su etapa de entrenamiento según las características que comparten con sus datos de entrenamiento.

El funcionamiento de la red que detallamos a continuación se basa el algoritmo de la regla delta generalizada , para un vector p de entrada $x = (x_{p1}, ..., x(pi), ..., x_{pN})$, en la capa de entrada , este se distribuye hacia las capas ocultas, luego la entrada en la $j - esima$ neurona oculta será:

$$EntradaNeurona_{pj} = \sum_{i=1}^{n} x_{pi} w_{ij} + \theta_j \tag{17}$$

Esto es la suma ponderada de del vector de pesos y la entrada , mas un valor θ que es un valor de $tendecia$ que puede verse como una neurona mas pero que tiene un valor de entrada de 1, este valor se puede ver como el valor umbral para el valor de activación de las neuronas, , hacemos $EntradaNeurona_{pj}$ = α_{pj} y $EntradaNeurona_{pk}$ = α_{pk}, luego la salida para este nodo es:

$$i_{pj} = f_j(\alpha_{pj}) \tag{18}$$

Luego en la capa de salida:

$$\alpha_{pk} = \sum_{i=1}^{n} x_{pj} w_{jk} + \theta_k \tag{19}$$

$$o_{pk} = f_k(\alpha_{pk}) \tag{20}$$

Donde o son los valores de salida de la red

A continuación se procede hacer el calculo del error y actualizar los pesos de las capas ocultas:

$$e_k = y_k - o_k \tag{21}$$

y es la salida deseada y $oeslaobtenida$, luego el error total está dado por la expresíon

$$E_p = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2 \tag{22}$$

y es el error que se debe minimizar

Se debe determinar como se actualizan los pesos, haciendo uso de este error, para esto se puede ver al error en función de las combinaciones de pesos en un espacio n-dimensional, donde n, es el número de pesos de la capa de salida, Por ejemplo se puede pensar en una red que solo tenga dos pesos de salida, luego se puede ver la función de error como una superficie , en función de los dos pesos

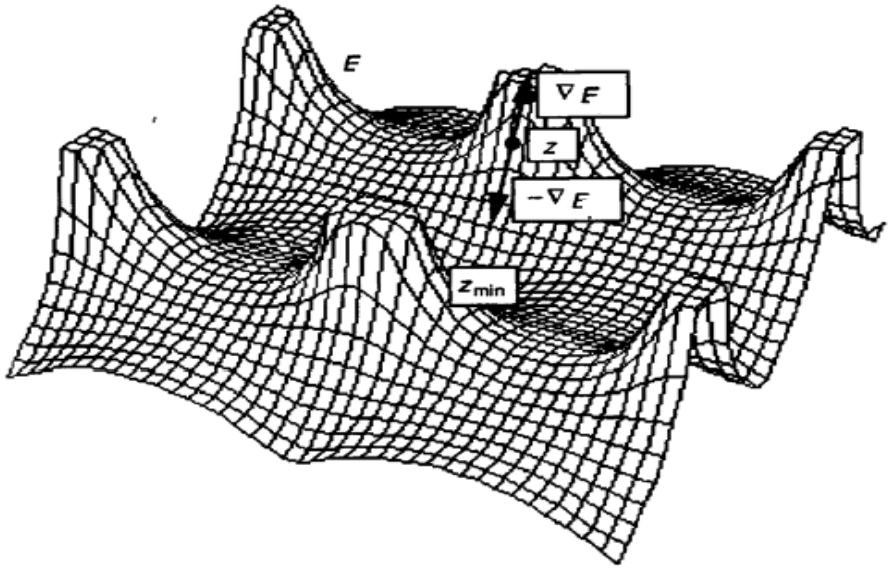

Figura 10: Error en función de dos pesos, los pesos deben ajustarse en función del gradiente negativo ∇E_p, hasta que el error alcance un punto z_{min}

Los que se quiere es encontrar de que manera los pesos de la capa de salida influyen en la función de error y de que manera estos pueden modificarse para minimizar el error, obviamente existen varios mínimos locales en la función de error, y la red erróneamente puede caer en un mínimo local que no sea óptimo para la solución, para modificar los pesos se tiene que modificarlos en dirección opuesta al gradiente:

$$\triangle w_{jk} = -\eta\frac{\partial E}{\partial w_{jk}} \tag{23}$$

Donde η es el valor de aprendizaje de la neurona; luego el calculo de $\frac{\partial E}{\partial w_{jk}}$ se hace utilizando la regla de la cadena

$$\frac{\partial E}{\partial w_{jk}} = -(y_{pk} - o_{ok})\frac{\partial f_k(\alpha_{pk})}{\partial \alpha_k}\frac{\partial \alpha_{pk}}{\partial w_{jk}} \tag{24}$$

El calculo de $\frac{\partial \alpha_{pk}}{\partial w_{jk}}$, lo podemos obtener mediante:

$$\frac{\partial \alpha_{pk}}{\partial w_{jk}} = \frac{\partial}{\partial wjk}\sum_{j=1}^{L} i_{pj}w_{jk} = i_pj \tag{25}$$

Luego la expresión quedaría:

$$\frac{\partial E}{\partial w_{jk}} = -(y_{pk} - o_{ok})\frac{\partial f_k(\alpha_{pk})}{\partial \alpha_{pk}}i_{pj} \tag{26}$$

La función f_k tiene que ser derivable, un ejemplo de ello es la función sigmoidea:

$$f_k(\alpha) = \frac{1}{1 + \frac{1}{e^{-\alpha}}} \tag{27}$$

donde $\alpha = \alpha_{jk}$, para este caso la actualización de pesos viene dado por:

$$w_{jk}(t+1) = w_{jk}(t) + \eta(y_{p_k - o_{pk}})o_{pk}(1 - o_{pk})i_{pj} \tag{28}$$

De donde decimos :

$$\delta_{pk} = (y_{p_k - o_{pk}})f_k(\alpha_{jk}) \tag{29}$$

entonces se puede reescribir la ecuación de actualización de pesos como sigue:

$$w_{jk}(t+1) = w_{jk}(t) + \eta\delta_{pk}i_{pj} \tag{30}$$

Ahora se procederá a actualizar los pesos de las capas ocultas , el procedimiento es el siguiente: debemos conocer de que manera influyen los pesos de la capa oculta de la red neuronal en los errores Ep de salida de la red

$$
\begin{aligned}
E_p &= \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2 \\
&= \frac{1}{2}\sum_{k=1}^{M}(y_k - f_k(\alpha_{pk}))^2 \\
&= \frac{1}{2}\sum_{k=1}^{M}(y_k - f_k(\sum_{j}^{L} i_{pj}w_{jk}))^2 \\
\frac{\partial E_p}{\partial wij} &= \frac{1}{2}\sum_{k=1}^{M}\frac{\partial}{\partial w_{ij}}(y_k - o_k)^2
\end{aligned}
$$

Entonces se puede calcular el gradiente de Ep con respecto a los pesos de las capas ocultas

$$\frac{\partial E_p}{\partial wij} = -\sum_{k}^{L}(y_k - o_k)\frac{\partial o_{pk}}{\partial \alpha_{pk}}\frac{\partial \alpha_{pk}}{\partial i_{pj}}\frac{\partial i_{pj}}{\partial \alpha_{pj}}\frac{\partial \alpha_{pj}}{\partial w_{ij}} \tag{31}$$

Luego de la derivación la expresión tenemos que la variación de los pesos estará dado por

$$\Delta_p w_{ij} = \eta f_j(\alpha_{pj})x_{pi}\sum_{k}^{L}(y_{pk} - o_{pk})f_k(\alpha_{pk})w_{jk} \tag{32}$$

luego hacemos:

$$\delta_{pj} = f_j(\alpha_{pj})\sum_{k}^{L}(y_{pk} - o_{pk})f_k(\alpha_{pk})w_{jk} \tag{33}$$

$$\delta_{pj} = f_j(\alpha_{pj})\sum_{k}^{L}\delta_{pk}w_{jk} \tag{34}$$

Lo que se observa es que en el calculo de los deltas de la capa oculta se emplean los valore de los deltas calculados para la capa de salida, es decir , una neurona de la capa oculta aporta en los deltas de la capa de salida y haciendo el proceso inverso, tomando los deltas de la capa de salida, puedo calcular el delta de una neurona de la capa oculta. La actualización de los pesos estaría dado por:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_{pj}x_i \tag{35}$$

Este procedimiento se hará hasta que se encuentre un error mínimo que sea global

A continuación se muestra el algoritmo de aprendizaje para la red neuronal de backpropagation mediante la regla delta

APRENDIZAJE-BACKPROPAGATION($Patrones$)

$w_{i,j} \leftarrow$ aleatorio
$w_{j,k} \leftarrow$ aleatorio
while $E(t) - E(t-1) <$ poca diferencia ▷ minimizar Error global
    do
        for $p \leftarrow 1$ **to** $length[Patrones]$
            do presentar$X_p = (x_{p1}, .., x_{pN})$ a las unidades de la capa de entrada
                for $j \leftarrow 1$ **to** L ▷ para todas las neuronas de la capa oculta
                    do
                        $\alpha_{pj} \leftarrow \sum_{i=1}^{N} x_{pi} w_{ij}$ ▷ activación para neurona j
                        $i_{pj} \leftarrow f_j(\alpha_{pj})$ ▷ salidas para la neurona j
                for $k \leftarrow 1$ **to** M ▷ para todas las neuronas de la capa de salida
                    do
                        $\alpha_{pk} \leftarrow \sum_{j=1}^{L} i_{pj} w_{jk}$ ▷ activación para neurona k
                        $o_{pk} \leftarrow f_j(\alpha_{pj})$ ▷ salidas para la neurona k(salidas de la red)
                for $k \leftarrow 1$ **to** M ▷ para todas las neuronas de la capa de salida
                    do
                        $\delta_{pk} \leftarrow (y_{pk-o_pk}) f_k(\alpha_{pk})$ ▷ términos de error para las neuronas de salida
                for $j \leftarrow 1$ **to** L ▷ para todas las neuronas de la capa oculta
                    do
                        $\delta_{pj} = f_j(\alpha_{pj}) \sum_{k=1}^{M} \delta_{pk} w_{jk}$ ▷ términos de error para las neuronas de la capa oculta
                for $k \leftarrow 1$ **to** M ▷ para todas las neuronas de la capa de salida
                    do
                        $w_{jk}(t+1) = w_{jk} + \eta \delta_{pk} i_{pj}$ ▷ actualizar pesos para las neuronas de salida
                for $j \leftarrow 1$ **to** L ▷ para todas las neuronas de la capa oculta
                    do
                        $w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} x_i$ ▷ actualizar pesos para las neuronas de la capa oculta
                for $k \leftarrow 1$ **to** M ▷ para todas las neuronas de la capa de salida
                    do
                        $E_p = E_p + \frac{1}{2} \delta_{pk}^2$

        $E \leftarrow \sum_{p=1}^{length[Patrones]} E_p$ ▷ Error global que deber ser minimizado

return w_{ij} y w_{jk}

el valor de η es un valor e aprendizaje de la red, es decir este factor me va a indicar que tan rápido o tan lento puede aprender la red por eso usualmente también se le llama parámetro de velocidad de aprendizaje, generalmente este toma valores menores que 1, pero generalmente para obtener buenos resultados se usan valores entre 0.05 y 0.25 , cuando se tiene valores muy pequeños la red neuronal hará mas iteraciones en el proceso de aprendizaje , pero se evitará rebotes de los mínimos locales encontrados , parámetros grandes no son muy útiles pues la red puede rebotar en el proceso de aprendizaje Una vez que la red ha aprendido , se tienen los pesos w_{ij} y w_{jk} de tal manera , que están listos para poder reconocer patrones de entrada semejantes, el algoritmo de reconocimiento es:

RECONOCIMIENTO-BACKPROPAGATION$(Patron - X_p, w_{ij}, w_{jk})$

presentar$X_p = (x_{p1}, .., x_{pN})$ a las unidades de la capa de entrada
for $j \leftarrow 1$ **to** L ▷ para todas las neuronas de la capa oculta
    do
        $\alpha_{pj} \leftarrow \sum_{i=1}^{N} x_{pi} w_{ij}$ ▷ activación para neurona j
        $i_{pj} \leftarrow f_j(\alpha_{pj})$ ▷ salidas para la neurona j
for $k \leftarrow 1$ **to** M ▷ para todas las neuronas de la capa de salida
    do
        $\alpha_{pk} \leftarrow \sum_{j=1}^{L} i_{pj} w_{jk}$ ▷ activación para neurona k
        $o_{pk} \leftarrow f_j(\alpha_{pj})$ ▷ salidas para la neurona k(salidas de la red)
return o_{pk}

6. Aplicación de la Red Neuronal Backpropagation en el reconocimiento de dígitos

Este tipo de red neuronal ha mostrado ser buena para problemas de clasificación de tramas que tengan ruido en la entrada, pues reconocerá patrones que se asemejen a los valores que esta red ha aprendido

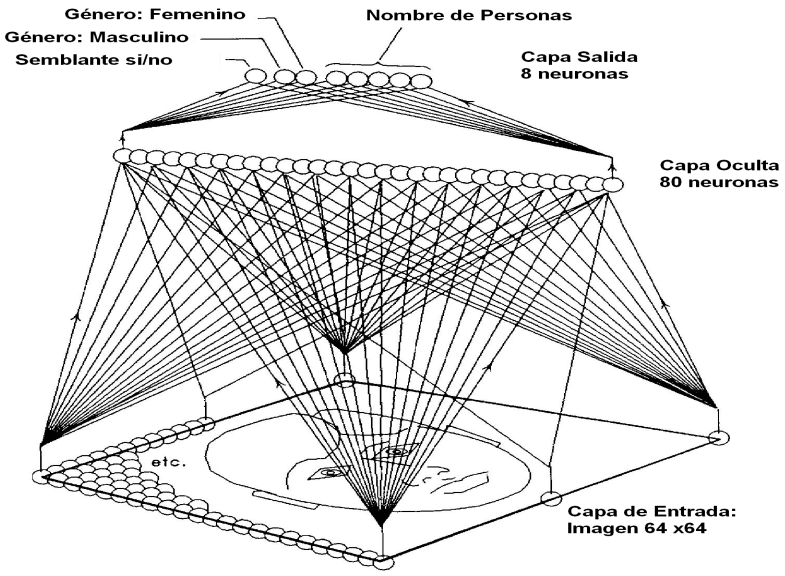

Figura 11: Red backpropagation para el reconocimiento de rostros

En la figura se ve una red para el reconocimiento de rostros, la entrada es una figura de un rostro y tras un proceso de aprendizaje la red aprenderá los nombres de las personas incluso el sexo de las personas

Para el presente trabajo se implementó este tipo de red para reconocimiento de dígitos, la red neuronal se construyó de la siguiente manera: Los patrones a ser presentados a la red , son imágenes de 49 pixels de tamaño (7x7), la red neuronal tiene 49 neuronas de entrada, 4 neuronas para la capa oculta y 4 neuronas para la capa de salida, se utilizó un valor de aprendizaje $\eta = 0{,}25$, la función de salida de cada neurona se consideró una función sigmoidal

Las imágenes tenían un 0 para indicar la ausencia de información y un 1 para indicar la

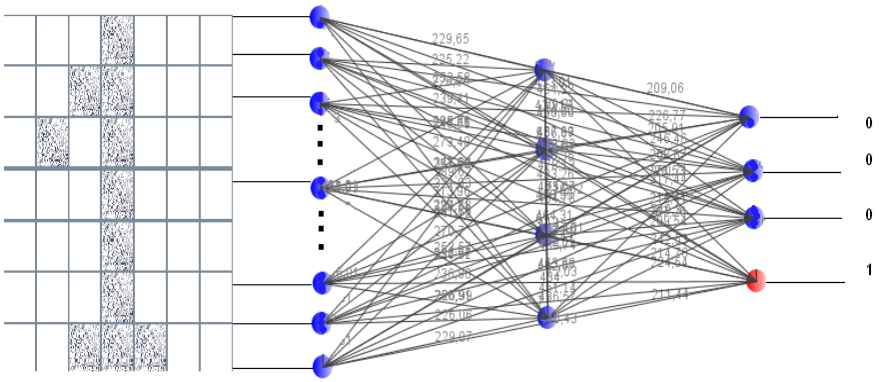

Figura 12: Estructura de la Red backpropagation que se implementó para el reconocimiento de dígitos en el presente trabajo, se ve como la entrada de la imagen con el valor 1 activa a la neurona de salida de color rojo para una red que ya aprendió

presencia de información

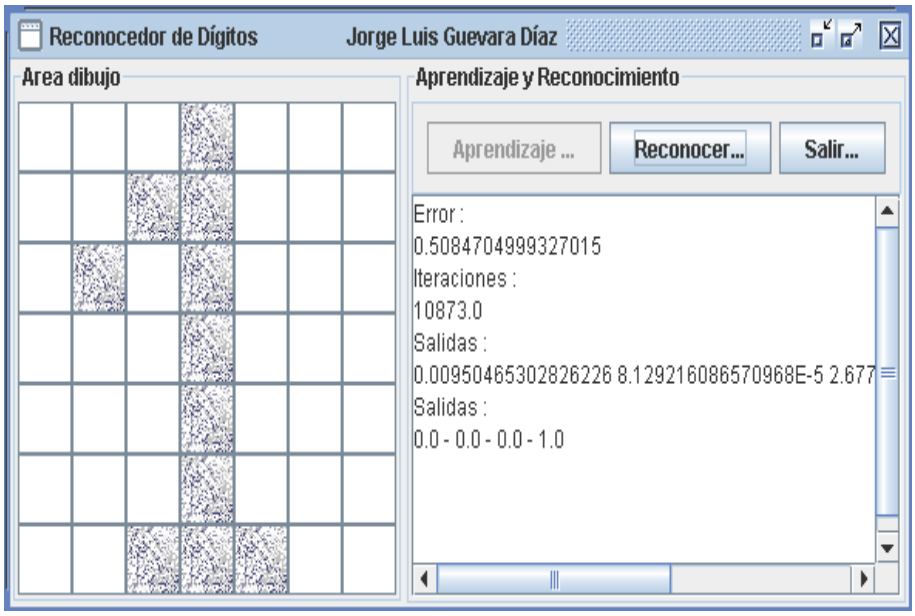

Figura 13: implementación realizada para el siguiente trabajo, en la figura se muestra la red reconociendo el dígito uno

La red neuronal se entrenó con 13 patrones de entrada correspondiente a los dígitos del 0 al 9 y para las letras A, B, C y los valores de salida asociados eran : $0, 0, 0, 0$ para indicar el 0 ,$0, 0, 0, 1$ para indicar el 1,$0, 0, 1, 0$ para indicar el 2 y así sucesivamente hasta el numero 9 cuyo valor asociado fué $1, 0, 0, 1$, del mismo modo para las letras que para este caso indicaban ruido

La red mostró buen comportamiento para reconocer tramas con ruido, se aplico el algoritmo de aprendizaje a la red con ciertos patrones y tras obtener un error de $0{,}5084$ con 10873 iteraciones estos son algunos de los resultados obtenidos

Para obtener mejores resultados se deben probar con varios entrenamientos, para poder obtener mejores ajustes en los pesos que disminuyan el error global

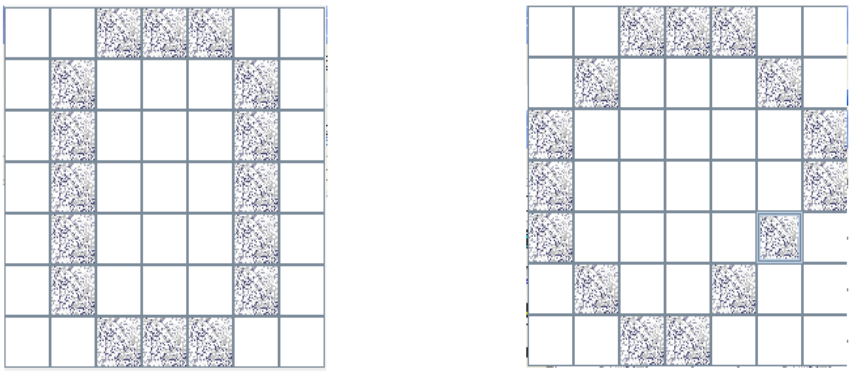

Figura 14: en la derecha se ve el valor de entrenamiento, en la izquierda el valor reconocido por la red

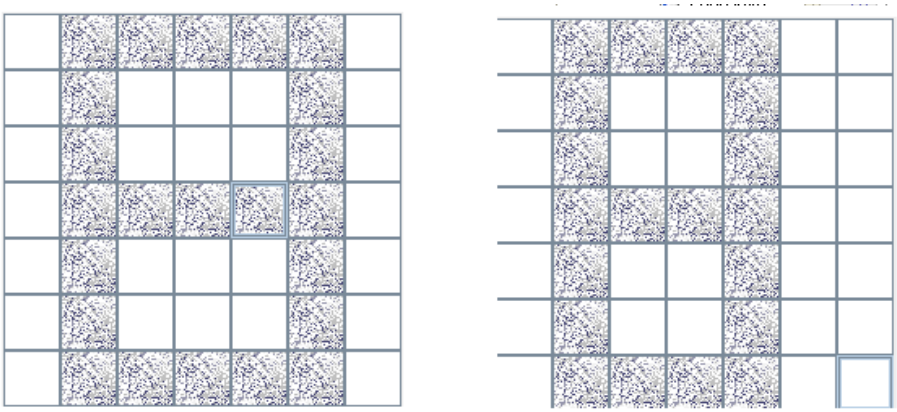

Figura 15: en la derecha se ve el valor de entrenamiento, en la izquierda la red no reconoció este valor

6.1. Análisis del algoritmo

El algoritmo de aprendizaje de la red neuronal puede tener una complejidad elevada si no se seleccionan los valores correctos de η, o incluso no llegar a aprender nunca, por eso se debe tener mucho cuidado en el momento de diseñar la red, se debe tratar de tener el mínimo número de neuronas en la capa oculta, pues mas unidades en la capa oculta significa mayor costo computacional,una de las limitaciones del algoritmo presentado es que se requiere que las entradas sean de las mismas dimensiones. Existen algoritmos de poda de neuronas ociosas, y algunas variantes del Backpropagation, pero hago un análisis del algoritmo detallado líneas más arriba.

Para analizar el algoritmo tendremos en cuenta que el análisis desde el punto de vista de una máquina secuencial pues en una máquina con procesadores paralelos el análisis sería diferente, y por ejemplo muchas de las propagaciones se darán en tiempo lineal ,empezando el análisis se tiene si tenemos p patrones , de tamaño n el costo inicial para presentar todos los patrones a la capa de entrada es $\theta(p)$ lienal en el número de patrones , pero si tenemos en cuenta la cantidad de información de cada patrón se tendrá $\theta(p * n)$

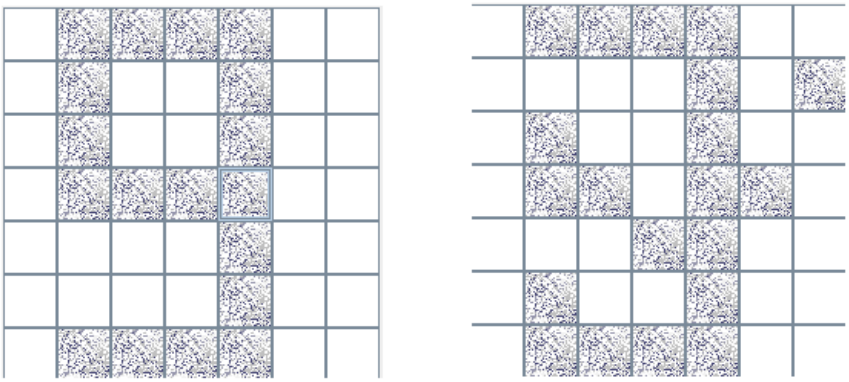

Figura 16: en la derecha se ve el valor de entrenamiento, en la izquierda la red si pudo reconocer este valor

Cuando propagamos el patron hacia las neuronas de la capa oculta para activar todas las neuronas de la capa oculta si se tiene en cuenta que el número de neuronas de la capa oculta es L se tiene $\theta(L * n)$ y para obtener las salidas de esta capa $\theta(L)$

Cuando se propaga las salidas obtenidas hacia la capa de salida teniendo en cuenta que el número de neuronas de la capa de salida es M se tendrá que para activar todas las neuronas de esta capa se tendr á$\theta(M * n)$

Cuando se hace el cálculo de los términos de error en la capa de salida se tiene $\theta(M)$ y cuando se hace éste calculo en las unidades de la capa oculta se tiene $\theta(M * L)$

En actualizar los pesos de la capa de salida se tiene $\theta(M)$ similarmente para los pesos de la capa oculta $\theta(L)$

Cuando se calcula los errores parciales que contribuyen al error global en la capa de salida $\theta(M)$ Cuando se calcula el error global $\theta(p)$

Resumiendo se tiene:

1. Capa de entrada $\theta(n)$
2. Capa oculta$\theta(L * n)$
3. Capa salida$\theta(L)$
4. Delta salida $\theta(M)$
5. Delta capa oculta $\theta(M * L)$
6. Pesos capa salida $\theta(M)$
7. Pesos capa oculta $\theta(L)$
8. Errores parciales $\theta(M)$
9. Error Global $\theta(p)$

Sabemos que $n + L + M$ = número de neuronas de la red, correspondientes a la capa de entrada , capa oculta y capa de salida, luego el tiempo de ejecución de esta parte es: $n + L * n + 2L + 4M + M * L + p$, se nota que el valor de la complejidad dependerá del número de neuronas de la red neuronal , a más neuronas mayor complejidad , por ejemplo si se supone que los valores de L y M fueran n , se tendría $\theta(n^2)$ por complejidad en esta parte del sistema.

Como se sabe que esto sucede p veces y si llamamos $\theta(\lambda)$ a la complejidad de $n + L * n + L + M + M * L + M$, se tiene $\theta(p * \lambda)$ finalmente teniendo en cuenta el número de iteraciones $iter$ hasta el que se minimice el error global , se tiene que la complejidad total en la fase de aprendizaje es $\theta(iter * p * \lambda)$

7. Conclusiones

Las redes neuronales artificiales nos ayudan a encontrar soluciones a diversos problemas para los cuales no existen algoritmos que den soluciones buenas en tiempos razonables, son útiles en diversos tipos de problemas: desde reconocimiento de tramas , problemas de optimización, Un estudio profundo en este campo requiere conocimientos de neurofisiología, para entender el comportamiento biológico y poder modelar computacionalmente diversos comportamientos de las neuronas biológicas.

Los modelos computacionales inspirados en el comportamiento biológico han mostrado ser muy buenos, pero son solo inspiraciones, El comportamiento humano ha resultado ser muy fascinante y complejo que todavía no es posible describir determinados fenómenos en forma precisa, conforme avancen los conocimientos en este campo, esto contribuirá mucho al avance al área de las redes neuronales artificiales

Referencias

[CS01] Leiserson C. Rivest R. Cormen, T. and C. Stein, *Introduction to algorithms, second edition.*, 2001.

[Del03] L Delgado, *Diseño de una red neuronal autoasociativa para resolver el problema de viajante de comercio (traveling salesman problem, tsp.*, Congreso Nacional de Estadística e Investigación Operativa Lleida, 8-11 de abril de 2003 (2003).

[Die01] R. Diestel, *Graph theory.*, 2001.

[Fre91] J.and D.Skapura Freeman, *Redes neuronales artificiales, algoritmos, aplicaciones y técnicas de programación.*, 1991.

[Koh81] T. Kohonen, *Automatic formation of topological maps of patterns in a self-organizing system.*, In Proc. 2nd Scandinavian Conf. Image Analysis, Espoo, Finland, 15-17 June 1981,pp. 214-220. Helsinki: Pattern Recognition Society of Finland. (1981).

[Rei03] G. Reinelt, *The traveling salesman: Computational solutions for tsp applications.*, Springer Berlin 2003. (2003).