La función de Transferencia Compet usando Matlab

E. Largo[1] R. León[2] H. Paz[3]

***Resumen*—Este artículo brinda un enfoque a la función de transferencia compet utilizada en inteligencia artificial específicamente redes neuronales, tiene como objetivo dar a conocer de forma clara lo que es la función compet, se explica su funcionamiento y utilización; así como un ejemplo ilustrativo haciendo uso del software Matlab.**

***Index Terms*—Neurona, función de activación, redes neuronales, capa oculta, aprendizaje supervisado y no supervisado.**

I. Introducción

En el presente documento se va a describir la función compet, cual es su utilización y funcionamiento, para esto se explica algunos conceptos básicos de relación directa con las redes neuronales, las funciones de activación, tipos de aprendizaje de manera que se facilite su comprensión a través de una amnera clara y sencilla. Cuál es el sentido de utilizar este tipo de redes y cual es el beneficio que aportan. Por ultimo se realiza una aplicación práctica utilizando una herramienta de software como es Matlab, se describe los comandos y sentencias utilizados en el desarrollo de la misma.

II. Redes Neuronales Auto organizadas[1]

Antes de empezar hablar sobre las redes auto organizadas debemos hacer una relación entre estas y el cerebro humano. El cerebro humano está dividido en áreas y su estructura depende de la función que tienen, las conexiones entre neuronas se realizan de acuerdo a una estructura, esta organización se la conoce como mapa.

Estos mapas tienen la característica de ser autonomos o auto organizados, es decir que no tienen una referencia en cuanto a los patrones ingresados, es por eso que el cerebro tiene la capacidad de procesar información sin tener una referencia de salida que podría tener, si no mas bien la organiza o clasifica de acuerdo a sus características y la localiza en ciertas áreas, por ejemplo cuando nos dicen que a través de la observación obtengamos un análisis, esta información va al cerebro y se localiza en una área donde un sentido pueda procesarla sin tener una referencia de la información de salida, estos procesos de localización ocurrirán con toda la nueva información que ingresa al cerebro.

El comportamiento visto anteriormente lo toman las redes autoorganizadas, estas redes pueden aprender a detectar regularidades y correlaciones en los datos de entrada y adaptar su respuesta futura de acuerdo con los datos de entrada. Las neuronas de una red competitiva aprenden a reconocer grupos de vectores de datos de entrada similares.

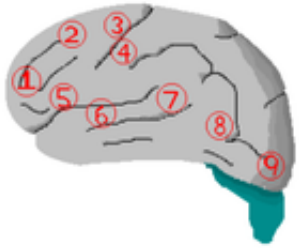

Figura 1. Representación de las áreas del cerebro.

Como se aprecia en la figura 1 los diferentes números corresponden:

1. Pensamiento.
2. Planificación de acciones.
3. Funciones motoras.
4. Mapa somatosensorial.
5. Habla.
6. Oído primario.
7. Reconocimiento de palabras.
8. Movimientos de los ojos.
9. Visión

III. Aprendizaje Supervisado [2]

Este tipo de aprendizaje consiste en proporcionar los patrones de entrada y la salida deseada. Para llegar a obtener la salida requerida se utiliza una fórmula matemática de minimización de errores de manera que se pueda obtener un resultado lo más acercado posible a la salida deseada. Pasos para el aprendizaje:

1. Se inicializa los pesos de forma aleatoria para cada patrón.
2. Realizar el cálculo de la salida a partir del patrón de entrada y los pesos aleatorios.
3. Encontrar el error de calculo entre la salida deseada y la salida actual.
4. Ajustar los pesos usando la regla de aprendizaje para disminuir el error medio.
5. Si el error es mayor de cierto criterio volver al paso 2.

IV. Aprendizaje No Supervisado [3]

En este tipo de aprendizaje no se conoce o no se le indica a la red cual es la salida deseada únicamente nos limitamos a darle los patrones de entrada, lo que hace la red estos casos

[1]E. Largo, Universidad Nacional de Loja, Loja, Ecuador eolargor@unl.edu.ec
[2]R. León, Universidad Nacional de Loja, Loja, Ecuador mrleonr@unl.edu.ec
[3]Tutor. H. Paz, Universidad Nacional de Loja, Loja, Ecuador hpaz@unl.edu.ec

es categorizar las entradas compara los patrones de entrada con patrones ya establecidos y los organiza de acuerdo a la similitud entre el patron de entrada y el establecido.

V. Redes Multicapa

Una red multicapa es una red de alimentación hacia adelante o también conocida como feedforward esta red se compone de una capa de unidades de entrada llamadas sensores y otra capa de unidades de salida y un número determinado de capas intermedias de unidades de proceso, también llamadas capas ocultas porque no tienen conexiones con el exterior. Cada sensor de entrada está conectado con las unidades de la segunda capa, y cada unidad de proceso de la segunda capa está conectada con las unidades de la primera capa y con las unidades de la tercera capa, así sucesivamente. Las unidades de salida están conectadas solamente con las unidades de la última capa oculta, como se muestra en la figura 2.
Un aspecto importante de la competición es que lleva a la optimización a nivel local sin necesidad de utilizar una serie de recursos en un control global. Esto es de especial importancia en los sistemas distribuidos en los que supone un enorme coste.

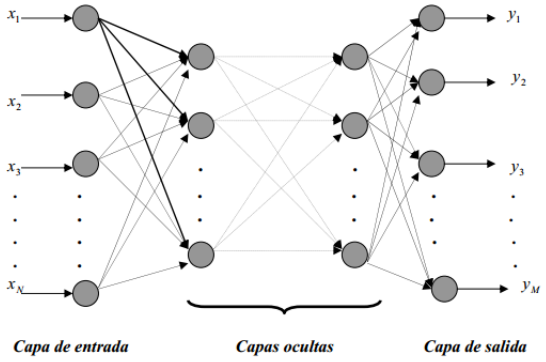

Figura 2. Topología de un Perceptrón Multicapa.

VI. Redes Neuronales Competitivas [4]

En este tipo de redes a diferencia de otro tipo de redes en las cuales todas las neuronas colaboran entre si de manera que a través del aprendizaje se pueda dar una representación al patrón de entrada, en el caso de una red competitiva las neuronas compiten unas con otras de manera que solo una se activa; dicho de otra manera solo esta neurona será capaz de representar el patrón de entrada. el objetivo de que una neurona compita por ser la ganadora es que es puede representar en un grupo o categoría.

VI-A. Estructura de una Red Competitiva

Una red competitiva está constituida por N sensores de entrada, M unidades de proceso (neuronas artificiales), y conexiones entre cada sensor y cada unidad de proceso, de manera que la conexión entre el sensor j y la unidad de proceso i tiene asociado un valor wij.

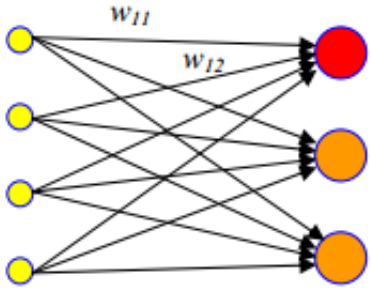

Figura 3. Arquitectura de la red.

Se han desarrollado algunos modelos basados en redes auto organizadas que funcionan como una red competitiva, continuación se menciona algunos de ellos.

VI-B. Red de Kohonen

Este tipo de redes fue presentado por Tuevo Kohonen en 1982 se basan en la zonas en en que esta divido el cerebro humano, este tipo de redes utiliza un aprendizaje no supervisado y es de tipo competitivo, esto significa que las neuronas de la capa de salida compiten entre si por activarse quedando solo una de ellas ganadora, los pesos de las conexiones se ajustan de acuerdo a la neurona ganadora.

El modelo tiene dos variantes:

- LVQ (learning vector quantization)
- TPM o SOM (topologic preserving map, o selforganizing map).

Los vectores de entrada a la red pueden ser de diferentes dimensiones, siendo en el caso de LVQ de una dimensión, y en el caso de TPM bidimensionales o incluso tridimensionales.

Arquitectura LVQ: Se trata de una red de dos capas con N neuronas de entrada y M de salida. Cada una de las N neuronas de entrada se conecta a las M de salida mediante conexiones hacia delante. Entre las neuronas de las capas de salida existen conexiones laterales, ya que cada una de ellas tiene influencia sobre sus vecinas a la hora de calcular los pesos de las conexiones hacia delante entre la capa de salida y la capa de entrada.

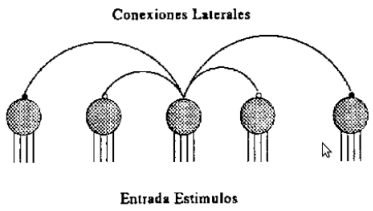

Figura 4. Arquitectura LVQ.

En las neuronas de la capa de salida, cada neurona está conectada con otras de su entorno, de manera que produce una excitación en las más próximas y una inhibición

en las más alejadas. Tanto la excitación como la inhibición laterales son gradualmente más débiles a medida que nos alejamos de la neurona en cuestión.

Arquitectura SOM: Trata de establecer una correspondencia entre los datos de entrada y un espacio bidimensional de salida, de modo que ante datos con características comunes se activen neuronas situadas en zonas próximas de la capa de salida.
Cada neurona de la capa de entrada está conectada con cada una de las neuronas de la capa de salida mediante pesos.
Las interacciones laterales siguen existiendo en relación con la distancia que se toma como una zona bidimensional alrededor de la neurona.

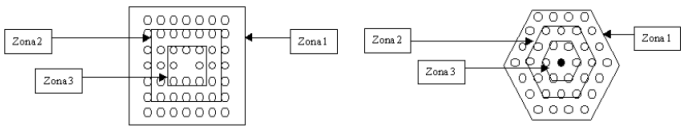

Figura 5. Arquitectura de la red.

VI-C. Aplicaciones de la red de Kohonen

Las aplicaciones que mas destacan de los mapas de Kohonen son las relacionadas con el reconocimiento de patrones, o extracción de características, por ejemplo:

- Reconocimiento de voz
- Codificación de datos
- Compresión de imágenes
- Resolución de problemas de optimización

VI-D. Red de Hamming

Este tipo de red es uno de los ejemplos más sencillos de aprendizaje competitivo aunque se dice que su estructura es complejo debido a que emplea el concepto de capas recurrente en su segunda capa. Las neuronas en la capa de salida de esta red compiten unas con otras para determinar la ganadora, la cual índica el patrón prototipo más representativo en la entrada de la red, la competición es implementada por inhibición lateral (un conjunto de conexiones negativas entre las neuronas en la capa de salida). Esta red consiste en dos capas; la primera capa, la cual es una red Instar, realiza la correlación entre el vector de entrada y los vectores prototipo, la segunda capa realiza la competición para determinar cual de los vectores prototipo está más cercano al vector de entrada.

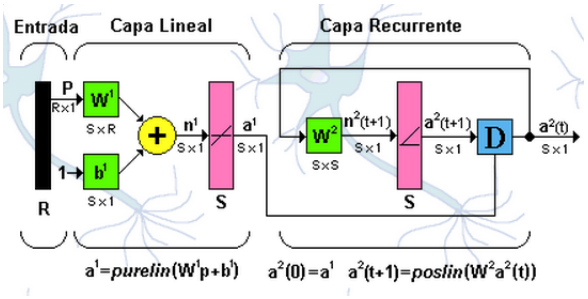

Figura 6. Red de Hamming.

Capa 1: La red Instar es capaz de clasificar solo un patrón; para que múltiples patrones sean reconocidos se necesitan múltiples Instar y es precisamente de esa forma como está compuesta la primera capa de la red de Hamming. Para una mejor comprensión de su funcionamiento se partirá de unos vectores prototipo que la red debe clasificar.

$$\{p_1, p_2, \ldots, p_Q\}$$

Figura 7. Vectores prototipo.

La matriz de pesos W1 y el vector de ganancias b1 para la capa uno serán:

$$W^1 = \begin{bmatrix} {}_1W^T \\ {}_2W^T \\ \vdots \\ {}_sW^T \end{bmatrix} = \begin{bmatrix} p_1^T \\ p_2^T \\ \vdots \\ p_Q^T \end{bmatrix}, \; b^1 = \begin{bmatrix} R \\ R \\ R \\ R \end{bmatrix}$$

Figura 8. Matriz de pesos y vector ganancias.

Donde cada fila de W1 representa un vector prototipo, el cual deseamos reconocer y cada elemento b1 es igual al número de elementos en cada vector de entrada (R) (el número de neuronas S es igual al número de vectores prototipo Q). Así la salida de la primera capa será:

$$a^1 = W^1 p + b^1 = \begin{bmatrix} p_1^T p + R \\ p_2^T p + R \\ \vdots \\ p_Q^T 1 + R \end{bmatrix}$$

Figura 9. Salida capa 1.

La salida de la capa 1 es igual al producto punto de los vectores prototipo con la entrada más el vectorR; este producto indica cuan cercano está cada vector de entrada a los patrones prototipo.

Capa 2: La red Instar emplea una función de transferencia poslin para decidir si el vector de entrada estaba lo suficientemente cerca al vector prototipo. En la capa 2 de la red de Hamming. Se utilizan múltiples Instar, así se determinará por medio de una capa competitiva el patrón prototipo

más cercano. Las neuronas en esta capa son inicializadas con la salida de la capa en realimentación, la cual índica la correlación entre los patrones prototipo y el vector de entrada. Las neuronas compiten unas con otras para determinar una ganadora; después de la competición solo una neurona tendrá una salida de no cero. La neurona ganadora índica cual categoría de entrada fue presentada a la red (cada vector prototipo representa una categoría).

VII. Ejemplo en Matlab

Para reforzar la teoría que se ha expuesto en este artículo se plantea un ejercicio aplicando la función de transferencia compet de redes neuronales, la cuál nos permitirá dar a conocer la neurona ganadora y los clusters posibles.

Desarrollo de Método newc

- **Métodos de Solución** El software desarrollado para su correcto funcionamiento se requiere que el usuario seleccione si desea dar solución a través de newc o newff.

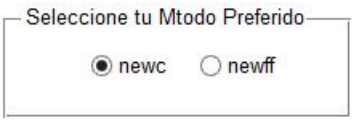

Figura 10. Método newc de resolución del problema.

- **Datos entrada de la neurona newc** En esta pantalla tenemos que ingresar algunos datos como son: número de neuronas, los vectores de entrada, número de iteraciones, número de clusteres para cada neurona y valores máximos y mínimos, estos valores se los puede elegir por defecto o tambien digitarlos.

Figura 11. Ingreso de Datos Pantalla No 1.

Figura 12. Ingreso de Datos Pantalla No 2.

- **Resolver con newc** Una vez que se tenga resueltos los dos pasos anteriores se procede a procesar los datos ingresados dando click en el botón.

Figura 13. Botón que ejecuta las acciones del método newc.

Luego de que el programa realice las operaciones establecidas a través del métdo newc, se procede a presentar los resultados; los valores y las neuronas ganadoras.

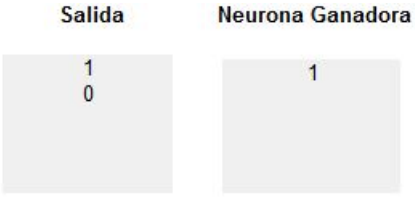

Figura 14. Neuronas ganadoras.

Ahora se procede a plotear los puntos de los clusters, los puntos ingresados por el usuario y las neuronas.

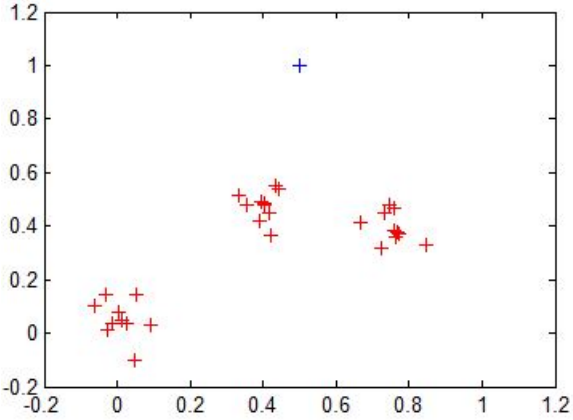

Figura 15. Gráfico inicial con valores de entrada.

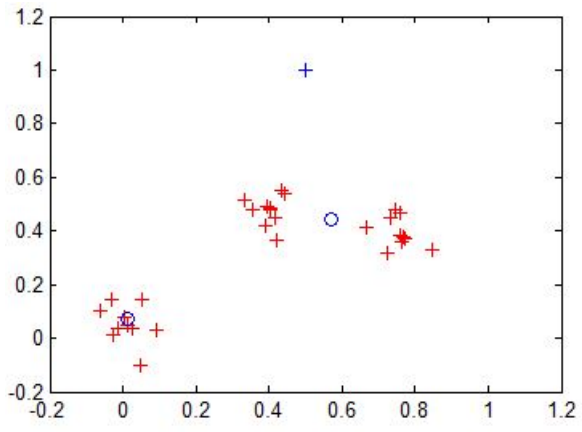

Figura 16. Gráfico final con neuronas resultantes.

Desarrollo de Método newff

- **Métodos de Solución** Si se decide dar solución al ejemplo por el método newff, realizamos la siguiente acción.

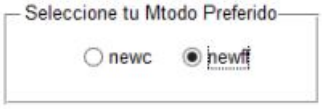

Figura 17. Método newff de resolución del problema.

- **Datos de Entrada de la Neurona** Este método tambien requiere de valores de entrada tales como: matriz de entrada, los valores máximos y mínimos, número de capas y la función de entrenamiento.

Figura 18. Ingreso de matriz.

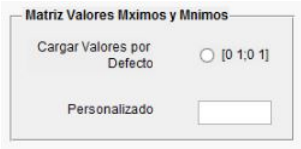

Figura 19. Ingreso de valores máximos y mínimos.

Figura 20. Ingreso de capas, neuronas y función de entrenamiento.

- **Resolver con newff** Luego de realizar todos los procesos con los datos de entrada se tiene pertinencia para poder dar solución al problema

Figura 21. btnnewff.

Luego se presentan los valores de las neuronas ganadoras y su respectiva gráfica de los puntos de los clusters, puntos ingresados por el usuario y neuronas. Dando solución al problema eficazmente.

- **Código del botón Compet**

```
function btnCompet_Callback(hObject, eventdata, handles)
    std_dev = 0.05; % desviacion estandar de cada cluster
    p = str2num(handles.txtPuntos); % numero de puntos/ cluster
    cl = 3; % numero de clusters clusters
    z = [0 1; 0 1]; % limites de los clusters
    S = nngenc(z, cl, p, std_dev);
    A=handles.txtEntradaUno;% recibe el vector 1
    B=handles.txtEntradaDos;% recibe el vector 2
    valor= [A num2str(B)];%concatena A con B en una sola matriz
    valor1=str2num(valor);% lo convierte en entero
    set(handles.text5,'String',num2str(valor1));% pone la matriz en text5
    % recibe el numero de neuronas
    numero_neuronas=str2num(handles.txtNumeroNeuronas);
    % iguala las neuronas al mismo numero de clusters
    set(handles.txtClusters,'String',handles.txtNumeroNeuronas);
    %si es 1 pone valores por defecto si no a eleccion del ususario
    if handles.rbtnMinMax==1
       m=[0 1;0 1];
    else
        m=str2num(handles.m1)
    end
    %crea la nueva rede competitiva
    % recibe como parametro 1 los valores maximos y minimos
    % el numero de neuronas
    netct=newc(m,numero_neuronas,.1);
    %grafica de los valores iniciales
    axes(handles.axes1);
    plot (S(1,:), S(2,:),'+r');
    %prueba asignandoles el peso con el valor de uno
```

Figura 22. Código Boton Compet No. 1

```
%prueba asignandoles el peso con el valor de uno
w = netct.IW{1};
%grafica de nuevo
axes(handles.axes1);
plot (S(1,:), S(2,:),'+r');
axes(handles.axes1);
hold on;
circles = plot(w(:,1),w(:,2),'ob');
hold on;
%añade los vectores de entrada
x = valor1;
%grafica de nuevo
plot (x(1,:), x(2,:), '+b')
%numero de veces que se va entrenar
netct.trainParam.epochs=str2num(handles.txtIteraciones);
netct=train(netct,S);
w = netct.IW{1};
delete(circles);
axes(handles.axes2);
plot (S(1,:), S(2,:),'+r');
hold on;
plot (w(:,1), w(:,2), 'ob');
%simulacion de la red
y = sim(netct, x)
result=vec2ind(y)
hold on;
plot (x(1,:), x(2,:), '+b')
%muestra en las cajas de texto la neurona que gano
set(handles.lblSalida,'String',num2str(y));
set(handles.lblNeuronaGanadora,'String',num2str(result));
```

Figura 23. Código Boton Compet No. 2

- **Código del botón Newc**

```
function btnNewff_Callback(hObject, eventdata, handles)
    %número de capas
    global vectorCapas;
    %función compet
    global funcion;
    %función de entrenamiento
    global fe;

    %1 cuando la función toma valores por defecto 0,1
    if handles.rbtnMinMax==1
       m=[0 1;0 1];
    else
        m=str2num(handles.m1);
    end
    %vectores de entrada
    A=handles.txtEntradaUno;
    B=handles.txtEntradaDos;
    %crea la matriz
    valor= [A num2str(B)];
    %conversión a numero
    valor1=str2num(valor);
    %se crea la red newff parámetros
    %número máximo y mínimo, número de neuronas por capa, función de
    %transferencia compet y función de entrenamiento
    net=newff(m,str2num(vectorCapas), {'compet','compet'},fe);
    %set(handles.txtNumeroNeuronas,'String');
    %p recibe la ntriz de entrada
     P = valor1;
```

Figura 24. Código Boton Newc No. 1

```
    %p recibe la ntriz de entrada
     P = valor1;
     %salidas deseadas
     T = [0 1 1 0];
    %entrenamiento visto en pantalla
    net.trainParam.show = 50;
    %Tasa de aprendizaje
    net.trainParam.lr = 0.05;
    %número de iteraciones para el entrenamiento
    net.trainParam.epochs =str2num(handles.txtIteraciones);
    %Objetivo de rendimiento
    net.trainParam.goal = 1e-3;
    %gráfica del aprendizaje
    axes(handles.axes2);
    plot (P(1,:), P(2,:), '+b')
    hold on;
    % siguiente entrenamiento la función
    %recibe de parámetro la red las entradas, y las salidas
    net1 = train(net, P, T);
    %nuevos pesos
    w = net1.IW{1,1};
    %gráfica final
    axes(handles.axes2);
    plot (w(:,1), w(:,2), 'or');
```

Figura 25. Código Boton Newc No. 2

VIII. Conclusiones

- Las redes competitivas utilizan un aprendizaje no supervisado.
- Reducción de tiempo en resolución de procesos.
- Optimización de recursos informáticos tales como: memoría, disco y procesador.

Referencias

[1] G. E. Pablo, .Aplicaciones de las Redes Neuronales en las Finanzas", pp. 24, 03/05/2014. [En linea]. Disponible en:http://eprints.ucm.es/6767/1/0205.pdf

[2] B. Alfonso, .Aprendizaje supervisado en redes neuronales", 03/05/2014. [En linea]. Disponible en: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/aprendizaje-supervisado-en-redes-neuronales.htm

[3] B. Alfonso. .Aprendizaje no supervisado", 03/05/2014. [En linea]. Disponible en: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/aprendizaje-no-supervisado-en-redes-neuronales.htm

[4] Redes Competitivas", 03/05/2014. [En linea]. Disponible en: http://medicinaycomplejidad.org/pdf/redes/Competitivas.pdf

BIOGRAFÍA

Elivar Largo Ríos Estudiante de la Carrera de Ingiería en Sistemas, Décimo Módulo.
Loja, Ciudad Loja - Ecuador, 2014.

Mario Richar León Ramón , estudiante de la Carrera de Ingiería en Sistemas, Décimo Módulo.
Loja, Ciudad Loja - Ecuador, 2014.