$$\begin{cases} w_i = \sum_{j=1}^{n+1} a_{ij} z_j \ , \ i = 1,\cdots,n+1 \\ \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2,\cdots,T \end{cases} \tag{3.5}$$

La primera expresión en (3.5) proporciona las dependencias de los pesos y el umbral de las nuevas variables libres $z_1, z_2, \cdots, z_{n+1}$. La segunda expresión en (3.5) proporcionan las relaciones de dependencia lineal entre las variables libres $z_1, z_2, \cdots, z_{n+1}$ y las variables dependientes $z_{n+2}, z_{n+3}, \cdots, z_T$.

De aquí al asignarle valores a las variables libres, de la primera expresión de (3.5) se pueden determinar los valores de los pesos y el umbral entre la capa de entrada y una neurona de la capa oculta, es decir, la ecuación de un hiperplano separador.

Como el problema radica en determinar entre las soluciones del sistema (3.3), una que tenga la mayor cantidad de componentes positivos para las variables auxiliares, se trabajará con las últimas $T-(n+1)$ filas de la matriz asociada al sistema (3.5), es decir, con el sistema

$$\sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2,\cdots,T \tag{3.6}$$

La matriz de este sistema de ecuaciones tiene la forma

$$\begin{bmatrix} a_{n+2,1} & a_{n+2,2} & \cdots & a_{n+2,n+1} & 1 & 0 & \cdots & 0 \\ a_{n+3,1} & a_{n+3,2} & \cdots & a_{n+3,n+1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{T,1} & a_{T,2} & \cdots & a_{T,n+1} & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.7}$$

y se denominará matriz de dependencia lineal.

Entonces los problemas (P3.1) y (P3.2) se transforman respectivamente en

$$\begin{cases} max \sum_{i=T_1+1}^{T} \operatorname{sgn}(z_i) \\ s.a \ \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2,\cdots,T \\ \quad z_i > 0 \ , \ i = 1,\cdots,T_1 \end{cases} \tag{P3.3}$$

y

$$\begin{cases} max \sum_{i=1}^{T_1} \operatorname{sgn}(z_i) \\ s.a \ \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2, \cdots, T \\ \quad z_i > 0 \ , \ i = T_1 + 1, \cdots, T \end{cases} \tag{P3.4}$$

El principio de solución de estos problemas será: realizar movimientos en el espacio de las variable auxiliares de forma tal que manteniendo con valores positivos los valores de las variables auxiliares correspondientes a una clase podamos aumentar el número de variables auxiliares positivas de la otra clase.

Para lograr que todas las variables auxiliares correspondientes a una misma clase sean positivas, se realizará un intercambio entre variables libres y variables dependientes mediante transformaciones elementales en la matriz (3.7).

El objetivo de este intercambio es obtener una columna, asociada a una variable libre , que tenga la mayor cantidad de elementos negativos, ya que al asignarle valores positivos a todas las variables libres, excepto a la variable libre correspondiente a esta columna, entonces existe un número positivo q, tal que si se le asigna a esta variable valores mayores que q, todas las variables dependientes con coeficiente negativo en esta columna toman valores positivos.

Para tener un control sobre las variables libres y variables dependientes, se define un vector de índices $\mathbf{U} = \big(u(1), u(2), \cdots, u(T)\big)$, donde las primeras $n+1$ componentes están asociadas a los subíndices de las variables libres y las restantes a los subíndices de las variables dependientes, además la posición de cada componente asociada a una variable dependiente, determina la fila que le corresponde a esta variable en la matriz de dependencia lineal según la siguiente regla:

$$\left.\begin{cases} z_{u(n+2)} \rightarrow \text{ fila } 1 \\ z_{u(n+3)} \rightarrow \text{ fila } 2 \\ z_{u(T)} \ \rightarrow \text{ fila } T-(n+1) \end{cases}\right\}$$

Supóngase que se ha seleccionado la columna $u(s)$, asociada a una variable

libre $z_{u(s)}$, $(1 \le s \le n+1)$, para aumentar en ella la cantidad de elementos negativos.

Para realizar las transformaciones en la matriz de dependencia lineal se buscará, entre las otras columnas asociadas a las restantes variables libres, elementos pivotes que aumenten la cantidad de elementos negativos en la columna $u(s)$.

3.4.2 Análisis para la selección del elemento pivote.

Supóngase que el elemento pivote lo se está buscando en una columna $u(r)$, $(r \neq s)$ y se denotará por $a_{k,u(r)}$.

Para la selección de los pivotes, es necesario analizar los cuatro casos siguientes :

1. $a_{k,u(s)} < 0$ y $a_{k,u(r)} > 0$
2. $a_{k,u(s)} > 0$ y $a_{k,u(r)} < 0$
3. $a_{k,u(s)} < 0$ y $a_{k,u(r)} < 0$
4. $a_{k,u(s)} > 0$ y $a_{k,u(r)} > 0$

Caso 1: $a_{k,u(s)} < 0$ y $a_{k,u(r)} > 0$

Se mantienen con valores negativos :

- El nuevo valor $\tilde{a}_{k,u(s)}$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$ y $a_{i,u(r)} < 0$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Se transforman a negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \ge 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| < \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se mantienen positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$ y $a_{i,u(r)} > 0$
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} > 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \geq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Se transforman a positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \leq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Caso 2: $a_{k,u(s)} > 0$ y $a_{k,u(r)} < 0$

Se mantienen con valores negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$ y $a_{i,u(r)} < 0$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Se transforman a negativos :

- El nuevo valor $\tilde{a}_{k,u(s)}$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| < \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se mantienen positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$ y $a_{i,u(r)} > 0$
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} > 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \geq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se transforman a positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} > 0$ y

$$\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \leq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$$

Caso 3: $a_{k,u(s)} < 0$ y $a_{k,u(r)} < 0$

Se mantienen con valores negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$ y $a_{i,u(r)} \geq 0$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Se transforman a negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| < \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se mantienen positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$ y $a_{i,u(r)} < 0$
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} > 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se transforman a positivos :

- El nuevo valor $\tilde{a}_{k,u(s)}$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \leq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Caso 4: $a_{k,u(s)} > 0$ y $a_{k,u(r)} > 0$

Se mantienen con valores negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$ y $a_{i,u(r)} \geq 0$.

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Se transforman a negativos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| < \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se mantienen positivos :

- El nuevo valor $\tilde{a}_{k,u(s)}$.
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} \geq 0$ y $a_{i,u(r)} < 0$
- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} > 0$, $a_{i,u(r)} > 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| > \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$.

Se transforman a positivos :

- Todos los nuevos valores $\tilde{a}_{i,u(s)}$ que cumplan $a_{i,u(s)} < 0$, $a_{i,u(r)} < 0$ y $\left|\frac{a_{iu(s)}}{a_{iu(r)}}\right| \leq \left|\frac{a_{ku(s)}}{a_{ku(r)}}\right|$

Como se puede apreciar, cuando se utilizan pivotes de los casos 1 y 2, en la columna $u(s)$ mantienen su signo todos aquellos elementos que tenían el signo igual al de su compañero en la columna $u(r)$, por lo que solamente interesan las variaciones que sufran los elementos de la columna $u(s)$ que tienen signo contrario al de su compañero en la columna del elemento pivote.

Para los casos 3 y 4, ocurre lo contrario, es decir, se mantienen del mismo tipo en la columna $u(s)$, todos aquellos elementos que tenían el signo contrario al de su compañero en la columna $u(r)$. Aquí interesan los cambios que sufran los elementos en la columna $u(s)$, que tienen signo contrario al de su compañero en la columna $u(r)$.

3.4.3 Transformación de la matriz de dependencia lineal.

Analicemos como resolver el problema (P3.3). El objetivo es transformar la matriz (3.7) de forma tal que se obtenga una columna, asociada a una variable libre correspondiente a la clase 1, que tenga la menor cantidad posible de coeficientes no negativos y además, que todos estos coeficientes estén en las filas de las variables dependientes correspondientes a patrones de la clase 2.

Primero, se selecciona una columna $u(s)$ de la matriz de dependencia lineal, asociada a una variable libre de la clase 1, que tenga la mayor cantidad de elementos negativos en las filas correspondientes a variables dependientes de la clase 1.

Si en la columna $u(s)$ existen elementos, asociados a las variables dependientes de la clase 1, con valores mayores o iguales a cero, significa que hay patrones mal clasificados de esta clase y por tanto, primero se debe transformar a negativos esos elementos, para luego tratar de aumentar la cantidad de patrones bien clasificados de la clase 2.

De aquí que el procedimiento se divide en dos fases. En la primera fase se convierten a negativos todos los elementos de la columna $u(s)$, asociados a variables dependientes de la clase 1 y en la segunda fase, manteniendo negativos estos elementos, se trata de aumentar la cantidad de negativos asociados a variables dependientes de la clase 2.

Después de haber seleccionado la columna $u(s)$, se determinan las cantidades de componentes mayores o iguales a cero correspondientes a las variables dependientes de ambas clases. Estos valores los denotamos por k_1 y k_2 respectivamente.

Nótese que si k_1 y k_2 son iguales a cero, el conjunto de patrones es linealmente separable.

Si $k_1 > 0$ se realizan intercambios entre las variables libres y las dependientes de forma tal que este valor se convierta en cero.

Analicemos primero el caso, cuando buscamos elementos pivotes de los tipos 1 y 2 para convertir k_1 en cero. Este algoritmo está desarrollado en el procedimiento

3.1.1 del apéndice A.

Como se vio anteriormente, aquí solo es necesario enfocarse hacia aquellos elementos que tengan el signo contrario al de su compañero en la columna $u(s)$.

Para facilitar este proceso, se determina un vector **B** con componentes

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases},$$

$$(i = 1, \cdots, T-n-1)$$

y para todo $b_i \neq 0$, se calcula $c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|$.

Para que un pivote seleccionado aumente la cantidad de elementos negativos en la columna $u(s)$, asociados a variables de la clase 1, debe cumplirse que :

- $c_k < c_i,\ \forall i : b_i = 1,\ u(i+n+1) \leq T_1$ (3.8)
- $\exists j : c_k > c_j,\ b_j = -1,\ u(j+n+1) \leq T_1$ (3.9)

Además, en el caso de pivotes del tipo 1, puede ser seleccionado un pivote asociado a una variable de la clase 1, si el valor de c_k es un mínimo único entre los valores de c_i de las restantes variables de esta clase, para las cuales $b_i = 1$.

En el caso de pivotes del tipo 2, además, hay que analizar el caso, cuando se selecciona un pivote asociado a una variable de la clase 1, para el cual se cumple solamente la condición (3.8), ya que en este caso su compañero en la columna $u(s)$ se transforma en negativo.

De aquí se puede concluir que :

1. Los pivotes del tipo 1 pueden ser :
   a) Un elemento $a_{k,u(r)}$ correspondiente a una variable dependiente de la clase 1, para el cual, el valor de c_k es un mínimo único entre los restante valores de c_i de las otras variables de esta clase, que tienen asociado un valor de $b_i = 1$.
   b) Un elemento $a_{k,u(r)}$ correspondiente a una variable dependiente de la clase 2, para el cual, el valor de c_k es menor que todos los valores de c_i para las variables de la clase 1, que tienen asociado un valor de $b_i = 1$.

2. Los pivotes del tipo 2 pueden ser :

   a) Un elemento $a_{k,u(r)}$ correspondiente a una variable dependiente de la clase 1, que cumpla solamente con la condición (3.8).

   b) Un elemento $a_{k,u(r)}$ que cumpla con las dos condiciones.

Analicemos ahora, el caso cuando se buscan elementos pivotes de los casos 3 y 4. Aquí solo es necesario enfocarse hacia aquellos elementos que tengan el mismo signo que su compañero en la columna $u(s)$. El algoritmo para este caso se encuentra desarrollado en el procedimiento 3.1.2 del apéndice A.

En estos casos, se determina un vector **B** con componentes

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases},$$

$$(i = 1, \cdots, T-n-1)$$

y para todo $b_i \neq 0$, se calcula $c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|$.

Para que el pivote seleccionado aumente la cantidad de elementos negativos en la columna $u(s)$, asociados a variables de la clase 1, debe cumplirse que :

- $c_k < c_i$, $\forall i : b_i = -1$, $u(i+n+1) \leq T_1$ (3.10)
- $\exists j : c_k > c_j$, $b_j = 1$, $u(j+n+1) \leq T_1$ (3.11)

Nótese además, que no puede ser seleccionado un pivote asociado a una variable de la clase 1 para el caso de pivotes del tipo 3.

Entre los elementos candidatos a pivote se seleccionará aquel, para el cual se obtenga el mayor valor de l_1, donde l_1 es la cantidad de elementos de la columna $u(s)$, asociados a variables de la clase 1, que se transforman en negativos.

Cuando ya se ha seleccionado el elemento pivote $a_{k,u(r)}$, para realizar el intercambio entre una variable libre y una variable dependiente en la matriz de dependencia lineal, se divide la fila k por $a_{k,u(r)}$ y se convierten en cero todos los restantes elementos de la columna $u(r)$. Además, se realiza un intercambio entre las componentes $u(r)$ y $u(k+n+1)$ del vector **U** y se actualiza el valor de $k_1 \leftarrow k_1 - l_1$.

Para el proceso de intercambiar variables se utiliza el procedimiento 3.3 del apéndice A.

Todo este proceso se repite hasta que $k_1 = 0$ y está desarrollado en el procedimiento 3.4.1 del apéndice A.

Cuando $k_1 = 0$, se pasa a verificar si se puede aumentar la cantidad de elementos negativos asociados a variables dependientes de la clase 2.

Aquí, el objetivo es aumentar la cantidad de elementos negativos asociados a variables de la clase 2, teniendo como restricción que deben mantenerse negativos, los elementos asociados a variables de la clase 1.

Esto lleva a que es necesario realizar algunas modificaciones a los procedimientos de búsqueda de los elementos pivotes.

El principio de funcionamiento de estos procedimientos será buscar pivotes que mantengan bien clasificadas a las variables de la clase 1 y para las variables de la clase 2, sea positiva la diferencia entre la cantidad de elementos que se transforman a negativos y la cantidad de elementos que se transforman a positivos.

Entre los elementos candidatos a pivote se seleccionará aquel, para el cual se obtenga el mayor valor de la diferencia $l = l_1 - l_2$ para variables asociadas a la clase 2, donde l_1 es la cantidad de elementos de $u(s)$ que se transforman a negativos y l_2 es la cantidad de elementos de $u(s)$ que se transforman a positivos.

En los pivotes de los casos 1 y 2, para garantizar que se mantengan negativos los elementos asociados a variables de la clase 1, primero se determina $c_{m1} = min\{c_i\}$ entre los valores de i que cumplen $u(i+n+1) \le T_1$, $b_i = 1$.

En estos casos los candidatos a pivote serán :

1. $a_{m1+n+1,u(r)}$, si c_{m1} es un mínimo único.
2. $a_{i+n+1,u(r)}$, si $c_i < c_{m1}$.

Nótese que, como la condición 2 solo se cumple para elementos asociados a variables de la clase 2 y que tengan el signo contrario al de su compañero en la columna $u(s)$, se reduce considerablemente el espacio de búsqueda. El algoritmo para la selección del elemento pivote está desarrollado en el procedimiento 3.5.1 del

apéndice A.

En la selección del pivote de los casos 3 y 4, para garantizar que se mantengan negativos los elementos asociados a variables de la clase 1, primero se determina $c_{m1} = min\{c_i\}$ entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = -1$.

En estos casos los candidatos a pivote serán todos aquellos elementos $a_{i+n+1,u(r)}$ para los cuales se cumple que $c_i < c_{m1}$.

Como esta condición solo se cumple para elementos asociados a variables de la clase 2 y que tengan el signo igual al de su compañero en la columna $u(s)$, se reduce considerablemente el espacio de búsqueda. El algoritmo para la selección del elemento pivote está desarrollado en el procedimiento 3.5.2 del apéndice A.

Cuando ya se ha seleccionado el elemento pivote $a_{k,u(r)}$, se realizan los intercambios entre una variable libre $z_{u(r)}$ y una variable dependiente $z_{u(k+n+1)}$ en la matriz de dependencia lineal y entre las componentes $u(r)$ y $u(k+n+1)$ del vector $\mathbf{U}$. Además, se actualiza el valor de $k_2 \leftarrow k_2 - l$.

El proceso de disminución de k_2 termina cuando ya no existen pivotes que mejoren la solución o cuando $k_2 = 0$. El algoritmo para disminuir el valor de k_2 está desarrollado en el procedimiento 3.7.1 del apéndice A.

Si como resultado de estas transformaciones, se encuentra una columna que tenga todos sus coeficientes negativos, es decir, $k_1 = 0$ y $k_2 = 0$, entonces el conjunto de patrones es linealmente separable y no tiene sentido tratar de resolver el problema (P3.4).

Si $k_2 > 0$, se pasa a resolver el problema (P3.4). En este caso se sigue el mismo procedimiento, con la diferencia de que la columna $u(s)$, que es seleccionada para una variable de la clase 2 y las restricciones de que sean todos negativos se les imponen a los coeficientes de la columna $u(s)$, asociados a las variables dependientes de esta clase.

Los algoritmos para transformar las matrices de dependencia lineal de los problemas (P3.3) y (P3.4) están desarrollados en los procedimientos 3.8.1 y 3.8.2 respectivamente, que se encuentran el apéndice A.

Entre las soluciones de los problemas (P3.3) y (P3.4), se selecciona la que separe mayor cantidad de patrones.

En las dos matrices finales, los valores de k_2 representan la cantidad de patrones mal clasificados. Para determinar, cuantos patrones de la clase 2 se lograron separar mediante la solución de (P3.3) y cuantos de la clase 1 se lograron separar mediante la solución de (P3.4), se calculan $p_1 = T - T_1 - k_2$ y $p_2 = T_1 - k_2$ respectivamente. De estos dos valores se determina $max\{p_1, p_2\}$ y se selecciona la matriz de dependencia lineal del problema para el cual se obtuvo el valor máximo.

3.4.4 Obtención de los valores de las variables auxiliares, pesos y umbral.

Ahora es necesario asignarle valores a las variables auxiliares, para luego determinar los valores de los pesos y el umbral, asociados al hiperplano separador.

Aquí, se seguirá el siguiente procedimiento para la asignación de valores a las variables auxiliares.

Primeramente, se calcularán los valores de las variables dependientes mediante las relaciones de dependencia lineal

$$z_{u(i)} = -\sum_{j=1}^{n+1} \tilde{a}_{i,u(j)}\, z_{u(j)} \ , \ (i = n+2, \cdots, T),$$

asignando a las variables libres $z_{u(s)} = 0$, $z_{u(j)} = 1$, $(j = 1, \cdots, n+1; j \neq s)$, luego para todas las variables dependientes con valores negativos y que tienen asociado un coeficiente negativo en la columna $u(s)$, se determina $q = \max_i \left\{ \left| \frac{z_{u(i)}}{a_{i,u(s)}} \right| \right\}$ y se le asigna a

$$z_{u(s)} = \begin{cases} \lceil q \rceil \ , \text{ si } q \notin \mathbf{N} \\ \lceil q \rceil + 1, \text{ si } q \in \mathbf{N} \end{cases}.$$

Finalmente, se le asignan a las variables dependientes, los valores

$$z_{u(i)} \leftarrow z_{u(i)} - \tilde{a}_{i,u(s)} z_{u(s)}, \ (i = n+2, \cdots, T)$$

De esta forma se logra que todas las variables libres y las variables dependientes que tienen asociado un coeficiente negativo en la columna $u(s)$ de la matriz de dependencia lineal, tomen valores positivos. Este algoritmo se encuentra desarrollado en el procedimiento 3.9 del apéndice A.

Para calcular los pesos y el umbral, se sustituyen los valores calculados de las z_i en la primera expresión del sistema de ecuaciones (3.5). Esto nos proporciona la ecuación de un hiperplano separador.

Si el valor de k_2 es mayor que cero, entonces existen patrones mal clasificados, por lo que uno de los dos subconjunto contiene patrones de las dos clases. A este subconjunto se le aplica el procedimiento descrito.

El algoritmo termina cuando, en el proceso de división se obtiene un subconjunto de patrones linealmente separable.

3.4.5 Resumen del algoritmo para determinar los hiperplanos separadores.

El algoritmo descrito se puede resumir en el siguiente procedimiento, que determina los valores de la matriz de pesos y umbrales entre la capa de entrada y la capa oculta.

Procedimiento 3.1: *Cálculo de hiperplanos separadores.*

Paso 1 : Ordenar el conjunto de entrenamiento de forma tal que primero se encuentren todos los patrones de la clase1.

Paso 2 : Determinar la matriz asociada al sistema de ecuaciones (3.4).

Paso 3 : Transformar esta matriz a la matriz asociada al sistema de ecuaciones (3.5).

Paso 4 : Extraer la matriz (3.7) de relaciones de dependencia lineal entre las variables auxiliares.

Paso 5 : Aplicar el procedimiento 3.8.1. Si $k_2 > 0$, ir al paso 6, en caso contrario, ir al paso 8.

Paso 6 : Aplicar el procedimiento 3.8.2.

Paso 7: Seleccionar $p = max\{p_1, p_2\}$. Si $p = p_1$, seleccionar la matriz de dependencia lineal del paso 5, en caso contrario, seleccionar la del paso 6.

Paso 8 : Aplicar procedimiento 3.9

Paso 9 : Determinar los valores de los pesos y el umbral a partir de la primera expresión del sistema de ecuaciones (3.5).

Paso 10 : Determinar si un subconjunto contiene patrones de las dos clases.

- Si $p = p_1$ y $k_2 > 0$
  - Eliminar los patrones X_i, para los cuales se cumple que $\tilde{a}_{i,u(s)} < 0$, $u(i) > T_1$
  - Actualizar $T \leftarrow T - p_1$
- Si $p = p_2$
  - Eliminar los patrones X_i, para los cuales se cumple que $\tilde{a}_{i,u(s)} < 0$, $u(i) \leq T_1$
  - actualizar $T \leftarrow T - p_2$, $T_1 \leftarrow T_1 - p_2$
- Si $k_2 > 0$ regresar al paso 1, en caso contrario parar.

Al concluir este procedimiento, se tienen todas las neuronas de la capa oculta con sus parámetros correspondientes.

3.4.6 Calculo de los pesos y el umbral entre la capa oculta y la capa de salida.

Si ahora, se le presenta a una red, como la que se muestra en la figura 19, todos los patrones de entrenamiento $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})^t \in R^n$, $(i = 1,...,T)$, se obtiene que con las imágenes $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})^t \in R^m$, $(i = 1,...,T)$ se puede formar un conjunto de entrenamiento $\{(Y_1, d_1),(Y_2, d_2),\cdots,(Y_T, d_T)\}$ que es linealmente separable.

Si a este nuevo conjunto de entrenamiento formado por las imágenes de los patrones originales, se le aplica el procedimiento 3.1 descrito anteriormente, al resolver el problema (P3.3) se obtiene un hiperplano separador de la forma $Y^t.V - v_{m+1} = 0$, que divide a este conjunto en las dos clases C_1 y C_2.

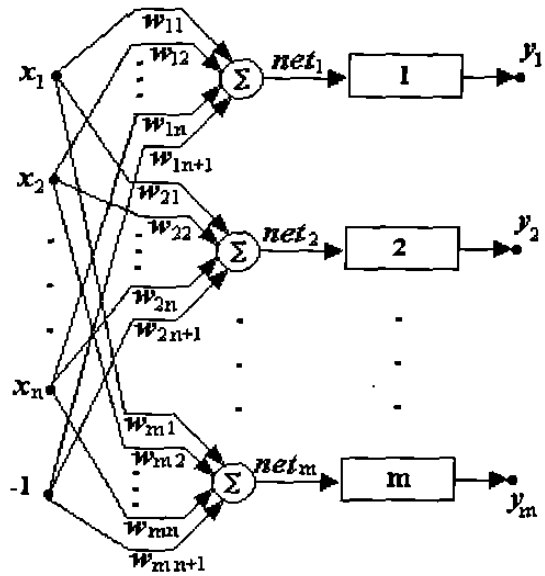

Figura 19 : Red neuronal transformadora a un conjunto linealmente separable.

De aquí se tiene, que es posible completar la red, como la que se muestra en la figura 20, la cual clasifica correctamente al conjunto original de patrones.

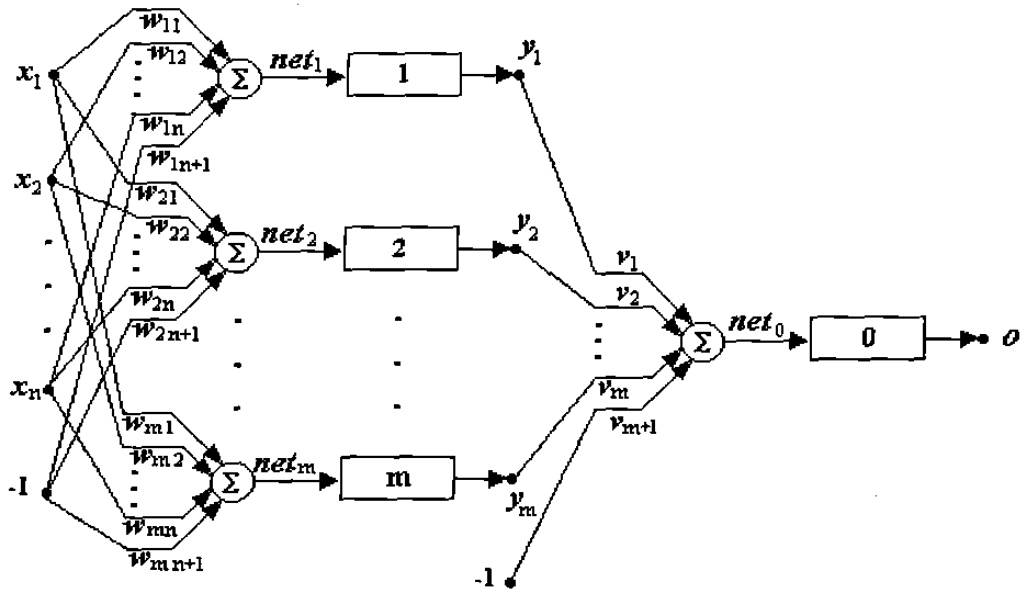

Figura 20 : Red neuronal clasificadora con funciones discretas de activación.

El proceso para determinar los valores de los pesos y el umbral entre las neuronas de la capa oculta y la neurona de la capa de salida se resume en el siguiente procedimiento.

Procedimiento 3.2: *Cálculo de los pesos y el umbral entre la capa oculta y la capa de salida.*

Paso 1: Formar el conjunto de entrenamiento $\{(Y_1,d_1),(Y_2,d_2),\cdots,(Y_T,d_T)\}$, donde

$Y_i=(y_{i1},y_{i2},\cdots,y_{im})^t$, $(i=1,...,T)$,

$$y_{ij}=\operatorname{sgn}\left(\sum_{k=1}^{n}x_{ik}w_{jk}-w_{j,n+1}\right),\ (j=1,\cdots,m\ ;\ i=1,\cdots,T).$$

Paso 2: Aplicar el procedimiento 3.1 al conjunto $\{(Y_1,d_1),(Y_2,d_2),\cdots,(Y_T,d_T)\}$.

3.5 Conclusiones.

En este capítulo, a partir de un análisis de la estructura de los espacios de patrones y de pesos, se desarrolló un algoritmo que nos permite obtener una red neuronal de tres capas con funciones discretas de activación en las neuronas, la cual clasifica correctamente un conjunto de patrones en dos clase o categorías.

Esto representa un nuevo método de diseño de redes neuronales para la clasificación, ya que los otros métodos reportados en la literatura se basan en el método clásico de aprendizaje del perceptrón discreto o en alguna de sus modificaciones.

En la primera fase del algoritmo presentado, en cada iteración se obtiene una neurona de la capa oculta con sus parámetros correspondientes. Al finalizar esta fase se obtiene una matriz de pesos entre la capa de entrada y la capa oculta.

En la segunda fase, se forma un conjunto de aprendizaje con las imágenes de los patrones originales al pasar por la capa oculta y se le aplica una sola vez el mismo procedimiento que al conjunto de aprendizaje original. Esto proporciona un vector de pesos entre la capa oculta y la capa de salida.

Se ha comprobado, además, que en el peor de los casos, el número de neuronas que se obtiene por este algoritmo es igual a (T-1), donde T es la cardinalidad del conjunto de entrenamiento, el cual coincide con el valor de la cota superior dado en [3,4].

CAPITULO 4

ENTRENAMIENTO DE UNA RED NEURONAL PARA LA CLASIFICACION CON FUNCIONES CONTINUAS DE ACTIVACION

4.1 Introducción.

El algoritmo de retropropagación del error (*backpropagation*) [103,104] está considerado como uno de los logros de mayor importancia en la Teoría de las Redes Neuronales de propagación hacia adelante.

Este algoritmo está diseñado para entrenar redes con varias capas de neuronas ocultas, pero está demostrado que una sola capa oculta es suficiente para que una red neuronal de propagación hacia adelante pueda aproximar de forma uniforme cualquier función continua soportada sobre un hipercubo unitario [15,33].

Es conocido que este algoritmo de entrenamiento presenta algunas deficiencias y esto ha sido el centro de atención de las críticas de diferentes autores.

Entre las deficiencias fundamentales que presenta, están:

1. Es desconocido el número de neuronas que se necesitan en la capa oculta para que el algoritmo de aprendizaje sea convergente.

2. La cantidad de ciclos de entrenamiento depende de los valores iniciales de los pesos, los cuales son seleccionados de forma aleatoria.
3. Alto número de operaciones en el proceso de entrenamiento de la red neuronal, tanto en su fase hacia adelante en el cálculo de las salidas, como en su fase hacia atrás en la modificación de los pesos.

Entre las deficiencias, la tercera es la que ha sido tratada más ampliamente en la literatura [7,19,81,99]. En la mayoría de los trabajos se pretende disminuir el número de operaciones en la fase de entrenamiento, pero todos se basan en realizar modificaciones al algoritmo de retropropagación del error.

En el presente capítulo, a partir de la distribución de pesos obtenida en el capítulo anterior, se desarrollará un algoritmo que aborda el problema de entrenamiento de manera diferente a como se ha hecho anteriormente.

La ventaja fundamental de este algoritmo es que reduce considerablemente el numero de operaciones del proceso de entrenamiento, en comparación con las diferentes variantes del algoritmo de retropropagación del error.

4.2 Formulación del problema.

En este capítulo se estudia el problema de clasificación de un conjunto de patrones en dos clases, o categorías, C_1 y C_2 por una red neuronal con funciones continuas de activación del tipo

$$f(net) = \frac{2}{1+e^{-\lambda net}} - 1 \tag{4.1}$$

Denotemos por

$$\{(X_1,d_1),(X_2,d_2),\cdots,(X_T,d_T)\}$$

el conjunto de aprendizaje, donde $\mathbf{X}_i = (x_{i1},x_{i2},\cdots,x_{in}) \in \mathbf{R}^n$, $(i = 1,...,T)$ representan los patrones de entrenamiento y d_i es la salida deseada del patrón X_i, siendo, además $d_i = \begin{cases} 1, \text{ si } \mathbf{X}_i \in C_1 \\ -1, \text{ si } \mathbf{X}_i \in C_2 \end{cases}$,

El problema de entrenamiento se puede formular como:

Determinar una matriz de pesos

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1} \\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} & w_{m,n+1} \end{bmatrix}$$

entre la capa de entrada y la capa oculta, y un vector de pesos

$$\mathbf{V} = \left[v_1 \, , \, v_2 \, , \, \cdots \, , \, v_m \, , \, v_{m+1} \right]$$

entre la capa oculta y la capa de salida, de forma tal que

$$E = \frac{1}{2} \sum_{i=1}^{T} \left(d_i - o_i \right)^2 < E_{max} \quad ,$$

donde

$$o_i = f\left[\sum_{j=1}^{m} v_j f\left(\sum_{k=1}^{n} x_{ik} w_{jk} - w_{j,n+1} \right) - v_{m+1} \right]$$

$$f(net) = \frac{2}{1 + e^{-\lambda net}} - 1$$

4.3 Análisis del problema planteado.

El primer problema que se presenta en el entrenamiento de esta red, es que se desconoce la cantidad de neuronas de la capa oculta, pero en [93] está demostrado que, si una red con funciones discretas de activación en las neuronas puede clasificar correctamente un conjunto de patrones, entonces una red que tenga la misma estructura, pero que utilice funciones continuas de activación en las neuronas, también puede hacerlo, usando el algoritmo de retropropagación del error como algoritmo de aprendizaje.

Esto significa que para determinar la cantidad de neuronas en la capa oculta de esta red con funciones continuas de activación, se puede utilizar el algoritmo desarrollado en el capítulo anterior para clasificar correctamente el conjunto de patrones, cuando las funciones de activación en las neuronas son discretas.

Este algoritmo, además de la cantidad de neuronas en la capa oculta, proporciona una matriz de pesos entre la capa de entrada y la capa oculta

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1} \\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} & w_{m,n+1} \end{bmatrix}$$

y un vector de pesos entre la capa oculta y la capa de salida

$$\mathbf{V} = \left[v_1 , v_2 , \cdots , v_m , v_{m+1} \right]$$

que pueden ser usados como una memoria inicial para el problema de entrenamiento con funciones continuas de activación. Esto salva la dificultad de la inicialización aleatoria de los pesos.

Este algoritmo proporciona una red entrenada, como se muestra en la figura 21, que clasifica correctamente el conjunto de patrones, utilizando en las neuronas funciones de activación bipolares discretas, es decir, funciones del tipo

$$f(net) = \begin{cases} 1 , \text{ si } net > 0 \\ -1 , \text{ si } net \leq 0 \end{cases}.$$

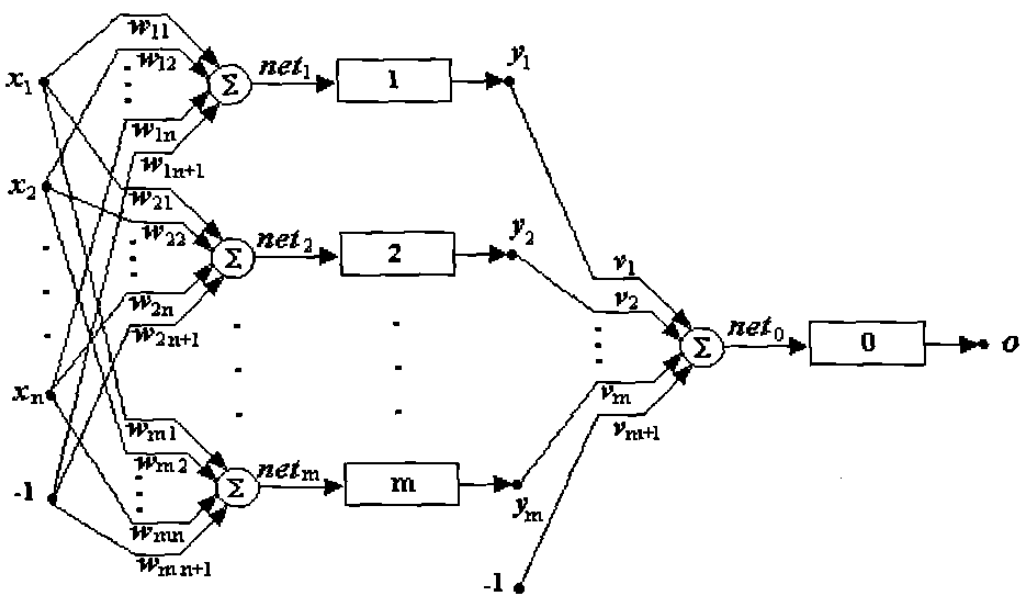

Figura 21 : Red neuronal con funciones discretas de activación.

En esta red, las salidas de la capa oculta para cada patrón de entrenamiento forman un vector con componentes del conjunto $\{-1, 1\}$, por lo que cada uno de estos vectores es un vértice del hipercubo $[-1, 1]^m$, es decir, se obtiene una matriz

$$Y=\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1m} \\ y_{21} & y_{22} & \cdots & y_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ y_{T1} & y_{T2} & \cdots & y_{Tm} \end{bmatrix},$$

donde cada fila $Y_i=\left(y_{i1},y_{i2},\cdots,y_{im}\right)$, $(i=1,\cdots,T)$ representa la imagen de cada patrón de entrenamiento X_i, al pasar por la capa oculta. Además, como la red clasifica correctamente al conjunto de patrones, entonces se cumple que

$$E=\frac{1}{2}\sum_{i=1}^{T}\left(d_i-o_i\right)^2=0,$$

donde

$$o_i=\operatorname{sgn}\left[\sum_{j=1}^{m}v_j\operatorname{sgn}\left(\sum_{k=1}^{n}x_{ik}w_{jk}-w_{j,n+1}\right)-v_{m+1}\right],$$

lo que significa que estas imágenes son linealmente separables por el hiperplano

$$v_1y_1+v_2y_2+\cdots+v_my_m-v_{m+1}=0. \tag{4.2}$$

Para resolver el problema de entrenamiento planteado, se cambiarán las funciones discretas de activación en la red de la figura 21, por funciones continuas del tipo (4.1), obteniéndose una red como la que se muestra en la figura 22.

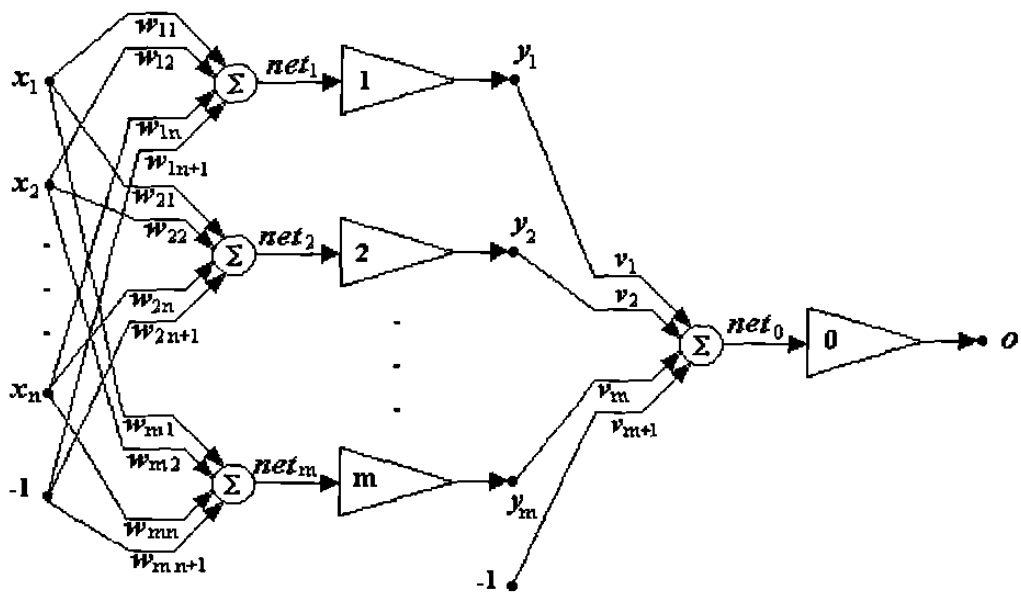

Figura 22 : Red neuronal con funciones continuas de activación.

Estas funciones tienen la propiedad de que $|f(net)| < 1$, y además cuando su argumento tiende a $\pm\infty$, su valores se aproximan a ± 1.

Por tanto, al utilizar este tipo de función en las neuronas de la capa oculta, los valores de y_{ij} cumplen que $|y_{ij}| < 1$, $i = 1,\cdots,T$, $j = 1,\cdots,m$, y en consecuencia los vectores

$$Y_i = (y_{i1}, y_{i2}, \cdots, y_{im}) \ , \ i = 1,\cdots,T \tag{4.3}$$

serán puntos interiores del hipercubo $[-1, 1]^m$ y a pesar de que pueden ser linealmente separables, generalmente no lo son por el hiperplano (4.2). Esto significa que para patrones mal clasificados, la diferencia $(d_i - o_i)$ en valor absoluto, siempre será mayor que 1, es decir, $|d_i - o_i| > 1$.

El problema planteado puede ser resuelto usando el algoritmo de retropropagacion del error, pero al existir patrones mal clasificados, se cumple que $|d_i - o_i| > 1$ y esto ocasiona un incremento en la cantidad de ciclos de entrenamiento. De aquí se tiene, que el proceso de modificación de los pesos, para lograr que el error sea menor que un valor prefijado con antelación, resulta demasiado costoso desde el punto de vista computacional.

Debido a esto se buscará otra forma de realizar el entrenamiento de la red, que no sea mediante el algoritmo de retropropagación del error.

De todo lo antes expuesto se puede resumir que el problema de entrenamiento planteado consiste en, a partir de la distribución de pesos que se obtuvo para la red de la figura 21, entrenar una red, como la que se muestra en la figura 22, que mantenga la misma estructura, pero que utilice en las neuronas funciones de activación bipolares continuas del tipo (4.1).

El algoritmo de entrenamiento que se propone, disminuye de forma considerable tanto la cantidad de operaciones en un ciclo, así como la cantidad de ciclos a realizar. Además no presenta la dificultad de que los pesos sean inicializados de forma aleatoria.

4.4 Descripción del algoritmo de solución.

Resolver el problema de entrenamiento planteado, significa determinar un punto $(o_1, o_2, \cdots, o_T)$ sobre la superficie de error que sea interior a una bola en $\mathbf{R}^T$ con centro en el punto $(d_1, d_2, \cdots, d_T)$ y de radio igual a $\sqrt{2E_{max}}$.

Para buscar este punto, el algoritmo de retropropagación del error modifica la posición de los hiperplanos separadores.

Para cambiar la posición de los hiperplanos separadores, es necesario modificar sus parámetros, es decir, los pesos de la red. Esto nos lleva a que en el proceso de entrenamiento es necesario modificar un total de $[m(n+1)+m+1]$ parámetros y esta modificación se debe realizar un gran número de veces para llegar a la solución.

En el algoritmo de solución propuesto, el principio de funcionamiento es completamente diferente. En lugar de modificar las posiciones de los hiperplanos separadores, lo que se modifica son las posiciones de las imágenes de los patrones al pasar por la capa oculta.

En las funciones continuas de activación aparece un parámetro λ que generalmente se le asigna un valor unitario, pero si se aumenta el valor de este parámetro en las neuronas de la capa oculta se pueden aproximar las imágenes de los patrones de entrenamiento a los vértices del hipercubo $[-1, 1]^m$ y convertir este conjunto de imágenes $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})$, $i = 1, \cdots, T$ en un conjunto linealmente separable por el hiperplano (4.2).

El algoritmo de solución propuesto, se basa en las propiedades, antes mencionadas, de las funciones de activación bipolares continuas y en considerar en estas funciones a λ como variable y no a la entrada neta, es decir,

$$f(\lambda) = \frac{2}{1+e^{-\lambda net}} - 1 \ .$$

Por lo tanto, en lugar de ajustar todos los pesos de la red, lo que se hace es modificar un parámetro λ en cada neurona.

Ahora, se considera que cada neurona j de la red tiene asociada una variable λ_j. Al aumentar los valores de λ_j en cada neurona de la capa oculta, las imágenes

de los patrones de entrenamiento se aproximan a los vértices del hipercubo $[-1, 1]^m$ y estos valores de λ_j se incrementarán hasta que el conjunto de las imágenes (4.3) sea separable por el hiperplano (4.2).

Analicemos ahora, como realizar el incremento de los valores de λ_j en las neuronas de la capa oculta para que el conjunto de las imágenes sea linealmente separable por el hiperplano (4.2).

El objetivo del procedimiento es mover las imágenes de los patrones mal clasificados hacia el otro semiespacio definido por el hiperplano separador, por lo que para incrementar los valores de las λ_j se utilizarán solamente los patrones mal clasificados. Además, como es deseable obtener valores pequeños de λ_j, el movimiento se realizará en la dirección del vector normal al hiperplano (4.2) y la magnitud del movimiento se determinará a partir de la imagen más alejada del hiperplano.

En primer lugar, se construye un conjunto de entrenamiento con las imágenes $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})$, $(i = 1, \cdots, T)$ de la forma $\{(Y_1, d_1), (Y_2, d_2), \cdots, (Y_T, d_T)\}$. Entonces, para los patrones mal clasificados se cumple que

$$d_i(v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1}) \leq 0 .$$

Como la distancia de los puntos $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})$, $(i = 1, \cdots, T)$ al hiperplano (4.2), se determina por la expresión

$$h_i = \frac{|v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1}|}{\sqrt{v_1^{\,2} + v_2^{\,2} + \cdots + v_m^{\,2}}},$$

para determinar el patrón mal clasificado que está más alejado del hiperplano, entre los valores de i que cumplen

$$d_i(v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1}) \leq 0 ,$$

se calcula

$$\min_i \{d_i(v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1})\} .$$

Supóngase que el valor del mínimo se alcanzó para el patrón $Y_k = (y_{k1}, y_{k2}, \cdots, y_{km})$. Como el movimiento se realiza en la dirección del vector normal al hiperplano, se deben incrementar las componentes del vector Y_k según la fórmula

$$y_{kj}^1 = y_{kj}^0 + \alpha\, v_j, \tag{4.4}$$

donde el superíndice 0 está asociado a los valores iniciales y el superíndice 1 a los valores modificados.

Denotemos por $Y_k^1 = (y_{k1}^1, y_{k2}^1, \cdots, y_{km}^1)$ a la proyección de Y_k^0 sobre el hiperplano. Para determinar el valor de α se utilizarán las coordenadas de este punto proyección.

Como este punto está sobre el hiperplano, entonces satisface su ecuación, es decir,

$$v_1 y_{k1}^1 + v_2 y_{k2}^1 + \cdots + v_m y_{km}^1 - v_{m+1} \equiv 0.$$

De aquí, si se sustituyen los valores de y_{kj}^1 por su expresión en (4.4), se obtiene

$$v_1(y_{k1}^0 + \alpha\, v_1) + v_2(y_{k2}^0 + \alpha\, v_2) + \cdots + v_m(y_{km}^0 + \alpha\, v_m) - v_{m+1} \equiv 0,$$

de donde es posible determinar el valor de α por la fórmula

$$\alpha = -\frac{v_1 y_{k1}^0 + v_2 y_{k2}^0 + \cdots + v_m y_{km}^0 - v_{m+1}}{v_1^{\,2} + v_2^{\,2} + \cdots + v_m^{\,2}}. \tag{4.5}$$

Ahora, si con este valor de α se calculan

$$y_{ij}^1 = \begin{cases} y_{ij}^0 + |\alpha|\, v_j, & \text{si } d_i = 1 \\ y_{ij}^0 - |\alpha|\, v_j, & \text{si } d_i = -1 \end{cases}, \quad (i = 1, \cdots, T;\ j = 1, \cdots, m),$$

entonces todos los patrones $Y_i^1 = (y_{i1}^1, y_{i2}^1, \cdots, y_{im}^1)$, $(i = 1, \cdots, T)$ estarán bien clasificados por el hiperplano (4.2), menos el patrón Y_k^1, que estará exactamente sobre el hiperplano.

Ahora es necesario determinar los valores de los parámetros λ_j que le corresponden a este valor de α.

Los valores de y_{ij} en la red de la figura 22, se determinan por la expresión $y_{ij} = \frac{2}{1+\exp(-net_{ij}\lambda_j)} - 1$, de donde es posible determinar qué valores deben tomar los parámetros λ_j para que las imágenes iniciales Y_i^0, $(i = 1,\cdots,T)$ se transformen en Y_i^1, $(i = 1,\cdots,T)$.

Según esta fórmula, los valores de y_{kj}^1 deben cumplir

$$y_{kj}^1 = \frac{2}{1+\exp(-net_{kj}\lambda_j^1)} - 1 \quad , \tag{4.6}$$

por lo que se puede despejar el valor de λ_j^1 de la forma

$$\lambda_j^1 = -\frac{1}{net_{kj}} \ln\left[\frac{2}{y_{kj}^1 + 1} - 1\right] \tag{4.7}$$

Ahora para que todos los patrones estén bien clasificados, es suficiente asignar

$$\lambda_j \leftarrow \lceil \lambda_j^1 \rceil + 1 \tag{4.8}$$

Si en las funciones de activación de las neuronas de la capa de la red de la figura 22 se le asignan a los parámetros λ_j los valores calculados por la expresión (4.8), se logra que $|d_i - o_i| < 1$. Si ahora se aumenta el valor de λ_0 en la neurona de salida, los valores de o_i se aproximan a los d_i y el valor del error se puede hacer tan pequeño como se quiera.

Una primera aproximación del valor de λ_0 se determina a partir del patrón Y_k, que es el más próximo al hiperplano y por lo tanto es el que aporta una mayor contribución al valor del error, es decir, determinamos cuánto debe valer λ_0 para que $\frac{1}{2}(d_k - o_k)^2 = E_{max}$.

A partir de esta relación se obtiene que

$$o_k = d_k\left(1 - \sqrt{2E_{max}}\right)$$

y por otro lado, según la expresión para el calculo de o_k, se tiene que

$$o_k = \frac{2}{1+\exp(-neto_k \lambda_0)} - 1$$

De la igualdad de estas dos expresiones para o_k, es posible despejar λ_0 y se obtiene

$$\lambda_0 = -\frac{1}{neto_k} \ln\left[\frac{2}{d_k\left(1-\sqrt{2E_{max}}\right)+1} - 1\right] \tag{4.9}$$

Ahora, realizando la asignación

$$\lambda_0 \leftarrow \lceil \lambda_0 \rceil + 1 \tag{4.10}$$

se garantiza que la contribución al error total de cada patrón por separado, sea menor que E_{max}, es decir,

$$\frac{1}{2}\left(d_i - o_i\right)^2 < E_{max}, \text{ para } i = 1,\cdots,T.$$

Este valor de λ_0 no garantiza que se cumpla la condición de error total

$$\frac{1}{2}\sum_{i=1}^{T}\left(d_i - o_i\right)^2 < E_{max}, \tag{4.11}$$

pero en la solución de problemas específicos, cuando este valor no es el óptimo, se encuentra muy cercano a él.

Después de calcular el valor de λ_0 por la expresión (4.9) y ajustarlo por (4.10), se pasa a verificar si se cumple la condición de error total (4.11). Para ello se calculan $o_i = \frac{2}{1+\exp(\lambda_0 neto_i)} - 1$, $(i = 1,\cdots,T)$ y $E = \frac{1}{2}\sum_{i=1}^{T}\left(d_i - o_i\right)^2$. Si $E \geq E_{max}$, entonces se incrementa $\lambda_0^{k+1} = \lambda_0^k + 1$ y de nuevo se calculan las salidas o_i, $(i = 1,\cdots,T)$ y el valor del error total. Este proceso se repite hasta que se cumpla la condición de error (4.11).

De todo lo expuesto, se puede resumir que algoritmo propuesto consta de dos fases o etapas de entrenamiento. En una primera fase se modifican los valores de los parámetros λ_j en las neuronas de la capa oculta , para que el conjunto de imágenes de los patrones de entrenamiento sea linealmente separable por el hiperplano (4.2) y se mantiene fijo el valor de λ_0 en la neurona de la capa de salida.

En la segunda etapa se incrementa el valor de λ_0 hasta que se alcance un valor del error menor que un valor prefijado con antelación, manteniendo fijos los valores de las λ_j en las neuronas de la capa oculta. Inicialmente, el incremento de λ_0 se realiza a partir del patrón que aporta el sumando de mayor valor al error total y si con este valor de λ_0, aún no se cumple la condición de error, se le dan incrementos unitarios, según la expresión $\lambda_0^{k+1} = \lambda_0^{k} + 1$.

Este algoritmo recibe como datos :

a) La matriz de patrones de entrenamiento

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{T1} & x_{T2} & \cdots & x_{Tn} \end{bmatrix},$$

donde cada fila representa un elemento del conjunto de patrones.

b) El vector de salidas deseadas para los patrones de entrenamiento

$$\mathbf{d} = \left[d_1 , d_2 , \cdots , d_T \right]^t$$

c) La matriz de pesos entre la capa de entrada y la capa oculta

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1} \\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} & w_{m,n+1} \end{bmatrix}$$

y el vector de pesos entre la capa oculta y la capa de salida

$$\mathbf{V} = \left[v_1 , v_2 , \cdots , v_m , v_{m+1} \right]$$

que se obtienen del algoritmo desarrollado en el capítulo 3.

4.5 Resumen del algoritmo.

Procedimiento 4.1: *Entrenamiento con funciones continuas de activación.*

Paso 1 : Inicializar los valores de $\lambda_j = 1$, $j = 0, \cdots, m$, el valor de $k = 0$ y el valor de E_{max}.

Paso 2 : Calcular la matriz de entrada a las neuronas de la capa oculta,

$$net = \begin{bmatrix} net_{11}\ net_{12} \cdots net_{1m} \\ net_{21}\ net_{22} \cdots net_{2m} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ net_{T1}\ net_{T2} \cdots net_{Tm} \end{bmatrix}$$

donde cada fila $\mathbf{net}_i = \left[net_{i1}, net_{i2}, \cdots, net_{im}\right]$ representa el vector de entradas a la capa oculta para el patrón de entrenamiento X_i y sus componentes se determinan de la forma

$$net_{ij} = \sum_{k=1}^{n} x_{ik} w_{jk} - w_{j,n+1}\ ,\ \ i = 1, \cdots, T\ ;\ \ j = 1, \cdots, m$$

Paso 3 : Calcular la matriz de salida de la capa oculta

$$Y = \begin{bmatrix} y_{11}\ y_{12} \cdots y_{1m} \\ y_{21}\ y_{22} \cdots y_{2m} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ y_{T1}\ y_{T2} \cdots y_{Tm} \end{bmatrix},$$

donde cada fila $Y_i = \left(y_{i1}, y_{i2}, \cdots, y_{im}\right)$, $(i = 1, \cdots, T)$ representa el vector de salidas de la capa oculta para el patrón de entrenamiento X_i y sus componentes se determinan por la expresión

$$y_{ij} = \frac{2}{1 + \exp(\lambda_j net_{ij})} - 1\,.$$

Paso 4 : Entre los valores de i para los que se cumple

$$d_i\left(v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1}\right) \le 0\,,$$

determinar el índice k, para el cual se alcanza

$$\min_i \left\{ d_i\left(v_1 y_{i1} + v_2 y_{i2} + \cdots + v_m y_{im} - v_{m+1}\right) \right\}$$

Paso 5 : Si $k \neq 0$, calcular

- $\alpha = -\dfrac{v_1 y_{k1}^0 + v_2 y_{k2}^0 + \cdots + v_m y_{km}^0 - v_{m+1}}{v_1^{\,2} + v_2^{\,2} + \cdots + v_m^{\,2}}$
- $y_{kj}^1 = y_{kj}^0 + \alpha\, v_j\,,\ (j = 1, \cdots, m)$

- $\lambda_j^1 = -\frac{1}{net_{kj}} \ln\left[\frac{2}{y_{kj}^1 + 1} - 1\right]$
- $\lambda_j \leftarrow \lceil \lambda_j^1 \rceil + 1$

Paso 6 : Calcular

- $y_{ij} = \frac{2}{1 + \exp(\lambda_j net_{ij})} - 1$, $i = 1, \cdots, T$; $j = 1, \cdots, m$.
- $neto_i = \sum_{j=1}^{m} \left(v_j y_{ij}\right) - v_{m+1}$, $i = 1, \cdots, T$.

Paso 7 : Si $k = 0$

- Calcular $o_i = \frac{2}{1 + \exp\left(\lambda_0 neto_i\right)} - 1$, $(i = 1, \cdots, T)$
- Entre todos los valores de i , determinar el índice k para el cual se alcanza $\min_i |o_i|$

en caso contrario

- Calcular $o_k = \frac{2}{1 + \exp\left(\lambda_0 neto_k\right)} - 1$

Paso 8 : Si $|d_k - o_k| > \sqrt{2E_{max}}$, entonces $\lambda_0 = -\frac{1}{neto_k} \ln\left[\frac{2}{d_k\left(1 - \sqrt{2E_{max}}\right) + 1} - 1\right]$

Paso 9 : Asignar $\lambda_0 \leftarrow \lceil \lambda_0 \rceil + 1$

Paso 10 : Calcular

- $o_i = \frac{2}{1 + \exp\left(\lambda_0 neto_i\right)} - 1$, $(i = 1, \cdots, T)$
- $E = \frac{1}{2} \sum_{i=1}^{T} \left(d_i - o_i\right)^2$

Paso 11 : Si $E > E_{max}$, asignar $\lambda_0 \leftarrow \lambda_0 + 1$, ir al paso 10 , en caso contrario parar.

4.6 Conclusiones.

En el presente capítulo se desarrolló una nueva forma de entrenar una red neuronal de propagación hacia adelante con funciones continuas de activación en las neuronas, a partir de una distribución inicial de pesos que se obtiene de una red que tiene la misma estructura, pero con funciones discretas de activación, la cual clasifica correctamente un conjunto de patrones de entrenamiento, en dos clase o categorías.

Con el procedimiento presentado se logra reducir considerablemente la cantidad de operaciones a realizar en el proceso de aprendizaje de una red neuronal en comparación con las diferentes variantes del algoritmo de retropropagación del error.

Con el algoritmo descrito, se reduce de forma considerable la cantidad de operaciones en un ciclo de entrenamiento, la cantidad de ciclos a realizar y la cantidad de parámetros a modificar en el proceso de aprendizaje.

En este procedimiento los valores de las entradas netas net_{ij} a las neuronas de la capa oculta son calculados una sola vez, mientras que el algoritmo de retropropagación del error tiene que hacerlo, como mínimo, en cada ciclo de entrenamiento.

La cantidad total de operaciones de multiplicación de este proceso es del orden de $[n.m.T]$, donde n es la dimensión de los patrones de entrenamiento, m es la cantidad de neuronas en la capa oculta y T es la cardinalidad del conjunto de aprendizaje.

Además, el algoritmo de retropropagación del error tiene que modificar, en su forma más eficiente, un total de $[m.(n+1)+m+1]$ parámetros de la red en cada ciclo de aprendizaje, mientras que el algoritmo propuesto tiene que modificar $[m+1]$ parámetros en el primer ciclo de entrenamiento y en los ciclos restantes, si son necesarios, solamente se modifica un parámetro mediante un incremento unitario.

CAPITULO 5

DISEÑO Y ENTRENAMIENTO DE UNA RED NEURONAL PARA LA CLASIFICACION DE MEMORIA ENTERA

5.1 Introducción.

Entre las aplicaciones fundamentales de los modelos de redes neuronales de propagación hacia adelante se encuentran la clasificación y el reconocimiento de patrones. Cuando la memoria de la red está formada por valores enteros pequeños, estos modelos resultan muy atractivos desde el punto de vista práctico, ya que sus diferentes formas de implementación son sumamente económicas. Por ejemplo, cuando los pesos están restringidos a valores enteros del intervalo $[-3, 3]$, se requieren solamente de 3 bits para almacenar los pesos de la red.

El caso ideal es cuando los pesos toman solamente valores del conjunto $\{-1, 0, 1\}$, que son las llamadas redes libres de multiplicación [47], ya que para calcular la entrada neta a cada neurona no es necesario realizar operaciones de multiplicación. Además, en este caso los pesos de la red pueden ser almacenados mediante cadenas binarias de longitud 2.

Cuando el entrenamiento de la red se realiza para la clasificación de patrones en dos clases o categorías, si el conjunto de patrones es linealmente separable, entonces existe un hiperplano que separa el conjunto de patrones, y aquí se puede implementar un perceptrón (discreto o continuo) que resuelve correctamente el problema de clasificación.

Si el conjunto de patrones no es linealmente separable, el problema de aprendizaje puede ser resuelto, si se le agrega a la red una capa oculta de neuronas [15,33], pero al considerar pesos enteros el problema de entrenamiento está considerado como NP-completo [46].

Por otra parte, en el proceso de entrenamiento de una red neuronal con capas ocultas, uno de los problemas más difíciles, aún cuando los pesos pueden tomar valores reales, es el diseño de la red, es decir, determinar la cantidad de neuronas en la capas ocultas para que el algoritmo de entrenamiento sea convergente. Este problema también está considerado como un problema NP-completo [90].

Diferentes autores [16,71,72,77,84,86] han tratado el problema de entrenamiento de redes con pesos discretos. Por ejemplo, en [22] se describen tres técnicas de discretización de los pesos para redes ópticas que parten del algoritmo de retropropagación del error y en [101] se dan comparaciones empíricas de diferentes técnicas de discretización.

Los resultados más relevantes, relacionados con la capacidad de aproximación de las redes con memoria entera, son obtenidos en [47,48,49], los cuales están contenidos también en la tesis doctoral [46].

En [55,60,62] se presentan resultados de carácter teórico sobre la relación entre el número de neuronas en la capa oculta y el acotamiento de las pesos de la red.

Una mejora computacional del algoritmo de discretización descrito en [46] es presentada en [1] para valores de pesos del conjunto $\{-3,-2,-1,\ 0,\ 1,\ 2,\ 3\}$.

En ninguno de estos trabajos se trata el problema del diseño de la red, es decir, se supone conocida la cantidad de neuronas en la capa oculta. En la literatura revisada encontramos solamente un trabajo [71], donde se trata el problema del diseño de la

red para pesos binarios, pero el problema es tratado de forma diferente a como es resuelto en este capítulo.

En el presente capítulo, se estudia el problema de entrenamiento de una red con una capa oculta para clasificar un conjunto de patrones en dos clases, que no son linealmente separable. Además los pesos están restringidos a valores del conjunto {-3,-2,-1, 0, 1, 2, 3}.

El objetivo de este trabajo es desarrollar un algoritmo eficiente que resuelva, tanto el problema de diseño de la red, así como de su entrenamiento.

Aquí, el proceso de entrenamiento está dividido en dos fases. En la primera fase del algoritmo propuesto, se obtiene una red entrenada de memoria entera con funciones discretas de activación. En la segunda fase se implementan funciones continuas de activación en las neuronas de la red y mediante un proceso de modificación de un parámetro en cada función de activación se logra completar el proceso de entrenamiento.

5.2 Formulación del problema.

El problema de entrenamiento se puede formular como :

Dado el $\{(X_1,d_1),(X_2,d_2),\cdots,(X_T,d_T)\}$ el conjunto de aprendizaje, donde $X_i=(x_{i1},x_{i2},\cdots,x_{in})\in\mathbf{R}^n$, $(i=1,...,T)$ representan los patrones de entrenamiento y d_i es la salida deseada del patrón X_i , siendo, además $d_i=\begin{cases}1, \text{ si } X_i\in C_1\\ -1, \text{ si } X_i\in C_2\end{cases}$, determinar una matriz de pesos

$$W=\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & w_{1,n+1}\\ w_{21} & w_{22} & \cdots & w_{2n} & w_{2,n+1}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ w_{m1} & w_{m2} & \cdots & w_{mn} & w_{m,n+1}\end{bmatrix}$$

entre la capa de entrada y la capa oculta, y un vector de pesos

$$\mathbf{V}=\left[v_1\ ,\ v_2\ ,\ \cdots\ ,\ v_m\ ,\ v_{m+1}\right]$$

entre la capa oculta y la capa de salida, donde $w_{ij}, v_i \in Z$, $i = 1,\cdots,m$; $j = 1,\cdots,n$, de forma tal que

$$E = \frac{1}{2}\sum_{i=1}^{T}\left(d_i - o_i\right)^2 < E_{max} \quad ,$$

donde

$$o_i = f\left[\sum_{j=1}^{m} v_j f\left(\sum_{k=1}^{n} x_{ik} w_{jk} - w_{j,n+1}\right) - v_{m+1}\right]$$

$$f(net) = \frac{2}{1+e^{-\lambda net}} - 1 \tag{5.1}$$

Nótese que los umbrales de las neuronas no están restringidos a tomar valores enteros.

La primera dificultad que aparece al tratar de resolver este problema es que se desconoce el valor de m, es decir la cantidad de neuronas en la capa oculta, por lo que primero se tratará el problema del diseño de la red y luego el problema de su entrenamiento con una memoria entera.

5.3 Análisis del problema del diseño de la red.

Para determinar el valor de m, se usará un procedimiento similar al descrito en el capítulo 3, con algunas modificaciones.

En este procedimiento se parte de que, cuando el conjunto de patrones no es linealmente separable, entonces existen un grupo de desigualdades que se incumplen en el sistema

$$d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) > 0,\ i = 1,\cdots,T \tag{5.2}$$

y al introducir T variables auxiliares $z_1, z_2, \cdots, z_T$, de la forma

$$d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) - z_i = 0, \quad i = 1,\cdots,T \tag{5.3}$$

se logra una doble correspondencia entre patrón bien clasificado y variable auxiliar positiva.

Con la introducción de estas variables auxiliares, el problema se transforma en determinar las soluciones $(w_1, w_2, \cdots, w_n, w_{n+1}, z_1, z_2, \cdots, z_T)$ del sistema de ecuaciones (5.3) que contengan valores enteros para las variables w_j, $(j = 1, \cdots, n)$ y la mayor cantidad de valores positivos para las variables auxiliares correspondientes a patrones de una misma clase.

Supóngase que en el conjunto de entrenamiento se tienen T_1 patrones de la clase 1 y $T_2 = T - T_1$ patrones de la clase 2 y además, que están ordenamos de forma tal que primero se encuentran los T_1 patrones de la clase 1.

El sistema de ecuaciones (5.3) puede ser escrito como

$$d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) = z_i \ , \ (i = 1, \cdots, T) \tag{5.4}$$

y para determinar el hiperplano que separe la mayor cantidad de patrones de una misma clase, es necesario resolver los siguientes problemas

$$\begin{cases} max \sum_{i=M+1}^{T} \text{sgn}(z_i) \\ s.\ a \quad d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) = z_i, \ \ i = 1, \cdots, T \\ \qquad z_i > 0, \ \ i = 1, \cdots, M \\ \qquad w_j \in Z, \ \ j = 1, \cdots, n \end{cases} \tag{P5.1}$$

$$\begin{cases} max \sum_{i=1}^{M} \text{sgn}(z_i) \\ s.\ a \quad d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right) = z_i, \ \ i = 1, \cdots, T \\ \qquad z_i > 0, \ \ i = M+1, \cdots, T \\ \qquad w_j \in Z, \ \ j = 1, \cdots, n \end{cases} \tag{P5.2}$$

Estos problemas, por su forma, son similares a los problemas (P3.1) y (P3.2) respectivamente, pero al restringir los pesos a que puedan tomar solamente valores enteros, se convierten en problemas de una complejidad aún mayor.

5.4 Descripción del algoritmo de solución para el problema de diseño.

Para resolver estos problemas, se hará primeramente una relajación, considerando que los pesos pueden tomar valores reales, para posteriormente aplicarle los procedimientos del capítulo 3 descritos en el apéndice A.

En estos procedimientos, se realiza un intercambio entre variables libres y variables dependientes del sistema (5.4), mediante un proceso de eliminación gaussiana, de forma tal que queden como variables dependientes las componentes del vector de pesos y el umbral y pasen a ser variables libres $n+1$ de las T componentes del vector de variables auxiliares, obteniendo

$$\begin{cases} w_i = \sum_{j=1}^{n+1} a_{ij} z_j \ , \ i = 1,\cdots,n+1 \\ \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2,\cdots,T \end{cases} \tag{5.5}$$

Por comodidad de notación, en este sistema se hizo la suposición de que las primeras $n+1$ variables auxiliares $z_1, z_2, \cdots, z_{n+1}$ son ahora las variables libres

La primera expresión en (5.5) proporciona las dependencias de los pesos y el umbral de las nuevas variables libres $z_1, z_2, \cdots, z_{n+1}$. La segunda expresión en (5.5) proporcionan las relaciones de dependencia lineal entre las variables libres $z_1, z_2, \cdots, z_{n+1}$ y las variables dependientes $z_{n+2}, z_{n+3}, \cdots, z_T$.

Como la resolución del problema relajado consiste en determinar entre las soluciones del sistema (5.4), una que tenga la mayor cantidad de componentes positivos para las variables auxiliares, se puede trabajar solamente con la segunda de las expresiones en (5.5), es decir, con el sistema

$$\sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0 \ , \ i = n+2,\cdots,T \tag{5.6}$$

La matriz de este sistema de ecuaciones tiene la forma

$$\begin{bmatrix} a_{n+2,1}\, a_{n+2,2} \cdots a_{n+2,n+1}\, 1\, 0 \cdots 0 \\ a_{n+3,1}\, a_{n+3,2} \cdots a_{n+3,n+1}\, 0\, 1 \cdots 0 \\ \vdots \quad \vdots \quad \cdots \quad \vdots \quad \vdots\, \vdots \cdots \vdots \\ a_{T,1} \quad a_{T,2} \quad \cdots \quad a_{T,n+1} \quad 0\, 0 \cdots 1 \end{bmatrix} \tag{5.7}$$

Para determinar el hiperplano separador, se tiene que resolver uno de los siguientes problemas, denotados por (P5.3) y (P5.4) respectivamente.

$$\begin{cases} max \sum_{i=M+1}^{T} \operatorname{sgn}(z_i) \\ s.a \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0\ ,\ i = n+2, \cdots, T \\ \qquad z_i > 0\ ,\ i = 1, \cdots, M \end{cases} \tag{P5.3}$$

$$\begin{cases} max \sum_{i=1}^{M} \operatorname{sgn}(z_i) \\ s.a \sum_{j=1}^{n+1} (a_{ij} z_j) + z_i = 0\ ,\ i = n+2, \cdots, T \\ \qquad z_i > 0\ ,\ i = M+1, \cdots, T \end{cases} \tag{P5.4}$$

Como se puede apreciar, en estos problemas no están incluidas las variables w_j, $(j = 1, \cdots, n+1)$, pero sus valores se pueden determinar de la primera expresión en (5.5).

Si a estos problemas se les aplican los procedimientos del capítulo 3, se obtiene una matriz de pesos entre la capa de entrada y la capa oculta, pero sus elementos tendrán valores reales. Para obtener una memoria con valores enteros se realizarán algunas modificaciones a estos procedimientos.

Debido a que en [6] se muestra que cuando los pesos toman valores del conjunto $\{-3, -2, -1, 0, 1, 2, 3\}$, se puede obtener una gran diversificación de hiperplanos separadores, en el algoritmo que se propone, se considera este conjunto para los valores de los pesos.

En el proceso de discretización de los pesos se explotan los siguientes resultados:

1. Si se multiplican los valores de los pesos y el umbral por una constante positiva, el hiperplano conserva la propiedad de ser separador.
2. La ecuación del hiperplano separador no es única, es decir, se pueden determinar diferentes ecuaciones de hiperplanos separadores para obtener los mismos subconjuntos de patrones.

Utilizando el primer resultado se puede realizar un escalamiento de los pesos a que tomen valores del intervalo [-3, 3], seleccionando $w = \max_{1 \le j \le n} |w_j|$ y multiplicando los pesos y el umbral por $\frac{3}{w}$.

El segundo resultado permite tener varias variantes de discretización, por lo que se puede seleccionar entre ellas la del hiperplano que separe mayor cantidad de patrones.

Analicemos en detalle este segundo resultado. Tanto en la resolución del problema (P5.3), como en la del (P5.4), se realizan intercambios entre las variables libres y las variables dependientes del sistema (5.6) para obtener todas las variables de una clase y la mayor cantidad de variables de la otra clase con valores positivos. Esto lleva a una matriz de dependencia lineal entre las variables auxiliares, que por comodidad de notación, se supone que tiene la misma forma que (5.7).

$$\begin{bmatrix} a_{n+2,1} & a_{n+2,2} & \cdots & a_{n+2,n+1} & 1 & 0 & \cdots & 0 \\ a_{n+3,1} & a_{n+3,2} & \cdots & a_{n+3,n+1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{T,1} & a_{T,2} & \cdots & a_{T,n+1} & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.8}$$

En esta matriz, cada fila está asociada a una variable dependiente y la asociación se determina a partir de las componentes $u(i)$, $(i = n+2, \cdots, T)$ del vector de índices $\mathbf{U} = \big(u(1), u(2), \cdots, u(T)\big)$, definido en el capítulo 3.

Analicemos primero, el problema (P5.3). Cuando se le aplica a este problema el procedimiento 3.8.1 del apéndice A, se obtiene una columna $u(s)$ en la matriz (5.8) correspondiente a una variable libre de la clase 1, que tiene coeficientes negativos en las posiciones asociadas a todas las variables dependientes de la clase 1 y en la mayor cantidad de posiciones asociadas a variables dependientes de la clase 2.

Si ahora, se le asignan valores a las variables libres, de la forma descrita en el procedimiento 3.9 del apéndice A, se obtiene un hiperplano separador.

Para obtener otras ecuaciones diferentes de hiperplanos separadores que dividan al conjunto de patrones en los mismos subconjuntos, se puede variar la forma de asignar valores a las variables libres.

Como el objetivo es que se mantengan bien clasificados los mismos patrones, se determinarán los posibles intercambios entre variables libres y variables dependientes que mantengan los mismos coeficientes negativos en la columna asociada a $z_{u(s)}$. Para ello se modifica en los procedimientos 3.5.1 y 3.6.1 la condición { si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i$ } por la condición { si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow i$ }. Estos procedimientos modificados se encuentran desarrollados en el Apéndice B.

Ahora, si se le aplican a la matriz (5.8) estos procedimientos modificados, es posible encontrar otras combinaciones para las variables libres que mantienen bien clasificados a los mismos patrones.

En el caso del problema (P5.4), se procede de manera similar, con la diferencia de que las condiciones se modifican en los procedimientos 3.5.1 y 3.6.1.

Estos procedimientos proporciona diferentes matrices de dependencia lineal y si se le aplica el procedimiento 3.9 para el cálculo de los valores de las variables auxiliares, descrito en el apéndice A, se obtienen diferentes hiperplanos separadores, que dividen al conjunto de patrones en los mismos subconjuntos.

Ahora para el proceso de discretización de los pesos, primero, se realiza un escalamiento de los pesos de forma tal que $|w_j| \leq 3$. Esto se hace en cada hiperplano separador, seleccionando $w = \max_{1 \leq j \leq n} |w_j|$ y multiplicando los pesos y el umbral por $\frac{3}{w}$. Luego se aproximan los pesos al valor entero más cercano.

Al discretizar los pesos, se está cambiando la posición del hiperplano y esto puede llevar a un empeoramiento de la solución, es decir, que patrones que estaban bien clasificados, ya no lo estén.

Aquí pueden darse las siguientes dos situaciones:

1. Que en los dos subconjuntos existan patrones de las dos clases.
2. Que disminuyó la cantidad de elementos en el subconjunto que contenía patrones de una sola clase.

En la primera situación, hay que mover el hiperplano separador hasta que se logre que en uno de los subconjuntos solo existan patrones de una misma clase.

En el caso, cuando el hiperplano se obtuvo mediante la discretización de una solución del problema (P5.3), se calculan, a partir del sistema de ecuaciones (5.4), los valores de las variables auxiliares correspondientes a los patrones de la clase 1, se selecciona el menor valor negativo

$$z_k = min\{z_i : z_i \leq 0 \ , \ i = 1,\cdots,T_1\} \ ,$$

y se le asigna al umbral el valor de

$$w_{n+1}^k = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn} .$$

Con este nuevo valor de umbral se calculan los valores de las variables auxiliares correspondientes a patrones de la clase 2, se selecciona el menor valor positivo

$$z_l = min\{z_i : z_i > 0 \ , \ i = M+1,\cdots,T\}$$

y se le asigna al umbral el valor de

$$w_{n+1}^l = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{ln} .$$

Ahora, se determina

$$w_{n+1}^* = \frac{w_{n+1}^k + w_{n+1}^l}{2}$$

Cuando el hiperplano separador se obtiene de una discretización de una solución del problema (P5.4), se realiza el mismo procedimiento, con la única diferencia que se empieza por los patrones mal clasificados de la clase 2.

Con esta forma de determinar el umbral, se está desplazando el hiperplano separador en la dirección de su vector normal, de forma tal que en un semiespacio estén todos los patrones de una clase y en el otro semiespacio se encuentre la mayor cantidad de patrones de la otra clase.

En la segunda situación, se debe verificar si se puede aumentar la cantidad de elementos en el subconjunto que contiene patrones de una sola clase. Para ello, se determina en el conjunto que contiene patrones de las dos clases el menor valor positivo de las variables auxiliares z_i. Supóngase que este valor se alcanza para la variable z_k, entonces si existen variables con valores negativos que cumplan que $z_i > -z_k$, se puede aumentar la cantidad de elementos en el conjunto que contiene patrones de una sola clase.

Para agregar los patrones correspondientes a estas variables auxiliares, primero, se determina

$$z_l = min\{z_i : -z_k < z_i \leq 0\}$$

y se calcula

$$w_{n+1}^l = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{ln}.$$

$$w_{n+1}^k = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn}.$$

Luego, se determina

$$w_{n+1}^* = \frac{w_{n+1}^k + w_{n+1}^l}{2}.$$

Con este valor de umbral, se logra pasar todos los patrones mal clasificados, que cumplían con la condición $-z_k < z_i \leq 0$, al conjunto que contiene patrones de una sola clase.

Todo este proceso de discretización de los pesos y del ajuste del valor de umbral, se resumen en los procedimientos 5.3.1 y 5.3.2, desarrollados en el apéndice B.

Con estos procedimientos de discretización, es posible desarrollar un procedimiento que determine, en cada iteración, un hiperplano separador con pesos enteros que clasifica correctamente la mayor cantidad de patrones.

Este hiperplano separa al conjunto de patrones en dos subconjuntos, uno contiene patrones de una sola clase y el otro patrones de las dos clases, por lo que a este último se la aplica el procedimiento descrito. El procedimiento termina cuando se encuentre un subconjunto linealmente separable.

Al terminar el procedimiento se obtiene un conjunto de m hiperplanos separadores, con los cuales podemos formar una red neuronal, como la que se muestra en la figura 23.

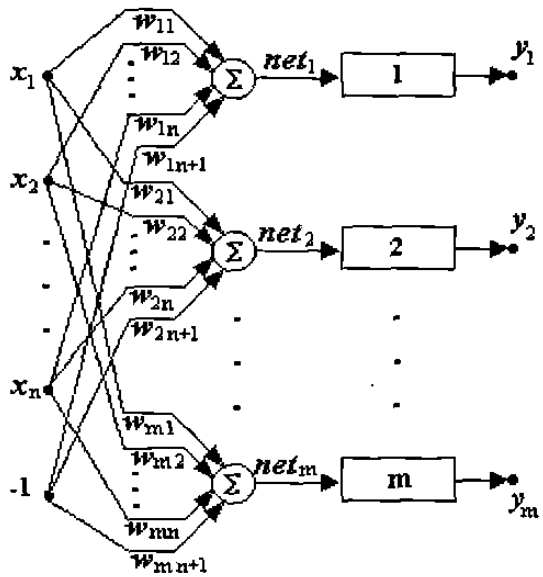

Figura 23 : Red neuronal transformadora a un conjunto linealmente separable.

Si ahora, se le presentan a esta red todos los patrones de entrenamiento $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})^t \in R^n$, $(i = 1,...,T)$, se obtiene que con las imágenes $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})^t \in R^m$, $(i = 1,...,T)$ se puede formar un conjunto de entrenamiento $\{(Y_1, d_1),(Y_2, d_2),\cdots,(Y_T, d_T)\}$ que es linealmente separable.

Si a este nuevo conjunto se le aplica el procedimiento descrito anteriormente, se obtiene un hiperplano separador con pesos enteros. Con este último hiperplano se puede completar la red neuronal con pesos enteros, como la que se muestra en la figura 24, que clasifica correctamente al conjunto original de patrones en las dos clases o categorías.

Para resolver el problema de entrenamiento con funciones continuas de activación en las neuronas de la red, se utiliza el procedimiento 4.1, desarrollado en el capítulo 4, para redes con pesos reales.

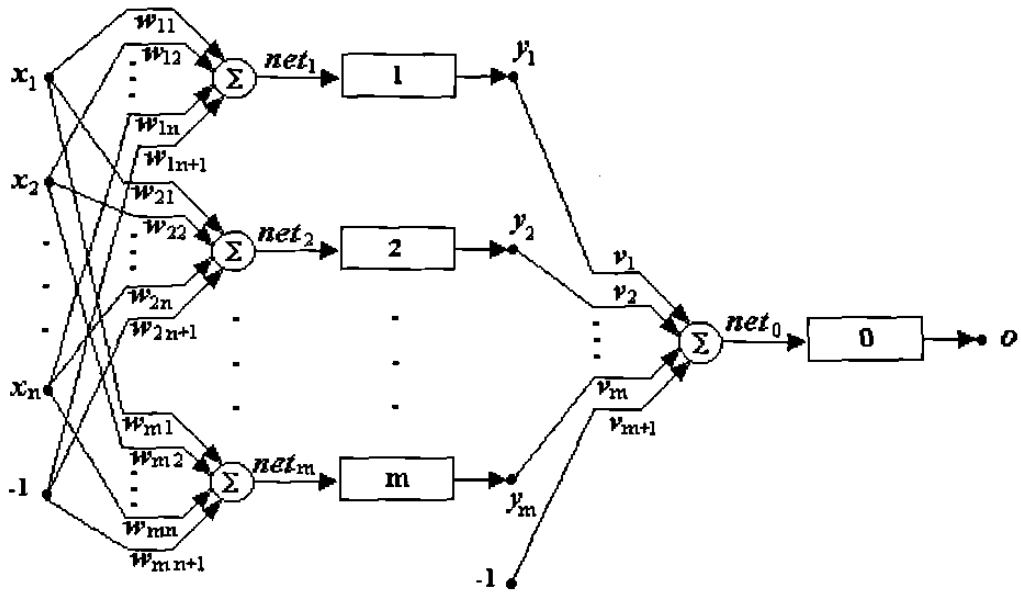

Figura 24 : Red neuronal de memoria entera con funciones discretas de activación.

5.5 Resumen del algoritmo de solución para el problema de diseño.

Procedimiento 5.4: *Obtención de una memoria entera entre la capa de entrada y la capa oculta.*

Paso 0 : Inicializar $m = 0$.

Paso 1 : En el conjunto de entrenamiento, colocar primero patrones de la clase 1.

Paso 2 : Formar el sistema de ecuaciones (5.4).

Paso 3 : Obtener el sistema de ecuaciones (5.5).

Paso 4 : Extraer la matriz de dependencia lineal (5.7).

Paso 5 : Aplicar procedimiento 3.8.1.

Paso 6 : Inicializar $r = 1$, $p = 0$.

Paso 7 : Aplicar procedimiento 3.9

Paso 8 : Calcular $w_j^m, (j = 1, \cdots, n+1)$, partir de la primera expresión del sistema de ecuaciones (5.5)

Paso 9 : Aplicar procedimiento 5.3.1.

Si $p_3 = p_1$,

- Asignar $p \leftarrow p_3$

- Asignar $\hat{W}^m \leftarrow W^m$
- ir al paso 12.

En caso contrario, si $p_3 > p$

- asignar $p \leftarrow p_3$
- asignar $\hat{W}^m \leftarrow W^m$
- ir al paso 10.

Paso 10 : Si $r = s$, asignar $r \leftarrow r+1$.

Paso 11 : Si $r \leq n+1$,

- Asignar $k \leftarrow 0$
- Aplicar procedimiento 5.1.1
- Si $k \neq 0$, entonces
  - Aplicar procedimiento 3.3
  - $r \leftarrow r+1$
  - Ir al paso 7

  en caso contrario $(k = 0)$,
  - Aplicar procedimiento 5.2.1
  - Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - $r \leftarrow r+1$
    - Ir al paso 7

    en caso contrario $(k = 0)$,
    - $r \leftarrow r+1$
    - Ir al paso 10

en caso contrario $(r > n+1)$, ir al paso 12.

Paso 12 : Aplicar procedimiento 3.8.2. Si $p \geq p_2$, ir al paso 19.

Paso 13 : Inicializar $r = 1$.

Paso 14 : Aplicar procedimiento 3.9

Paso 15 : Calcular $w_j^m, (j = 1, \cdots, n+1)$, partir de la primera expresión del sistema de ecuaciones (5.5)

Paso 16 : Aplicar procedimiento 5.3.2.

Si $p_4 > p$,

- Asignar $p \leftarrow p_4$
- Asignar $\hat{W}^m \leftarrow W^m$
- ir al paso 17.

Paso 17 : Si $r = s$, asignar $r \leftarrow r+1$.

Paso 18 : Si $r \leq n+1$,

- Asignar $k \leftarrow 0$
- Aplicar procedimiento 5.1.2
- Si $k \neq 0$, entonces
  - Aplicar procedimiento 3.3
  - $r \leftarrow r+1$
  - Ir al paso 14

  en caso contrario $(k = 0)$,
  - Aplicar procedimiento 5.2.2
  - Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - $r \leftarrow r+1$
    - Ir al paso 14

    en caso contrario $(k = 0)$,
    - $r \leftarrow r+1$
    - Ir al paso 17

en caso contrario $(r > n+1)$, ir al paso 19.

Paso 19 : Determinar si un subconjunto contiene patrones de las dos clases.

- Inicializar $k_2 = 0$.
- Si $p \approx p_3$,
  - Para cada valor de i que cumpla $\tilde{a}_{i,u(s)} < 0$, $u(i) > T_1$
    - Incrementar $k_2 \leftarrow k_2 + 1$
    - Eliminar el patrón X_i.

- Actualizar $T \leftarrow T - p_3$

- Si $p = p_4$
  - Para cada valor de i que cumpla $\tilde{a}_{i,u(s)} < 0$, $u(i) \leq T_1$
    - Incrementar $k_2 \leftarrow k_2 + 1$
    - Eliminar el patrón X_i.
  - Actualizar $T \leftarrow T - p_4$, $T_1 \leftarrow T_1 - p_4$
- Si $k_2 = 0$, entonces parar, en caso contrario,
  - Incrementar $m \leftarrow m + 1$
  - Regresar al paso 1

5.6 Conclusiones.

Con el algoritmo desarrollado en este capítulo son resueltos dos problemas: el diseño y el entrenamiento de una red neuronal con memoria entera.

En la primera fase del algoritmo propuesto, a partir de un análisis de la estructura de los espacios de patrones y de pesos, se desarrolla un procedimiento que permite calcular el número de neuronas en la capa oculta, determinando en cada iteración una neurona con sus parámetros correspondientes. Además, si el conjunto de patrones es linealmente separable el algoritmo para en la primera iteración.

Este procedimiento, además proporciona una memoria de pesos enteros, que corresponde a la solución del problema de clasificación de patrones por una red con funciones discretas de activación en las neuronas.

En la segunda fase del algoritmo, esta memoria es utilizada como la memoria inicial para el entrenamiento de una red que tiene la misma estructura, pero con funciones continuas de activación.

Aquí, en la segunda fase, en lugar de ajustar todos los pesos de la red, se modifica un solo parámetro en cada neurona, lográndose reducir considerablemente la cantidad de operaciones en el proceso de entrenamiento, en comparación con otros algoritmos que se basan en el algoritmo de retropropagación del error.

CAPITULO 6

CONCLUSIONES Y RECOMENDACIONES

En la actualidad los modelos de redes neuronales artificiales tienen gran impacto en diferentes áreas, debido a que su aplicación permite resolver una variada gama de problemas, tales como clasificación de patrones, reconocimiento de formas y voz, control y automatización de procesos, filtrado de señales, etc.

Sin embargo, a pesar de los logros obtenidos, aun existen problemas no resueltos, relacionados con el diseño, aprendizaje y funcionamiento de las redes neuronales.

Este trabajo resuelven algunos problemas que surgen en el diseño y aprendizaje de redes neuronales para la clasificación.

Cuando se tratan de implementar modelos de redes neuronales para la clasificación, generalmente el conjunto de entrenamiento no es linealmente separable, por lo que es necesario utilizar una capa oculta de neuronas. Determinar la cantidad de neuronas en esta capa oculta es un problema muy complejo y en la mayoría de los casos se determina por ensayo y error. De aquí, la importancia de desarrollar algoritmos que determinen la cantidad de neuronas en la capa oculta.

Otro problema que presentan los modelos de redes neuronales es que el proceso de entrenamiento resulta muy costoso desde el punto de vista computacional. Es

conocido que, en el caso del problema de filtrado de señales, se realiza una discretización de las señales, transformando el problema a uno de clasificación, pero es necesario utilizar funciones continuas de activación en las neuronas de la red. De aquí, la importancia del algoritmo desarrollado para entrenar una red con funciones continuas de activación, que realiza un número de operaciones muy inferior al de las diferentes variantes del algoritmo de retropropagación del error.

Cuando se utilizan modelos de redes neuronales para resolver determinado problema, es deseable que la memoria de la red este formada por valores enteros pequeños, ya que las diferentes formas de implementación resultan muy económicas. En esto radica la importancia de desarrollar algoritmos de entrenamiento que permitan obtener una red con memoria entera.

En la realización de este trabajo de investigación, aportamos algunos resultados a la Teoría de las Redes Neuronales, pero aún existen muchos problemas por resolver en este ámbito, como por ejemplo, generalizar los resultados obtenidos para redes clasificadoras en varias categorías, desarrollar procedimientos de post-procesamiento que disminuyan la cantidad de neuronas en la capa oculta y aplicar los algoritmos desarrollados a la solución de problemas reales. Todo esto puede ser objeto de estudio para futuros trabajos de investigación.

BIBLIOGRAFIA

1. Aguilar, M., Martínez, J. y Alvarez, A. Un Algoritmo de entrenamiento para Redes Neuronales con pesos enteros. Universidad Autónoma de Nuevo León, México, (1997).

2. Anderson, J. Introduction to Practical Neural Modeling. Cambridge, MA: MIT Press, (1994).

3. Attali, J. y Pages, G. Approximations of functions by a multilayer perceptron - a new approach. Neural Networks vol. 10 No. 6: 1069-1081, (1997).

4. Baldi, P. Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective. IEEE Tansactions on Neural Networks, vol. 6, No. 1 , (1995).

5. Baldi, P. y Hornik, K. Learning in linear neural networks: A survey. IEEE Transactions on Neural Networks, NN-6(4):837-858, (1995).

6. Barlett, E. Dynamic Node Architecture Learning: An Information Theoric Approach. Neural Network, Vol. 7, No. 1: 129-140, (1994).

7. Bianchini, M. Gori y M. Optimal Learning in Artifitial Neural Networks : a Theoretical View. Dipartimento di Sistemi e Informatica, Universita di Firenze, (1996).

8. Bianchini, M., Frasconi, P. y Gori, M. Learnig in Multilayer Networks used as Autoassociators. IEEE Tansactions on Neural Networks, vol. 6, No.1, (1995).

9. Brandt, R. y Lin, F. Supervised Learning in Neural Networks without Feedback Networks. Department of Electrical and Computer Engineering, Wayne State University, Detroit, (1997).

10. Campbell, C. Constructive Learning Techniques for Designing Neural Network Systems. Advanced Computing Research Centre, University of Bristol, England, (1997).

11. Campbell, C. y Pérez, C. Constructing Feedforward Neural Networks for Binary Classification Tasks. In European Symposium on Artificial Neural Networks, D. Facto Publications, Brussels: 241-246, (1995).

12. Campbell, C. y Coombes, S. Determining the optimal number of hidden nodes in a Feedforward Neural Network. Department of Engineering Mathematics, Bristol University, (1996).

13. Costa, R. y Larzabal, P. Inicialization of Supervised Training for Parametric Estimation. Neural Processing Letters vol. 9: 53-61, (1999).

14. Crespin, D. Generalized Backpropagation. Facultad de Ciencias, Universidad Central de Venezuela, (1995).

15. Cybenko, G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, **2**: 303-314, (1989).

16. Chieueh, T. y Goodman, R. Learnig Algorithms for Neural Networks with Ternary Weights. Neural network, Vol, 1: 166-172, (1988).

17. Dasgupta, B. y Sontag, E. Sample Complexity for Learning Recurrent Perceptron Mappings. DOMACS Technical Report 95-17, (1995).

18. Dasgupta, B., Siegelma, H. y Sontag, E. On the complexity of training Neural Networks with continuous activation functions. IEEE Transactions on Neural Networks, vol. 6: 1490-1504, (1995).

19. Ergezinger, S. An Accelerated Learning Algorithm for Multilayer Perceptrons: Optimization Layer by Layer. IEEE Tansactions on Neural Networks, vol. 6, No. 1 , (1995).

20. Vecci, L., Piazza, F. y Uncini, A. Learning and approximation capabilities of adaptive spline activation function neural networks. Neural Networks vol. 11 No. 2: 259-270, (1998).

21. Fiesler, E. Connectivity Maximization of Layered Neural Networks for Supervised learning. Proceeding of the International Conference on Artificial Neural Networks: 514, (1993).

22. Fiesler, E., Choudry, A. y Caulfield, H. A Weight Discretization Paradigm for Optical Neural Networks. Proceedings of the International Congress on Optical Science and Engineering, vol. SPIE-1281: 164-173, (1990).

23. Fujita, O. Statistical estimation of the number of hidden units for feedforward neural networks. Neural Networks vol. 11 No. 5: 851-859, (1998).

24. Fukushima, K. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Uneffected by Shift in Position. Biological Cybernetics 36:193-202, (1980).

25. Funahashi, K. On the approximate realization of continous mapping by neural networks. Neural Networks 2: 183-192, (1989).

26. Gallant, S. Perceptron based learning algorithms. IEEE Transactions on Neural Networks, 1 (2): 179-191, (1990).

27. Gori, M. y Tesi, A. On the problem of local minima in backpropagation. IEEE Trans. on Pattern Analysis and Machine Intelligence, **PAMI-14**: 76-86, (1992).

28. Gori, M. y Maggini, M. Optimal Convergence of on-line Backpropagation. Dipartimento di Sistemi e Informatica, Universita di Firenze, (1994).

29. Haykin, S. Neural Networks: A Comppehensive Foundation. Prentice Hall. 2nd Edition. (1998).

30. Hlavackova, K. Dependence of the rate of approximation in a feedforward network on its activation function. Neural Network World, 2: 163-171, (1996).

31. Hlavácková, K. Y Verleysen, M. Syntesis of Feedforward Neural Networks using Splines for Approximation of Functions. Institute of Computer Science, Academy of Science of Czech Republic, Technical Report No. 651, (1995).

32. Hopfield, J. Neural Networks and Phisical Systems with emergent collective computational abilities. Proccedings of the National Academy of Sciences of the U.S.A. 79: 2554-2558, (1982).

33. Hornik, K., Stinchcombe, M. y White, H. Multilayer feedforward network are universal approximators. Neural Networks, **2**: 359-366, (1989).

34. Hornik, K., Stinchcombe, M., White, H. y Auer, P. Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives. Neural Computation, vol. 6: 1262-1275, (1994).

35. Hornik, K. Some new results on neural network approximation. Neural Networks, vol. 6: 1073-1087, (1993).

36. Hornik, K. Approximation capabilities of multilayer feedforward networks. Neural Networks, vol. 4: 251-257, (1991).

37. Hornik, K. Functional learning and approximation in artificial neural networks. Neural Network World, vol. 1: 257-266, (1991).

38. Hornik, K. Some new results on neural network approximation. Neural Networks, vol. 6: 1069-1072, (1993).

39. Huang, Sh. y Huang, Y. Bounds on the number of hidden neurons in multilayer perceptrons. IEEE Transactions on Neural Networks, vol. 2, No.1,pp. 47-55, (1991).

40. Humpert, B. Improving backpropagation with a new error function. Neural Networks vol. 7 No. 8: 1191-1192, (1994).

41. Ito, Y. Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. Neural Networks, vol. 4: 385-394, (1991).

42. Ito, Y. Approximation Capability of Layered Neural Networks with Sigmoid Units on Two Layers. Neural Computation, vol. 6: 1233-1243, (1994).

43. Kainen, P., Kurková, V. y Vogt, A. Approximation by Neural Networks is not Continuous. Neurocomputing, 29: 45-56, (1999).

44. Karnin, E. Simple procedure for pruning Backpropagation Trained Neural Networks. IEEE Tansaction on Neural Networks, vol. 1, No. 2 , (1990).

45. Katsuura, H. y Sprecher, D. Comutational Aspects of Kolmogorov's Superposition Theorem. Neural Networks, vol. 7: 455-461, (1994).

46. Khan, A. Feedforward Neural Networks with Constrained Weights. Thesis for the degree of doctor of Philosophy,University of Warwick, Department of Engineering, (1996).

47. Khan, A. Multiplier-free Feedforward Neural Network is a Universal Approximator in C(R). Proceedings of the 2nd IEEE National Multi-topic Conference (INMIC'97), Islamabad, Pakistan: 33-36, (1997).

48. Khan, A. y Hines, E. Integer-Weight Neural Net. Electron. Letters, vol. 30, No.15: 1237-1238, (1994).

49. Khan, A. y Wilson, R. Integer-Weight Approximation of Continuous-Weight Multilayer Feedforward Net. Proccedings of the IEEE International Conference on Neural Networks, Washington, DC, (1996).

50. Kohonen, T. Self-Organization and Associative Memory. New York: Springer-Verlag. (1988).

51. Kohonen, T. Self-organized formation of topologically correct feature maps. Biological Cybernetics 43: 59-69, (1982).

52. Koiran, P. On the complexity of approximating mapping using feedforward networks. Neural Networks, vol. 6: 649-653, (1993).

53. Kurkova, V. y Kainen, P. Equivalent weight vectors in perceptron type networks. Neural Network World, vol. 6: 685-692, (1992).

54. Kurkova, V., Kainen, P. y Kreinovich, V. Dimension-independent rates of approximation by Neural Networks and variation with respect to half-spaces. In WC Neural Networks, vol. 1: 54-57, (1995).

55. Kurkova, V. Kainen, P. y Kreinovich, V. Estimates of the number of hidden units and variation with respect to half-spaces. Neural Networks vol. 10, No. 6: 1061- 1068, (1997).

56. Kurkova, V. Kolmogorov's theorem is r elevant. Neural Computation, vol. 3: 617-622, (1991).

57. Kurkova, V. Are sigmoidals the best activation functions in multilayer feedforward networks?. Neural Network World, vol. 1: 27-34, (1992).

58. Kurkova, V. Kolmogorov's theorem and multilayer Neural Networks. Neural Networks, vol. 5: 501-506, (1992).

59. Kurkova, V. Optimal weight spaces for feedforward neural network. IEEE Workshop CMP: 10-14, (1994).

60. Kurkova, V. Trade-off between the size of parameters and the number of units in one-hidden-layer networks. Neural Network World, vol. 2: 191-200, (1996).

61. Kurkova, V. Aproximation of functions by perceptron networks with bounded number of hidden units. Neural Networks, vol. 8: 745-750, (1995).

62. Kurkova, V. Trade-off between the size of the weights and the number of hidden units in feedforward networks. Institute of Computer Science, Academy of Science of Czech Republic, Technical Report No. V-965, (1996).

63. Kwok, T. y Yeung, D. Efficient Cross-Validation for Feedforward Neural Networks. Department of Computer Science, Hong Kong University of Science and Technology, (1995).

64. Lange, F. Fast and accurate training of multilayer Perceptrons using extended Kalman Filtr. Institute for Robotics and Systems Dynamics, DLR, Germany, (1995).

65. Lawrence, S. y Chung Tsoi, A. A Back Function Approximation with Neural Networks and Local Methods: Bias, Variance and Smmothness. Australian Conference on Neural Networks, ACNN'96: 16-21, (1996).

66. Lawrence, S., Lee Giles, C. y Chung Tsoi, A. Lessons in Neural Network Training : Overfitting may be hader than expected. Proccedings of the XIV National Conference on Artificial Intelligence, California: 540-545, (1997).

67. LeCun , Y. Une procedure d'apprentissage pour reseau a seuil assymetrique. Cognitiva 85: 599-604, (1985).

68. Leisch, F. y Hornik, K. Error-Dependent Resampling for Artificial Neural Network Classifiers. In Bernd Kröplin, editor, Neuronale Nerze in den Ingenieurwissenschafren: 1-14, (1997).

69. Manry, M., Apollo, S. y Yu, Q. Minimum Mean Square Stimation and Neural Networks. Neorocomputing, vol 13: 59-74, (1996).

70. Mayoraz, E. On the Complexity of Recognizing Regions Computable by two-Layered Perceptrons . Annals of Mathematics and Artificial Intelligence 24: 129-153, (1998).

71. Mayoraz, E. y Aviolat, F. Constructive Training Methods for Feedforward Neural Networks with Binary Weights. International Journal of Neural Systems, vol. 7, No. 2, (1996).

72. Mayoraz, E. y Robert, V. Maximizing the Robustness of a Linear Threshold Classifier with Discrete Weights. Network: Computation in Neural Systems, vol. 5: 299-315, (1994).

73. Mehrotra, K., Mohan, C. y Ranka, S. Bounds on the number of samples needed for neural learning. IEEE Transactions on Neural Networks, 2(6): 548-558, (1991).

74. Mhaskar, H. y Micchelli, C. Degree of approximation by neural and translation networks with a single hidden layer. Advances in Applied Mathematics, vol. 16: 151-183, (1995).

75. Mhaskar, H. Neural Networks for optimal approximation of smooth and analytic functions. Neural Computation, vol. 8: 164-177, (1995).

76. Minsky, M. y Papert, S. Perceptrons. Expanded Edition, Cambridge, MA: MIT Press, (1988).

77. Moerland, P., Fiesler, E. y Saxena, I. Discrete All-Positive Multilayer Perceptrons for Optical Implementation. Optical Engineering, vlo.37, No. 4, (1998).

78. Neruda, R. y Stedrý, A . Cascade Networks: Another Approach to Function Approximation. Institute of Computer Science, Academy of Science of Czech Republic, Technical Report No. V-697, (1997).

79. Parekh, R., Yang, J. y Honavar, V. Constructive neural network learning algorithms for multi-category pattern classification. Technical Report TR95-15, Artificial Research Group, Department of Computer Science, Iowa State University, U.S.A., (1995).

80. Parker, D. Learning-logic: Casting the cortex of the human brain in silicon. Technical Report TR-47. Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, (1985).

81. Peantionis, S. y Karras, D. Analysis of Backpropagation Algorithm with momentum. IEEE Transaction on Neural Networks, Vol. 5, No. 3: 505-506, (1994).

82. Rumelhart, D., Hinton, G. y Williams, R . Learning Representations by Back-propagating errors. Nature 323: 533-536, (1986).

83. Saarinen, S., Bramley, R. y Cybenko, G. Ill-conditioning in neural network training problems. SIAM Journal on Scientific Computing, vol. 14, No. 3: 693-714, (1993).

84. Saxena, I., Fiesler, E. y Moerland, P. A Method for All-Positive Optical Multilayer Perceptrons. In Procceding of the Third IEEE International Conference on Electronics, Circuits and Systems (ICECS'96), vol. 1, IEEE, Piscataway, NJ:448-451, (1996).

85. Scarselli, F. y Tsoi, A. Universal approximation using feedforward neural networks - a survey of some existing methods, and some new results. Neural Networks vol. 11 No. 1: 15-37, (1998).

86. Schiffmann, W. y Westphal, G. Training Multilayer Feedforward Networks with Limited Numerical Precision and Integer Arithmetic. Institute of Phisics, University of Koblenz, (1999).

87. Setiono, R. y Liu, H. Understanding Neural Networks via Rule Extraction. Department of System and Computer Science, National University of Singapore, (1996).

88. Setiono, R. y Chi Kwong Hui, L. Use of Quasi-Newton Method in a Feedforward Neural Network Contrucction Algorithm. IEEE Tansactions on Neural Networks, vol. 6, No. 1 , (1995).

89. Síma, J. Back-propagation is not efficient. Neural Networks, vol. 9, No. 6: 1017-1023, (1996).

90. Siu, K-Y., Roychowdhury, V. y Kailath, T. Discrete Neural Computation: A Theoretical Foundation. Prentice-Hall, Englewood Cliffs, NJ, (1995).

91. Sontag, E. Feedbakc stabilization using two-hidden-layer nets. IEEE Tans. on Neural Networks, vol. 3: 981-990, (1993).

92. Sontag, E. Feedforward nets for interpolation and classification. Comp. Syst. Sci. Vol. 45: 20-48, (1992).

93. Sontag, E. y Sussmann, H. Backpropagation separates where perceptrons do. Neural Networks, vol. 4: 243-249, (1991).

94. Sontag, E., DasGupta, B. y Siegelmann, H. On the complexity of training neural networks with continuous activation functions. IEEE Trans. Neural Networks vol. 6: 1490-1504, (1995).

95. Sontag, E. y Sussmann, H. Backpropagation can give rise to spurious local minima even for networks without hidden layers. Complex Systems 3: 91-106, (1989).

96. Sprecher, D. A universal mapping for Kolmogorov's superposition theorem. Neural Networks, vol. 6:1089-1094, (1993).

97. Stinchcombe, M. y White, H. Approximating and Learning unknown mappings using multilayers feedforward networks with bounded weights. In Proccedings of IJCNN, IEEE Press, Vol. III, New York: 7-16, (1990).

98. Tamburini, F. y Davoli, R. An algorithmic method to build good training sets for neural network classifiers. Laboratory for Computer Science, University of Bologna, Technical Report UBLCS-94-18, (1994).

99. Topchy, A., Lebedko, O. y Miagkikh, V. Fast Learning in Multilayered Neural Networks by means of Hybrid Evolutionary and Gradient Algorithms. Research Institute for Multiprocessor Computer Systems, Taganrog, Russia, (1995).

100. Torres, J., Gordon, M. Efficient adaptative learning for classification tasks with binary units. Département de Recherche Fondamentale sur la Matière Condensée, France, (1997).

101. Ventura, D. y Martínez, T. An Empirical Comparation of Discretization Methods. Proceedings of the Tenth International Symposium on Computer and Information Science: 443-450, (1995).

102. Wang, Z., Dimassimo, C., Tham, M. y Morris, A. A procedure for determining the topology of multilayer feedforward neural networks. Neural Networks vol. 7 No. 2: 291-300, (1994).

103. Werbos, P. Backpropagation: Past and Future. Proceedings of IEEE International Conference on Neural Networks, IEEE Press, Vol. I, New York: 343-353, (1988).

104. Widrow, B. y Lehr, M. 30 Years of Adadtive Neural Networks: Perceptron, Madaline, and Backpropagation. Proceedings of IEEE, vol. 76, No. 9: 1415-1442, (1990).

105. Yu, X. Can backpropagation error surface not have local minima?. IEEE Trans. on Neural Networks, vol. 3, No. 6:1019-1021, (1992).

106. Yu, X. y Chen, G. Efficient backpropagation learning using optimal learning rate and momentum. Neural Networks vol. 10 No. 3: 517-527, (1997).

107. Zurada, J. Introduction to Artificial Neural Systems. PWS Publiching Company, (1992).

APENDICE A

PROCEDIMIENTOS AUXILIARES PARA EL DISEÑO DE LA RED DE MEMORIA REAL

Los procedimientos 3.1.1, 3.1.2, 3.2.1 y 3.2.2 son para seleccionar elementos pivotes que disminuyen el valor de k_1.

Estos procedimiento buscan en una columna $u(r),(r \neq s)$, el elemento pivote (si es que existe), que transforma a negativos, la mayor cantidad de elementos de la columna $u(s)$. En los procedimientos 3.1.1 y 3.2.1 la columna $u(s)$ está asociada a una variable libre de la clase 1 y en los procedimientos 3.1.2 y 3.2.2 la columna $u(s)$ está asociada a una variable libre de la clase 2.

Procedimiento 3.1.1: *Buscar Pivote* 1 (Para el problema P3.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $l_1 \leftarrow 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \le T_1$, $b_i = 1$, determinar $c_{m1} = min\{c_i\}$.

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) \le T_1$, $b_i = 1$, entonces

- Para los valores de j que cumplen $u(j+n+1) \le T_1$, $c_j < c_{m1}$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow m1+n+1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) \le T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Si $u(i+n+1) \le T_1$, entonces para los valores de j que cumplen $u(j+n+1) \le T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j| + 1$
- Si $u(i+n+1) > T_1$, entonces para los valores de j que cumplen $u(j+n+1) \le T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

Procedimiento 3.1.2: *Buscar Pivote* 1 (Para el problema P3.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \ge 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $l_1 \leftarrow 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) > T_1$, $b_i = 1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) > T_1$, $b_i = 1$, entonces

- Para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_{m1}$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow m1 + n + 1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) \leq T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i + n + 1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Si $u(i+n+1) > T_1$, entonces para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j| + 1$
- Si $u(i+n+1) \leq T_1$, entonces para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i + n + 1$

Procedimiento 3.2.1: *Buscar Pivote* 2 (Para el problema P3.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $l_1 \leftarrow 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $b_i = 1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) \leq T_1$, $c_j < c_i$, $b_j = 1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = -1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) \leq T_1$, $c_j < c_i$, $b_j = 1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

Procedimiento 3.2.2: *Buscar Pivote* 2 (Para el problema P3.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\ b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $l_1 \leftarrow 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) > T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $b_i = 1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_i$, $b_j = 1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) \leq T_1$, $b_i = -1$, $c_i < c_{m1}$, hacer

- Para los valores de j que cumplen $u(j+n+1) > T_1$, $c_j < c_i$, $b_j = 1$, determinar $l_2 = \sum_j |b_j|$
- Si $l_2 > l_1$, entonces $l_1 \leftarrow l_2$, $k \leftarrow i+n+1$

El procedimiento 3.3 permite realizar el intercambio entre una variable libre $z_{u(r)}$ y una variable dependiente $z_{u(k+n+1)}$.

Procedimiento 3.3: *Pivotear.*

Paso 1 : Asignar

- $a_1 \leftarrow a_{k,u(r)}$
- $a_{k,j} \leftarrow \frac{a_{k,j}}{a_1}$, $j = 1,\cdots,T$

Paso 2 : $\forall i \neq k$, asignar

- $a_2 \leftarrow a_{i,u(r)}$
- $a_{i,j} \leftarrow a_{i,j} - a_2.a_{k,j}$; $j = 1,\cdots,T$

Paso 3 : Intercambiar $u(r)$ con $u(k)$.

El proceso de transformación de la columna $u(s)$, para que contenga elementos negativos, asociados a todas las variables dependientes de la clase 1, se puede resumir en el procedimiento 3.4.1 y para que contenga elementos negativos, asociados a todas las variables dependientes de la clase 2, se puede resumir en el procedimiento 3.4.2.

Procedimiento 3.4.1: *Disminuir* k_1 (Para el problema P3.3).

Paso 1 : Inicializar $r = 0$.

Paso 2 : Repetir, hasta que $r = n+1$

- $k = 0$
- $r \leftarrow r+1$
- Si $r \neq s$, aplicar procedimiento 3.1.1
- Si $k \neq 0$, entonces
  - Aplicar procedimiento 3.3
  - Actualizar $k_1 \leftarrow k_1 - l_1$
  - Si $k_1 = 0$, entonces $r = n+1$

Paso 3 : Si $k_1 \neq 0$, entonces

- Inicializar $r = 0$
- Repetir, hasta que $r = n+1$
  - $k = 0$
  - $r \leftarrow r+1$
  - Si $r \neq s$, entonces aplicar procedimiento 3.2.1
  - Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - Actualizar $k_1 \leftarrow k_1 - l_1$
    - Si $k_1 = 0$, entonces $r = n+1$

Paso 4 : Si $k_1 \neq 0$, entonces ir al paso 1, en caso contrario parar.

Procedimiento 3.4.2: *Disminuir* k_1 (Para el problema P3.4).

Paso 1 : Inicializar $r = 0$.

Paso 2 : Repetir, hasta que $r = n+1$

- $k = 0$
- $r \leftarrow r+1$
- Si $r \neq s$, aplicar procedimiento 3.1.2
- Si $k \neq 0$, entonces
  - Aplicar procedimiento 3.3
  - Actualizar $k_1 \leftarrow k_1 - l_1$
  - Si $k_1 = 0$, entonces $r = n+1$

Paso 3 : Si $k_1 \neq 0$, entonces

- Inicializar $r = 0$
- Repetir, hasta que $r = n+1$
  - $k = 0$
  - $r \leftarrow r+1$
  - Si $r \neq s$, entonces aplicar procedimiento 3.2.2
  - Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - Actualizar $k_1 \leftarrow k_1 - l_1$
    - Si $k_1 = 0$, entonces $r = n+1$

Paso 4 : Si $k_1 \neq 0$, entonces ir al paso 1, en caso contrario parar.

Los procedimientos 3.5.1, 3.5.2, 3.6.1 y 3.6.2 son modificaciones de los procedimientos 3.1.1, 3.1.2, 3.2.1 y 3.2.2, que se usan para seleccionar elementos pivotes que disminuyen el valor de k_2. Los procedimientos 3.5.1 y 3.6.1 son para el problema P3.3 y los procedimientos 3.5.2 y 3.6.2 son para el problema P3.4.

Procedimiento 3.5.1: *Buscar Pivote* 3 (Para el problema P3.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,v(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = 1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) \leq T_1$, $b_i = 1$, entonces

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_{m1}$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \leq c_{m1}$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow m1 + n + 1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Entre los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j| + 1.$$

- Entre los valores de j que cumplen $c_j \leq c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Procedimiento 3.5.2: *Buscar Pivote* 3 (Para el problema P3.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T - n - 1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) > T_1$, $b_i = 1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) > T_1$, $b_i = 1$, entonces

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_{m1}$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \le c_{m1}$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow m1 + n + 1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) \le T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \le c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Entre los valores de j que cumplen $c_j < c_i$, $b_j \approx -1$, determinar

$$l_1 = \sum_j |b_j| + 1.$$

- Entre los valores de j que cumplen $c_j \le c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Procedimiento 3.6.1: *Buscar Pivote* 4 (Para el problema P3.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\ b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$$l_1 = \sum_j b_j$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar

$$l_2 = \sum_j |b_j|$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$$l_1 = \sum_j b_j .$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar

$$l_2 = \sum_j |b_j| + 1$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Procedimiento 3.6.2: *Buscar Pivote* 4 (Para el problema P3.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) > T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $u(i+n+1) \leq T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$$l_1 = \sum_j b_j$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar

$$l_2 = \sum_j |b_j|$$

- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$l_1 = \sum_j b_j$.

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j| + 1$
- Calcular $l = l_1 - l_2$
- Si $l > max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Para tratar de disminuir, en la columna $u(s)$, la cantidad de elementos no negativos, en el problema P3.3, asociados a variables dependientes de la clase 2, se aplica el procedimiento 3.7.1 y en el problema P3.4, asociados a variables dependientes de la clase 1, se aplica el procedimiento 3.7.2

En estos procedimientos, el criterio de parada esta dado por la condición $contador = 0$, que se obtiene cuando no existe ningún elemento pivote que disminuya el valor de k_2.

Procedimiento 3.7.1: *Disminuir* k_2 (Para el problema P3.3).

Paso 1 : Inicializar $r = 0$, $contador = 0$.

Paso 2 : Repetir, hasta que $r = n + 1$

- $k = 0$
- $r \leftarrow r + 1$
- Si $r \neq s$, aplicar procedimiento 3.5.1
- Si $k \neq 0$, entonces
  - Aplicar procedimiento 3.3
  - Actualizar $k_2 \leftarrow k_2 - max$
  - $contador \leftarrow contador + 1$

Paso 3 : Inicializar $r = 0$.

Paso 4 : Repetir, hasta que $r = n + 1$

- $k = 0$
- $r \leftarrow r + 1$

- Si $r \neq s$, aplicar procedimiento 3.6.1
- Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - Actualizar $k_2 \leftarrow k_2 - max$
    - $contador \leftarrow contador + 1$

Paso 5 : Si $contador \neq 0$, entonces ir al paso 1, en caso contrario parar.

Procedimiento 3.7.2: *Disminuir* k_2 (Para el problema P3.4).

Paso 1 : Inicializar $r = 0$, $contador = 0$.

Paso 2 : Repetir, hasta que $r = n + 1$

- $k = 0$
- $r \leftarrow r + 1$
- Si $r \neq s$, aplicar procedimiento 3.5.2
- Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - Actualizar $k_2 \leftarrow k_2 - max$
    - $contador \leftarrow contador + 1$

Paso 3 : Inicializar $r = 0$.

Paso 4 : Repetir, hasta que $r = n + 1$

- $k = 0$
- $r \leftarrow r + 1$
- Si $r \neq s$, aplicar procedimiento 3.6.2
- Si $k \neq 0$, entonces
    - Aplicar procedimiento 3.3
    - Actualizar $k_2 \leftarrow k_2 - max$
    - $contador \leftarrow contador + 1$

Paso 5 : Si $contador \neq 0$, entonces ir al paso 1, en caso contrario parar.

Utilizando los procedimientos 3.4.1 y 3.7.1 se desarrolla el procedimiento 3.8.1 que selecciona una columna $u(s)$ de la matriz de dependencia lineal y la convierte mediante transformaciones elementales en una columna que contiene todos los elementos negativos para las variables de la clase 1 y la mayor cantidad de coeficientes negativos para las variables de la clase 2.

De forma similar, pero utilizando los procedimientos 3.4.2 y 3.7.2 se desarrolla el procedimiento 3.8.2 que selecciona una columna $u(s)$ de la matriz de dependencia lineal y la convierte mediante transformaciones elementales en una columna que contiene todos los elementos negativos para las variables de la clase 2 y la mayor cantidad de coeficientes negativos para las variables de la clase 1.

Procedimiento 3.8.1: *Transformación de la matriz de dependencia lineal*

(Para el problema P3.3)

Paso 1 : Seleccionar la columna $u(s)$ en la matriz de dependencia lineal

- Entre los valores de j que cumplen $1 \le j \le n+1$, $u(j) \le T_1$ y para todos los valores de i que cumplen $n+2 \le i \le T$, $u(i) \le T_1$, $a_{i,u(j)} < 0$, determinar el valor de s como el indice j para el cual se alcanza $\min_j \sum_i \text{sgn}\left(a_{i,u(j)}\right)$

Paso 2 : Calcular k_1 y k_2

- Inicializar $k_1 = 0$, $k_2 = 0$.
- Para cada valor de i que cumpla $u(i) \le T_1$, $a\left(i,u(s)\right) \ge 0$, asignar $k_1 \leftarrow k_1 + 1$
- Para cada valor de i que cumpla $u(i) > T_1$, $a\left(i,u(s)\right) \ge 0$, asignar $k_2 \leftarrow k_2 + 1$

Paso 3 : Si $k_1 > 0$, aplicar procedimiento 3.4.1

Paso 4 : Si $k_2 > 0$, aplicar procedimiento 3.7.1

Paso 5 : Calcular $p_1 = T - T_1 - k_2$

Procedimiento 3.8.2: *Transformación de la matriz de dependencia lineal*

(Para el problema P3.4)

Paso 1 : Seleccionar la columna $u(s)$ en la matriz de dependencia lineal

Si $\forall j$, tal que $1 \le j \le n+1$ se cumple que $u(j) \le T_1$, entonces

- Seleccionar en la columna $u(s)$ un elemento $a_{i,u(s)} > 0$ para $u(i) > T_1$
- Asignar $k \leftarrow i$, $r \leftarrow s$
- Aplicar procedimiento 3.3
- Asignar $s \leftarrow k$

en caso contrario

- Entre los valores de j que cumplen $1 \le j \le n+1$, $u(j) > T_1$ y para todos los valores de i que cumplen $n+2 \le i \le T$, $u(i) > T_1$, $a_{i,u(j)} < 0$, determinar el valor de s como el índice j para el cual se alcanza $\min_j \sum_i \text{sgn}\left(a_{i,u(j)}\right)$

Paso 2 : Calcular k_1 y k_2

- Inicializar $k_1 = 0$, $k_2 = 0$.
- Para cada valor de i que cumpla $u(i) > T_1$, $a\left(i,u(s)\right) \ge 0$, asignar $k_1 \leftarrow k_1 + 1$
- Para cada valor de i que cumpla $u(i) \le T_1$, $a\left(i,u(s)\right) \ge 0$, asignar $k_2 \leftarrow k_2 + 1$

Paso 3 : Si $k_1 > 0$, aplicar procedimiento 3.4.2

Paso 4 : Si $k_2 > 0$, aplicar procedimiento 3.7.2

Paso 5 : Calcular $p_2 = T_1 - k_2$

El siguiente procedimiento se utiliza para calcular los valores de las variables auxiliares.

Procedimiento 3.9: *Calculo de los valores de las variables auxiliares*

Paso 1 : Asignar a las variables libres $z_{u(s)} = 0$, $z_{u(j)} = 1$,$(j = 1,\cdots,n+1, j \neq s)$

Paso 2 : Calcular los valores de las variables dependientes mediante las relaciones de dependencia lineal $z_{u(i)} = -\sum_{j=1}^{n+1} \widetilde{a}_{i,u(j)}\, z_{u(j)}$, $(i = n+2,\cdots,T)$.

Paso 3 : $\forall\, \widetilde{a}_{i,u(s)} < 0$ tal que $z_{u(i)} < 0$, calcular $q = \max_i \left\{ \left| \frac{z_{u(i)}}{a_{iu(s)}} \right| \right\}$ y asignar

$$z_{u(s)} \leftarrow \begin{cases} \lceil q \rceil \text{ , si } q \notin \mathbf{N} \\ \lceil q \rceil + 1, \text{ si } q \in \mathbf{N} \end{cases}.$$

Paso 4 : Asignar $z_{u(i)} \leftarrow z_{u(i)} - \widetilde{a}_{i,u(s)} z_{u(s)}$, $(i = n+2,\cdots,T)$.

APENDICE B

PROCEDIMIENTOS AUXILIARES PARA EL DISEÑO DE LA RED DE MEMORIA ENTERA

Los procedimientos 5.1.1, 5.2.1, 5.1.2 y 5.2.2 permiten determinar elementos pivotes que transforman la matriz de dependencia lineal para obtener diferentes ecuaciones de hiperplanos con coeficientes reales, que separan los mismos patrones.

Procedimiento 5.1.1: *Determinar Pivote* 1 (Para el problema P5.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = 1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) \leq T_1$, $b_i = 1$, entonces

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_{m1}$, $b_j = -1$, determinar $l_1 = \sum_j |b_j|$
- Para los valores de j que cumplen $c_j \leq c_{m1}$, $b_j = 1$, determinar $l_2 = \sum_j b_j$
- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow m1 + n + 1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar $l_1 = \sum_j |b_j|$
- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = 1$, determinar $l_2 = \sum_j b_j$
- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Entre los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar $l_1 = \sum_j |b_j| + 1$.
- Entre los valores de j que cumplen $c_j \leq c_i$, $b_j = 1$, determinar $l_2 = \sum_j b_j$
- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Procedimiento 5.2.1: *Determinar Pivote 2* (Para el problema P5.3).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T-n-1)$$

Paso 2 : $\forall\, b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) \leq T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $u(i+n+1) > T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$$l_1 = \sum_j b_j$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar

$$l_2 = \sum_j |b_j|$$

- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar

$$l_1 = \sum_j b_j .$$

- Para los valores de j que cumplen $c_j \leq c_i$, $b_j = -1$, determinar

$$l_2 = \sum_j |b_j| + 1$$

- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Procedimiento 5.1.2: *Determinar Pivote* 1(Para el problema P5.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} \geq 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T - n - 1)$$

Paso 2 : $\forall\ b_i \neq 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i+n+1) > T_1$, $b_i \neq 1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Si $c_{m1} < c_i$, $\forall i \neq m1 : u(i+n+1) > T_1$, $b_i = 1$, entonces

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_{m1}$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \leq c_{m1}$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l \geq max$, entonces $max \leftarrow l$, $k \leftarrow m1 + n + 1$

Paso 6 : Para cada valor de i que cumpla $u(i+n+1) \leq T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar

$$l_1 = \sum_j |b_j|$$

- Para los valores de j que cumplen $c_j \le c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l \ge max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Paso 7 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Entre los valores de j que cumplen $c_j < c_i$, $b_j = -1$, determinar

$$l_1 = \sum_j \left| b_j \right| + 1.$$

- Entre los valores de j que cumplen $c_j \le c_i$, $b_j = 1$, determinar

$$l_2 = \sum_j b_j$$

- Calcular $l = l_1 - l_2$
- Si $l \ge max$, entonces $max \leftarrow l$, $k \leftarrow i + n + 1$

Procedimiento 5.2.2: *Determinar Pivote* 2 (Para el problema P5.4).

Paso 1 : Determinar

$$b_i = \begin{cases} 1, \text{ si } a_{i+n+1,u(r)} > 0 \text{ y } a_{i+n+1,u(s)} \ge 0 \\ -1, \text{ si } a_{i+n+1,u(r)} < 0 \text{ y } a_{i+n+1,u(s)} < 0 \\ 0, \text{ en caso contrario} \end{cases}, \quad (i = 1, \cdots, T - n - 1)$$

Paso 2 : $\forall\, b_i \ne 0$, calcular

$$c_i = \left| \frac{a_{i+n+1,u(s)}}{a_{i+n+1,u(r)}} \right|.$$

Paso 3 : Inicializar $max = 0$.

Paso 4 : Entre los valores de i que cumplen $u(i + n + 1) > T_1$, $b_i = -1$, determinar

$$c_{m1} = min\{c_i\}.$$

Paso 5 : Para cada valor de i que cumpla $u(i+n+1) \le T_1$, $b_i = 1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar $l_1 = \sum_j b_j$
- Para los valores de j que cumplen $c_j \le c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j|$
- Calcular $l = l_1 - l_2$
- Si $l \ge max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Paso 6 : Para cada valor de i que cumpla $b_i = -1$, $c_i < c_{m1}$, hacer

- Inicializar $l_1 = 0$, $l_2 = 0$
- Para los valores de j que cumplen $c_j < c_i$, $b_j = 1$, determinar $l_1 = \sum_j b_j$.
- Para los valores de j que cumplen $c_j \le c_i$, $b_j = -1$, determinar $l_2 = \sum_j |b_j| + 1$
- Calcular $l = l_1 - l_2$
- Si $l \ge max$, entonces $max \leftarrow l$, $k \leftarrow i+n+1$

Los procedimientos 5.3.1 y 5.3.2 son para discretizar los pesos y ajustar el valor de umbral en hiperplanos con pesos reales que son solución de los problemas (P.53) y (P5.4) respectivamente.

Procedimiento 5.3.1: *Discretización de los pesos de un hiperplano separador*

(Para el problema P5.3).

Paso 1: Discretización los pesos.

- Seleccionar $w = \max_{1 \le j \le n} |w_j|$.

- Asignar $w_j \leftarrow \frac{3w_j}{w}$, $(j = 1,\cdots,n+1)$.
- Aproximar los pesos al valor entero más cercano para $(j = 1,\cdots,n)$
  - Si $\lceil w_j \rceil - w_j \leq 0.5$ entonces $w_j \leftarrow \lceil w_j \rceil$, en caso contrario $w_j \leftarrow \lfloor w_j \rfloor$

Paso 2: Selección del valor de umbral:

- Calcular $z_i = d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}\right)$, para $i = 1,\cdots,T_1$
- Inicializar $p_1 = 0$
- Si existen $z_i \leq 0$, determinar
  - $z_k = min\{z_i : z_i \leq 0\ ,\ i = 1,\cdots,T_1\}$
  - $w_{n+1}^k = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn}$
  - $z_i = d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}^k\right)$, si $z_i > 0 \Rightarrow p_1 \leftarrow p_1 + 1$, $i = T_1 + 1,\cdots,T$
  - $z_l = min\{z_i : z_i > 0\ ,\ i = T_1 + 1,\cdots,T\}$
  - $w_{n+1}^l = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{ln}$
- en caso contrario , determinar
  - $z_k = min\{z_i : z_i > 0\ ,\ i = 1,\cdots,T_1\}$
  - $z_i = d_i\left(\sum_{j=1}^{n}\left(x_{ij}.w_j\right) - w_{n+1}^k\right)$, si $z_i > -z_k \Rightarrow p_1 \leftarrow p_1 + 1$, $i = T_1 + 1,\cdots,T$
  - $z_l = min\{z_i : -z_k < z_i \leq 0\ ,\ i = T_1 + 1,\cdots,T\}$
  - $w_{n+1}^l = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{ln}$
  - $w_{n+1}^k = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn}$
- Calcular $w_{n+1} = \frac{w_{n+1}^k + w_{n+1}^l}{2}$

Procedimiento 5.3.2: *Discretización de los pesos de un hiperplano separador*

(Para el problema P5.4).

Paso 1: Discretización los pesos.

- Seleccionar $w = \max_{1 \le j \le n} |w_j|$.
- Asignar $w_j \leftarrow \frac{3w_j}{w}$, $(j = 1, \cdots, n+1)$.
- Aproximar los pesos al valor entero más cercano para $(j = 1, \cdots, n)$
  - Si $\lceil w_j \rceil - w_j \le 0.5$ entonces $w_j \leftarrow \lceil w_j \rceil$, en caso contrario $w_j \leftarrow \lfloor w_j \rfloor$

Paso 2: Selección del valor de umbral:

- Calcular $z_i = d_i \left(\sum_{j=1}^{n} \left(x_{ij} . w_j \right) - w_{n+1} \right)$, para $i = T_1 + 1, \cdots, T$
- Inicializar $p_2 = 0$
- Si existen $z_i \le 0$, determinar
  - $z_k = min \{ z_i : z_i \le 0 \ , \ i = T_1 + 1, \cdots, T \}$
  - $w_{n+1}^k = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn}$
  - $z_i = d_i \left(\sum_{j=1}^{n} \left(x_{ij} . w_j \right) - w_{n+1}^k \right)$, si $z_i > 0 \Rightarrow p_2 \leftarrow p_2 + 1$, $(i = 1, \cdots, T_1)$
  - $z_l = min \{ z_i : z_i > 0 \ , \ i = 1, \cdots, T_1 \}$
  - $w_{n+1}^l = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{ln}$
- en caso contrario , determinar
  - $z_k = min \{ z_i : z_i > 0 \ , \ i = T_1 + 1, \cdots, T \}$
  - $z_i = d_i \left(\sum_{j=1}^{n} \left(x_{ij} . w_j \right) - w_{n+1}^k \right)$, si $z_i > -z_k \Rightarrow p_2 \leftarrow p_2 + 1$, $(i = 1, \cdots, T_1)$
  - $z_l = min \{ z_i : -z_k < z_i \le 0 \ , \ i = 1, \cdots, T_1 \}$

- $w_{n+1}^{l} = w_1 x_{l1} + w_2 x_{l2} + \cdots + w_n x_{1n}$
- $w_{n+1}^{k} = w_1 x_{k1} + w_2 x_{k2} + \cdots + w_n x_{kn}$

- Calcular $w_{n+1} = \dfrac{w_{n+1}^{k} + w_{n+1}^{l}}{2}$.

LISTA DE FIGURAS

Figura **Página**

Figura **Página**

RESUMEN AUTOBIOGRAFICO

Francisco Román Angel Bello Acosta

Candidato para el Grado de

Doctor en Ingeniería con Especialidad en Ingeniería de Sistemas

Tesis: **NUEVO ENFOQUE EN EL DISEÑO Y ENTRENAMIENTO DE REDES NEURONALES PARA LA CLASIFICACION**

Campo de Estudio: Ingeniería

Biografía: Nacido en Ciego de Avila, Cuba el 29 de Octubre de 1961, hijo de Arbelio Andrés Angel Bello Pardo y de Blanca Rosa Acosta Darias.

Educación: Egresado de la Universidad Estatal de Daguestán, Rusia. Grados obtenidos: Licenciado en Matemáticas y Maestro en Ciencias Físico-Matemáticas con especialidad en Ecuaciones Diferenciales, en 1985.

Experiencia profesional: Maestro de Tiempo Completo del Departamento de Matemáticas de la Universidad de Camagüey desde 1985 hasta 1997. Profesor Invitado en la Facultad de Ciencias de la Universidad de Angola, desde 1988 hasta 1990.