Implementación de una memoria asociativa lineal usando el lenguaje R

Implementation of an associative linear memory using the R language

Juan D. Velásquez, Ph.D.
Universidad Nacional de Colombia
jdvelasq@unal.edu.co

Recibido para revisión 25 de mayo de 2010, aceptado 4 de junio de 2010, versión final 26 de junio de 2010

***Resumen*—Como un aspecto fundamental en muchos cursos de redes neuronales artificiales, el desarrollo de destrezas de programación es un objetivo fundamental. Este artículo tiene dos objetivos: el primero es presentar el lenguaje R para el cómputo estadístico como una poderosa herramienta para ejemplificar algoritmos, para ejecutar cálculos numéricos, y para ilustrar conceptos usando gráficos complejos. El segundo objetivo es ilustrar la implementación práctica de un modelo de redes neuronales en el lenguaje R: teniendo en cuenta este objetivo, se seleccionó una memoria lineal asociativa.**

***Palabras Clave:*Redes Neuronales, Lenguajes de Programación, Algoritmos.**

***Abstract*— As a fundamental aspect in many artificial neural networks courses, the development of programming skills is a fundamental goal. This paper has two objectives: the first objective of this paper is introducing the R language for statistical computing as a powerful tool for exemplify algorithms, for executing numerical computations, and for illustrating concepts using complex graphics. The second objective is illustrating the practical implementation of a neural networks model in the R language: keeping in mind this objective, the linear associative memory was selected.**

***Keywords:* Artificial Neural Networks, Programming Languages, Algorithms**

I. INTRODUCCION

ES indudable que en la enseñanza de las asignaturas propias de las ciencias de la computación se haga un uso intensivo del computador como una herramienta primordial de trabajo. De ahí, que es natural que los estudiantes desarrollen destrezas y habilidades en la realización de cálculos y gráficos complejos, como un requisito para su futuro desempeño profesional y científico. En este sentido, pareciera que los desarrollos computacionales son el fin último en muchos de los trabajos prácticos desarrollados en diferentes asignaturas; consecuentemente con esto, hay una marcada tendencia a brindar elementos conceptuales en las clases teóricas, mientras que en las prácticas, se aprende a utilizar una herramienta particular con el fin de poner en práctica los conocimientos adquiridos.

No obstante, no es posible dejar a un lado la fuerte carga matemática asociada a asignaturas relacionadas con el aprendizaje estadístico, las redes neuronales artificiales, el reconocimiento y la clasificación de patrones o la optimización basada en algoritmos inspirados en la naturaleza y los procesos físicos. En este sentido, los estudiantes que toman dichas asignaturas tienen los mismos problemas de aprendizaje asociados a aquellos compañeros que toman asignaturas en el área de las matemáticas y la estadística.

En otras disciplinas el computador ha sido tomado como un elemento motivador, innovador y refrescante en la forma de enseñar dentro del aula de clase, que facilita el aprendizaje del estudiante [1,2] y que permite una mejor enseñanza [3]. Particularmente, y en este contexto, el computador puede ser usado para [1]:

- Desarrollar sesiones interactivas en el aula de clase con el ánimo de facilitar la comprensión de conceptos.
- Explorar ejemplos y ejercicios complejos que difícilmente pueden ser realizados con un papel y un lápiz.
- Visualizar problemas y su solución a través de gráficos y animaciones
- Ejemplificar el vínculo entre los desarrollos teóricos y las aplicaciones prácticas a partir de problemas reales.

Es bien sabido, que el estudiante, en general, sólo toma en serio el uso del computador cuando lo percibe como una parte integral del curso [2].

El desarrollo de ejemplos ilustrativos basados en la interacción con el computador presenta claras ventajas, tal como es discutido en [2]:

- Los resultados de las rutinas pueden ser usadas para la preparación de clases y demostraciones.
- Ejercicios bien planeados permiten direccionar dificultades específicas.
- El diseño y uso de funciones pequeñas y autocontenidas da gran flexibilidad en su aplicación, tal que son fácilmente adaptables a medida que se ajusta el material para la enseñanza de una asignatura.

Un primer problema surge en este contexto, es la selección del lenguaje computacional más apropiado para desarrollar los ejemplos de clase, teniendo en cuenta la necesidad de poder desarrollar sesiones interactivas para ejemplificar el desarrollo y la aplicación de algoritmos, y la elaboración de gráficos complejos, así como también la elaboración de trabajos prácticos por fuera del aula de clase. En este sentido, han surgido lenguajes generales diseñados específicamente para estas tareas, los cuales basan su diseño en:

- La manipulación de matrices: MATLAB, IDL (y sus clones como GNU Octave, FreeMat, Fawlty, Rlab, GNU data language y Jasymca), GAUSS, SciLab y OxMatrix.
- El uso intensivo de notaciones matemáticas o algebraicas como Maxima, APL, MAPLE o Mathematica.
- La extensión de lenguajes generales de programación como PDL (Perl Data Language) o PythonXY

Pero también han surgido ambientes especializados en tareas específicas que incluyen su propio lenguaje, tal como S-Plus o R. Particularmente, el lenguaje de programación R para la computación estadística es un ambiente para realizar cálculos estadísticos que es ampliamente aceptado y usado por dicha comunidad científica. No obstante, ha venido tomando fuerza en otras comunidades científicas aunque es casi completamente desconocido en la comunidad dedicada a las ciencias de la computación. De ahí, que resulta especialmente atractivo, cuando se considera que muchos de los problemas abordados en áreas como la inteligencia computacional, también pueden ser solucionados usando técnicas estadísticas.

El primer objetivo de este artículo, es presentar una justificación corta explicando por qué el lenguaje R es de interés en la docencia y en la práctica profesional en el área de las redes neuronales artificiales.

Y el segundo objetivo, es ejemplificar algunas de las particularidades de dicho lenguaje para el desarrollo de modelos de redes neuronales y su uso en la docencia.

Para cumplir con los objetivos propuestos, el resto de este artículo está organizado como sigue: En la Sección 2, se realiza una corta presentación del lenguaje R. Seguidamente, se hace un desarrollo conceptual corto de la memoria asociativa lineal. Posteriormente, se discute su implementación en la Sección 5. Seguidamente, se presentan ejemplos de su uso (Sección 6). Finalmente, se concluye.

II. SOBRE EL LENGUAJE R

El entorno de programación [4] es un clon de los lenguajes S [5,6,7] y S-plus [8], de tal forma que muchos programas escritos en S y S-plus pueden ejecutarse en R sin modificaciones. S y S-plus son lenguajes de muy alto nivel diseñados para [8]:

- La exploración y visualización de datos.
- El modelado estadístico.
- La programación con datos.

A continuación se describen las principales características del entorno.

A. Adquisición y licencia

El entorno R es un software libre en código fuente bajo la definición dada en la licencia GNU (General Public Licence) de la FSF (Free Software Fundation), el cual puede ser descargado de la Internet ya sea como código fuente o como un ejecutables para los sistemas operativos Linux (Debian, Redhat, SUSE o Ubuntu), Windows o MacOS. A la fecha de escritura de este artículo se encuentra disponible la versión 2.11.0. El entorno y todo el material complementario pueden ser descargados del sitio http://www.r-project.org/ o en cualquiera de los servidores web o ftp pertenecientes a CRAN.

B. Interfaz de usuario

La interacción con el usuario se basa en una interfaz de línea de comandos, que es bastante apropiada para la manipulación interactiva de datos por parte de usuarios experimentados. No obstante, la falta de una interfaz gráfica de usuario más elaborada frena a los nuevos usuarios, ya que es necesario un entrenamiento básico. En respuesta a esta falencia, se diseñado interfaces alternativas de usuario con el ánimo de facilitar el uso del entorno. Entre las más conocidas se encuentran: R-Commander [10], R-Integrated Computing Environment o R-ICE (Sriplung, 2006) y Tinn-R (http://www.sciviews.org/Tinn-R/). Adicionalmente, existe una extensión llamada "NppToR: R in Notepad++" (http://sourceforge.net/projects/npptor/) que permite utilizar el editor Notepad++ (http://notepad-plus.sourceforge.net) para escribir programas y mandarlos directamente al intérprete de R.

C. Lenguaje de programación

La sintaxis del lenguaje R es similar, al menos superficialmente, a la de C y C++ (Grunsky, 2002), pero su semántica sigue los paradigmas de la programación funcional y la programación orientada a objetos, tal como lo hacen lenguajes como LISP y Scheme; esto último implica que el lenguaje tiene la capacidad de manipular directamente los objetos del lenguaje, aplicar reglas de sustitución y evaluar expresiones.

R es un lenguaje orientado a objetos, tal que, inclusive los tipos de más básicos datos, tales como: booleanos, enteros, reales, caracteres, vectores, matrices, listas y hojas de datos son objetos mismos. Esta característica permite que el usuario interactúe de forma transparente, ya que las llamadas se realizan a funciones genéricas, como print, summary o plot, las cuales determinan internamente que método debe ser llamado dependiendo de la clase de objetos a las que pertenecen sus argumentos. R soporta internamente dos implementaciones para la programación orientada a objetos llamadas S3 [7], que fue diseñado para su uso interactivo, y S4 [6], el cual supera las deficiencias de S3, y adiciona nuevos elementos. Al igual que en muchos otros lenguajes orientados a objetos, R permite que el usuario defina sus propias clases específicas y los métodos correspondientes a cada una de ellas [11].

La popularidad del lenguaje R se debe a sus principales características:

1. Mecanismos para la manipulación y almacenamiento de datos de manera eficiente y rápida.
2. Una amplia colección de paquetes estadísticos para el análisis de datos.
3. Una amplia colección de paquetes de alto nivel para la construcción de gráficos y su posterior análisis [13, 14, 15, 16, 17].
4. Un mecanismo de extensión de la funcionalidad del entorno a través de paquetes [18, 19] que puede incluir rutinas compiladas usando Fortran 77 o lenguaje C.
5. Un lenguaje de programación, simple y efectivo, que incluye condicionales, saltos, definición de funciones recursivas y fácil manejo de los datos de entrada y salida. Operadores para ejecutar cálculos sobre vectores y matrices.
6. Un sistema para la depuración de código y manejo de excepciones.

III. MEMORIA ASOCIATIVA LINEAL

La motivación práctica es la necesidad de desarrollar un modelo simple de redes neuronales artificiales que permita asociar un patrón de entrada con un patrón de salida. En la Figura 1 se tiene una red de propagación hacia delante de una sola capa, sin neuronas adaptativas, en donde cada componente de los patrones de entrada y salida se asocia a una determinada neurona.

En el caso en que el mismo patrón es usado para la entrada y la salida de la red neuronal se habla de una auto-asociación. Este caso es de particular interés, ya que la red neuronal obtenida puede ser usada para reconstruir una entrada incompleta; en la parte inferior de la Figura 1 se ilustra un patrón de entrada arreglado en forma de matriz, y para el cual se desconocen los valores de la última fila. La red neuronal es capaz de reconstruir el patrón original a partir de la información incompleta suministrada. Igualmente, en la Figura 1 se ilustra el otro caso de interés que corresponde a limpiar un patrón contaminado para obtener el patrón original; en este caso, la entrada posee algunas entradas erróneas y la red realiza un proceso de limpieza para reconstruir el patrón original. En el caso de la red neuronal artificial presentada en la Figura 1, las neuronas de la capa de salida son activadas por una función de paso duro bipolar con el fin de que la red pueda recuperar el patrón original.

En la Figura 2. se presenta el esquema de una red neuronal artificial de una capa, con tres neuronas de entrada y dos neuronas de salida, la cual actúa como una memoria asociativa lineal. Nótese que la función de activación es lineal, al contrario del ejemplo presentado en la Figura 1.

Este tipo de red permite asociar el patrón de entrada:

$$\mathbf{a}_k = [a_{k1}, \ldots, a_{km}]^T \quad (1)$$

con el patrón de salida:

$$\mathbf{b}_k = [b_{k1}, \ldots, b_{kn}] \quad (2)$$

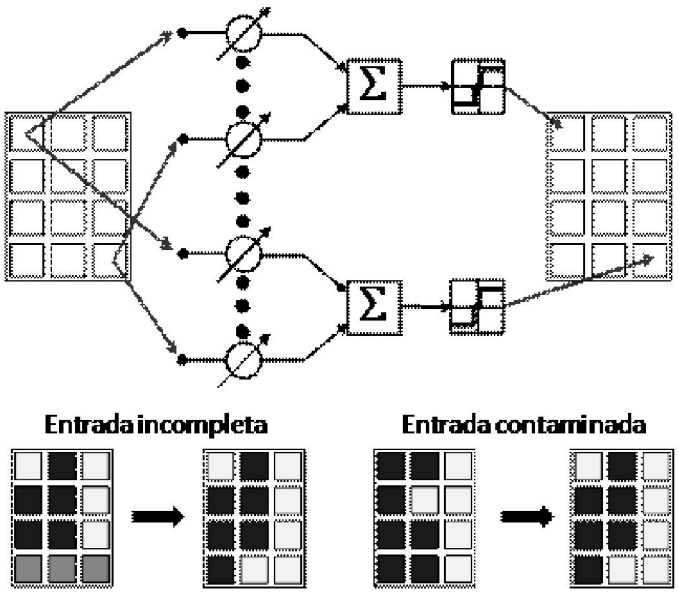

Figura 1. Motivación práctica del desarrollo del modelo.

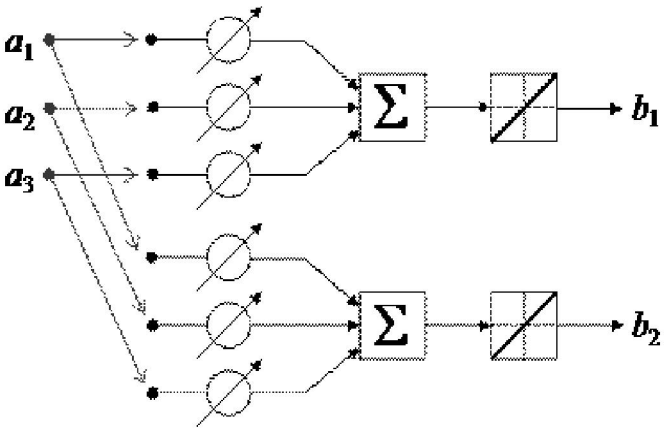

Figura 2. Esquema representativo de una memoria asociativa lineal.

Los pesos asociados a las conexiones de la red neuronal presentada en la Figura 2 pueden representarse en forma matricial, tal que la propagación de una señal a través de la red neuronal puede escribirse como:

$$\mathbf{b}_k = \mathbf{M}\mathbf{a}_k \quad (3)$$

donde $\mathbf{M}$ es la matriz de pesos.

A partir de la regla de aprendizaje de Hebb, es fácilmente demostrable que la matriz de pesos puede calcularse como:

$$\mathbf{M} = \sum_{k=1}^{Q} \mathbf{b}_k \mathbf{a}_k^T = [\mathbf{b}_1 \quad \cdots \quad \mathbf{b}_Q] \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_Q^T \end{bmatrix} = \mathbf{B}\mathbf{A}^T \quad (4)$$

Una memoria asociativa lineal, como la presentada en la Figura 2, asocia de forma perfecta cada vector de entrada $\mathbf{a}_k$ con cada vector de salida $\mathbf{b}_k$, cuando los vectores de entrada, (para , son ortogonales entre si. Esto es:

$$\mathbf{a}_u^T \cdot \mathbf{a}_v = \sum_{i=1}^{m} a_{ui} \cdot a_{vi} = \mathbf{0} \text{ con } u \neq v \quad (5)$$

Finalmente, se requiere que los vectores de entrada estén normalizados:

$$\|\mathbf{a}_k\| = 1, \qquad \text{para } k = 1, \ldots, K \quad (6)$$

Una vez obtenida la matriz $\mathbf{M}$, la red neuronal puede ser utilizada para obtener la salida $\mathbf{b}_p$ cuando se le presenta una entrada $\mathbf{a}_p$ que estar incompleta o contaminada con ruido. En este caso, se usa una función de transferencia de paso duro bipolar para la activación de las neuronas de la capa de salida, tal como en la red neuronal de la Figura 1. Este modelo corresponde a un perceptrón binario bipolar sin neurona adaptativa.

IV. IMPLEMENTACIÓN

El objetivo de esta sección es presentar un ejemplo simple pero ilustrativo que permite apreciar algunas de las particularidades del lenguaje, con el fin de demostrar su utilidad en el desarrollo de modelos de redes neuronales artificiales y su uso en la docencia. Específicamente, el código presentado en la Figura 3 ejemplifica:

- La implementación de una clase en el sistema de objetos S3 con el fin de modelar una memoria asociativa lineal.
- El proceso de sobrecarga de funciones
- El desarrollo de nuevos tipos de gráficos a partir de primitivas del lenguaje.

En las líneas 01-09 se define la función amemory, la cual crea internamente una instancia de la clase memoria asociativa (la variable obj) y la retorna como resultado de su ejecución (línea 08). La clase es modelada como una lista de elementos, de forma similar a lo que sería una estructura en un lenguaje como C++; en este caso, dicha lista contiene un único elemento que es la matriz que representa los pesos entre las conexiones (línea 06); la línea 05 corresponde a la ec. (4), en la cual se calcula la matriz de memoria a partir de los patrones de entrada y salida; en esta misma línea se normalizan los patrones de entrada (aquí se asumió que dichos patrones son binarios bipolares). En el sistema S3, la calidad de objeto se da al asignarle una clase a la lista que contiene los diferentes componentes del objeto (línea 07).

La función print.amemory() sobrecarga la función genérica print(), la cual es usada para imprimir el contenido de un objeto en pantalla. La convención para realizar la sobrecarga es utilizar el nombre de la función genérica seguido por un punto, y luego por el nombre del objeto dado en la propiedad class (línea 07). En el ejemplo realizado, sólo se imprime un mensaje en pantalla, para luego imprimir la matriz como tal.

La función predict.amemory() (línea 17) sobrecarga la función genérica predict(), y se usa para obtener la salida de la memoria asociativa cuando un nuevo patrón de entrada es presentado. Esta función corresponde a la ec. (3), pero, normalizando la entrada.

En este punto se ha codificado una memoria lineal asociativa. No obstante, el lenguaje R no tiene primitivas que permitan visualizar un patrón binario de entrada. Para subsanar este inconveniente se escribió la función arrayplot(), la cual imita el comportamiento de una función homónima existente en el software Mathematica. La función arrayplot() recibe un único argumento W que es una matriz de m filas por n columnas; esta función crea una ventana gráfica vacía (líneas 36-38) y luego dibuja un grupo de rectángulos adyacentes que representan los elementos de W (líneas 40-49). El color de cada rectángulo

está dado por el valor de cada elemento de W; el valor máximo corresponde al color negro, mientras que el mínimo corresponde al blanco.

V. EJEMPLOS ILUSTRATIVOS

A continuación se ilustrará el uso de las funciones implementadas en la sección anterior. En primer lugar, se crearan cuatro patrones binarios bipolares para crear una memoria lineal auto-asociativa:

```
> par(mfrow = c(1,4), mar = c(1, 1, 1, 1))
> P1 = c(+1,-1,-1,-1,-1,-1,-1,+1,+1,+1,+1,+1)
> P2 = c(-1,-1,-1,-1,+1,+1,+1,+1,-1,-1,+1,+1)
> P3 = c(+1,-1,-1,+1,+1,+1,-1,-1,-1,+1,-1,+1)
> P4 = c(-1,-1,-1,+1,+1,-1,+1,+1,+1,+1,-1,-1)
> arrayplot(matrix(P1,4,3))
> arrayplot(matrix(P2,4,3))
> arrayplot(matrix(P3,4,3))
> arrayplot(matrix(P4,4,3))
>
```

Los comandos anteriores generan los patrones de ejemplo para la memoria auto-asociativa, los cuales son graficados en la Figura 4. En este caso el +1 se representa como un cuadrado blanco, y el -1 como un cuadrado negro.

```
01  amemory <-
02  function(input, output)
03  {
04    # ecuacion (4)
05    W = output %*% t(input/sqrt(nrow(input)))
06    obj = list(W = W)
07    obj = structure(obj, class = "amemory")
08    return (obj)
09  }
10  print.amemory <-
11  function (obj, ...)
12  {
13    cat("Memoria asociativa\n")
14    print(obj$W)
15     cat("\n\n")
16  }
17  predict.amemory <-
18  function(obj, input = NULL)
19  {
20    # si es un vector lo convierte en matriz
21    if(!is.matrix(input))
22       input = cbind(input)
23    # normalize la entrada
24    input = A / sqrt(nrow(input))
25    return (sign(obj$W %*% input))
26  }
27  arrayplot <-
28  function(W)
29  {
30    par(pty = 's')
31    m = nrow(W)
32    n = ncol(W)
33    g = min(W)
34    r = max(W) - g
35     W = floor(1 +  99 * (W - g) / r)
36    plot.new()
37    plot.window(xlim = c(0.5, max(m, n)+0.5),
38           ylim = c(max(m, n)+0.5, 0.5))
39     palette(gray(seq(.1,.95,length.out=100)))
40    for (i in 1:m)  {
41      for (j in 1:n)  {
42         xleft = j - 0.40
43         xright = j + 0.40
44         ybottom = i - 0.40
45         ytop = i + 0.40
46         rect(xleft, ybottom, xright, ytop,
47            col = W[i,j])
48      }
49    }
50    rect(0.5, 0.5, n+0.5, m+0.5)
51  }
```

Figura 3. Código en el lenguaje R.

A continuación se procede a crear la memoria lineal auto-asociativa, la cual se almacena en la variable M:

```
> K = cbind(P1,P2,P3,P4)
> M = assmem(A = K, B = K, type = 'ALM')
```

El contenido de la variable es:

```
> M
Memoria asociativa
Tipo: ALM
Matriz de memoria:
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1.1547005	0.0000000	0.0000000	0.0000000
[2,]	0.0000000	1.1547005	1.1547005	0.0000000
[3,]	0.0000000	1.1547005	1.1547005	0.0000000
[4,]	0.0000000	0.0000000	0.0000000	1.1547005
[5,]	-0.5773503	-0.5773503	-0.5773503	0.5773503
[6,]	0.0000000	0.0000000	0.0000000	0.0000000
[7,]	-1.1547005	0.0000000	0.0000000	0.0000000
[8,]	-0.5773503	-0.5773503	-0.5773503	-0.5773503
[9,]	0.0000000	0.0000000	0.0000000	0.0000000
[10,]	0.5773503	-0.5773503	-0.5773503	0.5773503
[11,]	0.0000000	0.0000000	0.0000000	-1.1547005
[12,]	0.5773503	-0.5773503	-0.5773503	-0.5773503

```
          [,5]       [,6]       [,7]       [,8]
 [1,] -0.5773503  0.0000000 -1.1547005 -0.5773503
 [2,] -0.5773503  0.0000000  0.0000000 -0.5773503
 [3,] -0.5773503  0.0000000  0.0000000 -0.5773503
 [4,]  0.5773503  0.0000000  0.0000000 -0.5773503
 [5,]  1.1547005  0.5773503  0.5773503  0.0000000
 [6,]  0.5773503  1.1547005  0.0000000 -0.5773503
 [7,]  0.5773503  0.0000000  1.1547005  0.5773503
 [8,]  0.0000000 -0.5773503  0.5773503  1.1547005
 [9,] -0.5773503 -1.1547005  0.0000000  0.5773503
[10,]  0.0000000 -0.5773503 -0.5773503  0.0000000
[11,] -0.5773503  0.0000000  0.0000000  0.5773503
[12,]  0.0000000  0.5773503 -0.5773503  0.0000000
          [,9]      [,10]      [,11]      [,12]
 [1,]  0.0000000  0.5773503  0.0000000  0.5773503
 [2,]  0.0000000 -0.5773503  0.0000000 -0.5773503
 [3,]  0.0000000 -0.5773503  0.0000000 -0.5773503
 [4,]  0.0000000  0.5773503 -1.1547005 -0.5773503
 [5,] -0.5773503  0.0000000 -0.5773503  0.0000000
 [6,] -1.1547005 -0.5773503  0.0000000  0.5773503
 [7,]  0.0000000 -0.5773503  0.0000000 -0.5773503
 [8,]  0.5773503  0.0000000  0.5773503  0.0000000
 [9,]  1.1547005  0.5773503  0.0000000 -0.5773503
[10,]  0.5773503  1.1547005 -0.5773503  0.0000000
[11,]  0.0000000 -0.5773503  1.1547005  0.5773503
[12,] -0.5773503  0.0000000  0.5773503  1.1547005
```

El cual invoca internamente la función print.amemory(). La memoria puede ser utilizada para reconstruir patrones incompletos. Para ejemplificar este aspecto, se tomaron los patrones originales presentados en la Figura 4 y se reemplazaron las posiciones correspondientes a la última fila por ceros (note que la memoria fue diseñada para patrones binarios bipolares):

```
> par(mfrow = c(1,4), mar = c(1, 1, 1, 1))
> O1 = c(+1,-1,-1,-0,-1,-1,-1,+0,+1,+1,+1,+1,+0)
> O2 = c(-1,-1,-1,-0,+1,+1,+1,+0,-1,-1,+1,+1,+0)
> O3 = c(+1,-1,-1,+0,+1,+1,-1,-0,-1,+1,-1,+0)
> O4 = c(-1,-1,-1,+0,+1,-1,+1,+0,+1,+1,-1,-0)
> arrayplot(matrix(O1,4,3))
> arrayplot(matrix(O2,4,3))
> arrayplot(matrix(O3,4,3))
> arrayplot(matrix(O4,4,3))
```

La representación gráfica de los patrones incompletos es presentada en la Figura 5. Los patrones reconstruidos se grafican ejecutando los siguientes comandos:

```
> arrayplot(matrix(predict(M, A = O1),4,3))
> arrayplot(matrix(predict(M, A = O2),4,3))
> arrayplot(matrix(predict(M, A = O3),4,3))
> arrayplot(matrix(predict(M, A = O4),4,3))
```

La salida general es idéntica a la Figura 4. La memoria lineal asociativa también puede ser usada para reconstruir patrones contaminados con ruido. En este caso se generaron versiones contaminadas de los patrones originales, las cuales son presentadas en la Figura 6:

```
> par(mfrow = c(1,4), mar = c(1, 1, 1, 1))
> N1 = c(-1,-1,-1,-1,-1,+1,-1,-1,+1,+1,+1,+1)
> N2 = c(+1,-1,-1,+1,+1,+1,+1,+1,-1,-1,+1,-1)
> N3 = c(+1,-1,-1,-1,+1,+1,-1,-1,-1,-1,-1,-1)
> N4 = c(-1,-1,-1,+1,-1,-1,+1,+1,+1,+1,-1,-1)
> arrayplot(matrix(N1,4,3))
> arrayplot(matrix(N2,4,3))
> arrayplot(matrix(N3,4,3))
> arrayplot(matrix(N4,4,3))
```

Al igual que en el caso anterior, los patrones reconstruidos por la memoria lineal auto-asociativa son idénticos a los patrones originales:

```
> arrayplot(matrix(predict(M, A = N1), 4, 3))
> arrayplot(matrix(predict(M, A = N2), 4, 3))
> arrayplot(matrix(predict(M, A = N3), 4, 3))
> arrayplot(matrix(predict(M, A = N4), 4, 3))
```

VI. CONCLUSIONES

En este artículo se revisaron algunas de las principales características del lenguaje R para la computación estadística, y se ejemplificaron para desarrollar un modelo neuronal que representa una memoria lineal asociativa usando el sistema de objetos S3. Igualmente se ilustró como sería la interacción del usuario con el sistema al utilizar las funciones desarrolladas. Así mismo, se ejemplificó el desarrollo de una función para la construcción de un nuevo tipo de gráfico que no existe en el lenguaje.

Aunque existen muchos elementos del lenguaje que no pudieron ejemplificarse, se demostraron algunas de las bondades de esta herramienta para la elaboración de modelos de redes neuronales artificiales y para la preparación de gráficos. Ello permite vislumbrar el potencial de dicha herramienta en áreas como la inteligencia computacional tanto en la docencia como en la investigación. Se invita al lector a que indague con mayor profundidad sobre este lenguaje y busque nuevas aplicaciones en el área de las ciencias de la computación.

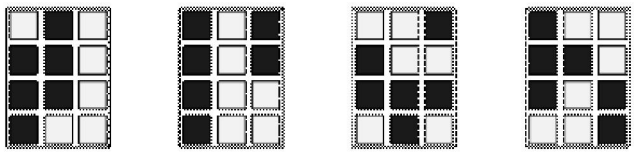

Figura 4. Patrones de ejemplo para construir una memoria lineal auto-asociativa.

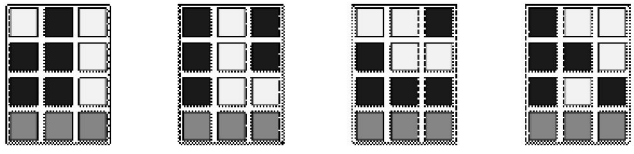

Figura 5. Patrones incompletos obtenidos al hacer ceros los elementos correspondientes a la última fila de los patrones originales

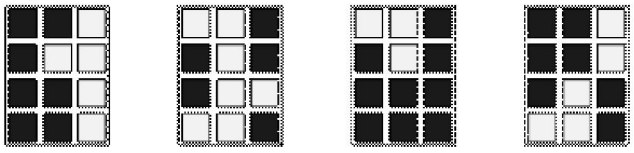

Figura 6. Patrones contaminados al invertir aleatoriamente algunos elementos de los patrones originales.

REFERENCIAS

[1] Gordon, G. 2006. Mathematica in the teaching of mathematics for computer science students in H.E. Proceedings of the 8th International Mathematica Symposium. Avignon (France).

[2] Pulley, L B. y Dolbear F. T. 1984. Computer Simulation Exercises for Economics Statistics. The Journal of Economic Education, Vol. 15, No. 1, pp. 77-87.

[3] Murray, P. M. 1999. Econometrics Lectures in a Computer Classroom. The Journal of Economic Education, Vol. 30, No. 3, pp. 308-321.

[4] Ihaka, R. y Gentleman, R. 1996. R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics Vol. 5, pp. 299–314.

[5] Becker, R., Chambers, J. M., & Wilks, A. 1988. The (new) S language: A programming environment for data analysis and graphics. Pacific Grove: Wadsworth & Brooks/Cole.

[6] Chambers, J. M. 1998. Programming with data: A guide to the S language. New York: Springer-Verlag.

[7] Chambers, J.M. y Hastie, T. J. 1992. Statistical Models in S. Chapman & Hall, London.

[8] Insightful. 2007. S-Plus 8 for Windows. User's Guide. Insightful Corporation, Seattle, WA.

[9] Fox, J. 2005. The R commander: A basic statistics graphical user interface to R. Journal of Statistical Software Vol. 14, No. 9.

[10] Sriplung, H. 2006. Integrated computing environment for R. R package Version 1.0–1. URL: http://www.r-ice.org.

[11] Grunsky, E. C. 2002. R: a data analysis and statistical programming environment — an emerging tool for the geosciences. Computers Geosciences, Vol. 28, No. 10, pp. 1219-1222.

[12] Spector, P. 2008. Data Manipulation with R. Springer.

[13] Murrell, P. 2005. R Graphics. Chapman & Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88486-X.

[14] Deepayan Sarkar. 2008. Lattice: Multivariate Data Visualization with R. Springer, New York.

[15] Maindonald, J. y Braun, J. 2007. Data Analysis and Graphics Using R. 2007. Cambridge University Press, Cambridge, 2nd edition.

[16] Wickham, H. 2009. ggplot: Elegant Graphics for Data Analysis. Use R. Springer.

[17] Cook, D. y Swayne, D. F. 2007. Interactive and Dynamic Graphics for Data Analysis. Springer, New York.

[18] Gentleman, R. 2008a. Bioinformatics with R. Chapman & Hall/ CRC, Boca Raton, FL.

[19] Gentleman, R. 2008b. R Programming for Bioinformatics. Computer Science & Data Analysis. Chapman & Hall/CRC, Boca Raton, FL.

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

Escuela de Ingeniería de Sistemas

Grupos de Investigación

Grupo de Investigación en Sistemas e Informática

Categoría A de Excelencia Colciencias
2004 - 2006 y 2000.

GIDIA: Grupo de Investigación y Desarrollo en Inteligencia Artificial

Categoría A de Excelencia Colciencias
2006 – 2009.

Grupo de Ingeniería de Software

Categoría C Colciencias 2006.

Grupo de Finanzas Computacionales

Categoría C Colciencias 2006.

Centro de Excelencia en Complejidad

Colciencias 2006

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/