UNIVERSIDAD
NACIONAL
DE COLOMBIA
Revista de la Facultad de Medicina 2000; 48 (2): 99-103

Un pequeño cerebro artificial basado en ácidos nucleicos

Jorge Eduardo Ortiz T. Profesor Asistente. jorgeo@ing.unal.edu.co. Rubén Darío Micán López rdmican@hotmail.com Estudiante X Semestre, Departamento de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad Nacional de Colombia

RESUMEN

En este artículo se presenta un modelo teórico de una Red Neuronal Artificial simple implementada en ADN. El trabajo muestra que, aunque lejos en el tiempo, existe la posibilidad de diseñar sistemas de cómputo que simulen algunas características del cerebro humano. Ese hecho contribuirá a solucionar problemas prácticos, en distintas áreas de la actividad humana, que hasta el momento no tienen solución con la actual tecnología electrónica de computadores. Este artículo da un primer paso en este sentido con el modelamiento de un perceptrón de dos entradas y salida discreta con funciones de activación escalón; para dicho fin se hace uso de sistemas de stickers.

Palabras Clave : Computación con ADN, Redes Neuronales, Stickers, Perceptrón, Simulación, Modelos Computacionales.

INTRODUCCIÓN

Día tras día se puede ver como los computadores se están arraigando cada vez más en la vida cotidiana, ya para nadie es un misterio la influencia que tiene Internet en los diversos sectores de la sociedad. Sin embargo, la inminente llegada de los límites físicos en la elaboración de microchips, la creciente necesidad de incrementar la velocidad y capacidad de procesamiento de los sistemas computacionales ha puesto de relieve lo que se puede llamar el agotamiento del paradigma de la computación electrónica tradicional basada en el modelo clásico de Turing.

A raíz de esto se ha venido dando una búsqueda intensiva de técnicas alternativas tomando los sistemas naturales como modelo, en esta línea básicamente se pueden distinguir dos tendencias: Los sistemas que a raíz de la observación de esquemas naturales buscan implementar abstracciones realizadas con base en sus características fundamentales (entre ellas vale la pena destacar las redes neuronales, la programación evolutiva, autómatas celulares, sistemas inmunológicos artificiales, etc.) y los que pretenden usar estructuras naturales para procesar información de una manera controlada, manejando las diversas señales de entrada y de salida (codificando y decodificando), la computación cuántica, y la computación con ADN son las dos manifestaciones más recientes de esta última tendencia.

Este artículo hace una interconexión de las dos tendencias mencionadas anteriormente, realizando un procesamiento de información con un modelo de red neuronal artificial bajo el soporte físico que proporciona un entorno de ADN. Para esto en la sección dos se explica una técnica usada en computación con ADN denominada el modelo de stickers, con el cual se realiza el modelamiento en la sección tres ilustrando mediante un sencillo ejemplo tanto de la neurona, como de la red (perceptrón). Posteriormente se analizan los resultados y su utilidad en la sección cuatro, finalmente se da la bibliografía y un apéndice en el que se da una breve síntesis de los aspectos más relevantes de las redes neuronales artificiales.

Sistema de Stickers

La computación con ADN tomó fuerza a raíz de la solución de problemas complejos[1] (como el del camino hamiltoniano(1) mostrándose el alto

[1]La complejidad de un problema de teoría de la computadora se mide con base en el tiempo y espacio en menoria que se necesita para solucionarlo

grado de paralelismo que se logra a través de este paradigma y vislumbró el potencial computacional de la complementariedad de Watson-Crick. Esto motivó una serie de estudios en dicha área, en los cuales se desarrollaron diversas técnicas, para resolver una amplia variedad de problemas de gran complejidad computacional. Uno de los procedimientos usados es el de Stickers. El modelo de Stickers (13) implanta un sistema de memoria, que hace uso de las hebras de ADN como medio físico de representación de la información. Cada hebra (memoria compleja) tiene n bases en total y k sub-hebras no superpuestas de m bases de longitud, no necesariamente consecutivas ($n \geq mk$), complementadas por un conjunto de minihebras *(stickers)* como se ilustra en la Figura 1. Cada minihebra se identifica con un bit (1 ó 0).

La selección del parámetro k se realiza de acuerdo con la complejidad de la información que se esté manejando y a la forma de representación de ésta. En el caso de n se tiene en cuenta el número de datos que se requieren representar sobre la memoria compleja, el valor de m no es importante desde el punto de vista computacional.

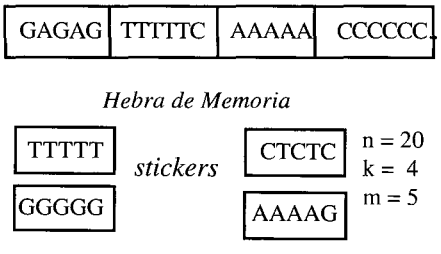


Figura 1. *Ejemplo de Stickers de Memoria*

Una subhebra puede estar en 1 (si está unida a su *sticker* complementario) ó 0 (si no está unida), de tal forma que se puede codificar una serie binaria en una hebra como se muestra en la figura 2, en la que se puede apreciar la codificación del número 1001. Cada hebra de memoria en la que se pueden representar combinaciones de unos y ceros se le denomina memoria compleja.

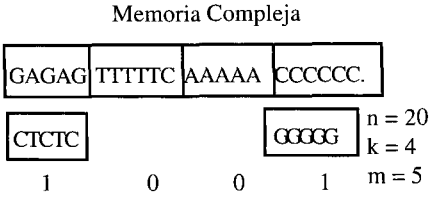


Figura 2. *Ejemplo de Memorias Complejas*

En este ejemplo las subhebras correspondientes a las posiciones pares constan únicamente de pirimidinas (C, T) y purinas (A, G) en las posiciones impares, esto permite la creación de bordes entre las subhebras (evitando errores en los *stickers*).

Para realizar un modelo de computación basado en *stickers* se definen las siguientes operaciones que son una variante de las que Adleman presentó originalmente (1):

Mezcla El contenido (memorias complejas) de dos tubos de entrada se unen en un nuevo tubo

Separación Dado un tubo N y un entero i el tubo +(N, i) consta de todas las memorias complejas de N donde la i-ésima subhebra se encuentre en 1

Activación Activar(N, i) produce un tubo en el cual la i-ésima subhebra de cada memoria compleja está en 1 (los activa de ser necesario)

Limpieza Contrario a activar, retorna un tubo en el cual cada memoria compleja tiene 0 en su n-ésima subhebra

Detección Detectar (N) retorna verdadero si existe al menos una memoria compleja en el tubo N

La entrada o tubo inicial de prueba es una librería de memorias complejas. En particular una (k,L) librería consiste de memorias con k subhebras, las últimas k-L en 0

Con base en este sistema se han desarrollado soluciones a diversos problemas, entre ellos el problema del mínimo conjunto de cobertura (7) y el de ruptura del encriptamiento estándar de datos (2)

Planteamiento del Modelo

Diseño de una neurona

Con base en el sistema de *stickers* y sus operaciones definidas en el apartado anterior se puede realizar la simulación de una neurona de dos entradas y salida discretas, con función de activación escalón. Para ello se define la siguiente disposición dentro de las hebras ADN

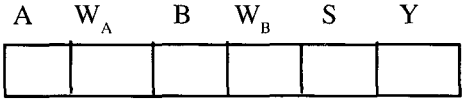


Figura 3. *Neurona modelada como Memoria Compleja*

Correspondiente al modelo a simular:

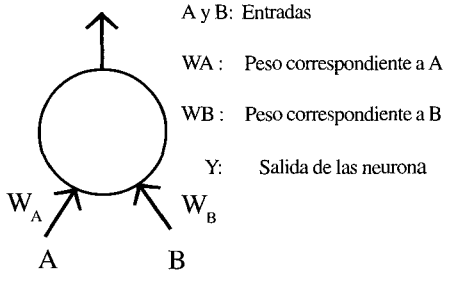


Figura 4. *Neurona*

Cada uno de los espacios es un conjunto de subhebras en los cuales se realiza un representación de los números correspondientes, un número estará representado por el número de *stickers* (número de bits en uno) que se hallen dentro del espacio que se tiene provisto para él, de tal forma que si se desea representar un cuatro se necesitará que halla cuatro subhebras en L^2. El espacio S se usa

[2] Esta forma de representación involucra una normalización sobre los números presentes, de tal forma que sólo se trabaje con enteros, aún a pesar de la constumbre de trabajar los pesos con valores entre 0 y 1

para realizar de cálculos intermedios, este debe ser el más amplio como se mostrará posteriormente. Dado que inicialmente se requiere que tanto en L como en Y los *stickers* se encuentren en 0, el tubo inicial será una librería (y + 1, s)[3]

Con base en esta distribución de las hebras ADN, el algoritmo (serie de pasos estructurados) que realizará la simulación se muestra a continuación:
Algoritmo Procesa_Neurona

(1) ultimo[4] = s
(2) Para (i = w_A, I≤b -1)
(3) Para (j = 1 to w_A - 1)
(4) N_I←+(N_0, j)
(5) Si (detectar (N_0←Activar (+(N_1,i),ultimo)))
Entonces
(6) ultimo = ultimo +1
(7) Para (i = w_B, i≤s - 1)
(8) Para (j = b, j ≤ w_B - 1)
(9) N_I← +(N_0, j)
(10) Si (detectar(N_0 ← Activar (+(N_1,i),ultimo)))
Entonces
(11) ultimo = ultimo +1
(12) Para (i = s,i ≤ s + umbral)
(13) N_0 ←+ (N_0, i)
(14) activar (N_0, y)

En los primeros pasos (1 - 11) se realiza un doble ciclo en el cual se van activando subhebras del espacio temporal S, con el fin de procesar los pesos junto con sus respectivas entradas, de tal forma que se vaya incrementando la variable ultimo en el caso en que se realice una activación, al final (11 - 14) se mira si se sobrepasó el umbral (correspondiente a la función de activación) para generar la salida, esto se podría también haber realizado mediante una comparación de la variable ultimo con el valor del umbral.

Ejemplo

Se puede observar como el siguiente ejemplo figura 5 realiza un seguimiento al algoritmo planteado

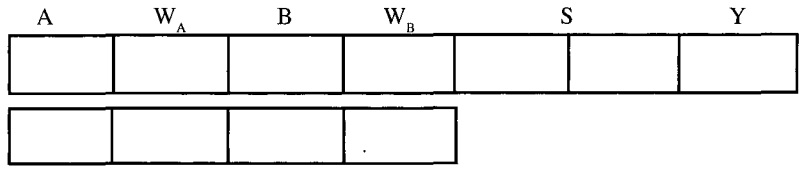


Figura 5. *Configuración Inicial del ejemplo.*

Correspondiente a los siguientes valores:

A = 1 W_A= 1 B = 1 W_B = 1 S = 00
Y = 0

Nótese que la única variable que ocupa dos subhebras es S. Siguiendo el algoritmo Procesa_Neurona, se presentan las siguientes configuraciones tomando como umbral uno

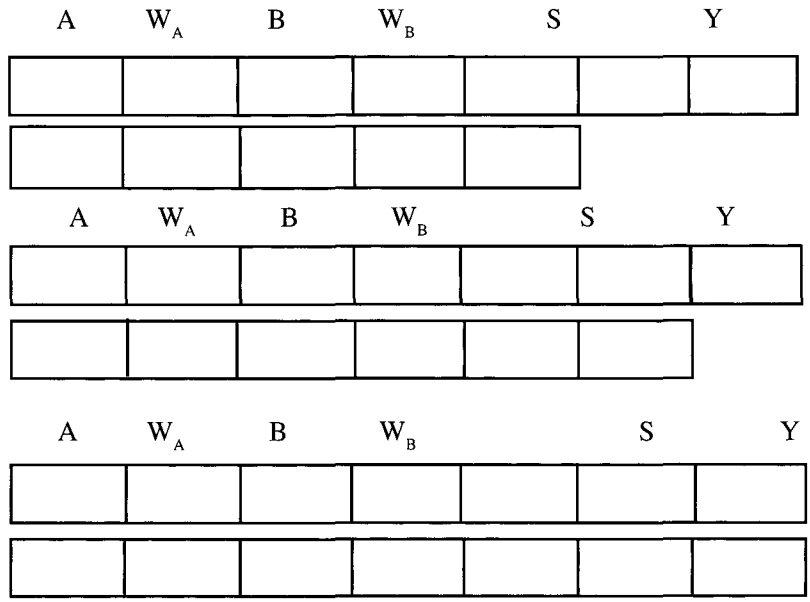


Figura 6. *Seguimiento del algoritmo*

Correspondientes a los valores que se muestran en la siguiente tabla:

A	W_A	B	W_B	S	Y
1	1	1	1	1	0
1	1	1	1	2	0
1	1	1	1	2	1

Este ejemplo corresponde a una neurona que realiza la función lógica OR, es un ejercicio interesante la verificación para los otros casos (cambiando los calores de las entradas A y B). Es importante destacar que el modelo se puede adaptar para manejar también valores negativos dando una subhebra para el signo en cada número que se representa y activando o limpiando los stickers correspondientes en los espacios temporales.

[3] Se usarán las letras minúsculas para notar la posición inicial de la ubicación dentro de memoria compleja de la variable correspondiente en mayúculas
[4] Es uan práctica común entre los programadores usar los nombres de las varioables sin tildes

Perceptrón

El perceptrón es la red neuronal (ver apéndice) más sencilla, en la cual se conectan varias neuronas, organizadas por capas, con función de activación escalón, de tal forma que las salidas de las neuronas de la capa n sirven como entradas de las neuronas de la capa n + 1. Como se ilustra en la figura 7.

La interconexión de neuronas se puede realizar sobre una memoria compleja de forma natural, dejando las salidas de la capa n como entradas de la capa n + 1, esto implica un procesamiento secuencial de la red, incluso para las neuronas de la misma capa. De tal forma que los cambios que se realizan sobre el algoritmo Procesa_Neurona para la interconexión son apenas de localización de las variables sobre el arreglo, estos cambios son relativos acorde al problema, sin embargo se mantiene la misma lógica de procesamiento. Una vez realizados estos cambios se ejecuta este algoritmo el número de veces correspondiente al número de neuronas.

Ejemplo

El modelamiento de un perceptrón de dos capas bajo el esquema presentado se muestra en la figura 6
Este modelo corresponde al perceptrón ilustrado en la figura 7

Discusión y Conclusiones

La simulación de una red neuronal bajo el modelo de computación con ADN es posible, como se mostró en este artículo, la presencia explícita de las entradas, los pesos y las salidas, permiten el uso de métodos de entrenamiento sin alterar su esencia. Por otra parte es válido destacar que el modelo no sólo se restringe a las redes feedforward. La posibilidad de usar funciones de activación diferentes a la escalón queda como tema de futuras investigaciones, al igual que el uso de representaciones más concisas. Sin embargo, se está desaprovechando el gran paralelismo de la computación con ADN, de ahí que el modelo planteado es bastante restringido a nivel práctico, más aun considerando la menor velocidad de la realización de operaciones en este tipo de computación con respecto a la que se realiza electrónicamente. Esto no descarta la posibilidad de mejores simulaciones logradas con el uso de otras técnicas con ADN

Figura 6. *Perceptrón como Memoria Compleja*

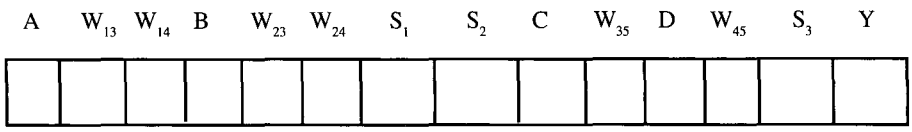
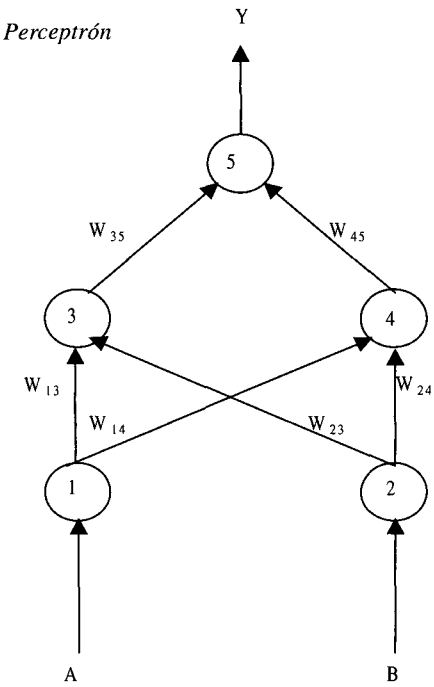


Figura 7.*Ejemplo de Perceptrón*



ANEXO

Redes neuronales artificiales

La capacidad de procesamiento paralelo de información, lo robusto del sistema y la posibilidad de aprendizaje que posee el cerebro humano son característica admirable desde diversos ámbitos, entre ellos la teoría computacional, esto da la pauta para buscar modelos que aproximen su funcionamiento. A estos modelos se les agrupa bajo el nombre de redes neuronales artificiales (RNA). Sus aplicaciones actualmente están en diversos áreas: Control, pronóstico de series de tiempo, reconocimiento de patrones, etc.

Elementos de una RNA

Al igual que su inspiración biológica cualquier modelo de RNA consta de dispositivos básicos de procesamiento: las neuronas, a partir de ellas se pueden generar representaciones de tal forma que un estado constante de ellas signifique una letra, un número...

La neurona artificial pretende emular de forma cercana las características de la neurona biológica:
- Estado de activación $a_i(t)$: Valor numérico que caracteriza la neurona en un instante dado

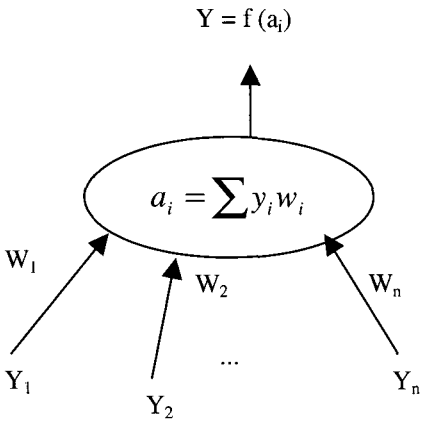
- Función de salida $f_i[a_i(t)]$: Es una función que transforma el estado de activación en una señal de salida Y_i.

- Señal de salida Y_i: Es el resultado de operar el estado de activación bajo la función de salida. $Y_i = f_i[a_i(t)]$, esta señal es enviada a través de los canales de interconexión unidireccionales a otras neuronas, o es la salida de la RNA

- Pesos w_{ij}: Cumple un papel análogo al de la sinapsis en el modelo natural, modificando la señal de entrada

Elementos de una RNA

- Capas: Es un conjunto de neuronas cuyas entradas provienen de la misma fuente (que puede ser otra capa de neuronas) y cuya salidas se dirigen al mismo destino (también puede ser otra capa de neuronas): Pueden ser de tres tipos:



- o De entrada: Reciben señales del entorno, sus entradas son las entradas de la red
- o Ocultas: Sus entradas y salidas son capas de la red
- o De salida: Sus salidas son las salidas de la red

- Estado de activación: Es un vector en el cual se tienen los estados de las neuronas en un momento dado $A(t) = (a_1(t), a_2(t), ..., aN(t))$

- Regla de entrenamiento: Es el proceso mediante el cual se definen los pesos entre las neuronas que hacen parte de la red

REFERENCIAS BIBLIOGRÁFICAS

1. **Adleman L.** "Molecular Computation of Solutions to combinatorial Problems",. Science 266, 1990004, pp 1021-1024.
2. **Boneh D., Lipton R.**. "Breaking DES Using Molecular Computing". Departament of Computer Science Princeton University. 1996
3. **Cardona C., Rojas C.** "Implementación de un autómata finito mediante computación con ADN". Memorias VII Congreso de Estudiantes de Ingeniería de Sistemas. 1998
4. **Kelly D.** Teoría de Autómatas y Lenguajes Formales. Prentice may. Madrid. 1995
5. **Lipton R.** "Speeding Up Computations Via Molecular Biology". Princeton University
6. **Paun G., Rozenberg G., Salomaa A.** DNA Computing. Ed. Springer. Berlín 1998
7. **Roweis S., Winfree E., Burgoyne R., Chelyapov N., Goodman M., Rothemound P., Adleman L.** A Sticker Based Architecture for DNA computation