MEMORIAS DEL PRIMER CONGRESO COLOMBIANO DE NEUROCOMPUTACION

EDITORES:

GERMAN HERNANDEZ
JOSE J. MARTINEZ
LUIS FERNANDO NIÑO
LUZ GLORIA TORRES

ACADEMIA COLOMBIANA DE CIENCIAS EXACTAS, FISICAS Y NATURALES
COLECCION MEMORIAS No. 4

ACADEMIA COLOMBIANA DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

COLECCIÓN MEMORIAS No. 4

MEMORIAS DEL PRIMER CONGRESO COLOMBIANO DE NEUROCOMPUTACION

EDITORES

Germán Hernández
José J. Martínez
Luis Fernando Niño
Luz Gloria Torres

Universidad Nacional de Colombia

SANTAFÉ DE BOGOTÁ, D. C. 1996

Carrera 3a. A, No. 17-34, Piso 3o., Apartado 44743
Fax: (571) 283 8552
E-mail: accefyn@colciencias.gov.co

Primera edición, 1996, Santafé de Bogotá, D. C.., Colombia

ISBN: 958-9205-17-8

CD: 006.33

Congreso Nacional de Neurocomputación (1er.: 1995: Santafé de Bogotá)
Primer Congreso Nacional de Neurocomputación: Memorias /ed. Germán Hernández...[et al.].– Santafé de Bogotá: Academia Colombiana de Ciencias Exactas, Físicas y Naturales, 1996.
xii, 180 p. – (Colección Memorias; no. 4)

1. Inteligencia artificial. 2. Autómatas celulares. 3. Redes neuronales. 4. Cibernética. I. Hernández, Germán II. Tít. III. Serie

adr - Sección de Catalogación, Departamento de Bibliotecas, Universidad Nacional de Colombia

Impresión: Editora Guadalupe Ltda.
Tel.: 269 05 32
Santa Fe de Bogotá, D.C., Colombia

Impreso en Colombia/ Printed in Colombia

MEMORIAS DEL PRIMER CONGRESO NACIONAL DE NEUROCOMPUTACIÓN

CONTENIDO

PRESENTACIÓN

Una de las aspiraciones de gran cantidad de científicos e investigadores ha sido la de simular en sistemas artificiales aspectos del comportamiento del cerebro humano. Antes de la aparición del computador ya se habían elaborado modelos de máquinas abstractas que pretendían emular la tarea de la mente considerada como secuencial: calcular. Con el advenimiento del computador aparece la metáfora *computador es cerebro*, cerebro es computador, que si bien ha tenido muchos detractores y defensores, ha contribuido enormemente al desarrollo de la *Inteligencia artificial* (IA) que, dicho de manera simplista, hace referencia a la construcción de sistemas artificiales que ejecutan tareas de alguna forma catalogadas como inteligentes.

La tecnología de la IA está basada en la premisa de que el razonamiento humano es efectuado por la manipulación de símbolos formales a los cuales se les puede dar una interpretación semántica. Es decir, el cerebro procesa información de modo secuencial. Aunque este enfoque ha producido logros impresionantes como los sistemas expertos, con tales técnicas, solo las tareas susceptibles de ser formalizadas son las que se pueden implementar en un computador secuencial.

Sin embargo, la mente humana es mucho más flexible en el procesamiento de información; ejecuta tareas que son muy difíciles de expresar mediante símbolos, tales como visión, control de movimientos (es demasiado complicado para un supercomputador simular el vuelo de un insecto), reconocimiento de imágenes, recuperación de información con datos incompletos, lenguaje natural, etc.; para la realización de dichas tareas se requiere que una gran cantidad de neuronas actúen simultáneamente.

En la búsqueda de alternativas que permitan sobrepasar las limitaciones de la computación secuencial y disponer de sistemas menos inflexibles, más tolerantes con las fallas y que nos ofrezcan la opción de resolver los problemas que con el enfoque simbólico han sido particularmente difíciles, los científicos de la información intentan elaborar sistemas artifi-

ciales que modelen de alguna forma el funcionamiento en paralelo del cerebro.

Aparece así la disciplina llamada *Neurocomputación* cuya manifestación más importante corresponde a las *redes neuronales artificiales*, que como su nombre lo indica, pretenden imitar la estructura neuronal de los sistemas biológicos naturales donde la información se representa mediante patrones de excitación y se procesa a través de interacciones mutuas entre las neuronas. De esta manera las redes neuronales artificiales están formadas por elementos de procesamiento llamados también *neuronas* o *nodos*, los cuales están interconectados según un patrón determinado (*topología de la red*). Cada uno de estos elementos recibe información que, bien pueden ser datos de entrada o información proveniente de otras neuronas, a través de las interconexiones que simulan las conexiones sinápticas de las neuronas naturales y que tienen asociado un valor llamado *peso* que emula la intensidad de la sinapsis, realiza un procesamiento simple y envía la señal de salida a otras neuronas o al exterior.

Aspectos importantes de la mente humana que se intentan imitar en las redes neuronales artificiales son la capacidad de aprender de la experiencia y la respuesta adecuada a nuevas situaciones. Corresponden a *aprendizaje* y *generalización* respectivamente, en la terminología de las redes neuronales.

Aunque el desarrollo de la neurocomputación lleva cerca de cincuenta años, solo hasta mediados de la década del ochenta se empezaron a ver sus grandes posibilidades en la solución de problemas y aplicaciones en prácticamente todas las áreas de ciencia y tecnología.

Los Departamentos de Matemáticas y Estadística e Ingeniería de Sistemas de la Universidad Nacional de Colombia, conscientes de la necesidad de dar a conocer los alcances y el estado del arte de esta disciplina, organizaron el ***Primer congreso nacional de neurocomputación***. Cerca de trescientas personas de diferentes lugares del país, principalmente de la comunidad académica, se dieron cita para atender los cursillos y conferencias plenarias de seis investigadores de talla internacional, quienes ofrecieron toda una gama de tópicos en el área y sus aplicaciones, desde los introductorios hasta temas de investigación avanzada. Durante el con-

greso, investigadores nacionales también tuvieron la oportunidad de compartir desarrollos realizados en algunas importantes universidades del país. Algunas de estas presentaciones se publican en este volumen, junto con las conferencias de los investigadores invitados.

Finalmente, el Comité organizador quiere expresar su agradecimiento a la Academia Colombiana de Ciencias Exactas, Físicas y Naturales por acoger este volumen en su colección de Memorias.

COMITÉ ORGANIZADOR

Santafé de Bogotá, junio de 1995

AGRADECIMIENTOS

El Comité Organizador agradece de manera especial a las entidades que hicieron posible la realización de este evento: Academia Colombiana de Ciencias Exactas, Físicas y Naturales, Colciencias, Fundación Mazda para el Arte y la Ciencia, , Icfes, Icetex, OEA, Sociedad Colombiana de Matemáticas.

CONFERENCISTAS INVITADOS

Dr. M. Nazmul Karim.

B.S., Bangladesh University of Engineering and Technology, Chemical Engineering.
M.S. , University of Manchester Institute of Science and Technology (UMIST), Control Engineering.
Ph.D., University of Manchester Institute of Science and Technology Chemical Engineering (Process Control)
Associate Department Chair, Department of Agricultural and Chemical Engineering, Colorado State University, Fort Collins, Colorado, USA. Actualmente es Profesor Visitante en el Department of Chemical and Process Engineering, University of Newcastle Upon Tyne, Inglaterra. Ha publicado gran cantidad de trabajos sobre aplicaciones de Redes Neuronales en Biotecnología.

Dr. Jorge Ricardo Cuéllar

Matemático, Universidad de los Andes
M.S., en Matemáticas, Universidad de los Andes
Doctor de la Universidad de Mainz
Actualmente es investigador en SIEMENS AG, Centro de Investigación, Munich, Alemania. Su amplia producción científica incluye áreas como Redes Neuronales y sus aplicaciones a la Estadística y Verificación de Sistemas Distribuidos.

Dr. Marco Tomassini

Dr. Chemical Physics, Perugia University, Italy
Investigador en Laboratoire de Systèmes Logiques EPFL. Département d'Informatique Lausanne, Suiza. Ha participado en proyectos en National Swiss Science Foundation sobre Programación Masivamente Paralela y Aplicaciones y sobre Simulación y Síntesis de circuitos lógicos por medio de métodos de Programación Genética Paralela, entre otros.

Dr. Stephen Piché

BSEE, University of Colorado at Boulder
MSEE, Stanford University
Ph.D. , Stanford University
Miembro de Technical Staff MCC Austin. TX. USA. Creador de un sistema para la detección de fraude en el uso de tarjetas de crédito por medio de Redes Neuronales. Por sus excelentes resultados mereció figuración en programas de noticias de la ABC y CNN.

Dr. Max Garzón

Matemático, de la Universidad Nacional de Colombia
M.S., University of Illinois, Urbana, USA.
Ph.D., University of Illinois, Urbana, USA.
Associate Professor Department of Mathematical Sciences, The University of Memphis, Memphis, TN, USA. Tiene una reconocida trayectoria como investigador en Redes Neuronales y Autómatas Celulares, según lo revela su vasta producción científica.

Ing. Jesús Alberto Delgado

Ingeniero Eléctrico, Universidad de los Andes, Bogotá.
M.S. en Ingeniería Eléctrica, Universidad de los Andes, Bogotá.
Actualmente cursa estudios de doctorado en Cybernetics Department, University of Reading, Inglaterra. Es profesor del Departamento de Ingeniería Eléctrica de la Universidad Nacional de Colombia. Ha publicado algunos textos y varios artículos sobre Redes Neuronales.

Prof. José Daniel Muñoz

Físico, Universidad Nacional de Colombia, Bogotá
Ingeniero Electrónico Universidad Distrital Francisco José de Caldas, Bogotá
M.S. en Física, Universidad Nacional de Colombia, Bogotá.
Profesor del Departamento de Física de la Universidad Nacional de Colombia. Trabaja actualmente en la formulación de las leyes físicas como reglas de Autómatas Celulares.

REDES HÍBRIDAS NEURO-DIFUSAS Y SU USO EN CONTROL INTELIGENTE DE PROCESOS*

Hernán Darío Álvarez Z.
Departamento de Procesos Químicos
Universidad Nacional de Colombia , Sede Medellín.
e-mail: hdalvare@perseus.unalmed.edu.co

Resumen. En este artículo se introduce el uso de las *redes híbridas neuro-difusas* (RHND) en control inteligente de procesos. Esta herramienta nueva, un híbrido entre redes neuronales artificiales y controladores difusos, tiene gran habilidad para incrementar las propiedades deseables y minimizar los problemas de las técnicas individuales, en las aplicaciones de control inteligente. Una RHND se aplicó al control de temperatura del reactor fluidizado de activación en la planta piloto de carbón activado de la Universidad Nacional de Colombia (Medellín). Su desempeño es superior al del mejor controlador PI en las mismas condiciones de operación. Así, se demuestran las grandes capacidades de estos híbridos en la implantación de controles inteligentes.

Abstract. This paper introduce the use of Neural-Fuzzy Hybrid Networks (NFHN) in intelligent process control. This new tool, a hybrid product between artificial neural networks and fuzzy logic controllers, have a great ability for increase the desirable properties and minimize the problems of individual techniques in intelligent process control applications. A NFHN was applied to the temperature control of fluidized bed activation reactor in the National University of Colombia (Medellín) active coal pilot plant. NFHN control performance is more attractive than the best designed PI control, used in the same reactor conditions. Therefore, capabilities of this hybrid configuration in intelligent control implementations are so great and call for most explorations.

1. Introducción. En el mundo del control asistido por computador, la introducción del neurocontrol y del control difuso (*fuzzy control*), ha generado una gran cantidad de desarrollos en ambas tecnologías, todos con la mira del control autónomo, paso último en el ideal del mando automáti-

* El profesor Àlvarez es co-investigador en el proyecto de investigación de **COLCIENCIAS**, CINDEC y Postgrado en Ingeniería de Sistemas de la Universidad Nacional de Colombia, Sede Medellín, titulado: "Diseño y Desarrollo Metodológico de SBCs (Sistemas Basados en Conocimientos) distribuidos y cooperantes aplicados a la Ingeniería" (con código 1118-14-091-94). Este trabajo hace parte de su labor investigativa en el marco de dicha investigación. Investigador del Grupo de Investigación y Desarrollo en Inteligencia Artificial y Reconocimiento de Patrones (UN-GIDIA).

co de un proceso industrial, pero que exige el dominio previo de las técnicas de control inteligente. Se empieza a ver como horizonte en este campo el uso de técnicas híbridas, que resalten las ventajas y soslayen las desventajas de las técnicas individuales. Una aproximación que muestra gran futuro son las redes híbridas, conformadas por nodos en los cuales se realizan funciones diferentes sobre los datos de entrada, de modo que al combinar sus salidas en red, se logra una función con alto grado de no linealidad, con características particulares de acuerdo con las funciones internas de dichos nodos.

Con esta visión, la manera tradicional de implantar un control difuso, mediante un programa (*shell*) de sistema experto, está cambiando, debido a la exigencia de mas facilidad de sintonía, adaptación y aprendizaje en estos sistemas de reglas de producción. A continuación se muestra una propuesta de red híbrida neuro-difusa RHND que, sin necesidad de un programa de sistema experto, implanta un control difuso con cualquier número de reglas, trasladando el significado de dichas reglas de producción y su relaciones internas a las conexiones de una red que posee nodos con funciones particulares. Se presenta la aplicación de una RHND para el control de la temperatura en un reactor en lecho fluidizado (activador), usado como última etapa en la producción de carbón activado. En este equipo, la temperatura interna resulta de difícil estabilización cuando se intenta su control con técnicas tradicionales (PI).

2. ¿Para qué el control inteligente y qué se espera de él? Aunque la tarea de control que cumple un ser humano se realiza de manera adecuada, está siempre de por medio el grado de especialización que alcanzan estos operadores, y su fragilidad emocional y física, lo cual deja al proceso en manos de condiciones azarosas, además de que no se presenta la misma operación del equipo cuando se cambia de un operador a otro. Como resulta obvio, la exigencia creciente hacia las empresas productoras de bienes y servicios por productos cada vez mejores en calidad y precio, que genera una competencia voraz, ha hecho que los procesos dejen de ser rentables cuando no es posible automatizarlos de modo que se logre repetibilidad en la calidad del bien o servicio ofrecido.

Todo esto motiva la búsqueda de los mecanismos necesarios para proveer del control automático adecuado a estos procesos. Se han ensayado arquitecturas con base en el control tradicional, pero siempre se han frustrado los intentos por la tan anhelada meta de controlar automáticamente el proceso. Por eso, todavía hoy, existen a nivel industrial un sin número de procesos que se controlan de manera deficiente o que en otro caso, son controlados en gran parte por operadores humanos, expertos en esta tarea. La complejidad de este tipo de procesos es el motivo principal de su incipiente automatización y su alta carga de mando manual.

El porqué estos procesos se alejan de un grado de automatización realmente significativo es, entre otras razones: Alta dimensionalidad del proceso, variación con el tiempo de las características del sistema, no linealidades en el comportamiento del proceso, modelación muy pobre de la dinámica del proceso y problemas de medición bajo incertidumbre. Es aquí donde se requiere un control que goce de características inteligentes, con el cual sea posible controlar estos procesos en un grado mucho mayor que el actual y con las mismas o mejores prestaciones con las que lo hace el operador humano.

Tomando propuestas de algunos autores, un control inteligente se puede definir de una manera estricta, como aquel mecanismo que es capaz de razonar, comprender y aprender de la tarea de control que realiza, mejorando permanentemente su desempeño. Apegados a esta definición, como un paso previo en el camino hacia un control autónomo, se ve con claridad que las características de aprendizaje son imprescindibles, y mucho mejor si se puede realizar mientras el sistema esta operando.

Es conocida la gran habilidad de las *redes neuronales artificiales* (RNA) para aprender las características de un grupo de patrones de entrada-salida, y la calidad de generalización que entregan cuando se les enfrenta a datos no presentados durante las etapas de entrenamiento. Esto hace pensar en aprovechar este tipo de tecnología para construir sistemas inteligentes de control (Fukuda, T. and Shibata, T. 1992).

De otro lado, también es conocido el excelente control que se logra sobre sistemas complejos cuando se aprovecha la experiencia del ope-

rador, de modo que usando su conocimiento, representado de alguna manera en el interior del computador, se ejerce control mediante (por ejemplo) un grupo de reglas típicas SI - ENTONCES, que toman variables del proceso para entregar valores de las acciones a realizar sobre el sistema (Pérez, G. y col. 1991). Para manejar el problema de información incompleta o con incertidumbre en estos sistemas, se puede usar la técnica de los conjuntos difusos (fuzzy sets), la cual permite razonar bajo incertidumbre y obtener valores adecuados en las conclusiones (Kevin, S. 1990; Sugeno, M. and Park, G. 1993).

Cada una de estas técnicas tiene documentadas diversas aplicaciones en control, pero cuando se han visto enfrentadas a la construcción de un control inteligente con un aprendizaje en línea eficiente, han mostrado algunos inconvenientes de implantación, que si se mezclan pueden ser minimizados.

3. La hibridación y sus logros. Entre los inconvenientes mas reportados en los sistemas de neurocontrol se pueden destacar:

- Gran cantidad de tiempo en la selección y preparación de los datos de entrenamiento, debido al gran número que se requiere para lograr buena generalización en la red entrenada.
- Mucho tiempo en el proceso de entrenamiento, aun con computadores de alta velocidad. En ocasiones decenas de horas para una red relativamente pequeña.
- La necesidad de entrenar la red con todos los patrones nuevamente, cuando se quiere cambiar un patrón o un punto de operación dados, con los consabidos costos en tiempo de entrenamiento.

De otro lado, el control difuso adolece también de inconvenientes cuando se enfrenta a la tarea de control, con miras a ser un mecanismo inteligente:

- Perdida de inteligibilidad en la base de conocimiento (por ejemplo un conjunto de reglas), cuando se deja libre para que un módulo de aprendizaje la modifique.

- Poca flexibilidad en los programas de implantación de sistemas expertos, cuando se desea cambiar un valor particular en una regla.
- Perdida de consistencia en la base de conocimiento cuando las reglas se modifican individualmente a la luz de una labor de aprendizaje.

No se mencionan aquí las ventajas que presentan ambas técnicas sobre otras tradicionales que han intentado abordar la labor de aprendizaje (por ejemplo con técnicas algorítmicas), puesto que diversos artículos de la literatura muestran implantaciones no híbridas que operan con éxito (Li, Y. and Lau, C. C. 1989; Yamakawa, Y. 1993; Karr, C. and Gentry, E. J. 1993).

Con el fin de evitar los inconvenientes de las técnicas individuales y reforzar sus ventajas, se han propuesto varios modelos de sistemas híbridos (Lin, C and Lee, C. S. 1991; Werbos, P. 1993), de los cuales es posible tomar aportes para construir un modelo que mezcle las habilidades de entrenamiento de las RNA, con el manejo de información bajo incertidumbre del razonamiento difuso, de modo que se construya una manera mucho mas eficiente de realizar aprendizaje en línea y desarrollar la tarea de control inteligente.

4. ¿Qué son las redes híbridas neuro-difusas (rhnd)? Estas redes toman su topología de las RNA típicas (Freeman, J.A. y Skapura, D.M. 1993), con nodos que desarrollan una función simple (sigmoide, logística o lineal), pero intercalan capas de nodos con funciones internas mas complejas (funciones de pertenencia, implicación y concreción en subconjuntos difusos). Esto no quiere decir que se pierdan las habilidades típicas de las RNA en los aspectos de generalización y aprendizaje, ganando además en velocidad de entrenamiento, puesto que en este caso el número de pesos (parámetros) a ajustar es mucho menor, con la ventaja de que se parte con un conocimiento heurístico del proceso, lo que facilita sobremanera el diseño del sistema en las etapas iniciales.

Como ejemplo, la Tabla 1 muestra la matriz de reglas de un control difuso simple. La Figura 1 muestra la RHND que implanta este control difuso, usando todas las posibles combinaciones de las variables de

entrada del proceso tomadas como universo de discurso (error e y cambio del error con el tiempo Δe) y usando una salida como cambio en la potencia aplicada ΔP.

	e es Negativo N	e es Cero Z	e es Positivo P
Δe es Positivo P	Regla 1, ΔP es Z	Regla 2, ΔP es P	Regla 3, ΔP es N
Δe es Cero Z	Regla 4, ΔP es P	Regla 5, ΔP es Z	Regla 6, ΔP es N
Δe es Negativo N	Regla 7, ΔP es P	Regla 8, ΔP es N	Regla 9, ΔP es Z

Tabla 1. Matriz de reglas para un control difuso simple, tipo PI.

5. Implantación de la red híbrida neuro-difusa. La red que se propone consta básicamente de 5 capas, cada una con un tipo particular de nodo genérico, de modo que se realiza toda la operación de un control difuso como si fuera una red. La función y característica de los nodos en cada capa se muestran para un control difuso tipo PI, aunque puede usarse para otros tipos de control difuso (Driankov et al. 1993).

La capa 1, o capa de entrada, es la encargada de cualquier preprocesamiento de las entradas que se estime necesario. La función que desarrollan estos nodos, en el modelo propuesto, es un acotado y escalado de los valores provenientes de sensores, de modo que los parámetros característicos del nodo son el valor central y el ancho del intervalo de la variable. Todos los universos de discursos quedan de este modo normalizados a valores de [-1, 1] ó [0,1]. En esta capa, los parámetros a entrenar no son los pesos de las entradas (pues estos siempre valen uno), sino los valores del centro y del ancho del intervalo de la variable que se trabaja.

La capa 2, denominada capa de difusión o de cálculo de factores de pertenencia μ, es la encargada de realizar la difusión (*fuzzification*) de los valores numéricos acotados y normalizados, para entregar en la salida de cada nodo un valor μ en el intervalo [0,1], que es el valor de pertenencia de la entrada al conjunto difuso que representa el nodo. Entre estos

nodos existen diversos tipos particulares, de acuerdo con la forma que se le de al conjunto difuso, por ejemplo: triángulos, campanas gaussianas, trapecios, etc. Cada una de estas figuras presenta un número diferente de parámetros, los cuales son los valores a ajustar en esta capa, porque aquí de nuevo los pesos de las conexiones ω_{ij} son iguales a uno.

La capa 3 es la capa que contiene la primera parte de la base de conocimientos. En esta capa cada nodo calcula el antecedente neto de una regla de producción del tipo:

SI <proposición difusa antecedente p> ***ENTONCES*** <proposición difusa consecuente q> ,

donde generalmente la proposición difusa antecedente es una función lógica **Y** entre dos o mas grados de pertenencia a conjuntos difusos diferentes. De este modo, la función interna que desarrollan estos nodos de reglas, depende del tipo de T-norma que se use para concluir sobre la proposición **Y**. Nótese que cualquier proposición difusa que combine **O, Y, NO** (*OR, AND* y *NOT*), puede convertirse con facilidad en dos o mas reglas simples con conectivo único tipo **Y**. La salida de cada nodo en esta capa es un factor de pertenencia μ que refleja el resultado de la operación lógica entre las entradas. Por ejemplo si se usa una T-norma de mínimo, la salida del nodo será el mínimo de todos los valores de entrada.

En los nodos de la capa 3, los parámetros de entrenamiento son los pesos de las conexiones entre estos nodos y los nodos de difusión, puesto que los parámetros internos no deberían modificarse, porque se cambiaría drásticamente la T-norma que se aplica, aunque queda abierta esta posibilidad de cambio. P. Werbos ha llamado a los pesos de las conexiones entre capa 2 y capa 3 elasticidades γ_{ij}, y muestra como variando estos valores, es posible dejar intactas las reglas como relación entre antecedente y consecuente, lo que permite preservar la inteligibilidad del sistema, incluso después de ser sometido a un proceso de entrenamiento (Werbos, P. 1993). Una interpretación a la luz del efecto final de estos pesos, que no es mas que reforzar un miembro del antecedente frente a los otros, podría ser que estas elasticidades se relacionan con un cierto tipo de meta-conocimiento, que el operador humano ejecuta, pero que no pue-

de detectarse con facilidad mediante técnicas de ingeniería del conocimiento y que está en los datos de entrada y salida del sistema. Debe recalcarse como estos nodos de reglas de la capa 3, no realizan la inferencia de cada regla. Ellos, solo realizan el calculo del factor de pertenencia de la proposición difusa antecedente. Pero, la conexión entre cada nodo de regla y la capa siguiente, representa la regla completa.

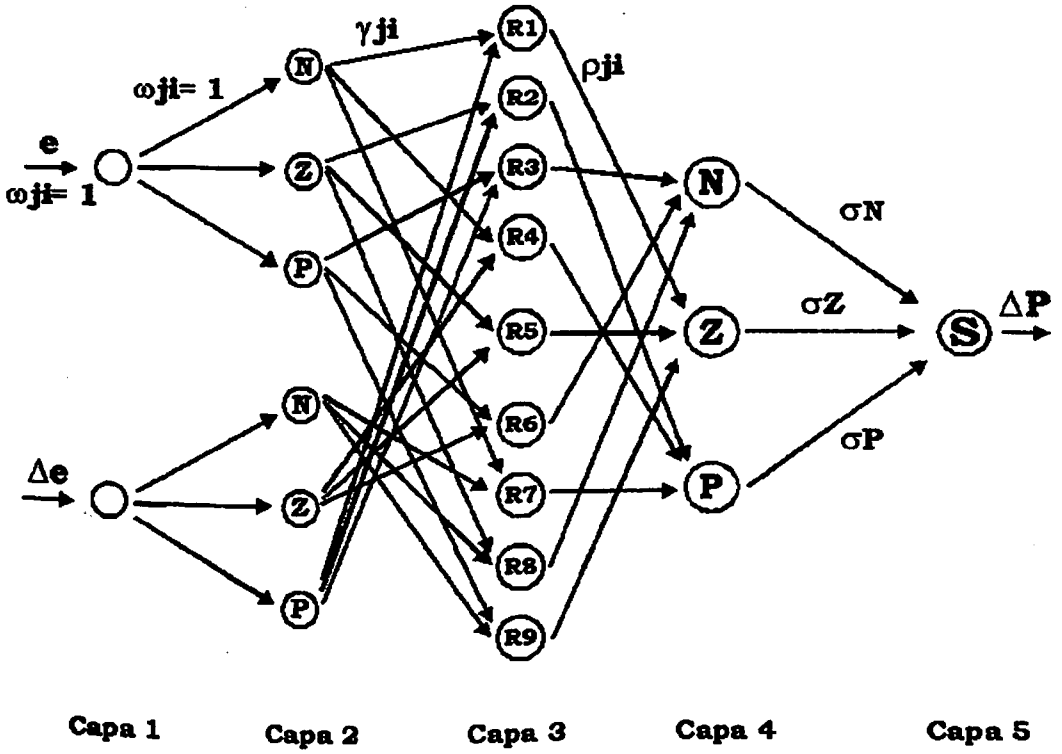


Figura 1. Red Híbrida neuro-difusa que implanta el control difuso de 9 reglas del ejemplo.

En los nodos de la capa 3, los parámetros de entrenamiento son los pesos de las conexiones entre estos nodos y los nodos de difusión, puesto que los parámetros internos no deberían modificarse, porque se cambiaría drásticamente la T-norma que se aplica, aunque queda abierta esta posibilidad de cambio. P. Werbos ha llamado a los pesos de las conexiones entre la capa 2 y la capa 3 elasticidades γ_{ij}, y muestra como

variando estos valores, es posible dejar intactas las reglas como relación entre antecedente y consecuente, lo que permite preservar la inteligibilidad del sistema, incluso después de ser sometido a un proceso de entrenamiento (Werbos, P. 1993). Una interpretación a la luz del efecto final de estos pesos, que no es mas que reforzar un miembro del antecedente frente a los otros, podría ser que estas elasticidades se relacionan con un cierto tipo de meta-conocimiento, que el operador humano ejecuta, pero que no puede detectarse con facilidad mediante técnicas de ingeniería del conocimiento y que está en los datos de entrada y salida del sistema. Debe recalcarse como estos nodos de reglas de la capa 3, no realizan la inferencia de cada regla. Ellos, solo realizan el calculo del factor de pertenencia de la proposición difusa antecedente. Pero, la conexión entre cada nodo de regla y la capa siguiente, representa la regla completa.

La capa 4, denominada capa de inferencia o de conjuntos del consecuente, es la encargada de tomar los valores de salida de los nodos de regla de la capa anterior, y mediante el uso de uno de lo criterios comunes de interpretación de la implicación $p \Rightarrow q$ evalúa la salida de la regla total:

SI p **ENTONCES** q.

Las entradas a cada nodo en esta capa, están ponderadas mediante pesos que modifican la participación que tiene el resultado del antecedente difuso, sobre la conclusión final que tendrá dicha regla. Estos pesos ρ_{ij}, del mismo modo que las elasticidades γ_{ij}, actúan como meta-conocimiento, que puede ser modificado durante el aprendizaje. En los nodos de esta capa, los parámetros de entrenamiento son los pesos ρ_{ij} de las conexiones que llegan de la capa anterior. Los parámetros internos de cada nodo no se deben modificar, pues hacerlo causa un cambio de la implicación que se usa, además de acuerdo con el tipo de concreción que se haga en la capa 5, uno de los parámetros de cada nodo puede ser el centro de los conjuntos difuso de llegada o valor lingüístico de la salida recomendada al *elemento final de control* (EFC).

Por último, la capa 5, salida en esta RHND, es la encargada de realizar la concreción (*deffuzification*), que permite convertir el conjunto difuso que resulta de la inferencia de todas las reglas, en un valor numérico, que en última instancia es el que se aplica al EFC del proceso. En los nodos de esta capa, se puede aplicar cualquier método de concreción, de las muchos que se han planteado al momento y dependiendo del que se escoja, la ponderación de las conexiones que llegan a estos nodos ρ_{ij}, pueden ser los centros de los conjuntos difusos del consecuente o estar fijos a uno, siempre que estos centros queden como parámetro interno en los nodos de la capa 4. Vale la pena resaltar que el entrenamiento puede modificar los valores de estos centros en cualquiera de los dos casos.

6. Entrenamiento de la red híbrida neuro-difusa. Sin importar la topología interna que se use, siempre que se respete el desempeño del control difuso en la red híbrida, es posible disponer de un buen número de parámetros de aprendizaje, sobre los cuales recae la tarea de "representación" y almacenamiento del conocimiento, o sea, el aprendizaje. El ajuste de estos parámetros se puede hacer supervisadamente, mediante cualquier técnica de entrenamiento de redes: *algoritmos genéticos* (Holland, J. H. 1975), *transmisión química bacteriana* (*Bacterial Chemotaxis*, Bremermann, H.J. and Anderson, R. W. 1990) e incluso retropropagación del error, con algunas modificaciones. Cualquier técnica que se use debe poder acceder sin distingo a todos los parámetros ajustables de la red, que aquí a diferencia de las RNA son mixtos: internos a los nodos y externos a estos como ponderaciones de conexiones.

Es importante mencionar el significado que puede darse a los parámetros ajustables, al clasificarlos como parámetros de ajuste grueso (aprendizaje) y parámetros de ajuste fino (adaptación o sintonía). Esta diferenciación es importante cuando se aspira a construir un control con características inteligentes, porque no todos los cambios del proceso son permanentes o de una duración temporal suficiente para provocar un cambio radical en las características y constitución del controlador (aprendizaje), y por el contrario, pueden ser corregidos con una sintonía o ajuste fino sobre los parámetros que permitan cambios leves (sintonía y adaptación).

7. Aplicación. La red propuesta fue desarrollada para implantar el control difuso de un reactor de lecho fluidizado, denominado *activador*, en el cual se realiza la gasificación remanente y activación del carbón molido, después del proceso de gasificación principal en un reactor precedente, denominado *pirolizador*. El activador hace parte de la planta piloto para la obtención en continuo de carbón activado en lecho fluidizado, método en proceso de patente por parte del grupo de investigación en carboquímica de la Facultad de Minas de la Universidad Nacional de Colombia, Sede Medellín (Aguirre y col., 1991). En este tipo de procesos, la diversidad de reacciones químicas que ocurren, además de los cambios súbitos en las variables de carga (carbón alimentado, flujo de gases de fluidización, etc.), hacen que un control tradicional PID no pueda manejar con eficiencia la globalidad del proceso: pirolización y activación en serie, e incluso se quede corto en el control individual de cada reactor.

El proceso se simuló mediante balances de materia y energía, así como correlaciones experimentales de diverso autores sobre el fenómeno reactivo y de transferencia de masa. Para la simulación y puesta en operación de la RHND en un computador secuencial, se desarrolló un código en C++, el cual permite la definición de nodos con funciones particulares, pero que dispone de 4 tipos base, de los cuales se desprenden subtipos relacionados. Estos tipos base son:

- Tipo 0. Nodos funcionales (capas 1 y 4), que aplican una o mas funciones.
- Tipo 1. Nodos difusos (capa 2), que entregan el factor de pertenencia μ.
- Tipo 3. Nodos de *T*-normas (capa 3) para operaciones lógicas *Y*.
- Tipo 4. Nodos de concreción (capa 5), que entregan un valor numérico.

Para que los parámetros de sintonía, adaptación y aprendizaje, estén disponibles con toda facilidad, el sistema almacena todos los parámetros internos y externos (pesos) de los nodos en un archivo matricial. Cuando ya se tiene operando la red, el objeto *NodoRed* posee funciones miembro que permiten el cambio de parámetros internos, mientras que el objeto red posee dos funciones independientes, una dedicada a sintonía y

adaptación y otra dedicada al aprendizaje. Ambas pueden variar parámetros internos y externos a los nodos.

La Tabla 2 muestra el desempeño del mejor controlador PI y el control difuso implantado con una RHND, cuando se aplican diversas perturbaciones en la carga de carbón semi-procesado (*char*) al activador. Este desempeño se mide en términos del sobre impulso en la temperatura del reactor y en el tiempo de establecimiento que requiere el controlador para volver al valor prefijado (en este caso 1173 °K) la temperatura del reactor.

Tipo de control y efecto medido.	**CONTROL PI**		**CONTROL DIFUSO**	
Perturbación en la carga del char.	Sobre impulso en °K.	Tiempo Establec. en minutos.	Sobre Impulso en °K.	Tiempo Establec. en minutos.
Aumento del 50% en flujo de char	14.8	> 30	2.2	7.0
Disminución del 50% en flujo de char	15.0	> 30	1.3	5.3

Tabla 2. Resultados comparativos de los dos controladores.

Como puede apreciarse en la Tabla 1, el desempeño del control difuso resultó muy superior comparado con el comportamiento del mejor control PI que se puede diseñar para este reactor, lo que demuestra la superioridad de esta técnica cuando se enfrenta al control de sistemas complejos.

8. Conclusiones. Entre los aspectos que hacen atractivo el uso de RHND para la implantación de controles difusos, frente al camino tradicional de usar un programa (shell) de sistema experto, pueden mencionarse:

- Capacidad de aprendizaje sin que se pierda significado de las reglas originales o se presenten distorsiones severas en la interpretación del sistema después del entrenamiento.

- Disponibilidad de algoritmos de entrenamiento supervisado, ampliamente probados en RNA, los que pueden usarse con muy pocas modificaciones en las RHND.
- Menor tiempo de inferencia frente a las implantaciones con programas de SE, si además se procesa de la forma masivamente paralela, puesto que se evitan las arduas búsquedas en árboles de decisión.
- La facilidad de programación de una RHND frente a la complejidad inherente de los programas de SE. Se permite la generación automática de código con mucha mas facilidad.

La principal desventaja que presenta esta aproximación es su capacidad limitada de explicación cuando se compara con la que tienen los programas de SE. Esta tarea que informa al usuario sobre el razonamiento que se uso para obtener una conclusión, es de importancia capital en las labores de tutoría que buscan mantener un grado mínimo aceptable de pericia en los operadores. Esta característica podría mejorarse mediante técnicas de grafos que indiquen, en un esquema de la RHND, el camino que tomó la inferencia.

Como punto final puede decirse que este tipo de mezclas de técnicas de procesamiento de información, hacen que el conjunto sea mejor que las partes, lo que trae como consecuencia la formación de estructuras con alta sinergía, que encajan muy bien en todo el marco de desarrollo de los sistemas autónomos de control.

BIBLIOGRAFÍA

Aguirre, J., Ocampo, A. y Espinel, J. *Desarrollo de un proceso para producción de carbón activado.* Informe de investigación, Universidad Nacional de Colombia, Sede Medellín. 1991.

Bremermann H. J. and Anderson R. W. *An Alternative to Back-propagation.* Center of Pure and Applied Mathematics. University of California. Internal Publication number PAM-483. Jan. 1990.

Driankov, D, Hellendomm, H and Reinfrank, M. *An Introduction to Fuzzy Control.* First edition. Springer-Verlag, Berlin-New York. 1993.

Freeman, J.A. and Skapura, D.M. *Redes neuronales, algoritmos, aplicaciones y técnicas de programación.* Primera edición en español, Addison-Wesley / Díaz de Santos. 1993.

Fukuda, T. and Shibata, T. *Theory and Applications of Neural Networks for Industrial Control Systems*. IEEE Transactions on Industrial Electronics. **39** (6), Dec. 1992.

Holland, J.H. *Adaptation in Natural and Artificial Systems*. Second print of original text (Michigan University 1975), for MIT Press, 1993.

Karr, C.L. and Gentry, E.J. *Fuzzy Control of PH Using Genetic Algorithms*. IEEE Transactions on Fuzzy Systems. **1** (1). Feb. 1993.

Li, Y.F. and Lau, C.C.. *Development of Fuzzy Algorithms for Servo Systems*. IEEE Control Systems Magazine. April 1989.

Lin, C. and Lee, C.S. *Neural-network-based Fuzzy Logic Control and Decision System*. IEEE Transactions on Computers. **40** (12), Dec. 1991.

Pérez, G., Rojas, G. y Urquijo, J. *Expert System for Real Time Control of Rotary Cement Kiln*. The World Congress on Expert Systems Proceedings. 1991.

Self, K. *Designing with Fuzzy Logic*. IEEE Spectrum. November 1990.

Sugeno, M. and Park, G. *An Approach to Linguistic Instruction Based Learning*. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems. **1** (1), Sept. 1993.

Werbos, P.J. *Neurocontrol and Elastic Fuzzy Logic Capabilities, Concepts and Applications*. IEEE Transactions on Industrial Electronics, **40** (2), April 1993.

Yamakawa, T. *A fuzzy Inference Engine in non linearAnalog Mode and its Application to a Fuzzy Logic Control*. IEEE Transactions on Neural Networks. **4** (3), May 1993.

HEBB PROBABLEMENTE APROXIMA A BAYES

Jorge Ricardo Cuéllar*
Siemens Corporate Research
ZFE T SE 11, D-81730 Munich

1. Introducción.

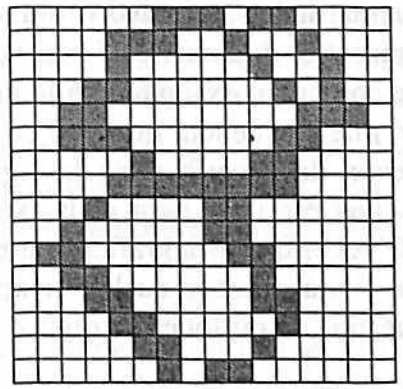


Figura 1. Un computador no "ve" un ocho. Sólo ve una matriz de ceros y unos.

Para los lectores, un símbolo "8" escrito sobre un sobre de correo se asocia inmediatamente y sin dificultad al número ocho (usualmente). Supongamos que usamos una cámara para fotografiar cifras escritas y (después de un preprocesamiento) las registramos como matrices de ceros y unos (o gráficamente, como rectángulos de puntos o cuadrados blancos y negros). Para un computador, cada cifra escrita es simplemente una de millones de millones de posibles gráficas de ceros y unos (ver la figura 1). Probablemente dos ochos escritos tomados al azar son dos

* E-mail Jorge.Cuellar@zfe.siemens.de

matices diferentes. Peor todavía: es posible que alguien escriba un nueve igual a como otra persona escribe a veces un ocho. Si uno supiera con qué frecuencia se escriben los ochos y los nueves de tal y tal forma, entonces sería posible encontrar una estrategia ideal para clasificar los ochos y los nueves. Esta estrategia es óptima en el sentido de que, aunque pueda cometer errores, es la estrategia que produce en promedio el menor número de errores. Esta estrategia se llama la "decisión bayesiana" (o de Bayes) [4].

Desafortunadamente, es totalmente imposible, en la práctica, saber con qué frecuencia los ochos y los nueves corresponden a tales y cuáles gráficas de ceros y unos. Suponiendo que usamos 16 por 16 puntos para representar los dígitos, hay $2^{16\times16} = 2^{256}$ posibles figuras: dibujos, letras, dígitos y mamarrachos, la mayoría sin sentido para los lectores. Si tomamos una muestra, digamos de 5 mil ochos diferentes, habrá muy pocas figuras que coincidan exactamente la una con la otra. Parece imposible estimar con qué frecuencia un ocho se escribe de tal o cual forma. La forma ingenua de hacerlo sería la siguiente: tomamos una muestra enorme de ochos escritos; para cada posible gráfica (matriz) contamos el número de veces que esta gráfica aparece y dividimos por el tamaño de la muestra. ¡Para obtener cualquier aproximación útil, tendríamos que tomar muestras de millones de dígitos "8"!

Por otra parte, existen redes neuronales que parecen resolver el problema de clasificación óptica de caracteres de una forma adecuada ([6], [7], [5], pág. 139).

El objetivo de este artículo es introducir una red neuronal con una nueva variación de la famosa regla de Hebb (para el caso del reconocimiento óptico) y explicar en qué sentido esta regla probablemente aproxima la decisión bayesiana. Además, el artículo es una introducción a la noción de *pab* (*probably almost bayes*) ([1], [8], [2], [3]), que a su vez es una variación de la noción *pac* (*probably almost correct*) de Valiant [9].

2. Los tres problemas. Imaginemos todos los números cero, uno, dos, ..., nueve que han sido escritos a mano en cheques, sobres de correo, en

libros de contabilidad, en tableros escolares, etc., etc., durante este siglo, en todo el mundo. Cada uno de ellos lo llamaremos un ejemplo. Se puede uno imaginar que hay millones de millones de ejemplos. Cada uno "es" un cero, o un uno, o un dos, ..., o un nueve. Si alguien escribió un ocho pero parece un nueve, decimos que éste es un ejemplo de un ocho, aunque la mayoría de las personas o de algoritmos de clasificación insistirían en que parece un nueve. Hay que distinguir lo que "es" de lo que "parece". Si el ejemplo X "es" un ocho, escribimos $c(X) = 8$.

Imaginemos que a cada uno de esos números lo representamos (después de fotografiar, cambiar de tamaño, etc.) como una matriz de 16×16 de ceros y unos, o, más simplemente (pero menos gráficamente), como un vector binario de tamaño $16 \times 16 = 256$. Cada uno de los bits $X_1,\ldots,X_{256}$ de este vector lo llamamos un *rasgo* o *característica* del ejemplo X. En general, cada ejemplo X tiene asociado un vector binario rasgos(X) = $(X_1,\ldots,X_N)$ de características o rasgos. El conjunto de todos los ejemplos es la *población* $\mathcal{X}$. Además, tenemos un conjunto $\mathcal{C}$ de *conceptos* (en nuestro ejemplo, los conceptos son uno, dos, ..., nueve). Cada ejemplo es un ejemplo de un concepto, es decir, existe una función $c : \mathcal{X} \to \mathcal{C}$.

O imaginemos que Interpol, el FBI, el DAS, etc., tienen entre todos una cantidad enorme de fotos de criminales. $\mathcal{C}$ será el conjunto de presuntos criminales, X el conjunto de fotos. Cada foto X "es" una foto de un criminal $c(X) \in \mathcal{C}$. Cada foto se puede catalogar por sus apariencias: escogiendo un vector de características, por ejemplo, "nariz recta", "pelo claro", "ojos negros", ..., obtenemos una función $\mathcal{X} \to \{0,1\}^N$ de rasgos. La foto X corresponde al vector (1,0,1,...) si X tiene la nariz recta, no tiene el pelo claro, tiene ojos negros, etc. Observemos que cada criminal tiene muchas fotos, y que los rasgos de las fotos de un mismo sospechoso no tienen por qué ser idénticas: según el momento, el lugar, el disfraz, etc., varían los rasgos de las fotos de la misma persona. Es decir, es posible que[1]

[1] Reservamos la letra X para presentar un ejemplo. $X^1, X^2, \ldots$ son también ejemplos. $X_1,\ldots,X_N$ son los rasgos del ejemplo X. X_i^j es el rasgo i del ejemplo X^j. El vector $(X_1,\ldots,X_N)$ se representará con $\overline{X}$. Obsér-

$$\exists X^1, X^2 \in \chi\,(c(X^1) = c(X^2) \quad \wedge \quad \text{rasgos}(X^1) \neq \text{rasgos}(X^2)).$$

Inclusive, es posible que dos fotos de dos criminales diferentes sean idénticas en todas sus características o rasgos escogidos para catalogarlas. Es decir, es posible que

$$\exists X^1, X^2 \in \chi\,(c(X^1) \neq c(X^2) \quad \wedge \quad \text{rasgos}(X^1) = \text{rasgos}(X^2)).$$

Supongamos, ahora, que nos ponemos en el papel de jueces que deben decidir si un vector $\overline{X}$ de rasgos corresponde a un concepto $c \in \mathscr{C}$ o no. Como se hace con frecuencia en ciertos concursos de deportes, de cine o de belleza, dado un concepto c y una serie de rasgos $\overline{X}$, damos un puntaje $G_c(\overline{X})$, dado por un número real. El valor de $G_c(\overline{X})$ representa nuestra convicción de que los rasgos $\overline{X}$ corresponden al concepto c. Formalmente, G, la colección de funciones discriminantes, es una función $\mathscr{C} \rightarrow (\{0,1\}^N \rightarrow \mathbf{R})$, que a cada concepto $c \in \mathscr{C}$ le asocia una función discriminante $G_c : \{0,1\}^N \rightarrow \mathbf{R}$, que, a su vez, le asocia a cada vector de rasgos $\overline{X} \in \{0,1\}^N$ el "puntaje" $G_c(\overline{X}) \in \mathbf{R}$. Este valor de $G_c(\overline{X})$ indica numéricamente si la "apariencia" $\overline{X}$ corresponde a la "verdad" $c(X) = c$. En la medida en que $G_{c_1}(\overline{X})$ es mayor que $G_{c_2}(\overline{X})$, más convencidos estamos de que los rasgos $\overline{X}$ corresponden más bien al concepto c_1 que al concepto c_2. Es decir, el conjunto de funciones discriminantes G_c, $c \in \mathscr{C}$, determina la *decisión*

$$D : \{0,1\}^N \rightarrow \mathscr{C}$$

si y sólo si

$$D(\overline{X}) = c_1 \quad \Leftrightarrow \quad \forall c_2 \neq c_1 \; G_{c_1}(\overline{X}) \geq G_{c_2}(\overline{X}).$$

(Un conjunto de funciones discriminantes puede determinar varias decisiones si hay empates: $D(\overline{X})$ puede escogerse como c_1 o como c_2 si $G_c(\overline{X}) = G_{c_2}(\overline{X})$ y los otros valores de $G_{c_1}(\overline{X})$ son más pequeños).

vese que $\overline{X}$ tiene menos información que X (porque $c(X)$ no depende de los rasgos, o, con otras palabras, las apariencias engañan) $\overline{X}^1$ es el vector de rasgos del ejemplo X^1, etc.

En el caso de información perfecta (conocemos toda la población χ, la verdadera clasificación $c : \chi \rightarrow \mathscr{C}$, los rasgos de cada ejemplo, rasgos: $\chi \rightarrow \{0,1\}^N$ y la probabilidad $P(X)$ de obtener aleatoriamente el ejemplo X), es fácil entonces tomar la mejor decisión:

$$G_c^{opt}(\overline{X}) = P(c(X) = c) \qquad P(\text{rasgos}(X) = \overline{X} \mid c(X) = c)$$

(X se considera como una variable aleatoria: $P(c(X)) = c$) es la probabilidad de que esa variable corresponda al concepto c, y $P(\text{rasgos}(X) = \overline{X} \mid c(X) = c)$ es la probabilidad de que X tenga los rasgos $\overline{X}$, dado que corresponde al concepto c).

La decisión D^{opt} dada por las funciones determinantes G_c^{opt} se llama la *decisión bayesiana*. Entre todas las decisiones posibles, $D : \{0,1\}^N \rightarrow \mathscr{C}$ es la decisión que minimiza el error:

$$\text{error}(D) = P(D(\text{rasgos}(X) \neq c(X)).$$

Desafortunadamente, estamos *lejos* (*muy* lejos) de la información perfecta: la probabilidad *a priori* de obtener el concepto c, $P(c(X)) = c$), y la probabilidad condicional $P(\text{rasgos}(X) = \overline{X} \mid c(X) = c)$ son muy desconocidas. Este es el primer problema: el *problema del épsilon*.

Si bien no podemos obtener la decisión optimal, tal vez nos conformaríamos con una decisión bastante cercana. Decimos que una decisión D aproxima a la decisión bayesiana D^{opt} si

$$\text{error}(D) \leq \text{error}(D^{opt}) + \varepsilon.$$

(Obsérvese que usamos el verbo aproximar, como un verbo parametrizado: $\text{aproximar}_{0.1}$, $\text{aproximar}_{0.01}$, $\text{aproximar}_{0.01}$, etc. son ejemplos.)

¿Cómo esperamos obtener una decisión D que $\text{aproxima}_{\varepsilon}$ a D^{opt}? La idea es tomar una muestra grande ejemplos para construir funciones G_c y una decisión D.

Aquí aparece el segundo problema (el *problema del delta*): si la muestra que tomamos es mala, los ochos dela muestra parecen nueves y los cincos parecen sietes, de modo que la función que obtenemos así estará lejos de la optimal. No todas las muestras nos van a producir una discriminante buena.

Supongamos que tenemos un método A para construir, dada una muestra $\mathcal{M}$, una colección de funciones discriminantes G_c. El método A es bueno si la decisión que produce tiene una alta probabilidad de aprender$_\varepsilon$ a la optimal. Formalizando un poco lo dicho: sea $\mathcal{P}_M(\chi)$ el conjunto de las muestras de tamaño M tomadas de la población χ. $\mathcal{P}_M(\chi)$ es un espacio de probabilidad: cada muestra $\mathcal{M} \in \mathcal{P}_M(\chi)$ tiene una probabilidad $P(\mathcal{M})$ de ser escogida.

Un algoritmo de aprendizaje es una unción (efectivamente calculable)

$$A : \mathcal{P}_M(\chi) \to (\mathcal{C} \to \{0,1\}^N \to \mathbf{R})$$

que a cada muestra $\mathcal{M}$ le asocia una colección de funciones discriminantes.

Entonces decimos que el algoritmo de aprendizaje A "probablemente aproxima$_\varepsilon$" a la decisión optimal, si la probabilidad de que $A(\mathcal{M})$ aproxime a la decisión bayesiana sea mayor que $1-\delta$.

El tercer problema (el *problema de M*) es el siguiente: claro que hay una solución "buenísima": tome muestras super inmensas (o mejor, mire *todos* los ejemplos que existen) y luego use el método más simple de contar frecuencias y estimar probabilidades. Una nueva restricción elimina esta posibilidad: el tamaño de la muestra no puede ser excesivo.

Dado un problema de aprendizaje

$$((\chi, P), \mathcal{C}, c : \chi \to \mathcal{C}, \text{rasgos: } \chi \to \{0,1\}^N),$$

un algoritmo A se llama *pab* (*probably almost bayes*) si podemos encontrar un polinomio p tal que si usamos $M = p(1/\varepsilon, 1/\delta)$ como tamaño de la muestra, el algoritmo probablemente$_\delta$ aproxima$_\varepsilon$ la decisión bayesiana.

Si el problema se puede parametrizar con N (por ejemplo, con el número de pixeles que se use para representar las imágenes), entonces queremos que $M = p(N, 1/\varepsilon, 1/\delta)$.

Un problema de aprendizaje es *pab-aprendible* si existe un pab algoritmo A para el problema.

3. La solución. Ahora nos restringimos otra vez al caso de los dígitos escritos a mano y asumimos que en la población χ cada dígito aparece con igual frecuencia, es decir,

$$P(c(X) = 0) = P(c(X) = 1) = \ldots = 1/10.$$

Entonces la decisión bayesiana es:

$$G_i^{opt}(\overline{X}) = P(\text{rasgos}(X) = \overline{X} \mid c(X) = i). \qquad (*)$$

Las dos ideas importantes son las siguientes:

- Vamos a aproximar las probabilidades $\mathcal{P}$ de la mano derecha de (*) con valores $\hat{\mathcal{P}}$ *multiplicativamente*:

$$\frac{1}{1+\varepsilon} \le \frac{\mathcal{P}}{\hat{\mathcal{P}}} \le 1+\varepsilon.$$

- Si una probabilidad $\mathcal{P}$ es demasiado pequeña (y, por lo tanto, prácticamente imposible de estimar), no nos molestaremos: un error en su estimación tampoco puede tener consecuencias graves, ya que se trata de un evento muy escaso. Es decir, si $\mathcal{P}$ es muy pequeña, no importa si $\hat{\mathcal{P}}$ aproxima o no multiplicativamente a P. Lo mismo vale para las probabilidades $\mathcal{P}$ cercanas a 1.

Consideremos primero un caso imaginario: si los rasgos (= pixeles) fueran independientes, entonces

$$P(\text{rasgos}(x) = (\overline{X}) \mid c(x) = c)$$
$$= P(\text{rasgos}_1(X) = X_1 \mid c(x) = c) \times \ldots \times P(\text{rasgos}_N(X) = X_N \mid c(x) = c)$$
$$= P_1(X_1) \times \ldots \times P_N(X_N)$$
$$= \prod_k p_k^{X_k} (1 - p_k)^{1 - X_k},$$

donde

$$p_k := P_k(X_k) := P(\text{rasgos}_k(X) = X_k \mid c(X) = c). \qquad (**)$$

Esto es cierto ya que

$$P_k(1) = p_k = p_k^1 (1 - p_k)^{1-1} \quad \text{y} \quad P_k(0) = 1 - p_k = p_k^0 (1 - p_k)^{1-0}.$$

Entonces, en vez de estimar todas las posibles probabilidades $P(\text{rasgos}(X) = \overline{X} \mid c(X) = c)$ para todos los 2^{256} posibles valores de $\overline{X}$, lo que se necesita es estimar las probabilidades $p_1, \ldots, p_{256}$ definidas en (**) con una "estimación-(ε, δ)":

$$P\left(\frac{1}{1+\varepsilon} \leq \frac{p_i}{\hat{p}_i} \leq 1 + \varepsilon \right) \geq 1 + \beta,$$

si p_i no está demasiado cercano a 0 o a 1. En tales casos tomamos $\hat{p}_i$ = 0 o 1, respectivamente.

Este procedimiento funciona: si los rasgos del problema son independientes, el problema es pab-aprendible. Desafortunadamente, los rasgos no son independientes: si hay un 0 o un 1 en un cierto punto de la gráfica, esto *sí* está relacionado con la presencia de ceros y unos en la *cercanía* del punto. ¡Si esta cercanía no es muy grande, el problema es pab! (Por esto el título del artículo debiera ser*: Hebb tal vez probablemente aproxima a Bayes*. El *tal vez* se refiere a esta hipótesis.)

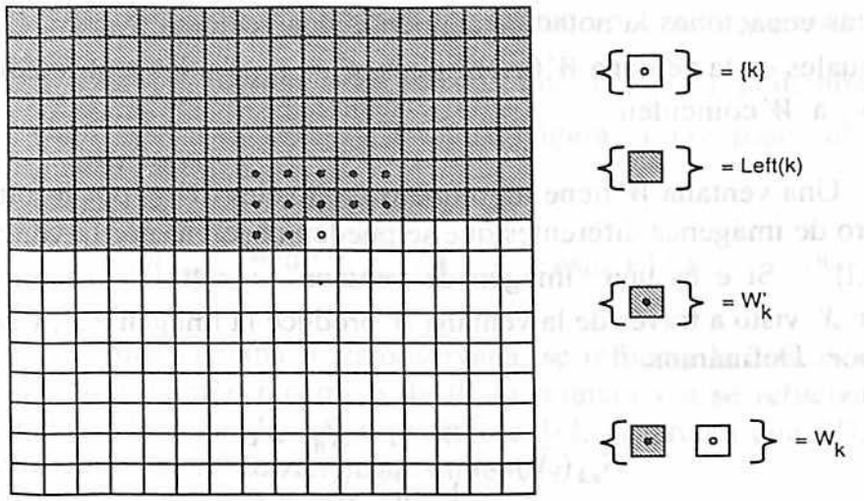


Figura 2. Dado el pixel k, Left(k) representa a todos los pixeles anteriores a k, W_k' es una ventana en Left(k) de puntos cercanos a k, y W_k es la ventana W_k' junto con el punto k.

Sea Left(k) = {1,2,3,...,k-1}. Gráficamente (figura 2), Left(k) es el conjunto de pixeles "antes" de la posición. Sea W_k una "ventana" alrededor del punto k, que contiene además de k solamente un conjunto de puntos en Left(k) cercanos a k. Sea $W_k' = W_k \setminus \{k\}$. Dado que $c(X)$ es el dígito "i", la probabilidad

$$P_i(\text{rasgos}(X) = \overline{X})$$
$$= \prod_k P_i((\text{rasgos}_k(X) = \overline{X}_k \mid \text{rasgos}_{1,\ldots,k-1}(x) = (X_1, \ldots, X_{k-1}))$$

se puede aproximar usando las ventanas W_k así: asumiendo que el pixel X_k no depende de todo $(X_1, \ldots, X_{k-1})$ sino solamente de la ventana $W_k' \subseteq \{X_1, \ldots, X_{k-1}\}$, podemos escribir

$$P_i(\text{rasgos}_k(X) = \overline{X}_k \mid \text{rasgos}_{1,\ldots,k-1}(x) = (X_1, \ldots, X_{k-1}))$$
$$= P_i(\text{rasgos}_k(X) = \overline{X}_k \mid \text{rasgos}(X)_{W_k} \overline{X})$$
$$= \frac{P_i(\text{rasgos}(X) =_{W_k} \overline{X})}{P_i(\text{rasgos}(X) =_{W_k'} \overline{X})}$$

En estas ecuaciones la notación siguiente: dos vectores v_1 y $v_2 \in \{0,1\}^N$ son iguales en la ventana W (escribimos $v_1 =_W v_2$) si las restricciones de v_1 y v_2 a W coinciden.

Una ventana W tiene un número de pixeles $|W|$ y, por lo tanto, el número de imágenes diferentes que se pueden observar por la ventana W es $\{0,1\}^{|W|}$. Si e es una "imagen de ventana" $e \in \{0,1\}^{|W|}$, entonces el vector $\overline{X}$ visto a través de la ventana W produce la imagen e si, y sólo si, $\overline{X}|_W = e$. Definamos

$$S_{e,k}(\overline{X}) = \begin{cases} 0 & \text{si} \quad \overline{X}_{W_k} \neq e \\ 1 & \text{si} \quad \overline{X}_{W_k} = e. \end{cases}$$

Observemos que $S_{e,k}(\overline{X})$ es un polinomio en (de orden $|W|$). Llamemos $P_{e,k,i}$ la probabilidad condicional de que dado $c(X) = i$, la imagen de X en la ventana sea e. Similarmente, $P'_{e,k,i}$ para la ventana W_k' y la imagen e'.

El lector podrá comprobar que

$$P_i(\text{rasgos}(X) =_{W_k} \overline{X}) = \prod_{e \in \{0,1\}^{W_k}} P_{e,k,i}^{S_{e,k}(\overline{X})},$$

ya que tan sólo para una imagen e se tiene $S_{e,k}(\overline{X}) = 1$, mientras que para el resto de ellas es 0. Juntando las piezas, tenemos:

$$P(\text{rasgos}(X) = \overline{X} \quad \cdot \quad c(X) = i) = \prod_k \frac{\prod_e (P_{e,k,i})^{S_{e,k}(\overline{X})}}{\prod_{e'} (P_{e',k,i})^{S_{e',k}(\overline{X})}};$$

tomando logaritmos, encontramos

$$G_i(\overline{X}) = \sum_k \sum_e \ln_{P_{e,k,i}} S_{e,k}(\overline{X}) - \sum_k \sum_{e'} \ln_{P_{e',k,i}} S_{e',k}(\overline{X}).$$

Obsérvese que $G_i(\overline{X})$ es un polinomio en $\overline{X}$ (ya que $S_{e,k}$ y $S_{e',k}$ lo son) con coeficientes definidos como logaritmos de probabilidades o de frecuencias. Aproximando $P_{e,k,i}$ por la frecuencia observada y escribiendo $\ln s \approx \sum_1^s 1/k - \gamma$ (donde γ es la constante de Euler), podemos describir el algoritmo como una red neuronal (figura 3). Los pesos ω de la red neuronal se calculan así:

$$\omega_{\,updated} \approx \ln s \approx \sum_1^s 1/k - \gamma \approx \omega_{old} + 1/s.$$

Cada vez que una ventana W es observada, se refuerza la conexión correspondiente al campo receptivo de W: la primera vez se refuerza con un valor de 1, la segunda con un valor de 1/2, la tercera con 1/3, etc. (Las ventanas W' se refuerzan negativamente.)

Esta red corresponde a una variación de la ley de Hebb, donde existen campos receptivos asociados a las ventanas (estos campos receptivos no se aprenden), y donde el refuerzo va decayendo según lo que en optimización estocástica se conoce como el procedimiento de Robins-Monro.

Una característica peculiar de la red propuesta es su velocidad de aprendizaje: sólo es necesario ver cada ejemplo una sola vez. En nuestros experimentos, con 50 mil dígitos reales, la red tomó menos de 5 minutos para aprender con un error menor del 2% en la muestra y menor al 4% en la población total.

4. Resumen. Asumiendo que digitalizamos caracteres escritos en un matriz de N puntos, y suponiendo que cada pixel sólo depende del valor de los pixeles en un entorno de tamaño $p(N)$ (donde p es un polinomio en N), entonces el problema de clasificación óptica es pab aprendible: la red neuronal con una variante de la ley de Hebb descrita en la sección 2, probablemente aproxima la decisión óptima de Bayes. El tamaño de la red y el tamaño de la muestra requerida tan sólo dependen polinómicamente de N y del inverso de la exactitud (ε , δ) requerida

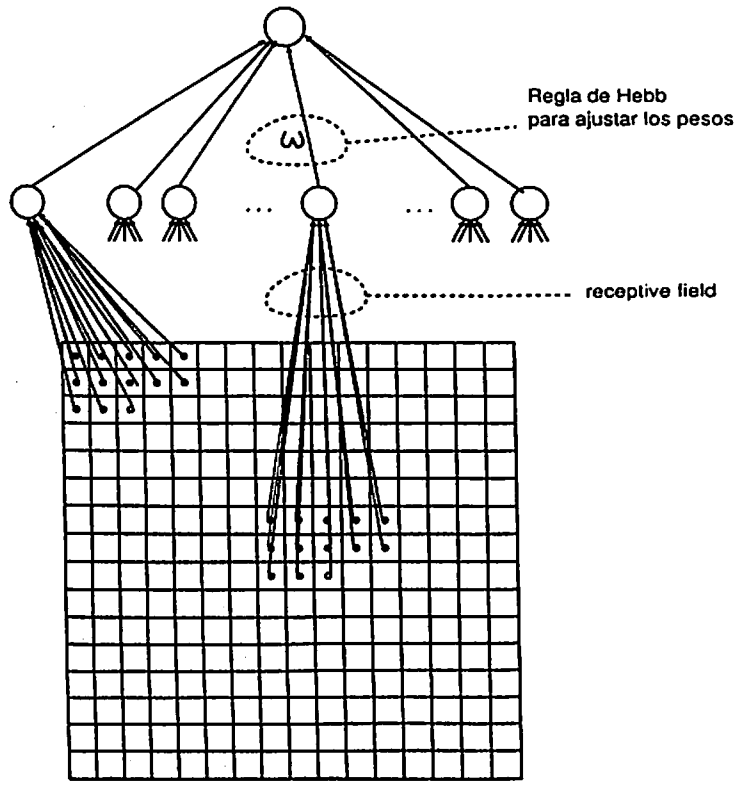


Figura 3. La red neuronal

REFERENCIAS

[1] Svetlana Anulova, Paul Fischer, Stefan Pölt, and Hans Ulrich Simon. *Pab-Decisions for Boolean and Real-valued features.* Proceedings of the fifth Annual ACM Workshop on Computational Learning Theory, ACM Press, New York, 1992.

[2] Jorge Ricardo Cuellar and Hans Ulrich Simon. *Neural Discriminant Analysis.* Forschungbericht Nr. 469 der Universität Dortmund, 1993.

[3] Jorge Ricardo Cuellar and Hans Ulrich Simon. *Neural Discriminant Analysis.* ALT93, Algorithmic Learning Theory, Proceedings LNCS 744, Springer-Verlag, 1993.

[4] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[5] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Volume 1 of Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, 1991.

[6] Y. Le Cun, B. Boser, J. S. Denker, S. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition.* Neural Computation 1 (1989), 541-551.

[7] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Handwritten Digit Recognition with a Backpropagation Network.*. Advances in Neural Information Processing Systems II (Denver 1989). De. D. S. Touretzky, 396-404. San Mateo: Morgan Kauffman, 1990.

[8] Stefan Pölt. *Extensions of the Pab-Decision Model.* Forschungsbericht Nr. 468 der Universität Dortmund, 1993.

[9] L. G. Valiant. *A Theory of the Learnable.* Communications of the ACM, 27 (11), (1984), 1134-1142.

LINEARIZING CONTROL USING A DYNAMIC NEURAL NETWORK

Alberto Delgado[+]
Electrical Engineering Department
National University of Colombia
Santafé de Bogotá, Colombia

Abstract. Along with the development of neural networks, there has been a great development in the area of non-linear control using Differential Geometry [6]. Control schemes have been developed which are usually complex. In this paper recurrent neural networks and linearizing control schemes proposed by Kravaris and Chung [9] are integrated.

Resumen. Paralelo al desarrollo de redes neuronales ha tenido un gran desarrollo el área de control no lineal usando Geometría Diferencial [6]. Se han elaborado esquemas de control generalmente complejos. En este trabajo se integran las redes neuronales recurrentes y los esquemas de control linealizables propuestos por Kravaris y Chung [9].

1. Introduction. Artificial Neural Networks (ANNs) have been shown to approximate non-linear mappings to any arbitrary degree of accuracy [1, 2]. It is this particular property of ANNs, which make them ideal candidates for modelling non-linear systems. ANNs can be categorised into two classes, namely feedforward networks and recurrent networks. Most current research in the application of neural networks to control has been carried out using feedforward networks. The approach used here is to formulate the problem in discrete time and is similar to the NARMA approach, see Narendra and Parthasarathy [3] and Cheng and Billings [4]. This method requires as inputs to the network a number of past values for each physical input and output of the system, where the number of past inputs and outputs needed are determined by trial and error.

An alternative to the feedforward network is the recurrent neural network first introduced by Hopfield [5] in the context of associative memories for pattern recognition. Recurrent neural networks have two

[+] Also: Cybernetics Department, University of Reading. E-mail: shrdelga@reading.ac.uk

salient features that distinguish them from feedforward networks. One is their node characteristics, which involve non-linear differential equations, while feedforward networks have only static non-linear node characteristics. The other major distinction is topology, in recurrent networks there are both feedforward and feedback connections, in other words the network is fully connected.

Along with the development of neural networks, there has been a tremendous development in the area of non-linear control using differential geometry [6]. Using the geometric approach, a number of control schemes, e.g. [7], [8] have been developed. These schemes require a state-space model of the system being controlled. So far most research has concentrated on using models developed from first principles. These models in order to retain accuracy are very complex, and not of much use for control. A simplified model, on the other hand is not a faithful representation of the system.

In this paper a control scheme which linearizes the system is discussed. The idea here is to integrate recurrent neural networks and the linearizing control scheme proposed by Kravaris and Chung [9]. The approach is to identify the non-linear plant using a recurrent neural network, and then synthesise the linearizing state feedback using this network.

The paper is organised as follows : section 2 deals with the definition of Lie derivatives and the concept of relative degree. The section 3 discusses the linearizing state feedback and the Globally Linearizing Control (GLC) structure. Section 4 presents the results of the identification after introducing the plant and the dynamic neural network. Finally, section 5, shows the simulation results of the GLC structure using the identified network to calculate the linearizing state feedback.

2. Mathematical preliminaries. Before proceeding any further, some definitions from differential geometry and Lie algebra are required.

2.1. Lie Derivatives. Consider a control affine single input - single output non-linear plant

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{1}$$

where $x \in R^n$, $u, y \in R$, $h(x)$ is a C^∞ scalar field on R^n and $f(x)$, $g(x)$ are C^∞ vector fields on R^n. For this system the Lie derivative of $h(x)$ with respect to $f(x)$ is given by

$$L_f h(x) = \sum_{i=1}^{n} \frac{\partial h(x)}{\partial x_i} f_i(x).$$

Further Lie derivatives can also be obtained, first along the vector field $f(x)$ and then along the vector field $g(x)$

$$L_g L_f h(x) = \sum_{i=1}^{n} \frac{\partial (L_f h(x))}{\partial x_i} g_i(x),$$

or recursively along the vector field $f(x)$

$$L_f^k h(x) = \sum_{i=1}^{n} \frac{\partial (L_f^{k-1} h(x))}{\partial x_i} f_i(x),$$

with $L_f^0 h(x) = h(x)$.

2.2. Relative Degree. The relative degree is an invariant characteristic of non-linear dynamic systems. This property of the non-linear system can be viewed as the number of times the external input, u, has to be integrated before it effects the output of the system, y.

More precisely, the *relative degree* r of a non-linear system given by (1) can be defined as satisfying the following :

1. $L_g L_f^k h(x) = 0$ for all x and all $k < r-1$.

2. $L_g L_f^{r-1} h(x) \neq 0$.

Thus, it can be seen that for a system with relative degree r

$$\frac{d^k y}{dt^k} = L_f^k h(x), \qquad k = 0,\ldots,r-1 \tag{2a}$$

$$\frac{d^k y}{dt^k} = L_f^k h(x) + L_g L_f^{r-1} h(x) u \ . \tag{2b}$$

3. Input - output linearization. Consider the feedback input

$$u = \frac{-L_f^r h(x) + v}{L_g L_f^{r-1} h(x)}, \tag{3}$$

with v an external input. Replacing (3) into (2b) yields the linearized equation

$$\frac{d^r y}{dt^r} = v \ .$$

The input (3) is called the linearizing state feedback or linearizing control, because it produces a linear system between the external input v and the system output y. Note that the state feedback cancels exactly the non-linear terms $L_f^r h(x)$ and $L_g L_f^{r-1} h(x)$.

In general, if the non-linear system (1) has relative degree r, then there is always a state feedback that makes the input - output $(v - y)$ behaviour of the closed loop system linear. This feedback is

$$u = \frac{v - L_f^r h(x) - \beta_1 L_f^{r-1} h(x) - \ldots - \beta_{r-1} L_f h(x) - \beta_r h(x)}{L_g L_f^{r-1} h(x)}. \tag{4}$$

Replacing (4) in (2b)

$$\frac{d^r y}{dt^r} = v - \beta_1 L_f^{r-1} h(x) - \ldots - \beta_{r-1} L_f h(x) - \beta_r h(x)$$

and using (2a), the input - output $(v - y)$ behaviour of the closed loop system is then governed by

$$\frac{d^r y}{dt^r} + \beta_1 \frac{d^{r-1} y}{dt^{r-1}} + \ldots + \beta_{r-1} \frac{dy}{dt} + \beta_r y = v. \tag{5}$$

The parameters $\beta_1, \ldots, \beta_r$ are based on the desired input - output characteristic, this means that the $(v - y)$ system can have arbitrarily placed poles. After linearizing the system (1), one can use an external PI loop

$$v = k_p\left[y_d - y\right] + k_i \int_0^t \left[y_d(t) - y(t)\right] dt \ , \tag{6}$$

to force the output $y(t)$ to track a given desired trajectory $y_d(t)$. This control structure was proposed by Kravaris and Chung [9] and it is called the Globally Linearizing Control (GLC) structure, see figure 1.

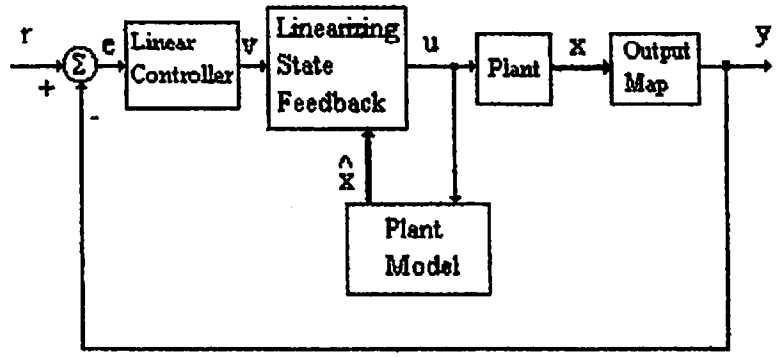


Fig. 1. Globally Linearizing Control (GLC) structure.
The design procedure using the GLC structure is simple : compute the linearizing state feedback (4) from the system model and then tune the PI controller (6).

4. Identification. As has been mentioned before, one of the prerequisites of the GLC is the availability of a model. A dynamic neural network, in the form of a Hopfield network, is used to represent the plant to be controlled. The recurrent network is used in two ways : (a) To calculate the

Lie derivatives of the linearizing feedback and (b) To generate, on -line, the state vector χ needed in the Lie derivatives.

4.1. **The Plant**. In order to test the GLC control structure using the dynamic neural network, a single link manipulator shown schematically in figure 2 was selected. This system can be represented by te following second order, non-linear, differential equation

$$m\ell^2\ddot{\theta}(t) + v\dot{\theta}(t) + mg\ell\sin\theta(t) = u(t),$$

where the length, mass and friction coefficients are $\ell = 1$ m, $m = 2.0$ kg and $v = 1.0$ kg.m^2/s, respectively [10].

The corresponding state representation of this system is as follows

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -9.8\sin x_1 - 0.5x_2 + 0.5u \\ y &= x_1 \end{aligned} \qquad (7)$$

with $x_1(0) = x_2(0) = 0$ and $y = \theta(t)$.

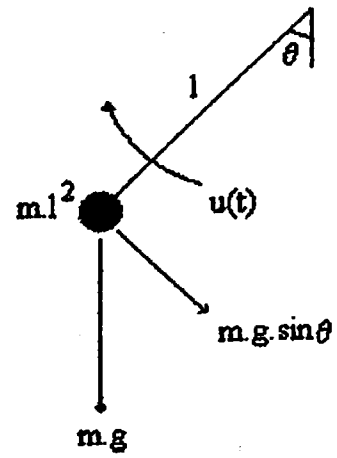


Fig. 2. Non-linear plant : single link manipulator.

The plant (7) was identified with the recurrent neural network

$$\begin{aligned} \dot{\chi} &= \tilde{f}(\chi) + \tilde{g}(\chi)u \\ y &= \tilde{h}(\chi) \end{aligned} \qquad (8a)$$

where

$$\tilde{f}(\chi) = -\frac{1}{\tau}\chi + \frac{1}{\tau}W\,\sigma(\chi) \qquad (8b)$$

$$\tilde{g}(\chi) = \Gamma$$

$$\tilde{h}(\chi) = \chi_1$$

$$u, y \in R, \chi \in R^N, W \in R^{NxN}, \Gamma \in R^{Nx1}$$

The model (8) is equivalent to the set of differential equations

$$\begin{aligned} \dot{\chi}_i &= \tilde{f}_i(\chi) + \tilde{g}_i(\chi)u \\ y &= \tilde{h}(\chi), \quad i = 1,\ldots N \end{aligned} \qquad (9a)$$

where

$$\tilde{f}_i(\chi) = -\frac{1}{\tau}\chi_i + \frac{1}{\tau}\sum_{j=1}^{N}\omega_{ij}.\sigma(\chi_j), \qquad (9b)$$

$$\tilde{g}_i(\chi) = \frac{1}{\tau}\gamma_i,$$

$$\tilde{h}(\chi) = \chi_1.$$

The figure 3 shows the non-linear plant and the dynamic neural network during the identification process.

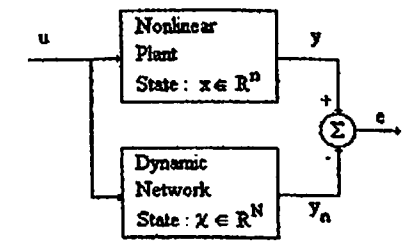


Fig. 3. The non-linear plant is identified with a dynamic neural network.

4.2.- Training. The dynamic neural network (8) was used to identify the non-linear plant (7) with $N = 10$ neurons, $\tau = 1.0$ and $\sigma(x) = \tanh(x)$. The network was trained repetitively over the fixed time interval $[0, t_f]$ and the matrices W, Γ were adjusted with the chemotaxis algorithm [11] to minimise the performance index (10). The training input was random noise and the initial conditions for the neuron states were selected at random, $t_f = 20s$.

$$J = \sqrt{\frac{1}{t_f}\int_0^{t_f} e^2(t).dt} = \sqrt{\frac{1}{t_f}\int_0^{t_f} \left[y(t) - y_d(t)\right]^2 .dt} \qquad (10)$$

The Fig. 4 shows the output of the plant and the output of the neural network for the training input, the performance index after the training was $J = 0.0071$.

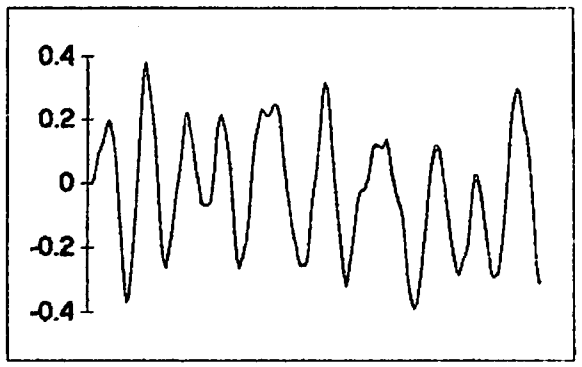


Fig. 4. Output of the plant (7) and output of the network (8) for the training input.

The parameters of the network after the training are shown below.

WEIGHT MATRIX : W

–0.2939	–0.3653	–0.2066	0.8793	–0.0018	0.2802	1.4299	0.1014	–1.8718	–0.5008
–0.6479	–0.2416	–0.5343	0.3697	–0.4885	0.1648	0.0286	0.5651	0.7921	0.1007
–0.0350	–03385	–0.2940	0.6123	–0.8631	–0.7015	0.1837	0.6268	–0.3450	–0.4412
–0.4586	0.2714	–0.2384	–0.5535	–0.1489	0.6143	–0.1640	–0.3803	–0.9092	–0.3313
1.2026	–0.1957	–0.1814	0.3155	0.2657	0.4056	0.5775	–0.5850	–0.5672	–0.8580
0.1457	0.9170	0.5478	0.5123	0.3164	0.1052	–0.4341	–0.0149	–0.2083	0.6149
–0.3362	0.8777	–0.1471	0.2594	–1.5892	0.5155	0.0304	–0.4979	–1.0185	0.9649
–0.1229	0.1729	0.6868	–0.0606	–0.1314	–0.1577	0.2163	–0.0134	0.1175	–0.0362
2.0552	–0.2044	0.5121	–0.0165	1.0107	–0.0879	0.7137	–0.3962	0.5940	–0.6766
–0.3861	–0.6172	–0.3773	–0.3986	0.8655	–0.6729	–0.3130	–0.5479	0.5881	–0.1427

WEIGHT VECTOR : Γ	*INITIAL CONDITIONS*
–0.0019	–0.0008
0.3195	–0.0124
–0.5381	–0.0013
0.0735	–0.0040
–0.1360	0.0045
–0.3005	0.0028
0.1083	–0.0028
0.0669	–0.0050
–0.1354	0.0003
–0.3340	0.0025

Applying the definition of relative degree (2) to the trained network, it can be seen that the relative degree is $r = 2$ and this is the same relative degree of the plant, so the network is capable to identify the dynamics of the plant and its relative degree [12 - 15].

In order to verify the generalisation of the neural network model, different inputs were used. One such input, is a sine wave

$$u(t) = \frac{\pi}{2}\sin\left(\frac{2.\pi.t}{2.5}\right) + \frac{\pi}{2}\sin\left(\frac{2.\pi.t}{5.0}\right) \qquad (11)$$

The figure 5 shows the output of the plant and the output of the network for the input (11), the performance index was $J = 0{,}0084$.

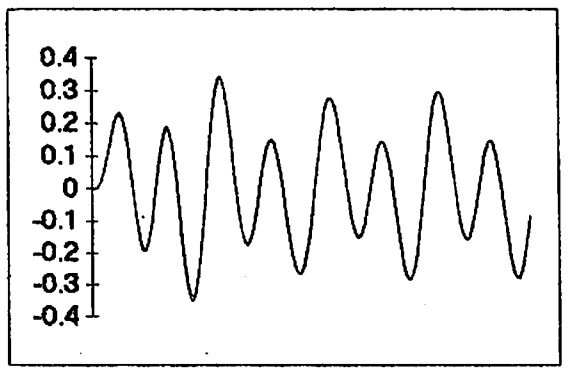


Fig. 5. Output of the plant (7) and output of the network (8) for the input (11).

5. Globally linearizing control . In this section some results obtained using the GLC structure are presented. The aim here is to ensure that the overall closed loop system matches some desired global dynamics. In this instance, a well damped second order dynamics of the form

$$\frac{y_d}{r} = \frac{1}{(s+1)^2}$$

was selected. The PI controller is

$$v = \left[k_p + \frac{k_i}{s} \right] e$$

where $k_p = 1.0$ and $k_i = 1.0$. To obtain the global desired dynamics with this PI controller, the linearized plant must obey the dynamics

$$\frac{y}{v} = \frac{1}{(s^2 + 3.s + 2)}$$

or the differential equation

$$\frac{d^2y}{dt^2} + 3.\frac{dy}{dt} + 2.y = v$$

The trained neural network (8) was used to calculate the linearizing state feedback

$$u = \frac{v - L_{\tilde{f}}^2 \tilde{h}(\chi) - \beta_1 . L_{\tilde{f}} \tilde{h}(\chi) - \beta_2 . \tilde{h}(\chi)}{L_{\tilde{g}} L_{\tilde{f}} \tilde{h}(\chi)} \qquad (12)$$

where $\beta_1 = 3$ and $\beta_2 = 2$.

The Fig. 6 shows the desired output and the plant output for the reference input $r = 0.5$, the performance index J=0.0197. The Fig. 7 shows the desired output and the plant output for the input (13), the performance index was $J = 0.0257$.

$$u(t) = \frac{\pi}{2} \sin\left(\frac{2.\pi t}{2.5} \right) \qquad (13)$$

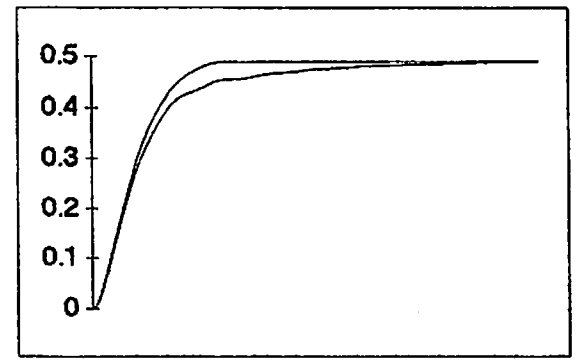


Fig. 6. Desired output y_d and plant output y in the GLC structure for a step reference $r = 0.5$.

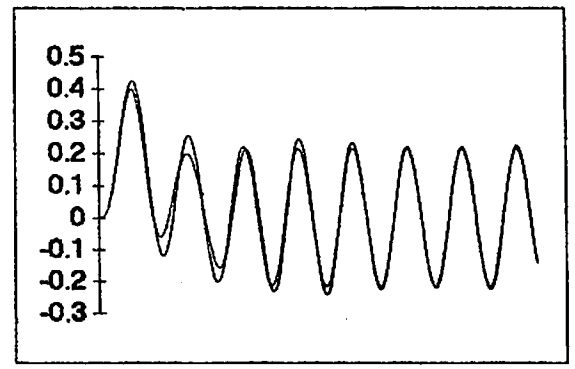


Fig. 7. Desired output y_d and plant output y in the GLC structure for the input (13).

The Fig. 8 shows the response for the step $r = 0.5$, when the mass of the single link manipulator was changed from 2.0 kg. to 2.5 kg, the performance index was $J = 0.0505$. The Fig. 9 presents the response for the step $r = 0.5$, when a disturbance of magnitude 0.1 was applied to the reference during the interval $\Delta t = [10.0, 12.0]s$, the performance index was $J = 0.0216$.

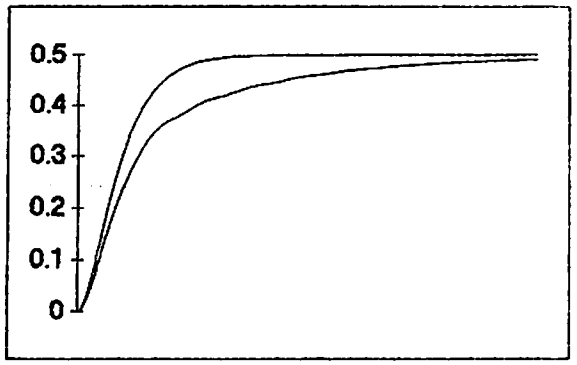


Fig. 8. Desired output y_d and plant output y in the GLC structure for a step reference $r = 0.5$. The mass of the plant was changed from 2.0 kg to 2.5 kg.

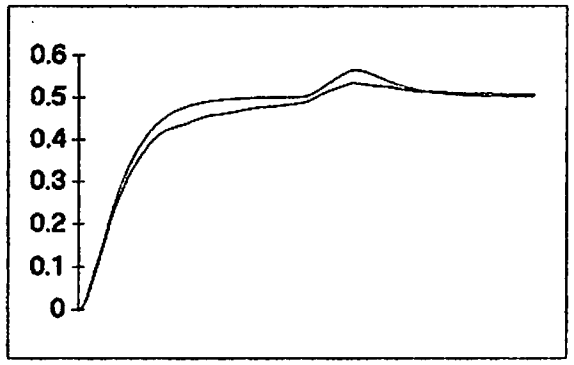


Fig. 9. Desired output y_d and plant output y in the GLC structure for step reference $r = 0.5$. A disturbance of magnitude 0.1 was introduced in the reference during the interval $\Delta t = [10.0, 12.0]s$.

6. Conclusions. It has been shown by way of simulations that the dynamic neural network (8) can identify the dynamics of non-linear systems of the class (1). Also this network can have any desired relative degree as was demonstrated in [12 - 14], during the identification the network adjusts itself to match the relative degree of the plant.

After the identification the neural network is utilised to calculate the linearizing state feedback to transform the non-linear plant into a linear system between v - y. The identified neural network is used in two ways : (a) To calculate the Lie derivatives of the linearizing feedback and (b) To generate, on-line, the state vector χ needed in the Lie derivatives.

The network was trained with a payload of 2.0 kg and the system was tested with a payload of 2.5 kg., it can be seen that the resulting response does not vary significantly. At the same time, it can be seen that inputs in the form of unmeasurable disturbances do not effect the performance. In fact with disturbance the index value is 0.0216 whilst without the disturbance it is 0.0197. These results show that the performance of the GLC structure is satisfactory and robust. It is important to mention that the neural network is only an approximation to the plant and that the linearizing state feedback is based on the exact cancellation of the non-linear terms $L_f^r h(x)$ and $L_g L_f^{r-1} h(x)$.

Acknowledgement. The author would like to thank to the Colombian Institute for the Development of Science and Technology (COLCIENCIAS) and to the National University of Colombia for supporting his PhD studies at the Cybernetics Department of the University of Reading.

REFERENCES

1. Cybenko, G.: *Approximation by superpositions of a sigmoidal function, Technical Report,* University of Illinois Urbana-Champaign, Department of Electrical and Computer Engineering, 1988.

2. Funahashi, K. I.: *On the approximate realization of continuous mappings by neural networks, Neural Networks*, 1989, 2, 183-192.

3. Narendra, K. S. and Parthasarathy, K.: *Identification and control of dynamical systems using neural networks*, IEEE Transactions on Neural Networks, 1990, 1, 4 - 26.

4. Chen, S. and Billings, S. A.: *Neural networks for non-linear dynamic system modelling and identification*. Int. J. Control, 1992, 56, 319 - 346.

5. Hopfield, J. J.: *Neurons with graded response have collective computational properties like those of two state neurons*, Proc. Nat. Acad. Sci. USA, 1984, 81, 3088-3092.

6. Isidori, A.: *Non-linear Control Systems*. Springer-Verlag, 1989.

7. Marino, R.: *Feedback linearization techniques in robotics and power systems*. In Fliess, M. and M. Hazewinkel, Editors, Algebraic and Geometric Methods in Non-linear Control Theory, D. Reidel Publishing Company, 1986, 523 - 543.

8. Marino, R. , Peresada, S. and Valigi, P. *Adaptive input - output linearizing control of induction motors*, IEEE Trans. on AC, 1993, 38, 208 -221.

9. Kravaris, C. and Chung, C. B.: *Non-linear state feedback synthesis by global input/output linearization*, AIChE Journal, 1987, 33, 592-603.

10. Jin, L., Nikiforuk, P. N. and Gupta, M.: *Direct adaptive output tracking control using multi-layered neural networks*. IEEE Proc. - D, 1993, 140, 393 - 398.

11. Bremermann, H. J. and Anderson, R. W.: *An alternative to Back Propagation : a simple rule of synaptic modification for neural net training and memory*. Report of the Center for Pure and Applied Mathematics, PAM - 483, University of California, Berkeley, 1990.

12. Delgado, A.: *Differential geometric control of non-linear systems with recurrent neural networks*. Internal Research Report of the Cybernetics Department, Reading University, 1994.

13. Delgado, A. and Kambhampati, C.: *A fully recurrent neural network for system identification and control*. To be published in IEE Proc. - D, 1995.

14. Kambhampati, C., Manchanda, S., Delgado, A., Green, G.G.R., Warwick, K. and Tham, M.T.: *On the relative order, inverses and the architecture of a class of recurrent networks*. Internal Research Report of the Cybernetics Department, Reading University, 1994. Submitted to Automatica.

EVOLUTIONARY COMPUTATION: EVOLUTION IN EVOLUTION

Max Garzón*
Institute for Intelligent Systems
Department of Mathematical Sciences
The University of Memphis
Memphis, TN 38152 U.S.A.

Abstract. Artificial neural nets are now well established alternatives to conventional computing and ordinary programming for their learning and adaptation capabilities. In a further effort toward the elimination of traditional programming, time-honored evolutionary paradigms such as natural selection and genetic recombination are currently being explored as productive alternatives in the search and design of computer programs and fault-tolerant systems. Connectionism presents then new problems, new challenges, and potential as useful tools in modeling, design and control of adaptative systems, intelligent autonomous agents and artificial life. Conversely, the availability of an explanatory mechanism to understand adaptive and cognitive processes has been used as evidence that Darwinian natural selection is perhaps not the most fundamental biological mechanism permitting, shaping, driving and limiting the fundamental property of adaptability in biological systems. This talk will discuss random neural networks as models on which to gauge the new problems, extent and some prospects of the synergy between biological mechanisms and artificial neural networks.

Resumen. Las redes neuronales artificiales son consideradas actualemnte como alternativas a la programación convencional gracias a su capacidad de aprendizaje y adaptación. Como un esfuerzo hacia la eliminación de la programación tradicional, hoy se exploran paradigmas evolucionarios tales como la selección natural y recombinación genética que permiten encontrar programas de computador y diseñar sistemas resistentes a faltas. conexionismo presenta así nuevos problemas, desafíos, y un nuevo potencial en el diseño y control de sistemas adaptativos, agentes artificiales autónomos y vida artificial en general. Viceversa, la disponibilidad de un mecanismo que permite entender este tipo de procesos adaptivos de la cognición ha sido utilizado como evidencia en la hipótesis que sostiene que el proceso de selección natural postulado por Darwin quizás no es el proceso biológico más fundamental que estructura, dirige, y a la vez limita el proceso básico de adaptación en sistemas biológicos naturales. La conferencia presenta el modelo guía de redes neuronales aleatorias con las cuales es posible hacer una estimación de nuevos problemas, el alcance y el potencial de la sinergía entre mecanismos biológicos aplicados a las redes neuronales artificiales.

Key Words: Neuronal nets, genetic algorithms, evolutionary programming, genome, natural selection, fitness landscape, random Boolean networks, control, universal neural network, neurocomputer system, information theory.

* garsonm@hermes.msci.memst.edu

1. Introduction. One of the stumbling problems with computers today is their programming. Despite historical advantages, programming is proving time and again to be the great advance and also the Achilles, heel of symbolic-based systems. Through over 40 years of symbolic-based artificial intelligence, a very delayed but certain fact has emerged: in order to create artificial systems that come anywhere close to resembling the capabilities of natural biological systems, a fundamental change in the basic philosophy and assumptions about the nature of computation will be required.

With this hindsight, it seems obvious that one should first imitate nature, the master creator of known prototypes of intelligent behavior. (This approach was attempted simultaneously with the emergence and growth of symbolic computation, but apparently we did not understand then enough about the power and limitations of the latter). Fundamental properties of natural systems are the capacity to adapt, learn, and survive in ever changing environments. The first two are some of the most attracting properties of artificial neural networks. Artificial neural nets are now well established alternatives to conventional computing and ordinary programming for their learning and adaptation capabilities. However, these properties also have turned out be very costly. One gets the impression that programming has been exchanged for learning and testing, and that the ability of the networks to adapt remain constrained by the type of architecture that the designers must choose and which remains fixed throughout the life of the network. In other words, adaptation and flexibility have not reached a reasonable degree that would make artificial systems autonomous or would give them any kind of survival ability in other than very artificial environments.

In a continued search toward the elimination of traditional programming, time-honored evolutionary paradigms, long ago arrived at in biology as basic mechanisms giving rise to the phenomenon of life (such as natural selection and genetic recombination), are currently being explored as productive alternatives in the search and design of computer programs and fault-tolerant systems. the purpose of this talk is to discuss random neural networks as models on which to gauge these new methods, their extent and potential, as well as some prospects for their synergy with biological mechanisms. Three main points are examined: first, what random neural nets tell us about natural selection; second, what can we learn by using evolutionary programming in the design of neural nets; and finally, we argue that a deep examination of the notion of *information* may be required for a true understanding of the profound changes that evolution is undergoing as we pass from "natural" evolution to evolution driven by humans minds. No formal definitions of the terms are given. The reader is referred to (Rumelhart-McClelland 1986), (Kauffman, 1993). or (Garzon, 1995a) for a systematic development of the concepts and arguments presented below.

2. Beyond NeoDarwinism. A fundamental question in human endeavours is the origin of *order* we encounter in a universe that generally appears multifaceted, so random and chaotic. Darwin's monumental work led him to propose his Theory Selection, which, roughly stated, ultimately places the source of order around us, living organisms in particular, in the accumulation of ever small incremental changes in this sea of randomness. These mutations are stored, so to speak, in surviving organisms that accumulate the advantages thanks to genome they produce when reproducing in their offspring. in a nutshell, no matter how extraordinary, living organisms are an accidental phenomenon that arise as an accumulation of lucky exceptions amidst the nemesis of order (randomness and chaos). To paraphrase the well known biologist (Monod, 1971), life is chance caught on a wing (Kauffman, 1993).

A century and a half later, many agree that this is rather far from a thorough *explanation* of the kind sought in evolutionary biology. Surely, there are other sources of order in nature. One of them, commonly described as self-organization, occurs everywhere, from crystals in physics to the very chemical reactions that allow, say, the reproduction of DNA in the interior of living cells. Starting from ideas of this sort, S. Kauffman set out to investigate whether natural selection is really the only driving force for order and self-organization. After years of computer simulations and analysis of the key regulatory processes in biological organisms, he has collected in (Kauffman, 1993) what appears to be very strong evidence for a surprising conclusion: that perhaps Darwin only got part of the truth. That order and self-organization are rather inherent properties of the dynamics of our universe, that in it complex systems can evolve to

"exhibit spontaneous order not because of selection, but despite it", that "some of the order in organisms may reflect not selection's successes, but its failure". (Themes, Kauffman, 1993).

Kauffman's basic concept is the idea of *fitness landscapes*. They are illustrated using random Boolean networks as models of genomes, whose nodes contain genes that interact according to the links established by the network. (A network can be seen as a digraph whose nodes hold certain bits of information, a gene say, and whose links determine the interaction between the various nodes.) In his *NK* model, each of *N* genes is influenced by *K* others, and the mutations and evolution are measured by using a fitness function that assigns to each network a fitness value. The evolution of a population of such genomes gives rise to a fitness landscape. The evolution of one organism is just an adaptive walk in such a landscape. In his view, the key to understanding the order and self-organization lies in the shape and evolution of fitness landscapes. And so he arrives at his fundamental hypothesis that I will try to state in a few sentences as follows. The entire universe ticks as a dynamical system, and in so doing it shapes the evolutionary fitness landscapes.

Now, evolution in every dynamical system eventually leads the system into its equilibrium attractors, located at the bottoms of its basins of attraction. These attractors are filled with organisms at the maximum stage of evolution, at maximum order and self-organization, and in equilibrium with the random fluctuations of natural selection, on the edge between order and chaos.

Several singular consequences follow from this picture. first, that life is a byproduct of self-organization in conglomerates of catalysts and protein of modest complexity, not a consequences of its particular historical developments. Second, that life, although arising in many different ways (not necessarily in the form of self-replicating DNA/RNA templates), reflects a more inherent order in the universe and, therefore *necessarily* would have a common lawlike statistical structure in all its manifestations. This is a heavy weight argument, for it implies, in particular, the virtually certain existence of (hitherto unknown to us) *laws and principles for life* of a similar nature to known principles for matter, energy, and other phenomena commonly occurring in physics and chemistry.

Third, that a complete picture for the understanding of life (natural and artificial) will require the integration of self-organization into a coherent picture together with Darwin's insightful discovery, natural selection , as the fundamental forces driving the evolution of complex systems and the emergency of order.

3. Evolutionary Methods for Artificial Neural Nets. On the basis of these philosophical assumptions, it is therefore quite natural to apply evolutionary methods in attempting to develop brain-like models, neural networks in particular. Research of this type requires a genome-like representation of neural nets, as well as the specification of a number of genetic operations and fitness functions that drive their evolution. Papers of this type are presented at current research meetings on evolutionary programming. In several works, (Gruau 1992, 1993, 1994) has presented a rather complete such system in which neural nets are *grown* from a single cell by a *generating procedure*, in contrast with the *descriptive* representation implicit in the weight matrix. The start cells are ACYC (an initial bare node) and/or CYC (a node with a recurrent link). Basic operators are PAR (parallel division, duplicates a cell, including its connections), CUT (cuts an input link), and END (makes a cell lose its reading head and become a neuron in the product net). These operators can be proved to be *complete*, in the sense that every neural architecture can be grown from a single cell by a suitable combination of their applications. Other properties such as modularity, universality, etc. can be obtained by adding a few more operators. Thus, every neural net can be represented by an expression describing the sequence of operations that grow it, which can be given by a binary tree that acts as a chromosome for the network. We can then apply a genetic algorithm to develop networks for specific functions. The method has been successful for a variety of tasks, including efficient networks for parity (a difficult task), adders, and various pattern recognition tasks.

More encompassing a applications of neural networks will have to deal with comprehensive *systems* capable of executing a wider variety of functions. if the dream of building anything resembling a mind based on artificial neural nets is ever going to come true in any form, neural net research must proceed to a second generation. The first modern outburst of

activity has mainly dealt with special purpose nets capable of perforing essentially transductions (in the form of feedforward nets) from some inputs to some outputs (which, granted, in some cases may become very complicated and impressive). A second generation will have to focus on the construction of systems in which many modules of special-purpose nets interact and feed back on one another to accomplish tasks in the service of a common higher-order function. A catalytic challenge in that direction might be the construction of a *computer system* entirely based on neural networks. This work is being attempted in several places and at several levels (software, VLSI, optical), as can be witnessed in Distante-et-al(1992), (Garzon-Franklin-et-al 1992, 1994). (Garzon, Franklin, Caulfield, 1993). The basic architecture of a general-purpose neurocomputer is *universal* in the sense that it is capable of running (on a fixed architecture) arbitrary neural networks fed as sort of programs to the network. Mange-at-al (1994) have managed to design neural multicellular nets (as opposed to one cell architectures, such as von Neumann universal cellular automaton computer (Burks, 1970) capable of self-repair against failures.

It is becoming increasingly clear, however, that a system of such complexity will leave very little room for design, in the sense that computer engineers are used to. Even in silicon, work such as (de Garis, 1994) show that and evolutionary approach may be more promising in term of how complex a system one might be able to grow. the basic dilemma is that design guarantees a degree of functionality of the finished product, but makes actual implementation very difficult. On the other hand, evolving the connections makes the implementation relatively easy, but it does not guarantee that certain minimal functionalities will be there in a single individual (perhaps in a population), or that the behavior of the newly grown networks will be necessarily consistent with humans goals and interests.

4. A Science of Information and Artificial Congnition. Upon reflection on these themes, one cannot help but think whether a better approach would be to study the fundamental concept that underlies the new age into which we are developing. Words such as *Knowledge* and *information* seem to capture well the spirit of our times. Naturally, some rigorous

study of these concepts should shed light on what's happening. perhaps unsurprisingly, little has been done in this direction.

To be sure, we have Shannon's theory of information (some would rather call it theory of *data communication*). Shannon defines information as a discrete spectrum of possibilities (e.g., color) and concentrates on the idea of *communicating* information. Communication takes place by selecting the right choice (e.g., red) and thereby discarding the remaining possibilities. His theory then goes on to establish a number of basic results on the existence of efficient ways to encode information into data, of designing communication codes to transmit the data in the presence of noise with arbitrarily small errors, of measuring the uncertainty of a given message, and so on. Nowhere is there (and perhaps there wasn't meant to be) s definition of just *what is information*, or an effort to unveil its fundamental properties. Information is taken to exist in some form, modifiable into data as symbolic strings over some alphabet (e.g., binary), transmitted over a communication channel, and finally, retrieved at the other end, presumably by another agent who can use the data to retrieve the original information. Implicitly, Shannon's is a theory of an anthropomorphic nature, in which the role of information processors is implicitly attributed to humans only, and about the origin of which there is no need for further inquiry.

It is only very recently that information per se has become the object of rigorous investigation. I am aware of only two beginnings for a rigorous science of information. First, there is information physics, as developed, for example, in the account of (Stonier 1990). Here information is assumed to have an objective physical existence, pretty much in the ranks of energy and matter in physics. Second, there is *situation theory*, as put forth by (Devlin, 1991). He assumes a cognitive agent capable of certain minimal cognitive abilities (such as *discrimination* and/or, at a higher level, *individuation* of animal unit of information) for *infons*. A *situation* is a very specific world state (in principle containing an infinite amount of infons) and certain cognitive relations that the agent produces to explicitate infons from the situation. Although Devlin sees himself constrained to information communication to begin with, his theory puts himself in the middle of vexing questions, such as how to define information, how information can be *created* by cognitive agents (which must be explained

away, lest one has a homuncular theory) in the process of cognizing the world, and how it is necessary for cognitive agents to have certain minimal capacities to correlate states in the world with states in their minds in the process. It is likely that artificial neural nets would be able to play a role in shedding light on these yet unanswered questions. (A further discussion and comparative summary account can be found in (Garzon, 1995b).) It is clear, that, however underdeveloped these theories may be, they illustrate poignantly how little we know about information, the stuff which is the heart and substance of the information era.

5. Conclusion. Connectionism does present new problems, new challenges, and potential as useful tools in modeling, design and control of adaptative systems, intelligent autonomous agents and artificial life. Conversely, the availability of an explanatory mechanism to understand adaptive and cognitive processes has been used as evidence that Darwinian natural selection is perhaps not the most fundamental biological mechanism enabling, shaping, driving and limiting the fundamental property of adaptability in biological systems. As we strive to apply biological mechanisms to the development of artificial neural nets, a further examination and elaboration of these mechanisms is likely to feed back into our understanding of their nature, even perhaps to help uncover a more accurate explanation of the origin of life and order in our universe. A science of information that attempts to develop an understanding of the fundamental concept that seems to underlie the entire field has not so far received significant attention. As a consequence, in the words of (Devlin, 1991), it feels that we understand the concept of information today perhaps as little as a man of the pre-historic iron age understood the molecular structure of iron.

REFERENCES

A. Burks, de. (1970), *Essays on Cellular Automata*. U. of Illinois Press, Urbana IL.

K. Devlin (1991), *Logic and information*. Cambridge University Press, Cambridge, England.

F. Distante, M. Sami, R. Stefanelli, and G. Storti-Gajani (1991), Mapping Neural Nets onto a Massively Parallel Architecture. A Defect-Tolerance Solution. *Proc. IEEE* **79**:4, 444-460.

M. Garzon, S. Franklin, W. Boyd, W. Bagget, D. Dickerson (1992), Design and Testing of a General-Purpose Neurocomputer. *J. Par. Distrib. Computing* **14**, 203-220.

M. Garzon, S. Franklin, D. Mundie (1994), VLSI Implementation of a General-Purpose Neurocomputer. *Proc. IMACS World Congress on neural networks*, Atlanta, 392-398.

M. Garzon (1995a), *Models of Massive Parallelism (Analysis of Cellular Automata and Neural Networks)*. Springer- Verlag, Berlin.

M. Garzon (1995b), Symbolic logic in a new AI: new challenges, new perspectives. Proc. X Latin American School of Mathematical Logic. Bogotá, Colombia, July 1995.

F. Gruau (1992), Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process.

F. Gruau (1993), Cellular Encoding of Genetic Neural Networks. *Proc. 5th Int. Conf. on Genetic Algorithms.*

F. Gruau (1994), Synthèse de réseau de neurones par codage cellulaire et algorithmes génétiques. PhD Dissertation, École Normale Superiéure de Lyon.

S.A. Kauffman (1993), *The Origins of Order (Self-organization and selection in Evolution)*. Oxford University Press, Oxford.

D. Mange, S. Durand, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, C. Piguet (1995), A New Paradigm for Developing Digital Systems Based on a Multicellular Organization. *Proc ISCAS Int. Symp. on Circuits and Systems*, Seattle WA.

J. Monod (1971), J. *Chance and Necessity*. Translated by A. Weinhouse, Knopf, New York NY.

D.E. Rumelhart, J.L. McClelland (eds.) (1996), *Parallel Distributed Processing*. The MIT Press, Cambridge MA, Vol. 1.

T. Stonier (1990), *Information and the Internal Structure of the Universe*. Springer-Verlag, London UK.

SISTEMAS COMPLEJOS ADAPTATIVOS

Germán J. Hernández *
Luis Fernando Niño **
Universidad Nacional de Colombia, Santafé de Bogotá
Departamento de Ingeniería de Sistemas

Abstract. We are developing a theory of adaptive complex systems based on automata networks and on the theory of adaptive systems. Automata networks are formal models of deterministic, nondeterministic, stochastic and fuzzy complex systems In this work Adaptive Automata Networks (AAN's) based on adaptive plans are presented as formal models of adaptive complex systems. Based in Evolutive Automata Networks (EAN's), the properties of completeness, compactness, closure of representation, closure of operators and scalability are defined for adaptive complex systems.

Resumen. Estamos desarrollando una teoría de sistemas complejos adaptativos basada en redes de autómatas y en la teoría de sistemas adaptativos. Las redes de autómatas son modelos formales de sistemas complejos deterministicos, no determinísticos, estocásticos y difusos. En este trabajo las *redes de autómatas adaptativas* (AAN), basadas en planes adaptativos, se presentan como modelos formales de sistemas complejos adaptativos. Basados en la *redes adaptativas evolutivas* (EAN), se definen las propiedades de completez, compacidad, clausura de representación, clausura de operadores y escalabilidad para sistemas complejos adaptativos.

Key Words: *Adaptive complex systems, automata networks, adaptive plan*

1. Introducción. En este trabajo se desarrolla una teoría para sistemas complejos adaptativos y se establecen algunas de sus propiedades. Se construye un modelo formal para sistemas complejos adaptativos basado en redes de autómatas [6;14] y en la teoría de los sistemas adaptativos [8]. En [14] las redes de autómatas son presentadas como modelos formales de sistemas complejos determinísticos. En [7] se extiende la definición para modelar sistemas complejos no determinísticos, estocásticos y difusos; además, se dotan de un mecanismo evolutivo, el cual se plantea como una metodología para el diseño de sistemas complejos artificiales autocontrolados.

* e-mail: ghernand@bacata.usc.unal.edu.co
** e-mail: lfnino@unisis.campus.unal.edu.co

En este trabajo, siguiendo el esquema presentado en [7], se definen las *redes de autómatas adaptativas* (RAA) como redes de autómatas sujetas a planes adaptativos. Se define el esquema de representación para redes de autómatas y se presentan los operadores que conforman los planes adaptativos en RAA. A partir del esquema representación y operadores se definen las propiedades de completez, compacidad, clausura, modularidad y escalabilidad para RAA asociados a sistemas complejos adaptativos.

2. Redes de autómatas. Un *autómata* es una máquina que asume un estado de un conjunto de estados S, recibe como entrada un estímulo y produce como salida una respuesta. El conjunto de estímulos de entrada se nota Σ y el conjunto de respuestas se nota Π. La evolución del autómata está gobernada por una dinámica de transición de estados δ, la cual asigna un nuevo estado al autómata dependiendo del estado en que se encuentre y del estímulo que esté recibiendo; el autómata recibe secuencialmente los estímulos y va cambiando su estado de acuerdo a δ. La salida está determinada por una aplicación [1] de salida ψ.

Formalmente, un autómata A es una quíntupla

$$A = (S, \Sigma, \Pi, \delta, \psi)$$

donde

(i) S es el conjunto de estados del autómata;
(ii) Σ es el conjunto de estímulos de entrada;
(iii) Π es el conjunto de respuestas del autómata;
(iv) $\delta : \Sigma \times S \to S$ es una aplicación parcial o total llamada la *dinámica de transición de estados*;
(v) $\delta : \Sigma \times S \to \Pi$ es una aplicación parcial o total llamada *aplicación de salida.*

Un autómata se dice

- *finito* si los conjuntos S, Σ y Π son finitos;

[1] relación o función

- *determinístico* si δ o ψ son ambas funciones parciales o totales;
- *no determinístico* si δ o si ψ es una relación;
- *estocástico*, en [10] es llamado autómata de aprendizaje, si δ o ψ es una aplicación estocástica;
- *difuso* si los conjuntos S, Σ y Π, y las aplicaciones δ y ψ son difusas [9].
- con estado inicial si el autómata comienza en estado s_0, con $s_0 \in S$.

Una *red de autómatas* [2,7] es un conjunto de autómatas finitos interconectados. Un autómata envía, únicamente, su estado a los demás autómatas que están conectados con él; es decir, la aplicación de salida sólo depende del estado y está definida como $\psi(e,s) = s$, para $e \in \Sigma$ y $s \in S$.

Para formalizar la noción de red de autómatas es necesario definir como es cada uno de los autómatas que la conforman y como están conectados entre si. Esto es, se debe definir un cierto conjunto $\{A_i : i \in I\}$, donde A_i es el i-ésimo autómata en la red, y un digrafo [2] $D = (I, \Gamma)$ que especifica las conexiones entre los autómatas. En ocasiones se supone que el conjunto de estados es el mismo para todos los autómatas; no se pierde generalidad porque si se tienen autómatas con estados diferentes se puede considerar S^* como la unión de todos los conjuntos de estados de los autómatas en la red y redefinir cada autómata cambiando su conjunto de estados por S^*.

Formalmente, una *red de autómatas* (RA) R es una dupla

$$R = (D, \{A_i\}_{i \in I})$$

donde

- $D = (I, \Gamma)$ es un digrafo que tiene como vértices al conjunto de índices I de los autómatas, y

[2] Un digrafo $D = (X, \Phi)$, es un par donde X es un conjunto y $\Phi \subseteq X \times X$ es una relación definida en X, en donde cada pareja $(a,b) \in \Phi$ se puede ver como una conexión que parte a y llega a b.

- $\{A_i = (S, \Sigma_i, \delta_i), i \in I\}$ es un conjunto de autómatas finitos - como los autómatas tienen el estado como salida, se omiten la aplicación y el alfabeto de salida. El alfabeto Σ_i es $S^{\Gamma^+(i)+1}$ con $\Gamma^+(i)$ el número de parejas en Γ con segunda componente i; es decir, lo que recibe el i-ésimo autómata como entrada es su propio estado y el de todos aquellos a los que está conectado y que le envían su estado como salida.

Las redes de autómatas son una generalización de los autómatas celulares [3,11,12] y de las redes neuronales [13,9].

Un *sistema complejo* (SC) [14] es un sistema compuesto de un gran número de elementos, en principio diferentes, interactuando. Para modelar un sistema complejo mediante una red de autómatas, cada elemento del sistema se modela como un autómata y las interacciones entre los elementos como conexiones entre los autómatas.

3. Sistemas adaptativos

3.1. Adaptación. La palabra "adaptación" toma diferentes significados en muchas de las disciplinas en las cuales es encontrada. En ingeniería, frecuentemente, se consideran equivalentes aprendizaje y adaptación, mientras que en sicología se hace una fuerte distinción entre ellas. La diferencia depende de qué tan amplia sea la definición de aprendizaje que se esté dispuesto a aceptar. En este trabajo consideramos el aprendizaje como una forma de adaptación que se realiza en función del tiempo.

Se considera que un sistema S (máquina, organismo o especie) es un *sistema adaptativo* si su comportamiento en un ambiente cambiante es "exitoso" en algún sentido. Si existe alguna medida del éxito entonces se tienen dos propiedades que distinguen un sistema adaptativo de un sistema no adaptativo: estabilidad y robustez.

La *estabilidad* se refiere a que los sistemas adaptativos se mantienen estables en una región de respuesta exitosa, a pesar de los cambios en el ambiente. Se puede definir estabilidad como *persistencia del éxito en un ambiente cambiante.*

La *robustez*, también llamada tolerancia a fallas o autoreparación, se refiere a que en los sistemas adaptativos si una parte del sistema es dañada o falla, el sistema enmascara gradualmente el efecto del daño hasta que el sistema alcanza un nivel de eficiencia aceptable. Se puede definir robustez como *persistencia del éxito independiente de las fallas en las partes.*

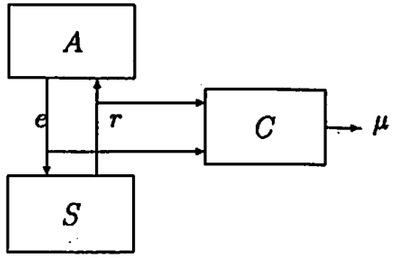


Figura 1. Configuración SAC

Un esquema funcional de un sistema adaptativo llamado *configuración SAC* (sistema-ambiente-crítico) está dado en la figura 1. El *estímulo* e es la entrada proveniente del *ambiente* A al sistema S, y la *respuesta* r es la salida del sistema. La medida del éxito, llamada "medida de efectividad" μ, está dada por la salida del *crítico* C. Entonces

- S es *estable* si μ permanece dentro de los valores prescritos cuando A cambia, y
- S es *robusto* si μ permanece dentro de los valores prescritos cuando S cambia.

En la anterior definición ninguno de los requerimientos del sistema adaptativo depende del tiempo.

Por otro lado, las definiciones de aprendizaje y autoreparación están relacionadas con el tiempo, entonces suponemos que la efectividad es una función del tiempo $\mu(t)$. Se dirá que S *aprende* en un intervalo de tiempo específico de tiempo $[0,T]$ si para un cambio en el ambiente A en

$t = 0$ se tiene $\mu(T) > \mu(0^+)$. Se dice que S se *autorepara* si para una falla en $t = 0$ se tiene $\mu(T) > \mu(0^+)$. Si se coloca un mínimo nivel de eficiencia alcanzable L_m, se tienen las definiciones con $\mu(T) > L_m > \mu(0^+)$.

3.2. Sistema Adaptativo. Un sistema adaptativo [14] es un sistema que modifica su estructura para mantener o mejorar la eficiencia en un ambiente cambiante. Para dar su definición se tiene:

- A el ambiente (cambiante) en el cual se encuentra el sistema,
- μ una medida de eficiencia de las diferentes estructuras del sistema, y
- τ un plan adaptativo, el cual determina las modificaciones estructurales sucesivas en el sistema en respuesta a cambios en el ambiente.

Formalmente, un *sistema adaptativo* SA es una cuádrupla

$$SA = (\mathscr{E}, \Omega, I, \tau)$$

donde

- $\mathscr{E} = \{E_k\}_{k \in K}$ es el conjunto de estructuras alcanzables, es decir, el dominio del plan adaptativo;
- $\Omega = \{w_j\}_{j \in J}$ es el conjunto de operadores que modifican las estructuras con $\omega_j : \mathscr{E} \to \mathscr{P}$, donde $\mathscr{P}$ es un conjunto de distribuciones de probabilidad sobre $\mathscr{E}$;
- I es el conjunto de posibles entradas al sistema desde el ambiente; y
- $\tau : I \times \mathscr{E} \to \Omega$ es el plan adaptativo,

con

$$\tau(I(t)), \mathscr{E}(t)) = \omega_t \in \Omega \text{ y } \omega_t(\mathscr{E}(t)) = \mathscr{P}(t+1),$$

donde $\mathscr{P}(t+1)$ es una distribución particular sobre $\mathscr{E}$ y $\mathscr{E}(t+1)$ se obtiene como una selección aleatoria en $\mathscr{A}$, de acuerdo a la distribución $\mathscr{P}(t+1)$.

4. Redes adaptativas de autómatas. Una *red adaptativa de autómatas* (RAA) es un sistema adaptativo que tiene como conjunto de estructuras admisibles un conjunto de redes de autómatas, que llamaremos *redes admisibles de autómatas.* Como ya se dijo, un sistema complejo tiene como modelo una red de autómatas; a continuación se muestra que un sistema complejo adaptativo (SCA) tiene como modelo una RAA.

Un SCA S tiene asociado un conjunto de estructuras admisibles $\mathscr{E}$. Cada estructura admisible para S es un sistema complejo S_k. Como todo sistema complejo se puede representar por una red de autómatas; en particular, cada S_k se puede representar con una red autómatas R_k; de este modo, el conjunto de estructuras admisibles se puede representar como

$$\mathscr{E} = \{R_k\}_{k \in K}$$

donde $R_k = (D_k, \{A_{k_i}\}_{i \in I_k})$ es la red de autómatas que modela a S_k. Asociamos como modelo para S la RAA que tiene como redes admisibles de autómatas las R_k, y se denota con R_S. Los operadores, el conjunto de estímulos y el plan adaptativo para R_S se definen a partir de los correspondientes en S, como se muestra más adelante.

4.1. Representación de RAA. Para representar una RAA se debe representar el conjunto de redes admisibles de autómatas; para esto se debe determinar un esquema de representación para cada RA en el conjunto. Las redes de autómatas de una RAA, que es modelo de un SCA, comparten un conjunto de características que les permiten ser modelos de estructuras admisibles para el SCA.

Para definir un esquema de representación de una RA se debe determinar como se representan el digrafo de conexiones y el conjunto de autómatas. Existen varias formas equivalentes de representar un digrafo, son equivalentes en su capacidad de representación pero algunas tienen ventajas sobre otras desde el punto de vista computacional [1].

Se selecciona la *matriz de adyacencia* para representar el digrafo de conexiones de una RA. Dado un digrafo $D = (I, \Gamma)$ de una RA con $|I|$

el número de elementos en I. La *matriz de adyacencia* de D, notada $M_D = (m_{ij})$, es una matriz de tamaño $I \times I$ definida como

$$\begin{cases} 1 & \text{si} \quad (i,j) \in \Gamma \\ 0 & \text{si} \quad (i,j) \notin \Gamma \end{cases}, \text{ con } i,j \in I$$

En esta matriz, si m_{ij} es 1, el vértice i y el vértice j están conectados por medio el arco (i,j).

Para representar un autómata en la red se tienen también varias formas equivalentes; para nuestro esquema de representación utilizamos la *matriz de transiciones*. Dado un autómata de un RA, $A = (S, \Omega, \delta)$, la *matriz de transiciones* de A, notada $T = (t_{q,\sigma})$, es una matriz de tamaño $|\Omega| \times |S|$, con

$$t_{q\sigma} = \delta(q,\sigma) \text{ donde } q \in Q, \sigma \in \Sigma.$$

Una red de autómatas, $R = (D, \{A_i\}_{i \in I})$ se representa como el conjunto I, la matriz de adyacencia de D, el conjunto de estados para los autómatas en R y el conjunto de matrices de transición para los A_i en R.

$$R \equiv \langle I, M_D, S, \{M_{A_i}\}_{i \in I} \rangle.$$

El esquema de representación para RAA es la k-upla de esquemas de representación para sus redes admisibles de autómatas.

4.2. Operadores sobre RA. Los operadores modifican la estructura de un sistema adaptativo, en particular, en una RAA modifican una red admisible de autómatas transformándola en otra red admisible de autómatas.

Para establecer el tipo de operadores que se pueden definir para una RAA se hace primero el análisis de las modificaciones que se pueden introducir en una RA, las cuales son:

- Modificar el conjunto de índices:

Las operaciones básicas sobre el conjunto de índices son agregar y eliminar índices. Esto implica agregar y quitar los autómatas correspondientes además de agregar y quitar sus conexiones.

- Modificar el conjunto de conexiones:
  Las operaciones básicas sobre el conjunto de conexiones son agregar y quitar conexiones.
- Modificar los autómatas:
  Para modificar un autómata se puede cambiar el conjunto de estados, el cual es igual para todos los autómatas en la red; por otro lado, se puede cambiar la dinámica de transición de estados de cada autómata. No se considera agregar y quitar autómatas porque estas modificaciones están asociadas a modificaciones en el conjunto de índices.

4.3. Operadores para RAA. Dada una RAA con conjunto de redes de autómatas admisibles

$$\left\{R_k = (D_k, \{A_{k_i}\}_{i \in I_k})\right\}_{k \in K}$$

donde $D_k = (I, \Gamma_k)$ se definen así:

(i) La familia de conjuntos admisibles de índices como

$$\mathcal{I} = \{I_k\}_{k \in K},$$

y el conjunto maximal de índices como

$$\hat{I} = \bigcup \mathcal{I} = \bigcup_{k \in K} I_k.$$

(ii) La familia de conjuntos admisibles de conexiones como

$$\mathcal{G} = \{\Gamma_k\}_{k \in K}$$

y se define el conjunto maximal de conexiones

$$\hat{\Gamma} = \bigcup \mathcal{G} = \bigcup_{k \in K} \Gamma_k = \hat{I} \times \hat{I}.$$

(iii) El conjunto de autómatas admisibles en el vértice i de I como

$$\mathcal{A}_i = \{A_{k_i}\}_{k \in K}.$$

A continuación se definen los tipos de operadores para una RAA:

- Operadores tipo índice:
  Son operaciones unitarias [3] sobre $\mathscr{I}$ y se pueden construir a partir de agregar y eliminar índices en los conjuntos admisibles de índices.
- Operadores tipo conexión:
  Son operaciones unitarias sobre $\mathscr{F}$ y se pueden construir a partir de agregar y eliminar conexiones en los conjuntos admisibles de conexiones.
- Operadores tipo autómata:
  Son operaciones unitarias sobre $\{\mathscr{A}\}_i$. En general, realizan una modificación de la dinámica de transición del autómata. Sólo bajo condiciones muy especiales se modifica el conjunto de estados, ya que esto implica, en general, una modificación global de todos los autómatas.

4.4. Propiedades. En [4] se definen propiedades para un esquema de representación y un conjunto de operadores genéticos sobre autómatas celulares y redes neuronales. Estas propiedades se definen aquí para el esquema de representación de RAA y sus operadores.

Dada una RAA $\mathscr{R}$ tenemos:

(i) Completez
Un esquema de representación es *completo* para $\mathscr{R}$ si cualquier red de autómatas admisible es representable.

(ii) Compacidad
Un esquema de representación $\mathscr{E}_1$ es topológicamente más *compacto* para $\mathscr{R}$ que un esquema de representación $\mathscr{E}_2$, si cada red admisible de autómatas para $\mathscr{R}$ que se representa computacionalmente con un código de tamaño s en $\mathscr{E}_2$ puede ser representada con un código de tamaño $O(s)$ en $\mathscr{E}_1$.

[3] Una operación n-naria α sobre un conjunto X es una función de X^n en X, $\alpha : X^n \to X$. Una operación unitaria es simplemente una función de X en X.

(iii) Escalabilidad
El esquema de representación es *escalable* en $\mathscr{R}$ si fijando un tamaño en la representación, con esquemas de tamaño fijo, se representa una familia de redes admisibles de autómatas para $\mathscr{R}$.

(iv) Clausura de representación
El esquema de representación es *cerrado en* $\mathscr{R}$ si toda representación es una red admisible de autómatas.

(v) Clausura de operadores
Un conjunto de operadores sobre $\mathscr{R}$ es *cerrado* si al aplicar cualquiera de ellos él produce una red admisible de autómatas. En particular, si los operadores se definen como se estableció en la sección anterior entonces son cerrados.

BIBLIOGRAFÍA

1. Aho A. V., Hopcroft J. E. and Ullman J. D. (1974), **The Design and Analysis of Computer Algorithms**, Addisson-Wesley.

2. Garzón M. (1990), *Cellular Automata and Discrete Neural Networks*, Physica D **45**, 431-440.

3. Garzón M. (1995), Models of Masive Parallelism: Analysis of Cellular Automata and Neural Networks, Springer--Verlag.

4. Garzón M. (1996), *Evolutionary Computation: Evolution in Evolution*.En *Memorias del Primer Congreso Nacional de Neurocomputación*. Academia Colombiana de Ciencias Exactas, Físicas y Naturales, 43-51.

5. Goldberg D. E. (1989), *Genetic Algorithms: In Search, Optimization, and Machine Learning*, Addisson-Wesley.

6. Goles E. (1990), *Automata Networks*, Birkhäuser.

7. Hernández G. y Niño L. F. (1994), *Teoria de control en sistemas complejos*, Memorias 1er. Congreso Nacional de Automática. Cali

8. Holland J. H. (1992), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press.

9. Kosko B. (1992), *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach*, Prentice-Hall.

10. Narendra K. and Thathachar M.A.L. (1989), *Learning Automata : An Introduction*, Prentice--Hall.

11. Toffoli T. and Margolus N. (1986), *Cellular Automata Machines*, MIT Press.

12. Torres L. G., Hernández G. y Niño L. F.(1992), *Autómatas Celulares*. Libro de cursillo, IX Coloquio Distrital de Matemáticas y Estadística, Universidad Nacional de Colombia, Santafé de Bogotá.

13. Torres L. G., Hernández G. y Niño L. F.(1993), *Redes Neuronales*. Libro de cursillo, Coloquio Distrital de Matemáticas y Estadística, Universidad Nacional de Colombia, Santafé de Bogotá.

14. Weisbuch G. (1991), *Complex Systems Dynamics*. Lecture Notes Volume II. Santa Fe Institute in the Complexity of Sciences. Addisson-Wesley

LAS PALABRAS EN EL ESPACIO DE CANTOR

Rafael F. Isaacs G.
Universidad Industrial de Santander
Universidad Nacional de Colombia

Abstract . The code space introduced by Barnsley [1988] is used to illustrate how some compact topological spaces are continua image of Cantor Space. This idea is used to build Peano Curves.

Resumen. Utilizando el espacio de los códigos presentado por Barnsley [1988] se ilustra cómo algunos espacios se pueden ver como imagen continua del espacio de Cantor. Se utilizan estas ideas para construir curvas de Peano.

Nadie nos podrá expulsar del paraíso
que creó Cantor para nosotros...
David Hilbert.

Seguramente el paraíso al que se refiere Hilbert es la aritmética transfinita, creación maestra (¿o descubrimiento?) de Cantor que ojalá en un futuro no sea del conocimiento exclusivo de los matemáticos. En toda persona, así como en la humanidad, una vez formado el concepto de número, cuando ya se sabe contar indefinidamente, se adquiere el concepto de infinito, en el sentido de que puede hablar del infinito aunque no pueda precisar de qué se trata. Así la comunidad matemática durante siglos maneja este concepto primitivo que no se formaliza hasta los estudios de Georg Cantor a finales del siglo XIX. Según ese concepto primitivo de infinito, este ``elemento" extraño e inalcanzable, es uno sólo, único. Cantor descubre que además de ser un concepto trabajable formalmente, el infinito no es único, sino que se deben considerar diferentes infinitos: que, aunque los números enteros y los pares comparten el mismo tipo de infinito que el mundo de las fracciones (números racionales), éste es un infinito definitivamente menor que el de los puntos de un segmento de recta. Y así, son muchísimos los infinitos que aparecen en este paraíso...La cuestión es tan llamativa y maravillosa que ese mismo paraíso, que por momentos debía necesariamente aparecerse como apenas una ilusión mental, junto con el casi nulo reconocimiento y las difíciles circunstancias que rodearon su práctica profesional, fue causante de la en-

fermedad mental que acompañó a Cantor hasta la muerte. Además de creador de paraísos, la vida de este gran pensador tiene características de redentor.

Pero en este trabajo, no nos queremos referir a este apasionante tema de la aritmética transfinita, sino a otra creación de Cantor de gran trascendencia en el desarrollo de la topología durante el actual siglo: el hoy en día conocido como espacio de Cantor.

Universalidad del espacio de Cantor. En medio de sus investigaciones alrededor del continuo (el infinito correspondiente a un segmento de recta), Cantor propuso considerar el siguiente conjunto definido de manera recursiva: al segmento de recta [0,1] se le quita la tercera parte intermedia, es decir, el intervalo abierto (1/3,2/3), obteniendo dos intervalos cerrados; a estos dos tercios que quedan, se les quita sus tercios intermedios (novenos del inicial) y así sucesivamente (ver la gráfica 1).

Gráfica 1. Construcción del conjunto de Cantor.

El conjunto límite, que hoy en día llamamos conjunto de Cantor (a veces también conjunto ternario de Cantor), tiene la característica de contener tantos puntos como un segmento de recta (muchos más de lo que uno en principio se imagina) pero, sin embargo, no contener ningún segmento de recta (por esto no se "ve").

Importante porque la engañosa intuición lo hace pensar a uno que un conjunto con cardinal ("número de elementos") mayor o igual que el continuo, necesariamente contiene algún intervalo de recta. Esta observación fue suficiente para que Cantor resaltara su conjunto por su carácter paradójico. Sin embargo poco después de su muerte, los fundadores de la topología descubrieron importantes propiedades topológicas del

de la topología descubrieron importantes propiedades topológicas del conjunto de Cantor, que le otorgan sobrados méritos para que sea otro objeto importante del "paraíso que creó Cantor para nosotros". A estas propiedades, en especial a dos teoremas clásicos de la topología general, queremos referirnos con la intención de ilustrar estos resultados y su trascendencia y sin interesarnos en su demostración, por cierto bastante abstracta. Estos dos resultados podrían expresarse en tono bíblico así:

Génesis. *En un principio el espacio era de Cantor: totalmente disconexo aunque compacto y perfecto (sin puntos aislados). Entonces llegó Dios y pegó de infinitas maneras, así aparecieron todos los espacios métricos compactos...*

La primera propiedad es una caracterización topológica que nos permite referirnos a un objeto topológico más general: el espacio de Cantor. El otro teorema , debido a Alexandrov y Urysohn [1929], da al espacio de Cantor un papel predominante en la topología. En términos matemáticos dice que todo espacio métrico compacto es imagen continua del espacio de Cantor. Esto significa de manera intuitiva que los espacios métricos compactos se obtienen del espacio de Cantor pegando puntos. Nuestro objetivo en este trabajo es mostrar cómo se harían estos pegamientos para ciertos casos clásicos y menos clásicos, y también aplicar estas ideas para construir curvas de Peano.

El teorema caracterizador del espacio de Cantor es debido a Hausdorff y asegura que todo espacio métrico que sea totalmente disconexo, compacto y perfecto es homeomorfo al conjunto de Cantor. Ser homeomorfo significa ser igual topológicamente, entonces por definición, el único espacio con estas propiedades es el que llamamos espacio de Cantor. Pero espacios métricos con estas tres propiedades se encuentran de muchas maneras en múltiples ambientes. El conjunto de Cantor, el conjunto construido con puntos de la recta real entre 0 y 1, es sólo una representación del espacio de Cantor. Pero hubiéramos podido partir de cualquier otro intervalo y obtenemos el mismo espacio. Esto intuitiva y analíticamente es inmediato, pero el asombro se inicia al saber que si partimos de cualquier intervalo y partimos por ejemplo en cuatro partes de las cuales escogemos dos y así sucesivamente, el conjunto límite, que

inicialmente construimos. Pero no se crea que todas estas representaciones del espacio de Cantor deben suceder en la recta real. Se pueden encontrar interesantes representaciones en el plano y el espacio euclídeos, así mismo hacer varias construcciones partiendo de entidades matemáticas de naturaleza aparentemente extraña. Por ejemplo, se puede caracterizar el espacio de Cantor como cierto espacio ordenado (ver Oostra [1993]) o como determinada álgebra de Boole (ver Acosta [1988]).

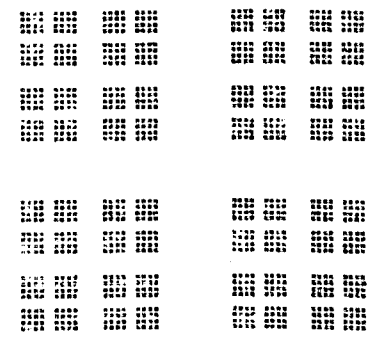
Mandelbrot [1987] introduce tardíamente en su discurso el conjunto de Cantor, teniendo en cuenta la repulsión que éste causa entre personas interesadas por aplicaciones de la matemática. El siguiente ejemplo, apoyándose en los variados ambientes en que se presenta dicho espacio, proporciona una representación en una atmósfera mucho más concreta.

Los autómatas celulares son modelos teóricos que representan cierto tipo de la computación en paralelo. Pues bien, su efecto se puede ver como el de una función continua sobre un espacio que es homeomorfo al conjunto de Cantor. En otras palabras, los autómatas celulares son cierto tipo (bastante amplio) de funciones continuas entre el espacio de Cantor (ver Culik [1987]).

La universalidad del espacio de Cantor otorga un soporte de universalidad a los autómatas celulares, que hace referencia a las inumerables aplicaciones de estos modelos en la ingeniería y otras ramas de la tecnología. El que los autómatas celulares sean sólo un tipo de funciones continuas sobre el espacio de Cantor también pone límites a sus alcances. Por otra parte el hecho de encontrar el espacio de Cantor, en unos entes de naturaleza claramente discreta lo sitúa como puente entre dos campos aparentemente alejados de la matemática: lo discreto y lo continuo. Esperamos que lo que sigue sirva para aclarar estas ideas.

El espacio de los códigos. Una manera geométrica de construir el espacio de Cantor es considerar conjuntos autosimilares cuyas componentes sean disyuntas obteniendo conjuntos con dimensión entre 0 y 1. Barnsley [1988] introduce el espacio de los códigos para ``numerar'' los elementos

de conjuntos atractores de sistemas iterativos de funciones, es decir de conjuntos autosimilares en espacios métricos completos. Para imaginarnos geográficamente esta representación del espacio de Cantor pensemos en una ciudad (Cantorlandia) en donde la nomenclatura no se da en forma rectangular tradicional (Calle 3 Norte, Carrera 6 Occidente), sino que las direcciones son sucesiones de puntos cardinales, por ejemplo, Norte-Norte-Sur, que se puede abreviar NNS y que significa el sur del norte del norte y para buscar este sector de la ciudad usted se sitúa en el norte, éste está claramente dividido en NN, NS, NE y NO, entonces escoge el NN y este sector también está dividido en cuatro sectores: NNN, NNS, NNE y NNO (ver la gráfica 2). Es claro que un punto de la ciudad, tendrá que ser representado por una sucesión infinita de los signos N, S, E y O. Conviene además considerar que la tarifa para ir de el Norte al Sur es de 1/3 (entre puntos correspondientes), y para ir de NNS a NSE se debe pagar 1/9+1/27. ¿Cuánto se debe pagar por para ir de NNS a SNO? Como el número de puntos cardinales que se usen es intranscendente topológicamente (desde que sea finito) pensaremos en N símbolos: $N = 0,1,2,\ldots,$ N-1. La geometría de Cantorlandia es la del espacio de los códigos que a continuación se presenta.



Gráfica 2. Cantorlandia o el espacio de los códigos cuando $N = 4$.

Consideremos un conjunto de símbolos $\Sigma = \{0,1,\ldots,N\}$. Una sucesión de elementos de Σ es llamada un *código* sobre Σ. Al conjunto de los códigos, que notamos Σ^{ω}, lo dotamos de la métrica d que definimos

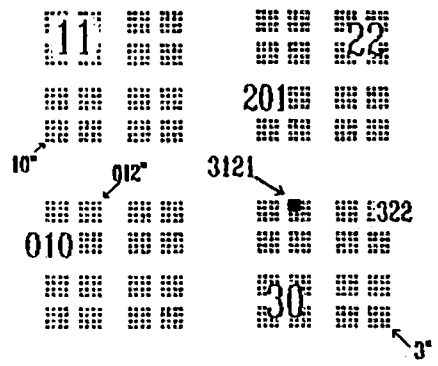
a continuación. Si $u = u_1u_2u_3...$ y $v = v_1v_2v_3...$ entonces la distancia entre ellos está dada por:

$$d(u,v) = \sum_{k=1}^{\infty} (u_k \Delta v_k) / 3^k$$

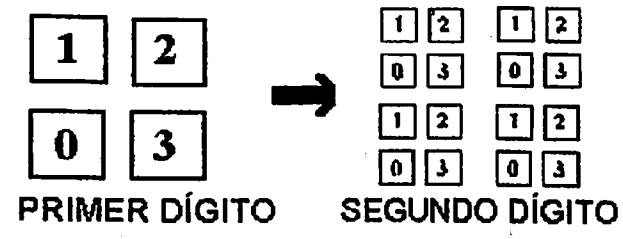
donde $u_k \Delta v_k$ es 1 si u_k y v_k son diferentes y 0 si son iguales. Las vecindades con radio 1/3 son exactamente N. para cada $i \in \Sigma$ el conjunto de todos los códigos que se inician con i, es una de esas vecindades. Así, tenemos N vecindades de radio $(1/3)^k$, cada una correspondiendo a una sucesión finita de k elementos de Σ.

En el ejemplo de Cantorlandia se tiene $N = 4$. En la gráfica 2 los 4 cuadrados mayores representan cada uno el conjunto de sucesiones que empiezan por 0, 1, 2 y 3. Son las cuatro vecindades abiertas de radio 1/3. Cada una de estas vecindades está compuesta de otras cuatro vecindades des radio 1/9 que en la gráfica son los cuadrados más pequeños. Por ejemplo, las sucesiones que tienen como primer término 2 y como segundo término 1, o sea que empiezan por 21, forman una vecindad de radio 1/9, que está metida en una única vecindad de radio 1/3: la de las sucesiones que empiezan por 2. Para imaginarnos dónde queda representada una sucesión dada, conviene enumerar los diferentes cuadrados aprovechando la propiedad de autosimilitud. En efecto, obsérvese que cada cuadrado es una ``copia reducida" del todo. Podríamos por ejemplo considerar que cada uno de los cuatro cuadrados iniciales (las cuatro vecindades de radio 1/3) son el resultado, cada uno, de reducir el espacio a la mitad y trasladar convenientemente. Así, al numerarlos como en la izquierda de la gráfica 3 y al aplicar las cuatro transformaciones, obtendríamos la numeración para las vecindades de radio 1/9 que aparece a la derecha.

Si el proceso se continúa indefinidamente obtendríamos el punto que representa la sucesión. Así, la sucesión de sólo ceros -que notamos 0^*- sería el extremo inferior izquierdo, mientras que la de sólo doses -notada 2^*- sería el vértice contrario. Estos y otros puntos se representan en la gráfica 4. Aceptamos como notación para los códigos que el asterisco como exponente significa repetición indefinida. Así, 03333... se notará 03^* y 21313131.... lo notaremos $2(13)^*$ - o $21(31)^*$.



Gráfica 3. Una forma de representar el espacio de los códigos cuando N = 4.



Gráfica 4. Numeración generada por el proceso que se muestra en la gráfica anterior.

Nótese que además, los puntos que se pueden escribir con sólo ceros y unos conforman la vertical del extremo izquierdo, los que se escriben con sólo unos y doses conforman la horizontal del extremo superior, etc.

Palabras y Acciones. Volvamos ahora al caso general. Sea Σ^* el conjunto de las palabras sobre Σ, es decir las sucesiones finitas de signos de Σ. Asociado a una palabra $w \in \Sigma$, tenemos el conjunto V_w de los códi-

gos de Σ^ω que se inician con w, que es una vecindad de radio $(1/3)^k$, donde k es la longitud de w. Entonces tenemos que estos conjuntos V_w (que son a la vez abiertos y cerrados) forman una base del espacio métrico (Σ^*, d) descrito anteriormente, lo que nos garantiza que este espacio es totalmente disconexo.

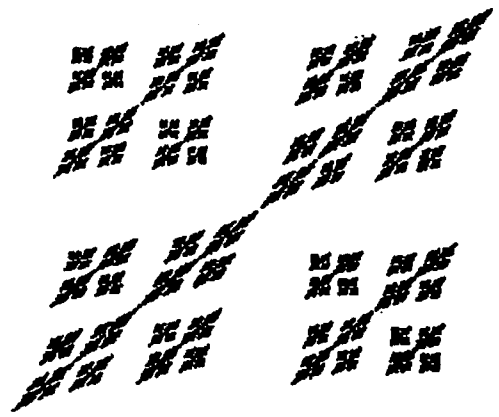
Por otra parte, es claro que ningún código de Σ^ω está aislado ya que los V_w tienen cada uno una cantidad no numerable de elementos. De esto se deduce que el espacio de los códigos es perfecto. Para ver que es compacto se puede recurrir al teorema de Tychonoff, ya que los abiertos que resultan son los mismos de Σ^ω cuando se considera la topología producto y Σ dotado de la discreta.

Que el espacio de los códigos así construido, sea compacto, totalmente disconexo y perfecto, por el teorema caracterizador del espacio de Cantor, tenemos que este espacio es topológicamente equivalente a dicho espacio. Es decir (Σ^*, d) es un viejo conocido con un ropaje especial. Tenemos que con esta vestimenta las palabras aparecen como los abiertos básicos del espacio de Cantor (en el principio fue el verbo...). Pero las palabras, que forman un monoide con la yuxtaposición, además actúan sobre el espacio de Cantor. Si definimos wu como el código que se obtiene al anteponer la palabra $w \in \Sigma^*$ al $u \in \Sigma^\omega$, tenemos al semigrupo Σ^* actuando sobre el conjunto Σ^ω en el sentido algebraico. Esta acción respeta además la estructura topológica y métrica del espacio de los códigos. En efecto, la aplicación $T_w: \Sigma^\omega \to \Sigma^\omega$ definida por $T_w(u) = wu$ es continua y reduce todas las distancias en $(1/3)^k$, donde k es la longitud de w.

Espacios Codificados. Ya estamos preparados para visualizar cómo se hacen, de manera fácil, los "infinitos pegamientos" sobre el espacio de Cantor (o de los códigos, pues topológicamente no se diferencian). Pegar topológicamente significa hacer relaciones de equivalencia. Una relación de equivalencia R sobre el espacio de los códigos será una *congruencia* si para toda palabra $w \in \Sigma^*$ y todo par de códigos $x, y \in \Sigma^\omega$ se tiene que

$$xRy \Rightarrow wxRwy.$$

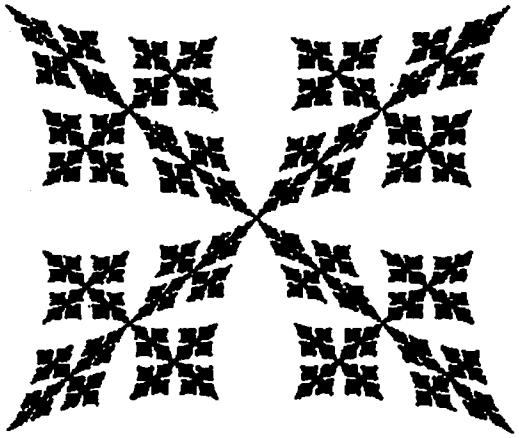
Cuando se tiene una congruencia de este tipo, el hecho de que dos elementos estén relacionados implica que infinitas parejas de elementos lo están. Así, si igualamos (pegamos) 02^* con 20^*, para tener congruencia debemos igualar a 102^* con 120^* y a 3202^* con 3220^*, etc. Una representación geométrica de esta congruencia (en donde no se iguala nada más), siguiendo la numeración que se dio para el caso $N = 4$ (gráfica 2), se muestra en la gráfica 5. Este espacio aunque no es totalmente disconexo tiene infinitas componentes conexas.



Gráfica 5. Espacio resultante para $N = 4$ cuando se hace $02^* \equiv 20^*$.

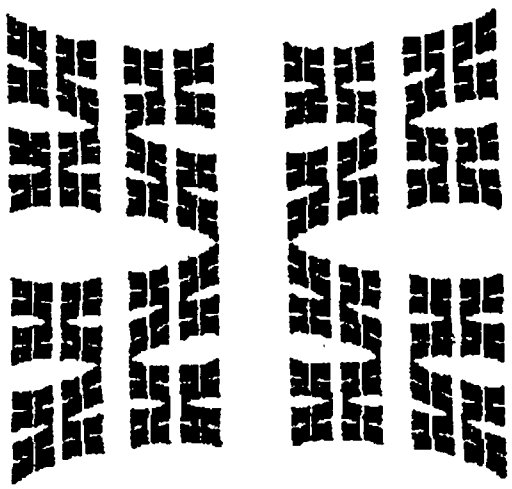
En general si S es un conjunto de parejas de Σ^ω existe la menor relación de congruencia que contiene a S que podemos notar $< S >$. Si $(x, y) \in S$ entonces $(wx, wy) \in < S >$, para toda palabra $w \in \Sigma^*$, y de esta forma se pueden conseguir todos los elementos de $< S >$ a partir de las parejas de S. Tenemos entonces una útil manera de representar congruencias cuando el conjunto S es finito. Al identificar puntos según S, la congruencia $< S >$ identifica infinitos puntos correspondientes en todos los abiertos básicos de Σ^ω. De aquí en adelante siempre que hablemos de igualar o identificar códigos se entenderá que es por congruencia, lo que significa infinitas identificaciones. Los espacios resultantes (cocientes) construidos de esta manera, los llamaremos *espacios de congruencia* sobre Σ^ω. Los espacios de congruencia pueden considerarse

codificados en cuanto para cada punto en uno de dichos espacios, existe uno o varios códigos que lo identifican.



Gráfica 6. Espacio resultante para N=4 cuando se hace $02^* \equiv 20^* \equiv 13^* \equiv 31^*$.

Mostremos otros ejemplos para el caso $N = 4$ ya examinado. Si igualamos 02^* no sólo con 20^* sino además con 13^* y con 31^* obtendríamos un espacio como el de la gráfica 6, que ya es conexo. Otro camino diferente para llegar a este espacio se sugiere en la Gráfica 7.



Gráfica 7. Espacio resultante para $N = 4$ cuando se hace $02^* \equiv 13^*$ y $31^* \equiv 20^*$, otro camino para llegar a la gráfica 6.

Partiendo del espacio representado en la gráfica 6, para obtener un cuadrado homeomorfo a [01]× [0,1]. Se necesita igualar además las esquinas medias de los lados (es decir, 01^* con 10^*, 12^* con 21^*, 23^* con 32^* y 30^* con 03^*).También se deberán igualar otros puntos, pero no nos preocupemos por ellos.

¿Qué hay que igualar para obtener el intervalo [0,1]?

Es como numerar [0,1] en base 4... . Se divide el intervalo en cuatro segmentos iguales, eso determina la primera cifra. Luego se vuelve a dividir cada uno de los intervalos en cuatro pedazos para determinar la segunda cifra, etc. Entonces debemos identificar tres parejas de puntos: 03^* con 10^*, 13^* con 20^* y 23^* con 30^*. Estas parejas sí son suficientes para obtener el intervalo.

Relación por códigos y curvas de Peano. En general todo conjunto compacto autosimilar K se puede poner en correspondencia con Σ^ω el espacio de los códigos de tal forma que a cada $x \in K$ le corresponda uno o varios códigos. Esta correspondencia se puede hacer de diferentes maneras.

Pero el espacio de los códigos también puede servir de "puente" para relacionar espacios diferentes. Si tenemos dos espacios de congruencia sobre Σ^ω, los puntos de cada uno quedan codificados y es natural intentar relacionarlos por intermedio de sus códigos. Esta relación por códigos asocia a un punto de un espacio con un punto del otro espacio si existe un código $u \in \Sigma^\omega$ que corresponde a ambos puntos. Naturalmente esta relación en general no es función. Estableceremos condiciones suficientes para que lo sea.

Si $\alpha_1: \Sigma^\omega \to K_1$ y $\alpha_2: \Sigma^\omega \to K_2$ son funciones sobreyectivas buscamos una función $\beta: K_1 \to K_2$ tal que $\alpha_2(\beta(x)) = \alpha_1(x)$. Para que β quede bien definida y de manera única es necesario que para cada w_1, $w_2 \in \Sigma^\omega$ si $\alpha_1(w_1) = \alpha_1(w_2)$ entonces $\alpha_2(w_1) = \alpha_2(w_2)$. En este caso la imagen de β en un $x \in K_1$ se calcula buscando un $x \in \Sigma^\omega$ tal que $\alpha_2(w)$

$= x$, la imagen de x será $\alpha_2(w) = x$. Se tiene que el siguiente diagrama conmuta:

$$\begin{array}{ccc} \Sigma^\omega & \xrightarrow{id} & \Sigma^\omega \\ \downarrow\alpha_1 & & \downarrow\alpha_2 \\ K_1 & \xrightarrow{\beta} & K_2 \end{array}$$

En caso de que K_1 sea el espacio cociente, la continuidad de β se deduce de que α_2 es continua. Esto lo resumimos con el siguiente

Teorema. *Sean K_1 y K_2 espacios de congruencia sobre Σ^ω tal que para todo w_1, $w_2 \in \Sigma^\omega$ se tenga que $w_1 \equiv w_2$ (en K_1) implique $w_1 \equiv w_2$ (en K_2), entonces la relación por códigos es una función continua $\beta: K_1 \to K_2$.*

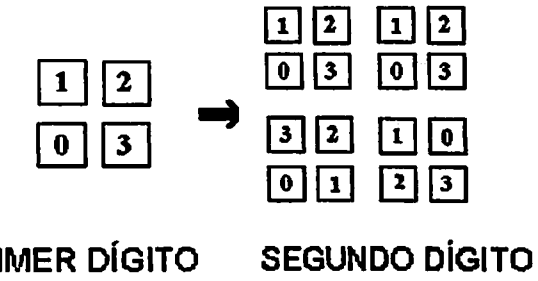
Colorario. *Si numeramos el intervalo* [0,1] *en base* 4 *y si K es un espacio de congruencia sobre Σ^ω en donde se cumple:*

$$03^* \equiv 10^*,\ 13^* \equiv 20^*,\ 23^* \equiv 30^*,$$

entonces la relación por códigos es una curva que llena todo K.

Según este corolario, para hacer una curva de Peano (curva que llena todo el cuadrado), es suficiente codificar convenientemente el cuadrado $[0,1] \times [0,1]$ y aquí podríamos decidir con exactitud cuál es la imagen de cada punto. La codificación que se deriva de la gráfica 2, no nos sirve pues 03^* corresponde al punto medio de la horizontal inferior, mientras que 10^* es el punto medio pero de la vertical izquierda.

Sin embargo, sin consideramos Σ^ω como conjunto autosimilar con otras contracciones como las que se insinúan en la gráfica 8 obtendremos resultados satisfactorios.



Gráfica 8. Inicio de otra codificación para el cuadrado que llamamos α .

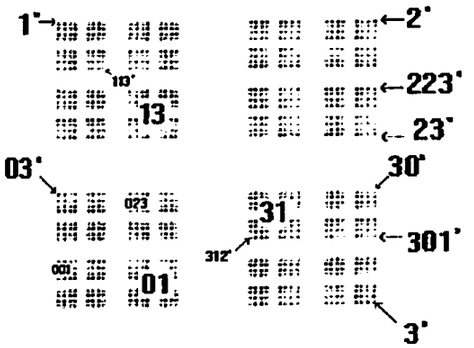
La gráfica 9 muestra la siguiente iteración del proceso, que nos indica el tercer dígito de la numeración α. La diferencia con el proceso correspondiente a la gráfica 2 radica en que aquí las transformaciones 0 y 3 además de contraer por el factor 0.5, suponen una reflexión, cada una sobre diferentes diagonales. Lo que realmente ha cambiado, es la representación en el plano del espacio de Cantor. La gráfica 10 muestra la nueva situación de algunos códigos y palabras (abiertos).

12	12	12	12
03	03	03	03
32	10	32	10
01	23	01	23
30	32	10	30
21	01	23	21
12	32	10	12
03	01	23	03

Gráfica 9. Tercer dígito de la codificación α .

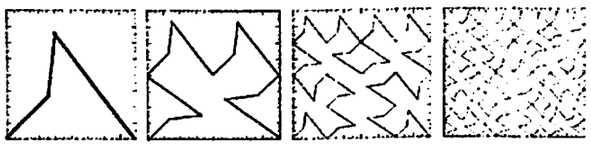
Ahora bien, según esta representación del espacio de Cantor al codificar $[0,1] \equiv [0,1]$ se deben incluir por lo menos las siguientes congruencias: $03^* \equiv 10^*$, $12^* \equiv 21^*$, $23^* \equiv 30^*$, $32^* \equiv 01^*$ para pegar los lados, y $02^* \equiv 13^* \equiv 20^* \equiv 31^*$ para pegar el centro. Entonces es claro que las congruencias $03^* \equiv 10^*$, $13^* \equiv 20^*$ y $23^* \equiv 30^*$, que definen el interva-

lo [0,1], se cumplen en esta codificación del cuadrado y por tanto la relación es una curva de Peano.



Gráfica 10. Algunos puntos y abiertos según la codificación α .

Esta presentación de una curva de Peano tiene la ventaja de que nos indica exactamente dónde está situada la imagen de cada punto del intervalo (gráfica 10). Pero también las cuatro contracciones nos dan un algoritmo para presentarla en forma tradicional como sucesión de curvas "visibles".



Gráfica 11. Algoritmo que muestra la formación de la curva de Peano generada por la codificación

Tome C_0 una curva simple que inicie en \$0^*\$ y finalice en \$3^*\$, entonces C_1 será la unión de las imágenes de C_0 por medio de las cuatro transformaciones y en general, C_{n+1} será la unión de las imágenes de C_n por medio de las cuatro transformaciones. La sucesión de curvas así formadas, converge a la curva de Peano propiamente dicha (gráfica 11).

REFERENCIAS

1. Acosta L. *Una demostración algebraica de la unicidad del conjunto de Cantor.* Tesis de Magister Scientiae,Universidad Nacional de Colombia, Santafé de Bogotá, 1988

2. Alexandroff, P. and P. Urysohn. *Mémorie sur les espaces topologiques compacts.* Verh. Nederl. Akad. Wetensch. Afd. Naturk. Sect. **1,14** (1929), 1-96.

3. Culik, K., Pach, J. and Yu, S. *On the Limit Sets of Cellular Automata.* Research Report CS-87-47, University of Waterloo, Computer Science Departament, 1987

4. Barnsley, M. *Fractals Everywhere.* Academic Press, 1988

5. Estévez, E. *El espacio de los códigos.* Trabajo de grado, Universidad Industrial de Santander, Bucaramanga, 1994

6. Mandelbrot, B. *Los objetos fractales.* Tusquets Editores, 1987.

7. Oostra, V., A. *La topología del orden lexicográfico en el conjunto de Cantor.* Preprint, 1993.

8. Sabogal, S. *El espacio de los códigos.* Memorias del IV Encuentro de Geometría y sus aplicaciones. Santafé de Bogotá, 1993.

9. Willard, Stephen. *General Topology.* Addison-Wesley Publishing Company, 1970

GLOBAL AND LOCAL NEURAL NETWORK MODELS IN BIOTECHNOLOGY

M. N. Karim[1], T. Yoshida[2], S. L. Rivera[3], V. M. Saucedo[1], B. Eikens[1], and G. S. Oh[2]

Abstract. Biological processes are non-deterministic systems. In general, models dealing with microbial pathways and microbial physiology are too complex, and normally they have too many uncertain parameters that are difficult to evaluate as the intracellular concentrations are almost impossible to measure on-line. It is well known that microbial systems go through different *phases* during the cultivation process, and different types of models are needed to accurately describe the behavior of these processes. It also is important to generate models based on on-line measured variables so that models can be used in process control and on-line optimization of biological systems. Neural networks are a data-driven modeling approach. They have good generalization and prediction capabilities. Different types of networks exist such as feed forward-back propagation networks and recurrent neural networks. Localized networks such as Radial Basis Function networks have found use in on-line process control due to the fact that these networks require less computational time. Another type of network, Cascade Correlation, has been developed to generate optimal structure for a particular application. Important issues in selecting and using neural networks in biotechnology are: proper scaling of data, selection of the appropriate structure of the network including appropriate choice of the input and output variables, and the purpose of the network. Case studies dealing with fuel alcohol production using renewable biomass from agricultural wastes by fermentation with *Zymomonas mobilis* and recombinant *Escherichia coli* are provided. Preliminary results for the production of MAb using hybridoma cells also are included.

Resumen. Los procesos biológicos son sistemas no deterministicos. En general, los modelos que tienen que ver con caminos microbianos y fisiología microbiana son muy complejos y normalmente tienen demasiados parámetros que son difíciles de evaluar como las concentraciones intracelulares que son casi imposibles de medir en línea. Se sabe que los sistemas microbianos atraviesan diferentes fases durante el proceso de cultivo, y se necesitan diferentes tipos de modelos para describir con exactitud el comportamiento de estos procesos. Es también importante generar modelos basados sobre variables medida en línea de tal forma que puedan ser usados en control de procesos y en la optimización en línea de sistemas biológicos. Las redes neuronales son un enfoque para realizar modelos basados en datos. Gozan de buenas capacidades de generalización y predicción. Existen diferentes tipos de redes tales como las recurrentes y las de retropropagación.

---

[1] Department of Chemical and Bioresource Engineering, Colorado State University, Fort Collins, Colorado, USA.
[2] International Center for Cooperative Research in Biotechnology, Faculty of Engineering, Osaka University, Suita-shi, Osaka 565, Japan.
[3]Department of Chemistry and Chemical Engineering, Stevens Institute of Technology, Hoboken, New Jersey, USA.
* *Corresponding author*

Redes específicas como las redes con funciones de base radial han encontrado uso en control de procesos en línea debido al hecho que requieren menos tiempo de computación. Otro tipo de red, la red de correlación de cascada se ha desarrollado para generar estructura óptima para una aplicación particular. Aspectos importantes al seleccionar y usar redes neuronales en biotecnología son: escalamiento de datos apropiado, selección de la estructura de la red adecuada incluyendo selección de las variables de entrada y salida, y el propósito de la red. Se presentan algunos casos que tienen que ver con la producción de alcohol combustible por fermentación con *Zymomonas mobilis* y *Escherichia coli* recombinante usando biomasa renovable a partir de desechos de agricultura. Se incluyen también los resultados preliminares para la producción de MAb usando células hybridoma.

Key Words: neural network estimator, radial basis function, feed forward neural network, *Zymomonas mobilis*, recombinant *Escherichia coli*

1. Introduction. Biological processes are time variant and nonlinear in nature. The time variant nature of these processes results from the fact that microbial species are slowly but continuously going through changes in physiological and morphological conditions. Many researchers define these as different physiological *states* or *phases* of the microbial population (1,2). These phenomena are observed in batch, fed batch, and even in "steady state" continuous cultivation. Of course, all cells go through the normal genetic mutation process, which also makes the microbial system time variant. The nonlinear behavior of a cultivation process may be due to many factors such as kinetics of cellular growth and product formation, thermodynamic limitations of the process, mass transfer and heat transfer effects, and fluid mechanics (turbulence) inside the cultivation vessel. In order to quantify the behavior of microbial systems, one needs to make appropriate measurements. In a cellular environment, there are components that are excreted by the cells (extra-cellular products) and others remain within the cells (intra-cellular metabolites). Measurement of extracellular products is easier, and methods may be developed for the on-line determination of these variables. However, intra-cellular components are almost impossible to measure on-line. For on-line process optimization schemes, one may need quantitative values of both intra- and extra-cellular components. Models or methods that can provide these values on-line are desirable. Neural network is a viable paradigm of modeling that can be used for this purpose.

In an intelligent computer process control operation, one needs frequent measurements of on-line state variables. If these state variables are not available, one needs to "estimate" them. These estimates that can be obtained using a neural network (3) may be used for on-line optimization (in a "supervisory" capacity) providing set points to the process controllers. Bioreactors are normally equipped with traditional process controllers. In order to improve the performance of the cultivation process, an intelligent monitoring system and a data reconciliation software may be added to the process computer (4). One

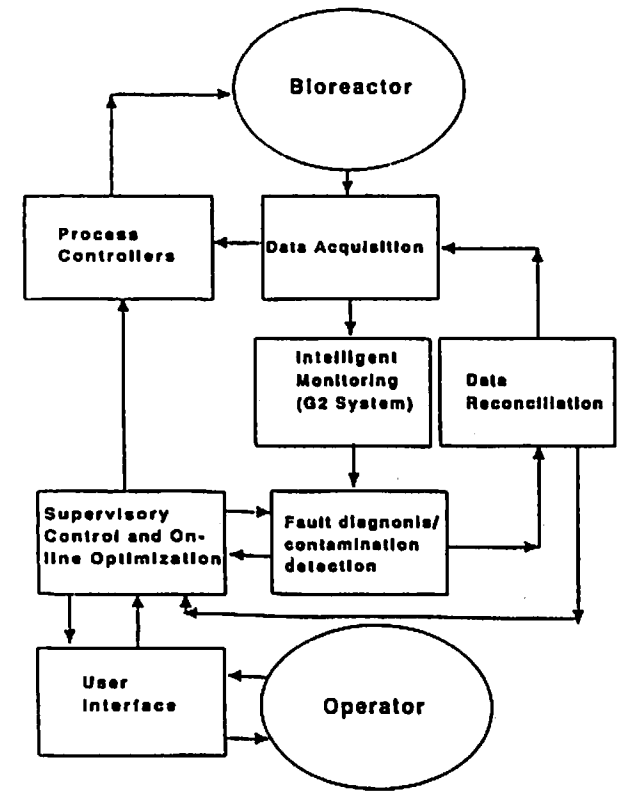


Figure 1. An intelligent bioreactor control configuration.

can include fault diagnosis (both for hardware and biological components) in the software. These can be accomplished using neural networks. Figure 1 shows a scheme for an intelligent process control environment for a bioreactor.

2. Modeling of Biological Processes. The purpose of modeling is to understand the fundamental behavior of the microbial processes. Models are useful for on-line process control and optimization as mentioned earlier. Modeling is a prerequisite for scale-up of bioprocesses. It reduces the cost of process development by eliminating unnecessary experimentation. Modeling allows one to study the interactions of various parameters on the cultivation process. Modeling exercise can provide valuable information regarding process safety and environmental hazards.

Most common biological models come from fist principles by applying principles of conservation of mass and energy. Into this framework, one incorporates thermodynamic, kinetic, heat, and mass transfer correlations. These types of models are called deterministic models, and these could be "unstructured" or "structured" depending on the complexity and assumptions made on the cellular behavior. Unstructured models usually have fewer parameters, whereas structured or mechanistic models may have numerous parameters. Unstructured models that are based on the assumption that all cells in a bioreactor behave uniformly as one species and are governed by simple kinetic expressions such as Monod Kinetics are more commonly used. These models, however, normally fail to predict true transient behavior of the microbial process especially during the early and late states of cultivation processes (5). The structured approach tries to incorporate intra-cellular mechanisms such as dynamics of mRNA synthesis in the models (6). However, these models have too many parameters, many of which cannot be determined from the *in vivo* experimentation, and reliability of these parameters are questionable. Different from first principle models, input-output models are "black box" models where one is mainly concerned with obtaining a prediction of the outputs based on the input data. This modeling approach (known as "process identification") is very common in process control and optimization schemes. Very powerful and reliable methods exist that can provide robust "identification" of various nonlinear and time variant processes (7).

One of the important issues in modeling is whether models should be simple or complex. Ideally, simple models are preferred.

However, before a modeling exercise is stated, one must determine the purpose of any particular model. For on-line use, normally the model should have few parameters and should converge fast to the solution. In some circumstances, modeling efforts are governed by the available data. The data used may be on-line or off-line. Usually off-line data have large sampling times, and thus the number of data points are limited. This dictates the number of parameters one can estimate that are meaningful. The choice of a particular modeling paradigm also depends on the type of hardware (computer) in which the modeling process is carried out. Figure 2 shows different modeling paradigms that have been used in bioprocesses (8). One may use more than one paradigms to generate a meaningful and useful model. For example, neural network models ("approximation models") can be combined with "logical" and "classification" models to give a comprehensive picture of a cultivation process.

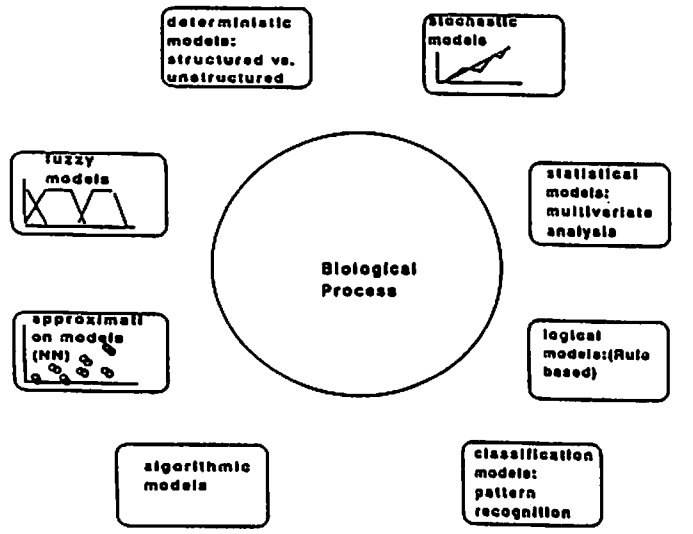


Figure2. Different modeling paradigms.

3. What Is a Neural Network? Neural networks are designed to mimic the actions of neurons in the human brain. Inputs to the artificial neural network (ANN) are processed and fed to neural nodes where they are summed and activated after which the signals are sent to an output

processing node that predicts the value of the output variable. Normally an ANN is created with an input layer, one or more hidden layers, and an output layer. The most common ANN is the well known feedforward back propagation neural network (FBNN). Just as it may happen in human neurons, the ANN needs to be trained with appropriate training data sets. Once the ANN has been trained, it can be used as a model to predict the desired output variables of a process. Typical inputs to an ANN used as a state estimator (3) in bioprocessing are on-line measured variables such as temperature, pH, feed flow rates, percent carbon dioxide or carbon dioxide evolution rate, redox potential (for anaerobic cultivation), and optical density. The output variables are normally state variables that are measured off-line. Figure 3 shows a typical configuration of an ANN.

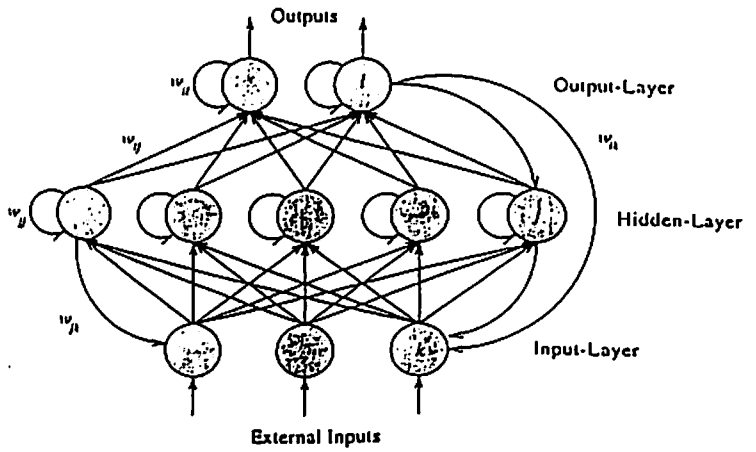


Figure 3. A feed forward artificial neural network configuration.

In general, a FBNN has N neurons distributed in one input layer, one hidden layer, and one output layer. When an input sequence or vector, x_p, from a pattern p is presented and propagated through the network, a corresponding vector, y_p, is obtained from the output layer. Each connection between neurons has a weight w_{ij} associated with it. The bias unit provides a threshold for the activation of the neuron. A typical neuron, j, performs two calculations: a weighted linear combination of its inputs to obtain the activation S_{pj} and a nonlinear mapping of this activation value, usually by means of a non-decreasing and differentiable

sigmoid function, $f(s) = 1/(1+e^s)$. The output from a neuron is given by:

$$y_{pj} = f(S_{pj}) = f(\sum_{1}^{N}(w_{ij})(x_{pi}) + w_{N1,j})$$

The set of training examples consists of P input/output vector pairs (x_p, y_p). Weights are initially randomized to small values and then selected to minimize the objective function defined as the total squared estimation error $E(w)$ for all output units and all patterns:

$$E(w) = \sum_{p=1}^{P}\frac{1}{2}\sum_{k=1}^{L}(d_{pk} - y_{pk})^2 .$$

In the above equation, d_{pk} is the experimental data, and L is the number of output nodes. The objective function $E(w)$ is a continuous differentiable function of every weight, and thus a nonlinear numerical optimization technique is required to minimize it. The problem is to find a set of weight w which will minimize the function $E(w)$ during the training operation. The back-propagation algorithm typically optimizes the objective function by the method of Steepest Descent requiring the gradient calculation. The Steepest Descent algorithm is rather slow, and recently more efficient algorithms such as the Conjugate Gradient method and Genetic Algorithms have been used for solving the unconstrained optimization problem (3,9).

4. Different Types of Neural Networks. While considering a particular application, one must decide what type of networks should be used. Neural networks are classified either as global or local networks. Among the global networks, the most widely used networks are:

Multi-Layer Perceptrons (FBNN belongs to this class)
Recurrent Neural Network (RNN)
Cascade Correlation Network (CCN)

Different from FBNN, the recurrent neural networks are more general in the sense that connections are allowed both ways between a pair of neurons and even from a neuron to itself. The RNN allows dynamics of the network to be considered, and it is thus useful for temporal association. However, the number of weights (for a fully recurrent network) to be determined may easily become quite large and for this reason, some prefer Cascade Correlation types of networks where connections and nodes are added as required (10).

Local networks such as Radial Basis Function Network (RBFN) is particularly suitable for applications where on-line control and optimization is the main goal (11)). In RBFN, one is required to locate *centers* among the input/out pairs such that the sum of squares of the distance of the center to the training data set is minimized. The centers are equivalent in concept to the weights in FBNN. However, RBFN may fail in predicting values if the prediction space does not contain any center. A global network will perform better in these circumstances. The following table gives a comparison between the local and global networks.

Feature	Local	Global
Learning speed	Fast	Slow
Node requirement	High	Low (could be high)
Input scaling	Sensitive	Less sensitive
Node input	$q = \|x - c\|$	$q = x^T w$
Activation	$f(q) = e^{-q/\sigma}$ (for Gaussian function)	$f(q) = 1/(1+e^{-q}$

5. Modeling Criteria. Selecting appropriate criteria to discriminate between different types of models is a prerequisite for a good modeling approach. A compromise must be made between the desire to have a simple model with a fewer number of parameters versus the desire to have more accurate predictions at the cost of a large number of

parameters. Akaike's Information Criterion (AIC) serves this purpose very well. This index has been used effectively in on-line process control application by many researchers for model discrimination (12,13). The AIC is defined as:

$$AIC(\phi) = 2 \log\left(\frac{\sigma^2}{N}\right) + \phi q .$$

A common value of ϕ is 4 and q is the number of model parameters. N is the number of data points and σ^2 is the standard deviation of the error between the model and data. A plow of $AIC(\phi)$ versus q generally yields a minimum, and the appropriate model parameters are given by the value of q which minimizes the value of $AIC(\phi)$.

Another concept that may be of interest is the use of the periodogram, $I(\omega_j)$, which is a function of frequency, ω_j, and gives an indication regarding the frequency content of input signals that are able to excite the process output. The periodogram of N data points, x_t, is defined as:

$$I(\omega_j) = N^{-1} \left\| \sum_{t=1}^{N} x_t \, e^{-it\omega_j} \right\|^2 .$$

One must select the input vectors such that the periodogram due to these inputs has a significant number of non-zero values at the frequency range of interest (14).

Selection of the input set for training purposes also can be made by using Principle Component Analysis (15) of the candidate data set. Principle components of the data set belonging to the same group will be *aligned* in the same direction. This type of approach allows one to reject data containing false or wrong information. However, one must be careful about rejecting data sets containing *outliers* that may be meaningful and represent the behavior of the biological system although it may not be obvious from the data set.

6. Neural Networks in Biotechnology. Application of neural networks in biotechnology has gained significant momentum in recent years. Many researchers (3, 16, 17, 18, 19, 20, and 21) around the world have used neural networks in various applications such as estimation of biological process state variables, biological phase determination, and application in conjunction with expert systems.

Before one applies a neural network to a set of data, one must resolve some of the issues such as topology of the network. The topology determines how many layers are to be used and how many nodes are to be included in each layer. In most applications, these questions are resolved by trial and error. usually the input and output nodes re fixed, and in general, one hidden layer is enough for estimating any nonlinear function (22). One then uses some error criteria or number of iterations (fixed) to determine how many nodes are to be used in the hidden layer. Usually the topology that gives the smallest error is selected as the "best" choice. Figure 4 shows this concept.

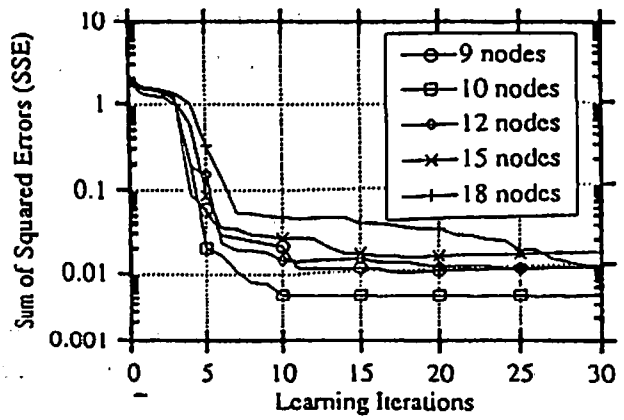


Figure 4. Sum of the squared error as a function of number of iterations and number of nodes

It is feasible that one may have to use different topology for different physiological states of the biological system (23). One also may use algorithms that automatically adds or deletes nodes during an on-line application. These algorithms are adaptive evolutionary algorithms and are subjects of intensive research. Another issue in training a network is to determine whether to use the data in sequential or parallel configuration. In other words, if one has M experimental data under

different conditions, should one present the data to the network one set at a time or use all the data in the matrix form and process them in a parallel fashion? The authors' experience is that sequential training is the best way to present the data (3). In sequential training after the network has converged using a particular data set, the weights are saved and used as initial weights for the next data set and so on. Once a network has been trained with appropriate data sets, one must use some other data set to test the network for its "generalization" or predicting capability.

7. Case Studies. Three case studies will be given to illustrate the usefulness of ANN in biotechnology.

Case Study 1. This case study deals with *Zymomonas mobilis* fermentation producing ethanol. The purpose was to use the neural networks as a state estimator, i.e., predict the state variables (biomass, ethanol, and glucose concentrations) from the on-line measured variables such as temperature, redox potential, percent CO_2, and optical density (O.D.).

Materials and Methods. Details of materials and methods can be found elsewhere (3). *Z. mobilis*, strain ATCC 10988, was used in this study. It was obtained as lyophilized pellets from the American Type Culture Collection (Rockville, Maryland). The media contained: 3.0 g/l years extract, 0.5 g/l Bacto-peptone, 50 g/l glucose, 1.0 g/l $(NH_4)_2SO_4$, 1.0 g/l KH_2PO_4, and 0.5 g/l $MgSO_4$. $7H_2O$. Cultivations were conducted in a 7-liter Chemap bioreactor with a working volume of 3 liters. Prior to inoculation, the entire assembly was sterilized at 121C for 60 minutes. Cultures were seeded with 10 percent volume inoculum, made anaerobic by continuous sparging of nitrogen. Stirrer speed was set at 150 rpm to provide gentle agitation. The following measurements were made: pH (Ingold), temperature (J-type thermocouple), carbon dioxide concentration (Infrared Analyzer), acid/base addition (Watson Marlow pump, total gas flow rate (Omega mass Flowmeter), and redox potential (Ingold). All signals were received by an HP 3497A Data Acquisition and Control Unit and processed by a 386 IBM-PC compatible microcomputer (40 MHz). Off-line samples (30 ml) were collected for the analysis of

biomass (dry weight and O.D., Bausch and Lomb Spectronic 21), ethanol and glucose concentrations (HPLC, Waters with a Biorad Organic Acid column).

Five experiments were conducted at different temperatures: 30 C, 33 C, 35 C, 37 C, and 39.5 C to obtain information about the process behavior at different environmental conditions. pH was maintained constant at 6. Data was recorded from the available on-line sensors and from off-line sample analysis performed every 15 minutes. Another experiment was performed in which the temperature was not kept constant but was varied following a profile generated by an optimization procedure (24). This data set was used for a "generalization" test of the neural networks.

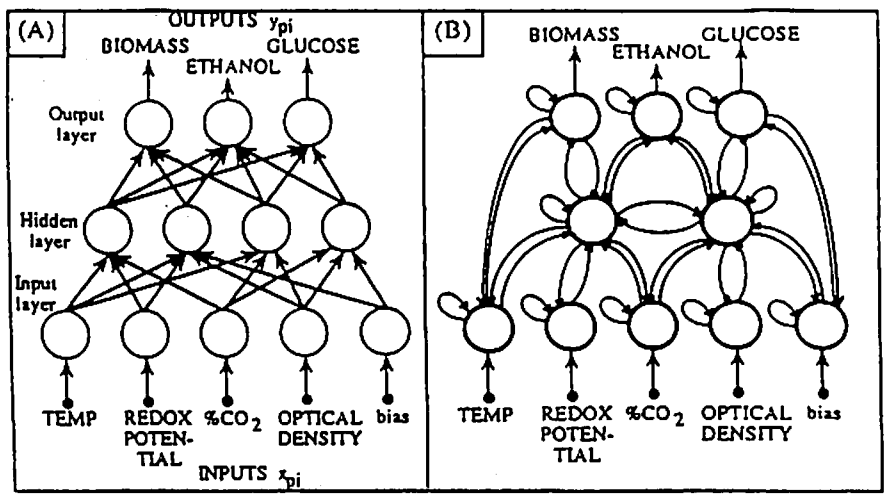


Figure 5. State estimation neural network topology for the *Zymomomas mobilis* fermentation case study for ethanol production.

The FFBN network selected after optimizing the nodes had the following topology: four input nodes plus a bias node, three hidden layer nodes plus a bias node, and three output layer nodes Figure 5 illustrates the concept. A fully recurrent neural network (RNN) with two hidden nodes also was trained to compare the performance of these two different types of networks. Typical profiles of on-line measured variable are

shown in Figure 6. Both networks were trained by presenting all the data sequentially.

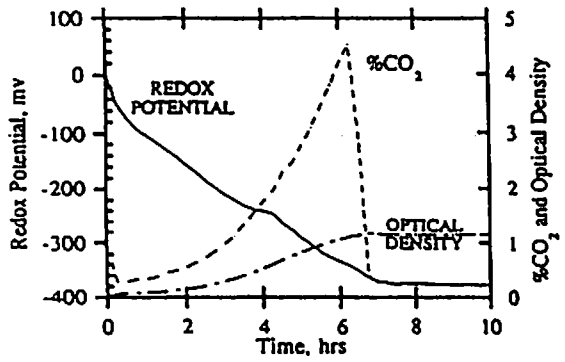
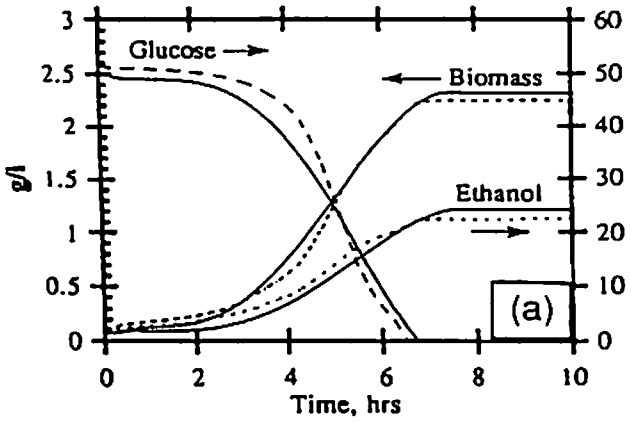


Figure 6. Typical on-line varaiables for the *Zymomonas mobilis* fermentation case study for ethanol production.

After the networks were trained, they were tested for "recall" and "generalization" characteristics. The "recall" characteristic of the network is used to test if the network can predict a pattern on which it was trained in the past. In order to see if the network can predict state variables at conditions other than the training sets, "generalization" tests are performed where the network is asked to predict state variables using a data set in which it has not been trained. Figures 7 and 8 show these results. Figure 7 shows results for the FFBN, and Figure 8 shows results for the RNN networks in "recall" and "generalization" modes. It is seen that RNN has a better "generalization" characteristic as it can predict all the state variables with less error. The "recall" characteristics are comparable for both networks.

Case Study 2. This case study was conducted in order to determine the effectiveness of local versus global networks in "identifying" a recombinant *E. coli* fed-batch fermentation for ethanol production using xylose. The identification procedure is a prerequisite for on-line process control and optimization. In "identification," one assumes that the state variables are available for on-line measurement. In the present case, an on-line HPLC system was used for measuring concentrations of ethanol, xylose, and organic acids. The purpose of the network was to predict the

state variables *N*-step ahead so that an optimizing model predictive control (MPC) algorithm could be implemented in "real time."



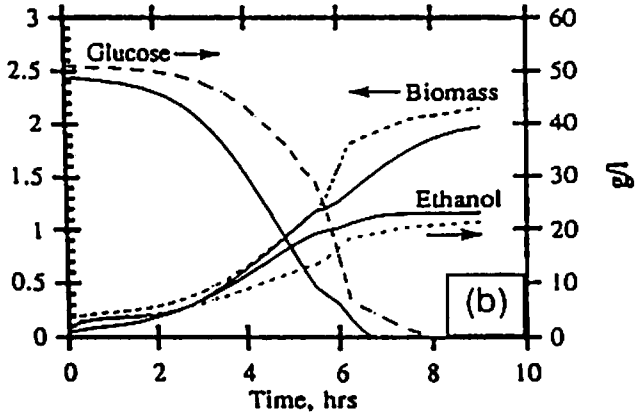
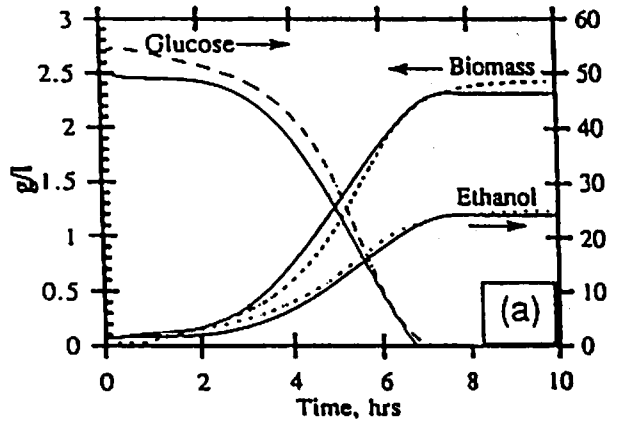


Figure 7. Results of state estimation using a feedforward neural network for the *Zymomonas mobilis* fermentation case study for ethanol production.



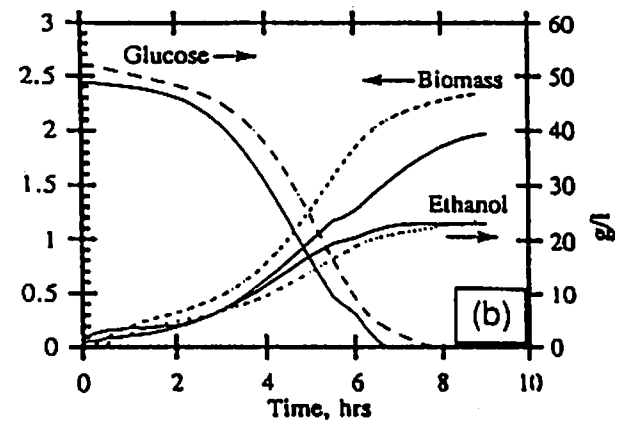


Figure 8. Results of state estimation using a recurrent neural network for the *Zymomonas mobilis* fermentation case study for ethanol production.

Materials and Methods. Details of the methods and materials are given elsewhere (25). The strain used is a recombinant *Escherichia coli* ATCC 11303 with plasmid PLOI297 containing the pyruvate decarboxylase and alcohol dehydrogenase genes from *Z. mobilis* (26). The plasmid also contained an antibiotic resistance marker for guard against the segregational instability. Luria broth containing xylose was used as the media. Tetracycline (10 mg/l) was added to the medium. The temperature was controlled at 34C, and the pH was regulated at 6.5. The experiment was conducted in the same 7-liter Chemap fermenter as described in Case Study 1. The feed concentrations were 120 g/l and 200 g/l. The initial culture volume was 2 liters, and the final volume was 5 liters. Variables measured on-line were pH, T, percent CO_2, concentrations of xylose, ethanol, and organic acids (every half hour by on-line HPLC). Due to the limited number of ports in the fermenter, cell mass (dry weight) and O.D. (measured by Spectronic 21) were analyzed off-line by taking samples every half hour. Feed was started after the exponential growth phase. Two random feeds were affected: one with a mean of 0.5 ml/min (feed xylose concentration: 120 g/l) and the other 1.0 ml/min (feed xylose concentration: 200 g/l after 50 hours). The feed profile is shown in Figure 9. The corresponding changes in biomass concentration is shown in Figure 10.

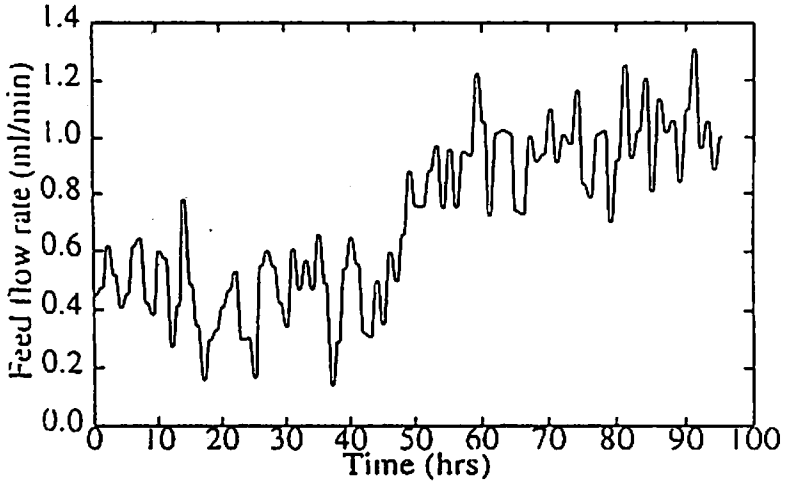


Figure 9. Feed rate profile for the identification experiment for the recombinant *Escherichia coli* fermentation case study.

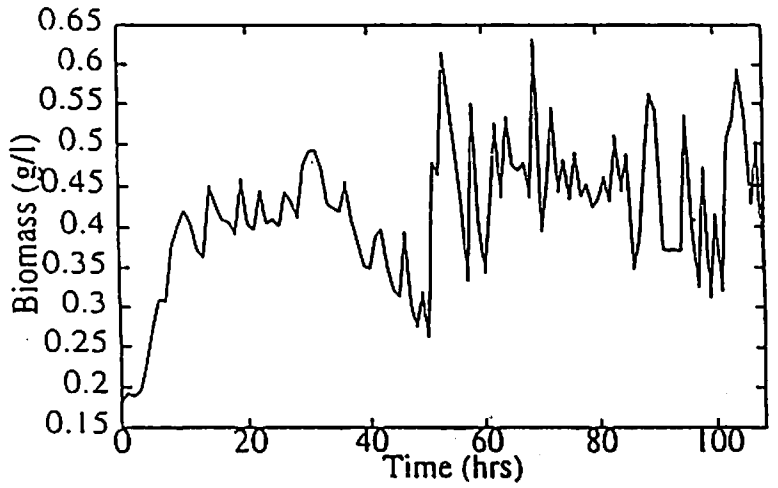


Figure 10. Biomass concentration, *g/l*, profile for changes in the feed flow rate of Figure 9.

A global network (Multi-Layer Perceptrons, MLP) and a local network (RBFN) were trained on the data for ethanol, xylose, and total acids. The data until 70 hours (the fed-batch operation) were used for training purposes, and it is shown in Figures 11, 12, and 13 that both types of networks were trained reasonably well using these data points. The one-step ahead prediction (after 70 hours) is compared, and the RBFN prediction is worse than the MLP prediction as seen in Figures 11, 12, and 13 after time greater than 70 hours. This is surprising as RBFN is expected to perform well in a localized environment. The reason for this unexpected behavior lies in the fact that the "centers" used to train the RBFN network minimized the distance in each "cluster" of the training data set, and these centers are outside the test data set (Figure 14), and thus extrapolation was attempted in this study. The RBFN is rather unreliable under these circumstances (27).

Though time-consuming with respect to convergence, MLP networks (global) are more robust with respect to prediction of variables for on-line process optimization. The configuration of the MLP network used for the one step ahead prediction is shown in Figure 15.

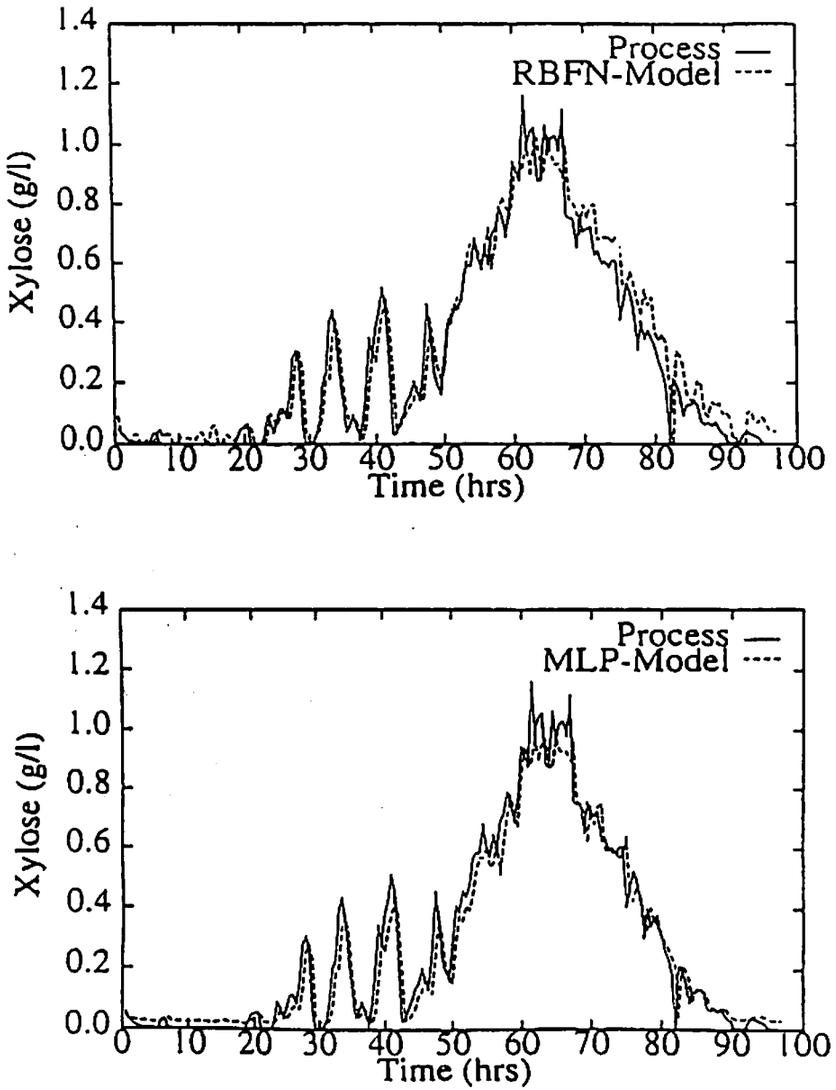


Figure 11. One step prediction of xilose concentration, *g/l*, using radial basis function (RBFN) and multi-layer perceptron (MLP) networks.

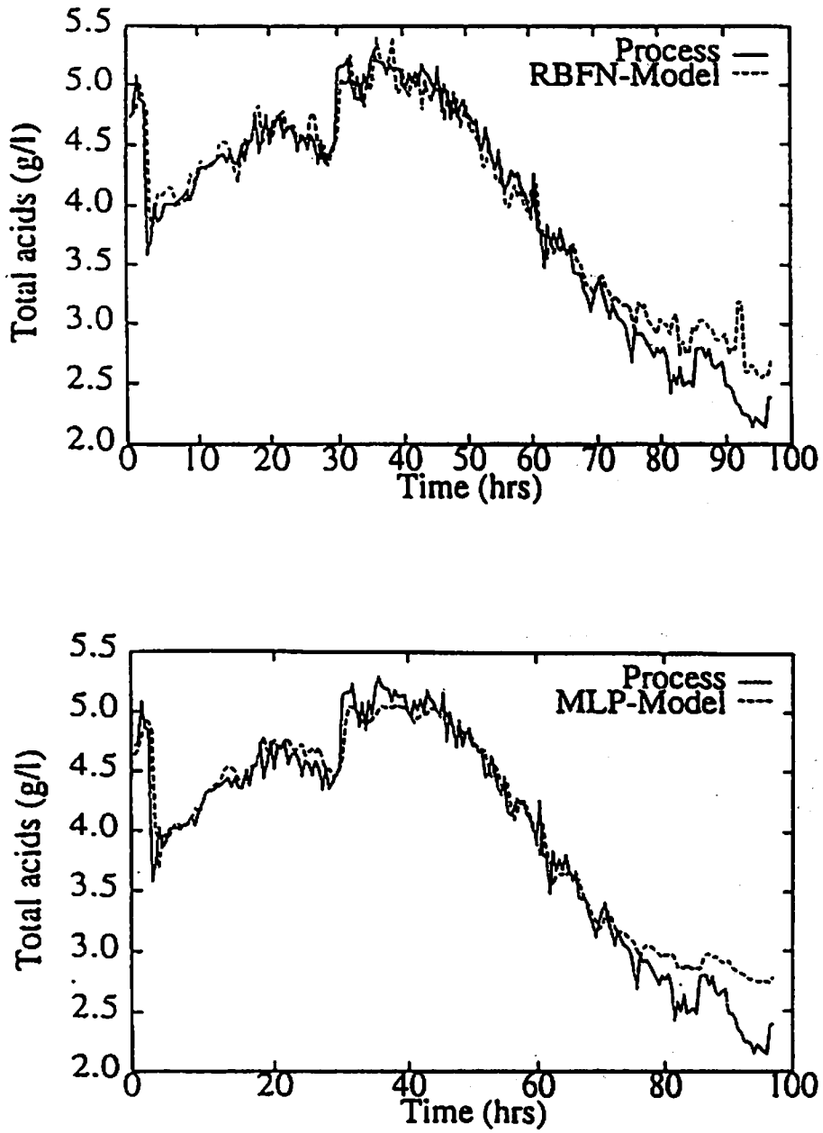
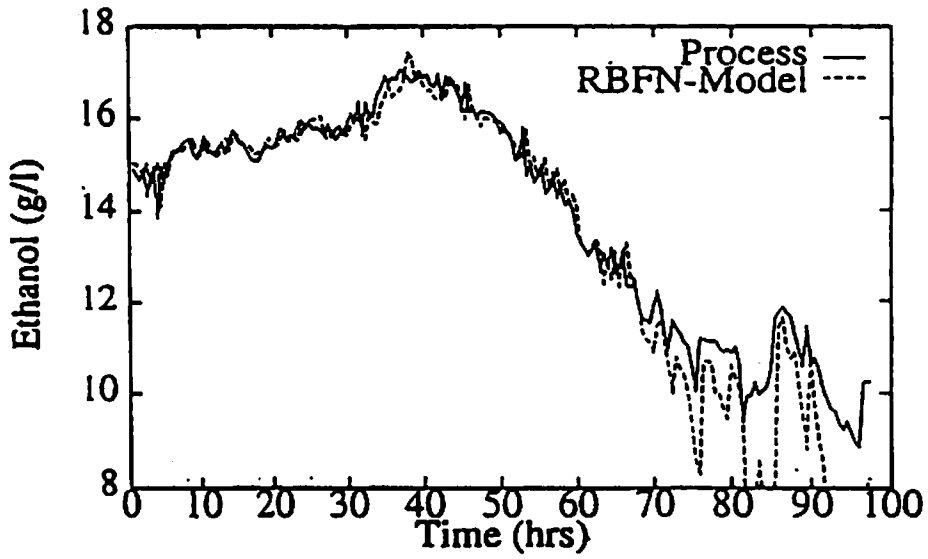


Figure 12. One step prediction of total acid concentration, *g/l*, using radial basis function (RBFN) and multi-layer perceptron (MLP) networks.



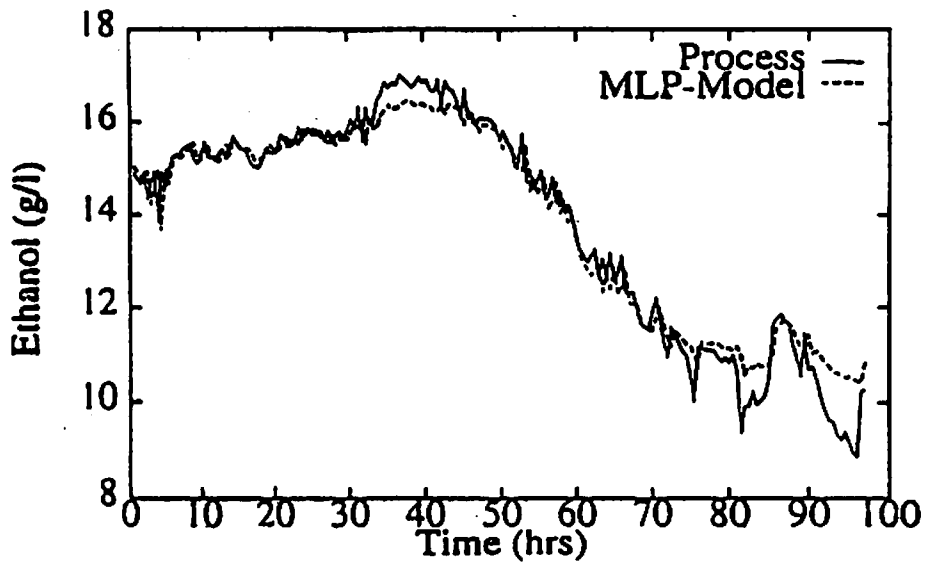


Figure 13. One step prediction of ethanol concentration, *g/l*, using radialbasis function (RBFN) and multi-layer perceptron (MLP) networks.

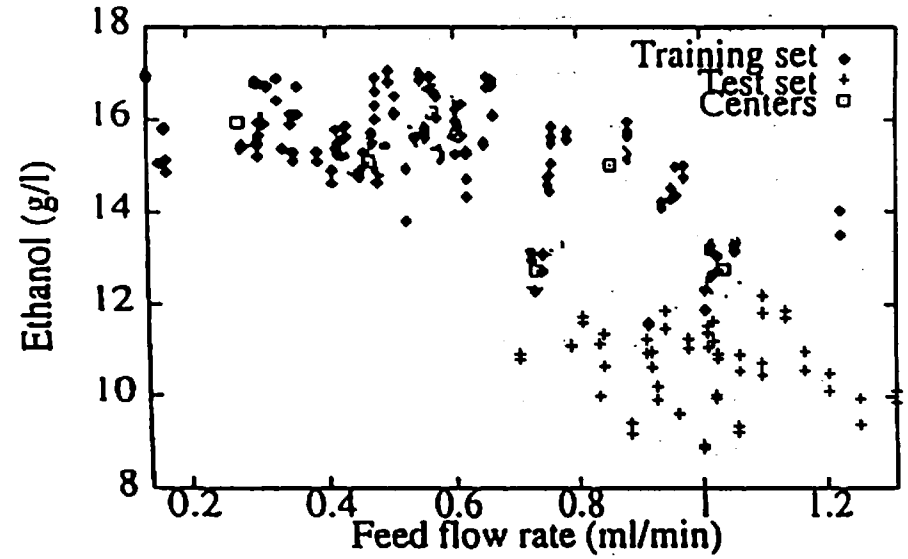


Figure 14. Scatter plot of ethanol concentration, g/l, and feed rate, ml/min, to show the location of the training and test data, and RBFN centers

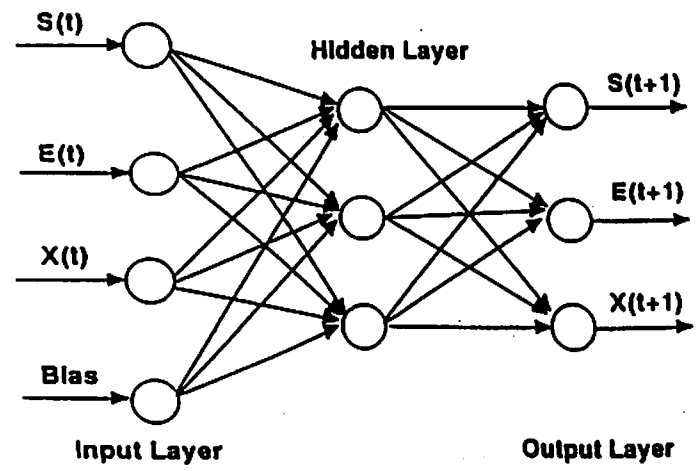


Figure 15. Configuration of the one step ahead prediction (MLP network)

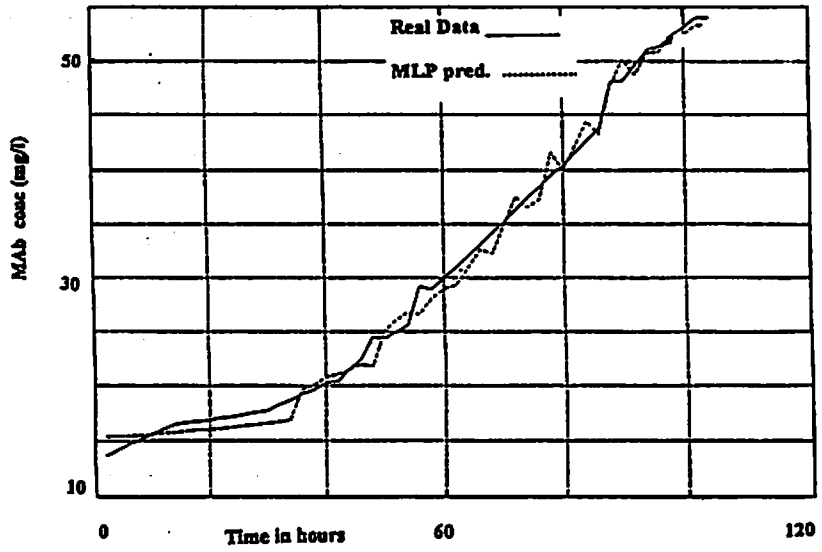


Figure 16. Prediction of monoclonal antibody concentration (Mab), (*mg/l*), using an MLP network.

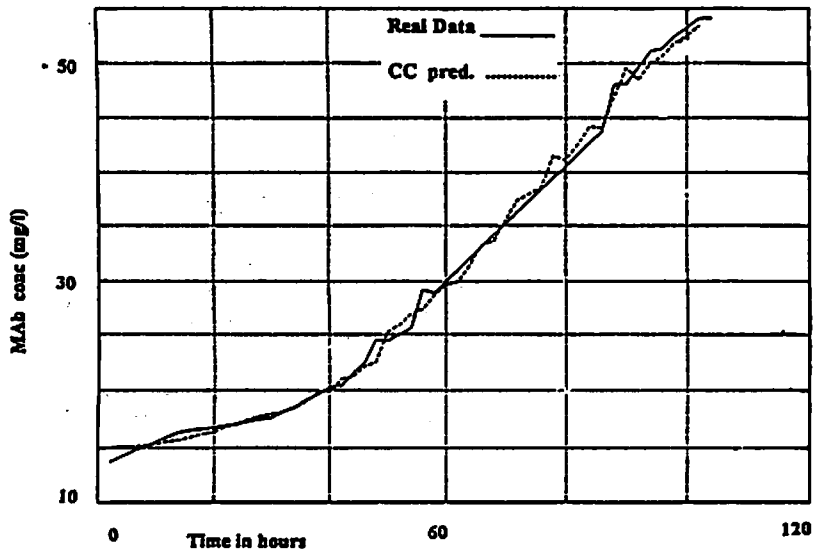


Figure 17. Prediction of monoclonal antibbody concentration (MAb), (*mg/l*), using a cascade caorrelation (CC) network.

Figure 18. Prediction of glutamine concentration, *mg/l*, using an MLP network.

Case Study 3. In this brief case study dealing with monoclonal antibody (MAb) production using hybridoma cells, different global networks were used to predict MAb and viable cell concentrations. The details of this case study are reported elsewhere (23). Since MAb assay and viable cell counts are time-consuming, it will be beneficial if these variables can be predicted while the cultivation is still on-going so that one can make some decisions regarding the optimum feeding policy and the harvest time of the fed-batch culture.

Materials and Methods. The details of materials and methods are given in (23).

Cell line and medium: The cell line used throughout was AFP-27, a mouse-mouse hybridoma producing an IgG monoclonal antibody against human a-fetoprotein. The cells were grown in Dulbecco's Modified Eagle's medium (4.5 mg ml-l glucose-DMEM-H or 1.0 mg ml-l glucose-DMEM-L, Gibco Labs),

supplemented with 5% fetal bovine serum (Bioproducts) and 1% penicillin streptomycin solution (Sigma).

Inoculum culture: For the preparation of seed for the inoculum culture, 120~150 ml of cell suspension was incubated in a T-flask until viable cell concentration reached $1.0x10^6$ cells/ml at 37C in a T-flask. Inoculum cultivation was carried out in a 900 ml spinner vessel with 550 ml of the medium (Shibata Bio). The culture was maintained at 37C and agitated at 50 rpm. For oxygen supplied to the culture, the head space of the vessel was ventilated by air at a rate of 100 ml min-1. The pH was maintained at 7.2 by the supply of CO_2 with an on-off control. When the total cell concentration reached $5.0~7.0 \times 10^5$ cell/ ml, the preculture was terminated and used as the inoculum for a further series of cultivation.

Culture conditions: All experiments were conducted using a Chemap FZ-2000 fermenter equipped with a 4-flat blade turbine impeller and with an initial medium volume of 2.2 l. The agitation speed was maintained at 50 rpm. Oxygen was supplied from the head space and by bubble-free aeration through silicon tubing. When the DO decreased below 50 percent, pure oxygen was added to maintain the DO level at 50 percent. However, the DO control system was inactivated during measurement of the oxygen uptake rate. The temperature and pH were controlled at 37C and 7.2 (by 1.0 N NaOH). The fermenter was interfaced with a computer system for monitoring and control as described below.

Feeding policy: Three experiments were performed with three different types of feeding. The first experiment (EX 8) was done with "low feed rates" of glucose, and the glutamine and feed rates of these were the same. The second experiment (EX 13) also had low feed rates, however, the feed rates of glucose and glutamine were varied independently based on feeding strategies dependent on the specific glucose uptake rate. In the third experiment (EX 11), the feed rate was "high," but the glucose and glutamine feed rates were the same.

Instrumentation and computer control system: The software system used for on-line monitoring, data processing, and control was constructed using a real-time expert system shell, G2 (Gensym, Cambridge), installed on a UNIX-based, SUN Sparcstation 2 computer.

The following inputs were used in the network: base addition rate (BAR), dissolved oxygen (D.O.), oxygen uptake rate (OUR), laser measurement (LM), glucose feed rate (GLUFR) and glutamine feed rate (GLNFR). The following off-line measurements were used as the outputs (state variables for the hybridoma cell culture process): glucose concentration (GLU), glutamine concentration (GLN), lactate concentration (LA), ammonia concentration (AM), viable cell concentration (VC), and concentration of MAb (Mab).

Three different types of global neural networks were compared for their predicting capabilities. After trial and error, it was found that a structure of 6-10-5 (if MAb measurement is deleted from the set) or 6-10-6 were satisfactory in training both the MLP and the RNN networks on all three data sets. However, for CC, only one hidden node was needed, but in this case, the input and output layers were fully connected. All global networks, RNN, CC, and MLPs gave a good prediction of the MAb and viable cells. Figure 16 and 17 show results of prediction of MAb using a CC and a MLP network. The prediction of other variables such as ammonia concentration also was good. The prediction of glutamine concentration was not very good as shown in Figure 18. This may be due to the fact that glutamine concentrations are not well correlated by the selected input variables. Further studies are needed to verify this anomaly.

8. Conclusions. Neural networks are suitable for modeling biological systems. Before using neural network models, one must be careful about the scope of the modeling exercise. Careful selection of data are needed for training the chosen networks. The data set should be sufficiently rich to excite different *modes* of the bioprocess studied. A compromise must

be made with respect to the number of parameters, size of the data set, and the accuracy desired. In the three case studies presented in this paper, it was found that RNN is well suited for "generalization." Local networks such as RBFN should be used with caution as they perform poorly when extrapolation is required. In general, global networks such as RNN, MLP, and Cascade Correlation have been successfully employed in modeling *Z. mobilis*, recombinant *E. coli*, and hybridoma cell culture processes in batch and fed-batch operations.

Acknowledgements. The corresponding author wishes to thank the International Center for Cooperative Research in Biotechnology, Osaka University, for the financial support received during his stay in Osaka, Japan, where most of this research was undertaken. Additional financial support was provided by a National Science Foundation (USA) grant.

REFERENCES

1. Halme, A., A. Visala, and J. Kankare. 1992. *Functional state concept in modeling biotechnological processes*. In *Modeling and Control of Biotechnical processes* (M. N. Karim and G. Stephanopoulos, Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 153-158.

2. Yoshida, T., K. B. Konstantinov, W. Ruenglertpanyakul, and R. M. Matanguihan. 1992. *On-line monitoring of the physiological state of the cell population in a bioreactor*. In *Modeling and Control of Biotechnical Processes* (M. N. Karim and G. Stephanopoulos, Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp 15-20.

3. Karim, M. N. and S. L. Rivera. 1992. *Artificial neural networks in bioprocess state estimation*. Adv. Biochem. Engr. Biotechnol. 46:1-22.

4. Karim, M. N. and A. Halme. 1988. *Reconciliation of measurement data in fermentation using an on-line expert system*. Proc. IFAC Symp. Comp. Appli. Fermentation Tech., Cambridge, United Kingdom. pp. 37-46.

5. Shuler, M. L. and F. Kargi. 1992. *Bioprocess Engineering: Basic Concepts*. Prentice Hall, Englewood Cliffs, New Jersey.

6. Shuler, M. L. 1985. *On the use of chemically structured models for bioreactors*. Chem. Engr. Commun. 36:161-189.

7. Ljung, L. 1992. *System Identification: Theory for the User*. Prentice Hall, Englewood Cliffs, New Jersey.

8. Wiechert, W., T. Honer, C. Hausmann, M. Mollney, and M. Kinder-Theissen. 1992. *Modeling of bioprocesses within a network transformation framework: A software engineering concept*. In *Modeling and Control of Biotechnical Processes* (M. N. Karim and G. Stephanopoulos, Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 441-444.

9. Rivera, S. L., R. Bajaj, and M. N. Karim. 1993. *On-line model parameter estimation of a bioprocess using genetic algorithms*. AIChE Annual Meeting, St. Louis, Missouri.

10. Fahlman, S. E. and C. Lebeire. 1990. *The cascade correlation learning architecture*. In *Advances in Neural Information Processing System* (D. S. Touretzky, Ed.), Morgan Kaufmann, Germany.

11..Eikens, B. and M. N. Karim. 1994. *Real-time control of a waste neutralization process using radial basis functions*. Proc. ADCHEM'94 Conf., Kyoto, Japan. pp. 125-130.

12. Saucedo, V. M., B. Eikens, and M. N. Karim. 1994. *Identification techniques for a recombinant fed-batch fermentation for ethanol production*. In *Advances in Bioprocess Engineering* (E. Galindo and O. T. Ramirez, Eds.), Kluwer Academic Press.

13. Chen, S. and S. A. Billings. 1989. *Representation of nonlinear systems: The NARMAX Model*. Intl. J. Control. 51:1013-1032.

14. Brockwell, P. and R. A. Davis. 1992. *Time Series: Theory and Methods*, 2nd edition. Springer-Verlag, New York, New York.

15. Saner, U. and S. Stephanopoulos. 1992. *Application of pattern recognition techniques to fermentation data analysis*. In *Modeling and Control of Biotechnical Processes* (M. N. Karim and G. Stephanopoulos, Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 123-128.

16. Simutis, R., I. Havlik, and A. Lubbert. 1992. *Process state estimation and prediction in a production-scale beer fermentation using fuzzy aided extended Kalman filter and neural networks*. In *Modeling and Control of Biotechnical Processes*.(M. N. Karim and G. Stephanopoulos Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 95-100.

17. Uozumi, N., T. Yoshino, S. Shiotani, K. I. Suehara, F. Arai, T. Fukuda, and T. Kobayashi. 1993. *Application of image analysis with neural network for plant somatic embryo culture*. J. Fermentation and Bioengr. 76(6):505-509.

18. Linko, P. and Y. H. Zhu. 1991. *Neural network programming in bioprocess variable estimation and state prediction*. J. Biotechnol. 21:253-270.

19. Shi, Z. P. and F. Shimizu. 1992. *Fuzzy control of fed-batch fermentation with the aid of neural networks*. In *Modeling and Control of Biotechnical Processes*.(M. N. Karim and G. Stephanopoulos Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 167-172.

20. Unger, L. H. 1990. *A bioreactor benchmark for adaptive network-based process control.* In *Neural Networks for Control* (R. S. Sutton Miller, W. T. Webros, and P. J. Webros, Eds.), MIT Press, Cambridge, MA. pp. 387-397.

21. Ruenglertpanyakul, W., K. B. Konstantinov, and T. Yoshida. 1992. *Application of neural networks to variable estimation and stage identification in phenylalanine production.* In *Modeling and Control of Biotechnical Processes* (M. N. Karim and G. Stephanopoulos, Eds.), Pergamon Press Ltd., Headington Hill Hall, Oxford, United Kingdom. pp. 429-432.

22. Cybenko, G. 1989. Mathematics of control. Signal and Systems. 2:303-315.

23. Oh, G. S., B. Eikens, T. Yoshida, and M. N. Karim. 1995. *Neural networks in estimation and control of antibody production using hybridoma cells in fed-batch cultures.* 6th Intl. Conf. Comp. Applic. Biotech. Garmisch-Partenkirchen, Germany. pp. 14-17.

24. Rivera, S. L. and M. N. Karim. 1990. *Application of dynamic programming for fermentative ethanol production.* Proc. American Control Conf., San Diego, California. pp. 2144-2149.

25. Hilaly, A. K. 1992. *Structured Mathematical Modeling and Optimization of Xylose Fermentation by a Recombinant* Escherichia coli. Ph.D. Dissertation, Colorado State University, Fort Collins.

26. Ingram, O., T. Conway, P. Clark, W. Sewell, and J. Preston. 1988. *Genetic engineering of ethanol production in* Escherichia coli. Appl. Environ. Microbiol. 54(3):397-403.

27. Karim, M. N. and B. Eikens. 1995. *Global versus local neural networks in identification and control: A case study of the waste water neutralization process.* In *Neural Networks for Chemical Engineers* (A. Bulsari, Ed.), Elsevier Science, Amsterdam, Netherlands.

RECONOCIMIENTO DE FONEMAS MEDIANTE EL USO DE REDES NEURONALES

B. Mora, O. Gualdrón y Y. Torres
Laboratorio de Óptica y Tratamiento de Señales (LOTS)
Universidad Industrial de Santander, Bucaramanga

Abstract. Today is being studied the application of new methods in speech recognition, such as Artificial Neural Networks, supported in digital processing technics as spectral treatment. The present work uses the Fourier Spectral Representation and Speech Digital Filtering after being digitized with a sound card. This representation is obtained from short time signal intervals taking as base on the quasi-stationary principle, isolating them with a Hamming Window. By reducing the data to quantized small vectors, the labeled information is presented in three Self-Organizing Neural Networks, those which through a training process can classify phonetic groups in a two-dimensional space, corresponding to the spanish language phonemes.

Resumen. Hoy en día se estudia la aplicación de nuevas metodologías para el reconocimiento de voz, tales como las redes neuronales artificiales, apoyadas en técnicas sencillas de procesamiento digital como el tratamiento espectral. El presente trabajo, emplea la representación espectral de Fourier y el filtrado digital de la voz, antes digitalizada con una tarjeta de sonido. Esta representación se obtiene a partir de intervalos cortos de señal basándose en el principio de cuasiestacionariedad, aislándolos con una ventana de Hamming. Reduciendo los datos a vectores con un número bajo de componentes, se presenta la información etiquetada a tres redes neuronales autoorganizativas, las cuales mediante un proceso de entrenamiento se encargan de discriminar grupos fonéticos (fonemas del idioma español) en un espacio bidimensional.

1. Introducción. La voz es un proceso estocástico muy complicado. Desde el punto de vista acústico, incluso en voz clara del mismo locutor, las distribuciones espectrales se superponen y tienen una forma muy compleja. Las funciones estadísticas de densidad no son gaussianas, luego no pueden ser aproximadas analíticamente, además que el espectro acústico de un fonema varía en el contexto de otros fonemas.

Hoy en día, se estudia la aplicación de nuevas metodologías tales como las *redes neuronales artificiales* (RNA) al reconocimiento de voz, apoyada en técnicas sencillas de procesamiento digital como el tratamiento espectral.

Este trabajo emplea la representación espectral de la señal temporal, la cual ha sido digitalizada utilizando una tarjeta de sonido. Se aplican métodos clásicos de la teoría de señales, tales como la transformada de Fourier y el filtrado digital, sin tener en cuenta las propiedades estadísticas especiales de las señales naturales, ya que el objetivo primordial es identificar los fonemas de una pronunciación y no la identidad de un locutor en particular.

La representación espectral se obtiene a partir de intervalos pequeños de señal, basándose en el principio de cuasiestacionariedad, el cual asume que las señales vocales son estacionarias en intervalos cortos de tiempo.

Con el fin de reducir la cantidad de datos a emplear en la etapa de aprendizaje y reconocimiento de la RNA, las distribuciones de frecuencia resultantes se agrupan en vectores con un número bajo de componentes.

Los vectores espectrales son utilizados como entradas a RNA con entrenamiento no supervisado llamada *mapa auto-organizativo* o *red de Kohonen*, la cual por medio de un proceso de entrenamiento se encarga de organizar clases diferentes en un espacio de dos dimensiones. Este espacio bidimensional o mapa de Kohonen, corresponde a la capa de salida, en la cual para cada vector de entrada solo es activado un elemento o nodo, identificando así los fonemas del idioma español.

Los sistemas de reconocimiento de voz varían en su complejidad y en sus capacidades. Básicamente hay tres características involucradas, y cada una de estas varía de relativamente fácil a relativamente difícil. Estas características incluyen locutor *independiente* o locutor dependiente, vocabulario *completo* o *vocabulario limitado* y *palabras aisladas* o discurso continuo

El presente trabajo es base fundamental para un sistema de reconocimiento de palabras aisladas del tipo multilocutor con un vocabulario no muy largo de palabras (como se subrayó). El número de palabras no es (aparentemente) el obstáculo principal, ya que el objetivo es la identificación de los fonemas que las componen.

2. Estructura del lenguaje. La mínima unidad acústica del lenguaje hablado se llama fonema, expresado en los sonidos articulados por la lengua y representado por las letras del alfabeto. En español, lo normal es

que cada letra corresponda a un solo fonema. Sin embargo, no siempre se da esta correspondencia; es por esto que existen 24 fonemas (Fig. 1) y 28 letras[3].

La fonología distingue dos clases de fonemas: los vocálicos y los consonánticos; pero para efectos de reconocimiento de voz se aprecian tres categorías acústicamente independientes basadas en su espectro de potencia: los fonemas vocálicos, fricativos y los explosivos.

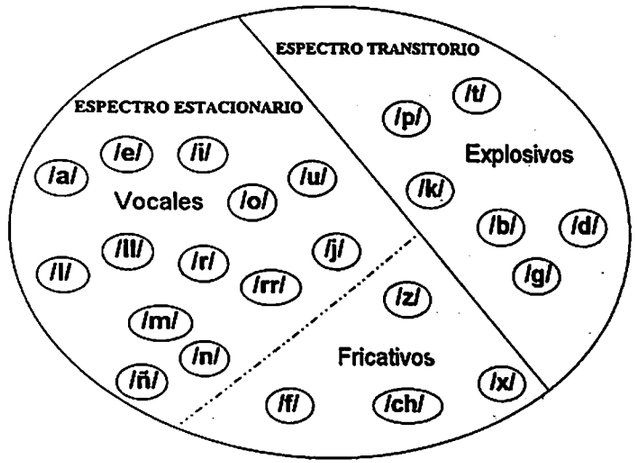


Figura 1. Fonemas del Idioma Español. Algunos fonemas no tienen correspondencia directa con las letras que representan; por ejemplo, /z/ representa la letra Z, S en todas las ocurrencias y la letra C en las sílabas CE, CI. La C ante la A, O, U (CA, CO, CU), está representada por el fonema /k/.

Los dos primeros grupos están bien definidos debido a que su espectro es estacionario, pero los explosivos son identificables solamente con base en las propiedades transitorias de la señal. De allí que como se mencionará posteriormente, se deben crear Redes Neuronales adicionales para reconocer este grupo.

El habla es un proceso estocástico muy complejo y a pesar de que se tiene una impresión contraria, ni los fonemas, ni las sílabas, ni siquiera las palabras, constituyen elementos discretos que se puedan separar fácilmente de forma automática. No existen pausas entre estos elementos, y además se influyen unos a otros debido a los movimientos articulatorios del aparato fonador (efecto de coarticulación)[2].

3. Captura y delimitación de las señales de voz. Las pronunciaciones se almacenaron en forma digital, utilizando las facilidades de *Multimedia Windows*. La interfaz de alto nivel es llamada MCI (*Media Control Interface*) y usa archivos en forma de bloques de memoria para grabar y reproducir señales de Audio. Los archivos generados tienen la extesión WAV y se conocen como archivos de sonido[7].

La tarjeta digitalizadora de sonido utilizada fue una *Sound Blaster Pro*. El modo escogido para grabar las pronunciaciones fue el tipo *Monoaural*. Entre las tasas de muestreo a seleccionar (11025, 22050 y 44100 Hz) se consideró que 11025 Hz era más que suficiente (en sonido telefónico se toman generalmente 3500 Hz), con una precisión de 8 bits por dato muestreado.

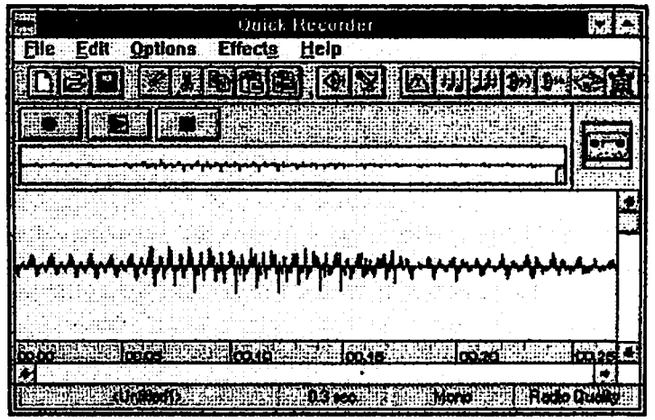


Figura 2. Captura de la pronunciación "monas"

Los archivos WAV se crearon utilizando el programa *Quick Recorder* (Fig. 2). Se capturaron en total 150 pronunciaciones de palabras cotidianas, procurando que en su conjunto cubrieran todas las ocurrencias de los fonemas. Este conjunto de pronunciaciones constituye la muestra.

Se eliminaron de las palabras almacenadas, las señales diferentes a voz asociadas a silencio o ruido de fondo al comienzo y final de la grabación. Para este efecto se implementó el algoritmo de Rabiner y Sambur[8], el cual opera con base en dos mediciones hechas a la señal capturada: su función de energía y su función de densidad de ceros.

4. Análisis Espectral de la señal de voz. En muchos sistemas de reconocimiento de voz, el procesamiento acústico codifica las señales de voz en coeficientes LPC (*Linear Predictive Code*)[1], los cuales contienen aproximadamente la misma información que una descomposición espectral. Sin embargo, el análisis del espectro en intervalos cortos de tiempo ha sido tradicionalmente la más importante técnica de procesamiento de voz. Recordando, la voz en términos generales no es estacionaria pero sobre un intervalo de tiempo suficientemente corto se puede considerar que sí lo es; así la transformada discreta de Fourier (DFT)[10 , 12] de un corto segmento de voz, es muy efectiva.

La FFT (algoritmo eficiente para la evaluación de la DTF) en un tiempo corto se define como :

$$X_\ell(w)=\sum_{n=0}^{N-1}X_\ell(n)W(n)\exp(-jw_k n)$$

donde

$$X_\ell(n)=X(n+1),\quad n=0,1,\ldots,N-1$$

$$\ell=0,L,2L,3L,\ldots$$

y

$$w_k=\frac{2\pi k}{M},\qquad k=0,1,\ldots,M-1.$$

Esta ecuación se interpreta como la transformada de Fourier de M ejemplos de longitud ℓ, ponderados por una ventana $W(n)$. La ventana utilizada se llama una *ventana de Hamming.*, implementada como un filtro temporal digital pasabanda, el cual aísla y suaviza el intervalo temporal en cuestión.

La FFT se evaluó sobre 256 puntos de señal temporal, con una ventana de Hamming de la misma longitud. Se calculó la transformada cada 11.6 milisegundos (el intervalo entre 10 y 25 ms es satisfactorio

para transmitir o almacenar el espectro, brindando una representación confiable de la voz[12]), lo cual indica que se utilizó un traslape de 128 muestras entre dos conjuntos consecutivos de datos, dado que los 256 puntos representan 23.2 ms de señal.

El espectro de Fourier (se tomó el módulo de los primeros 128 puntos de la FFT; Fig. 3) se distribuye en un rango de 0 a 5500 Hz, correspondiente a la capacidad óptima vocal y auditiva humana; de tal manera que existe una correspondencia directa entre esta tasa de muestreo y las características naturales. Esto corrobora la adecuada frecuencia de muestreo seleccionada (además, se escogió 11025 Hz debido a la poca cantidad de datos muestreados).

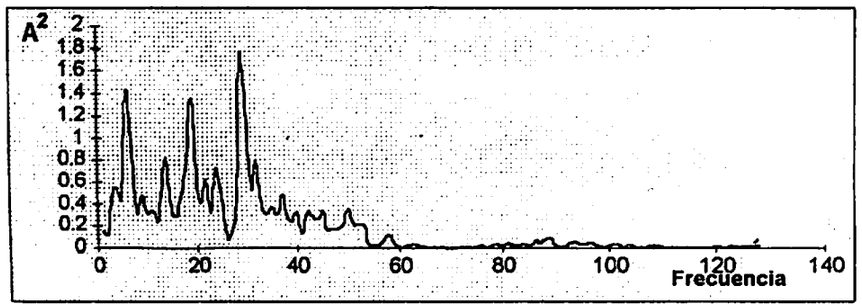


Figura 3. Representación espectral de Fourier del fonema /a/, aplicando la ventana de Hamming.

El valor real de la frecuencia f_r se halla empleando la siguiente expresión:

$$f_r = \frac{k}{M} f_c$$

donde k es el valor en el eje frecuencial, M el número de puntos de la FFT y f_c la tasa de muestreo.

5. Filtrado digital. Para el procesamiento digital de la voz se utilizan especialmente los *filtros digitales recursivos* o IIR[12] (respuesta impulsional de duración infinita). El filtro digital representativo de la familia de los recursivos, se denomina *filtro digital elíptico de tipo pasa bajo*[12].

El diseño de este filtro se llevó a cabo utilizando el *software* implementado en el trabajo de grado desarrollado en la UIS por [2].

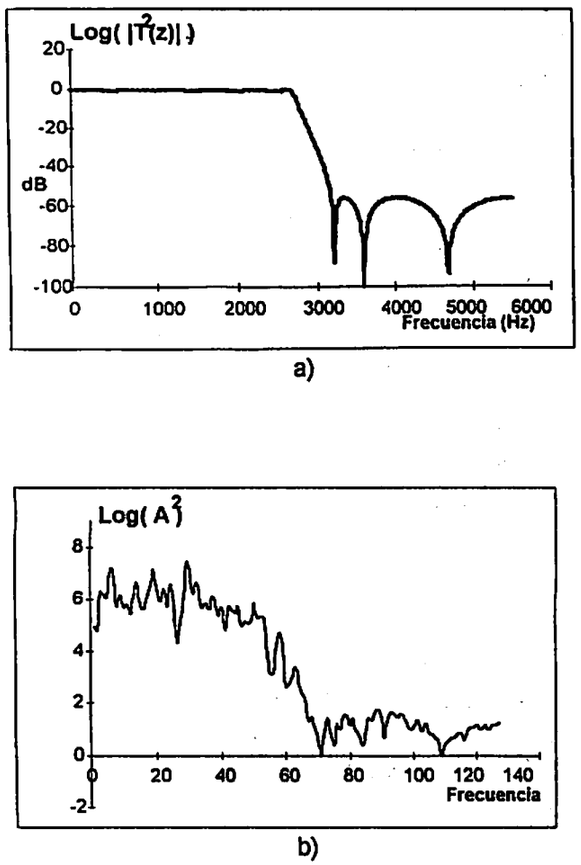


Figura 4. Filtro Digital Elíptico pasa bajo de orden 6 (a) y efecto de aplicar logaritmo y filtrado sobre el espectro mostrado en la figura 6 (b).

Antes de realizar el filtrado se aplicó un logaritmo natural a la señal frecuencial, especialmente para resaltar las frecuencias altas. Luego se aplica el proceso inverso de atenuación con el filtro digital.

En resumen, la distribución de frecuencias se lleva a la forma logarítmica y se atenúa nuevamente aplicando el filtro digital (Fig. 4). Este proceso aparte de suavizar el espectro (eliminando ruido), prepara la señal frecuencial para el siguiente proceso de canalización espectral que se describirá a continuación.

6. Vectores cuantizados. Es necesario reducir los datos a emplear en la etapa de aprendizaje y reconocimiento de la red neuronal, con el fin de hacer eficiente el sistema neuronal. El espectro mencionado se agrupó en un nuevo vector real característico de dimensión dieciséis en un espacio euclidiano. Cada valor del vector resultante representa la suma ponderada de la amplitud de las bandas de frecuencias, cuyos límites se determinaron según el criterio descrito en [11].

Después de la integración del espectro, se restó a cada vector la media de todos sus componentes y se normalizó cada uno a una amplitud constante. Esto se realizó con el fin de que todos los vectores en los que se dividen las pronunciaciones, tengan como media cero, hallándose un vector desviación normalizado. La presentación de un conjunto bipolar de entrenamiento a la red neuronal se refleja típicamente en un mejor desempeño.

El espectro de los respectivos fonemas de la voz, ocupa diferentes regiones de este espacio de dimensión dieciséis, así que ellos pueden ser clasificados con algún método de discriminación óptima[6]. Esto garantiza que el vector desviación normalizado característico elegido es apropiado para la clasificación y posterior reconocimiento de los fonemas.

Los vectores espectrales cuantizados son llamados cuasifonemas (Fig. 5). Esta denominación se deriva del pequeño intervalo de tiempo que representa cada uno. Se toman 23 ms de pronunciación, en contraste con la duración de un fonema que varía en el rango de 40 a 400 ms. La unión secuencial de varios cuasifonemas representan un fonema.

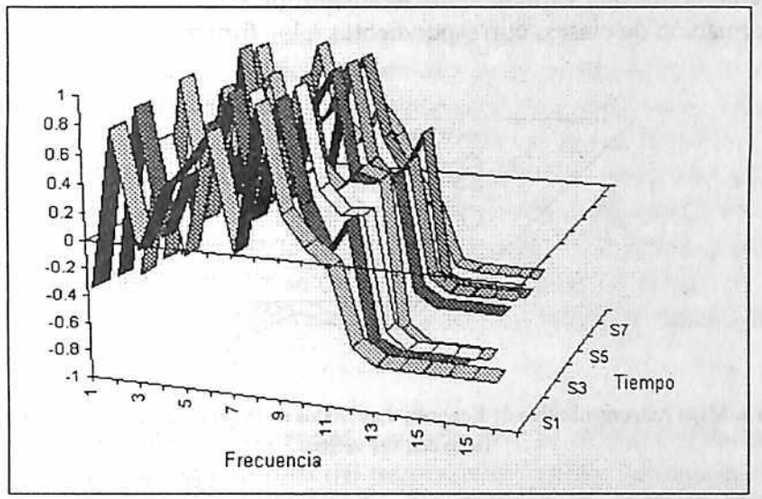


Figura 5. Vectores de cuantización de 184 ms de pronunciación del fonema /a/. Cada valor del eje S corresponde a 23 ms.

7. Red Neuronal: Los Mapas Autoorganizativos de Kohonen (SOM). El SOM es una Red Neuronal Artificial de entrenamiento no supervisado, con un funcionamiento similar al desempeño cerebral. Está compuesta de dos capas; y mapea un espacio de entrada n-dimensional, sobre un espacio de dos dimensiones, mientras se mantienen las relaciones espaciales entre los elementos vecinos de la red formada. Todos los elementos de entrada están completamente conectados con la capa de salida (Fig. 6), por una conexión (sinapsis) con un peso asociado. Los nodos de salida están extensiva y localmente interconectados.

Los vectores de entrada se presentan secuencialmente en el tiempo sin especificar la salida deseada (aprendizaje no supervisado). A medida que la entrada ingresa a la red, los pesos forman cúmulos o vectores centrados que asemejan el espacio de entrada, tal que la función de densidad de puntos de los vectores centrados, tiende a aproximar la función de densidad de probabilidad (PDF) de los vectores de entrada. En resumen, los pesos serán organizados de tal manera que los nodos de la capa de kohonen son topológicamente sensibles a las entradas que son fisicamente similares. Así, los nodos de salida serán ordenados de una manera natu-

ral, siendo esta, una característica altamente deseable para el proceso de autoformación de clases, correspondientes a los fonemas en cuestión.

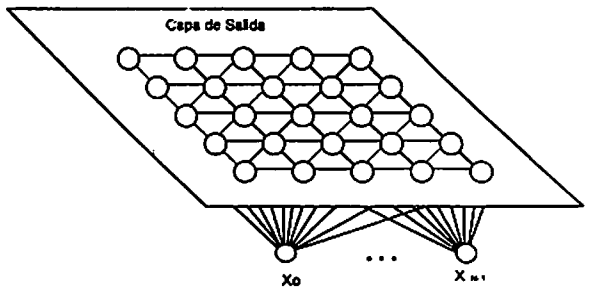


Figura 6. Mapa Autoorganizativo de Kohonen. Los Nodos de la capa de salida están interconectados con sus vecinos.

8. Ambiente Experimental. Para la adquisición de datos se contó con un micrófono, una tarjeta de sonido Sound Blaster Pro como convertidor Análogo/Digital y un microcomputador 486 DX2 a 66 Mhz, video SVGA de 1024x768 puntos, DD de 500 Mb y 32 Mb en RAM. Un PC con características modestas proporcionaría un buen desempeño, dado que la rata de muestreo (11025Hz) y la precisión (8 bits) seleccionados, no exigen mayor almacenamiento (por cada segundo de pronunciación 11Kb son necesarios).

El trabajo se desarrolló bajo la plataforma *Windows*. Las señales verbales se capturaron con el programa *Quick Recorder* (Fig. 2), el cual permite que el usuario escoja las características de captura, además de editar los archivos generados.

Para el procesamiento digital se seleccionó el programa *MathLab Windows*, herramienta matemática a manera de intérprete que contiene un grupo de funciones y permite añadir otras para el manejo de los archivos WAV, visualización de gráficas y procesamiento matemático, incluyendo FFT.

El simulador de Redes Neuronales empleado se llama *Explorenet*, el cual tiene implementados más de 10 paradigmas de RNA. Una característica importante es poder definir un diseño de RNA como un conjunto de objetos (incluyendo la misma RNA) con las relaciones (mapeo: definición del flujo de información) entre ellos. Entre los módulos más utilizados incluyen, archivos de entrada y salida, formatos, procesamiento de datos, RNA, imágenes, gráficas de barras y gráficas de dispersión. El módulo de redes neuronales es el más importante, ya que se encarga de definir el paradigma neuronal a utilizar, el diseño de la red (entradas, capas ocultas y nodos de salida) y el entorno de simulación. Para usar este módulo, primero se debe especificar cuál es el modelo a utilizar (SOM, BPN, HOP, SPR, ..)[5], para luego cargar la red usando las constantes en tiempo de cargue (en las cuales se definen los parámetros de tamaño de la red y frecuencias iniciales de aprendizaje) y las constantes en tiempo de ejecución.

9. Diseño del Sistema Neuronal. Los vectores característicos de 16 bandas espectrales, producto final del procesamiento espectral, son utilizados como entrada a la RNA. El SOM por medio de un proceso de entrenamiento se encarga de organizar clases diferentes de discriminación en un espacio bidimensional.

Para los fonemas con espectro estacionario (los vocales y fricativos), se creó una RNA con 16 entradas y un mapa rectangular (capa de salida) de 8 filas por 12 columnas.

Para los fonemas explosivos asociados al espectro transitorio, se crearon dos RNA cada una con 16 nodos en la capa de entrada y un mapa de 3x3 como capa de salida. Cada red se diseñó para identificar los fonemas /k/, /p/, /t/ y /b/, /d/, /g/ respectivamente.

Número de Red	Dimensión Entrada	Tamaño Mapa	Fonemas que Discrimina
SOM 1.	16	8x12	/a/ /e/ /i/ /o/ /u/ /m/ /n/ /ñ/ /j/ /l/ /ll/ /r/ /rr/ /f/ /z/ /ch/ /x/ /y/
SOM 2.	16	3x3	/k/ /p/ /t/
SOM 3.	16	3x3	/b/ /d/ /g/

Tabla 1. Redes Neuronales entrenadas para el reconocimiento de fonemas.

El tamaño óptimo de los mapa se determinó sobre la base de varios entrenamientos. Para el principal, se tomó 8x12 por que a pesar de

que las clases discriminadas están más pegadas en el mapa, ellas se distribuyen casi uniformemente en el área rectangular.

10. Entrenamiento. Se realizó sobre dos muestras (una de palabras aisladas y otra de fonemas), digitalizadas en la etapa inicial (Fig. 2). La primera muestra contiene 150 palabras cotidianas, las cuales en conjunto ocupan 2 minutos de pronunciación y 1.3. Mb de almacenamiento en disco. El procesamiento acústico reduce todos los datos a 10500 vectores característicos de 16 posiciones cada uno.

La segunda muestra está conformada por un conjunto de fonemas, tomados de la primera muestra y aislados manualmente. Cada fonema se tomó de 10 contextos diferentes de las señales. Todos los fonemas juntos ocupan 20 segundos de pronunciación, y con el procesamiento acústico se redujeron a 1600 vectores característicos.

La red neuronal diseñada para reconocer los 18 fonemas fricativos y vocales, llamada SOM1 (tabla 1), se entrenó con la primera muestra. Cada vector fue presentado a la red 8 veces, y en total se entrenó la RNA con 80000 iteraciones (el entrenamiento de la RNA demora aproximadamente 30 minutos).

Las dos redes restantes SOM2 y SOM3 (tabla 1) diseñadas para reconocer los fonemas explosivos, se entrenaron de manera similar, con los fonemas explosivos conocidos, tomados de la segunda muestra.

Los mapas se calibraron con los mismos vectores de la segunda muestra. El proceso de calibración consiste en identificar la respuesta del mapa de Kohonen para un grupo de vectores, correspondientes al mismo fonema. De manera similar se hace para todo los fonemas. Al final de la calibración se conoce qué posición del mapa corresponde a cada fonema.

11. Resultados. La figura 7 describe todo el procesamiento de la voz desde el momento en que es pronunciada ante un micrófono, hasta la identificación de los fonemas que componen dicha pronunciación.

Cuando haya concluido el procesamiento acústico para una pronunciación en particular, cada vector característico (que representa un cuasifonema), corresponde a la entrada que pasa por las tres redes neuronales. En cada red se activa una sola neurona ganadora, y entre los tres

nodos ganadores se escoge aquel cuya respuesta en el mapa fonético represente una mayor grado de similitud con la clase patrón (menor distancia euclidiana), correspondiente al fonema asociado en la etapa de entrenamiento.

El criterio que se usó para identificar la mayoría de los fonemas está basado en el grado de estabilidad de los cuasifonemas, es decir, si las respuestas de la RNA se mantienen constantes para el conjunto secuencial de entradas, dado que la mayoría de los fonemas tienen un único estado estacionario.

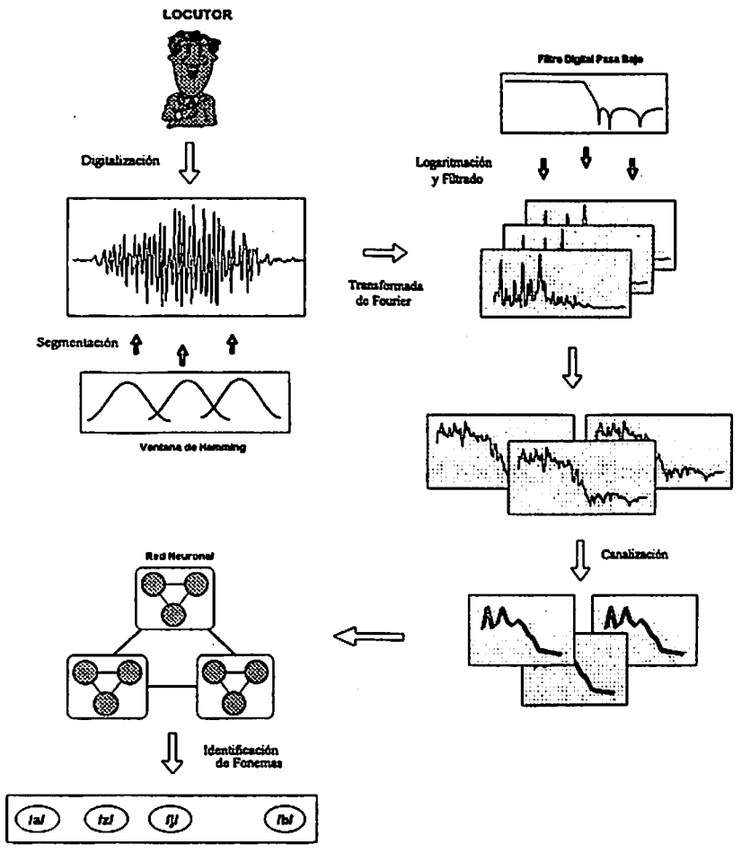


Figura 7. Representación general del proceso de Reconocimiento de Fonemas con Redes Neuronales.

En el caso de los fonemas explosivos no se podría esperar que la respuesta a un conjunto de vectores del mismo fonema fuera constante, ya que ellos se localizan sólamente en los estados transitorios de la señal sonora. Aún así, dado que el entrenamiento de las redes neuronales auxiliares se realizó con un conjunto selecto de fonemas conocidos, tomando varios estados de la señal, se puede considerar que son estacionarios en ciertos intervalos de tiempo.

Los siguientes gráficos muestran los mapa fonéticos del sistema neuronal diseñado. Se observa que los fonemas están bien diferenciados y organizados de tal manera que sus vecinos son los más parecidos espectralmente.

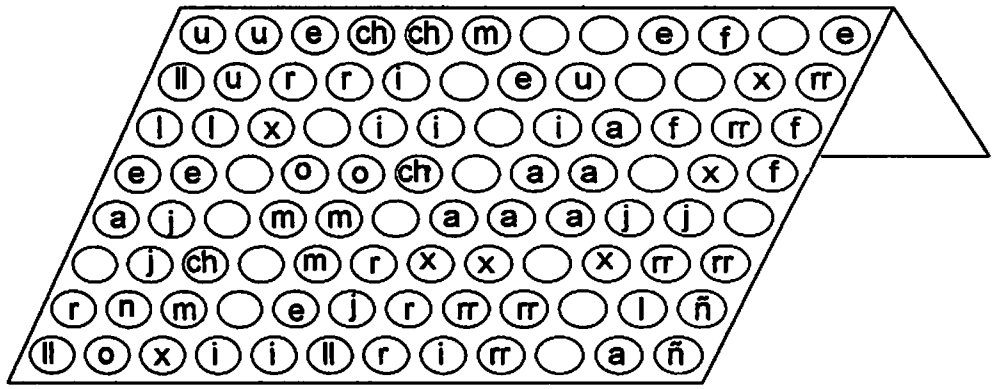


Figura 8. Mapa fonético neuronal para la identificación de los fonemas vocales y fricativos, como resultado del proceso de entrenamiento descrito. Los nodos blancos responden a ningún o más de un fonema.

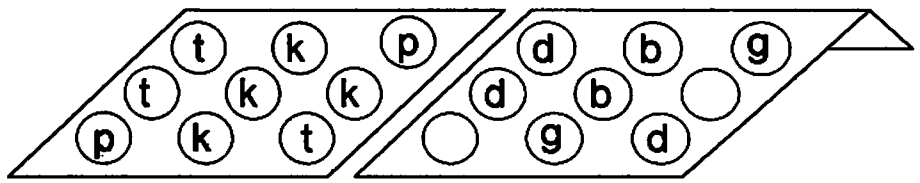


Figura 9. Mapas fonéticos resultantes para la identificación de los fonemas explosivos.

Los mapas fonéticos obervados, constituyen la meta final del trabajo descrito. Se intentó mostrar la aplicación de una manera ágil y sencilla, de las

técnicas clásicas de tratamiento de señales, para combinarlas con tecnologías actuales como lo son las Redes Neuronales Artificiales, en aras de resolver problemas que requieren un tratamiento especial.

12. Conclusiones y Recomendaciones. La Transformada Rápida de Fourier es una herramienta muy efectiva en el procesamiento de la voz para intervalos cortos de tiempo, ya que es exacta, rápida y refleja la formación de grupos consonánticos, característica deseable para el posterior proceso neuronal.

No existen diferencias marcadas en cuanto a exactitud de reconocimiento entre los fonemas estacionarios y transitorios, dado el carácter adaptivo de las redes neuronales que se encargan de eliminar dichas diferencias. La exactitud del reconocimiento depende básicamente de una buena delimitación de la señal real de voz en una pronunciación y posterior filtrado para eliminar el ruido introducido en el proceso inicial de digitalización.

Las Redes Neuronales Autoorganizativas constituyen un modelo muy efectivo en el problema de reconocimiento de la voz, ya que se encargan de formar clases representativas correspondientes a los fonemas en un espacio bidimensional, a partir de un conjunto reducido de vectores de entrenamiento.

El presente trabajo es la base para una gama de aplicaciones en el reconocimiento de voz, en las cuales su estudio se centraría en la interpretación de los mapas fonéticos. La interpretación se refiere a la identificación de los fonemas en la ruta que tome dentro del mapa, una secuencia de vectores espectrales (cuasifonemas). La unión de todos los fonemas de una manera secuencial, correspondería a la pronunciación de entrada.

La aplicación más inmediata sería un sistema de reconocimiento de palabras con un vocabulario largo. Para la interpretación del mapa se podrían construir un conjunto de reglas gramaticales que identifiquen todas las variaciones de los fonemas en el contexto de otros fonemas. Las reglas gramaticales resultarían sencillas debido a que en el español la ortografía es siempre idéntica a la transcripción fonética. Esta reglas se pueden construir utilizando principios de inteligencia artificial tal como una base

de conocimientos, o aplicando nuevamente las redes neuronales, especialmente las Memorias Asociativas [5].

De otra parte, el reconocimiento de locutores nuevos se podría lograr pronunciando unas 200 palabras y ajustando el mapa con un reentrenamiento adecuado.

BIBLIOGRAFÍA

[1] Beleño L., Ojeda, V. *Diseño de un Modelo Gráfico de Voz para Personas con Problemas de Lenguaje.* Proyecto de Grado LOTS, UIS . 1992

[2] Casacuberta F., Viadal L.E. *Reconocimiento Automático del Habla: Metodologías y Arquitecturas.* Editorial Marcombo, Pág. 157-167, 1987.

[3] Celis V., Rey N., Rodriguez C. *Español sin Fronteras, Grado 7°.* Editorial Voluntad, 1988.
]

[4] Ibarra A., Medina A. *Desarrollo de Software Especializado en el Diseño e Implementación de Filtros Digitales.* Proyecto de Grado UIS, 1993.

[5] Lippman R. *An Introduction to Computing with Neural Nets.* IEEE ASP Magazine, p.p. 4-22, April,1987.

[6] Kohonen T. *The Neural Phonetic Typewriter.* Computer, p.p. 11-22, March, 1988.

[7] Petzold C. *Storing Sound: A Look at Waveform Audio Sound Files.* PC Magazine, p.p. 369-374, february, 1992.

[8] Rabiner L.R., Sambur M.R. *An Algorithm for Determining the Endpoints of Isolated Utterances.* The Bell System Technical Journal, p.p. 297-315, february, 1975.

[9] Rabiner L.R., Gold B. *Theory an Application of Digital Signal Processing.* Prentice Hall; 1975.

[10] Shafer R., Rabiner L.R. *Digital Representatios of Speech Signals.* Proc. IEEE, Vol 63, No. 4, april 1975.

[11] Waibel A., Hampshire J. *Building Blocks for Speech.* Byte, p.p. 235-242, august, 1989.

[12] Witten I.H. *Principles of Computer Speech.* Academic Press, 1982. [12]

AUTÓMATAS CELULARES Y FÍSICA DIGITAL

José Daniel Muñoz Castaño*
Departamento de Física
Universidad Nacional de Colombia - Sede Bogotá
Ciudad Universitaria, Bogotá, Colombia, Sur América

Resumen: Se presentan los autómatas celulares como una alternativa digital para construir modelos en física y en otras ciencias. Primeramente se define qué es un autómata celular, se muestran sus características y se describen algunas de sus propiedades básicas. A continuación se revisan algunas de sus aplicaciones, primero a la física, con ejemplos de la dinámica de fluidos y de los sistemas de Ising, y luego a la biología y a las ciencias de la computación. Finalmente, se discute la posibilidad y las ventajas comparativas de utilizar los autómatas celulares como estructuras alternativas para construir modelos y expresar leyes científicas en forma digital directa, sin pasar por los modelos continuos usuales.

Abstract: Cellular Automata, as a digital alternative to build models in physics and in other sciences, are introduced. First, we define what a cellular automata is, and show its characteristics and some of its basic properties. Next, some of its applications are reviewed, fisrt in physics, with examples in fluid dynamics and Ising systems, and next in biology and computer sciences. Finally, we discuss the posibility and comparative adventages we could achieve if we use cellular automata as an alternative structure to buid models and to express scientific laws in digital direct manner without using usual continuous models.

1. Introducción. La mayoría de los modelos de la física y de otras ciencias se construyen utilizando expresiones de funciones continuas de números reales o complejos, como, por ejemplo, ecuaciones diferenciales, integrales funcionales, etc., a los que llamamos modelos analógicos. Pero muchas veces estos modelos no se pueden resolver exactamente; se recurre entonces a métodos aproximativos que con frecuencia se implementan en el computador, con los consiguientes problemas de digitalización del modelo. ¿Será posible construir modelos y leyes de la

* E-mail:jdmunoz@hemeroteca.icfes.gov.co

física utilizando expresiones digitales? Tales modelos se podrían implementar directamente en el computador, evitando los problemas de digitalización, y serían una herramienta alternativa excelente para aquéllos casos en que los modelos analógicos son difíciles de trabajar. En esta dirección, los autómatas celulares aparecen como una buena posibilidad.

El objetivo de este artículo es presentar a los autómatas celulares como una estructura alternativa idónea para construir modelos e incluso expresar leyes de la física y de otras ciencias en una forma directamente digital, sin pasar por los modelos analógicos convencionales. Con este fin se describe en primer lugar qué es un autómata celular, y cuáles son algunas de sus propiedades básicas. A continuación se ilustra con algunos ejemplos el uso que se le puede dar a los autómatas celulares para construir modelos y expresar leyes tanto en física como en biología y en ciencias de la computación. Los modelos se han escogido como representativos de diferentes paradigmas de modelado en autómatas celulares, y por lo tanto se espera que aporten una visión bastante amplia de las técnicas que se pueden emplear y de los problemas que se pueden tratar con ellos. Finalmente, se propone y se discuten las ventajas de utilizar autómatas celulares para expresar leyes y construir modelos en física y en otras ciencias, y de esta forma resolver algunos de los inconvenientes planteados en esta introducción.

2. ¿Qué es un autómata celular? Un autómata celular (digamos bidimensional), se puede ver de la siguiente forma: se toma el plano y se divide como un embaldosado homogéneo en celdas, a lo que llamamos una *teselación homogénea*. Cada celda puede estar en uno de un conjunto finito o numerable Σ de *estados*. El haberle asignado un estado a cada celda del autómata es lo que llamamos una configuración C. De otra parte cada celda escoge un conjunto de celdas , indicando sus posiciones relativas con respecto a sí misma, al que llama su vecindad. Todas las celdas están conectadas a un único reloj de cómputo. A cada pulso del reloj cada celda consulta el estado de las celdas de su vecindad y de acuerdo con una tabla a la que llamamos su regla de evolución establece su nuevo estado. Si todas las celdas tiene el mismo conjunto de estados posibles, la misma forma de vecindad y la misma regla de evolución el sistema se llama un autómata celular (Toffoli y Margolus, 1987).

El autómata puede ser de 1, 2, 3, 4, etc. dimensiones. La teselación puede ser finita o infinita, con condiciones de frontera abiertas o periódicas, el conjunto de estados no necesita tener ninguna estructura algebraica adicional, la vecindad puede ser simétrica o no serlo y puede incluir o no la propia celda, y la regla de evolución, aunque se puede expresar en forma más reducida (p. ej., con una fórmula) es simplemente una tabla. Estos cuatro elementos: *teselación homogénea*, *estados*, *vecindad* y *regla de evolución*, definen el autómata celular.

Tal vez el ejemplo más conocido de autómata celular es el *juego de la vida* (*Life*) de John Hourton Conway, que se define de la siguiente forma (Berlechamp, Conway y Guy 1982; Gardner, 1983):

Vida

Teselación:	Cuadrícula homogénea
Estados:	Dos estados: vivo (negro), y muerto (blanco).
Vecindad:	Un cuadrado 3 x 3 centrado en la celda y que la incluye.
Regla de Evolución:	*Una celda viva permanece viva sólo si dentro de sus 8 vecinos hay 2 o 3 celdas vivas. De lo contrario, se muere. *Una celda muerta cambia a viva sólo si dentro de sus 8 vecinos hay exactamente 3 celdas vivas. De lo contrario, sigue muerta.

nw	n	ne
w	c	e
sw	s	se

Figura 1. Vecindad para el juego Vida

Si se parte de configuraciones aleatorias, vida desarrolla estructuras que oscilan, se mantienen estables, e incluso viajan a través del autómata, como las estructuras de la figura 2. También son muchas las configuraciones iniciales que al evolucionar generan estructuras complejas de singular belleza.

Esta característica de generar comportamientos complejos a partir de reglas muy sencillas es típica de los autómatas celulares. Como ejemplo, observe la siguiente regla:

Mayoría Alineada

Observo el cuadrado 3 x 3 centrado en mí. Si la mayoría son unos, me pongo en 1, y viceversa; excepto para 4 o 5 unos: si hay 4 unos, me pongo en 1, y si hay 5 unos, me pongo en cero.

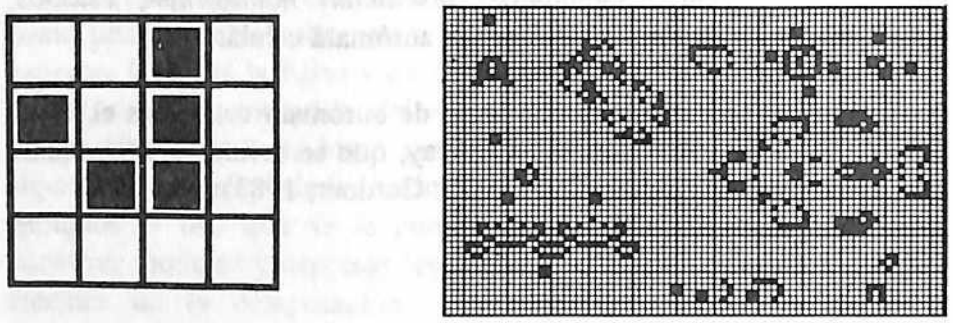


Figura 2. Configuraciones de **Vida** . A la izquierda, un *glider*, que se desliza una celda en diagonal cada cuatro (4) iteraciones; a la derecha, otras configuraciones interesantes y hermosas.

Una regla tan sencilla modela bastante bien el comportamiento de la tensión superficial entre líquidos inmiscibles (ver figura 3).

Figura 3. Mayoría Alineada: a partir de un patrón aleatorio, se forman zonas de solo unos o ceros. Al continuar la evolución, las fronteras se van alineando paulatinamente.

Sin embargo, esta aplicación se obtuvo jugando con reglas locales y viendo a qué se parecía el comportamiento obtenido; y lo que queremos es algo diferente: conocido el comportamiento global, deducir una regla de evolución local que lo genere. A lo primero se le llama el *problema directo*, y al segundo el *problema inverso*. También podemos hacer algo intermedio: a partir de un modelo de las interacciones locales de los elementos del sistema, construir una regla de evolución local y ponerla a prueba viendo si el autómata reproduce el comportamiento global del sistema que se está modelando.

Un ejemplo exitoso de esto es el modelo desarrollado por Martin Gerhardt y Heike Schuster de la Universidad de Bielefeld (Dewdney, 1988) para ciertas reacciones químicas reversibles llamadas excitables, en las que un catalizador favorece la reacción directa, al tiempo que los efectos de esa reacción transforman al catalizador en otro que favorecer la reacción inversa, y viceversa; esta reacción oscilante crea patrones de ondas de fase característicos, como los que se muestran en la figura 3. El modelo, que se hizo originalmente para la reacción $2CO+O_2=CO_2$ en presencia de cristalitos de paladio sirve también para modelar la reacción química de Belousov-Zhabotinski, y logra generar la estructura de ondas espirales que se observa en la reacción real (ver figura 4.).

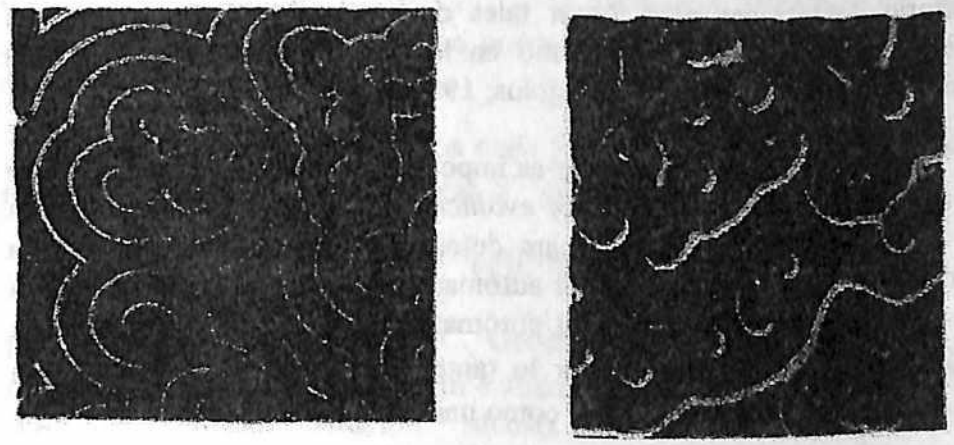


Figura 4. Reacción de Belousov-Zhabotinski. A la izquierda una fotografía de la reacción, a la derecha, su modelo de autómata celular (Tomado de Dewdney, 1988).

Los autómatas celulares definidos son así como el equivalente de los modelos de campos regidos por ecuaciones diferenciales parciales, y por lo tanto se muestran muy útiles en la solución de problemas de campos en los que la ecuación diferencial es muy difícil de resolver., como en la hidrodinámica, los modelos de Ising, clima, etc.. Igualmente son muy apropiados para tratar problemas en que los actores son más o menos iguales, se rigen por reglas parecidas y se presentan discretos, como en tráfico automotor, evolución galáctica, medios granulares, gases de Fermi, etc.. Prácticamente, si uno desea construir un modelo local y homogéneo de un fenómeno, puede intentar construirlo con un autómata celular; como en los modelos de ecosistemas, epidemiología, contaminación, etc.. Las memorias de los congresos bianuales sobre autómatas celulares (Farmer et. al., 1984; Horowitz, 1990; Boccara et. al., 1993) y la lista FAQ de autómatas celulares (CA-FAQ), son buenas referencias para encontrar ejemplos de dichas aplicaciones.

3. Propiedades. Un problema importante para permitir el modelado en física consistía en encontrar reglas locales que poseyeran la reversibilidad microscópica, la linealidad o la conservación de partículas que presentan muchas teorías físicas. Sin embargo, en los últimos 10 años la teoría de AC reversibles se ha desarrollado mucho, y hoy en día se conoce un buen repertorio de técnicas para lograr tales comportamientos, e incluso la mecánica estadística correcta, como en los modelos de Ising y en la dinámica de fluidos (Toffoli y Margolus, 1987).

Para la discusión que sigue es importante distinguir entre la *regla de evolución local* δ y la *regla de evolución global* τ. La primera es la que utiliza una celda particular para determinar su siguiente estado; la segunda establece la evolución del autómata como un todo, indicando a qué configuración C_{t+1} cambia el autómata cuando se encuentra en la configuración C_t a tiempo t (por lo tanto, $C_{t+1} = \tau(C_t)$). Esta regla global también se puede visualizar como una tabla de dos columnas (C_t y C_{t+1}), y E^N entradas (una por cada configuración posible), donde E es el número de estados y N es el número total de celdas del autómata.

3.1 Linealidad. Sea un autómata celular con regla de configuración global τ. Si para cualesquiera dos configuraciones *Ca* y *Cb* se cumple que:

$$\tau(Ca + Cb) = \tau(Ca) + \tau(Cb),$$

el autómata se llama lineal. Esto implica, naturalmente, haber definido de alguna manera la operación de suma entre configuraciones. Una forma de hacerlo consiste en numerar los n estados de cada celda de 0 a n-1, y sumarlos módulo n.

3.2 Cómo utilizar el pasado. Hasta ahora, nuestra regla genera C_{t+1} sólo a partir de C_t, o en otras palabras, es una regla de primer orden en el tiempo. ¿Cómo se puede hacer una regla de segundo o más orden en el tiempo, sin cambiar la definición de autómata celular? Duplicando el autómata: construimos un nuevo autómata celular en el que cada celda tiene ahora como estados posibles parejas ordenadas ($E1$,$E2$) del producto cartesiano Σ^2, y hacemos que, a cada instante t, $E1_t$ corresponda a C_t y $E2_t$ a C_{t-1}. Finalmente, dividimos la regla de evolución en dos partes: para el primer miembro hacemos que $E1_{t+1}$ dependa de $E1_t$ y $E2_t$, y para el segundo que sea simplemente copiar el primero antiguo, $E2_{t+1} = E1_t$; de esta forma C_{t+1}, formado por todos los $E1_{t+1}$, depende de C_t y C_{t-1}, y el pasado se actualiza de forma adecuada. El mismo proceso se puede realizar para construir reglas de órdenes mayores en el tiempo.

3.3 Reversibilidad. Una regla local δ de regla global τ se llama reversible si existe una regla local δ^{-1} de regla global τ^{-1} tal que:

$$\tau^{-1}(\tau(C)) = C$$

para cualquier configuración C. Generalmente la regla inversa δ^{-1} es diferente a la regla directa δ; sin embargo, existen casos especiales en los que, luego de una operación * sencilla (como invertir ceros por unos, o completar mód n), la misma regla directa hace evolucionar el sistema hacia atrás. En ese caso se dice que la regla δ es *auto-reversible bájo la operación* * :

$$\tau(*(\tau(C))) = C, \qquad \tau^{-1} = \tau \circ *.$$

Hay una forma fácil, descubierta por Fredkin, de construir una regla reversible a partir de otra regla cualquiera una vez se ha definido la operación de resta entre configuraciones (Toffoli y Margolus, 1987): sea τ una regla cualquiera; definamos ahora τ' como:

$$\tau'(C_t) = C_{t+1} = \tau(C_t) - C_{t-1}$$

Esta regla es reversible bajo la operación de intercambio entre C_{t+1} y C_{t-1}. En efecto, ya que

$$C_t = \tau(C_{t-1}) - C_{t-2}$$

es claro ver que,

$$\tau(C_{t-1}) - C_t = C_{t-2},$$

y que, por lo tanto, la misma regla τ' comienza a devolver en el tiempo al autómata.

Un ejemplo de una regla lineal y auto-reversible bajo intercambio es la regla **parity** de Ed Fredkin (Toffoli y Margolus, 1987), que para un autómata de teselación cuadrada y un conjunto de estados 1 o 0 calcula el siguiente estado, haciendo XOR entre su contenido, el de sus cuatro vecinos n s e w, y el que era su contenido en el pasado. La regla es capaz de tomar un dibujo bonito, evolucionarlo hasta volverlo una sopa de ruido, y recuperar el dibujo original.

4. Aplicaciones Físicas. Son muchas las áreas de la física en las que se han construido modelos de autómatas celulares. (Grassberger, 1984; Grossing, 1984, 1988; Herrmann, 1993, t'Hooft, 1990, Peng, 1994; Smith, 1990, Svorzil, 1986, Vichniac, 1984; Vollmar 1995; Witten, 1981,Wolfram, 1983, 1985a, 1985b). A continuación, analizaremos con detenimiento dos de ellas que son buenos ejemplos de las técnicas y características de los modelos con autómatas celulares, y que han servido

de paradigmas para otras aplicaciones: los modelos para dinámica de fluidos y para sistemas de Ising.

4.1. Dinámica de fluidos con autómatas celulares. Una de las ideas utilizadas para modelar dinámica de Fluidos con AC consiste en simular el movimiento y la interacción microscópica de las partículas del fluido. Por ejemplo, supongamos partículas idénticas y de la misma velocidad que se mueven horizontal y verticalmente por una red cuadrada de caminos, y que chocan elásticamente entre sí. Este modelo, llamado **HPP-GAS** (Hardy, de Pazzis y Pomeau, 1976), se muestra en la Figura 5a. Para implementarlo en un autómata celular se puede tomar una rejilla cuadrada en la que cada celda corresponde a un nodo de la red de caminos, y 4 bits del estado de la celda indican si hay o no hay en ella partícula viajando en dirección N, S, E o W. Cada paso de evolución se hace en dos etapas: en la primera, se mueven las partículas conservando su dirección; en la segunda, se evalúa si hay un choque para modificar los contenidos de los bits de la celda. Ya que en un AC cada celda puede sólo cambiar su contenido y no el de sus vecinas, es necesario establecer un protocolo entre celdas vecinas para que, al cambiar cada celda entre los estados de tener y no tener partículas, lo hagan de tal forma que simulen su movimiento y conserven su número.

Sin embargo, existe otra técnica más sencilla que logra este doble propósito: la técnica de programación por bloques. Ésta consiste en partir el AC en bloques iguales formados de varias celdas (ej.: 2 x 2), y definir una regla de evolución que diga cómo cambia el bloque como un todo, y que llamamos regla de evolución de bloque (ver Figura 5b.) Para que la información de bloques diferentes se mezcle, estos se arman con celdas diferentes en cada ciclo de reloj. En nuestro ejemplo, una regla de bloque posible, llamada **TM-GAS** (Toffoli y Margolus, 1987), es: en los ciclos pares girar el bloque 90° a la derecha, y en los ciclos impares girarlo 90° a la izquierda. Con esto, las partículas viajan horizontal y verticalmente. Para introducir los choques, hacemos una excepción en la regla anterior: cuando haya dos partículas en diagonal, lo dejamos inalterado y lo mismo hacemos cuando alguna de las celdas del bloque cae sobre una barrera.

La dirección inicial de una partícula viene dada por su posición inicial dentro del bloque, dirección que mantiene hasta el próximo choque; de esta forma tenemos densidades iniciales ρ_N, ρ_S, ρ_E, ρ_W de partículas que se mueven en las direcciones N, S, E y W, respectivamente (Figura 6). Para hacer que el fluido tenga, por ejemplo, una velocidad global de desplazamiento en dirección E, por ejemplo, hacemos $\rho_N = \rho_S$ y $\rho_E > \rho_W$. Sin embargo, los choques cambian estas densidades, ya que convierten un par de partículas E-W en uno N-S, o viceversa. Para que las densidades se mantengan uniformes requerimos que ocurran aproximadamente el mismo número de choques E-W que N-S. Esto se logra haciendo que $\rho_N \times \rho_S = \rho_E \times \rho_W$.

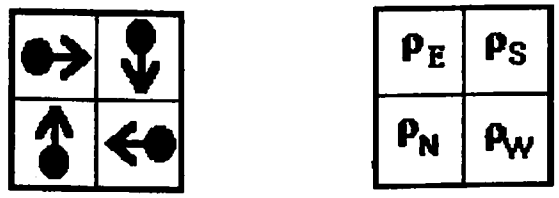


Figura 6. En **HPP-GAS**, la dirección de movimiento de una partícula queda determinada por su posición inicial en el bloque. Aquí, las direcciones correspondientes si se inicia con ún giro CCW, y las densidades a las que adicionan

El rozamiento con los obstáculos se modela haciendo más o menos rugosa su frontera, lo que de lejos genera el comportamiento de un rozamiento viscoso. De la misma forma, la viscosidad del fluido se modela colocando un conjunto de partículas fijas distribuidas aleatoriamente; su mayor o menor densidad determina el grado de viscosidad. Algunas características especiales como la isotropía de la difusión se pueden modificar introduciendo generadores aleatorios que modifiquen probabilísticamente el comportamiento de la regla de evolución.

Una vez iniciada la ejecución de la regla y establecido el flujo, es posible post-procesar los datos de la simulación para calcular parámetros como la fuerza de rozamiento, la vorticidad, etc.. Con esta regla sencilla ya se pueden modelar gran cantidad de fenómenos, como flujos de Couette, túneles de viento, propagación, reflexión, difracción e interferencia de ondas de densidad, etc. (Figura 7)

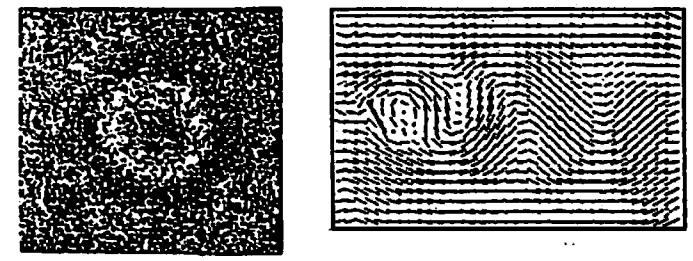


Figura 7. Aplicaciones de **HPP-GAS**. A la izquierda, una onda de densidad generada colocando en un gas en reposo una densidad central más alta (tomado de Toffoli y Margolus, 1987). A la derecha, un túnel de viento construido con un gas que se mueve hacia la derecha, utilizando la técnica que se acaba de describir (de Salem y Wolfram, 1986).

Sin embargo, **HPP-GAS** adolece de varios defectos, entre los que cabe destacarse que no hay uno, sino dos conjuntos de partículas que conservan su momentum: las N-S y las E-W, ya que las colisiones se realizan sólo cuando la suma de momenta es cero. Esto se solucionó con el modelo **FHP-GAS** (Frisch, Hasslacher y Pomeau, 1986), en el que las partículas se mueven a lo largo de una rejilla de triángulos equiláteros. Cada punto de intersección se puede modelar por una celda hexagonal, a la que pueden llegar partículas procedentes de sus seis celdas vecinas. Los choques se hacen también a momentum cero, pero el choque de tres partículas en Y mezcla los momenta de todo el sistema. Este modelo se implementa generalmente en la forma de evolución de dos tiempos, pero también existe una técnica de bloques que implementa una parte de la regla. **FHP-GAS** cumple la ecuación de Navier-Stokes en el límite de celdas muy pequeñas, y de ahí su éxito en gran cantidad de simulaciones; sin embargo, sólo lo hace para una relación fija del primer y segundo coeficientes de rozamiento, y predice algunas variaciones de densidad demasiado grandes para un líquido incompresible,. además de otras dificultades (Orzag, 1986). Finalmente, para 3 dimensiones no existen modelos que cumplan la ecuación de Navier-Stokes; sin embargo, se han desarrollado modelos de 4 dimensiones que proyectados en 3 dimensiones dan el comportamiento correcto. Actualmente hay bastante investigación para desarrollar modelos que predigan cada vez con mayor precisión los resultados experimentales.

4.2 Modelos de Ising. Otro campo de aplicación bastante extendido de los autómatas celulares en física son los modelos de Ising, que se basan en la interacción de espines de un material magnético (Toffoli y Margolus, 1987). Uno de los modelos más sencillos para materiales ferromagnéticos consiste simplemente en colocar un espín en cada celda de una teselación cuadrada; la energía de interacción entre dos espines vecinos se toma como 1 si son antiparalelos, y 0 si son paralelos, de forma que podemos decir que cada uno colabora a la energía del sistema con 1/2 o 0 unidades de energía, respectivamente. Si no hay fuentes externas, la regla de evolución ha de conservar la energía total del sistema.

Una regla posible consiste en tomar cada celda y contar cuánta energía de interacción sostiene con sus cuatro vecinos N, S, E y W; si la energía total no varía al invertir el espín central, se realiza la inversión; si no, se deja igual. Como la energía no varía sólo si los vecinos permanecen sin cambiar, es necesario que dos vecinos inmediatos no cambien a la vez; una forma de lograrlo consiste en colorear la teselación como un tablero de ajedrez, y que la regla de evolución la ejecuten a un pulso sólo las blancas, y al siguiente sólo las negras.

Algunos modelos de Ising suponen que la energía del sistema se puede almacenar de otra forma en la estructura cristalina. Otro modelo que incluye dicha posibilidad consiste en tomar el modelo anterior y agregarle una "alcancía"de ahorros de energía a cada celda. Si al efectuar la inversión de espín "sobran" dos unidades de energía (energía final mayor que la inicial), y la alcancía está vacía, se realiza la inversión y la energía sobrante se deposita en la alcancía como una monedita de dos unidades de energía. Si la inversión de espín requiere dos unidades de energía, y la alcancía está llena, éstas se toman de la alcancía y se realiza la inversión. El modelo tiene la ventaja de que se puede introducir "a mano" cada vez más energía al sistema llenando los bancos, lo que hace que los dominios evolucionen a distribuciones antiparalelas de espines, parecidas a las de los materiales antiferromagnéticos.

Finalmente, es posible introducir un *reservoir* térmico y una inversión por temperatura haciendo que las inversiones ocurran de forma

probabilista, con una distribución de probabilidad que dependa de la energía requerida para la inversión. Para llegar a una distribución canónica se necesita que:

$$\frac{P(\Delta E)}{P(-\Delta E)} = e^{-\frac{\Delta E}{kT}}$$

ya que los cambios posibles en energía son de -4,-2,0,2, o 4 unidades de energía, las probabilidades p de realizar dichos cambios deben estar relacionadas por:

$$\frac{p_{+4}}{p_{-4}} = \left(\frac{p_{+2}}{p_{-2}}\right)^2; \qquad \text{y} \qquad \frac{p_{+2}}{p_{-2}} = e^{-\frac{2J}{kT}},$$

con J una cantidad que compensa las unidades. Una forma de lograrlo es escoger las probabilidades como:

$$p_{-4} = p^2 \; ; p_{-2} = p \; ; p_0 = p_{+2} = p_{+4} = 1,$$

con lo que

$$T = \frac{2J}{k \log p}.$$

Estos modelos logran incluir en sus comportamientos cambios de fase en la magnetización, como el que se puede observar en la Figura 8c, y que corresponde a un modelo con reservoir térmico; además, hacen evolucionar las fronteras de los dominios magnéticos en formas similares a las de los materiales que simulan, como los de las figuras 8a y 8b. Sin embargo, se requiere una gran potencia de cómputo para lograr predicciones contrastables.

5. Aplicaciones a otras ciencias. Los autómatas celulares han sido aplicados también para modelar sistemas de muchas otras áreas del saber, como la biología, las ciencias de la computación, los modelos de cooperación en sociología, el tráfico automotor, etc.. Como ejemplos,

veamos algunas aplicaciones a la biología y a las ciencias de la computación.

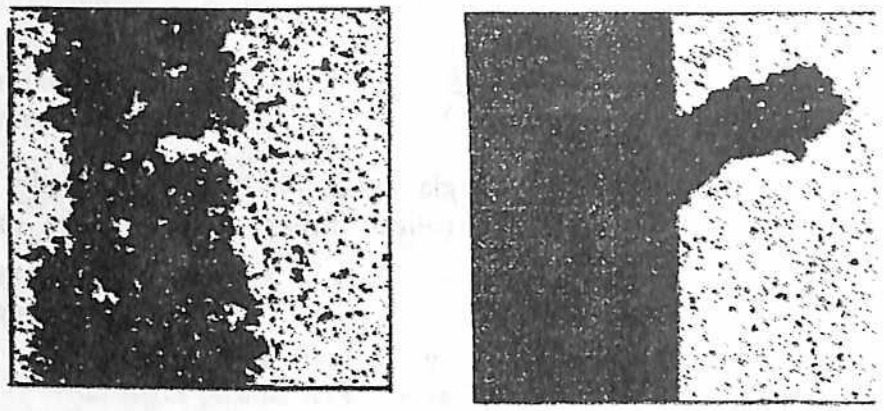


Figura 8. Modelos de autómata celular para sistemas de Ising. A la izquierda (a), configuración obtenida con el modelo sencillo de sólo espines (las celdas negras contienen espines hacia arriba, y las blancas espines hacia abajo). Al centro (b), configuración obtenida con el modelo don bancos de energía, luego de haber introducido a mano mucha energía adicional al sistema; en la parte izquierda se muestra el patrón antiferromagnético de tablero de ajedrez que surge en estas condiciones, y en la parte derecha se colorean las dos fases antiferromagnéticas que pueden surgir (una se obtiene de la otra intercambiando las celdas blancas y negras). A la derecha (c), curva de magnetización μ & la temperatura (expresada como función exponencial), para un sistema con reservoir térmico; observe especialmente la existencia en el modelo de un cambio de fase y de una temperatura crítica correspondiente (tomados de Tofffoli y Margolus, 1987)

5.1 Biología

5.1.1 Vida Artificial. Una de las personas que más impulsó el trabajo en autómatas celulares fue John von Neumann. Su objetivo era estudiar las características de la información de los sistemas vivos (von Neumann, 1966). Para ello, su idea era construir sistemas de información que tuvieran comportamientos similares a los de la vida; a este campo es a lo que se le llama vida artificial. Por una sugerencia de Stanislaw Ulam (Ulam, 1952), von Neumann utilizó para ello autómatas celulares.

La idea principal consiste en utilizar la regla de evolución del autómata como si fueran las leyes de la química, y buscar configuraciones de estados que aprovechen dichas redes para crecer, reproducirse, moverse, alimentarse, o cosas similares. La idea inicial de von Neumann era construir una estructura dotada de un programa que al ejecutarse la duplicara, transfiriera una copia del mismo a estructura copia, y lo activara para que ésta pudiera generar a su vez nuevas copias. von Neumann pedía, además, que el autómata tuviera computación universal, es decir, que las instrucciones del programa fueran capaces de generar estructuras capaces de realizar cualquier proceso de cómputo de una máquina de Turing que uno quisiera llevar a cabo.

von Neumann construyó un autómata con 29 estados por celda y 5 vecinos en cruz, y una configuración que ocupaba varios miles de celdas y que cumplía con las condiciones anteriores. Este autómata sirvió de base para trabajos posteriores como el de Codd (Codd, 1966), que lograba un resultado similar con 8 estados por celda. En el autómata de Codd, la información viaja a lo largo de un citoplasma de unos limitado por membranas celulares de doses. Las instrucciones viajan como un 0 seguido por el número de la instrucción; el 0 le sirve para saber hacia dónde debe viajar. Cuando una instrucción llega a una bifurcación, se duplica; de esta forma, un bucle cerrado se convierte en un generador periódico de instrucciones, que funciona como una base de tiempo. El autómata de Codd posee instrucciones para estirar o retraer las puntas, bien hacia delante, la derecha o la izquierda; y posee computación universal como pedía von Neumann.

En 1984, Chris G. Langton (Langton, 1984), hizo notar que si no se pedía computación universal, sino simplemente que la duplicación

dependiera más de la configuración inicial que de la regla de evolución, era posible modificar un bucle de base de tiempo del autómata de Codd para que se auto-duplicara. Tal modificación se conoce como el bucle-autoreplicativo de Langton, y es una de las estructuras más sencillas capaces de auto-reproducirse con este criterio; que entre otras cosas se ha constituido en el criterio más aceptado para identificar un organismo de vida artificial. El bucle de Langton va creciendo en forma de una colonia en la que los únicos organismos vivos son los del borde exterior, ya que los internos pierden su contenido de información cuando intentan duplicarse extendiendo su extremidad hacia un espacio ocupado. El bucle, luego de la primera duplicación, se muestra en la figura 9.

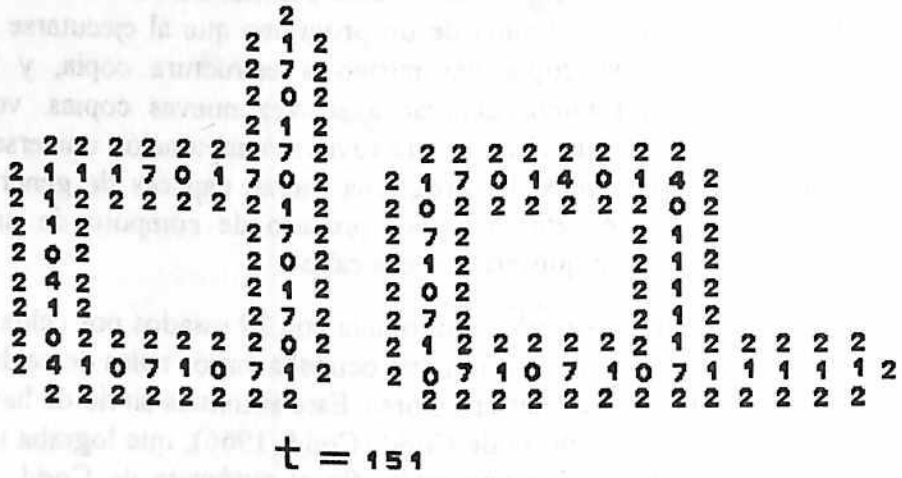


Figura 9. Bucle autorreplicativo de Langton luego de una división (tomado de Langton, 1984)

5.1.2 Evolución Simulada. Los autómatas celulares permiten construir ecosistemas virtuales en los que se pueden estudiar variables ecológicas o procesos de evolución. Por ejemplo, tomemos una región plana cerrada toroidalmente donde surgen al azar bacterias, y en la que se mueven microbios que comen bacterias (Dewdney, 1989). Cada vez que un microbio se come una bacteria, adquiere cierto número de unidades de energía (digamos 20); y cada vez que se mueve pierde una unidad de energía. Cuando a un microbio se le acaba la energía, se muere y desaparece; pero cuando adquiere más de cierta cantidad de energía

(digamos 200 unidades), se duplica. El movimiento de cada microbio está gobernado por seis genes, que determinan las probabilidades de cambiar su movimiento 60° o 120° a derecha o izquierda, invertir su dirección o seguir en la misma. Cuando un microbio se duplica, cada hijo hereda los genes del padre con una ligera modificación aleatoria en cada uno. De ésta forma, se permite la evolución por selección de aquellos cuyo movimiento los ha llevado a comer más eficientemente(ver Figura 10).

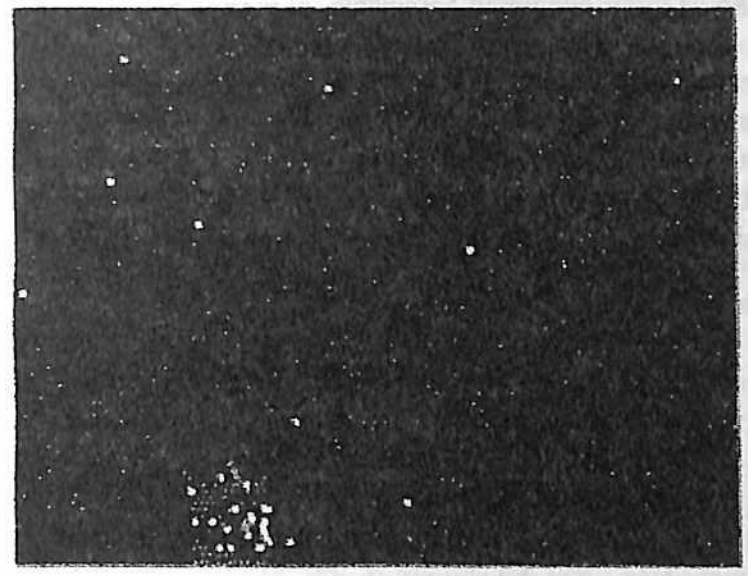


Figura 10. Evolución Simulada. Los microbios sobreviven comiendo bacterias que recogen en su movimiento. Dicho movimiento se rige por genes que sólo los que más comen logran pasar a la siguiente generación (tomado de Dewdney, 1989).

A medida que transcurre la evolución, aquéllos microbios que tienen movimientos erráticos mueren rápidamente, y sobreviven aquéllos que se mueven mayormente en línea recta, a los que llamamos cruceros. Es más, después de muchas iteraciones es posible encontrar a veces microbios que barren sistemáticamente la pantalla, línea por línea y píxel por píxel, de una forma verdaderamente sorprendente. Es conveniente también tener una pequeña región cuadrada donde la probabilidad de aparición de bacterias sea muy grande (ej.: 70%), a la que llamamos el jardín del edén. En él los microbios se reproducen a un ritmo muy grande,

y sirven como semillero de nuevas combinaciones de genes para ser probadas en el espacio abierto. Pero también allí ocurre evolución: en efecto, luego de un momento llegan a ser tantos que la comida comienza a escasear y empieza a ser importante la forma de moverse; surge así una segunda seudoespecie que se mueve en pequeñas espirales, y a la que llamamos viradores.

Es interesante hacer notar que si el número de bacterias permanece constante no se logra evolución apreciable, al menos por un buen período de tiempo. Una forma de lograrla rápidamente consiste en mantener más bien la energía total del sistema constante, es decir, la suma de las de las bacterias más las de los microbios. Para esto se puede hacer que surjan bacterias a partir de la energía que pierden los microbios al moverse. Igualmente, es bueno tener un tiempo mínimo de maduración del microbio antes de que pueda reproducirse (por ejemplo, de 200 pasos).

Estos modelos tiene la ventaja de que la información completa que ofrecen es finita y conocida; por lo tanto, caracterizadores del sistema como la entropía, la complejidad algorítmica, la biodiversidad, el flujo de biomasa, etc., se pueden medir exactamente. Así, estos modelos sirven como laboratorios idóneos para desarrollar nuevos caracterizadores; y poner a prueba cuantitativamente las relaciones que se propongan entre ellas, como por ejemplo autoorganización, criticalidad o relaciones termodinámicas en o fuera del equilibrio.

5.2. Ciencias de la Computación

5.2.1. Cabeza y Cola de Electrón. A lo largo de muchos años, se han ido desarrollando diferentes modelos de autómata celular que remedan el comportamiento de circuitos eléctricos digitales. En estos modelos, los circuitos se construyen con compuertas interconectadas por alambres por los que viajan trenes de pulsos que representan las señales a procesar. Generalmente, las compuertas se construyen con celdas iguales a las que forman los alambres (es decir, que están en el mismo estado que ellas).

Un ejemplo de tales modelos es el que se muestra en la figura 11 (Dewdney, 1990). Hay cuatro estados por celda: alambre, cabeza de

electrón, cola de electrón y fondo, que siguen las siguientes reglas: el fondo sigue siendo fondo, la cola de electrón se convierte siempre en alambre, la cabeza de lectrón en cola de electrón, y el alambre se convierte en cabeza de electrón sólo si uno o dos de sus 8 vecinos nw, n, ne, w, e, sw, s y se son cabeza de electrón; de resto sigue como alambre. Con esta regla sencilla es posible construir diodos y compuertas OR como las que se muestran en la Figura 11, y también compuertas XOR. Además, es posible hacer generadores periódicos de electrones con lazos cerrados en los que gira un electrón que al llegar periódicamente a la derivación del bucle se duplica (como en el autómata de Codd); con ellos se pueden hacer compuertas NOT, que junto con las OR permiten construir toda la lógica binaria. Este modelo permite también construir memorias RAM.

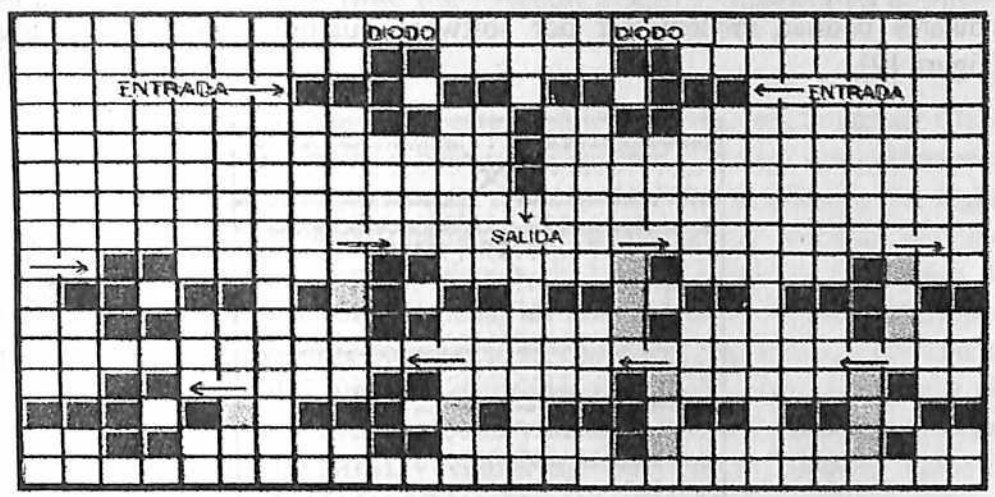


Figura 11. Compuerta OR y funcionamiento de un diodo construídos con "Cabeza y Cola de Electrón".

5.2.2 Modelos de compuertas con técnica de bloques. Otro modelo similar utiliza la técnica de programación por bloques, ya descrita cuando tratamos el modelo **TM-GAS** de dinámica de fluidos, y funciona con dos planos de bits. En el primer plano se encuentran dibujados los circuitos, en el segundo viajan las señales. En principio, si en el bloque solo hay una celda de circuito, no pasa nada; si hay dos, cada una copia el contenido de señal de la otra, con lo que se logra que las señales vayan viajando por el circuito; si hay tres, cada una toma como señal la OR de

las otras dos, con lo que se logran compuertas OR y bifurcaciones; y finalmente, si hay cuatro, cada una copia la señal de la celda diagonalmente opuesta en el bloque. Estos comportamientos se pueden modificar por señales colocadas fuera del circuito pero al lado de los mismos, y que permanecen fijos, de la siguiente forma: si en el bloque hay una celda de circuito y al lado una señal externa, toma una copia de dicha señal, con lo que se convierte en un generador; si hay dos y una señal externa, cada uno toma la señal invertida del otro, con lo que se convierte en un inversor; y si hay tres y una señal externa, cada uno toma como señal el AND, en vez del OR de las otras dos, con lo que ya tenemos compuertas AND que completan la lógica directamente. Con esta regla es posible modelar circuitos complejos en un espacio muy reducido, y se ha llegado a proponer incluso que circuitos hardware construídos con reglas similares podrían implementar por software cualquier *chip* integrado (Figura 12).

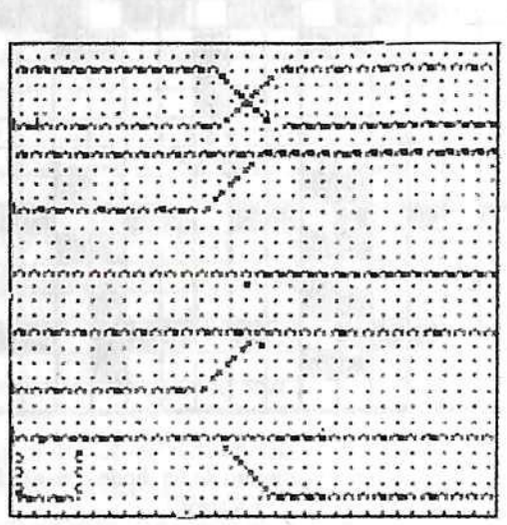


Figura 12. Modelo de compuertas con técnica de bloques. De arriba hacia abajo, intersección, compuerta OR, compuerta NOT, compuerta AND, y buche generador junto a una derivación.

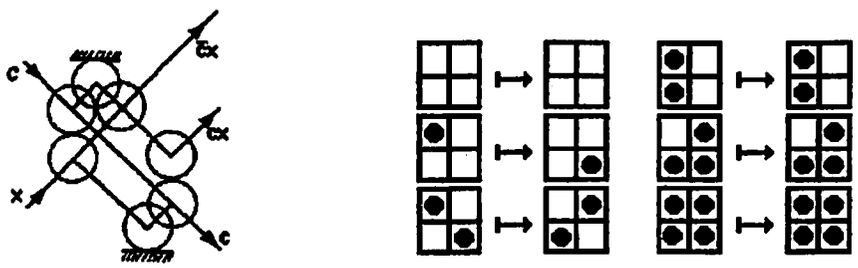
5.2.3 Modelo de computación con bolas de billar (BBM). Finalmente, deseo nombrar el modelo de computador de bolas de billar, que es un paradigma de teoría de computación desarrollado por Edward Fredkin y Tomasso Toffoli en MIT a principios de los 80 para estudiar propiedades termodinámicas de la computación (Fredkin y Toffoli, 1982),

como los costos termodinámicos en entropía que han de pagarse al realizar un proceso de cómputo. El modelo consiste de un conjunto de bolas que viajan sobre una mesa de billar completamente libre de rozamiento y chocan con las bandas de la mesa, entre sí y con espejos fijos en el interior de la mesa de forma completamente elástica. Los datos, tanto de entrada como de salida, se representan como la presencia o ausencia de bolas de billar en ciertas posiciones de la mesa, en ciertos instantes y con determinadas direcciones; y el computador se programa colocando espejos en posiciones adecuadas. Los modelos de bolas de billar, o también llamados computadores balísticos, pueden realizar cualquier tipo de cómputo. Las bolas son capaces de chocar y rebotar con las barreras y entre sí, y se pueden realizar circuitos enteros basados por ejemplo en demultiplexores 1 a 2, como el que se muestra en la Figura 13a.

¿Se puede implementar este modelo en un autómata celular? *Claro* que sí, pero es más: bolas y barreras se pueden construir como formadas por un mismo tipo de partículas, con una sencillez asombrosa. La regla, llamada **BBC-CA** y desarrollada por Norman Margolus del MIT en 1984 (Margolus, 1984), utiliza la técnica de programación de bloques y se muestra en la Figura 13b. En ella, las bolas elásticas se representan por parejas de partículas que se mueven en diagonal, y las barreras por hileras dobles de partículas unidas, cuidadosamente colocadas, tal como se muestra en la Figura 13c. Las bolas y las barreras son capaces de realizar todos los choques de BBM, y realizar cualquier tipo de cómputo, como el que se muestra en la Figura 13d.

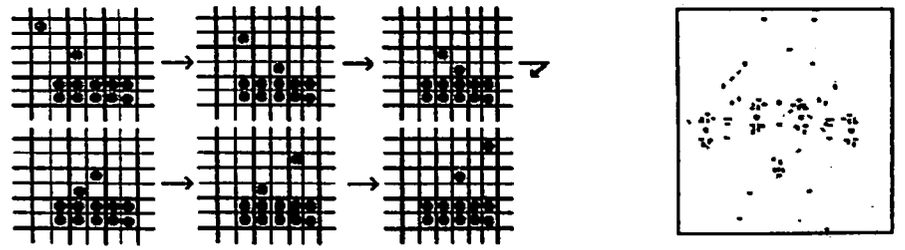
6. Discusión. Usualmente, los modelos de la física y de otras ciencias se han construido utilizando expresiones de funciones continuas de números reales o complejos, como p. ej. ecuaciones diferenciales, integrales funcionales, etc.. Éstas expresiones, a las que llamamos analógicas, suelen utilizar notación simbólica y son adecuadas para ser trabajadas a mano, sin equipo adicional; igualmente, las funciones analógicas que arrojan como resultados cuando se resuelven analíticamente son fáciles de utilizar.

Sin embargo, en la mayoría de los casos no es posible resolverlas exactamente, y se hace necesario acudir a métodos aproximativos por lo general bastante engorrosos. Con el advenimiento de los computadores llegó la posibilidad de utilizarlos para implementar dichos métodos aproximativos, bien fuese utilizando métodos numéricos o manipulación simbólica. Pero, obtener resultados numéricos contrastables con experimentos utilizando computadores en esta forma puede ser bastante difícil, porque entre otras razones hay muchos pasos de aproximación en el proceso (el algoritmo finalmente implementado suele ser la aproximación digital de una aproximación analógica del modelo propuesto por la teoría que se esté utilizando); la digitalización que ocurre al pasar el modelo al computador puede introducir desviaciones que no se esperaban del modelo analógico; y los resultados suelen ser valores aislados y no distribuciones de probabilidad fáciles de contrastar con el experimento (dar la distribución utilizando teoría del error suele ser muchísimo más complejo que el trabajo ya grande de obtener el resultado numérico). Todo esto puede llevar a que fácilmente el algoritmo implementado diverja o dé resultados poco confiables; además, por lo general requieren de una enorme potencia de cómputo.



(a)

(b)



(c) (d)

Figura 13. Modelos de computación con bolas de billar (BBM). Arriba a la izquierda (a), un demultiplexor 1 a 2 construido con BBM. Arriba a la derecha (b), la regla de bloques **BBM-CA** de Margolus, que implementa un computador de bolas de billar con autómatas celulares. Abajo a la izquierda (c), una bola chocando con una barrera, implementados con **BBM-CA**; la bola está formada por las dos partículas en diagonal; la barrera permanece fija gracias a su cuidadoso posicionamiento; el movimiento inicia con los bloques formados por las líneas gruesas. Abajo a la derecha (d), un cómputo en proceso de ser ejecutado por un computador construido con **BBM-CA**. (Tomados de Toffoli y Margolus, 1987).

De otra parte, observemos que muchas veces los datos que se obtiene en los experimentos son de carácter digital. En efecto, generalmente el número de resultados que el instrumento de medida puede entregar es finito, o al menos numerable, debido a que su resolución es finita, y por lo tanto se utiliza un número finito de cifras para anotar dicho resultado: una regla de 10 cm graduada en mm no puede brindar micras. Esta idea es muy antigua y ya fue expresada por Schrödinger en sus escritos (Schrödinger, 1976). Además, cada vez que coloquemos la regla sólo nos puede brindar hasta 10 cm, de forma que en un número finito de pasos sólo nos puede dar un número finito de veces 10 cm. De otra parte, el número de veces que efectivamente se repite una medida en un experimento es finito, de forma que el histograma obtenido se forma de un número finito de bloques, por así decirlo. Por lo tanto, los resultados de medición así obtenidos suelen ser histogramas digitales; histogramas que generalmente luego aproximamos por una distribución analógica para alimentarlos a los modelos analógicos.

Ante esta situación, surge la siguiente pregunta: ¿será posible construir, además de modelos analógicos, modelos y leyes de la física directamente en forma digital, en una forma tal que se pudieran

implementar directamente en el computador? ¿En una forma que, por ejemplo, pudiera tomar directamente los resultados digitales de las medidas de entrada y generar directamente las predicciones también como un histograma digital, similar al que se obtiene al realizar las medidas de salida? De esta forma habría menos pasos de aproximación, ya que el modelo daría directamente la salida; sería más rápido de implementar en muchos casos, ya que suele suceder que nuestras ideas originales al modelar un comportamiento sean digitales; no habría errores de digitalización al pasar al computador, porque el modelo ya sería digital; y además la predicción sería fácil de contrastar con el experimento, si se da por ejemplo como un histograma del mismo tipo que el obtenido en el experimento. Ésta es una alternativa muy interesante que se viene trabajando desde hace algún tiempo (Toffoli, 1984, 1989; Fredkin, 1990), y que brindaría una herramienta adicional para tratar especialmente aquéllos fenómenos de difícil solución analítica.

En esta dirección, los autómatas celulares aparecen como una buena posibilidad. Al ser locales y homogéneos, son algo así como el equivalente digital de los modelos analógicos de campos regidos por ecuaciones diferenciales parciales. Al igual que en estos, las condiciones de un problema específico se alimentan como condiciones iniciales o de frontera del problema. La ley de evolución hace las veces de la ecuación diferencial parcial, y al igual que ésta es local y homogénea. Por lo tanto se presentan como una estructura digital muy apropiada para expresar leyes y construir modelos en ciencia directamente en forma digital.

Además, un problema expresado como autómata celular es directamente paralelizable: varios procesadores pueden compartir el trabajo tomando cada uno un conjunto generalmente conexo de celdas, y sólo han de transmitirse la información de los estados de la frontera. Incluso se puede llegar a implementar un microprocesador por celda: los computadores de propósito especial implementados de ésta forma son más económicos y sencillos de construir que los sistemas convencionales de propósito general, ya que los procesadores pueden no requerir incluso saber sumar, sino simplemente mirar una tabla; el sistema de interconexión entre microprocesadores es fijo y sencillo; y no se requieren complicados sistemas operativos que repartan el trabajo entre los microprocesadores:

todos saben qué hacer (Margolus, 1986). Algunos de los sistemas más sencillos que se han construido de ésta forma pueden conectarse directamente a un PC, y por un precio inferior a US$2000 ofrecen una potencia similar a la de un Cray-1 para estas aplicaciones especiales (Toffoli y Margolus, 1987). Por todo lo anterior, los autómatas celulares aparecen también como excelentes alternativas de cómputo, que al aprovechar con una organización diferentes los recursos de un computador pueden ser más rápidas y sencillas de implementar, más eficientes e incluso más económicas de realizar que los modelos convencionales.

En resumen, los autómatas celulares pueden ser excelentes alternativas, tanto a nivel teórico como computacional, para construir modelos y expresar leyes en las ciencias; modelos que además son digitales y por ello directamente implementables en el computador, y que además pueden llegar a arrojar resultados fácilmente contrastables con las medidas experimentales. Invito, pues, a construir modelos y ejecutar predicciones utilizando autómatas celulares. Si se quiere construir un modelo local y homogéneo de un sistema o proceso, bien vale la pena intentar construir como autómata celular.

7. Agradecimientos. Deseo agradecer primeramente de todo corazón a Dios por su apoyo y ayuda constante en la realización de este artículo; y por las personas maravillosas que ha puesto en mi camino para llevarlo a cabo. Igualmente a la Virgen María por su intercesión y guía, siempre dispuesta a ayudarme y pendiente de mí, siempre conmigo. Quiero agradecer también a los profesores Germán Hernández, Luis Fernando Niño, Luz Gloria Torres y J. J. Martínez, del *Grupo de Sistemas Complejos* de la Universidad Nacional de Colombia por su apoyo entusiasta y fraterna para trabajar en este tema, y por su gentil invitación para participar en el *I Congreso Nacional de Neurocomputación*. Igualmente, quiero agradecer a los profesores Fernando Zalamea, Juan Manuel Tejeiro, Hernán Estrada y Diógenes Campos de los Departamentos de Matmáticas y de Física de la Universidad Nacional de Colombia por las enriquecedoras discusiones sobre estos temas y por las oportunidades que me han brindado para desarrollarlo; a mis padres José

Ma. Muñoz y Nydia de Muñoz por su apoyo constante y su ayuda en la revisión del texto. Además, deseo agradecer a la *Fundación Mazda para el Arte y la Ciencia* (Contrato No. 91-113), y al *CINDEC* de la Universidad Nacional de Colombia (Proyecto *Autómatas Celulares y Física Digital - Fase I*) por brindarme el apoyo que hace posible esta investigación.

BIBLIOGRAFÍA

Berlechamp, Elwyn, Conway, John, Guy, Richard. *Winning Ways*. Academic Press, 1982.

Boccara N. et. al., Ed.. NATO Advanced Study Institute 1993. *Cellular Automata and Cooperative Systems*. Kluwer Academic Publishers.

CA-FAQ = http://alife.santafe,edu/alife/topics/cas/ca-faq/ca-faq.html

Codd, E. F. *Cellular Automata.* Academic Press, New York, 1966.

Dewdney, A. K. *Juegos de ordenador.* Investigación y Ciencia, No. 145, 1988.

Dewdney, A. K. *Juegos de ordenador.* Investigación y Ciencia, No. 154, 1989.

Dewdney, A. K. *Juegos de ordenador.* Investigación y Ciencia, No. 162, 1990.

Dewdney, A. K. *Juegos de ordenador.* Investigación y Ciencia, Nos. 106, 122, 127, 134, 139, 140, 145, 149, 154, 157, 162,(1985-1990).

Fredkin, Edward, and Toffoli, Tomasso.. *Conservative Logic*. Int. J. of Theo. Phys. **21** (1982), págs. 219-253.
Fredkin, Edward. *Digital Mechanics*. Physica D **45** (1990), págs. 254-270.

Frisch, U., Hasslacher, B., y Pomeau, Y.. *Lattice-Gas Automata for the Navier-Stokes Equation*. Physical Review Letters **56** (7 April, 1986), No. 14, págs. 1505-1508.

Gardner, Martin. *Wheels. Life and other Mathematical Amazements*. W. H. Freeman and Company, 1983.

Grassberger, *Chaos and Diffusion in Deterministic Cellular Automata*. Phisica D **10** (1984), págs. 52-58.

Grössing, Gerhard. *Real Quantum Cybernetics*. Physics Letters A, **121** (4 de Mayo de 1987), No. 6, págs. 259-266.

Grössing, Gerhard y Zeilinger, Anton. *Quantum Cellular Automata*. Complex Systems **2** (1988), No. 2, págs. 197-208.

Hardy, J., de Pazzis, O. y Pomeau, Y.. *Molecular Dynamics of a Classical Lattice Gas: Transport Properties and Time Correlation Functions*. Physical Review A **13** May 1976), No. 5, págs. 1949-1961.

Herrmann, H., *Proceedings of the Granada School*, ed. L. Garrido, J. Marro, 1995.

t'Hooft, Gerhard. *Quantization of Discrete Deterministic Theories by Hilbert Space Extension*. Nuclear Physics B **342** (1990) No. 3, págs. 471-485.

Horowitz, Ed.. Physica D-45 (1990). *Cellular Automata*. North Holland Physics Publishers, Amsterdam.

Langton, Christopher G.. *Self-Reproduction in Cellular Automata*. Physica D **10** (1984), págs. 135-144.

Margolus, Norman. *Physics-Like Models of Computation*. Physica D **10** (1984), págs. 81-95.

Margolus, Norman, Toffoli, Tommaso y Vichniac, Gérard. *Cellular-Automata Supercomputers for Fluid-Dynamics Modeling*. Physical Review Letters **56** (21 April 1986), No. 16, págs. 1694-1696.

Orzsag, Steven A. y Yakhot, Victor. *Reynolds Number Scaling of Cellular-Automaton*. Physical Review Letters **56** (21 April 1986), No. 16, págs. 1691-1693.

Peng, G. y Herrmann, H. J., *Density Waves and 1/f Density Fluctuations in Granular Media*, Physi. Rev. E, **49**, R1796 (1994)

Salem, J. y Wolfram, S.. *Thermodynamics and Hydrodynamics of Cellular Automata", Theory and Applications of Cellular Automata*. Editado por Stephen Wolfram,. World Scientific, 1986. págs. 362-366.

Schrödinger, Erwin. *Causalidad y mecánica ondulatoria*. en *Sigma. El mundo de las matemáticas*, Tomo 2, Editado por James R. Newman, traducido por Xavier Campi. 3a. Ed.. Barcelona, Ediciones Grijalbo S. A., 1976, págs. 332-345.

Smith, Mark A.. *Representations of Geometrical and Topological Quantities in Cellular Automata*. Physica D **45** (1990), págs. 271-277.

Svozil, Karl. *Are Quantum Fields Cellular Automata?*. Physics Letters A, **119** (15 December 1986), No. 4, págs. 153-156.

Toffoli, Tommaso. *Cellular Automata as an Alternative to (Rather than an Approximation of) Differential Equations in Modeling Physics*. Physica D **10** (1984), págs. 117-127.

Toffoli, Tommaso y Margolus, Norman. *Cellular Automata Machines: a new environment for modeling*. The MIT Press, Cambridge MA, 1987.

Toffoli, Tommaso. *How Cheap can Mechanic's First Principles Be?* Proceedings of the 1988 Workshop on Complexity, Entropy and the Physics of Information (Santa Fe, New Mexico, May-June, 1989). Editado por Wojciech H. Zurek. Santa Fe Institute Studies in the Sciences of

Complexity, Proceedings volume VIII, Addisson-Wesley Publishing Company, Redwood City, California, 1990, págs. 301-318.

Ulam, Stanislaw. *Random Process and Transformations*. Procceedings of the 1950 Int. Congr. Mathem. 2 (1952), 264-275.

Vichniac, Gérard. Y.. *Simulating Physics with Cellular Automata*. Physica D **10** (1984), págs. 96-116.

von Neumannn, John. *Theory of Self Reproducing Automata*. Editor: Arthur Burks, University of Illinois Press (1966).

Witten Jr., T. A. y Sander, L. M. *Diffusion-Limited Aggregation, a Kinetic Critical Phenomena*. Physical Review Letters, Vol. 47 (9 Novembe 1981) No. 19, págs. 1400-1403.

Wolfram, Stephen. *Statistical Mechanics of Cellular Automata*. Reviews of Modern Physics **55** (July 1983), No. 3, págs. 601-644.

Wolfram, S., Ed..Physica D-10 (1984). *Cellular Automata*. North Holland Physics Publishers, Amsterdam. 247 pp.

Wolfram, Stephen. *Undecidability and Intractability in Theoretical Physics*. Physics Review Letters **54** (1985) No. 8, págs. 735-738.

Wolfram, Stephen. *Origins of Randomness in Physical Systems*. Physical Review Letters **55** (29 July 1985), No. 5, págs. 449-452.

CREDIT CARD FRAUD DETECTION USING NEURAL NETWORKS

Stephen W. Piché*
Pavilion Technologies, Inc.
12112 Technology Blvd.
Austin, TX 78727-6298, USA

Abstract. As the use of credit cards has grown world-wide over the past several decades, credit card fraud has also grown. In recent years, sophisticated counterfeiting schemes have accounted for significant losses to credit card companies such as Visa International. As counterfeit methods have become more sophisticated, credit companies have had to rely on new pattern recognition technologies to control fraud. In this paper, a fraud detection system based upon neural network technology is presented. This system was employed on-line in the summer of 1993 and has been credit with significantly reducing Visa International's counterfeit fraud.

Resumen. A medida que el uso de tarjetas de crédito ha aumentado en el mundo en las últimas décadas, el fraude también ha crecido. En años recientes, esquemas sofisticados de falsificación han representado grandes pérdidas a compañías de tarjetas de crédito como Visa Internacional. Ya que los métodos de falsificación se han vuelto más sofisticados, las compañías de crédito han tenido que valerse de las nuevas tecnologías de reconocimiento de patrones para controlar el fraude. En este trabajo, se presenta un sistema de detección de fraude basado en la tecnología de redes neuronales. El sistema fue empleado en línea en el verano de 1993 y ha contribuido significativamente en la reducción de falsificación en Visa Internacional.

1. Credit Card Fraud. Credit card fraud can be categorized into two common types: stolen, and counterfeit. The fraud activity associated with a stolen card is relatively easy to detect with simple pattern recognition systems. Typically, this activity is characterized by a spending burst which is stopped either when the card is reported stolen or the bank discontinues authorization of transactions. Because of the sudden high activity, many banks have been able to develop pattern recognition systems which effectively identify such fraud in a short period of time.

* While the author worked as a member of the technical staff at the Microelectronics and Computer Technology, Corp. (MCC) in Austin, TX, he designed and developed the neural network system for Visa International's Cardholder Risk Identification Service (CRIS). This paper, which presents results on the performance of this system, is based upon publicly available information.
E-mail Address: piche@pav.com

As the credit card companies have developed systems to reduce fraud from stolen cards, thieves have responded by turning to counterfeiting credit cards. Credit cards can be counterfeited by obtaining a valid account number and then using this number to manufacture a counterfeit card. The advantage of counterfeiting to the thief is that the account holder is often unaware of the fraudulent activity until they receive their monthly statement. Because the account holder is unaware, the counterfeit thief can more closely mimic the spending patterns of a valid card holder thus fooling a fraud detection system designed to prevent stolen transactions. Counterfeit credit card fraud dramatically rose in the early 90's. For instance, according to Visa, its world-wide fraud rate increased from 0.09% in 1990 to 0.14% in 1992 with counterfeit driving the increase [1].

To reduce counterfeit fraud at Visa, the Neural Networks Project at the Microelectronics and Computer Technology Company (MCC) was employed to develop a neural network based fraud detection system. The next section describes neural networks and their advantages for solving difficult pattern recognition problems. The third section reviews publicly available results for the fraud detection system developed for Visa. The last section presents conclusions and future directions for the use of neural networks in area of fraud detection.

2. Neural Networks. An artificial neural network consists of a large number of interconnected simple computing devices whose design is motivated to some extent by biological neural structure. A neural network, like many systems, is used to compute an output response to a set of inputs. It is unique, however, in that the function it computes is usually tuned adaptatively, and that its structure allows parallel computation.

The most commonly used neural network architecture is the sigmoidal Madaline (also know as a feedforward neural network or multilayer perceptron)[2]. The Madaline is a feedforward layered network which is composed of "Many Adalines." An Adaline (an acronym for ADaptive LINear Element) consists of a single computing device (an adder and output operator) which is connected to a set of inputs through weights. The input to an Adaline is denoted by the vector $X \in R^{[N\times 1]}$. The

N-1 inputs, $\{x_2,\ldots,x_N\}$, are allowed to take on any value while the first input, x_1, known as the bias input, is always 1. The weights, represented by $W \in R^{[N\times 1]}$, are adaptable, and may be tuned using a training algorithm such as the least-mean-square (LMS) algorithm[3]. The output of the Adaline is

$$y = f(X^t W). \tag{1}$$

If the function f is Linear, the Adaline is a Linear device. If f is implemented by the signum function, then the Adaline is referred to as a threshold Adaline. The output of such a unit is defined as

$$y = \text{sgn}\,(X^t W) = \begin{cases} 1 & \text{if} \quad X^t W \geq 0 \\ -1 & \text{if} \quad X^t W \leq 0 \end{cases}$$

The output of a sigmoidal Adaline is

$$y = \tanh\,(X^t W), \tag{2}$$

where the output operator f implements the sigmoid (hyperbolic tangent) function.

Although an Adaline consists only of one "neuron", it has proven to be a useful computing device. Linear Adalines have been trained using the LMS algorithm to implement finite impulse response filters and controllers using only input and desired response data [4]. Many pattern classification problems have been solved by training the Adaline using either the LMS algorithm or the perceptron rule[5,3]. However, the Adaline is only capable of implementing a linear surface or decision boundary. For cases where nonlinearities are required, a Madaline may be used.

A Madaline consists of layers of many Adalines. Typically, the input to each of the first layer Adalines is the input to the Madaline. The outputs of these first layer units are computed using the weights associated with each unit along with the Madaline input. The outputs of the first layer form the inputs to a second layer of Adalines. In a manner similar to the first layer, the outputs of the second layer are calculated using the associated weights and inputs. The outputs of the third and subsequent lay-

ers are calculated in a similar manner. There is no limit to the number of layers in a Madaline, however, it is common to use only a few layers.

Madalines are very useful computing devices which have been used successfully in speech recognition, control, robotics and system identification applications [6,7,8,9,10,11]. By using a large number of either threshold or sigmoidal Adalines in the first layer and linear units in the second, any continuous nonlinear function of the inputs may be approximated to an arbitrary accuracy [12,13]. Furthermore, by using threshold Adalines in the second layer instead of linear units, any nonlinear decision surface may be approximated. A number of training algorithms for adapting the weights of a sigmoidal Madaline have been developed [14,15,16]. Because of the success of these algorithms in training neural networks, the Madaline has become one of the most commonly used neural network architectures.

3. Fraud Detection using Neural Networks. Because of the many successful applications of neural networks, Visa decided to develop a fraud detection system based upon this technology. In the fall of 1992, the Neural Network Project at MCC was employed to develop the neural network portion of the system.

Developing a fraud detection system for Visa was particularly challenging because Visa possessed data only on recent transactions associated with an account and had no information on the account holder. For example, Visa had information on the dollar amounts of recent transactions, what type of goods or services were bought, and the geographic locations of the merchants, but they had no information on the card holder's income, nor did they have a history of the account over the past few years. Because member banks had this information for developing their own internal fraud detection systems, it was uncertain at the beginning of the project whether a fraud detection system developed for Visa could have performance superior to those already in use at the member-banks. In addition, Visa was interested in reducing counterfeit fraud, which is more difficult to detect than fraud associated with stolen cards.

A pilot test of the neural network based fraud detection system developed by MCC proved that despite the lack of information, a properly trained system could provide benefits to Visa's member banks and signficantly reduce counterfeit fraud. Based upon documents submitted to Visa, the 10 US and five Canadian banks which participated in the pilot between August 1993 and December 1994 saved $43 million [1]. Furthermore, according to Visa, Bank of America saved $4.4 million between August and December of 1993, and First Chicago saved more than $2.3 million in the first six months of its trial. Fraud losses at Bank of America were down 24% in 1994 over the previous year with the neural network fraud detection system being credit with much of the reduction.

Due to the success of the fraud detection system, a technology transfer between MCC and Visa was conducted during the summer of 1994. Visa now uses the software developed at MCC to further refine and develop neural network models. Because of the success of the initial models which were oriented at detecting counterfeit activity, Visa has recently developed additional models which focus on fraud activity due to stolen and lost cards.

4. Conclusion. Neural network based technology has proven to be an effective technique for reducing counterfeit losses at Visa. In a 17 month pilot test, the neural network based system was able to detect $43 million dollars in fraud at 15 banks. Many of these banks had their own internal fraud detection system, therefore, the savings reported represents fraud that was missed by these systems.

As the computational capabilities of computers continue to increase, the use of neural networks in the area of fraud detection will become even more prevalent. Increases in computational speed will allow the combination of multiple neural networks. It will also allow the use of more computationally expensive nearest neighbor based neural network algorithms. In addition, better neural network development tools in the future should allow for faster and more accurate creation of fraud detection models.

BIBLIOGRAPHY

[1] R. E. Calem. To catch a thief. *Forbes ASAP*, 44-45, June 1995.

[2] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proc. IEEE*, 78 (9) (1990), 1415-1442.

[3] Widrow and M. E. Hoff, Jr. Adaptive Switching Circuits. In *1960 IRE Western Electric Show and Convention Record, Part 4*, pages 96-104, August 23 1960.

[4] B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[5] F. Rosenblatt. On the Convergence of Reinforcement Procedures in Simple Perceptrons. *Cornell Aeronautical Laboratory Report VG-1196-G-4*, Buffalo, New York, February 1960.

[6] A. Guez and J. Selinsky. A trainable neuromorphic controller. *Journal of Robotic Systems*, 363-388, 1988.

[7] M. Jordan. Generic Constraints on Underspecified Target Trajectories. In *Proceedings of the International Joint Conference on Neural Networks*, pages 217-225, Washington, DC, June 1989.

[8] K. S. Narendra and K. Parthasarathy. Identification and Control of Dynamic Systems Using Neural Networks. *IEEE Trans. on Neural Networks*, 4-27, March 1990.

[9] D. Nguyen and B. Widrow. Neural Networks for Self-Learning Control Systems. *IEEE Control Systems Magazine*, April 1990.

[10] Terrence J. Sejnowski and Charles R. Rosenberg. Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1:145-168, 1987.

[11] Alexander Waibel and Toshiyuki Hanazawa and Geoffrey Hinton and Kiyohiro Shikano and Kevin J. Lang. Phoneme Recognition Using Time Delay Neural Networks. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-37 (3): 328-339, 1989.

[12] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Proceedings of International Joint Conference on Neural Networks*, pages 593-611, June 1989.

[13] K. Hornik and M. Stinchcombe and H. White. Multi-layer feedforward networks are universal approximators. *Neural Networks*, 2(2): 359-366, 1989.

[14] D. Andes and B. Widrow and M. Lehr and E. Wan, MRIII: A Robust Algorithm for Training Analog Neural Networks. In *Proceedings of the International Joint Conference on Neural Networks*, , pages 533-36, Washington, DC, January 1990.

[15] D. E. Rumelhart and G. E. Hinton and R. J. Williams. Learning Internal Representations by Error Propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, chapter 8, The MIT Press, Cambridge, MA, 1986.

[16] P. Worbes. Back propagation through time: what does and how to do it. *Proc. IEEE*, /8(10):1550-1560, 1990.

[17] R. J. Williams and J. Peng. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 490-501, Winter 1990.

ADAPTIVE ARTIFICIAL PROCESSES

M. Tomassini
Swiss Scientific Computing Center CSCS, Manno
and
Laboratoire de Systèmes Logiques
École Polytechnique Fédérale de Lausanne
Switzerland

Abstract. Adaptive artificial processes such as *Evolutionary Algorithms* have aroused intense interest in the past few years because of their versatility in solving difficult problems in search, optimization and machine learning. Witnessing the vitality of the field, a large number of different applications have been reported. In this article we will first briefly review the foundations of the field and then we will describe as illustrative examples a couple of typical applications of simulated evolutionary processes.

Resumen. Procesos artificiales "adaptativos" tales como algoritmos genéticos evolutivos han despertado interés en los últimos años porsu versatilidad para resolver problemas difíciles en búsqueda, optimización y aprendizaje. Una gran cantidad de aplicaciones que han sido reportadas son testigos de la vitalidad del campo. En este artículo revisamos en primera instancia, los fundamentos del campo y luego describimos como ejemplo ilustrativo dos aplicaciones típicas de procesos evolutivos simulados.

1. Introduction. Evolutionary Algorithms (EAs) are artificial adaptive processes that find their origin and inspiration in the biological world. Darwinian evolutionary theory with the survival of the fittest in a changing environment seems to be generally accepted as the main driving force behind the dynamical multiple equilibria among the bewildering number of natural species. In this view, there are a few processes that, acting on populations of individuals, promote this enormous, apparently purposeless complexity through continuos optimization and change: reproduction, mutation, competition and selection. Reproduction belongs to the very essence of life. Mutation is guaranteed by the fact that no absolutely error-free biological process can exist. Competition and selection come about because any growing population only has a finite amount of resources available to sustain growth.

EAs try to abstract and mimic some of the traits of the ongoing struggle for evolution in order to do a better job in problems that require adaptation, search and optimization. But biological evolution took millions of years, is an ongoing process and operates in an exceedingly complex system of interactions. Evolution-inspired methodologies are man-made artifacts that only capture major distinctive features of natural evolution. Simplified as they are, they offer the advantage of running at electronic speeds, of being amenable to mathematical analysis and of not being overly constrained by physical laws thus giving rise to artificial worlds in which man can experiment with different rules.

Evolutionary Algorithms is a general term effectively encompassing a number of related but not identical methodologies that all exploit ideas from natural evolution and selection. The main approaches are *Genetic Algorithms*, *Evolution Strategies*, *Evolutionary Programming* and *Genetic Programming*. For practical reasons, we will concentrate on *Genetic Algorithms* (Gas) which are widely used [1], [2] although much of what we say can also be applied to the other techniques. Good references to related methods are [3], [4] and [5].

Evolutionary Algorithms have found increasing application to many problems in diverse areas such as hard function and combinatorial optimization, neural nets evolution, routing planning and scheduling, management and economics, machine learning and pattern recognition.

The next section gives a basic description of the functioning of genetic algorithms and the successive sections will be devoted to the presentation of some representative applications in the optimization field. We conclude by summarizing the present status of the field and the prospects for the future.

2. Genetic Algorithms Basics. The main components of a standard Genetic Algorithm are the following:

- a constant size *population* of individuals, usually randomly initialized.
- each individual represents a point in the search space for a given problem through a suitable *coding*.

- a *fitness value* is assigned to each individual in the population. This fitness measure plays the role of an environment.
- individuals are ranked and *selected* according to their fitness in such a way that more fit individuals are more likely to reproduce.
- genetic operators such as *crossover* and *mutation* are applied to pairs of individuals or to single individuals in order to produce new feasible solutions to a problem.

The elements that allow the genetic population to evolve are the selection process and the genetic operators. Selection can be done proportionally to fitness. This can be visualized as a spinning wheel having one slot per individual in the population with a slot size proportional to the individual's fitness. The wheel is spinned a number of times equal to the population size and each time an individual is selected to be a member of the next generation. Obviously, fitter individuals will have more chances of being reproduced. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. But selection alone cannot generate any new point in the search space. Here genetic operators enter into play. Let us assume for simplicity that binary coded strings are used to represent individuals, which is often the case but not always. For example, in a mathematical function optimization problem, a string of bits might represent a coordinate vector of a point in a domain of n-dimensional Euclidean space or it might stand for a sequence of numbers representing tasks in a task scheduling problem.

Once the new population has been produced, strings are paired at random and recombined through crossover. Here we will explain one-point crossover. Suppose that the following two strings have been selected for recombination:

0010011010 and 1110010001

a position is selected at random between 1 and the length of the string minus one, each position being equally likely. Suppose that position 6 has been chosen (marked by the vertical bar):

001001 | 1010

111001 | 0001

Then, after swapping all bits from position 6 to the end of the string one obtains two new strings called the *offspring*:

001001 | 0001

111001 | 1010

These two new individuals will enter the new population in place of their parents. Crossover is applied with a certain frequency called crossover rate p_c usually between 0.5 and 0.8. According to the theory ([1]), crossover directs the search towards promising regions.

After crossover, mutation can be applied to population members with a frequency p_m around 0.01. The usual interpretation of bit mutation rate is the following: for each string in the population and for each bit within the string generate a random number r between 0 and 1, if $r \leq p_m$ flip the bit. Mutation, is essentially background noise that is introduced to prevent premature convergence to local optima by randomly sampling new points in the search space

A typical GA is a procedure that iteratively repeats the above steps and since GAs are stochastic algorithms without converge guarantee, termination may be triggered by reaching a maximum number of iterations, called generations, or by finding an acceptable solution. The following general schema summarizes a standard genetic algorithm:

```
produce an initial population of individuals
while termination condition not met do
      evaluate the fitness of all individuals
      select fitter individuals for reproduction
      produce new individuals
      generate a new population by inserting some new good
      individuals and by discarding some old bad individuals
      mutate some individuals
end while
```

The above pseudocode is a sequential formulation of a GA. This has been the common approach until recently and it was justified by the

lack of adequate software and hardware and it made it easier to reason about mathematical properties of the GAs as well. However, it is clear that GAs offer many natural opportunities for parallel and distributed execution because many steps are independent. There are several possible parallel GA models, the most popular being the fine-grained or *grid* models, and the coarse-grain or *island* models. In the grid models, large populations of individuals are spatially distributed on a low-dimensional grid and individuals interact locally within a small neighborhood. Massively parallel SIMD machines are especially suited for the grid model. In the island model the population is subdivided into smaller subpopulations which evolve independently and simultaneously according to a standard GA. Periodic migrations of some selected individuals between islands allow to inject new diversity into converging subpopulations. Microprocessor-based multicomputers and workstation clusters are well suited for the implementation of this kind of parallel GA. Today, with the widespread availability of parallel architectures, parallel GAs are becoming more common. Parallel GAs are further discussed in refs. [6,7].

Applications to Optimization Problems. It has already been said in the introduction that Evolutionary Algorithms in general and GAs in particular have been widely applied to many different types of optimization problems. To give the flavor of that kind of usage of GAs, we will present here a summary of some representative applications to mathematical optimization, combinatorial optimization and to a real-life financial problem. A complete and useful bibliography of about 3000 papers on Evolutionary Algorithms can be obtained by anonymous ftp from ftp.uwasa.fi, directory cs/report94-1, as a number of compressed postscript files.

3.1. Unconstrained Global Function Optimization. Given a function *f(x)*, the global function minimization problem is to find x^* such that:

$$f(x^*) = \min\{f(x) \quad \forall x \in D\}, \quad D \in R^n, \quad f: R^n \to R$$

We consider a discrete subset S of points $x = (x_1, x_2, \ldots, x_n)^T$ of D. Each of the n coordinates x_i is represented by a binary string. The string is decoded as an unsigned integer and mapped into the corresponding real interval. The concatenation of the n coordinate substrings form a chromosome that is the representation of a particular point $x \in S$.

Test functions f_6 and f_7, f_8, and f_9 were used with various kinds of parallel Genetic Algorithms [7,8]. These are difficult multimodal test functions for which the location of the global minimum is known. They are defined as:

$$f_6(x) = \sum_{i=1}^{6} \left[x_i^2 - \cos(18 x_i) \right],$$

$$f_7(x) = \sum_{i=1}^{10} -x_i \sin(\sqrt{|x_i|}),$$

$$f_8(x) = \sum_{i=1}^{10} x_i^2 / 4000 - \prod_{i=1}^{10} \cos(x_i / \sqrt{i}),$$

$$f_9(x) = -\sum_{i=1}^{m} 1/(x - A(i))(x - A(i))^T + c_i,$$

With $x_i \in [-5.12, 5.12]$ for f_6, $x_i \in [-500, 500]$ for f_7, $x_i \in [-600, 600]$ for f_8 and $x_i \in [-500, 500]$ for f_9. Function f_8 is particularly hard.

Parallel Evolutionary Algorithms have proved to be very effective on these difficult problems which were all easily solved. Although more specialized and faster methods exist for mathematical optimization, Evolutionary Algorithms have the advantage of being easier to understand and use for general problem solving. Besides, they offer real advantages for noisy, discontinuous and non-differentiable functions.

3.2. Combinatorial Optimization. There exist many hard combinatorial optimization problems whose worst-case time complexity is bound by an exponential function of the instance size. Therefore, only small size instances can be solved exactly and approximate methods that are not guaranteed to find the globally optimal solution are the only possibility for

larger instances. Many important industrial and management decisions imply hard optimization such as routing, planning and scheduling problems. Here we briefly describe an easily stated but difficult to solve problem, the goal being to bring about the key issues of Genetic Algorithms for combinatorial optimization in a simple context.

The following number partitioning problem ([9]) is to be solved: n numbers x_i, $(i = 1, \ldots, n)$ are to be partitioned into K groups in such a way that the differences between the group sums are minimized. The number of partitions grows with K^n.

There are various ways of encoding a solution to this problem for the genetic algorithm ([9]). The simplest one consists in encoding each partition as a string of integer numbers

$$(p_1, p_2, \ldots, p_n) \quad \forall p_i \in \{1,2,\ldots,K\}, \quad i = 1,2,\ldots,n,$$

which means that the first number in the string belongs to group p_1, the second to group p_2 and so on.

This is called *group number* encoding and can make use of standard crossover and mutation operators; however, it has a mayor problem: an offspring may contain less than K groups. For example, with $n = 6$ and $K = 3$ the following parents: (1 2 1 1 3 2) and (1 2 3 3 1 2), assuming that the crossover point is position two from the right, would give rise to the offspring (1 2 1 1 1 2) and (1 2 3 3 3 2), where in the first offspring the 3rd group is empty. These individuals are obviously unacceptable and "repair" techniques or penalty functions are needed to treat them. There exists much better encodings for this problem ([2],[9]) which guarantee that only legal individuals will ever be produced. This requires a definition of crossover and mutation that is tailored to the problem at hand ([2]). However, even with the naive representation, a fine-grained massively parallel algorithm implemented on a Connection Machine CM-200 was able to obtain better results than the standard sequential method [10].

We took as our test problem the same problem as in [9], with n=34 numbers to be distributed into K=10 groups, for which the size of

the search space is 10^{34}. The algorithms were run 30 times each for a maximum number of 70 generations and we found the best results by using an unusually high mutation frequency. This seems to relieve convergence to local minima in this problem by randomly reshuffling the population. With the use of mutation we were able to find a partition with an objective function value of 690, which is about three times better than the results of [9] with the same encoding.

3.3. Portfolio Selection with Distributed Gas. The central problem in portfolio selection is to find a number of assets and their weights in such a way that a certain measure of risk is minimized for any given level of expected return on investment. Classically, risk is measured as the standard deviation of the variance of the probability distribution of future returns. In this framework, quadratic programming is used for solving the portfolio selection problem. More recent approaches are based on semivariance and downside risk, which roughly means that investors only perceive risk below the mean of the distribution of returns. For these new models deterministic algorithms such as quadratic programming are not very useful. There are tens or even hundreds of assets in a given portfolio and the risk-return surface is no longer convex, as when variance is used, but it becomes a very rugged, non-convex, highly multimodal landscape. When deterministic algorithms for optimization fall short, stochastic and heuristic methods may become attractive. Genetic algorithms where thus used to solve the portfolio allocation problem in ref. [11].

Choosing an optimal portfolio can be viewed as a multi-objective optimization problem in that an investor wants to minimize risk while maximising expected return. As the level of risk increases, the expected return attached to optimal portfolios draws a convex non-decreasing curve, which is called efficient frontier, which is the set of Pareto-optimal, i.e. non-dominated, portfolios. In other words, on the efficient frontier, a larger expected return corresponds to a greater risk. This can be expressed as a two-objective optimization problem in the parameters region **w**:

$$\min_{\mathbf{w}} \{\text{Risk}(\mathbf{w})\}$$

$$\max_{\mathbf{w}} \{\text{Return}(\mathbf{w})\},$$

subject to

$$\sum_{w_i \geq 0} w_i = 1$$

These two objectives can be parametrized to yield a parametric objective

$$\min_{\mathbf{w}} \{\lambda \ \text{Risk}(\mathbf{w}) - (1-\lambda)\text{Return}(\mathbf{w})\},$$

where parameter λ is a trade-off coefficient ranging between 0 and 1. When $\lambda = 0$ the investor disregards risk and only seeks to maximize expected return; when $\lambda = 1$ the risk alone is minimized, whatever the expected return. Since there is no general way to tell which particular trade off between risk and return is to be considered the best, optimizing a portfolio means finding a whole range of optimal portfolios for all the possible values of the trade off coefficient; the investors will thus be able to choose the one they believe appropriate for their requirements. A natural way to achieve that in an evolutionary setting is to have several distinct populations evolve for a number of trade-off coefficient values. The greater this number, the finer the resolution with which the investor will be able to explore the efficient frontier. Because it is likely that slight variations in the trade-off coefficient do not significantly worsen a good solution, a natural way to sustain the evolutionary process is to allow migration or cross-breeding between individuals belonging to populations corresponding to values of the trade-off coefficient that are close together. This suggests a distributed implementation where populations are linearly arranged according to their relevant trade-off value.

Distributed GAs can be used to speedup the search on a cluster of workstations. The population topology used was a two-way string of processors in which exchange of individuals only takes place between nearest neighbours i.e., those with similar trade-off coefficients. Very good results have been found in [11] for portfolios with up to 150 assets. A comparison with a previous sequential solution showed that the parallel version was not only obviously faster, it also converged on the average towards better solutions in all cases over many different portfolios.

4. Conclusions. In this paper we have described the basic ideas behind Evolutionary Algorithms and, in particular, Genetic Algorithms. We have shown that satisfactory solutions to difficult problems, both in the academy and in real-life, can be found through EAs. Intelligence is a by-product of evolution ([4]) and thus it is likely that more flexible and adaptable software and hardware systems will be possible through the use of simulated evolutionary techniques. Although this strand of ideas had come to a nearly total break in the sixties and seventies, thanks to a few researchers and groups, we are today in a resurgence phase which holds promises for the future. Present hardware and software technology are much more suitable for evolutionary computing. For example, parallel computing is mature enough to allow routine implementation of Evolutionary Algorithms of unprecedented complexity on a variety of architectures. This in turn allows to tackle more difficult problems since Genetic Algorithms are a relatively slow technique.

On the hardware side, some recent research shows that, in principle, simulated evolution can also be applied to actual machines and circuits [12]. Machines that are capable of evolving their own architecture as a function of a task that can also vary, like a natural environment, are an astonishing prospective. For the time being, little can be said about the feasibility of this so-called 'evolutionary engineering' ([12]) approach. But new and more powerful reconfigurable hardware and forthcoming nanotechnologies might make this possible on a larger scale. This is a new and promising avenue that has to be further explored.

REFERENCES

[1] G. Goldberg,. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Second Edition, Berlin, 1994.

[3] T. Bäck, F. Hoffmeister and H.P. Schwefel, *A Survey of Evolution Strategies*, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991.

[4] D.B. Fogel, *Evolutionary Computation*, IEEE Press, New York, 1995.

[5] J. Koza, *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[6] V.S. Gordon and D. Whitley, *A Machine-Independent Analysis of Parallel genetic Algorithms*, Complex Systems, 8, 181, 1994.

[7] H. Mühlenbein, M. Schomish and J. Born, *The Parallel Genetic Algorithm as Function Optimizer*, Parallel Comput. 17, 619, 1991.

[8] M. Tomassini, *The Parallel Genetic Cellular Automata: Application to Global Function Optimization*, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer-Verlag, Wien, 385, 1993.

[9] D. R. Jones and M. A. Beltramo, *Solving Partitioning Problems with Genetic Algorithms*, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991.

[10] M. Tomassini, *Cellular Evolutionary Algorithms for Mathematical and Combinatorial Optimization*, Proceedings of the 2nd European Connection Machine Users Meeting, Paris, 1993.

[11] A. Loraschi, A. Tettamanzi, M. Tomassini and P. Verda, *Distributed Genetic Algorithms with an Application to Portfolio Selection Problems*, in *Proceedings of the Int. Conf. On Artificial Neural Nets and Genetic Algorithms*, D. W. Pearson, N. C. Steele and R. F. Albrecht (Editors), Springer-Verlag, 384, 1995.

[12] H. de Garis, *CAM-BRAIN, The Evolutionary Engineering of a Billion Neuron Artificial Brain by 2001 which Grows/Evolves at Electronic Speeds inside a Cellular Automata Machine*, in *Proceedings of the Int. Conf. On Artificial Neural Nets and Genetic Algorithms*, D. W. Pearson, N.C. Steele and R. F. Albrecht (Editors), Springer-Verlag, 84, 1995.

ALGORITMOS GENÉTICOS: UNA NUEVA PERSPECTIVA PARA EL DISEÑO DE REDES NEURALES

J. Fernando Vega Riveros*
Harold Molina**
Pontificia Universidad Javeriana
Departamento de Ingeniería Electrónica.
Santafé de Bogotá D. C. Colombia

Resumen. Uno de los principales problemas ante los cuales se encuentran los diseñadores de redes neurales es la escogencia de la topología de estas. Generalmente el diseño de la topología se ha realizado por medio de pruebas de ensayo y error, generando una gran pérdida de tiempo en la búsqueda de una topología altamente eficiente. Se presenta un método para automatizar la búsqueda de la, o las, topologías más eficientes para la resolución de un problema determinado. El criterio de búsqueda está basado en el proceso de la evolución natural, y es conocido como los algoritmos genéticos. Mediante este proceso se diseñaron perceptrones multinivel utilizando el algoritmo de retropropagación del error para solucionar tres problemas característicos en los sistemas de reconocimiento de patrones como son la *o* exclusiva, el reconocimiento de patrones binarios, y señales análogas.

Abstract. A problem neural networks designers face is to choose an adequate topology. Often, this choice is based on trial and error, which leads to prolonged design times. In this paper we present a method to automate the search fo the most eficient topology or group of topologies and associated parameters to solve a given problem. The search criterion is based on a method inspired in the natural evolution process known as genetic algorithms. Multilayer perceptrons, trained with the error backpropagation algorithm, were designed using genetic algorithms. Three problems were used to test our approach: the exclusive or (EXOR), binary and analog pattern recognition.

Palabras Claves: entrenamiento genético, algoritmos genéticos.

1. Introducción.

Los algoritmos genéticos (AG) y las redes neurales (RN), inspirados en procesos biológicos, han encontrado amplia aplicación en problemas de aprendizaje de máquinas y optimización. En el caso particular de las redes neurales, uno de los

* e-mail fvega@javercol.javeriana.edu.co
** e-mail hmolina@venus.javeriana.edu.co

problemas que más requiere experiencia e intuición es la determinación de los parámetros topológicos y de aprendizaje de la red, tales como número de niveles, número de neuronas por nivel, tasa de aprendizaje, etc. Este es un problema de optimización de gran interés por la posible existencia de múltiples soluciones y las intrincadas relaciones entre los parámetros.

Dependiendo de la dificultad del problema y la experiencia del diseñador en algunos casos se puede optar por utilizar la solución que primero funciona aunque no lo haga en la forma más óptima posible o se abandona la aproximación de redes neurales completamente. Los AG han encontrado aplicación en una variedad de problemas de optimización similares donde a través de un proceso evolutivo encuentran un conjunto de soluciones subóptimas (Goldberg, 1989; Marshall, 1991). Este conjunto puede ser usado por el diseñador para escoger la solución que más convenga a su aplicación o simplemente quedarse con la mejor ofrecida por el algoritmo. Este proceso no solamente simplifica el problema de diseño sino que ofrece una metodología más sistemática.

Se presenta en este artículo una metodología basada en algoritmos genéticos para determinar los parámetros básicos de diseño de redes neurales eficientes del tipo *feedforward* y las cuales utilizan el algoritmo de entrenamiento conocido como retropropagación del error (*error backpropagation*). Se discuten los resultados de la aplicación de los AG al diseño de RN para tres problemas representativos.

2. Descripción del problema. Para aplicar el proceso de entrenamiento genético a una red neural, se deben definir las características de la red tales como tipo de red neural, topología, algoritmo de entrenamiento y las características que se utilizarán como elementos del cromosoma que representen a los distintos individuos durante el entrenamiento genético.

Para este trabajo se decidió utilizar redes neurales multinivel entrenadas con el algoritmo de retropropagación del error (*error backpropagation*) para tres problemas diferentes:

1. La *o* exclusiva (EXOR)
2. Reconocimiento de imágenes binarias
3. Reconocimiento de señales analógicas

El primer problema ha sido discutido ampliamente en la aplicación de redes neurales (Minsky, 1969). El segundo problema es típico de reconocimiento de imágenes monocromáticas, mientras el tercero es importante en el análisis de señales. Estos problemas son representativos de las aplicaciones más populares de las redes neurales *feed forward* multinivel.

3. Redes neurales. Las redes neurales, inspiradas en el funcionamiento del sistema nervioso, son arreglos de procesadores elementales interconectados entre si. Cada uno de estos procesadores o neuronas suma las entradas ponderadas y su resultado pasa a través de una función no lineal conocida como función de activación. Las entradas a una neurona pueden provenir de fuentes externas o de otras neuronas en la red. Así mismo la salida de una neurona es enviada a otras neuronas o al entorno.

Las neuronas de una red neural artificial pueden organizarse en estratos o niveles, y dependiendo de la presencia de caminos de retroalimentación se pueden clasificar como recurrentes o no recurrentes. Las redes neurales no recurrentes y en particular las *feedforward* han sido ampliamente utilizadas en una variedad de problemas debido a su estabilidad inherente y la simplicidad del algoritmo de entrenamiento de retropropagación del error (Haykin, 1994).

En las redes neurales multinivel las neuronas del primer nivel reciben sus entradas directamente del entorno. Las salidas de estas neuronas se conectan a las entradas de las neuronas del segundo nivel, y así sucesivamente, hasta el último nivel, cuyas salidas son entregadas al entorno.

4. Algoritmos genéticos. Los algoritmos genéticos (AG) se basan en la supervivencia de los más aptos, i.e., menor valor de una función de costo, y utiliza operadores tales como reproducción, réplica, mutación, *cross-over* y selección. El problema se resuelve

caracterizando la solución al problema como unas cadenas de información a las cuales se les denomina cromosomas. En nuestro caso particular, los cromosomas representan diferentes características de la topología de la red neural, v.g., número de niveles, número de neuronas por nivel, etc.. La reproducción o réplica consiste en copiar un cromosoma de acuerdo con una medida de aptitud: mientras mayor es la aptitud mayor es la probabilidad de reproducción. La mutación consiste en cambiar aleatoriamente el valor de alguna característica dentro del cromosoma. La mutación se aplica de acuerdo con una distribución de probabilidad. El *cross-over* consiste en el cruce de las características de dos individuos en alguna parte de los cromosomas padres. Este cruce también se efectúa de acuerdo con una distribución de probabilidad conjunta de los padres. Después de aplicar los anteriores operadores genéticos, se seleccionan probabilísticamente los sobrevivientes de acuerdo con la medida de aptitud o una función de costo seleccionada.

En nuestra aproximación se utilizan cromosomas de longitud variable los cuales contienen en forma codificada el número de niveles, el número de neuronas por nivel, la tasa de aprendizaje y el coeficiente del *momentum*. La estructura del cromosoma se muestra a continuación:

```
struct { float learning;
         float momentum;
         unsigned int nlayer;
         unsigned int* nneurons },
```

donde *learning* es la tasa de aprendizaje, *momentum* es el valor de dicho parámetro, *nlayer* representa el número de niveles de la red y *nneurons* almacena el número de neuronas por nivel. Una generalización de esta estructura permitiría almacenar valores diferentes de tasas de aprendizaje y *momentum* por cada nivel.

Para la aplicación del operador de mutación se utilizaron dos distribuciones estadísticas: la distribución uniforme y la triangular. En el primer caso, la mutación es independiente de la medida de aptitud de los individuos. En el segundo caso, la probabilidad de mutación es mayor para los individuos con menor medida de aptitud. Esta segunda distribución busca evitar que el desempeño de los mejores individuos se deteriore por la mutación, mientras que a los individuos menos aptos se les da mayor probabilidad de cambiar sus características y con ello mejorar el promedio de desempeño de la población.

El operador de reproducción o réplica no se utilizó porque se aseguró que de generación en generación se mantuviera la población constante con los mejores individuos, i.e., elitismo.

Se probaron dos operadores de *cross-over*. El primero fue el operador de *cross-over* uniforme, en el cual, se escoge aleatoriamente una máscara binaria que indica de que padre se toma esa característica del cromosoma; así por ejemplo, en los puntos de la máscara donde el valor es 0 uno de los hijos hereda las características del padre 1 y en los de valor 1, las características del padre 2. El operador funciona en forma inversa para el otro descendiente. En una prueba posterior se utilizó el operador *enhanced ordered cross-over* (EOX) (Yi, 1991). El operador EOX selecciona aleatoriamente dos posiciones de los cromosomas padres. El primer padre se deja sin alteraciones y se hace una copia de trabajo del segundo padre. Esta copia se sujeta a una rotación hasta que en el segundo punto de *cross-over* coincidan los valores de los cromosomas. El hijo se obtiene tomando la parte central del primer padre y completando las posiciones restantes con elementos de los dos extremos de la copia de trabajo del segundo padre que no estén en el centro. En la figura 1 se muestra un ejemplo. En la selección, para mantener una población constante, se utilizó una probabilidad de supervivencia la cual se aplicaba en orden ascendiente de aptitud, *i.e.*, de menor a mayor costo, hasta que los malos individuos que sobrevivieran, sumados a los mejores individuos de la generación anterior completaran el número de individuos deseado. Por ejemplo, supóngase que se desea tener una población constante de 50 individuos. Supóngase también que hay 75 herederos después de aplicar los operadores de mutación y cross-over. Si sobrevivieran 2 individuos con valor alto de la función de costo, la población se completaría con los 48 mejores individuos.

2o. padre	f	j	k	l	|	m	n	o	d	|	a	b	h
1er. padre	a	b	c	d	|	e	f	g	h	|	i	j	k
					|					|			
					|	co	pi	a	↓	|			
2o. padre	l	m	n	o	|	d	a	b	h	|	f	j	k
hijo	m	n	o	d	|	e	f	g	h	|	j	k	l

Figura 1

5. Implementación en una estación de trabajo. Los problemas descritos fueron implementados originalmente sobre una estación de trabajo DEC ALPHA 3000 300L con un reloj de 120 MHz. Los procesos tienen un tiempo de ejecución del orden de 16 horas CPU para procesos de 40 generaciones, cada una de 40 individuos. Para cada individuo en cada generación se limitó el número máximo de iteraciones a 10000 o a lograr un Error Cuadrado Medio (MSE) de 0.01% utilizando el algoritmo de retropropagación del error. Por esta razón, se utilizó como función de costo la siguiente:

$$c = \overline{e^2} i \qquad (1)$$

donde c representa el costo, $\overline{e^2}$ representa el error cuadrático medio e i representa el número de iteraciones para convergencia. Esta función de costo mide por lo tanto dos criterios para calificar la eficiencia de una red neural, favoreciendo la supervivencia de aquellos individuos que combinen el entrenamiento más corto con el más bajo error.

Es importante resaltar que el tiempo real de este proceso es superior a 16 horas dependiendo del grado de utilización de la estación de trabajo en otros procesos y por otros usuarios.

6. Resultados. En los tres problemas que se utilizaron para probar nuestra aproximación se obtuvieron una variedad de topologías en diferentes ejecuciones. Las redes variaban desde 3 a 7 niveles y desde 2 hasta 8 neuronas por nivel. Como era de esperarse, tanto el número de niveles como el número de neuronas por nivel dependían del problema. Por ejemplo, en el caso de la operación EXOR se obtuvo repetidamente una topología de 3 niveles con 2 neuronas en el nivel oculto. En el caso de los problemas de reconocimiento de patrones se lograron soluciones en un espectro más amplio lo cual muestra que para un mismo problema de suficiente complejidad pueden existir múltiples soluciones de desempeño equiparable. Este espectro de soluciones no habría sido fácilmente obtenible por las técnicas tradicionales de diseño de redes neurales.

El AG también exploró una amplia variedad de soluciones lo cual se manifestó en que el mejor individuo tuviera un desempeño muy superior al promedio de la población. Un ejemplo de esto se muestra en la Figura 2.

La versión secuencial del sistema, dependiendo del problema tomó entre 2 y 16 horas de CPU para su ejecución.

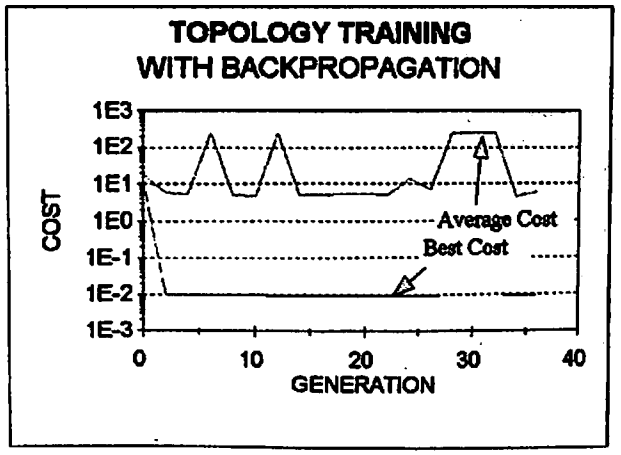


Figura 2

7. Conclusiones. Se mostró una novedosa metodología para el diseño de redes neurales *feedforward* basada en algoritmos genéticos. Esta metodología ofrece un acercamiento sistemático al diseño que no depende tanto de la experiencia o la intuición del diseñador y que le ofrece un espectro de alternativas equiparables.

Esta metodología puede ser extendida en varias direcciones. Por un lado, los cromosomas pueden incluir otros parámetros adicionales como tasa de aprendizaje, o momentum por nivel. Se pueden incluir restricciones adicionales que puedan mejorar el desempeño computacional de la red diseñada o facilitar su implementación, v.g., restringir el número total de niveles, o de neuronas por nivel. La metodología puede también utilizarse para el diseño de redes neurales recurrentes y utilizar otros algoritmos de aprendizaje.

REFERENCIAS

D. Goldberg, 1898. *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, Reading,MA, USA.

S. Haykin, 1994. *Neural Networks: Comprehensive Foundation,* Macmillan, New York, NY. USA.

V. Maniezzo, 1994. *Genetic Evolution of The Topology and Weight Distribution of Neural Networks,* IEEE Transactions on Neural Networks, Vol 5, No. 1, January, pp. 39-53

S.J. Marshall, R.F. Harrison, 1991. *Optimization and training of feedforward neural networks by genetic algorithms,* IEEE Second International Conference on Artificial Neural Networks, London, UK, 39-43.

M.L. Minsky, S.A. Papert, 1969. *Perceptrons,* MIT Press, Cambridge, MA, USA.

S. Yi, L. Gou-Lie, 1991. *New Crossover operators in genetic algorithms,* Third International Conference on Tolls for Artificial Intelligence TAI 91, IEEE Computer Society Press, Los Alamitos, CA, USA.

COLECCION JULIO CARRIZOSA VALENZUELA

Volumen 1 - **Castillo, G.** 1992. Física Cuántica, teoría y aplicaciones. Tomo primero. 1/16 xxxii + 410, 77 figs.

Volumen 2 - **Bernal de Ramírez, I.** 1993. Análisis de Alimentos. 1/16 XVIII + 314 pp, 28 figs.

Volumen 3 - **Castillo, G.** 1994. Física Cuántica, teoría y aplicaciones. 1/16. Tomo segundo, XX+406 pp, 45 figs.

Volumen 4 - **Cáceres, D.** (Editor). 1995. Creando ciencia Crean docencia. 1/16. 140 pp, 38 figs.

Volumen 5 - **Romero, C. M. & Blanco L. H.** 1996. Tópicos en Química Básica. 1/16. 240 pp, 56 figs.

COLECCION MEMORIAS

Volumen 1 - Memorias del Seminario Nacional "El quehacer teórico y las perspectivas holista y reduccionista 1/16. VIII + 184 pp.

Volumen 2 - Memorias del Seminario Konrad Lorenz sobre Etología. 1/16 IV + 38 pp.

Volumen 3 - Memorias del Seminario Taller sobre Alta Montaña Colombiana. 1/16. 116 pp.

Volumen 4 - Primer Congreso Nacional de Neurocomputación.

Las anteriores publicaciones pueden ser solicitadas directamente a la Academia,
Apartado **44763**, Santafé de Bogotá, D.C., o Fax **(571) 2838552.**
E.Mail accefyn@colciencias.gov.co
Se ofrecen en venta o mediante intercambio por publicaciones similares.

Este libro se terminó de
imprimir el día 9 de mayo de 1996
en los talleres gráficos de
Editora Guadalupe Ltda.,
Santafé de Bogotá, Colombia


IN TERRA
AD SIDERA
CODATZII
CABAVITO