REDES NEURONALES APLICADAS AL RECONOCIMIENTO DE PATRONES EN ECOGRAMAS

Gonzalez, R.[1], Rodriguez, L. [1], Fernández, V. [1], Perichinsky, G. [2]y García Martínez, R. [3,2]

(1) Laboratorio de Sistemas Inteligentes.
Departamento de Computación.
Facultad de Ingeniería.
Universidad de Buenos Aires.
Paseo Colon 850. 4to Piso. (1063)
Capital Federal. Argentina.

(2) Laboratorio de Base de Datos y Sistemas Operativos
Departamento de Computación.
Facultad de Ingeniería.
Universidad de Buenos Aires.
Paseo Colon 850. 4to Piso. (1063)
Capital Federal. Argentina.

(3) Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS). Escuela de Postgrado
InstitutoTecnológico de Buenos Aires
Madero 399. (1106) Buenos Aires.
Argentina
rgm@itba.edu.ar

Resumen.

En este trabajo se discute la implementación una red neuronal para reconocimiento de patterns de ecogramas. La red elegida para la implementación fue la red de Hopfield, la cual fue entrenada con distintos tamaños y cantidades de patrones.
En la primera parte del trabajo se da una breve introducción a las características, configuración, aprendizaje y funcionamiento de la red de Hopfield. Luego, se explica la aplicación desarrollada y se exponen los detalles y resultados obtenidos. Finalmente, se extraen las conclusiones y futuras lineas de investigación del trabajo realizado.

1 - El modelo de Hopfield.

1.1 - El problema de la memoria asociativa.

El problema de la memoria asociativa es el problema del 'átomo de Bohr ' del campo de la computación. Ilustra de la forma más simple posible en que manera la computacion colectiva puede funcionar.
El problema básico es el siguiente: Guardar un conjunto de p patrones ξ_i^μ de tal forma que cuando se le presenta un nuevo patrón ζ_i, la red responde produciendo el patrón que más se le parezca de los patrones almacenados.
Los patrones son etiquetados por μ = 1, 2, ... , p, mientras que las unidades de la red son etiquetadas con i = 1, 2, 3 ...., N. Tanto los patrones almacenados ξ_i^μ como los patrones de testeo ζ_i pueden tomar los valores 0 o 1 para cada i, a pesar de que adoptaremos una convención diferente en poco tiempo.
Podríamos resolver este problema serialmente en una computadora convencional simplemente guardando una lista de los patrones ξ_i^μ y escribiendo un programa que calcule la distancia de Hamming,

$$\sum_i \left[\xi_i^\mu \cdot \left(1-\zeta_i\right) + \left(1-\xi_i^\mu\right) \cdot \zeta_i \right] \qquad (1.1)$$

entre el patrón de prueba ζ_i y cada uno de los patrones almacenados, encontrando cual de ellos es el de menor distancia e imprimiendo el patrón correspondiente.
Aqui queremos ver como logramos ésto mediante una red de McCullough-Pitts. Esto es, si comenzamos en la configuración $n_i = \zeta_i$, queremos saber cual es el

conjunto de pesos (si lo hay) W_{ij} que van a llevar a la red al estado con $n_i = \xi_i^{\mu o}$, donde el patrón μ_o es el que tiene la menor distancia de Hamming de ζ_i . Por lo tanto queremos que la memoria sea ***direccionable por contenido*** e insensible a pequeños errores en los patrones de entrada.
Una memoria direccionable por contenido puede ser muy poderosa. Supongamos, por ejemplo, que almacenamos información codificada sobre varios científicos famosos en una red. Luego el patrón inicial "evolución" debería ser suficiente para traer toda la información acerca de Darwin, y "$E=mc^3$" debería traer la información de Einstein, sin tener en cuenta el error en el patrón de entrada. Notemos que *algún* patrón va a ser recuperado siempre (a menos que inventemos un patrón de no saber); la red no debería nunca retornar una combinación lineal de sus patrones almacenados, digamos de Darwin y Wallace en respuesta a "Evolución" pero va a elegir el que mejor matchee de acuerdo a lo que fue almacenado. Esto depende en la no-linealidad de la red y obviamente tiene ventajas para muchas aplicaciones prácticas.
Otros ejemplos comunes de aplicaciones para una memoria asociativa son el reconocimiento y la reconstrucción de imágenes, y la búsqueda de información bibliográfica partiendo de referencias parciales.

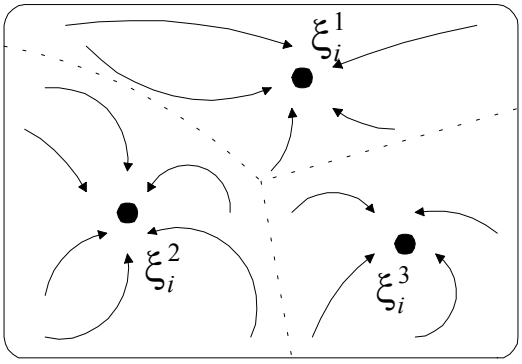

Fig. 1 - Espacio de configuración esquemática de un modelo con tres atractores.

La figura 1 muestra esquemáticamente la función de las memorias dinámicas asociativas (direccionables por contenido) que utilizamos en este trabajo. El espacio de todos los posibles estados de la red - el **espacio de configuración** - está representado por la región dibujada. Dentro de esa región los patrones almacenados son **atractores**. La dinámica del sistema lleva a los puntos iniciales a uno de los atractores, como muestran las trayectorias dibujadas. El espacio completo de configuración es dividido en zonas de atraccion de los diferentes atractores. La figura está muy idealizada, y en particular el espacio debería ser realmente un conjunto discreto de puntos (en un hipercubo), no una región continua. Pero es una imágen útil para recordar.

1.2 - El modelo.

Por conveniencia matemática vamos a transformar la formulación a una donde el valor de activación de las neuronas sean +1 (activas) y -1 (no activas) en vez de 1 y 0. Las vamos a notar como S_i en vez de n_i. La conversión de notación se realiza mediante $S_i = 2\ n_i - 1$. La dinámica de la red correspondiendo a (1.1) o a (1.3) ahora es :

$$S_i := \mathrm{sgn}\left(\sum_j W_{ij} \cdot S_j - \theta_i \right) \qquad (1.2)$$

donde la función sgn es la función signo:

$$\mathrm{sgn}(x) = \begin{cases} 1 \ \text{si x} \geq 0; \\ -1 \ \text{si x} < 0; \end{cases} \qquad (1.3)$$

y el umbral θ_i se relaciona con μ_i mediante:

$$\theta_i = 2\mu_i - \sum_i W_{ij}$$

En el resto del trabajo desechamos estos términos de umbral tomando a $_i = 0$, debido a que no son útiles en los patrones que se van a considerar. Por lo tanto utilizamos:

$$S_i := \mathrm{sgn}\left(\sum_j W_{ij} \cdot S_j \right) \qquad (1.4)$$

Hay por lo menos dos formas en las cuales podemos realizar la actualización indicada por la ecuación anterior. Podemos hacerla *sincrónicamente*, actualizando todas las neuronas simultáneamente en cada paso de tiempo. O podemos realizarlo en forma *asincrónica*, actualizándolas una por vez. Ambos tipos de modelos son interesantes, pero la actualización asincrónica es más natural para el modelo de cerebro biológico y el modelo de redes artificiales. La elección de una actualización sincrónica requiere un clock central, y es potencialmente sensible a los errores de timing.

En el caso asincrónico, que vamos a adoptar, podemos proceder de dos formas:

> En cada paso del clock, seleccionar al azar una neurona i para actualizar y aplicar la regla (1.4).
>
> Dejar que cada unidad independientemente elija actualizarse a sí misma de acuerdo a (1.4), con alguna probabilidad constante por unidad de tiempo.

Estas elecciones son equivalentes (excepto por la distribución de intervalos de actualización) porque la segunda opción da como resultado una secuencia aleatoria; hay una probabilidad muy pequeña de que dos unidades se decidan a actualizarse en el mismo instante. La primer opción es apropiada para simulación (utilizada en este trabajo), con control central, mientras que la segunda es apropiada para unidades de hardware autónomas.
También tenemos que especificar por cuánto tiempo, es decir por cuántas actualizaciones vamos a permitirle a la red que evolucione antes de pretender que los valores de sus neuronas nos devuelvan el patrón requerido. Una posibilidad en el caso de actualización sincrónica es requerir que la red vaya al patrón correcto luego de la primera iteración. En este trabajo se utilizó un modo de actualización asincrónico por lo que se pretende sólamente que la red converja a una situación estable en la cual los S_i no cambien.

1.2.1 - Un patrón.

Para fundamentar la elección de los pesos de interconexión vamos a analizar primero el caso en el que hay simplemente un sólo patrón ξ_i que queremos memorizar. La condición para que este patrón sea estable es :

$$\mathrm{sgn}\left(\sum_j W_{ij} \xi_j \right) = \xi_i \qquad \forall \mathrm{i} \qquad (1.5)$$

ya que entonces la regla (2.4) no produce más cambios. Es fácil ver que esto es cierto si tomamos:

$$W_{ij} \ \alpha \ \xi_i \xi_j \qquad (1.6)$$

ya que $\xi_j^2 = 1$. Para nuestra posterior conveniencia tomamos a la constante de proporcionalidad como 1/N, donde N es el número de neuronas en la red, con lo que nos queda :

$$W_{ij} = \frac{1}{\mathrm{N}} \ \xi_i \xi_j \qquad (1.7)$$

Además, también es obvio que aunque un número (menor que la mitad) de bits del patrón inicial Si sean erróneos (esto es que son distintos de $_j$), éstos van a ser sobrepuestos en la suma de la entrada neta :

$$h_i = \sum_j W_{ij} S_j \qquad (1.8)$$

por la mayoría que son correctos, y sgn(h_i) va a dar aún ξ_j. Una configuración inicial cercana (en distancia de Hamming) a ξ_j va a converger rápidamente a ξ_j. Esto significa que la red va a corregir errores como deseábamos, y podemos decir que el patrón ξ_j es un **atractor**.
Realmente hay dos atractores en este caso simple; el otro atractor está en -$_j$. Este es llamado un **estado invertido**. Todas las configuraciones iniciales con más de la mitad de bits diferentes del patrón original van a converger al estado

invertido. El espacio de configuración es simétricamente dividido en dos zonas de atracción, como se puede ver en la figura 2.

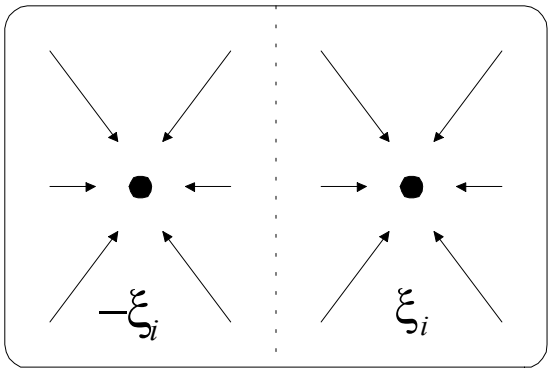

Fig 2 - Espacio de configuración esquemática para el caso de un patrón, incluyendo el estado reverso

1.2.2 - Muchos Patrones

Lo que acabamos de analizar es correcto para un patrón, pero cómo hacemos para que el sistema converja al patrón más similar al ingresado entre muchos patrones almacenados ? La respuesta es simplemente hacer a W_{ij} una superposición de términos como (1.7), uno para cada patrón:

$$W_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^{\mu}\xi_j^{\mu} \qquad (1.9)$$

donde p es el número total de patrones almacenados etiquetado por μ.
Esta fórmula es usualmente llamada "*regla de Hebb*" o "*regla generalizada de Hebb*" debido a la similitud entre (1.9) y la hipótesis hecha por Hebb [1949] sobre la forma en que la fuerza de las conexiones sinápticas en el cerebro cambia en respuesta a la experiencia: Hebb sugirió que los cambios son proporcionales a la correlación entre los disparos de las neuronas pre- y post-sinápticas. Si aplicamos nuestro conjunto de patrones ξ_j^{μ} a la red durante la **fase de entrenamiento**, y ajustamos los pesos Wij de acuerdo a dichas correlaciones pre/post, llegamos directamente a la fórmula (1.9). Técnicamente, de cualquier forma, (1.9) va más allá de la hipótesis original de Hebb debido a que cambia los pesos positivamente cuando ninguna de las neuronas está disparando ($\xi_j^{\mu} = \xi_j^{\mu} = -1$). Ésto probablemente no sea razonable desde el punto de vista fisiológico. La ecuación (1.9) puede también causar que una conexión particular cambie de excitatoria a inhibitoria o viceversa a medida que se agregan más patrones, algo que no podría ocurrir nunca en sinapsis reales. Es posible modificar la ecuación de varias formas para remediar estos defectos, pero en este trabajo la vamos a utilizar en su forma más simple.
Un modelo de memoria asociativa usando la regla de Hebb (1.9) para todos los posible pares i j, con neuronas binarias y actualización asincrónica es llamado **modelo de Hopfield**.

Examinemos ahora la estabilidad de un patrón particular ξ_j^ν. La condición de estabilidad (1.5) se generaliza de la siguiente forma:

$$\mathrm{sgn}\left(h_i^\upsilon\right) = \xi_i^\upsilon \qquad \forall \mathrm{i} \qquad (1.10)$$

donde la entrada neta h_j^ν a la neurona i en el patrón ν es:

$$h_i^\upsilon \equiv \sum_j W_{ij}\xi_j^\upsilon = \frac{1}{N}\cdot\sum_j\sum_\mu \xi_i^\mu \xi_j^\mu \xi_j^\upsilon \qquad (1.11)$$

Ahora separamos la sumatoria en μ en el término $\mu = \nu$ y todo el resto :

$$h_i^\upsilon = \xi_i^\upsilon + \frac{1}{N}\cdot\sum_j\sum_{\mu\neq\upsilon} \xi_i^\mu \xi_j^\mu \xi_j^\upsilon \qquad (1.12)$$

Si el segundo término fuese cero, podríamos concluir inmediatamente que el patrón ν es estable de acuerdo a (1.10). Ésto es cierto también si el segundo término es lo suficientemente pequeño : si su magnitud es menor a 1 no puede cambiar el signo de h_j^ν , y la ecuación (1.10) va a ser satisfecha.
Podemos ver que el segundo término, que llamaremos término cruzado, es menor que 1 en muchos casos de interés si p (que es el número de patrones) es lo suficientemente pequeño. Vamos a discutir los detalles en el próximo punto; por ahora asumamos que este término es lo suficientemente pequeño para todos los i y ν. Por lo tanto los patrones almacenados son estables, es decir, si comenzamos con el sistema en uno de ellos, éste se va a quedar ahí. Además, una fracción de bits diferentes de los almacenados en los patrones van a ser corregidos de la misma forma que en el caso del patrón único; éstos son superados en la sumatoria $\sum_j W_{ij}S_j$ por la vasta mayoría de bits correctos.

Por lo tanto una configuración cercana (en distancia de Hamming) a ξ_j^ν converge a ξ_j^ν . Ésto muestra que los patrones elegidos son realmente atractores del sistema, como habíamos anticipado en la figura 1.2. El sistema por lo tanto trabaja como esperamos, como una memoria direccionable por contenido.

1.3 - Estados Espúreos.

La regla de Hebb (1.9) nos da (para p suficientemente pequeños) un sistema dinámico que posee atractores - mínimos locales de la función de energía - en los puntos deseados x_i^m , los cuales suelen llamarse ***estados recuperables***. Pero lo que no se ha mostrado es que estos son los únicos atractores. Y en realidad lo que sucede es que hay otros.
En primer lugar los estados revertidos - x_i^m son mínimos y tienen la misma energía que los patrones originales. La dinámica y la función de energía ambas tienen una simetría perfecta, S_i « $-S_i$ para todos los i. Esto no es demasiado problemático para los patrones recuperables; se podría acordar revertir todos los bits restantes cuando el signo de un bit en particular es -1.

En segundo lugar, encontramos estados de mezcla -***mixture states*** - x_i^{mix} estables, los cuales no son iguales a un único patrón, sino que corresponden a combinaciones lineales de un número impar de patterns. Los más simples de estos son combinaciones simétricas de tres patterns almacenados
Las ocho combinaciones de signos son posibles pero se considerará el caso donde todos los signos se eligen +. Los otros casos son similares. Se puede observar que en promedio x_i^{mix} tiene el mismo signo que x_i^{m1} ; solo si ambos x_i^{m2} y x_i^{m3} tienen el signo opuesto, el signo total puede ser revertido. Por lo tanto x_i^{mix} está a una distancia de Hamming N/4 de x_i^{m1} y por supuesto de x_i^{m2} y x_i^{m3} también. Los estados mezcla se sitúan a puntos equidistantes de sus componentes. Esto también implica que $\sum_i \xi_i^{\mu_1} \xi_i^{mix} = N/2$ en promedio.

Por lo tanto la condición de estabilidad (1.10) se sigue satisfaciendo por el mixture state. De una forma similar 5,7,... patterns pueden combinarse. El sistema no elige un número par de patterns porque ellos pueden sumar cero en algunos lugares mientras que la neurona solo tiene + o - 1.
En tercer lugar, para p grande hay mínimos locales que no están correlacionados con ningún número finito de los patterns originales. Éstos suelen ser llamados ***spin glass states*** debido a su gran correspondencia con los modelos de los spines del vidrio de la mecánica estadística.
Se puede concluir que la memoria no funciona perfectamente, ya que se encuentran todos estos mínimos adicionales además de los que uno busca. La segunda y tercera clases son generalmente llamadas ***mínimos espúreos***. Por supuesto, sólo se caerá en uno de ellos si se comienza en una zona cercana a los mismos. Además estos estados tienden a tener zonas de atracción más pequeñas si las comparamos a los estados recuperables.

2 - Aplicación e Implementación de la red Neuronal.

El objetivo de este trabajo es crear un sistema de reconocimiento de patterns de ecogramas. Para ello se utilizó la red neuronal descripta en la sección 2. La idea es que una vez entrenada la red neuronal se le presenten a ésta patterns de ecogramas con ruido o distorsionados. La red deberá reconocerlos en el conjunto de patterns aprendidos. Si bien éste es el objetivo de este trabajo, para el entrenamiento de la red no se utilizaron patterns de ecogramas reales sino patterns arbitrarios ingresados manualmente. Se utilizaron patterns de distintos tamaños y en distintas cantidades :

- 10 por 10. - Conjuntos de 3 patterns.
- 10 por 10. - Conjuntos de 5 patterns.
- 10 por 10. - Conjuntos de 10 patterns.
- 15 por 15. - Conjuntos de 3 patterns.
- 15 por 15. - Conjuntos de 5 patterns.
- 15 por 15. - Conjuntos de 10 patterns.
- 20 por 20. - Conjuntos de 5 patterns.
- 20 por 20. - Conjuntos de 10 patterns.

En todos los casos se trabajó con patterns de 4 colores. Para ello se utilizó una codificación de dos bits que requería dos neuronas por cada elemento del patrón.

Se desarrollo un sistema de interacción con el usuario que permite crear nuevas secuencias de patterns, grabarlas para ser utilizados posteriormente e imprimirlas, ya sea por pantalla o impresora. El funcionamiento del software es el siguiente: una vez cargada una secuencia de patterns (ya sea nueva o levantada del disco), se ingresa el pattern con ruido a reconocer y la red comienza a iterar para lograr su objetivo. La finalización de esta etapa de funcionamiento puede darse de dos formas: una es presionando una tecla y la otra es cuando el número de iteraciones alcanza un máximo definido en el programa (50.000 iteraciones).

3 - Conclusiones.

Las conclusiones que se pueden extraer del trabajo realizado son las siguientes:

- El funcionamiento de la red neuronal fue bueno cuando la cantidad de patrones a aprender era 3. Este comportamiento se verifico en los casos de patrones de 10 por 10, 15 por 15, 20 por 20.

- El funcionamiento de la red cuando la cantidad de patrones era 5 fue en algunos casos aceptable mientras que en otros fue malo. Creemos que se debió principalmente a la correlación existente entre los patrones a memorizar. En el punto teórico de este trabajo se demostró cuál era la capacidad de aprendizaje de patrones aleatorios no correlacionados; pero en este caso la codificación utilizada provocó que los patrones fueran correlacionados con lo que aparecieron estados espúreos estables. Habría que analizar de que forma codificar la información para reducir la correlación entre distintos patrones.

- El funcionamiento de la red cuando la cantidad de patrones era 10 fue pésimo. Aparecieron gran cantidad de estados espúreos que impidieron el aprendizaje de los patrones ingresados.

Podemos agregar que para las distintas cantidades de patrones a memorizar en general se verificó que la red degradaba su funcionamiento a medida que se aumentaba el tamaño de los patrones.

Como última conclusion cabe decir que esta implementación no fue exitosa para lograr el objetivo deseado, con lo que habría que analizar bien si otro tipo de red o modelo de aprendizaje no tendría un mejor comportamiento para este caso.

4 - Futuras lineas de investigación.

En la realización de este trabajo surgieron las siguientes líneas de investigación:

- Estudiar una codificación más eficiente con el objetivo de lograr que los patrones ingresados no estén correlacionados. De esta forma se completaría el trabajo con redes de Hopfield antes de descartar totalmente este tipo de red para esta aplicación.

- Analizar si otro tipo de red, como podría ser la Back-Propagation no tendría mejor performance para este tipo de problemas.

5 - Bibliografía.

1. Introduction to the theory of neural computation. John Hertz, Anders Krogh and Richard G. Palmer. Ed. Addison-Wesley Publishing Co.

2. Neural Networks - Algorithms, Applications and Programming techniques. James A. Freeman, David M. Skapura. Ed. Addison-Wesley Publishing Co.