

Redes de Hopfield

Gonzalo V. Castiglione, Alan E. Karpovsky, Martín Sturla

Estudiantes Instituto Tecnológico de Buenos Aires (ITBA)

12 de Mayo de 2012

Entrega Final - Informe

Resumen—El presente informe busca analizar la implementación de una Red de Hopfield con el fin de analizar su funcionalidad y comportamiento como memoria asociativa direccionable por el contenido.

Palabras clave—memoria asociativa direccionable por el contenido, hopfield, imágenes, memorización, ruido, atractores, estados espúreos

I. INTRODUCCIÓN

Se analizó el comportamiento de una Red de hopfield como memoria asociativa direccionable por el contenido. El trabajo consistió en la implementación una Red de Hopfield en el lenguaje *Java*. Se tomó un conjunto de patrones a memorizar los cuales pueden visualizarse en la sección **Anexo A: Patrones** y se realizaron numerosas pruebas con el fin de evaluar cuestiones como ser si los patrones son verdaderos atractores, cuál es la máxima cantidad de patrones que la red puede memorizar, qué sucede si se le da como entrada a la red un patrón con ruido, etc.

Si bien el algoritmo de hopfield es asíncrono, se decidió implementar una variante sincrónica del mismo para comprar los resultados obtenidos con cada uno (modelo de Little).

En las siguientes secciones se explicará el modelado del problema, como así también las mejoras implementadas, los resultados obtenidos y las conclusiones que se puedan extraer de ellos.

II. DESARROLLO

A. Modelado del problema

Algunas definiciones preliminares:

- Ψ : Conjunto de patrones a memorizar
- N : Longitud de los patrones de entrada (ej. $N = 64$ si cada imagen es de 8×8 pixels)
- p : Cantidad de patrones distintos que contiene Ψ

Se representó a la Red de Hopfield como una clase abstracta debido a la distinción antes nombrada del

algoritmo **sincrónico** versus el algoritmo **asincrónico** (permite hacer implementaciones concretas que extiendan de ella).

La clase contiene dos variables: un vector y una matriz que están definidos como sigue:

- **float[N][N] weights**: Matriz de pesos W_{ij} correspondientes a los patrones que se desean memorizar.
- **int[N] states**: Vector con los estados de cada una de las N neuronas.

Nótese que si bien el vector de estados es de tipo entero, los estados definidos para este problema son 1 o -1 dado que las unidades que están siendo utilizadas son **bipolares**

Por otra parte se decidió representar a los patrones μ como vectores de enteros *int[] pattern* los cuales contienen un 1 en la posición i en el caso de que el i -ésimo pixel del patrón sea negro o contienen un -1 en la posición i en el caso de que el i -ésimo pixel del patrón sea blanco.

Todos los patrones de entrada fueron *normalizados*, es decir que las imágenes cuyo mapa de color estaba en escala de grises fueron pasadas a imágenes blanco y negro; asimismo una de las imágenes que no contenía fondo (fondo transparente) fue modificada dejando intacta la imagen pero agregándole un fondo blanco.

B. Inicialización de la red

Para los métodos ofrecidos públicamente por la red, se utilizó la convención dada en clase. Se escribió un método *storePatterns* inicializa la matriz de pesos sinápticos de la red neuronal utilizando la regla del producto externo de *Hebb*. Nótese que la matriz de pesos, una vez inicializada, no cambia durante toda la ejecución del algoritmo. Por cuestiones de eficiencia y dado que la matriz de pesos es simétrica con $W_{ij} = 0$, el algoritmo recorre sólo la diagonal superior de la misma y **espeja** los resultados.

C. Función de activación

La función de activación utilizada es la función signo.

D. Actualización de estados

Como ya se comentó, se implementaron dos variantes para la actualización de estados: el Modelo de Little y el Modelo de Hopfield los cuales se desarrollarán a

12 de Mayo, 2012.

continuación. En líneas generales ambos buscan iterar hasta la convergencia.

Para el caso del modelo de Little, la convergencia está dada cuando el vector de estados permanece invariante respecto del vector en el paso anterior o se detecte un ciclo de longitud dos en las actualizaciones del vector de estados.

Para el caso de Hopfield, se considera convergente al estado si tras k iteraciones el vector no ha cambiado. Se decidió tomar $k = 5N$ donde N es la longitud del patrón. Esto se hizo dado que si queda únicamente un pixel para actualizar, la probabilidad de no elegirlo en aN iteraciones es $(\frac{N-1}{N})^{aN}$. Si se considera N suficientemente grande, esta probabilidad tiende a e^{-a} . Para nuestro particular de $a = 5$ y $N = 4096$, esta probabilidad equivale a 0,7%, lo cual fue considerado aceptable.

Ambas son equivalentes excepto por la distribución del intervalo de actualizaciones. El modelo de Hopfield da una secuencia aleatoria por lo cual hay baja probabilidad de que dos unidades elijan ser actualizadas en el mismo momento.

La primera es útil para simulaciones y la segunda es apropiada para unidades de hardware autónomas.

Es importante destacar que el método de iteración hasta la convergencia puede caer en **ciclos de longitud 2** por lo que es sumamente importante no sólo chequear que el vector de estados no haya cambiado respecto del paso anterior sino también hay que hacer esta verificación respecto de 2 pasos anteriores.

D.1 Sincrónica (Modelo de Little)

En cada paso se actualizan todas las neuronas de forma simultánea de acuerdo a la siguiente regla:

$$S_i(n+1) = \text{sgn} \left(\sum_{j=1}^N W_{ij} S_j(n) \right) \quad (1)$$

D.2 Asincrónica (Modelo de Hopfield)

En cada paso se actualiza una unidad por ves; la misma es seleccionada de forma completamente aleatoria y se utiliza la ecuación ?? para la actualización.

D.3 Bit de paridad

Se agrega al vector de estados un bit en la última posición inicializado en uno, para indicar que no es un patrón inverso. De esta manera, luego de calcular el vector de estados, si el bit de paridad se encuentra en -1 , simplemente se invierte el patrón; caso contrario esto puede indicar un patrón inicial o un estado espúreo.

D.4 Matriz pseudo-inversa

La idea detrás de esta corrección de los pesos es que los patrones no siempre pueden ser aleatorios, como se asume

en algunos cálculos de la capacidad teórica de una red. Claramente en el caso de las imágenes provistas por la cátedra, los patrones no son azarosos. Muchos de ellos están correlacionados, por ejemplo, dos patrones en donde predomine el negro en zonas similares.

El problema detrás de los patrones correlacionados es que el término crosstalk entre estos es mayor. Esto hace que los estados atractores tengan menor estabilidad, y además puede aumentar la cantidad de estados espúreos. Como explica Hertz en su libro de teoría neuronal, para patrones correlacionados la red puede no aprender patrones incluso cuando la cantidad de patrones es significativamente menor a la cantidad de neuronas. La solución es usar una matriz psuedo-inversa.

Por su parte, el cálculo de esta matriz pseudo-inversa está dado por:

$$w_{ij} = \frac{1}{N} \sum_{\mu\nu} \xi_i^\mu (Q^{-1})_{\mu\nu} \xi_j^\nu \quad (2)$$

siendo

$$Q_{\mu\nu} = \frac{1}{N} \sum_i \xi_i^\mu \xi_i^\nu \quad (3)$$

Para el cálculo de la inversa de las matrices,

Es fácil ver que la matriz Q será invertible si los patrones son linealmente independientes. Por lo que esta implementación presenta dos problemas, Por un lado, si los patrones no son linealmente independientes, no se puede invertir la matriz. Hertz explica que la manera de subsanar dicho problema es utilizar un subconjunto linealmente independiente de los patrones que sea generador del mismo espacio. En el trabajo, se decidió simplemente usar la inversa de Moore-Penrose de la matriz Q , que en particular será la inversa si los patrones son linealmente independientes. En el caso de que no lo sean, al menos existe dicha matriz, a pesar de que el comportamiento de la red será indeterminado.

El segundo problema de la psuedo-inversa consiste en que el cálculo de una matriz inversa es muchas veces numéricamente inestable. Para intentar subsanar este problema, se utilizó la librería *JAMA* para su cálculo (único uso).

III. RESULTADOS

A. Verdaderos atractores

B. Estados espúreos

El siguiente ejemplo muestra la respuesta de la red al intentar memorizar un número grande de patrones. Se utilizó una red asincrónica y se la hizo memorizar los archivos “foot”, “line1”, “line2”, “line3”, “line4”

A continuación se muestra el resultado de pasarle como entrada la imagen “foot” pero incompleta. En este caso particular se optó por borrar la mitad izquierda de la

imagen:



Figura 1: Respuesta de una red de hopfield asincrónica ante un patrón incompleto.

C. Patrones con ruido o incompletos

El siguiente ejemplo muestra la respuesta de la red ante un patrón incompleto. Se utilizó una red asincrónica y se la hizo memorizar los archivos “foot”, “line1”, “line2”, “line3”, “line4”.

A continuación se muestra el resultado de pasarle como entrada la imagen “foot” pero incompleta. En este caso particular se optó por borrar la mitad izquierda de la imagen:



Figura 1: Respuesta de una red de hopfield asincrónica ante un patrón incompleto.

Si en cambio se utiliza solamente ruido, en una red asincrónica que memorizó los patrones “f”, “h”, “a”, “foot-print” y le pedimos que evolucione partiendo de “f”, la red rápidamente converge a la imagen deseada:



Figura 1: Respuesta de una red de hopfield asincrónica ante un patrón incompleto.

D. Patrones invertidos

En el siguiente ejemplo se muestran estados intermedios de la evolución de una Red de Hopfield asincrónica a la cual se la iniciacilizó con las siguientes imágenes provistas por la cátedra: “line1”, “line2”, “line3”, “line4”, “h”.

La red neuronal en cuestión utiliza la mejora del bit de paridad como así también la inicialización a través de la matriz pseudo-inversa.

Utilizando la red neuronal antes descrita, se le pasó como entrada la imagen “h” invertida y con ruido. A continuación vemos su evolución; la primer imagen es el estado inicial, luego se ven dos estados intermedios y por último el final:



Figura 1: Visualización de estados intermedios para una red de hopfield asincrónica con la mejora del bit de paridad y la utilización de la matriz pseudo-inversa

E. Patrones desconocidos

El siguiente ejemplo muestra la respuesta de la red ante un patrón desconocido. Se utilizó una red asincrónica y se la hizo memorizar los archivos “line1”, “line2”, “line3”, “line4”.

A continuación se muestra el resultado de pasarle como entrada la imagen “h” la cual la red no conocía:



Figura 2: Respuesta de una red de hopfield asincrónica ante un patrón desconocido.

F. Cantidad de patrones almacenables

En una primer instancia, se intentó que la cantidad de patrones almacenables en la red sea similar a la capacidad sugerida teóricamente por Hertz en su libro, que corresponde a aproximadamente $0,138N$. Esto correspondería a más de 500 imágenes. Sin embargo esto no fue observado empíricamente.

La razón detrás de esta discrepancia entre lo teórico y lo práctico surge de asumir varias hipótesis al deducir dicho valor para la capacidad. En un primer plano, el valor surge de asumir que todos los patrones son azarosos. Como ya se ha explicado en la sección de matrices psuedo-inversas, este no es el caso para las imágenes tratadas; el término de crosstalk es mucho mayor en este caso que en el caso de tener patrones azarosos. En segundo lugar, el cálculo se refiere únicamente a la estabilidad en los estados atractores; surge de exigir que el error en los mismos sea menor a un 1%. Sin embargo no hace referencia a la posibilidad de que los patrones de entrada converjan a un estado espúreo, lo cual se observó seguido en la práctica al usar más de 6 o 7 imágenes.

El número exacto de cantidad de patrones máximos almacenables es difícil de determinar, dado que depende de qué patrones se usen. Patrones con mayor correlación (mayor término de crosstalk) por lo general traían mayores problemas. Además, se observó que la implementación psuedo-inversa tuvo resultados consistentemente mejores que la regla de Hebb.

IV. CONCLUSIÓN

ANEXO A: PATRONES



Figura 1: Visualización de los 25 patrones con la que se inicializa a la red