Variaciones sobre el modelo de Hopfield: dilución extrema de las conexiones

Introducción

Motivaciones

Es bien sabido en el ámbito de las neurociencias que el modelo de Hopfield† ofrece una solución real al problema de la memoria asociativa‡. Sin embargo la idea original no es ni por mucho perfecta. Algunos de los aspectos negativos más contundentes son:

a) Extrema conectividad: cada neurona de la red está conectada con todas las demás. Esto es biológicamente ilógico y computacionalmente ineficiente. Recordemos que ésta característica era de utilidad para la faceta analítica del modelo, a la hora de demostrar que es posible realizar P*P (y no N*N) iteraciones para actualizar la red. Aquí 'P' es el # de patrones atractores (*memorias*) y 'N' el # de neuronas.

b) Sinapsis simétrica: esta es otra característica del modelo “tradicional” de Hopfield que carece de sentido biológico. Era una cualidad buena para el cálculo analítico, pues permite demostrar que H(**S**,t) es una función decreciente y por ende que el modelo converge.

c) Memorias independientes: si hay demasiada superposición entre los patrones atractores las redes de Hopfield comienzan a perder capacidad de reconocimiento. Sin embargo en la realidad las memorias pueden estar muy correlacionadas.

d) Los atractores son puntos fijos, lo que para la biología y para ciertos sistemas tecnológicos es muy malo

e) $\alpha < 0.138$ es un límite demasiado bajo para el # de memorias que la red permite almacenar manteniendo la capacidad de reconocimiento (recordemos que $\alpha = P/N$)

† http://hebb.mit.edu/courses/9.641/2002/lectures/lecture16.pdf
‡ http://www.bcp.psych.ualberta.ca/~mike/Pearl_Street/Dictionary/contents/A/associative_memory.html

Elección

En el presente trabajo se trató de contrarrestar la 1[ra] de estas complicaciones, es decir la conectividad extrema de las neuronas de la red. En particular se desarrolló una alternativa al modelo tradicional conocida como *Hopfield ultra diluido*. En ella sólo una fracción infinitesimal de las conexiones originales permanece.

Definiendo a 'K' como el *nivel de conectividad media,* es decir “el # promedio de (otras) neuronas a las que está conectada cada neurona”, puede pensarse en NxN variables C_{ij} con el siguiente comportamiento:

$$C_{ij} = \begin{cases} 0 \ , \ si\ no\ hay\ señal\ de\ la\ neurona\ \boldsymbol{j}\ hacia\ la\ neurona\ \boldsymbol{i} \\ 1 \ , \ si\ el\ peso\ w_{ij} \neq 0,\ es\ decir\ si\ \boldsymbol{j}\ transmite\ hacia\ \boldsymbol{i} \end{cases} \quad \Rightarrow \quad K = \left\langle \sum_{j \neq i}^{N} C_{ij} \right\rangle \quad (0)$$

La red neuronal es rediseñada usando estas nuevas variables con el propósito de imprimir sobre ella la naturaleza deseada sobre las conexiones. Para ello no es necesario modificar la dinámica de red (1), sino que basta con redefinir los pesos sinápticos (2):

$$S_i(t+1) = \mathrm{sgn}(h_i(t)) \quad , \quad donde \quad h_i(t) = \sum_{j \neq i}^{N} w_{ij} S_j(t) \qquad (1)$$

$$w_{ij} = \frac{1}{K} C_{ij} \sum_{\mu=1}^{P} \xi_i^{\mu} \xi_j^{\mu} \qquad (2)$$

Es posible probar que la superposición m_ν del estado **S** de la red con una memoria almacenada $\boldsymbol{\xi^\nu}$ satisface: $m_\nu = \int d\eta\ P(\eta)\ \mathrm{sgn}(m_\nu + \eta) = \mathrm{erf}\left(\frac{m_\nu}{\sqrt{(2\alpha')}}\right)$. Pero para ello hay que hacer dos suposiciones: **a)** la “dilución” de ij y de ji son independientes, es decir C_{ij} no depende de C_{ij}, y por ende la matriz $\{w_{ij}\}$ se vuelve asimétrica; **b)** estamos en el régimen estrictamente diluido $K \ll N$, donde en particular basta con asegurar que $K < \ln(N)$

Todas estas modificaciones a la red hacen que el parámetro α=P/N ya no sea de utilidad. La capacidad máxima de almacenamiento de memorias p_{max} es siempre proporcional al # promedio de unidades a las que una neurona está conectada. Por consiguiente lo que nos interesa ahora es el valor de α'=P/K. En *Hopfield ultra diluido* se obtiene un valor límite de reconocimiento $\alpha_c' = 2/\pi \simeq 0.636619772$

Puede verse entonces que si bien el objetivo original era el de disminuir la extrema conectividad de la red, en el desarrollo surgieron (o fue necesario asegurar) condiciones que contrarrestan otros aspectos negativos de Hopfield tradicional. Específicamente estamos hablando de la simetría de las conexiones sinápticas, eliminada para asegurar un resultado teórico, y el límite poco satisfactorio $\alpha_c \simeq 0.138$, superado por el nuevo $\alpha_c' \simeq 0.637$

Implementación

Consideraciones previas

El marco teórico no fue seguido al pie de la letra. Las transgresiones principales fueron dos: la dilución no fue probabilística y no se respetó a rajatabla la condición K≪N.

Lo primero hace alusión al hecho de que en lugar de generar las mentadas variables C_{ij} con métodos estocásticos asegurando que se satisfacía la condición (0), se fijó el # total de vecinos de todas las neuronas en K. Esto quiere decir que no había un "nivel de conectividad medio" = K, sino que cada neurona tenía exactamente K vecinos conectados a ella de los cuales recibía impulsos.

Las "condiciones atenuantes" de esta transgresión son que una neurona no puede ser vecina de sí misma, y que los vecinos nunca se repiten para una misma neurona. O sea que ∀i ∈ {1,...,N} se tiene j1 ≠ j2 ⇒ vecino(i,j1) ≠ vecino(i,j2), esto ∀j1,j2 ∈ {1,...,K}

Es posible demostrar que el comportamiento promedio de la red no varía considerablemente del descripto en la introducción a pesar de esta modificación. La ventaja de este método es la simplicidad que se gana a la hora de codificarlo, que además se traduce en rapidez de ejecución.

Lo segundo se debe a que la restricción K < ln(N) no fue satisfecha, lo que fue una consecuencia directa de la ley de obediencia debida (profe: favor de insertar su excusa aquí)

Experimentación

Para tres tamaños distintos de red* se ejecutó el algoritmo descripto por las ecuaciones (1) y (2). En cada caso se empleó una cantidad creciente de patrones

* N $\stackrel{\text{def}}{=}$ # de neuronas de la red = 2500, 5000, 10000

almacenados: se comenzó estudiando la evolución de la red con P $\stackrel{def}{=}$ # de memorias = 1, luego con P = 2, …, así hasta P = 20. Siempre se sortearon 20 vecinos por neurona.

Para cada valor de P se realizaron 50 corridas del algoritmo, para luego utilizar el valor medio muestral de la superposición m_ν en el ploteo de los resultados. En cada una de estas corridas se inicializó al estado **S** sobre un patrón atractor $\boldsymbol{\xi}^\nu$ de entre los P posibles, se dejó evolucionar la red sin mediciones durante 30 ciclos ("relaxation"), y finalmente se realizaron 100 ciclos bajo observación.

En cada uno de los últimos 100 ciclos se midió la superposición m_ν del estado **S** con el patrón $\boldsymbol{\xi}^\nu$ escogido. El promedio obtenido fue registrado. Esto se realizó 50 veces para cada P, y el promedio de esas 50 corridas es el valor impreso en los gráficos para cada valor de α'.

Resultados

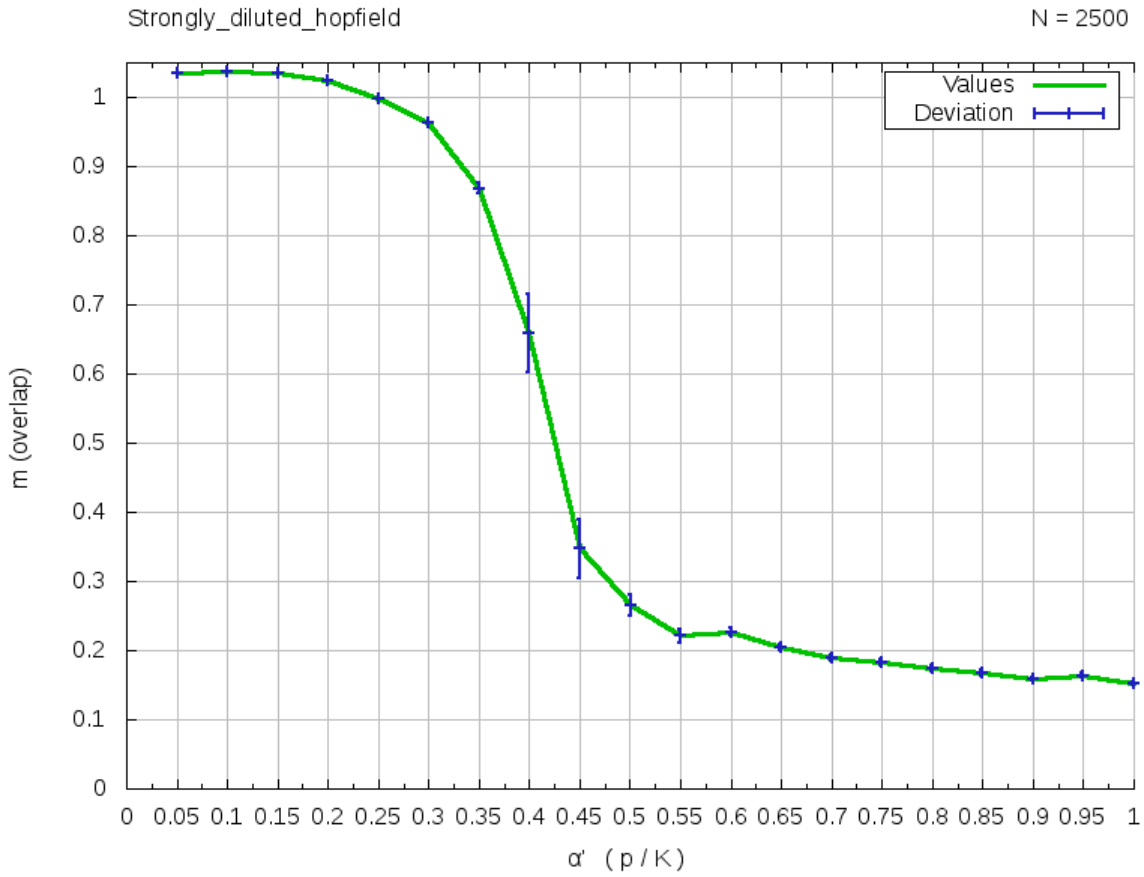

Gráfico 1: 2500 neuronas, 20 vecinos por neurona

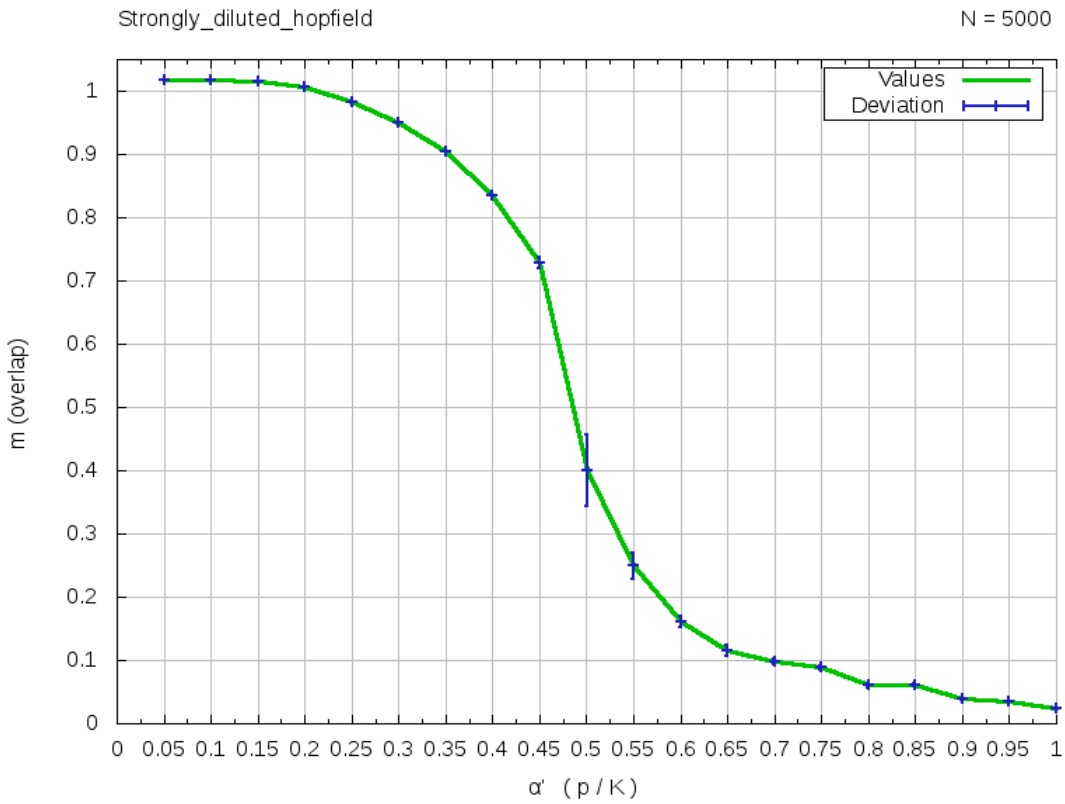

Gráfico 2: 5000 neuronas, 20 vecinos por neurona

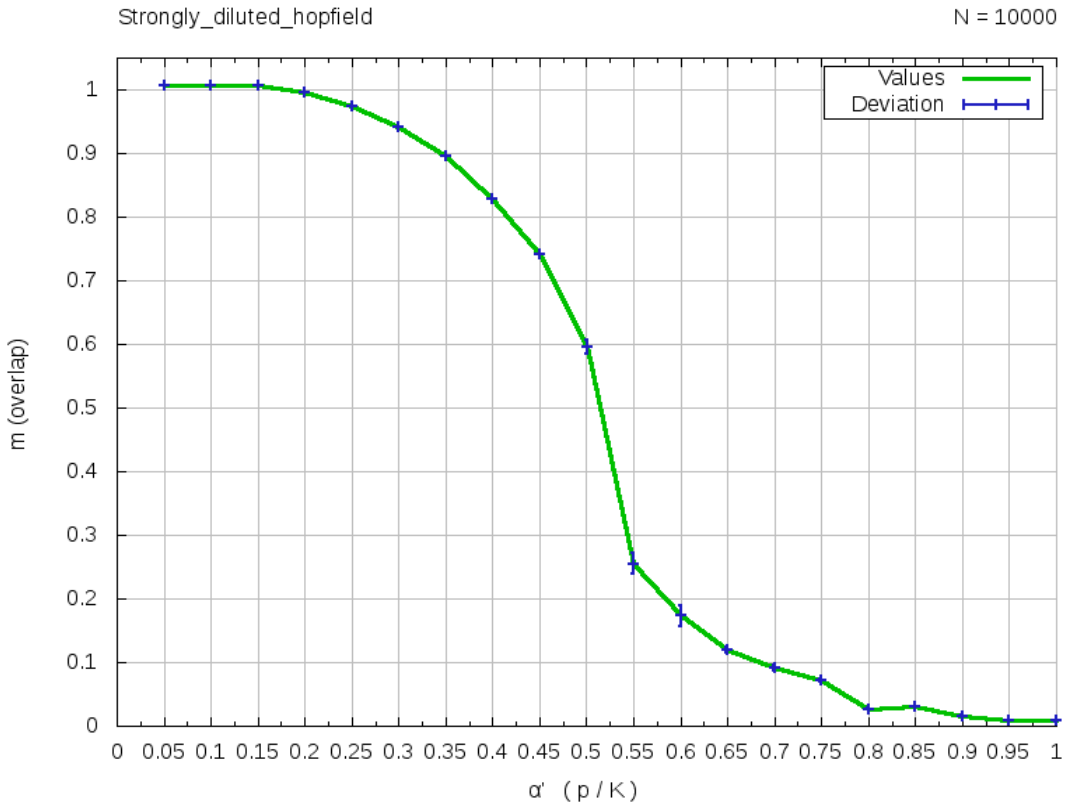

Gráfico 3: 10000 neuronas, 20 vecinos por neurona

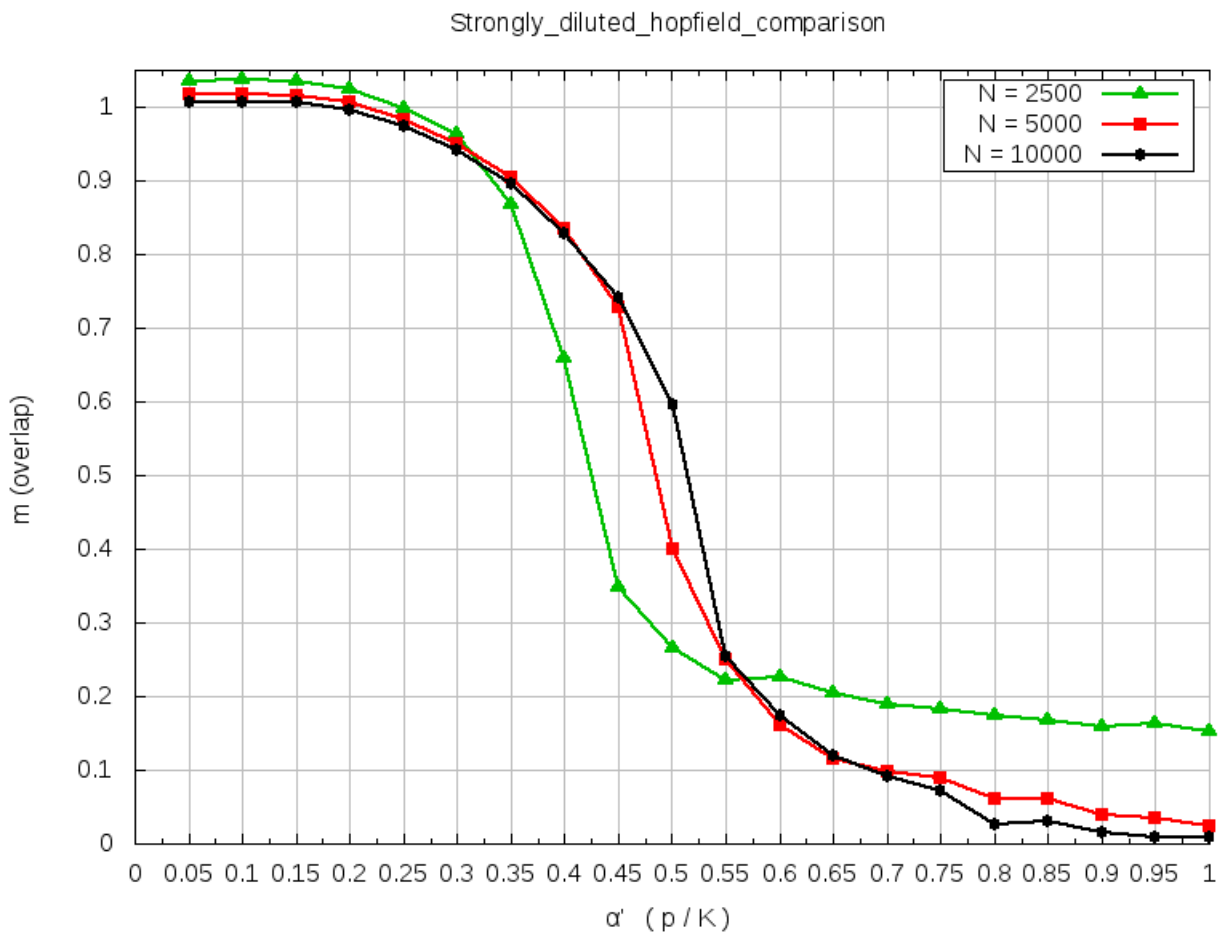

Gráfico 4: comparación de los tres tamaños de red

Análisis de resultados

Lo primero que salta a la vista al ver los resultados es que el comportamiento empírico de la red se acerca más a los valores teóricos a medida que aumenta el # de neuronas. El Gráfico 4 fue especialmente generado para observar tal comportamiento.

Lejos de significar un problema esto era de esperarse, ya que el análisis al que son sometidas las ecuaciones (1) y (2) en ningún momento considera valores chicos de N. De hecho la sección de la bibliografía que realiza el desarrollo teórico para el cálculo del valor de α'_c asegura que las aproximaciones utilizadas son válidas para grandes valores de K.

Por ende las patologías que podrían ocurrir cuando no se satisface que $N \gg 1$ se ven naturalmente minimizadas a medida que $N \to \infty$

También es posible apreciar que si bien se dijo que $\alpha_c' \simeq 0.637$, donde $\alpha' \stackrel{def}{=} P/K$, en la práctica el reconocimiento muy bueno ($m_\nu \cong 1$) se mantuvo sólo hasta valores de α' cercanos a 0.25. De ahí en adelante a medida que el # de memorias del sistema aumentaba, la red se desplazaba de la memoria inicial donde se la colocaba, cada vez con mayor frecuencia.

En el mejor de los casos (Gráfico 3, 10000 neuronas) cuando α' = 0.4 el "reconocimiento" de la red disminuye a valores de $m_\nu \cong 0.8$, y comienza a presentar un decaimiento que recuerda al exponencial. Ya cuando α' = 0.6 se tiene que $m_\nu \cong 0.2$, y de ahí en adelante la superposición entre **S** y la memoria $\boldsymbol{\xi}^\nu$ es prácticamente nula.

La razón aducida para explicar esta patología es el hecho de que con una conectividad media fija = 20 vecinos por neurona, incluso si N=10000 se tiene que 20 > ln(N) ≅ 9.21. Como no se respetó la restricción $K \ll N$, una de las principales suposiciones del marco teórico, era de esperar que los límites de reconocimiento obtenido fuesen inferiores al máximo calculado en la bibliografía.

Para confirmar esta hipótesis se volvió a correr el programa respetando el límite superior planteado para K. Se usaron 10000 neuronas y 9 vecinos por neurona, obteniendo resultados más similares a las predicciones teóricas, como se observa a continuación:

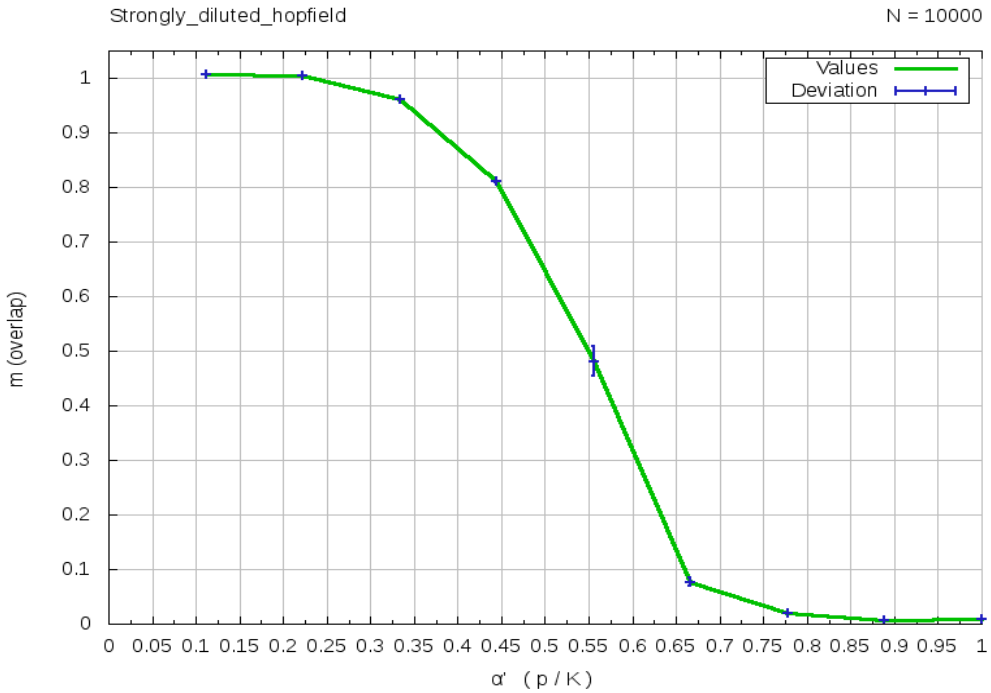

Gráfico 5: 10000 neuronas, 9 vecinos por neurona (K < ln(N))

Por último y a modo de cierre, el lector atento habrá notado que en los Gráficos 1 y 2, al comenzar las corridas de la red con valores bajos de P (con α' entre 0 y 0.25) ¡¡¡ se tiene una superposición $m_\nu > 1$!!!

Afortunadamente la explicación es sencilla: en la implementación, el código emplea un almacenamiento eficiente de los datos (en particular del estado **S** y las memorias $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^{P}$), guardando 1 elemento por bit, donde "elemento" se refiere a una neurona o un valor ξ_i^μ .

De esta manera es posible condensar la información, y minimizar la cantidad de memoria que la computadora emplea cuando está ejecutando el programa. Sin embargo si el N escogido no es un múltiplo del # de bits que ocupa una variable de tipo *long* en el ordenador donde se ejecuta el programa (64 bits en este caso), ocurren problemas.

Al almacenar tanto **S** como las memorias $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^{P}$ habrá siempre cierta cantidad de bits "de más", exactamente 64 – r, donde r = N%64 = resto de dividir N por 64. Por defecto (y por buenas prácticas de programación) estos bits siempre son reseteados, es decir que tienen valor = 0. Esto sucede en ambos lados: **S** y la memoria escogida $\boldsymbol{\xi}^\nu$, y por ende esos bits de más siempre contribuyen a aumentar el valor de m_ν. A medida que N crece r/N disminuye y el efecto total se va atenuando, hasta ser casi imperceptible como lo atestigua el Gráfico 3.

Conclusión

Estoy cansado. Me imagino que usted también. Vamos a dormir.