Trabajo práctico 1
Redes Neuronales y Aprendizaje Profundo

Ignacio Ezequiel Cavicchioli
Padrón 109428
icavicchioli@fi.uba.ar

10/9/2025

1 Introducción

Este documento presenta el desarrollo de las consignas del trabajo práctico N°1 de la materia de **Redes Neuronales y Aprendizaje Profundo**. El código correspondiente fue realizado en *Jupyter notebooks*, *Python*, adjuntados a la entrega en formato PDF. Toda imagen o implementación requeridas para el análisis se explicitarán en el presente archivo, por lo que la lectura del código en sí queda a discreción del lector. La teoría relevante será presentada y discutida en la sección pertinente.

2 Ejercicio 1

2.1 Consignas

Entrene una red de Hopfield ‘82 con las imágenes binarias disponibles en el campus.

a Verifique si la red aprendió las imágenes enseñadas.

b Evalúe la evolución de la red al presentarle versiones alteradas de las imágenes aprendidas: agregado de ruido, elementos borrados o agregados.

c Evalúe la existencia de estados espurios en la red: patrones inversos y combinaciones de un número impar de patrones. (Ver Spurious States, en la sección 2.2, Hertz, Krogh & Palmer, pág. 24).

d Realice un entrenamiento con las 6 imágenes disponibles. ¿Es capaz la red de aprender todas las imágenes? Explique.

2.2 Desarrollo

Una red de Hopfield es, en esencia, una memoria direccionable por contenido (*content-adressable memory*)[1]; permite obtener un estado memorizado completo a partir de un estado incompleto suficientemente parecido. Como se va a mostrar en los varios incisos, la red de Hopfield no es una memoria ideal, pero bajo ciertas condiciones, el error en la recuperación de estados está acotado.

La red de Hopfield a tratar es una colección de neuronas de McCulloch-Pitts. En pocas palabras, cada neurona tiene como "entrada" los estados de todas las otras neuronas menos sí misma. Estos se ponderan por un peso $w_{i,j}$ calculado por aprendizaje Hebbiano, se suman, conformando una h, que se pasa por una función de activación, determinando la "salida", o estado futuro, de la neurona. En los ejercicios se trabajó con una función de activación del tipo signo, que devuelve -1 o 1 según el signo de h. Para $h = 0$ se determinó que la salida toma valor 1.

La red es entrenada por medio del aprendizaje de todos los $w_{i,j}$ según la regla de aprendizaje (1), que se puede generalizar matricialmente como se ve en (2). A cada imágen le corresponde una única iteración ($\Delta w_{i,j}$), por lo que solo se suma una vez a la matriz $\mathbf{W}$. En términos más informales, la red solo "ve" las imágenes 1 vez. El peso η se mantuvo unitario para todo entrenamiento.

$$\begin{aligned} \Delta w_{i,j} &= \eta x_i x_j \\ w_{i,j_n} &= w_{i,j_{n-1}} + \Delta w_{i,j} \end{aligned} \tag{1}$$

$$\begin{aligned} \Delta \mathbf{W} &= \eta \mathbf{x}\mathbf{x}^T \\ \mathbf{W}_n &= \mathbf{W}_{n-1} + \Delta \mathbf{W} \end{aligned} \tag{2}$$

El proceso de aprendizaje equivale a aprender la correlación entre todas las imágenes. Imágenes muy parecidas (por algún criterio, ej.: distancia de Hamming) van a quedar "cerca" en la función energética subyacente. Esto trae a colación un aspecto importante para toda memoria, que es la capacidad. Como se va a demostrar más adelante, la capacidad es función de la cantidad de neuronas y correlación ente estados memorizados.

Para" ejecutar" la red neuronal alcanza con calcular todos los estados siguientes de las neuronas. Esto se puede hacer de dos formas:

- Actualización sincrónica: se computa el siguiente estado de todas las neuronas a la vez.
- Actualización asincrónica: las neuronas son aleatoriamente actualizadas, sin seguir un orden particular. La idea es que todas se actualicen al finalizar el ciclo.

[1] Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Campus, J. J. Hopfield

Para este trabajo se usó una actualización sincrónica, pero la actualización asincrónica tiene beneficios en lo relativo a convergencia de la red.

Ahora bien, ¿Cómo se parte de una matriz de pesos y se termina con un sistema que recuerda estados? El libro "*Introduction To The Theory Of Neural Computation*" provisto por el Campus de la materia explica que los estados memorizados son atractores en el espacio de todos los estados posibles. Una forma de pensarlo es como que los patrones o estados aprendidos son concavidades en una tela y el estado actual es una canica puesta sobre esta tela. Si se deja que la canica se mueva, va a caer en alguna de estas hendiduras de la tela. Como se va a tratar más adelante, existen otros atractores que no son los patrones aprendidos: los inversos de estos, las combinaciones lineales de un número impar de patrones (estados espurios de mezcla) y otros estados espurios que aparecen como consecuencia de la dinámica de la red.

Ahora que se tiene cierta noción sobre las redes de Hopfield se puede pasar al desarrollo de las consignas como tales. La primera pide verificar que la red entrenada haya aprendido las imágenes. Recordando, si los patrones fueron memorizados, serían atractores, y la red se mantendría en ellos aunque se itere, lo cual constituye una forma simple de verificar el aprendizaje.

Por un tema de las dimensiones de las imágenes, los tres primeros incisos se realizaron con dos redes neuronales de tres imágenes cada una, para luego unificar los tamaños en el cuarto. Las figuras 1 y 2 muestran, para cada subfigura, la imagen enseñada (izquierda) y la recordada (derecha) de las dos redes neuronales. Ambas redes devolvieron correctamente las imágenes enseñadas.

La segunda consigna pide evaluar la propiedad de la red de recordar contenido. La idea es iniciar con una imagen alterada, permitir que la red evolucione y verificar a que estado convergió. Idealmente la red va a poder devolver el estado deseado, que sería el inicial sin las alteraciones. Las figuras 3 y 4 presentan visualmente los resultados de partir de una imagen alterada. Ambas redes neuronales fueron capaces de converger a los estados memorizados correctos, habiendo partido de los estados alterados. Esto se logró en 2 iteraciones, ya que se requieren dos para verificar la convergencia de la red (ausencia de cambios entre iteraciones).

El tercer ítem de las consignas exige evaluar la existencia de los estados espurios previamente mencionados. Para esto, se eligieron algunos estados espurios representativos que permiten demostrar su presencia. El único tipo de estado que no se pudo evidenciar es el tercer tipo, los verdaderamente espurios. Recordando y profundizando, los estados espurios serían:

- **Estados inversos de los memorizados:** si un patrón memorizado es $\mathbf{x}$, entonces $-\mathbf{x}$ también es un atractor de la red, ya que posee la misma energía.
- **Combinaciones lineales de un número impar de patrones:** se generan estados de mezcla a partir de un conjunto de patrones memorizados $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}\}$ con k impar, mediante

  $$\mathbf{s} = \text{sign}\left(\sum_{i=1}^{k} \mathbf{x}^{(i)}\right),$$

  donde la función signo se aplica componente a componente. Estos estados son atractores espurios porque no corresponden a ningún patrón individual entrenado, pero emergen de la dinámica de la red.
- **Estados verdaderamente espurios:** atractores que no están relacionados directamente con los patrones memorizados; su aparición es más rara y depende de la red y la configuración de los patrones.

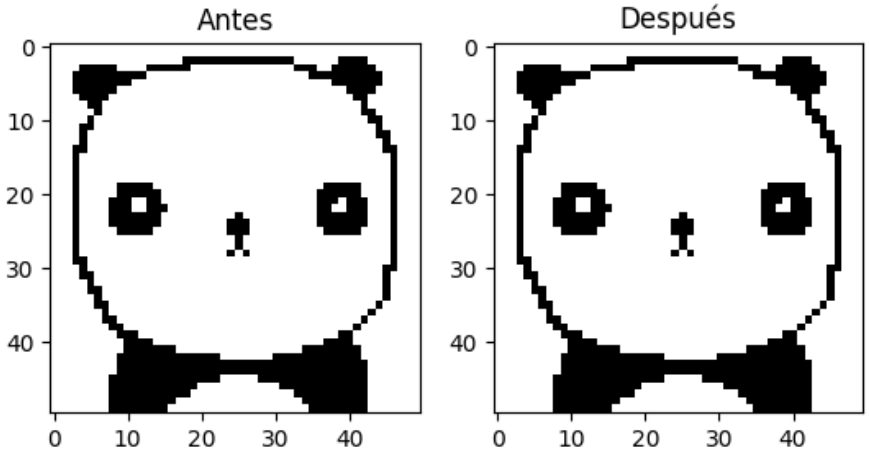
La figura 5 muestra, a la izquierda, el estado entregado a la red (el inverso del panda) y, a la derecha, el estado al que convergió en una iteración, indicando que es un punto de estabilidad en la red 1.

La figura 6 demuestra que el inverso del estado de "v" también constituye un atractor estable en la red 1.

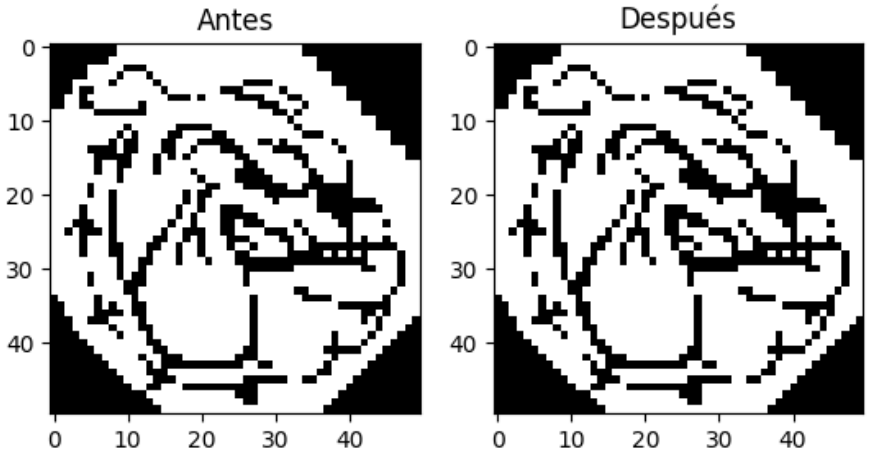
La figura 7 muestra un caso similar, partiendo del inverso del torero editado. La convergencia al estado inverso indica que el torero inverso es un atractor en la red 2.

La figura 8 presenta como entrada una combinación lineal de los tres patrones usados para entrenar la red 1. La red no cambia de estado al iterar, lo que indica que esta combinación es estable.

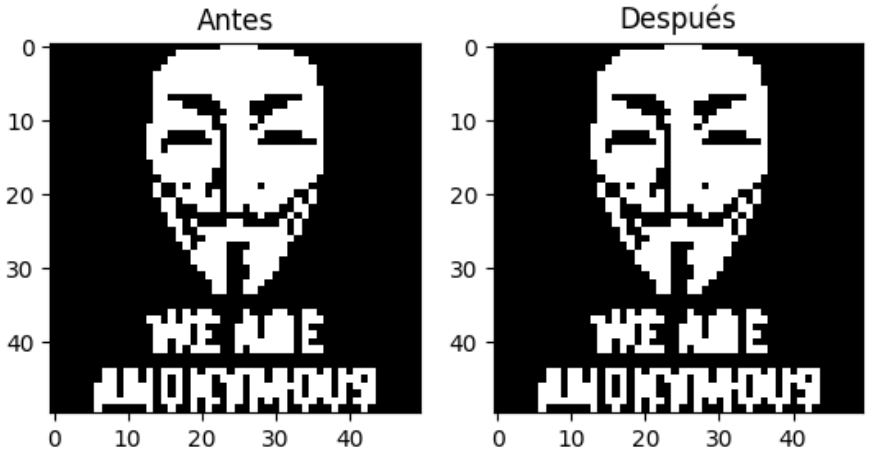
Finalmente, la figura 9 muestra como entrada una combinación lineal de los tres patrones usados para entrenar la red 2, incluyendo la paloma editada. La convergencia al estado de combinación lineal indica que esta combinación también es estable y actúa como atractor.



(a) Imagen 1

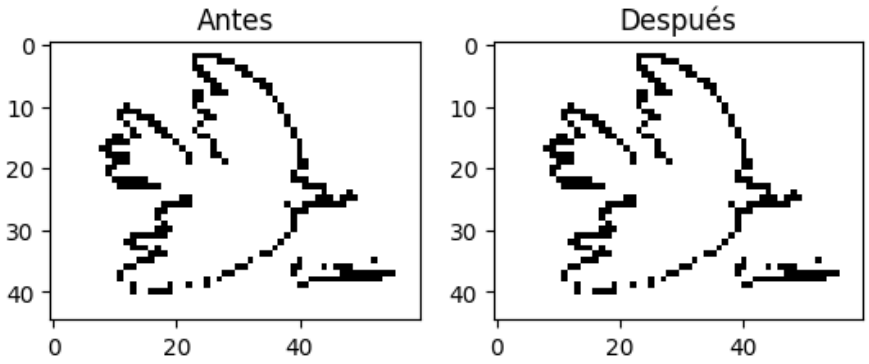


(b) Imagen 2

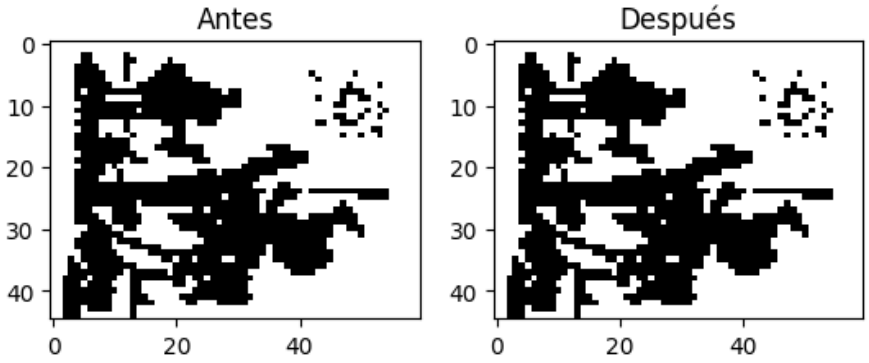


(c) Imagen 3

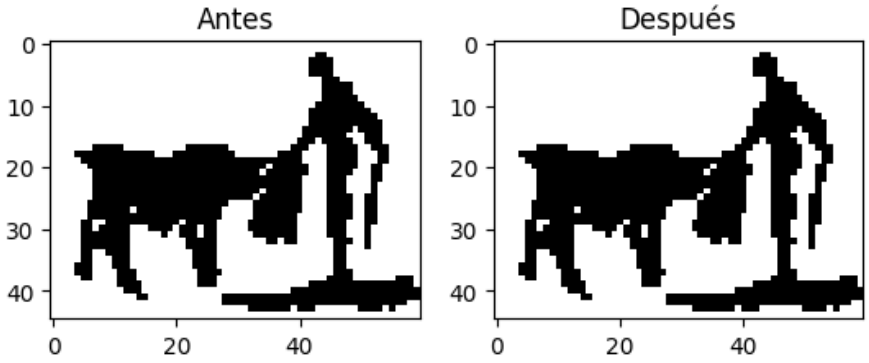
Figure 1: Imágenes del inciso 1 - Red 1.



(a) Imagen 4



(b) Imagen 5



(c) Imagen 6

Figure 2: Imágenes del inciso 1 - Red 2.

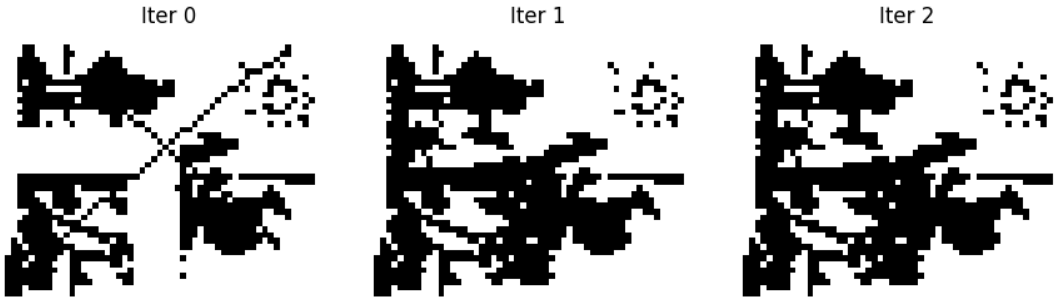
(a) Imagen 1

(b) Imagen 2

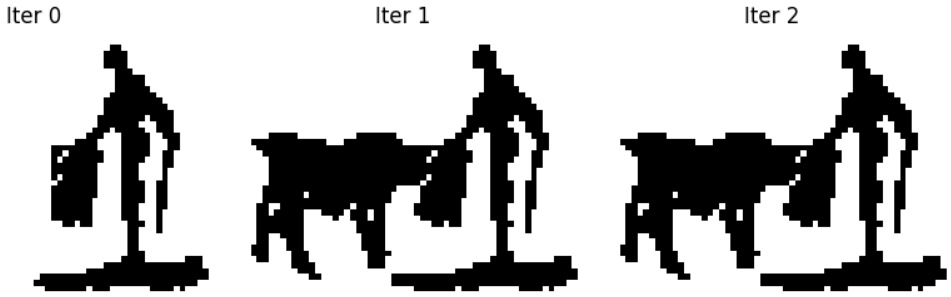
(c) Imagen 3

Figure 3: Imágenes del inciso 2 - Red 1.

(a) Imagen 4



(b) Imagen 5



(c) Imagen 6

Figure 4: Imágenes del inciso 2 - Red 2.

Figure 5: Figura 1: El estado inverso del panda es estable.

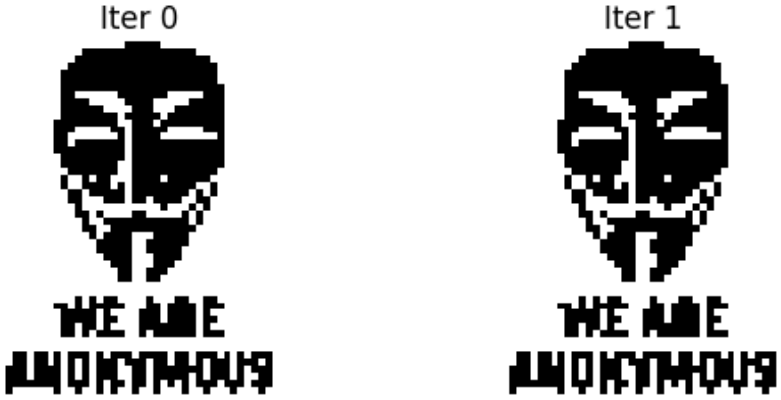


Figure 6: Figura 2: El estado inverso del torero editado es un atractor.

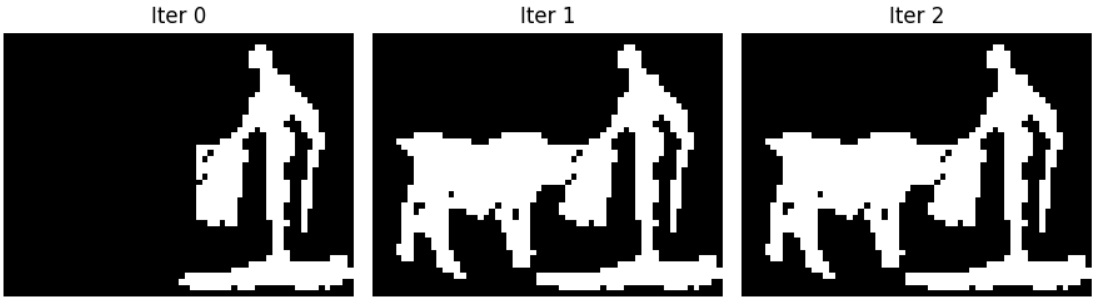


Figure 7: Figura 3: La combinación lineal de los 3 estados enseñados a la red 1 es estable.

Figure 8: Figura 4: La combinación lineal con un estado alterado es un atractor.

Figure 9: Figura 5: descripción breve de la imagen.

El cuarto y último inciso del primer ejercicio requiere entrenar a una red con todas las imágenes y verificar si fueron aprendidas. La nueva red fue entrenada sobre las imágenes redimensionadas a un tamaño en común, que inicialmente se fijó en 60×60 . El resultado se puede ver en la figura 10

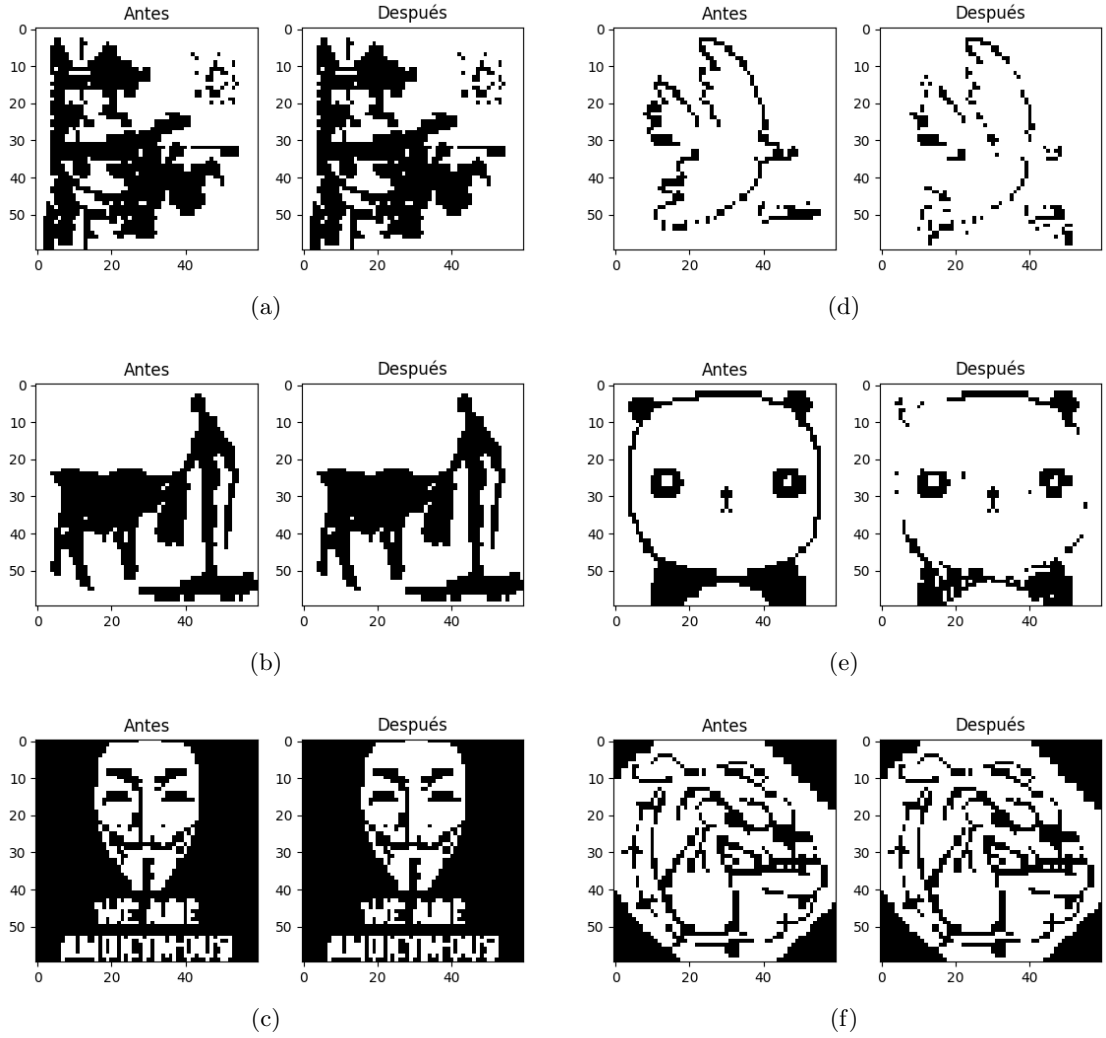


Figure 10: Imágenes enseñadas (a izquierda de cada subfigura) y recordadas (a derecha).

2.3 Análisis

3 Ejercicio 2

3.1 Introducción

3.2 Resultados

3.3 Análisis

4 Conclusiones

Leí tus notebooks y te dejo una devolución organizada: Lo que está bien en tu análisis

* **Implementación clara**: programaste la regla de Hebb y la dinámica de actualización de Hopfield de manera correcta y modular. * **Experimentos variados**: probaste con diferentes números de patrones y neuronas, lo que te permitió mostrar la relación entre capacidad y error. * **Visualización**: los gráficos permiten ver cómo evoluciona la red y en qué punto empieza a fallar la memoria. * **Discusión inicial**: mencionás la capacidad límite (aprox. ‘0.14N‘) y observás cómo se degrada el rendimiento al aumentar la carga.

Aspectos en los que podrías ahondar

1. **Profundizar en la teoría**

* Explicar mejor por qué la capacidad máxima se aproxima a ‘0.14N‘ (derivación a partir de resultados de Amit, Gutfreund y Sompolinsky). * Diferenciar entre *memorizar patrones* y *recuperarlos con ruido* (estabilidad de atractores vs. basins of attraction).

2. **Dinámica de actualización**

* Comparar **actualización síncrona vs. asíncrona** y sus consecuencias en la convergencia. * Mostrar ejemplos donde la red entra en **ciclos** o estados espurios.

3. **Ruido y robustez**

* Evaluar qué pasa si los patrones iniciales tienen cierto porcentaje de bits cambiados. * Graficar probabilidad de recuperación exitosa vs. nivel de ruido inicial.

4. **Estados espurios**

* Mencionar y, si podés, mostrar ejemplos de **estados espurios mixtos** (combinaciones lineales de patrones almacenados). * Discutir qué implican para la capacidad real de la red.

5. **Extensiones posibles**

* Comentar variantes como Hopfield continuo (con funciones sigmoides), o usar matrices de pesos con aprendizaje estocástico. * Mencionar relación con máquinas de Boltzmann y redes modernas de memoria asociativa.

—

Para tu **documento en LaTeX** te convendría estructurarlo así:

1. Introducción breve (qué es una red de Hopfield y para qué sirve). 2. Regla de aprendizaje (con ecuación). 3. Dinámica y convergencia. 4. Experimentos y resultados (capacidad, ruido, errores). 5. Limitaciones y próximos pasos (espurios, generalización).

¿Querés que te arme un **esqueleto en LaTeX** con estas secciones, listo para que pegues tus resultados?