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.

2.3 Resultados

A continuación se presentan los resultados

2.4 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?