Trabajo práctico 5: Memorias asociativas

IGNACIO LEMBO FERRARI[1]

[1] ignaciolembo@ib.edu.ar

17 de noviembre del 2023.

1. MODELO DE HOPFIELD SIN RUIDO

Se utilizó una red de Hopfield sin ruido para resolver el problema de memorizar patrones. Para esto se crearon p patrones con N neuronas cada uno, x_i^μ $(i = 1, ...N, \mu = 1, ..., p)$, donde cada neurona i puede tomar valores ± 1 con igual probabilidad. A partir de estos patrones se calculó la matriz de conexiones mediante la regla de aprendizaje de Hebb

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} x_i^\mu x_j^\mu \quad \forall \quad i \neq j, \qquad w_{ii} = 0. \tag{1}$$

Sea s un estado con N neuronas. En este modelo con dinámica determinista, se determina la evolución de la neurona i-ésima del estado s con la siguiente regla

$$s_i(t+1) = \mathrm{sgn}\left(\sum_{j=1}^{N} w_{ij} s_j(t)\right). \tag{2}$$

En este punto existen dos maneras de actualizar el vector de neuronas, de forma secuencial o paralela. En el primer caso, se actualiza neurona por neurona de manera que la neurona i-ésima (con $i > 0$) se actualiza con información de neuronas ya actualizadas. En el segundo caso, se calculan todos los $h_i(t)$ y luego se actualizan todas las neuronas al mismo tiempo y solo dependen de la información de neuronas en la iteración anterior. Luego, se deja evolucionar al sistema, tanto con dinámica secuencial o paralela hasta que el estado s converja, es decir, $s(t+1) = s(t)$. En el caso de no darse la condición de convergencia se repite todo el proceso de actualización de las N neuronas para todo el estado s. Se tomó cada patrón μ como condición inicial, es decir, $x_i^\mu = s(0)$.

En el caso de converger, se puede calcular el overlap m^μ para cada patrón μ como

$$m^\mu = \frac{1}{N}\sum_{i=1}^{N} x_i^\mu s_i^\mu. \tag{3}$$

Veamos que las neuronas solo pueden tomar valores ± 1, el overlap será 1 en el caso que todas las neuronas en $s(t)$ coincidan con el patrón x^μ. En este caso, la red reconoce perfectamente el patrón. Valores de overlap menores a 1 implican que la red tiene problemas para reconocer el patrón.

Este estudio se realizó para $N = 500, 1000, 2000, 4000$ neuronas y $\alpha = 0.12, 0.14, 0.16, 0.18$ donde $p = N\alpha$ es el número de patrones. Esto se realizó tanto para la dinámica secuencial como paralela. En el caso de la dinámica paralela no se obtuvo convergencía de los estados s. Por otra parte, en la dinámica secuencial, sí se obtuvo convergencia. En la Fig. 1 se muestran los histogramas para todos los estudios realizados. Se observa que para $\alpha = 0.12, 0.14$ la distribución del overlap es unimodal entorno a 1 para todo N. Al aumentar α el reconocimento de la red empeora y se observa que el overlap se distribuye en forma bimodal torno a $m = 1$ y $m \approx 0.3$. Para estos últimos casos de α grande se obtiene una distribución bimodal donde al aumentar N se observan cada vez más valores entorno a $m \approx 0.3$. En el caso particular, de N y α máximos se observa una distribución unimodal en torno a $m \approx 0.3$.

2. MODELO DE HOPFIELD CON RUIDO

Se simuló la dinámica de Hopfield con ruido. En este caso, a diferencia de la sección anterior, la ley que actualiza las neuronas $s_i(t)$ a tiempo t está dada de forma estocástica por

$$P_r(s_i(t+1) = \pm 1) = \frac{\exp(\pm\beta h_i(t))}{\exp(\beta h_i(t)) + \exp(-\beta h_i(t))}. \tag{4}$$

donde $\beta = 1/T$ y $h_i(t) = \sum_{j=1}^{N} w_{ij} s_j(t)$.

Al igual que antes se tomó cada patrón como condición inicial, $x_i^\mu = s(0)$. Se dejó evolucionar al sistema recorriendo las N neuronas de forma secuencial hasta recorrer un total de 10 veces cada neurona.

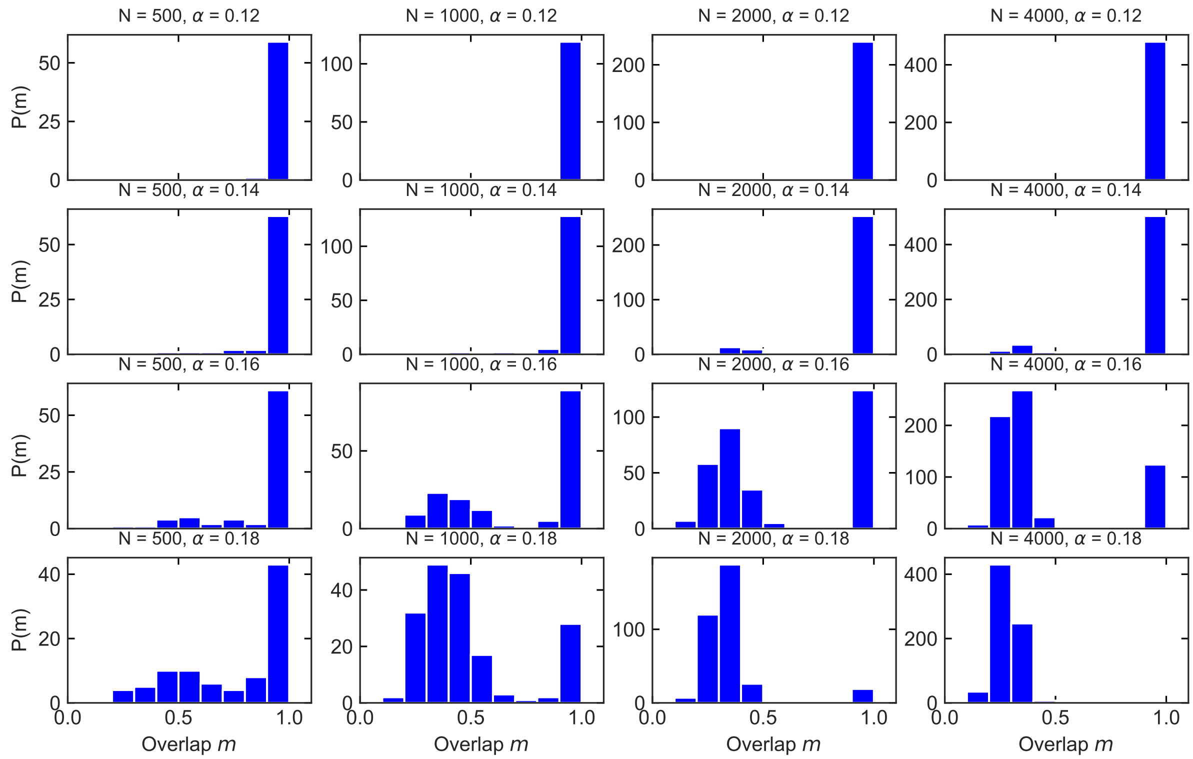

Fig. 1. Distribución del overlap m para una red de Hopfield sin ruido con dinámica secuencial entrenada con p patrones. Se muestran estudios para $N = 500, 1000, 2000, 4000$ y $\alpha = 0.12, 0.14, 0.16, 0.18$ donde N es el número de neuronas por patrón y $p = N\alpha$ el número de patrones.

En la Fig. 2 se muestra el overlap promedio $\overline{m}$ en función de la temperatura T para una red de Hopfield con ruido con dinámica secuencial entrenada con $p = 40$ patrones y $N = 4000$ neuronas cada uno. Se grafican además la franja de incerteza dada por el desvío estándar de los valores de m. En primer lugar, se observa una temperatura crítica cerca de $T = 1.0$ donde el overlap pasa de 1 a 0. lo cual es esperado para $p \ll N$. Se observa que para valores bajos de T el error cuadrático medio es bajo y se agranda a medida que aumenta T, luego de la transición este error se mantiene aproximadamente constante. A T pequeño entonces vemos que la red reconoce bien los patrones ya que el overlap es 1 y luego de la transición a $T = 1.0$ la red deja de reconocer los patrones y el overlap se va a 0.

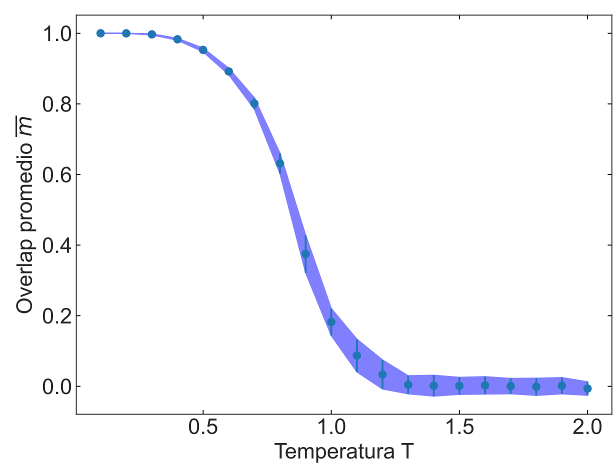

Fig. 2. Overlap promedio $\overline{m}$ en función de la temperatura T para una red de Hopfield con ruido con dinámica secuencial entrenada con p patrones. Se utilizaron $p = 40$ patrones con $N = 4000$ neuronas cada uno. Se grafican además la franja de incerteza dada por el desvío estandar de los valores de m.

A. APÉNDICE

A. Ejercicio 1

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import random

#Ploteo
import seaborn as sns
#sns.axes_style("whitegrid")
sns.set_style("ticks")

def genera_patrones(p, N):
    return np.array([[random.choice([-1, 1]) for _ in range(N)] for _ in range(p)])

Ns = [500,1000,2000,4000]
alphas = [0.12,0.14, 0.16, 0.18]

fig, axs = plt.subplots(nrows=len(alphas), ncols=len(Ns), figsize=(10, 6))

j = 0
for N in Ns:
    idx = np.arange(N)
    l = 0
    for alpha in alphas:

        #Generacion de patrones
        x = genera_patrones(int(N*alpha), N)
        #Matriz de conexiones
        w = np.zeros((N,N))
        #Vector de overlaps
        m = np.zeros(int(alpha*N))

        for u in range(int(alpha*N)):
            w += (1/N)*np.dot(x[u].reshape(-1,1), x[u].reshape(1,-1))
        np.fill_diagonal(w, 0)

        #Secuencial
        print("Secuencial - N=", N, " - alpha=", alpha)
        for u in tqdm(range(int(alpha*N))):
            s = x[u].copy()
            f = True
            while f:
                r = 0
                #np.random.shuffle(idx)
                f = False
                for i in idx:
                    h = np.sign(np.dot(w[i], s))
                    if(s[i]*h < 0):
                        f = True
                    s[i] = h
                #if(r==1000):
                #    f = False
                #    print("Corto por limite de iteraciones")
                #r += 1

            #overlap
            m[u] = (1/N)*np.dot(x[u],s)

        bin_width = 0.1  # Ancho constante de los bins
        bin_edges = np.arange(0, 1 + bin_width, bin_width)
        axs[l,j].hist(m, bins=bin_edges, alpha=1, color='b')
        axs[l,j].set_title(f"N = {N}, $\\alpha$ = {alpha}", fontsize = 9)
        axs[l,j].tick_params(direction='in', top=True, right=True, left=True, bottom=True)
        axs[l,j].tick_params(axis='x', rotation=0, labelsize=10, color='black')
        axs[l,j].tick_params(axis='y', labelsize=10, color='black')
        axs[l,j].set_xlim(0, 1.1)
        if(l != len(Ns)-1 ):
            axs[l,j].axes.xaxis.set_ticklabels([])
        #if(j != 0):
```

```
            #axs[l,j].axes.yaxis.set_ticklabels([])

        """
        #Paralelo
        m = np.zeros(int(alpha*N))
        print("Paralelo - N=", N, " - alpha=", alpha)
        for u in tqdm(range(int(alpha*N))):
            s_j = x[u].copy()
            s_i = s_j.copy()
            f = True
            idx = 0
            while f:
                s_i = np.sign(np.dot(w, s_j.reshape(-1,1)))
                if((s_i == s_j).all() or idx == 1000):
                    f = False
                s_j = s_i
                idx += 1
            print("Alcanzo el limite de iteraciones")
        """
        l = l + 1
    j = j + 1

for j in range(len(Ns)):
    axs[len(Ns)-1,j].set_xlabel("Overlap $m$")
for l in range(len(alphas)):
    axs[l,0].set_ylabel("P(m)")

fig.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej1/hist_sec.pdf")
fig.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej1/hist_sec.png", dpi=600)
plt.show()
```

B. Ejercicio 2

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import random

#Ploteo
import seaborn as sns
#sns.axes_style("whitegrid")
sns.set_style("ticks")

def genera_patrones(p, N):
    return np.array([[random.choice([-1, 1]) for _ in range(N)] for _ in range(p)])

def signo_T(h, T):
    p = np.exp(h/T) / (np.exp(h/T)+np.exp(-h/T))
    q = 1 - p
    s = np.random.choice([1, -1], p=[p, q])
    return s

N = 4000
p = 40
Ts = np.arange(0.1, 2.1, 0.1)
its = 10

#Vector de overlaps
m_p = np.zeros(len(Ts))
m_std = np.zeros(len(Ts))
fig1, ax1 = plt.subplots(figsize=(8,6))

#Generacion de patrones
x = genera_patrones(p, N)
#Matriz de conexiones
w = np.zeros((N,N))
for u in range(p):
    w += (1/N)*np.dot(x[u].reshape(-1,1), x[u].reshape(1,-1))
np.fill_diagonal(w, 0)

t = 0
```

```
    idx = np.arange(N)
    for T in tqdm(Ts):
        m = np.zeros(p)
        for u in range(p):
            s = x[u].copy()
            for it in range(its):
                ##np.random.shuffle(idx)
                for i in idx:
                    h = np.dot(w[i,:], s)
                    s[i] = signo_T(h, T)
            #overlap
            m[u] = (1/N)*np.dot(s,x[u])
        m_p[t] = np.mean(m)
        m_std[t] = np.std(m)
        t += 1

    ax1.errorbar(Ts, m_p, yerr=m_std, fmt='o')
    ax1.fill_between(Ts, m_p+m_std, m_p-m_std, alpha=0.5, color="blue")
    ax1.set_xlabel(r"Temperatura T", fontsize=18)
    ax1.set_ylabel(r"Overlap promedio $\overline{m}$", fontsize=18)
    ax1.tick_params(direction='in', top=True, right=True, left=True, bottom=True)
    ax1.tick_params(axis='x', rotation=0, labelsize=18, color='black')
    ax1.tick_params(axis='y', labelsize=18, color='black')

    fig1.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej2/m_vs_T.pdf")
    fig1.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej2/m_vs_T.png", dpi=600)

    plt.show()
```