

Trabajo práctico 5: Memorias asociativas

IGNACIO LEMBO FERRARI¹

¹ignaciolembo@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 ($i = 1, \dots, N, \mu = 1, \dots, p$), 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

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

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

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

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

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

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 convergencia de los estados s . Por otra parte, en la dinámica secuencial, sí se obtuvo convergencia. En la Fig. ?? 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 reconocimiento 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

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

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.

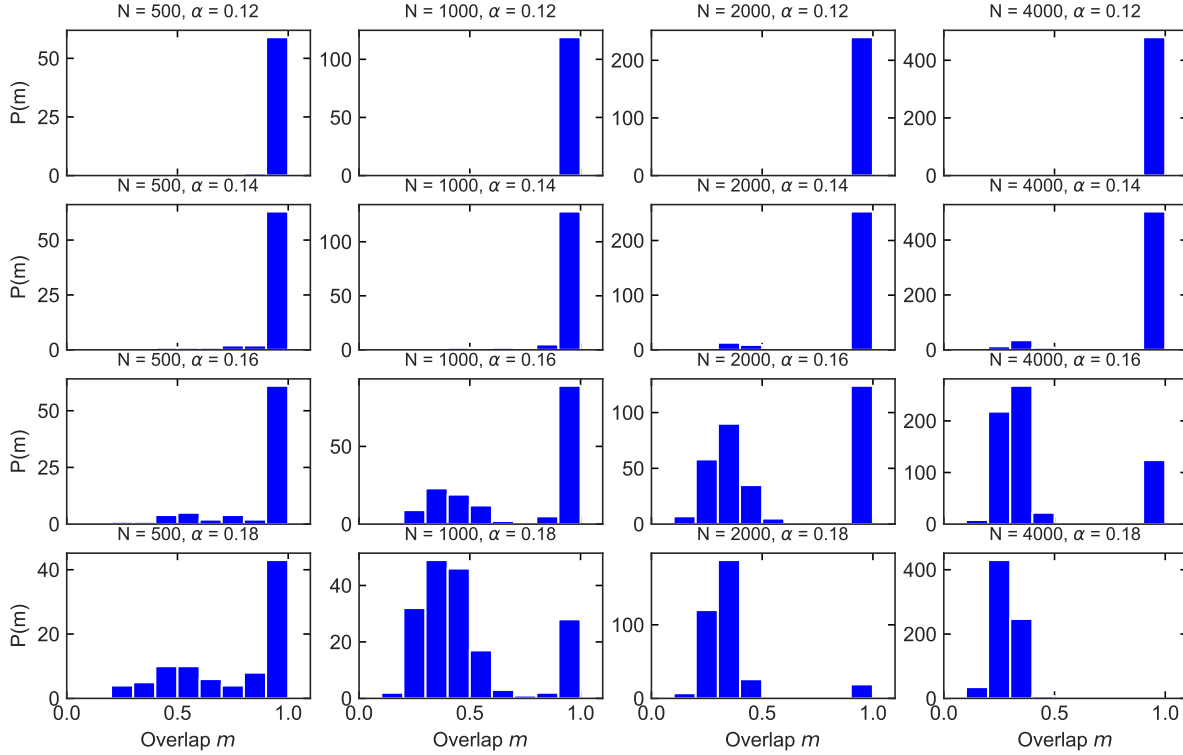


Fig. 1. Distribución del overlap m para una red de Hopfield sin ruido con dinámica secuencial entrenada con p patrones. Se muestran estudios para $N = 500, 1000, 2000, 4000$ y $\alpha = 0.12, 0.14, 0.16, 0.18$ donde N es el número de neuronas por patrón y $p = N\alpha$ el número de patrones.

En la Fig. ?? se muestra el overlap promedio \bar{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.

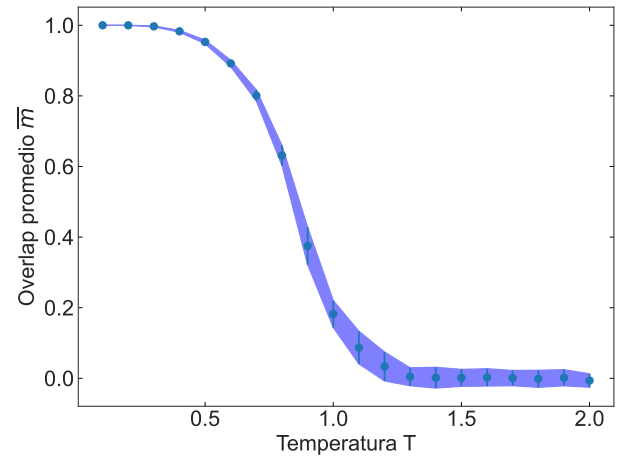


Fig. 2. Overlap promedio \bar{m} en función de la temperatura T para una red de Hopfield con ruido con dinámica secuencial entrenada con p patrones. Se utilizaron $p = 40$ patrones con $N = 4000$ neuronas cada uno. Se grafican además la franja de incerteza dada por el desvío estándar de los valores de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

A. APÉNDICE

A. Ejercicio 1

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from tqdm import tqdm
4  import random
5
6  #Ploteo
7  import seaborn as sns
8  #sns.axes_style("whitegrid")
9  sns.set_style("ticks")
10
11 def genera_patrones(p, N):
12     return np.array([[random.choice([-1, 1]) for _ in range(N)] for _ in range(p)])
13
14 Ns = [500,1000,2000,4000]
15 alphas = [0.12,0.14, 0.16, 0.18]
16
17 fig, axs = plt.subplots(nrows=len(alphas), ncols=len(Ns), figsize=(10, 6))
18
19 j = 0
20 for N in Ns:
21     idx = np.arange(N)
22     l = 0
23     for alpha in alphas:
24
25         #Generacion de patrones
26         x = genera_patrones(int(N*alpha), N)
27         #Matriz de conexiones
28         w = np.zeros((N,N))
29         #Vector de overlaps
30         m = np.zeros(int(alpha*N))
31
32         for u in range(int(alpha*N)):
33             w += (1/N)*np.dot(x[u].reshape(-1,1), x[u].reshape(1,-1))
34             np.fill_diagonal(w, 0)
35
36         #Secuencial
37         print("Secuencial - N=", N, " - alpha=", alpha)
38         for u in tqdm(range(int(alpha*N))):
39             s = x[u].copy()
40             f = True
41             while f:
42                 r = 0
43                 #np.random.shuffle(idx)
44                 f = False
45                 for i in idx:
46                     h = np.sign(np.dot(w[i], s))
47                     if(s[i]*h < 0):
48                         f = True
49                         s[i] = h
50                 #if(r==1000):
51                 #     f = False
52                 #     print("Corto por limite de iteraciones")
53                 #r += 1
54
55             #overlap
56             m[u] = (1/N)*np.dot(x[u],s)
57
58         bin_width = 0.1 # Ancho constante de los bins
59         bin_edges = np.arange(0, 1 + bin_width, bin_width)
60         axs[l,j].hist(m, bins=bin_edges, alpha=1, color='b')
61         axs[l,j].set_title(f"N = {N}, $\alpha$ = {alpha}", fontsize = 9)
62         axs[l,j].tick_params(direction='in', top=True, right=True, left=True, bottom=True)
63         axs[l,j].tick_params(axis='x', rotation=0, labels=10, color='black')
64         axs[l,j].tick_params(axis='y', labels=10, color='black')
65         axs[l,j].set_xlim(0, 1.1)
66         if(l != len(Ns)-1 ):
67             axs[l,j].axes.xaxis.set_ticklabels([])
68         #if(j != 0):

```

```

69         #axs[l,j].axes.yaxis.set_ticklabels([])
70
71     """
72     #Paralelo
73     m = np.zeros(int(alpha*N))
74     print("Paralelo - N=", N, " - alpha=", alpha)
75     for u in tqdm(range(int(alpha*N))):
76         s_j = x[u].copy()
77         s_i = s_j.copy()
78         f = True
79         idx = 0
80         while f:
81             s_i = np.sign(np.dot(w, s_j.reshape(-1,1)))
82             if((s_i == s_j).all() or idx == 1000):
83                 f = False
84             s_j = s_i
85             idx += 1
86         print("Alcanzo el limite de iteraciones")
87     """
88     l = l + 1
89     j = j + 1
90
91 for j in range(len(Ns)):
92     axs[len(Ns)-1,j].set_xlabel("Overlap $m$")
93 for l in range(len(alphas)):
94     axs[l,0].set_ylabel("P(m)")
95
96 fig.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej1/hist_sec.pdf")
97 fig.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej1/hist_sec.png", dpi=600)
98 plt.show()

```

B. Ejercicio 2

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from tqdm import tqdm
4  import random
5
6  #Ploteo
7  import seaborn as sns
8  #sns.axes_style("whitegrid")
9  sns.set_style("ticks")
10
11 def genera_patrones(p, N):
12     return np.array([[random.choice([-1, 1]) for _ in range(N)] for _ in range(p)])
13
14 def signo_T(h, T):
15     p = np.exp(h/T) / (np.exp(h/T)+np.exp(-h/T))
16     q = 1 - p
17     s = np.random.choice([1, -1], p=[p, q])
18     return s
19
20 N = 4000
21 p = 40
22 Ts = np.arange(0.1, 2.1, 0.1)
23 its = 10
24
25 #Vector de overlaps
26 m_p = np.zeros(len(Ts))
27 m_std = np.zeros(len(Ts))
28 fig1, ax1 = plt.subplots(figsize=(8,6))
29
30 #Generacion de patrones
31 x = genera_patrones(p, N)
32 #Matriz de conexiones
33 w = np.zeros((N,N))
34 for u in range(p):
35     w += (1/N)*np.dot(x[u].reshape(-1,1), x[u].reshape(1,-1))
36 np.fill_diagonal(w, 0)
37
38 t = 0

```

```
39     idx = np.arange(N)
40     for T in tqdm(Ts):
41         m = np.zeros(p)
42         for u in range(p):
43             s = x[u].copy()
44             for it in range(its):
45                 ##np.random.shuffle(idx)
46                 for i in idx:
47                     h = np.dot(w[i,:], s)
48                     s[i] = signo_T(h, T)
49                 #overlap
50                 m[u] = (1/N)*np.dot(s,x[u])
51         m_p[t] = np.mean(m)
52         m_std[t] = np.std(m)
53         t += 1
54
55     ax1.errorbar(Ts, m_p, yerr=m_std, fmt='o')
56     ax1.fill_between(Ts, m_p+m_std, m_p-m_std, alpha=0.5, color="blue")
57     ax1.set_xlabel(r"Temperatura T", fontsize=18)
58     ax1.set_ylabel(r"Overlap promedio $\overline{m}$", fontsize=18)
59     ax1.tick_params(direction='in', top=True, right=True, left=True, bottom=True)
60     ax1.tick_params(axis='x', rotation=0, labelsize=18, color='black')
61     ax1.tick_params(axis='y', labelsize=18, color='black')
62
63     fig1.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej2/m_vs_T.pdf")
64     fig1.savefig(f"../Redes-Neuronales/Practica_6/resultados/ej2/m_vs_T.png", dpi=600)
65
66     plt.show()
```