Práctica 6 - Memorias asociativas

Zablotsky, Amir Nicolás
Redes Neuronales 2022
Instituto Balseiro, UNCuyo

EJERCICIO 1

En este ejercicio se emplearon distintas redes de Hopfield para memorizar una serie de patrones. Este problema consiste en entrenar la red con patrones $\vec{\xi}^{\mu}$ tal que, al presentarle un patrón a la red, esta devuelva como salida el patrón almacenado más cercano al de entrada.

Se estudiaron redes de tamaño $N \in \{500, 1000, 2000, 4000\}$ y parámetros de carga $\alpha = \frac{p}{N} \in \{0{,}12, 0{,}14, 0{,}16, 0{,}18\}$. Para cada red, se generaron los patrones ξ_i^{μ} $(i = 1, 2, ..., N; \mu = 1, 2, ..., p)$, siendo cada uno de los valores ± 1 con probabilidad $\frac{1}{2}$.

Para cada red, se evaluó la matriz de conexiones dada por

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^{\mu} \xi_j^{\mu}, \text{ si } i \neq j \\ 0, \text{ si } i = j. \end{cases}$$

Luego, para cada red (con un dado N y $p = \alpha N$) se llevó a cabo la actualización iterando la dinámica determinista

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

tomando cada patrón como condición inicial, hasta converger al punto fijo $\vec{s}^{\mu}$ correspondiente a cada patrón. Una vez obtenidas las salidas, se calculó el overlap para cada patrón, según la expresión

$$m^{\mu} = \frac{1}{N} \sum_{i=1}^{N} s_i^{\mu} \xi_i^{\mu}.$$

En la Fig. 1 se observan los histogramas normalizados del overlap para las redes con distinta dimensión N y parámetro de carga α. Lo primero que notamos es que, en las cuatro redes con $\alpha = 0{,}12$, el overlap es cercano a uno, lo cual implica que la red puede reconocer de manera correcta los patrones almacenados en la matriz de conexiones. Esto está en acuerdo con el resultado teórico del límite $N \to \infty$, según el cual $m = 1$ para $\alpha \lesssim 0{,}138$, y $m = 0$ para $\alpha \gtrsim 0{,}138$.

Cuando el parámetro de carga es mayor a $\alpha_c \simeq 0{,}138$, vemos que el overlap medio disminuye, y vemos que cuanto más grande es α, más disminuye m, centrándose alrededor de $m \simeq \frac{1}{3}$. La razón de esta diferencia respecto al resultado teórico, según el cual $m = 0$ cuando $\alpha > \alpha_c$, es que estamos trabajando con redes de tamaño finito y no $N \to \infty$.

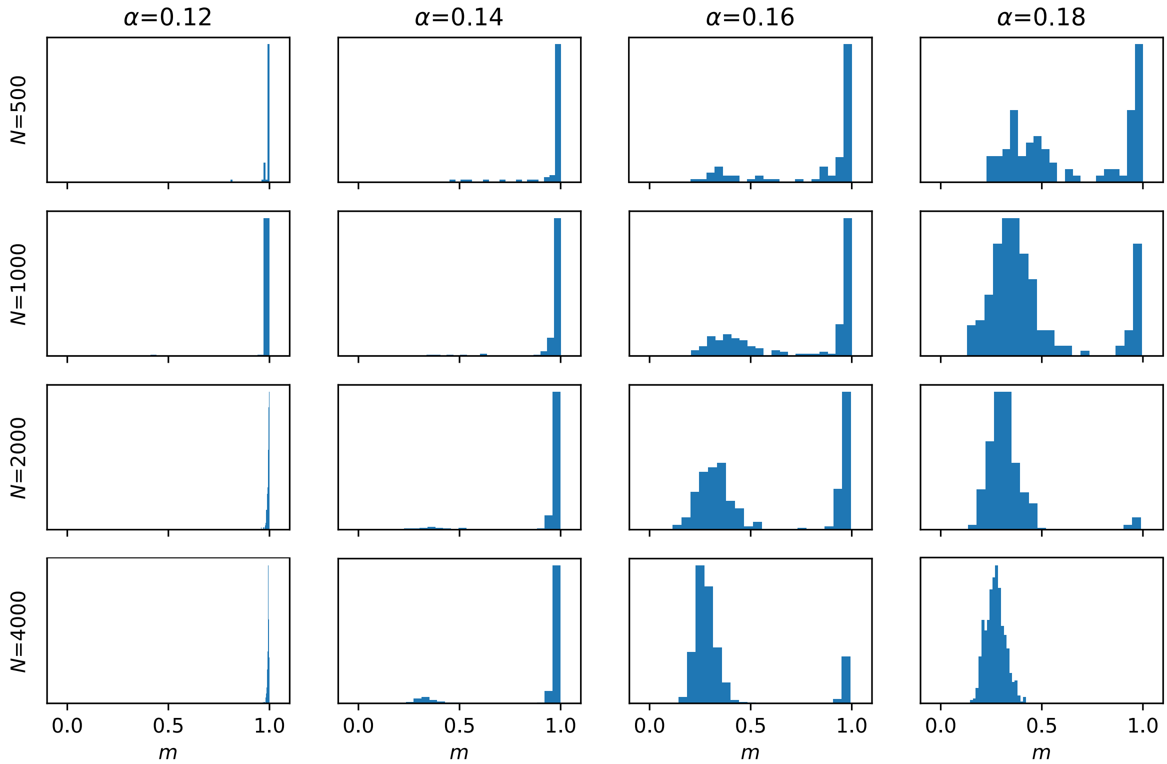

Figura 1: Histogramas normalizados del overlap, para redes de Hopfield de distinto tamaño y con distintos parámetros de carga.

EJERCICIO 2

En este ejercicio se estudió la dinámica de Hopfield con ruido usando la regla

$$Pr\left(s_i\left(t+1\right)=\pm 1\right)=\frac{\exp\left(\pm\beta h_i\left(t\right)\right)}{\exp\left(\beta h_i\left(t\right)\right)+\exp\left(-\beta h_i\left(t\right)\right)},$$

donde $h_i\left(t\right)=\sum_{j=1}^{N} w_{ij}s_j(t)$. Se estudió una red con $N=4000$ con $p=40$ patrones de entrada $\vec{\xi}^{\mu}$ (con componentes de valor ± 1 equiprobables), a partir de los cuales se computó la matriz de conexiones $\vec{w}$ de la misma forma que en el ejercicio anterior. Luego, para cada μ, se tomó el patrón $\vec{\xi}^{\mu}$ como condición inicial para la salida y se iteró 10 veces la regla estocástica descrita arriba. De esta manera, se obtiene la salida $\vec{s}^{\mu}$ asociada a cada patrón de entrada.

En la Fig. 2 se ilustra el overlap en función del parámetro de ruido $T=\frac{1}{\beta}$. Podemos ver que para valores chicos de T, el overlap es cercano a 1, lo cual indica que los patrones son puntos fijos estables de la dinámica y la red puede reconocerlos de manera correcta. Cuando aumentamos el parámetro de ruido, la media del overlap tiende a 0 y la incerteza crece. Esto se debe a que, en el límite $T\to\infty$, $Pr\left(s_i\left(t+1\right)=\pm 1\right)=\frac{1}{2}$, para todo h_i. La razón por la cual la capacidad de la red tiende suavemente a 0 es, nuevamente, por el hecho de que estamos trabajando con una red de tamaño finito.

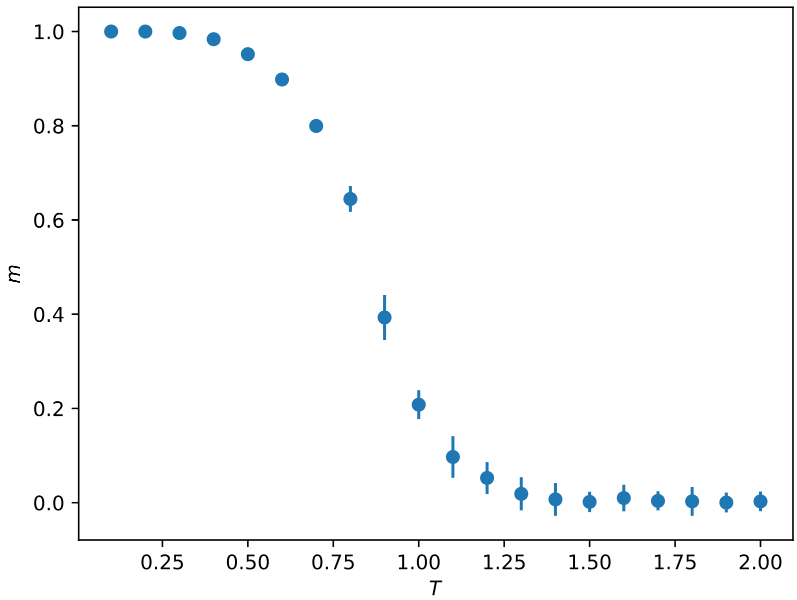

Figura 2: Overlap en función de la “temperatura” (parámetro de ruido).

APÉNDICE

Código Ejercicio 1

```python
import numpy as np
import matplotlib.pyplot as plt

def patrones_entrenamiento(N_,p_):
    return np.random.choice([-1,1],(N_,p_)).astype(float)

def get_J(patrones):
    J_= np.dot(patrones,patrones.T)/patrones.shape[0]
    np.fill_diagonal(J_,0)
    return J_

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

fig, axs = plt.subplots(nrows=4, ncols=4, figsize=(10, 6))

for N in Ns:
    for alfa in alphas:
        p = int(alfa*N)
        patrones = patrones_entrenamiento(N,p)
        J= get_J(patrones)

        m=[]
        for mu in range(p):
            xi=np.copy(patrones[:,mu])
            S=np.copy(patrones[:,mu])

            S_new=np.dot(J,S)
            S_new[S_new>0]=1
            S_new[S_new<0]=-1

            iter=1
            while (not np.array_equal(S,S_new) and iter<150):
                S=np.copy(S_new)
                S_new=np.dot(J,S)
                S_new[S_new>0]=1
                S_new[S_new<0]=-1
                iter+=1
            m.append((xi*S).mean())
        m=np.array(m)
        axs[Ns.index(N),alphas.index(alfa)].hist(m,bins=20)
        axs[Ns.index(N),alphas.index(alfa)].set_xlim(-0.1,1.1)

for ax in axs[-1,:]:
    ax.set_xlabel("$m$")
for ax in axs[:,-1]:
    ax.yaxis.tick_right()
    ax.set_ylabel("$N$="+str(Ns[axs[:,-1].tolist().index(ax)]))

for ax in axs[0,:]:
    ax.set_title("$\\alpha$="+str(alphas[axs[0,:].tolist().index(ax)]))

plt.show()
```

Código Ejercicio 2

```python
def prob1(h, T):
    return (np.exp(h/T) / (np.exp(h/T)+np.exp(-h/T)))

N=4000
p=40 #alpha=0.01
m=[] #(mean, std)
patrones = patrones_entrenamiento(N,p)
J= get_J(patrones)

for T in np.arange(0.1,2.1,0.1):
    m_=[]
    ids = np.arange(N)
    for mu in range(p):
        xi=np.copy(patrones[:,mu])
        S=np.copy(patrones[:,mu])

        for j in range(10):
            np.random.shuffle(ids)
            for i in ids:
                h_i = J[i] @ S

                if np.random.rand() <= prob1(h_i, T):
                    S[i] = 1.0
                else:
                    S[i] = -1.0
        m_.append((xi*S).mean())
    m.append((np.mean(m_),np.std(m_)))

m=np.array(m)
plt.errorbar(np.arange(0.1,2.1,0.1),m[:,0],yerr=m[:,1],fmt='o')
plt.xlabel("$T$")
plt.ylabel("$m$")

plt.show()
```