# تمرین سری سوم

بکتاش انصاری

99521082

(1

-1

$$W = \begin{bmatrix} 0.2 & 0.9 \\ 0.4 & 0.7 \\ 0.6 & 0.5 \\ 0.8 & 0.3 \end{bmatrix}$$

$x_1 = [1, 0, 0, 0]$

$x_2 = [0, 1, 1, 0]$

$x_3 = [0, 0, 0, 1]$

$x_4 = [1, 1, 0, 0]$

$w_1 = [0.2, 0.4, 0.6, 0.8]$

$w_2 = [0.9, 0.7, 0.5, 0.3]$

برای ورودی $x_4$ ابتدا باید را مشخص کنیم:

$$\|x_1 - w_1\| = \sqrt{(0.8)^2 + (0.4)^2 + (0.6)^2 + (0.8)^2} = 1.34$$

$$\|x_1 - w_2\| = \sqrt{(0.9)^2 + (0.7)^2 + (0.5)^2 + (0.3)^2} = 0.91$$

چون $w_2$ فاصله کمتری دارد، برنده است. حال $w_2$ را آپدیت کنیم:

($h$ برای همسایه‌ها صفر در نظر گرفته شده)

$$\Delta w = \alpha (x_1 - w_2) = [0.05, -0.35, -0.25, -0.15]$$

$$w_2 = w_2 + \Delta w = [0.95, 0.35, 0.25, 0.15]$$

حال برای ورودی $x_2$ با وزن‌های جدید محاسبه را انجام می‌دهیم

$$\|X_2 - w_1\| = \sqrt{(0.2)^2 + (0.6)^2 + (0.4)^2 + (0.8)^2} = 1.095$$

~~$\|X_2 - w_2\| = \sqrt{(0.1)^2 + (0.3)^2 + (0.5)^2 + (0.3)^2} = 0.66$~~

$$\|X_2 - w_2\| = \sqrt{(0.95)^2 + (0.65)^2 + (0.75)^2 + (0.15)^2} = 1.38$$

در این حالت وزن اول انتخاب می‌شود $w_1$

حال آن را آپدیت می‌کنیم:

$$\Delta w = \alpha(X_2 - w_1) \Rightarrow$$

$$\Delta w = [-0.1, 0.3, 0.2, -0.4]$$

$$w_1 = w_1 + \Delta w = [0.1, 0.7, 0.8, 0.4]$$

همین فرآیند را تکرار می‌کنیم برای تمام ورودی‌ها و دوباره تکرار می‌کنیم تا $\Delta w$ ها به مقدار بسیار کمی یا صفر برسند و وزن‌ها دیگر تغییر نکنند

(2

۲- برای محاسبهٔ وزن‌ها داریم:

$$w_{ij} = \sum_{k=1}^{4} x_i^k x_j^k$$

$w_{11} = 1+1+1+1 = 4=0$ $w_{12} = 1+1+1+1 = 4$
$w_{13} = 1+1-1-1 = 0$ $w_{14} = 1+1-1-1 = 0$
$w_{21} = 4$ $w_{22} = 0$ $w_{23} = 1+1-1-1 = 0$
$w_{24} = 1+1-1-1 = 0$ $w_{31} = 0$
$w_{32} = 0$ $w_{33} = 0$ $w_{34} = 1+1+1+1 = 4$
$w_{41} = 0$ $w_{42} = 0$ $w_{43} = 4$ $w_{44} = 0$

قطر اصلی صفر و مثلث بالا و پایین قرینه داریم:

| | | | |
|---|---|---|---|
| 0 | 4 | 0 | 0 |
| 4 | 0 | 0 | 0 |
| 0 | 0 | 0 | 4 |
| 0 | 0 | 4 | 0 |

threshold 0

حال برای هر کدام از ورودی‌ها داریم:

$x_1 = [1, 1, -1, -1]$

| | ۱ | ۲ | ۳ | ۴ |
|---|---|---|---|---|
| t=0 | 1 | 1 | -1 | -1 |
| t=1 | 1 | 1 | -1 | -1 |
| t=2 | ⋮ | ⋮ | ⋮ | ⋮ |
| t | 1 | 1 | -1 | -1 |
| | 4 | 4 | -4 | -4 |

قابل ذخیره سازی است
پایدار است

$x_2 = [-1, -1, 1, 1]$

| | ۱ | ۲ | ۳ | ۴ |
|---|---|---|---|---|
| t=0 | -1 | -1 | 1 | 1 |
| t=1 | -1 | -1 | 1 | 1 |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| t | -1 | -1 | 1 | 1 |
| | -4 | -4 | 4 | 4 |

قابل ذخیره سازی است
پایدار است

$x_3 = [-1, -1, -1, -1]$

| | ۱ | ۲ | ۳ | ۴ |
|---|---|---|---|---|
| t=0 | -1 | -1 | -1 | -1 |
| t=1 | -1 | -1 | -1 | -1 |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| t | -1 | -1 | -1 | -1 |
| | -4 | -4 | -4 | -4 |

قابل ذخیره سازی است
پایدار است

$x_4 = [1, 1, 1, 1]$

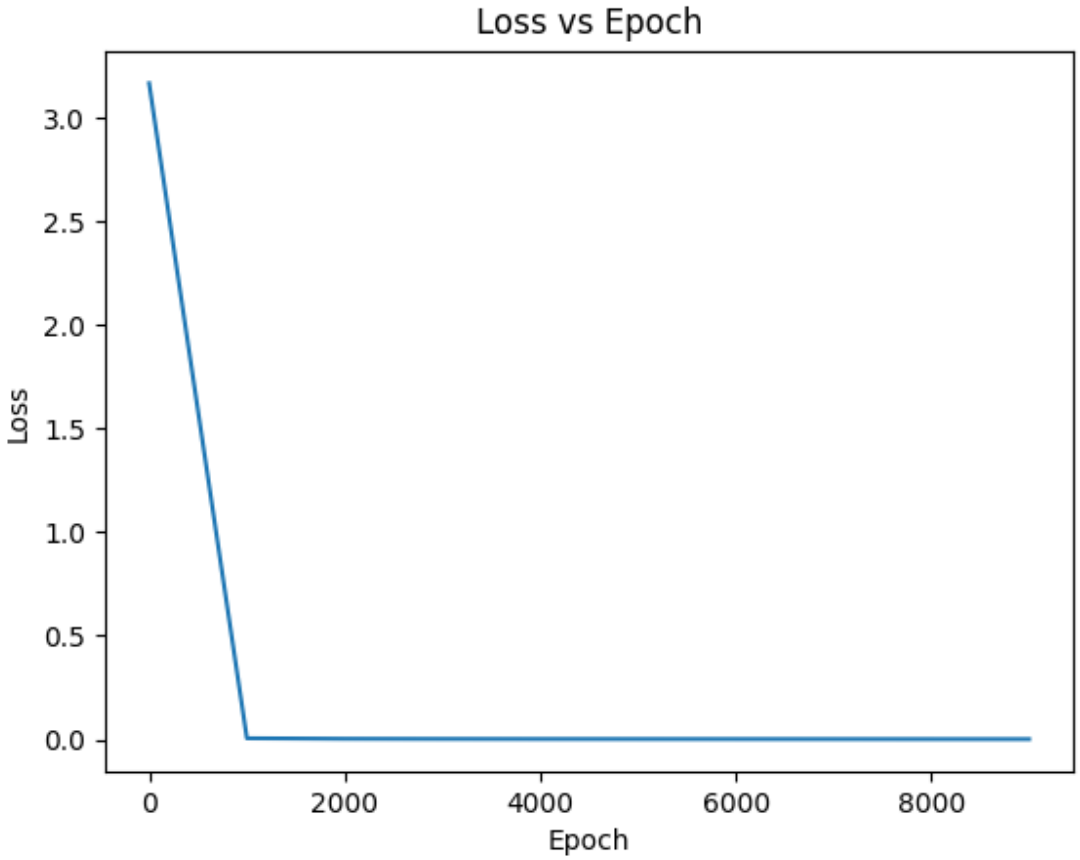
| | ۱ | ۲ | ۳ | ۴ |
|---|---|---|---|---|
| t=0 | 1 | 1 | 1 | 1 |
| t=1 | 1 | 1 | 1 | 1 |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| t= | 1 | 1 | 1 | 1 |
| | 4 | 4 | 4 | 4 |

قابل ذخیره سازی است
پایدار است

3)

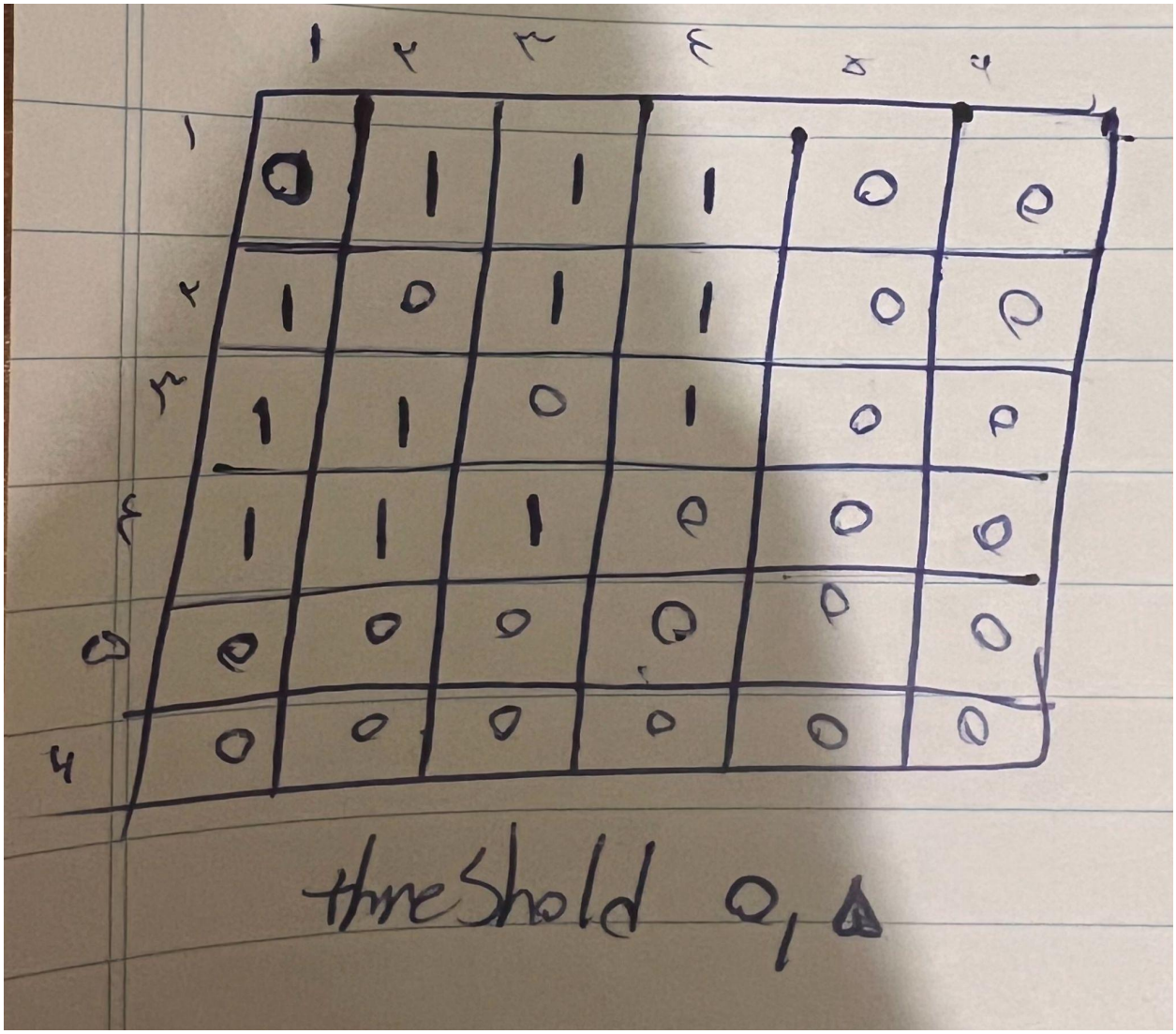
برای آنکه بتوانیم x^2 را پیاده سازی کنیم ابتدا باید داده های ورودی مدل را مشخص کنیم.
داده های ورودی مدل ما شامل x ورودی میشود و خروجی نیز باید مشخص کند که خروجی تابع چه مقدار میشود.
پس ابتدا میتوانیم یک دیتاست train طراحی کنیم که مدل روی آن آموزش ببیند.
مدل طراحی شده شامل یک نورون لایه ورودی و 16 نورون لایه میانی و 1 نورون لایه خروجی میباشد.
برای تابع activation از ReLU استفاده کردم و خروجی را نیز تغییر ندادم
برای آموزش مدل از 10000 سمپل استفاده کردم.
در نهایت مقادیری رندوم بین 3- و 3 تولید کردم و روی آنها مدل را ارزیابی کردم و مقدار loss مدل بصورت MSE را خروجی دادم.

```
Epoch 0: Loss = 3.167678127115519
Epoch 1000: Loss = 0.005281414998201611
Epoch 2000: Loss = 0.0027760398337641112
Epoch 3000: Loss = 0.002128004139333027
Epoch 4000: Loss = 0.0017821539609074286
Epoch 5000: Loss = 0.0015859556516567698
Epoch 6000: Loss = 0.001442374167119069
Epoch 7000: Loss = 0.0013318473774203913
Epoch 8000: Loss = 0.0012164093628266753
Epoch 9000: Loss = 0.0011154379908878175
Loss on Test Set = 0.0035097018233714253
```



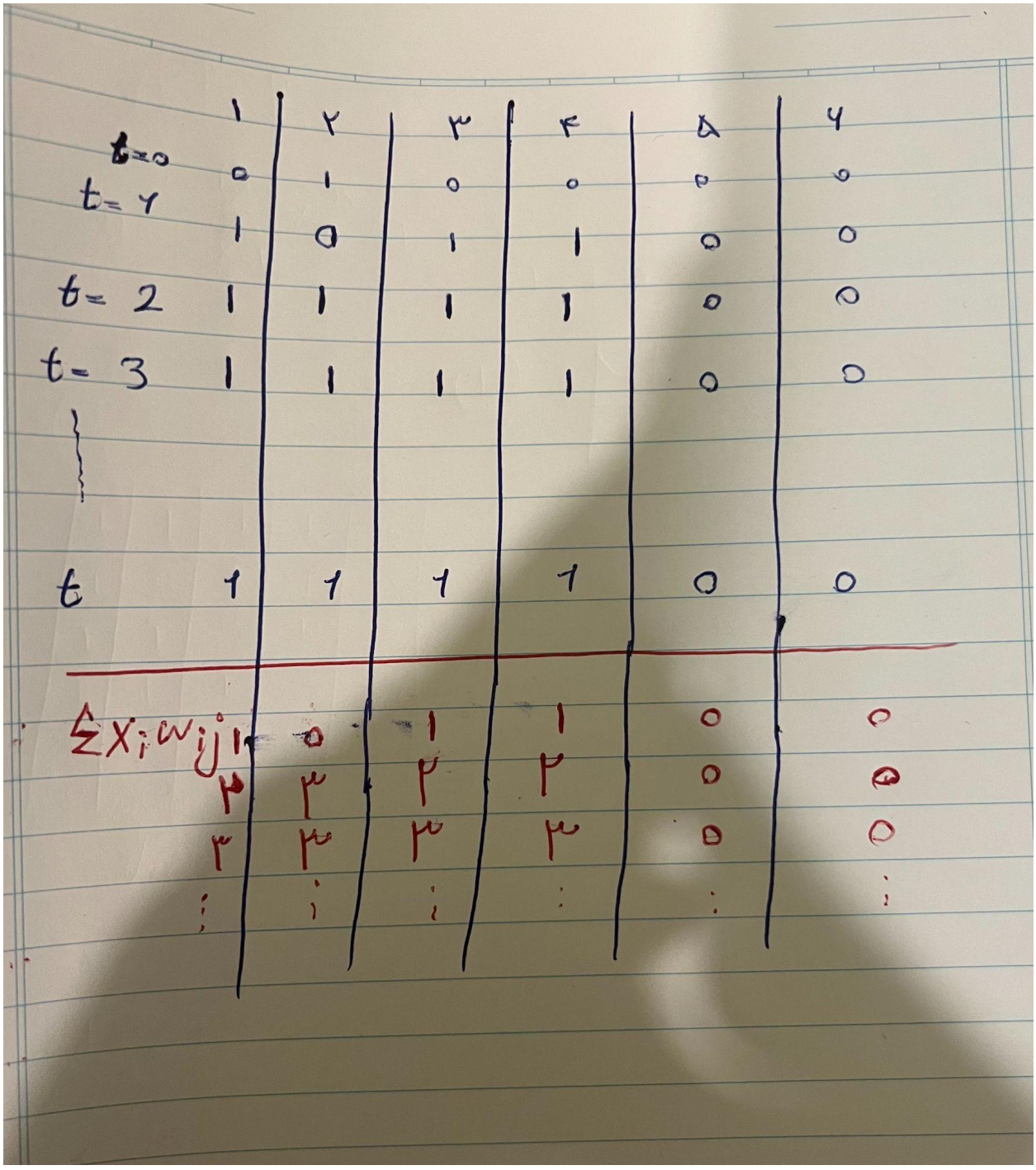
4)

برای این سوال ابتدا باید pattern = 111100 را به مدل آموزش دهیم به این شکل که یک جدول 6 در 6 برای وزن های مدل ایجاد میکنیم و با استفاده از hebbian rule وزن هارا باتوجه به تنها پترن ایجاد میکنیم که وزن های ما عبارت اند از:




در این جدول مقدار threshold را نیز 0.5 در نظر گرفتم

سپس با استفاده از این جدول و مقدار ورودی 010000 شروع به پر کردن جدول میکنیم




در نهایت مشاهده میکنیم که مقدار خروجی مدل به مقدار 111100 همگرا میشود.

(5

این مسئله با شبکه عصبی Hopfield قابل حل است. و این کار توسط Hopfield انجام شده است و این شبکه اولین شبکه عصبی ای میباشد که از آن برای حل این سوال استفاده شده است.

در این سوال ما باید به دنبال تابع انرژی ای بگردیم که در طول زمان مقدار انرژی آن کاهش پیدا کند و مارا به یک local optimum برساند. که هاپفیلد این کار را انجام داده است.

و تابع انرژی مورد نظر به این شکل خواهد بود:

$$E = \frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} X_{xj}X_{xj} + \frac{B}{2}\sum_{i=1}^{N}\sum_{x=1}^{N}\sum_{y=x}^{N} X_{xi}X_{yi} + \frac{C}{2}\left(\sum_{x=1}^{N}\sum_{i=1}^{N} X_{xi} - N\right)^2 + \frac{D}{2}\sum_{x=1}^{N}\sum_{y=1}^{N}\sum_{i=1}^{N} d_{xy}X_{xi}\left(X_{y,i+1} + X_{y,i-1}\right) \quad (5)$$

$$\frac{dU_{xi}}{dt} = -\frac{\partial E}{\partial X_{xi}} = -A\left(\sum_{i=1}^{N} X_{xi} - 1\right) - B\left(\sum_{y=1}^{N} X_{yi} - 1\right) - D\sum_{y=1}^{N} d_{xy}X_{y,i+1} \quad (6)$$

ساختار این شبکه عصبی به این شکل خواهد بود که اولین لایه ورودی شبکه است و دومین لایه نورون های شبکه. ورودی نورون ها شامل ورودی شبکه بعلاوه feedback نورون های خروجی خواهد بود.

$$z_j = \sum_{i}^{N} w_{ij}y_i + x_j$$

در اینجا wij وزن و yi = [y1, y2, ...,yN خروجی نورون ها و xi ورودی شبکه است

هاپفیلد متوجه شد که این شبکه برای این مسئله با تعداد شهر های کمتر از 30 بسیار خوب عمل میکند.

ساختار شبکه گفته شده به مانند شکل زیر است:

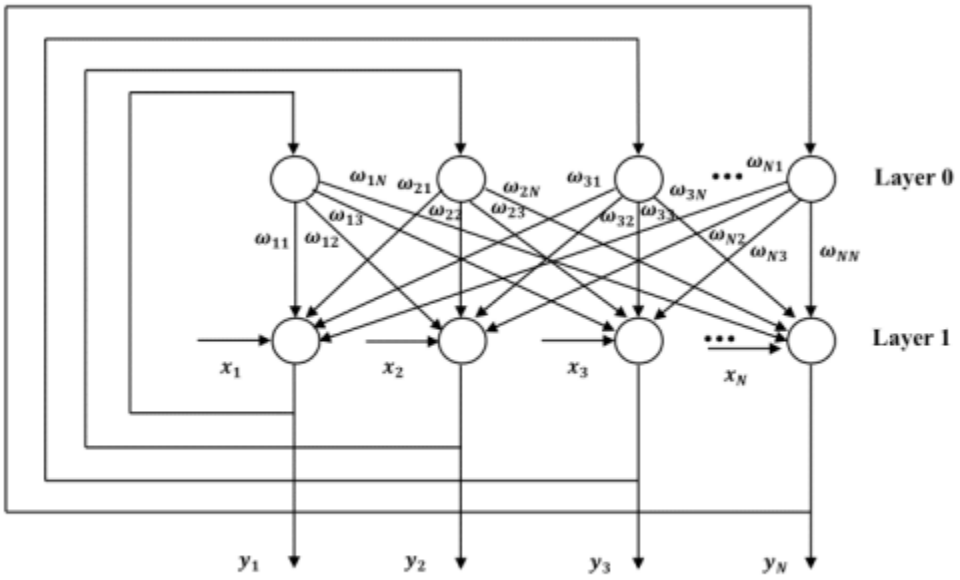



Fig. 1. Two layer of Hopfield network

منبع :

https://www.sciencedirect.com/science/article/pii/S1877050922000850/pdf?md5=61f20b69dbdddb00cf8a364c6488c131&pid=1-s2.0-S1877050922000850-main.pdf