Hessam Koucheki - 403 72 39 87

1 Ann $P_1 = [1, 1, 1, 0, 1]$ $\theta = 0$

P 1.1

update

async.

$0+1-1+0-3=-3 \to 0 \to [0,1,1,0,1]$ ✓

$0+0+3+0+0=3 \to 1 \to [0,1,1,0,1]$ X

$0+3+0+0-2=1 \to 1 \to [0,1,1,0,1]$ X

$0-1+1+0+1=-1 \to 0 \to [0,1,1,0,1]$ X

$0+0-2+0+0=-2 \to 0 \to [0,1,1,0,0]$ ✓

Start over

$0+1-1+0+0=0 \to 0 \to [0,1,1,0,0]$ X

$0+0+3-1+0=2 \to 1 \to [0,1,1,0,0]$ X

$0+3+0+0+0=3 \to 1 \to [0,1,1,0,0]$ X

$0+-1+1+0+0=0 \to 0 \to [0,1,1,0,0]$ X

$0+0-2+0+0=-2 \to 0 \to [0,1,1,0,0]$ X

مرا در منزل جانان چه امن عیش چون هر دم جرس فریاد می دارد که بر بندید محمل ها

Since we have no updates in the last operation over the whole network, we stop going further; thus, the output for the input $p_1 = [1,1,1,0,1]$, after running and updating weights is: $[0, 1, 1, 0, 0]$

sync. $\quad p_1 = [1,1,1,0,1] \quad \theta = 0$

$0+1-1+0-3=-3 \rightarrow 0 \quad n1=0$ ✓
$1+0+3+0+0=4 \rightarrow 1 \quad n2=1$ ✗
$-1+3+0+0-2=0 \rightarrow 0 \quad n3=0$ ✓
$2+-1+1+0+1=3 \rightarrow 1 \quad n4=1$ ✓
$-3+0-2+0+0=-5 \rightarrow 0 \quad n5=0$ ✓

$\Rightarrow p_1 = [0,1,0,1,0]$

starting over:

$0+1+0+2+0=3 \rightarrow 1 \quad n1=1$ ✓
$0+0+0-1+0=-1 \rightarrow 0 \quad n2=0$ ✓
$0+3+0+1+0=4 \rightarrow 1 \quad n3=1$ ✓
$0-1+0+0+0=-1 \rightarrow 0 \quad n4=0$ ✓
$0+0+0+1+0=1 \rightarrow 1 \quad n5=0$ ✗

$\Rightarrow [1,0,1,0,0]$

we have changes, so keep going.

$[1,0,1,0,0]$

starting over:

$0+0-1+0+0=-1 \to 0 \quad n1=0$

$1+0+3+0+0=4 \to 1 \quad n2=1$

$-1+0+0+0+0=-1 \to 0 \quad n3=0$

$2+0+1+0+0=3 \to 1 \quad n4=1$

$-3+0-2+0+0=-5 \to 0 \quad n5=0$

$\Rightarrow [0,1,0,1,0]$

$\downarrow$

this is exactly the out put for second iteration, meaning that we ~~e~~ enter a loop. So, stopping the process.
The output is $[0,1,0,1,0]$ which is unreliable.

1.2 Why results differ?
That's because of the nature of the Hopfield network. which tells us to tray to update asynchronically to reach the convergence. if we don't, the network take much bigger steps and may jump over the local minima, resulting in oscillation and reach an amorphuous network, and a loop, probably :)

چو کحل بینش ما خاک آستان شماست   کجا رویم بفرما از این جناب کجا

4

۱۴۰۳/۱/۱۱ P1.3

async $[0,1,0,1,0]$

$0+1+0+2+0=3 \to 1 \to [1,1,0,1,0]$ ✓

$1+0+0-1+0=0 \to 0 \to [1,0,0,1,0]$ ✓

$-1+0+0+1+0=0 \to 0 \to [1,0,0,1,0]$ X

$2+0+0+0+0=2 \to 1 \to [1,0,0,1,0]$ X

$-3+0+0+1+0=-2 \to 0 \to [1,0,0,1,0]$ X

have two changes(✓) let's start over:

$0+0+0+2+0=2 \to 1 \to [1,0,0,1,0]$ X

$1+0+0+-1+0=0 \to 0 \to [1,0,0,1,0]$ X

$-1+0+0+1+0=0 \to 0 \to [1,0,0,1,0]$ X

$2+0+0+0+0=2 \to 1 \to [1,0,0,1,0]$ X

$-3+0+0+1+0=-2 \to 0 \to [1,0,0,1,0]$ X

We have no updates in the iteration, so: $[1,0,0,1,0]$

---

sync. $[0,1,0,1,0]$

$$\left.\begin{array}{ll} 0+1+0+2+0=3 & n1=1 ✓ \\ 0+0+0-1+0=-1 & n2=0 ✓ \\ 0+3+0+1+0=4 & n3=1 ✓ \\ 0-1+0+0+0=-1 & n4=0 ✓ \\ 0+0+0+1+0=1 & n5=1 ✓ \end{array}\right\} \Rightarrow [1,0,1,0,1]$$

one more iteration : $[1,0,1,0,1]$

$0+0-1+0-3=-4 \to 0 \quad n1=0$

$1+0+3+0+0=4 \to 1 \quad n2=1$

$-1+0+0+0-2=-3 \to 0 \quad n3=0$

$2+0+1+0+1=3 \to 1 \quad n4=1$

$-3+0-2+0+0=-5 \to 0 \quad n5=0$

$\Rightarrow [0,1,0,1,0]$

we, again, reach to the same pattern which was our input, meaning that we are in a loop. So, we stop here and do not proceed. As we discussed earlier, the hopfield network, should work asyncronically to avoid this problem.

end of problem 1

6

۱۴۰۳/۱/۱۴

Problem 2

In this question, we have four potential pattern, each with four elements to store on the network. This means that our network has only 4 neurons (because of 4 elements). With this number of neurons, we can approximately store one or two patterns since:

$$P_{max} \approx 0.14 \times N \Rightarrow 0.14 \times 4 \approx \underline{\underline{0.6}}$$

(The number is less than one pattern, btw)

But the problem here is that we have 4 patterns to store and they are symmetrical two by two. Having this, the network may struggle finding the minima for a pattern since patterns that are symmetrical compete with each other.

Symmetrical patterns make it easier for the network to "learn" them, but there might be problem with "retrieval" since a noisy pattern may

(7)

۱۴۰۳/۱/۱۰

converge to the other pattern which is symmetrical to the desired pattern. The benefit in storage is coming from the fact that the Hopfield network, naturally, stores some spaurious patterns that are symmetrical to those we want to store. In conclusion, although having symmetrical patterns helps us in storing the pattern, it may be difficult for the network to distinguish between patterns since they are symmetrical. Thus, we may be able to store 4 patterns using 4 neurons, but their retrieval may be fairly challenging.

weight matrix:

$$x_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad x_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad x_3 = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \quad x_4 = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$

$$W = \frac{1}{N} \sum_{1}^{P} x_i x_i^T = \frac{1}{4}(x_1 + x_2 + x_3 + x_4) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

چه نسبت است به رندی صلاح و تقوا را
سماع وعظ کجا نغمه رباب کجا

۱۴۰۳/۲/۱۷

But since we don't have self-reinforcments, we should zero all w_{ij} where $i=j$. So, the final weight matrix becomes:

$$W = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

end of problem 2

· Problem 3

As you can observe through the provided code, I implemented both methods, random center selection and k-means, to choose initial centers for the RBF model on IRIS dataset. I also run the code multiple time with various center counts from 1 to 24. In addition to that, my code calculates the time that each classification takes as well as the amount of accuracy.

۱۴۰۳/۲/۱۸

Comparison

The k-means center selection, obviously, gives a better accuracy while taking more time to run. That has several reasons. The first reason is that by using k-means, we can ensure that the centers are distributed evenly over the feature space. When random selection is used, the centers may be chosen very close to each other, so it affects the network and model's performance. Moreover, by choosing the centers randomly, we may end up with a selection from a dense region of the data; While the k-means avoids this. After several runs, the accuracy for k-means selection reaches to 100% with 10 centers, ~~while~~ and stays that high for more centers; while the random selection needs

10

۱۴۰۳/۲/۱۹

a few more centers to give 100% accuracy, and it also fluctuate in accuracy with further runs with more centers. However, k-means selection is sometimes more than 20 times slower than the random selection. You can observe the log (output) of the code to compare them.

End of Q3