به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری سوم

نام و نام خانوادگی	فاطمه سلیقه
شماره دانشجویی	۸۱۰۱۹۸۳۰۶
تاریخ ارسال گزارش	۹۹/۲/۲۳

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

سوال ۱ – Character Recognition using Hebbian Learning Rule

۱. بله این شبکه می تواند تمام ورودی ها را به خروجی مطلوب برساند

۲. با ۲۰ درصد نویز حدود ۹۸ درصد و با ۴۰ درصد نویز حدود ۵۴ درصد را می تواند به درستی تشخیص دهد . همان طور که می بینم با افزایش تعداد نویزها درصد تشخیص درست کاهش می یابد .

```
true answers in  1000  data with noise 0.1 is  100.0
true answers in  1000  data with noise 0.11 is  100.0
true answers in  1000  data with noise 0.12 is  100.0
true answers in  1000  data with noise 0.13 is  99.9
true answers in  1000  data with noise 0.14 is  100.0
true answers in  1000  data with noise 0.15 is  99.6
true answers in  1000  data with noise 0.2 is  98.7
true answers in  1000  data with noise 0.3 is  86.4
true answers in  1000  data with noise 0.4 is  54.0
true answers in  1000  data with noise 0.5 is  0.0
true answers in  1000  data with noise 0.6 is  0.0
true answers in  1000  data with noise 0.7 is  0.0
true answers in  1000  data with noise 0.8 is  0.0
true answers in  1000  data with noise 0.9 is  0.0
```

۳. با ۲۰ درصدو یا ۴۰ درصد از دست رفتن اطلاعات ۱۰۰ درصد می توان مدل درست را تداعی نمود . میزان مقاومت نسبت به از دست دادن اطلاعات بالا است و تا ۵۰ درصد از دست دادن اطلاعات می توان تشخیص های درستی داد .

```
true answers in  1000  data with loss 0.1 is  100.0
true answers in  1000  data with loss 0.2 is  100.0
true answers in  1000  data with loss 0.3 is  100.0
true answers in  1000  data with loss 0.4 is  100.0
true answers in  1000  data with loss 0.5 is  99.5
true answers in  1000  data with loss 0.6 is  98.2
true answers in  1000  data with loss 0.7 is  97.2
true answers in  1000  data with loss 0.8 is  89.3
true answers in  1000  data with loss 0.9 is  74.5
true answers in  1000  data with loss 0.93 is  65.3
true answers in  1000  data with loss 0.95 is  41.6
true answers in  1000  data with loss 0.99 is  0.0
```

۴. همان طور که در قسمت ۲ و ۳ مشاهده نمودیم میزان مقاومت شبکه دربرابر از دست دادن اطلاعات بالاتر از نویز است . با ازدست دادن اطلاعات تا ۵۰ درصد تقریبا می توان دقت ۱۰۰ درصدی در تشخیص داشت اما با ایجاد نویز در داده ها تا ۵۰ درصد ، تشخیص مدل درست به صفر درصد می رسد .

۵. برای اطمینان از صحت اطلاعات برای محاسبه درصد تشخیص درست ، هر بار ۱۰۰۰ داده را ورودی می دهیم و صحت تشخیص خروجی را بررسی می کنیم و درصد می گیریم . برای محاسبه درصد مقاومت ، برای هر درصد ۱۰۰ بار درصد تشخیص درست را محاسبه نموده و میانگین می گیریم . میانگین های به دست آمده به صورت زیر هستند :

```
mean 5% :  100.0
mean 6% :  100.0
mean 7% :  100.0
mean 8% :  99.99699999999999
mean 9% :  99.99599999999998
```

```
mean 10% :  99.98899999999996
mean 11% :  99.995
mean 12% :  99.93699999999994
mean 13% :  99.88799999999992
mean 14% :  99.88799999999992
mean 15% :  99.60000000000002
mean 16% :  99.36599999999997
mean 17% :  99.42399999999996
```

اگر می خواهیم قطعا دقت تشخیص ما ۱۰۰ درصد باشد درصد مقاوت ۷ درصد است اما با در نظر گرفتن امکان بسیار کم خطا می توان درصد مقاومت را ۱۲ درصد گرفت که میانگین دقت 99.9 درصد دارد .

سوال ۲ – Storage Capacity in an Auto-associative Net

۱. با استفاده از فرمول زیر می توان ورودی را در شبکه ذخیره نمود :

$$W = \sum_{p=1}^{P} s(p)s(p)^T - pI_n$$

ماتریس وزن به دست آمده به صورت زیر است :

```
array([[ 0.,  1.,  1., -1.],
       [ 1.,  0.,  1., -1.],
       [ 1.,  1.,  0., -1.],
       [-1., -1., -1.,  0.]])
```

۲. اگر بردار $s2 = [1,1,-1,1]$ را به نمونه ها اضافه کنیم و در شبکه ذخیره نماییم وپس از آن مجدد ماتریس وزن W را محاسبه کنیم می بینیم که شبکه می تواند هر دو ورودی را به درستی تشخیص دهد .

نکته ای که وجود دارد این است که ضرب داخلی دو بردار $s2 = [1,1,-1,1]$ و $s = [1,1,1,-1]$ برابر صفر است و این دو بردار بر هم عمود هستند .

ماتریس وزن به دست آمده به صورت زیر است :

```
array([[ 4.,  2.,  0.,  0.],
       [ 2.,  4.,  0.,  0.],
       [ 0.,  0.,  4., -2.],
       [ 0.,  0., -2.,  4.]])
```

و می توان مشاهده نمود که شبکه می تواند ورودی ها را به درستی تداعی کند :

```
xA = (np.array(S)).dot(np.array(W2))
tA = np.sign(xA)
comparison = tA == S
comparison.all()
```

```
True
```

```
xA = (np.array(S2)).dot(np.array(W2))
tA = np.sign(xA)
comparison = tA == S2
comparison.all()
```

```
True
```

۳. از آنجایی که بردار با ابعاد ۴ است $2^4 - 1 = 15$ بردار متفاوتی که ایجاد می شوند را امتحان نموده و نتیجه را بررسی می کنیم . نتیجه ای که به دست آوردم این است که بردار غیر متعامد قانونی ندارد هم می تواند ذخیره شود هم نشود . اما اگر بردار دو به دو متعامد به شبکه اضافه کنیم تا جایی که شبکه ظرفیت دارد می تواند اضافه شود .

برای مثال با اضافه نمودن بردار $s3 = [-1,1,-1,1]$ می توان دید که عمل تداعی به درستی صورت نمی گیرد :

```
xA = (np.array(S)).dot(np.array(W3))
tA = np.sign(xA)
comparison = tA == S
xA = (np.array(S3)).dot(np.array(W3))
tA = np.sign(xA)
comparison3 = tA == S3
(comparison.all() and comparison3.all())
```

```
False
```

۴. اگر بردار $s4 = [-1,-1,1,-1]$ رادر شبکه ذخیره نماییم سه بردار $s, s2, s4$ را می توان به درستی تداعی نمود . این بردار نیز بردار s عمود است . اما بر بردار s2 عمود نیست .

```
xA = (np.array(S)).dot(np.array(W4))
tA = np.sign(xA)
comparison = tA == S
xA = (np.array(S2)).dot(np.array(W4))
tA = np.sign(xA)
comparison2 = tA == S2
xA = (np.array(S4)).dot(np.array(W4))
tA = np.sign(xA)
comparison4 = tA == S4
comparison.all() and comparison2.all() and comparison4.all()
```

```
True
```

با اضافه کردن بردار $s5 = [-1,-1,-1,1]$ نیز شبکه می تواند به درستی ورودی ها را تداعی کند

```
xA = (np.array(S)).dot(np.array(W5))
tA = np.sign(xA)
comparison = tA == S
xA = (np.array(S2)).dot(np.array(W5))
tA = np.sign(xA)
comparison2 = tA == S2
xA = (np.array(S4)).dot(np.array(W5))
tA = np.sign(xA)
comparison4 = tA == S4
xA = (np.array(S5)).dot(np.array(W5))
tA = np.sign(xA)
comparison5 = tA == S5
comparison.all() and comparison2.all() and comparison4.all() and comparison5.all()
```

```
True
```

اما با تست سایر بردار ها ی متمایز دیگر از بعد ۴ شبکه نمی تواند ورودی ها را به درستی تداعی کند

بنابراین حداکثر ۴ بردار را می توان در شبکه ذخیره نمود تا قابل بازیابی باشد .

۵. در هر auto associative MNN با ماتریس modified Hebb از با n ورودی می توان n-1 بردار را ذخیره کرد . هر بردار ذخیره شده می تواند یک بردار ویژه برای ماتریس باشد که یک مقدار ویژه متناظر دارد . نکته ای که وجود دارد این است که n-1 بردار متعامد را می تواند ذخیره کند اما اگر بردار ها متعامد نباشند می توان بردار های بیشتری هم ذخیره نمود .

من توانستم ۴ بردار ذخیره نمایم . البته ۴ بردار دو به دو برهم عمود نیستند .

اما بردارهای متعامد به صورت زیر ذخیره شدند:

```
S7 = [1,1,1,-1]
S8 = [1,1,-1,1]
S9 = [1,-1,1,1]
W7 = np.outer(S7,S7) + np.outer(S8,S8) + np.outer(S9,S9)  - 3*np.identity(4)
xA = (np.array(S7)).dot(np.array(W7))
tA = np.sign(xA)
comparison7 = tA == S7
xA = (np.array(S8)).dot(np.array(W7))
tA = np.sign(xA)
comparison8 = tA == S8
xA = (np.array(S9)).dot(np.array(W7))
tA = np.sign(xA)
comparison9 = tA == S9
comparison7.all() and comparison8.all() and comparison9.all()
```

```
True
```

اما دیگر امکان ذخیره سازی بردار متعامد [-1,1,1,1] وجود نداشت . بنابراین ظرفیت حافظه برای بردار های متعامد ۳ است .

سوال ۳ – Iterative Auto-associative Net

۱. با استفاده از فرمول زیر می توان ورودی را در شبکه ذخیره نمود :

$$W = \sum_{p=1}^{P} s(p)s(p)^T - pI_n$$

ماتریس وزن به دست آمده به صورت زیر است :

```
array([[ 0.,  1.,  1., -1.],
       [ 1.,  0.,  1., -1.],
       [ 1.,  1.,  0., -1.],
       [-1., -1., -1.,  0.]])
```

۲. در حالتی که اطلاعات ۳ تا از ۴ مقدار ورودی از دست رفته باشد ۴ حالت وجود دارد که در هیچ کدام از این ۴ حالت شبکه نمی تواند مدل را به درستی بازیابی نماید .

```
def testLoss(C):
    xC = (np.array(C)).dot(np.array(W))
    tC = np.sign(xC)
    comparison = tC == C
    return comparison.all()

print(testLoss([1,0,0,0]))
print(testLoss([0,1,0,0]))
print(testLoss([0,0,1,0]))
print(testLoss([0,0,0,-1]))
```

```
False
False
False
False
```

۳. در حالتی که ۳ تا از ۴ داده ورودی نویز داشته باشند ۴ حالت به وجود می آید که در هیچ کدام از این حالت ها نمی توان مدل را به درستی بازنمایی نمود .

```
def testnoise(C):
    xC = (np.array(C)).dot(np.array(W))
    tC = np.sign(xC)
    comparison = tC == C
    return comparison.all()

print(testnoise([ 1,-1,-1, 1]))
print(testnoise([-1, 1,-1, 1]))
print(testnoise([-1,-1, 1, 1]))
print(testnoise([-1,-1,-1,-1]))
```

```
False
False
False
False
```

همانطور که از نتایج به دست آمده مشخص است با تغییر ۷۵ درصد از الگوی ورودی نمی توان هیچ کدام از نتایج را به درستی به دست آورد .

اما در حالتی ۵۰ درصد داده ها از دست رفته باشند نیز بررسی شد و شبکه به درستی پاسخ داد. برای تست شبکه در حالت نویزی هم مشاهده شد که اگر ۵۰ درصد داده ها نویز داشته باشند شبکه قادر به تداعی نیست اما اگر تنها ۲۵ درصد داده ها نویز داشته باشند شبکه می تواند همه را به درستی تداعی کند .

۴. ابتدا با استفاده از فرمول زیر ماتریس Hebb را مقدار دهی اولیه می کنیم .

$$W = \sum_{p=1}^{P} s(p)s(p)^T - pI_n$$

سپس الگوریتم شبکه discrete Hopfield را در تابع DisHopfieldNet پیاده سازی می کنیم .

و برای قسمت از دست دادن اطلاعات ۴ ورودی را به شبکه می دهیم :

```
input : [1,0,0,0] . output :  [ 1  1  1 -1]
input : [0,1,0,0] . output :  [ 1  1  1 -1]
input : [0,0,1,0] . output :  [ 1  1  1 -1]
input : [0,0,0,-1] . output :  [ 1  1  1 -1]
```

همان طور که مشاهده می نماییم با از دست دادن ۷۵ درصد اطلاعات بر خلاف قسمت قبل می توانیم تمام ورودیها را به درستی تداعی کنیم . بنابراین مقاومت شبکه discrete Hopfield در برابر از دست دادن اطلاعات بالاتر از auto associative net with modified Hebb است .

برای تست قسمت نویز ۷۵ درصد در ورودی هم نتایج به صورت زیر است :

```
input : [1,-1,-1,1] . output :  [-1 -1 -1  1]
input : [-1,1,-1,1] . output :  [-1 -1 -1  1]
input : [-1,-1,1,1] . output :  [-1 -1 -1  1]
input : [-1,-1,-1,-1] . output :  [-1 -1 -1  1]
```

اما همانطور که می بینیم شبکه discrete Hopfield هم چنان در برابر ۷۵ درصد نویز مقاوم نیست .

همچنین شبکه را برای ۵۰ درصد نویز هم بررسی نمودیم و شبکه مقاوم نبود . اما در برابر ۲۵ دردصد نویز مقاوم بود .

بنابراین می توان گفت که شبکه Hopfield در از دست دادن اطلاعات مقاومت بیشتری دارد .

۵. با امتحان بردار های مختلف ۵ بردار زیر را می توان در شبکه ذخیره نمود :

```
S  = [ 1, 1, 1,-1]
S2 = [-1,-1, 1,-1]
S3 = [ 1, 1,-1, 1]
S4 = [-1,-1,-1, 1]
S5 = [-1,-1,-1,-1]
```

همان طور که می بینیم ۴ بردار اول همان هایی هستند که در سوال ۲ در شبکه ذخیره شد و بیشتر از این تعداد نتوانستیم ذخیره کنیم اما در این سوال و در شبکه Hopfield یک بردار دیگر هم توانستیم ذخیره کنیم یعنی ظرفیت شبکه یکی افزایش پیدا کرده است . البته در حالتی که از modified hebb استفاده نکنیم می توان بیشتر از این مقدار را نیز ذخیره نمود .

۶. الگوریتم discrete Hopfield بهتر عمل می کند زیرا هم حافظه بیشتری دارد هم اینکه مقاومتش در برابر از دست روفتن اطلاعات افزایش پیدا کرده است .

سوال ۴ – Recurrent Hetero-Associative Network

در ابتدا برای پیاده سازی شبکه BAM لازم است تا ماتریس Hebb را مقدار دهی اولیه کنیم .

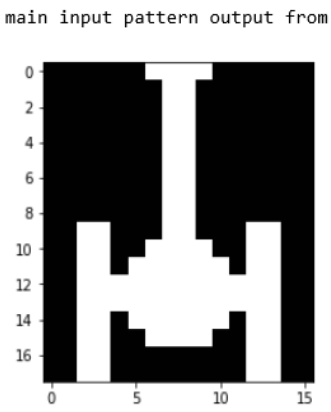
که به صورت زیر انجام می گیرد :

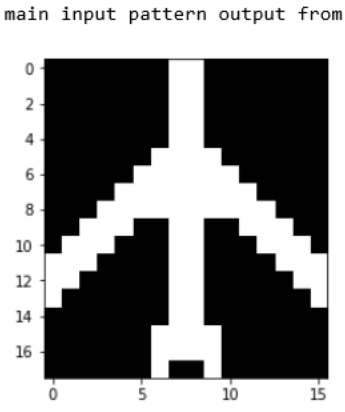
$$w_{ij} = \sum_{p} s_i(p) t_j(p)$$

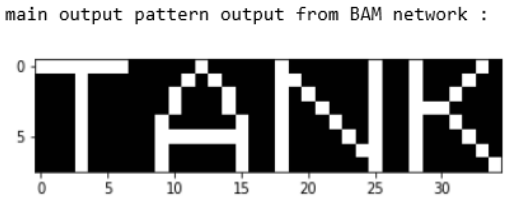
سپس الگوریتم BAM را در تابع BAM پیاده سازی می نماییم .

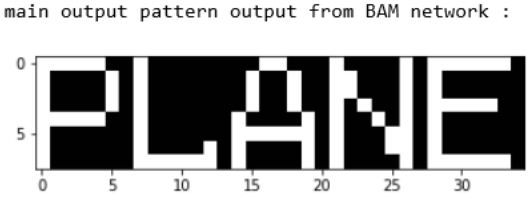
الف)

برای صحت عملکرد تابع BAM دو الگوی ورودی و خروجی را به تابع می دهیم و x,y نهایی را دریافت می کنیم تا ببینیم که آیا درست پاسخ می دهد یا خیر . نتایج به صورت زیر است :



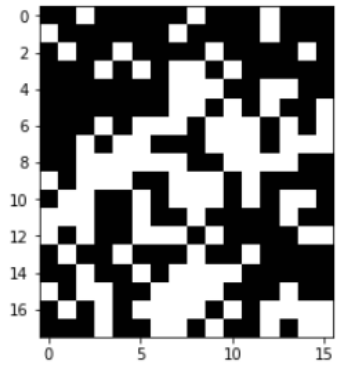




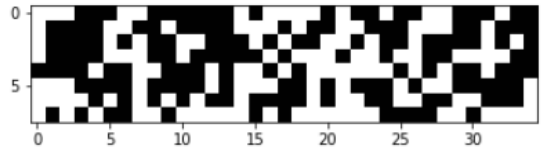


ب) فاصله hamming به معنای تعداد ورودی هایی از دو الگو است که با هم تفاوت دارند . برای ایجاد فاصله hamming ۳۰ درصد باید ۳۰ درصد از ورودی های الگوی ورودی و خروجی را تغییر می دهیم یعنی ۱ را ۱- و ۱- را ۱ می گذاریم . سپس به عنوان ورودی به شبکه BAM داده و x,y نهایی را دریافت نموده و نمایش می دهیم . الگوهای هواپیما به همراه نویز به صورت زیر هستند :

```
disturb pattern of plane input pattern :
```

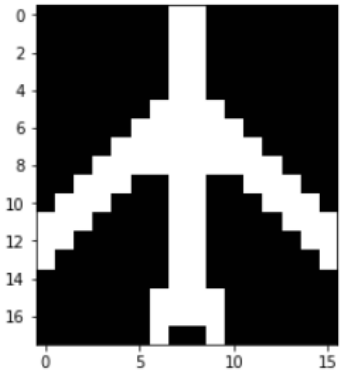


```
disturb pattern of plane output pattern :
```

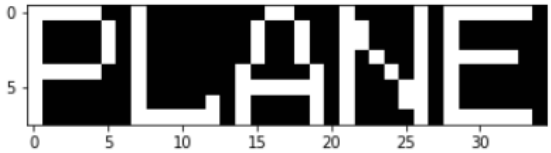


خروجی شبکه BAM برای ورودی های بالا :

```
x , output from BAM network :
```

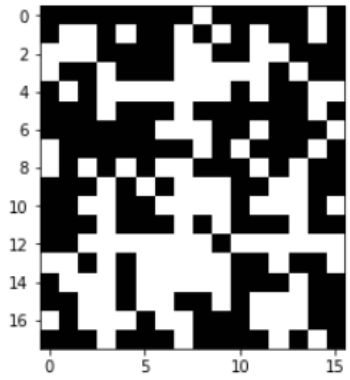


```
y , output from BAM network :
```

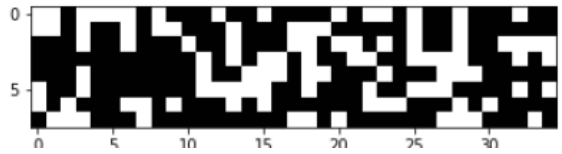


الگوی تانک به همراه نویز به صورت زیر است :

```
disturb pattern of tank input pattern :
```

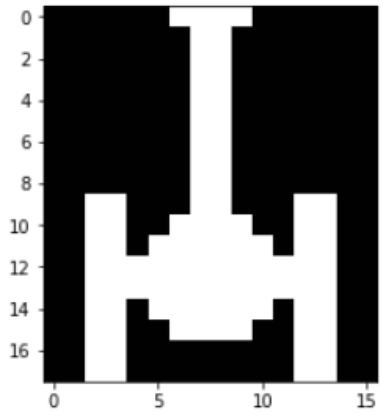


```
disturb pattern of tank output pattern :
```

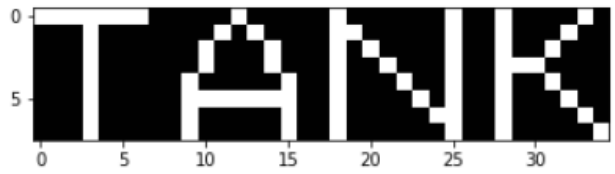


خروجی شبکه BAM به صورت زیر است :

```
x , output from BAM network :
```



```
y , output from BAM network :
```



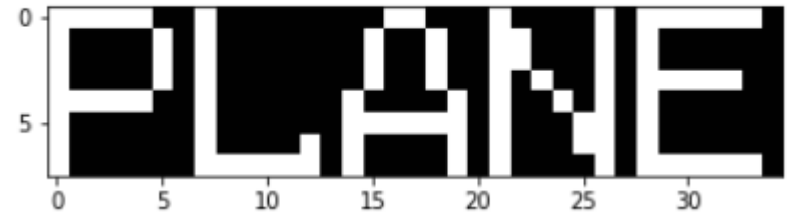
سپس برای هزار مورد نمونه داده ها راتست کرده تا دقت به دست آید و نتیجه به صورت زیر است :

```
noise(1000,0.3)
```

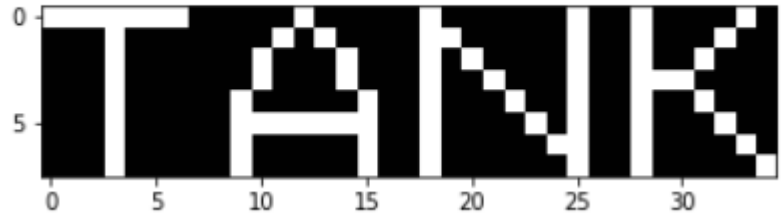
```
100.0
```

البته اگر منظور سوال این است که فقط الگوی ورودی نویز داشته باشد ، تنها ورودی را نویز دار کرده و خروجی ها به صورت زیر است:

```
y , output from BAM network :
```



```
y , output from BAM network :
```



شبکه را برای درصد های مختلفی از نویز بررسی نموده و نتایج زیر حاصل شد :

```
accuracy with 10% noise :  100.0
accuracy with 20% noise :  100.0
accuracy with 30% noise :  100.0
accuracy with 40% noise :  99.0
accuracy with 50% noise :  0.0
accuracy with 70% noise :  0.0
```

همانطور که می بینیم میزان مقاومت شبکه برای فاصله hamming حدود ۴۰ درصد است . یعنی اگربیشتر از ۴۰ درصد داده ها نویز داشته باشند شبکه قادر نخواهد بود الگوها را به درستی تداعی کند .این الگوریتم بهتر از سایر الگوریتم هایی که بررسی نموده ایم کار می کند زیرا به صورت بازگشتی عمل میکند و اطلاعات خروجی را در ورودی استفاده می نماید . اگر فاصله hamming با الگوهای اصلی کمتر

باشد شبکه بهتر می تواند پاسخ گو باشد . در هر بار تکرار شبکه در واقع ورودی و خروجی به الگوهای اصلی نزدیک و نزدیک تر می شوند و در نهایت الگویی را خروجی می دهند که به آن نزدیک تر باشند .