به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری چهارم

نام و نام خانوادگی	امیر محمد رنجبر پازکی
شماره دانشجویی	۸۱۰۱۹۹۳۴۰
تاریخ ارسال گزارش	۱۴۰۰/۴/۱۸

فهرست گزارش سوالات

سوال ۱ – SOM

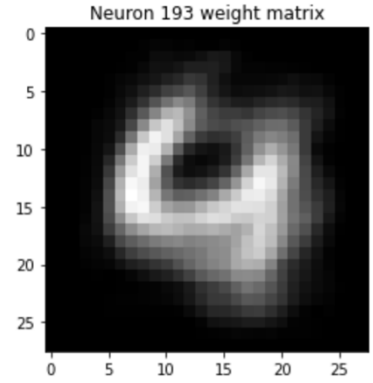
در این سوال یک شبکه‌ی SOM به صورت عمومی تعریف شده‌است تا خواسته‌ی بخش‌های مختلف را پاسخگو باشد.
این شبکه ابتدا وزن‌ها را از توزیع نرمالی با میانگین ۰.۵ و انحراف معیار ۰.۲ می‌گیرد تا وزن‌ها اکثرا بین ۰ و ۱ قرار بگیرند.
در این شبکه‌ها وزن‌های نورون‌ها خروجی سعی دارند تا به میانگین دسته‌ای از الگوهای ورودی نزدیک شوند. به همین دلیل، در نهایت این وزن‌ها بین صفر و یک خواهند بود.
این شبکه دارای یک تابع train برای نزدیک‌تر کردن وزن‌ها به به داده‌های آموزش و به نوعی یادگیری مرکز دسته‌هاست.
یک تابع test نیز برای اختصاص دادن داده‌ها به نورون‌های خروجی زده‌شده‌است که بر مبنای یافتن کمینه اختلاف وزن با بردار ورودی دسته را می‌یابد.
همه‌ی شبکه‌های این قسمت عکس‌های ۲۸ در ۲۸ را به عنوان ورودی می‌گیرند و با ۶۲۵ نورون(دسته) به عنوان خروجی سعی بر دسته‌بندی داده‌ها دارند. بدیهی است بخشی از این دسته‌ها هیچ داده‌ای را به خود جذب نکند.
داده‌ها mnist توسط keras بارگیری شده‌اند و ۲۰۰۰ نمونه برای آموزش و ۱۰۰۰ نمونه برای آزمون از آن‌ها انتخاب شده‌است. این نمونه‌ها به صورت تصادفی تولید شده‌اند تا سوگیری بابت انتخاب داده‌ها نداشته‌باشیم. توزیع داده‌ها در دسته‌های مختلف متوازن است.

```
Train labels:
{0: 189, 4: 202, 8: 201, 1: 233, 2: 181, 5: 180, 3: 225, 9: 188, 7: 217, 6: 184}
Test labels:
{0: 97, 1: 114, 7: 100, 2: 111, 5: 76, 4: 98, 3: 92, 8: 91, 6: 99, 9: 122}
```

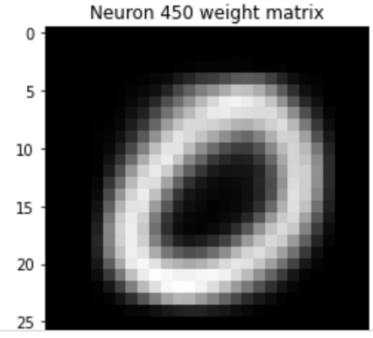
شکل ۱- تعداد داده‌ها در کلاس‌های مختلف داده‌ی آموزش و آزمون

الف) در این قسمت هیچ فاصله‌ای برای همسایگی در نظر گرفته‌نشده‌است تا شعاع مجاورت خطی صفر باشد. در حقیقت، در این قسمت هر نورون خروجی به صورت مستقل با باقی نورون‌ها رقابت می‌کند. این شبکه به تعداد ۴۰ epoch آموزش داده‌شده‌است و سپس، با استفاده از داده‌های آزمون تست شده‌است. تعداد labelها از هر کلاس و همچنین تصویر وزن‌ها(مرکز دسته‌ها) برای کلاس‌هایی که حداقل یک داده به آن‌ها رسیده‌است، خروجی داده‌شده‌است که در زیر قابل مشاهده است.

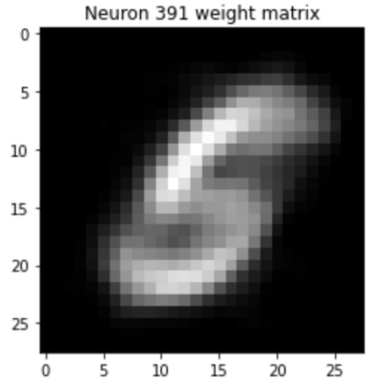
```
Train time: 123.08998656272888 s
Winner nodes count: 13
Node number: 193
Correct labels count in node: {0: 2, 1: 0, 2: 5, 3: 0, 4: 35, 5: 0, 6: 21, 7: 0, 8: 0, 9: 20}
```



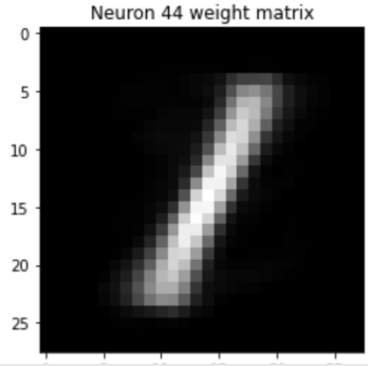

```
Node number: 450
Correct labels count in node: {0: 65, 1: 0, 2: 2, 3: 1, 4: 0, 5: 1, 6: 1, 7: 0, 8: 1, 9: 0}
```



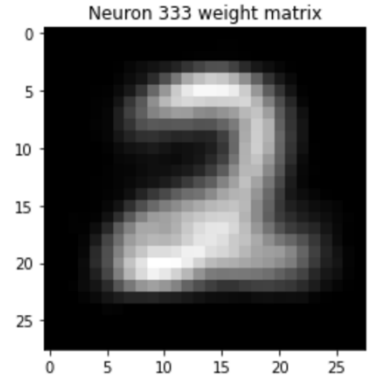
```
Node number: 391
Correct labels count in node: {0: 8, 1: 0, 2: 1, 3: 7, 4: 1, 5: 22, 6: 5, 7: 0, 8: 4, 9: 0}
```



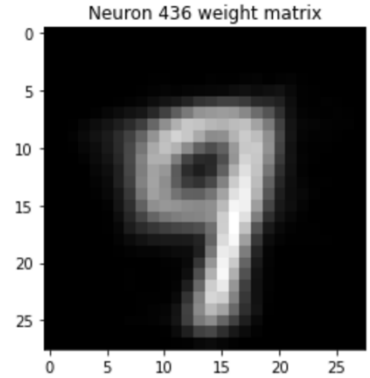

```
Node number: 44
Correct labels count in node: {0: 0, 1: 53, 2: 7, 3: 0, 4: 3, 5: 0, 6: 0, 7: 6, 8: 1, 9: 0}
```



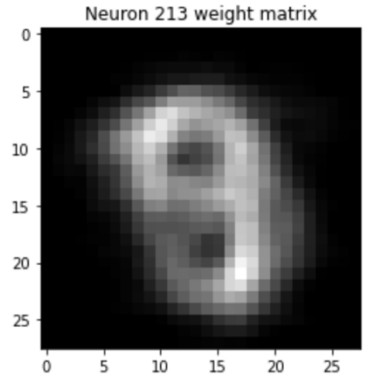

```
Node number: 333
Correct labels count in node: {0: 1, 1: 0, 2: 84, 3: 4, 4: 0, 5: 0, 6: 1, 7: 0, 8: 0, 9: 0}
```



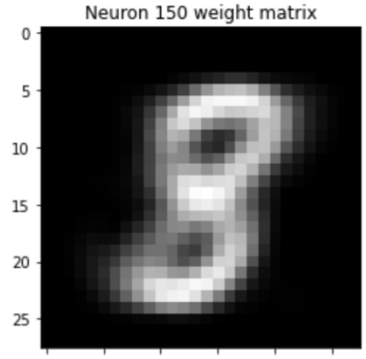

```
Node number: 436
Correct labels count in node: {0: 0, 1: 0, 2: 1, 3: 0, 4: 27, 5: 0, 6: 0, 7: 32, 8: 1, 9: 55}
```



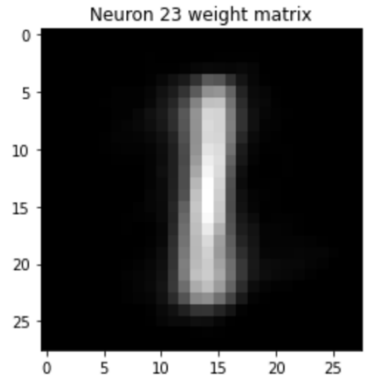
```
Node number: 213
Correct labels count in node: {0: 7, 1: 1, 2: 4, 3: 7, 4: 7, 5: 10, 6: 4, 7: 15, 8: 6, 9: 17}
```



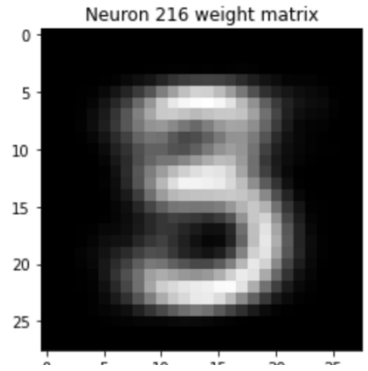

```
Node number: 150
Correct labels count in node: {0: 1, 1: 0, 2: 1, 3: 26, 4: 0, 5: 14, 6: 0, 7: 0, 8: 57, 9: 2}
```



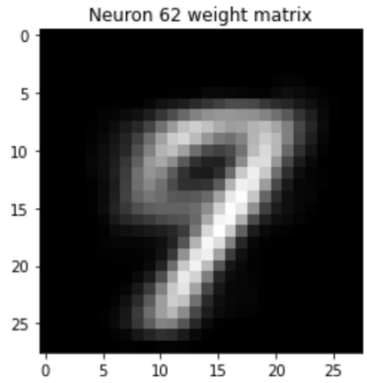

```
Node number: 23
Correct labels count in node: {0: 0, 1: 60, 2: 2, 3: 3, 4: 4, 5: 1, 6: 3, 7: 4, 8: 0, 9: 0}
```



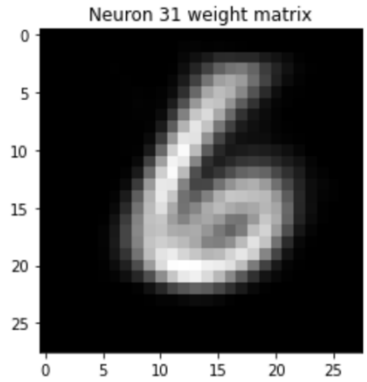

```
Node number: 216
Correct labels count in node: {0: 7, 1: 0, 2: 1, 3: 44, 4: 0, 5: 19, 6: 0, 7: 0, 8: 17, 9: 2}
```



```
Node number: 62
Correct labels count in node: {0: 0, 1: 0, 2: 1, 3: 0, 4: 16, 5: 3, 6: 0, 7: 40, 8: 2, 9: 25}
```




```
Node number: 31
Correct labels count in node: {0: 6, 1: 0, 2: 1, 3: 0, 4: 3, 5: 1, 6: 63, 7: 0, 8: 1, 9: 0}
```

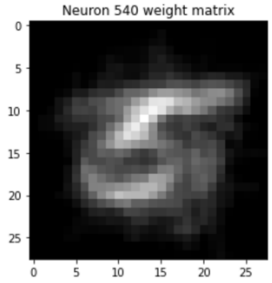


شکل ۲- خروجی شبکه‌ی **SOM** با شعاع مجاورت صفر

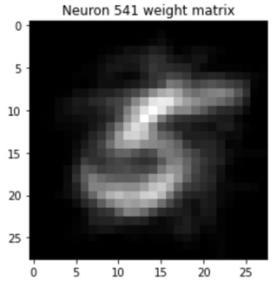
ب) برای در نظر گرفتن همسایگی خطی با شعاع مجاورت ۲، فاصله همسایه‌ها به صورت یک آرایه شامل ۲- و ۱- و ۱ و ۲ به کلاس موردنظر داده‌شده‌است. پس از آموزش مطابق قسمت قبل و تست با داده‌ی آزمون نتایج زیر به دست آمده‌است. در این قسمت ۸۸ نورون خروجی حداقل یک داده‌ی آزمون را جذب خود کرده‌اند. به همین دلیل، خروجی برای برخی از گره‌ها در زیر نشان داده‌شده‌است.

```
Train time: 126.44360041618347 s
Winner nodes count: 88
```

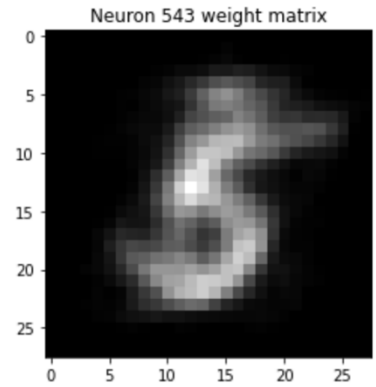
```
Node number: 540
Correct labels count in node: {0: 1, 1: 0, 2: 1, 3: 1, 4: 0, 5: 2, 6: 0, 7: 0, 8: 0, 9: 0}
```



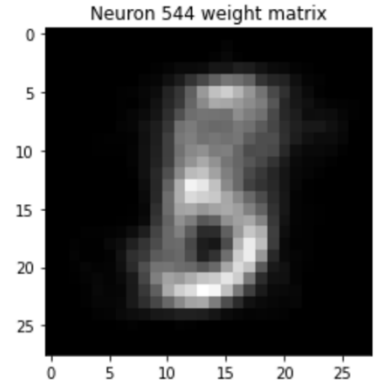

```
Node number: 541
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 1, 4: 0, 5: 6, 6: 0, 7: 1, 8: 0, 9: 0}
```



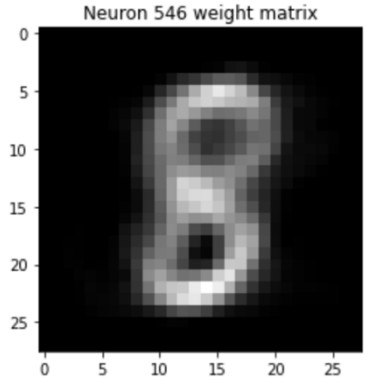
```
Node number: 543
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 2, 4: 0, 5: 1, 6: 1, 7: 0, 8: 0, 9: 0}
```



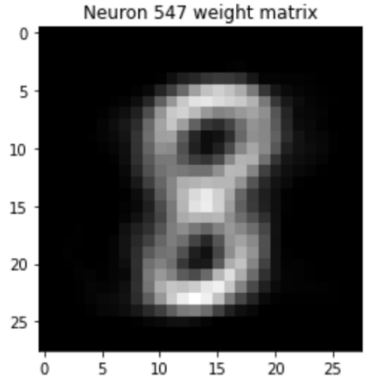

```
Node number: 544
Correct labels count in node: {0: 2, 1: 0, 2: 0, 3: 1, 4: 1, 5: 2, 6: 3, 7: 0, 8: 0, 9: 0}
```



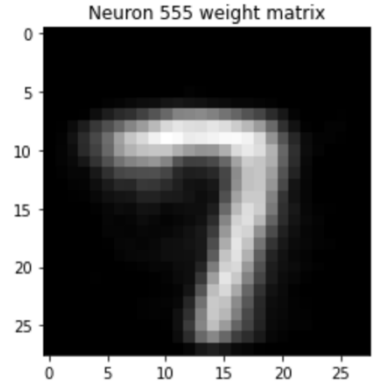

```
Node number: 546
Correct labels count in node: {0: 2, 1: 0, 2: 1, 3: 0, 4: 0, 5: 2, 6: 0, 7: 0, 8: 3, 9: 0}
```



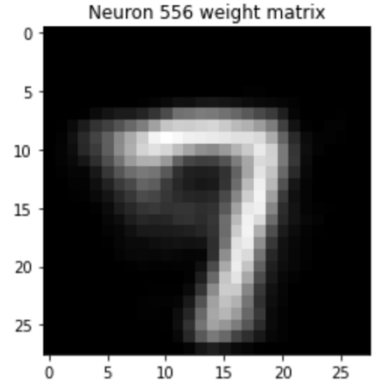

```
Node number: 547
Correct labels count in node: {0: 0, 1: 0, 2: 1, 3: 2, 4: 0, 5: 0, 6: 0, 7: 0, 8: 16, 9: 0}
```



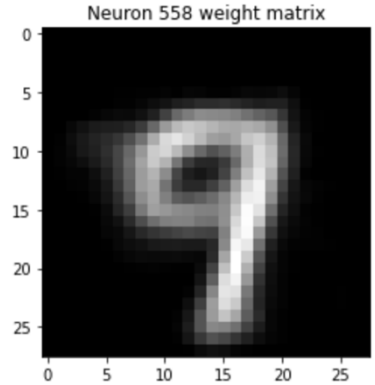
```
Node number: 555
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 3, 8: 0, 9: 0}
```



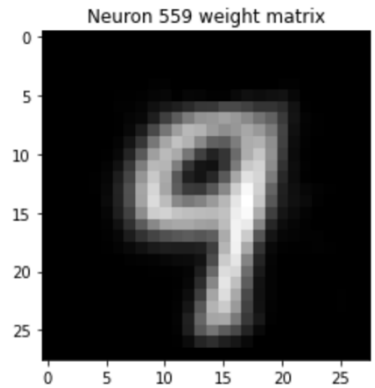

```
Node number: 556
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 11, 8: 0, 9: 0}
```



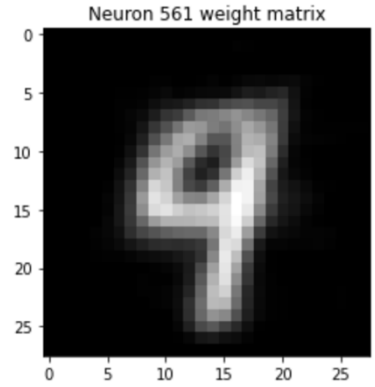

```
Node number: 558
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 17}
```



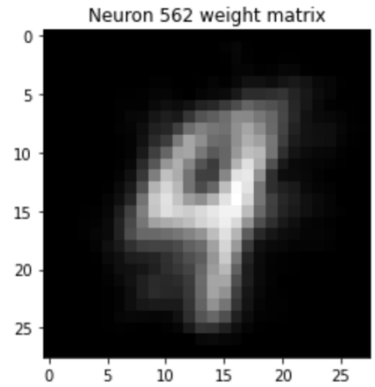

```
Node number: 559
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 8, 5: 0, 6: 0, 7: 0, 8: 0, 9: 23}
```



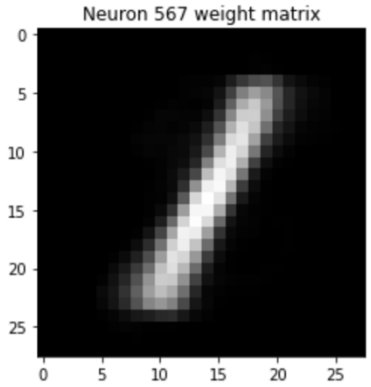
```
Node number: 561
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 9, 5: 0, 6: 0, 7: 0, 8: 0, 9: 11}
```



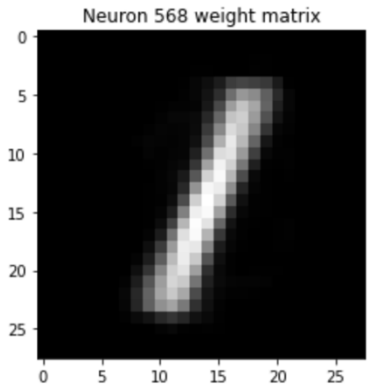

```
Node number: 562
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 6, 5: 0, 6: 0, 7: 0, 8: 0, 9: 4}
```



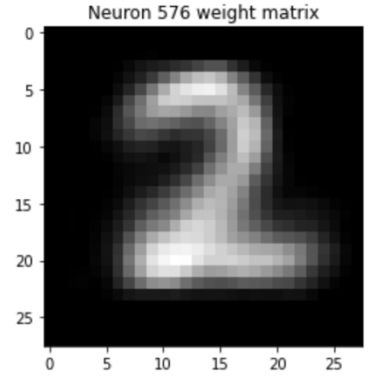

```
Node number: 567
Correct labels count in node: {0: 0, 1: 17, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 2, 8: 0, 9: 0}
```



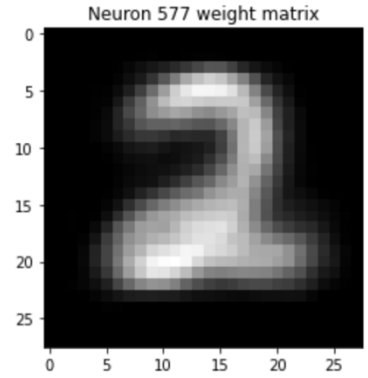

```
Node number: 568
Correct labels count in node: {0: 0, 1: 11, 2: 1, 3: 0, 4: 0, 5: 0, 6: 0, 7: 2, 8: 0, 9: 0}
```



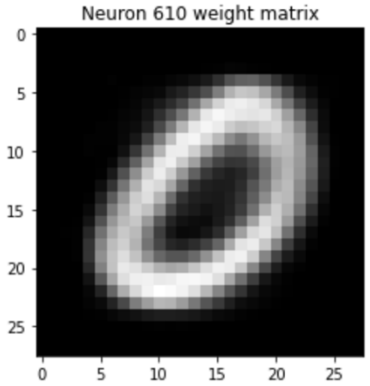
```
Node number: 576
Correct labels count in node: {0: 0, 1: 0, 2: 25, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



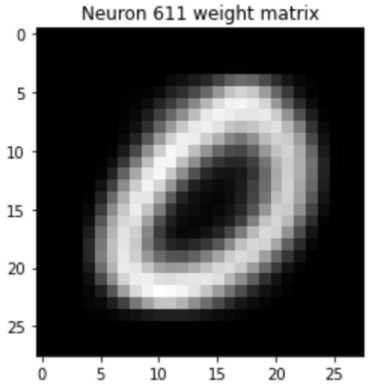

```
Node number: 577
Correct labels count in node: {0: 0, 1: 0, 2: 8, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



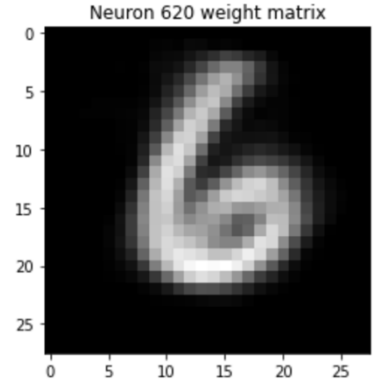

```
Node number: 610
Correct labels count in node: {0: 15, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



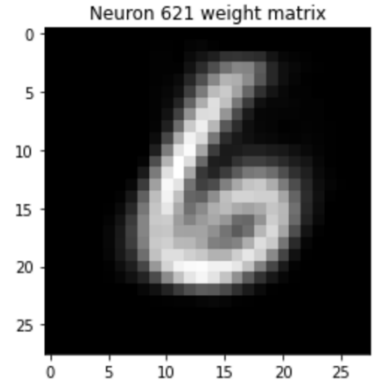

```
Node number: 611
Correct labels count in node: {0: 6, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



```
Node number: 620
Correct labels count in node: {0: 1, 1: 0, 2: 1, 3: 0, 4: 0, 5: 0, 6: 16, 7: 0, 8: 0, 9: 1}
```




```
Node number: 621
Correct labels count in node: {0: 1, 1: 0, 2: 0, 3: 0, 4: 1, 5: 0, 6: 10, 7: 0, 8: 0, 9: 0}
```



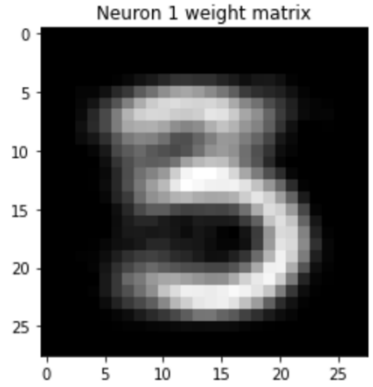
شکل ۳- برخی از خروجی‌های شبکه‌ی **SOM** با شعاع مجاورت خطی **R=2**

همانطور که مشاهده می‌کنید، افزودن شعاع مجاور باعث می‌شود تا نورون‌ها با یکدیگر همکاری کنند و راحت تر به الگوها نفوذ کنند و بتوانند مرکز دسته‌ی آن‌ها را بیابند. با استفاده از این کار تعداد دسته‌ها کشف‌شده بیشتر شده‌است چرا که نورون‌های بیشتری به میان الگوها نفوذ کرده‌اند و در نهایت، از ۱۳ دسته به ۸۸ دسته‌رسیده‌ایم. دسته‌های تشکیل شده تمرکز خوب دارند و وزن‌های آن‌ها به دسته موردنظر بسیار شبیه شده‌است که این عملکرد خوب شبکه را می‌رساند.
همچنین نسبت به قسمت الف تمام ارقام دسته‌هایی شناسایی شده دارند که در بالا دو نمونه از هر رقم آورده‌شده‌است. از هر رقم چند دسته وجود دارد که به دلیل همراهی نورون‌ها و به روزرسانی آن‌ها با یکدیگر است. همانطور که می‌بینید نورون‌های مجاور هم الگوی مشابه را تشخیص داده‌اند.

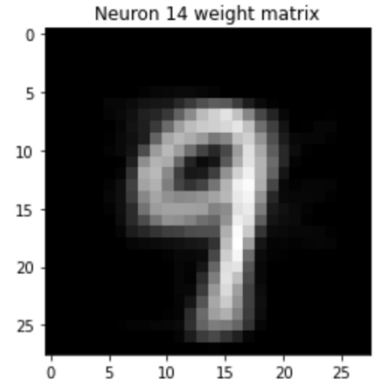
ج) برای در نظر گرفتن همسایگی مربعی با شعاع یک از همان شبکه فقط با درنظر گرفتن ساختار دوبعدی وزن و مپ کردن اندیس‌های بالا و پایین استفاده شده‌است. همانطور که در صورت سوال گفته‌شده‌است وزن‌های نورون‌ها با حداقل یک خروجی بسیار نزدیک به الگوی آن نورون هستند. این به دلیل سعی شبکه بر کمینه کردن فاصله‌ی بین وزن‌ها و الگوهای آن دسته است. در این جا نیز به دلیل مشابه آن چه در تحلیل قسمت قبل ذکر شد، تعداد دسته‌ها افزایش داشته‌است و دسته الگوهای بیشتری شناسایی شده‌اند که به دلیل همکاری بیشتر نورون‌ها با یکدیگر است.
۲ یا ۳ نمونه از خروجی‌های این بخش در زیر آمده‌است.

```
Train time: 131.9727520942688 s
Winner nodes count: 327
```

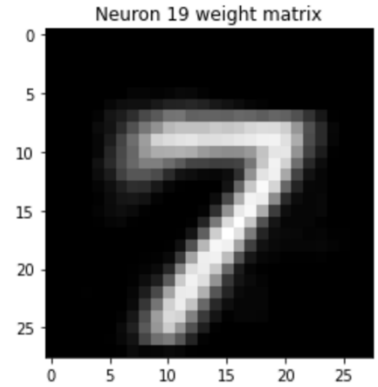
```
Node number: 1
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 5, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



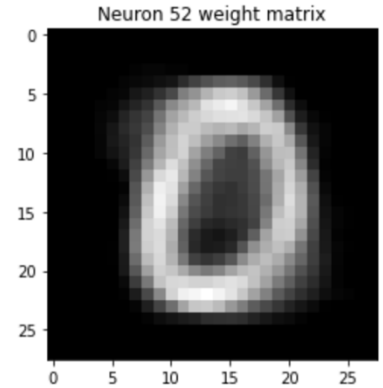

```
Node number: 14
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 10}
```




```
Node number: 19
Correct labels count in node: {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 7, 8: 0, 9: 0}
```




```
Node number: 52
Correct labels count in node: {0: 5, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0}
```



شکل ۴- ۴ نمونه از خروجی‌های شبکه‌ی **SOM** با همسایگی مربعی به شعاع **R=1**

همانطور که مشاهده می‌کنید این وزن‌ها بسیار شبیه به الگوهایی هستند که به این نورون اختصاص داده‌شده‌اند.
لازم به ذکر است در این سوال به دلیل گفته‌نشده نحوه‌ی کاهش شعاع همسایگی، شعاع همسایگی تا انتها ثابت درنظر گرفته‌شده‌است. اگر شعاع همسایگی کاهش یابد، گروه‌های نزدیک با هم به رقابت می‌پردازند و ممکن است از تعداد دسته‌ها کاسته شود.

سوال ۲ – MaxNet

در این سوال الگوریتم و شبکه‌ی MaxNet در قالب یک کلاس پیاده‌سازی شده‌است. این شبکه‌ی در هر به روزرسانی عملا جمع وزن‌دار ورودی‌هایش را حساب می‌کند که ورودی خودش به خودش ۱ و مابقی یال‌ها ϵ است. با این کار در هر مرحله به میزان ϵ در مجموع گره‌های دیگر از وزن یک گره کاسته‌می‌شود.

این الگوریتم با استفاده از بردار اولیه داده‌شده مورد آزمایش قرار گرفته‌است. مراحل به روزرسانی گره‌ها به همراه مقدار بیشینه در زیر گزارش شده‌است.

```
Initial values:
1.2 1.1 0.5 1.5 1.13 0.8
Iteration 1 values:
0.5461 0.43310000000000015 0 0.8851 0.46699999999999997 0.09410000000000007
Iteration 2 values:
0.30179100000000003 0.17410100000000012 0 0.6848609999999999 0.21240799999999993 0
Iteration 3 values:
0.16251290000000004 0.018223200000000106 0 0.595382 0.061510109999999896 0
Iteration 4 values:
0.07474790970000003 0 0 0.5638899927 0 0
Iteration 5 values:
0.0014422106490000336 0 0 0.554172764439 0 0
Iteration 6 values:
0 0 0 0.55398527705463 0 0
Max value is: 1.5
```

شکل ۵- مقدار گره‌ها در هر **iteration** به همراه مقدار به دست آمده گره نهایی به عنوان بیشینه

همانطور که مشاهده می‌کنید این شبکه به درستی عمل کرده‌است.

برای پیداکردن بیشینه قدرمطلق اعداد از سه شبکه‌ی MaxNet استفاده شده‌است. این شبکه جدید در قالب کلاس MaxAbsNet پیاده‌سازی شده‌است. این شبکه به این صورت عمل می‌کند که ابتدا با گذراندن بردار اصلی از یک شبکه‌ی MaxNet بیشینه مقادیر مثبت را می‌یابد. سپس، با ضرب کردن تمام اعداد در ۱- و دادن آن به یک شبکه‌ی MaxNet بیشینه قدرمطلق در اعداد منفی را می‌یابد. در نهایت، بین این دو بیشینه، با استفاده از یک MaxNet دیگر و ورودی دادن این دو عدد بیشینه قدرمطلق را در کل می‌یابد.

```
class MaxAbsNet:

  def __init__(self, values, epsilon):
    self.values = values
    self.epsilon = epsilon

  def find_max_abs(self):
    max_net_1 = MaxNet(self.values.copy(), epsilon)
    positive_max = max_net_1.find_max()

    negative_values = [-1*element for element in self.values]
    max_net_2 = MaxNet(negative_values, epsilon)
    negative_max = max_net_2.find_max()

    max_list = [positive_max, negative_max]
    max_net_3 = MaxNet(max_list, epsilon)
    return max_net_3.find_max()
```

شکل ۶- الگوریتم پیاده‌شده برای پیداکردن بیشینه قدرمطلق اعداد

در زیر خروجی این شبکه به ازای بردار دوم داده‌شده آورده‌شده‌است که درست محاسبه شده‌است. بیشینه قدرمطلق ۱.۵ است که مربوط به ۱.۵- است.

```
Max Absolute Value is: 1.5
```

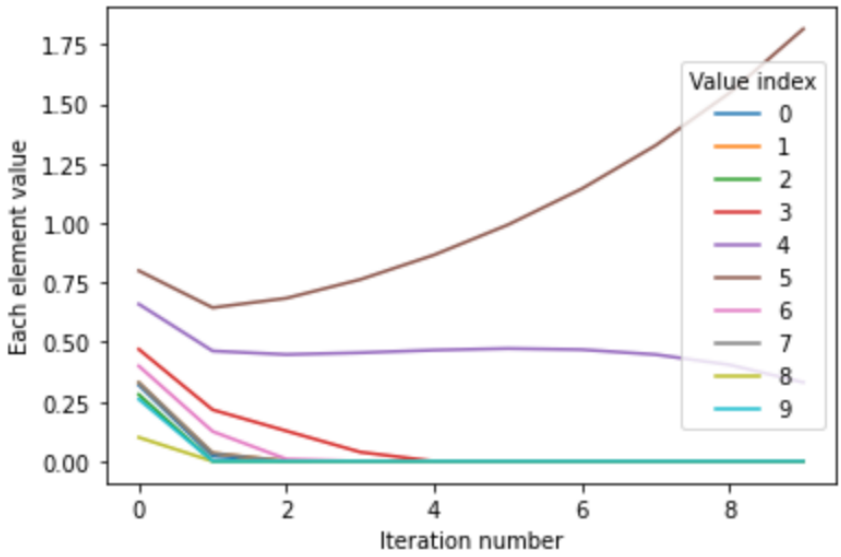
شکل ۷- خروجی شبکه برای بردار دوم (بیشینه قدرمطلق اعداد)

سوال ۳ – Mexican Hat

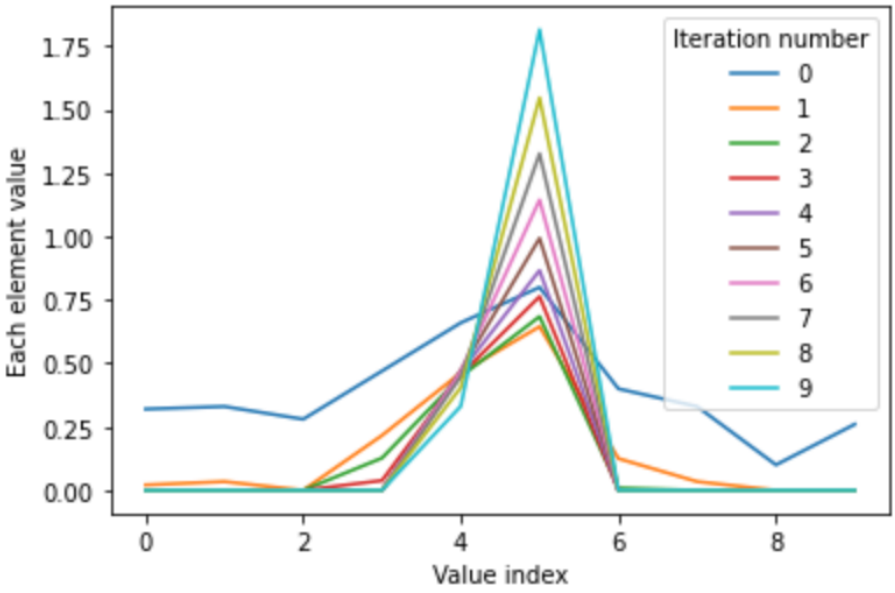
الگوریتم MexicanHat که به نوعی برای یافتن بیشینه نرم است در قالب کلاس MexicanHat پیاده‌سازی شده‌است. این الگوریتم با مکانیزم همکاری گره‌های همسایه و منازعه گره‌های کمی دورتر سعی بر پیداکردن ناحیه و یا همسایگی دارد که مقادیر در آن بیشینه است. به همین دلیل مرکز قسمتی که همسایگی قدرتمندتری است، بالاتر می‌رود و به حالت یک کلاه مکزیکی در می‌آید.
تابع فعالسازی گفته‌شده در دل این کلاس پیاده‌سازی شده‌است. در صورت سوال گفته‌شده‌است که بیشینه مقدار آرایه را بیابیم. به همین دلیل نیاز به استفاده از یک MaxNet بر روی خروجی این MexicanHat است تا به ما بیشینه عنصر را بدهد. البته این بیشینه عنصر لزوما بیشینه آرایه نیست و مرکز دسته‌ای است که بیشینه همسایگی آرایه است. البته در ورودی داده‌شده‌ی این سوال این دو مورد هم ارز هستند و مرکز همسایگی بیشینه، بیشینه آرایه نیز هست و خروجی در نهایت درست است.
دو حالت گفته‌شده با شبکه اجرا شده‌اند که نتایج آن در زیر قابل مشاهده‌است.
حالت اول. شعاع همکاری صفر و شعاع منازعه بی‌نهایت در نظر گرفته‌شده‌است. البته در پیاده سازی شعاع منازعه طول آرایه لحاظ شده‌است تا همه عناصر را در بر بگیرد. این شبکه در حقیقت MaxNet است چرا که تمام عناصر به یکدیگر به منازعه می‌پردازند تا در انتها عنصر بیشینه باقی بماند. MaxNet حالت خاص MexicanHat است. خروجی گام به گام شبکه‌ی MexicanHat به همراه مقدار خروجی از MaxNet نهایی به عنوان خروجی در زیر قابل مشاهده است.
همچنین نمودارهای مقدار هر عنصر در iterationهای مختلف، وضعیت کل لیست در iterationهای مختلف و سیگنال خروجی در iterationهای مختلف در زیر آمده‌است.

```
0.32 0.33 0.28 0.47 0.66 0.8 0.4 0.33 0.1 0.26
Iteration: 1
0.02100000000000002 0.033999999999999975 0 0.21599999999999986 0.463 0.6449999999999999 0.12499999999999989 0.033999999999999975 0 0
Iteration: 2
0 0 0 0.12699999999999984 0.44810000000000005 0.6846999999999999 0.008699999999999875 0 0 0
Iteration: 3
0 0 0 0.038249999999999784 0.45568000000000014 0.7632599999999998 0 0 0 0
Iteration: 4
0 0 0 0 0.4666650000000002 0.8665189999999997 0 0 0 0
Iteration: 5
0 0 0 0 0.4733461000000002 0.9931562999999995 0 0 0 0
Iteration: 6
0 0 0 0 0.4686996900000002 1.1444529499999994 0 0 0 0
Iteration: 7
0 0 0 0 0.4479943330000003 1.326473570999999 0 0 0 0
Iteration: 8
0 0 0 0 0.4049458425000004 1.5469688518999987 0 0 0 0
Iteration: 9
0 0 0 0 0.33123812581000056 1.8158680380299983 0 0 0 0
Max value is: 0.8
```

شکل ۸- مقادیر **MexicanHat** در **Iteration**های مختلف و سیگنال خروجی حالت اول

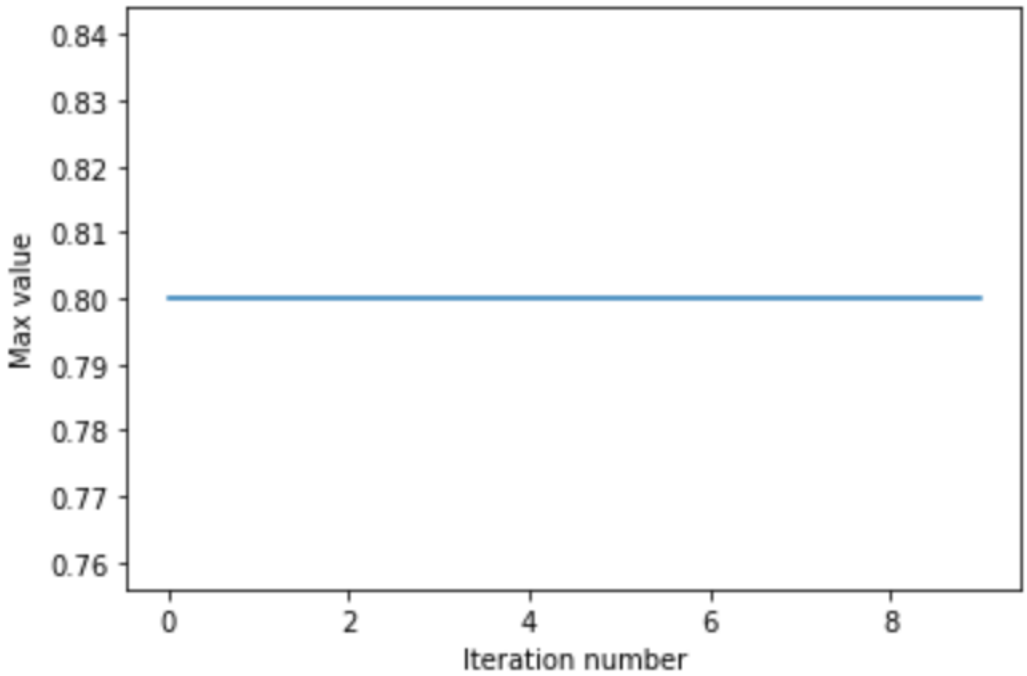


شکل ۹- نمودار مقدار عنصر بر حسب شماره **iteration** در حالت اول



شکل ۱۰- نمودار مقدار عنصر بر حسب شماره عنصر در **iteration**های مختلف در حالت اول

همانطور که مشاهده می‌کنید شکل کلاه مکزیکی و تشدید آن کاملا مشهود است.



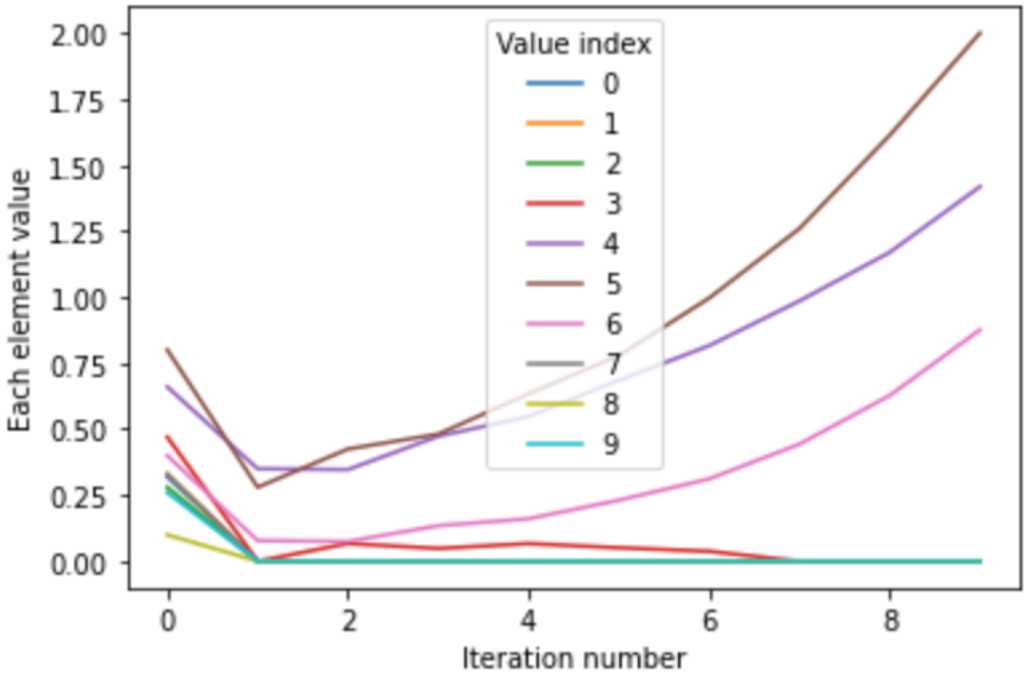
شکل ۱۱- نمودار سیگنال خروجی در هر **iteration** حالت اول

این الگوریتم برای ۱۰ iteration اجرا شده‌است.

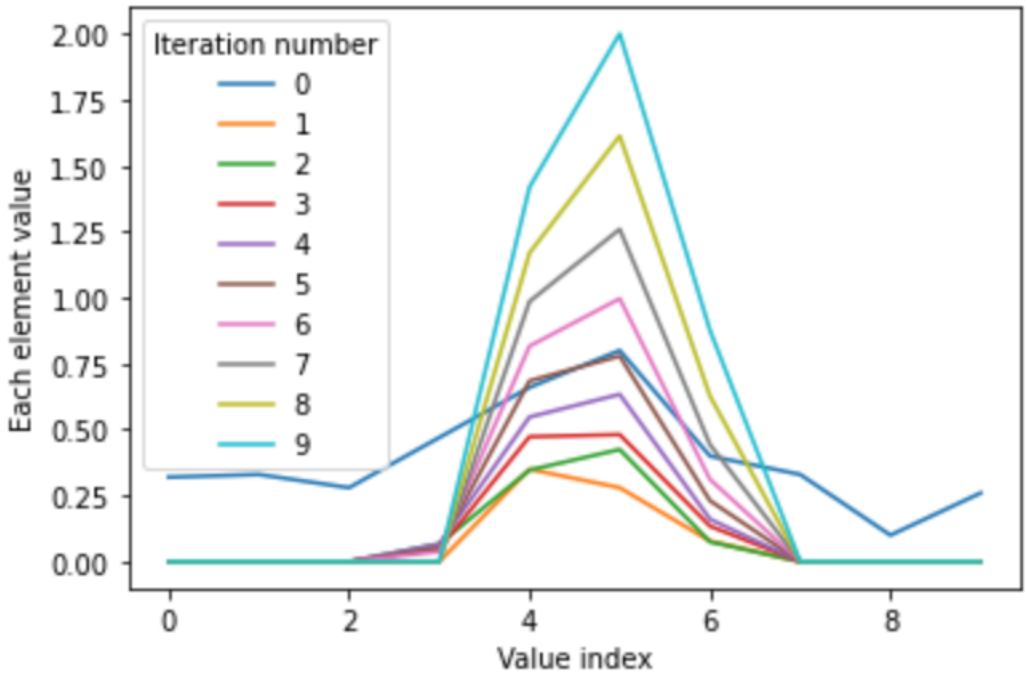
حالت دوم. شعاع همکاری ۱ و شعاع منازعه ۵ در نظر گرفته‌شده‌است. این حالت نرم MexicanHat است. خروجی گام به گام شبکه‌ی MexicanHat به همراه مقدار خروجی از MaxNet نهایی به عنوان خروجی در زیر قابل مشاهده است. همچنین نمودارهای مقدار هر عنصر در iterationهای مختلف، وضعیت کل لیست در iterationهای مختلف و سیگنال خروجی در iterationهای مختلف در زیر آمده‌است.

```
0.32 0.33 0.28 0.47 0.66 0.8 0.4 0.33 0.1 0.26
Iteration: 1
0 0 0 0 0.34999999999999964 0.2799999999999999 0.07799999999999985 0 0 0
Iteration: 2
0 0 0 0.06679999999999986 0.3467999999999998 0.4247999999999996 0.07479999999999998 0 0 0
Iteration: 3
0 0 0 0.04831999999999992 0.4731199999999995 0.48111999999999966 0.13431999999999988 0 0 0
Iteration: 4
0 0 0 0.0666879999999998 0.5478079999999994 0.6338079999999993 0.16068799999999991 0 0 0
Iteration: 5
0 0 0 0.05089919999999987 0.6847071999999991 0.7787071999999993 0.23089919999999978 0 0 0
Iteration: 6
0 0 0 0.03752127999999966 0.8162284799999989 0.9962284799999989 0.31152127999999996 0 0 0
Iteration: 7
0 0 0 0 0.9853784319999983 1.2593784319999988 0.4431499519999998 0 0 0
Iteration: 8
0 0 0 0 1.1695941375999983 1.612744089599998 0.6273656575999996 0 0 0
Iteration: 9
0 0 0 0 1.418456673279998 2 0.8762281932799991 0 0 0
Max value is: 0.8
```

شکل ۱۲- مقادیر **MexicanHat** در **Iteration**های مختلف و سیگنال خروجی حالت دوم

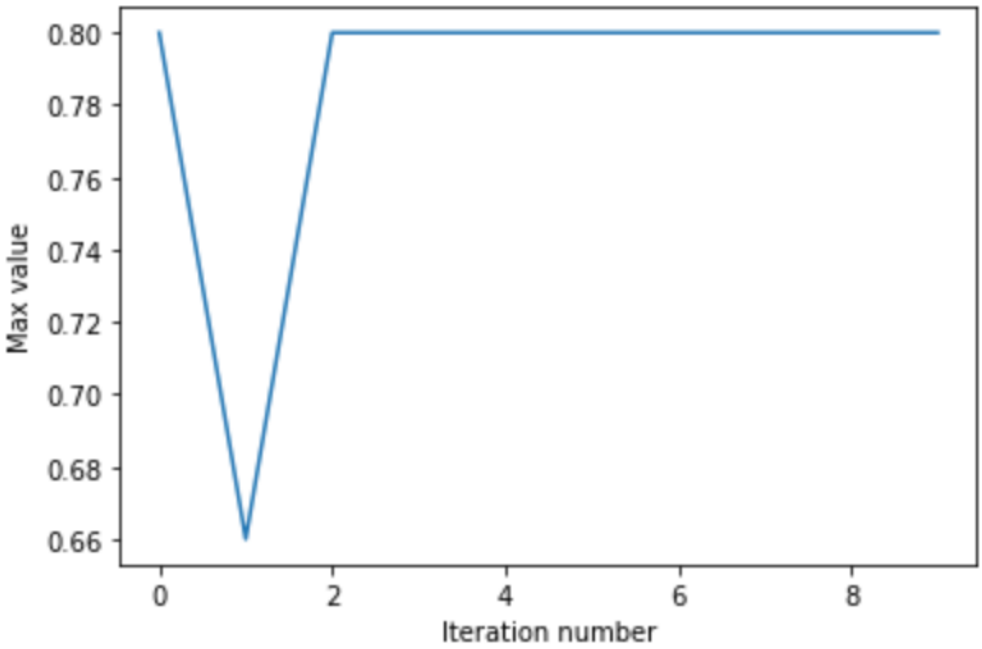


شکل ۱۳- نمودار مقدار عنصر بر حسب شماره **iteration** در حالت دوم



شکل ۱۴- نمودار مقدار عنصر بر حسب شماره عنصر در **iteration**های مختلف در حالت دوم

همانطور که مشاهده می‌کنید شکل کلاه مکزیکی و تشدید آن کاملا مشهود است.



شکل ۱۵- نمودار سیگنال خروجی در هر **iteration** حالت دوم

این الگوریتم برای ۱۰ iteration اجرا شده‌است. همانطور که مشاهده می‌کنید به دلیل نرم بودن بیشینه MexicanHat در iteration ۱ مقدار ۰.۶۶ بیشینه شده‌است. این الگوریتم بیشینه همسایگی را می‌یابد. البته در نهایت مقدار درست به دست‌آمده‌است. برای پیدا کردن پیشینه مطلق باید از تنظیمات حالت یک استفاده کرد.

سوال ۴ – Hamming Net

الف) شبکه‌ی HammingNet شبکه‌ای است که میزان شباهت هر بردار ورودی را با هر یک از بردارهای پایه خود به دست می‌آورد و در پایان برداری که بیشینه شباهت را به بردار ورودی داشته‌است، بازگردانده می‌شود و گویی به بردار ورودی assign می‌شود.

بردارها به صورت biplor هستند. یعنی، شامل ۱ و ۱- هستند. به همین دلیل، ضرب داخلی دو بردار را می‌توان به شکل زیر نیز نوشت.

$$x.y = a - d$$

که در آن a تعداد بیت‌های برابر(حاصل ضرب ۱) است و d تعداد بیت‌های نابرابر(حاصل ضرب ۱-) است.
همچنین، می‌توان نوشت.

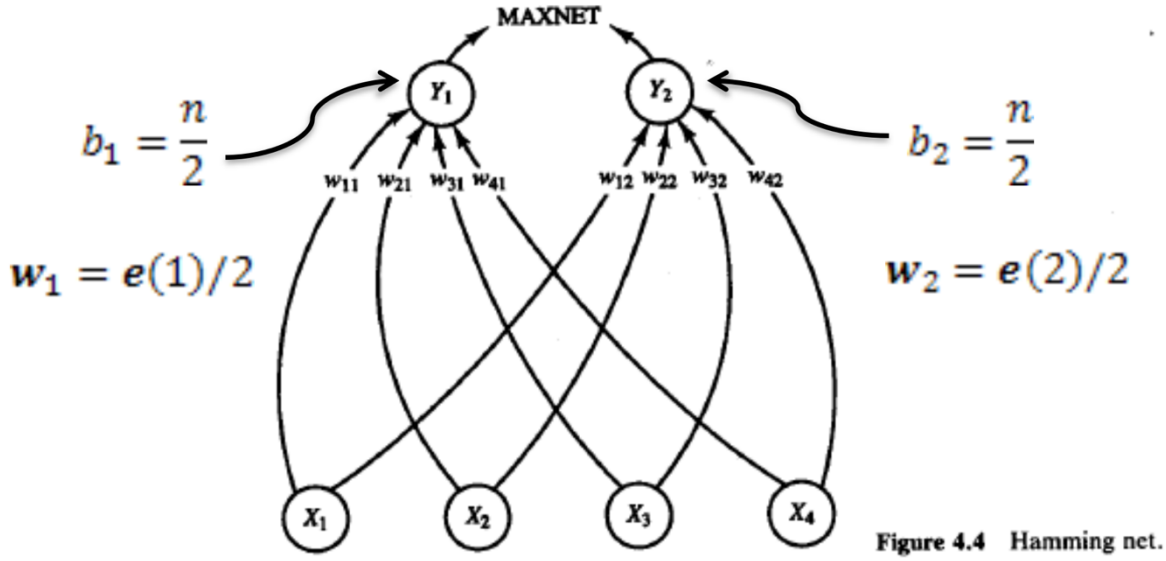
$$n = a + d$$

که n تعداد بیت‌های بردار است.

با حذف d در بین این دو معادله داریم.

$$a = x\frac{y}{2} + \frac{n}{2}$$

که a میزان شباهت (تعداد بیت‌های یکسان) دو بردار است.
پس، با شبکه‌ای با ساختار زیر می‌توان شباهت با هر بردار پایه را به دست آورد.



شکل ۱۶- شماتیک کلی **HammingNet**

در قسمت پایین این شبکه به تعداد بردارهای پایه خروجی داریم که وزن‌های هر کدام عناصر بردار پایه تقسیم بر دو هستند. تعداد ورودی‌های نیز به تعداد عناصر بردار است که هر عنصر روی آن قرار می‌گیرد. مقدار bias نیز n تقسیم بر دو قرار داده‌می‌شود تا با محاسبه مجموع ضرب وزن‌ها در مقادیر ورودی و جمع با bias در هر گره معادله a شکل بگیرد که بیانگر تعداد بیت‌های یکسان ورودی اعمال‌شده با بردار پایه متناظر آن گره باشد. حال کافی است این مقادیر را به عناون ورودی‌های یک MaxNet بدهیم تا این شبکه برای ما بیشینه مقادیر را برگرداند. البته این شبکه جایگاه را برمی‌گرداند تا بتوانیم بردار پایه‌ای را که بیشترین میزان شباهت با بردار ورودی را دارد، به عنوان خروجی بازگردانیم.
این شبکه در قالب کلاس HammingNet مشابه توضیحات داده‌شده پیاده‌سازی شده‌است.
ب) بردارهای گفته‌شده به شبکه‌های HammingNet با بردارهای پایه گفته‌شده در بخش قبل داده‌شدند و نتیجه برای هر بردار ورودی به صورت زیر بود.

```
Main vector:
[1, 1, 1, 1, 1, 1]
Most similiar vector(s):
[[1, -1, 1, -1, 1, -1], [1, 1, 1, -1, -1, -1]]
```

شکل ۱۷- خروجی شبکه‌ی **HammingNet**(شبیه‌ترین بردار پایه) برای بردار **V1**

برای این بردار لازم به ذکر است که دو بردار پایه با بیشینه شباهت یعنی ۳ بیت یکسان وجود دارد. بنابراین، اگر شبکه‌ی MaxNet یک کران بالا برای تعداد iterationها نداشته‌باشد، نمی‌تواند این بیشینه را بیابد و در loop بی‌نهایت گیر می‌کند. پس از تعداد بیشینه iteration اولین بردار متناظر مقدار ناصفر برگردانده شده‌است.

```
Main vector:
[-1, 1, -1, -1, 1, 1]
Most similiar vector:
[-1, 1, -1, 1, -1, -1]
```

شکل ۱۸- خروجی شبکه‌ی **HammingNet**(شبیه‌ترین بردار پایه) برای بردار **V2**

```
Main vector:
[-1, -1, 1, 1, 1, 1]
Most similiar vector:
[1, -1, 1, -1, 1, -1]
```

شکل ۱۹- خروجی شبکه‌ی **HammingNet**(شبیه‌ترین بردار پایه) برای بردار **V3**

```
Main vector:
[-1, -1, 1, 1, -1, 1]
Most similiar vector:
[-1, 1, -1, 1, -1, -1]
```

شکل ۲۰- خروجی شبکه‌ی **HammingNet**(شبیه‌ترین بردار پایه) برای بردار **V4**

```
Main vector:
[-1, 1, 1, -1, -1, -1]
Most similiar vector:
[1, 1, 1, -1, -1, -1]
```

شکل ۲۱- خروجی شبکه‌ی **HammingNet**(شبیه‌ترین بردار پایه) برای بردار **V5**