تمرین چهارم شبکه عصبی و یادگیری عمیق

محمد هاشمی
810197423
md.hashemi@u.ac.ir

چکیده – در این تمرین هدف استفاده از شبکه *SOM* برای یادگیری شبکه‌ای منوط بر تشخیص داده‌های دیتاست *MNITS* می‌باشد. در قسمت دوم این گذارش به طراحی شبکه‌ای مبنی بر ساختار *Recurrent Neural Networks* جهت تشخیص بیماری بر اساس انتخاب 6 نشانه شاخص از 11 نشانه در یک داده از بیمار است.

کلید واژه- SOM,Recurrent Neural Networks,MNITS

1- مقدمه

وقتی با داده‌های زبانی (مثلا متون) کار می‌کنیم اغلب با توالی از حروف، کلمات یا جملات سروکار داریم. در بیشتر موارد ترتیب این توالی برای ما اهمیت دارد. شبکه‌های بازگشتی این امکان را به ما می‌دهند تا یک توالی با طول نامشخص را به یک بردار با اندازه ثابت بازنمایی کنیم و در عین حال بسیاری از خواص نحوی و ساختاری توالی ورودی را حفظ کنیم. به عقیده بعضی محققان این شبکه‌ها مهم‌ترین نوآوری شبکه عصبی در حوزه پردازش زبان طبیعی تاکنون بوده‌اند.

به‌طورکلی، شبکه بازگشتی (RNN) تابعی است که یک ورودی با طول غیرثابت (مثلا جمله) به صورت دنباله‌ای از n بردار din بعدی دریافت می‌کند و یک بردار خروجی y با ابعاد dout را بازمی‌گرداند:

$$y_n = RNN(x_{1:n})$$

$$x_n \in \mathbb{R}^{d_{in}},\ y_n \in \mathbb{R}^{d_{out}}$$

که در عبارت بالا x1:n جمله ورودی ماست که کلمات آن به بردارهای نهفته مثلا word2vec نگاشت و به یکدیگر چسبانده شده‌اند. این بردارها به عنوان ورودی شبکه بازگشتی (RNN) داده و در نهایت یک خروجی yn به ما بازگردانده میشود.
خروجی yn که گاهی اوقات با hn هم نمایش داده میشود در کارهای مختلف می‌تواند مورد استفاده قرار گیرد. مثلا فرض کنید یک تسک بانظارت (supervised) مثل طبقه‌بندی اسناد یا تحلیل متن (sentiment) داریم. برای هر سند یک بازنمایی yn بدست آورده میشود و سپس این yn به عنوان ورودی classifier داده میشود و شبکه آموزش داده میشود.

2- پیاده‌سازی شبکه ROM برای تشخیص داده‌های دیتاست MNITS

در این قسمت از تمرین می‌خواهیم یک شبکه با ساختار SOM را جهت تشخیص داده‌های ورودی MNTS و پاسخ به درخواست‌های تمرین انجام دهیم.

2-1- شعاع مجاورت نرون‌ها را صفر بگیریم :

طبق کد ضمیمه شده در ابتدا نرخ آموزش را مقدار 1 قرار می‌دهیم. پس از هر epoch نرخ آموزش را ضرب در ضریبی کوچکتر از 1 می‌کنیم تا مرحله به مرحله نرخ آموزش کوچکتر شود و با قدم‌های دقیق‌تری به پاسخ برسیم.
شرط توقف یادگیری را نیز رسیدن به حداکثر تغییر در وزن ها در یک epoch به مقدار حداقل 0.001 قرار می‌دهیم که در صورت تغییرات کمتر الگوریتم متوقف شود.

در انتهای آموزش شبکه 20 نرون برتر را به همراه تعداد داده‌های در بر گرفته و نیز تعداد داده‌های مربوط به هر کلاس را که در بر گرفته‌است خروجی می‌گیریم.

همچنین وزن‌های متناظر با هر کدام از این نرون‌های برنده را نیز به صورت یک تصویر خروجی‌می گیریم.

مشاهده می‌شود با توجه به رندوم بودن چیدمان نرون‌ها نرونی که نزدیک ترین فاصله را با توده داده‌ها دارد اکثر داده‌ها را به خود اختصاص می‌دهد. بنابراین بهتر است مقادیر اولیه نرون‌ها را به صورت پخش شده در نظر بگیریم تا این اتفاق مشکل‌زا نشود.

(به دلیل کوچک بودن عکس‌ها تمامی عکس‌ها در پوشه Pics موجود است با نام‌های مربوطه عکس:(20dar10_rand_0))

در این حالت نظم خاصی مشاهده نمی‌شود.

تعداد داده های مربوط به هر نرون برای 50 ایپاک :

در این حالت در ایپاک 29 شرط توقف برقرار می‌شود :

	Numbers
0	62
1	59
2	49
3	47
4	45
5	45
6	42
7	37
8	37
9	36
10	36
11	36
12	35
13	34
14	33
15	31
16	28
17	28
18	27
19	26

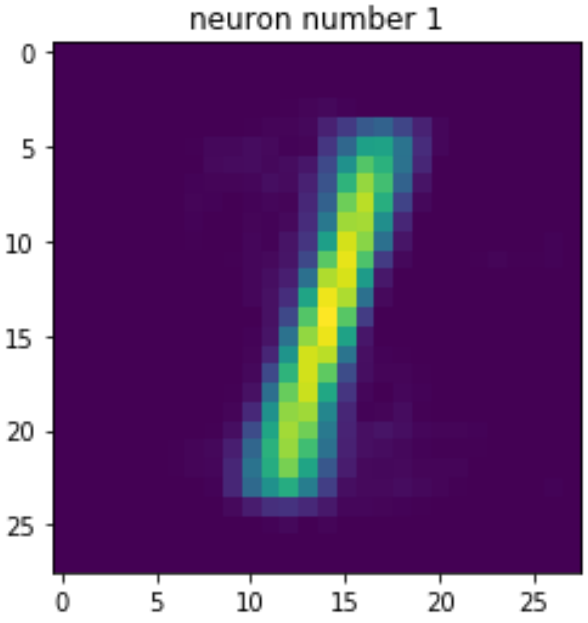
طبق انتظار داده‌ها بر روی نرون‌های بیشتری پخش شده‌اند. ولی 20 نرون اول اکثر داده‌ها را به خود اختصاص داده‌اند.

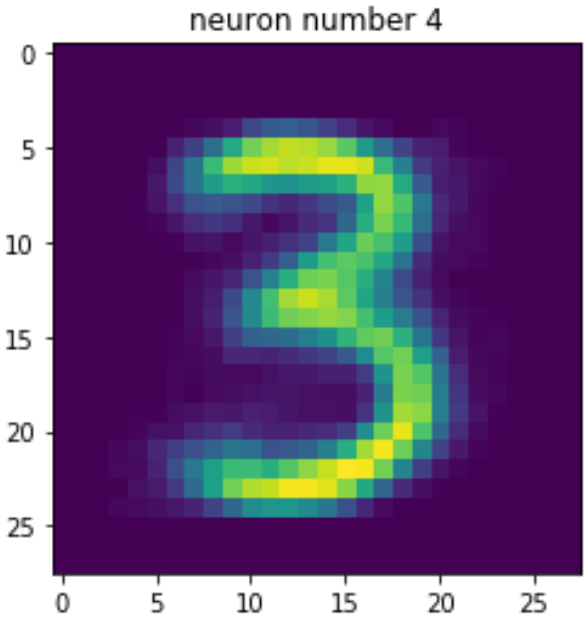
	1	2	3	4	5	6	7	8	9	10
0	6	9	7	4	5	8	6	6	8	3
1	4	8	7	3	4	6	4	10	6	7
2	4	4	7	6	7	3	5	8	3	2
3	3	6	2	9	4	2	5	2	4	10
4	5	7	5	4	6	2	3	5	3	5
5	6	5	5	1	4	5	4	5	3	7
6	2	2	5	5	5	4	7	8	1	3
7	2	4	1	3	3	3	6	7	6	2
8	2	6	5	6	4	5	2	4	1	2
9	1	3	4	4	2	3	5	5	2	7
10	3	2	6	6	6	3	2	2	6	0
11	4	5	3	2	2	3	5	5	0	7
12	2	6	3	3	5	1	6	4	3	2
13	4	3	4	2	3	5	3	4	2	4
14	1	3	3	2	4	3	3	2	9	3
15	6	6	3	2	3	1	5	2	0	3
16	1	2	6	2	5	4	1	3	2	2
17	4	5	4	3	1	3	2	4	1	1
18	3	2	3	4	5	2	0	2	2	4
19	2	2	1	3	3	7	0	2	4	2
20	3	4	0	2	1	6	0	6	1	2

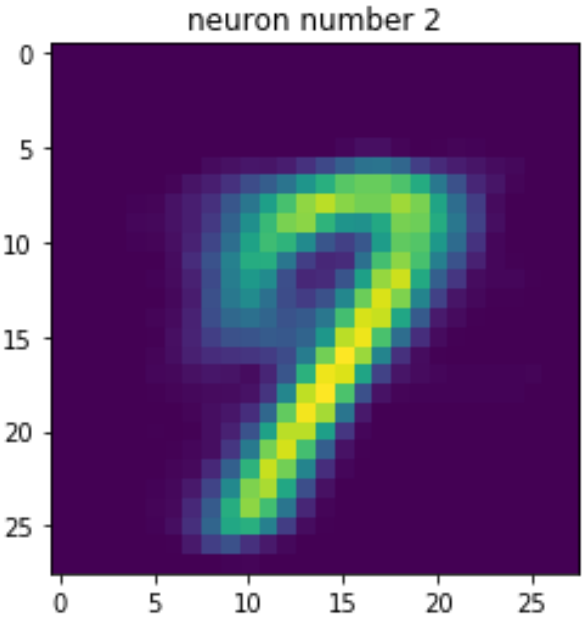
مشاهده می‌شود برخی از نرون‌ها صرفا کلاس‌های خاصی را به خود اختصاص می‌دهند.

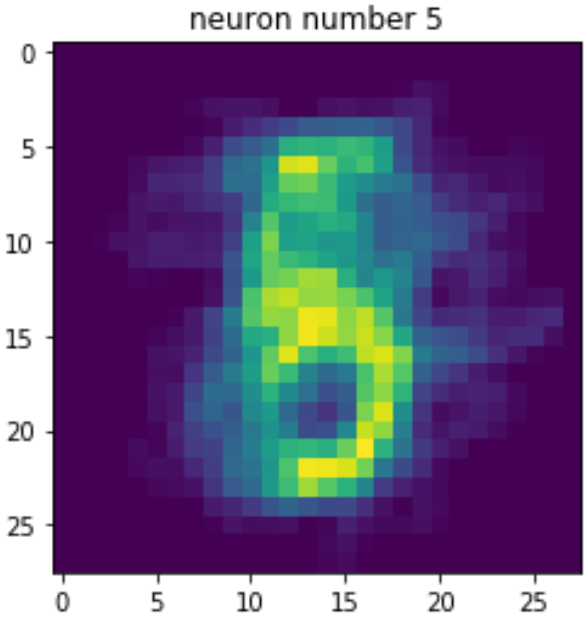
خروجی وزن 5 نرون اول در این حالت :

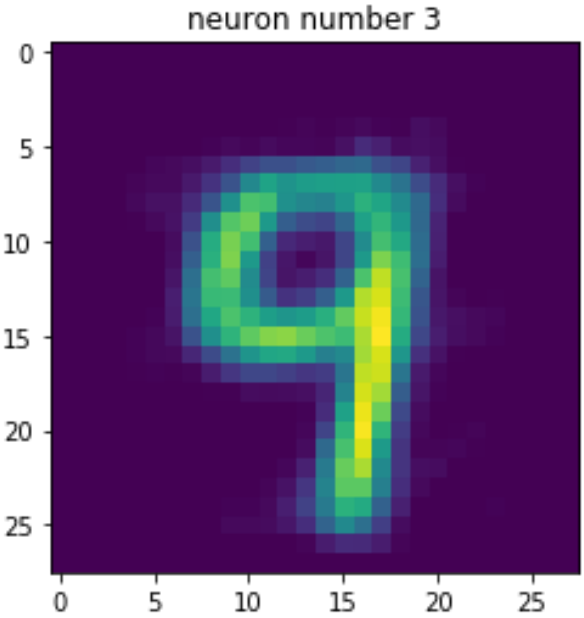
خروجی وزن‌ها نشان می‌دهد نرون به چه کلاسی نزدیک‌تر است.



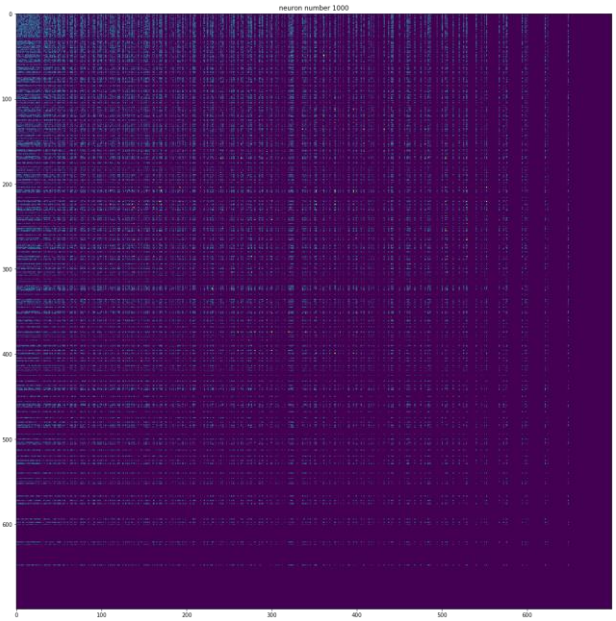




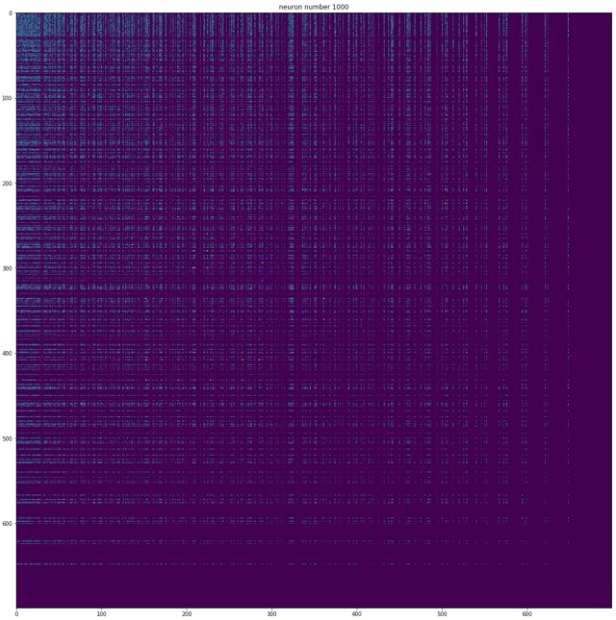




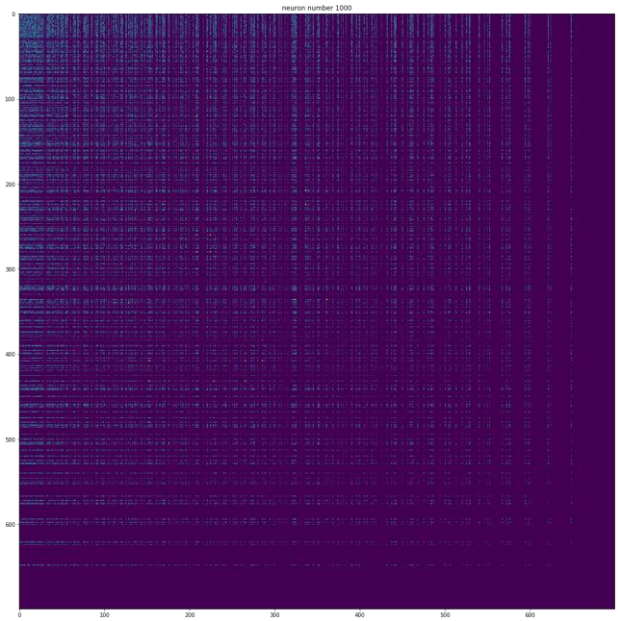
برای درک بهتر این تغییرات شکلی با ابعاد 700 در 700 رسم می‌کنیم که در بر دارنده 25 در 25 کلاستر هر یک به ابعاد 28 در 28 می‌باشد. با توجه به شکل زیر بهتر است اندکی رزولوشن عکس را بالا ببریم از این رو وزن‌های زیر 0.45 را حذف می-کنیم.



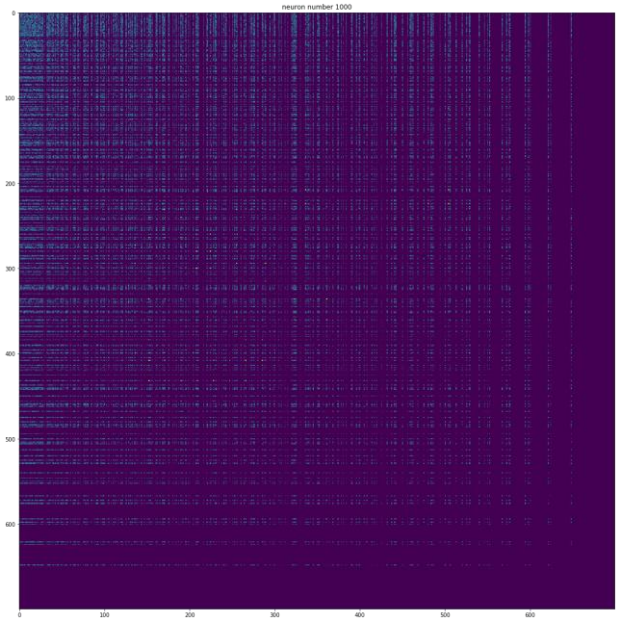
برای مثال در اینجا عکس ایپاک‌های 5و12و20و25 را بررسی می‌کنیم. مشاهده می‌کنیم به مرور زمان وزن‌های مختلف حول یک دسته مربع مانند بالا سمت چپ جمع می‌شوند. این همان کلاستری است شامل نورون‌هایی که بیشترین برندگی را دارند.



شکل برای ایپاک 5



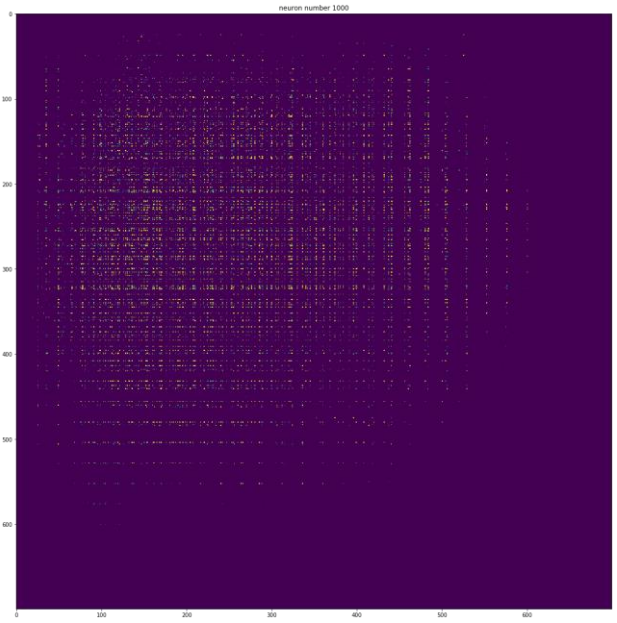
شکل برای ایپاک 20



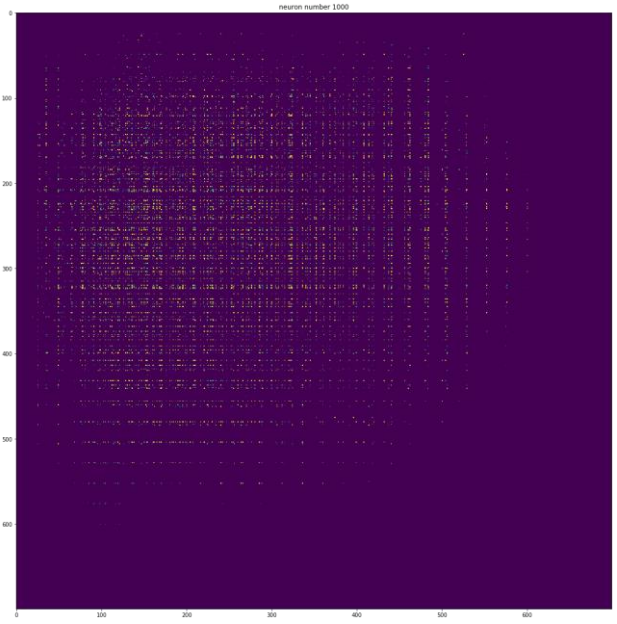
شکل برای ایپاک 25

حال ورودی‌ها را به صورت انتخاب 625 وزن از ورودی MNITS در نظر می‌گیریم. این بار تمامی مراحل بالا را طی می‌کنیم. درکمال تعجب تفاوت بسیار چشم گیری در قدرت وزن‌ها و جمع شدگی آن‌ها مشاهده می‌کنیم.

ابتدا 20 نورون برنده را در شکل زیر مشاهده می‌کنیم.



ایپاک 5



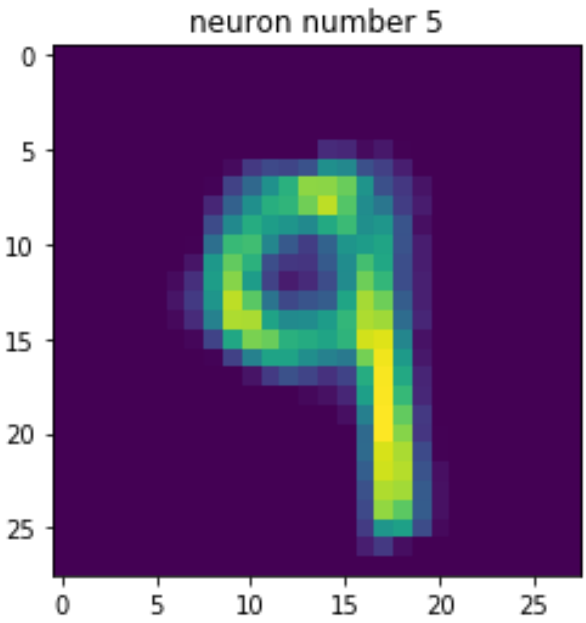
ایپاک 10

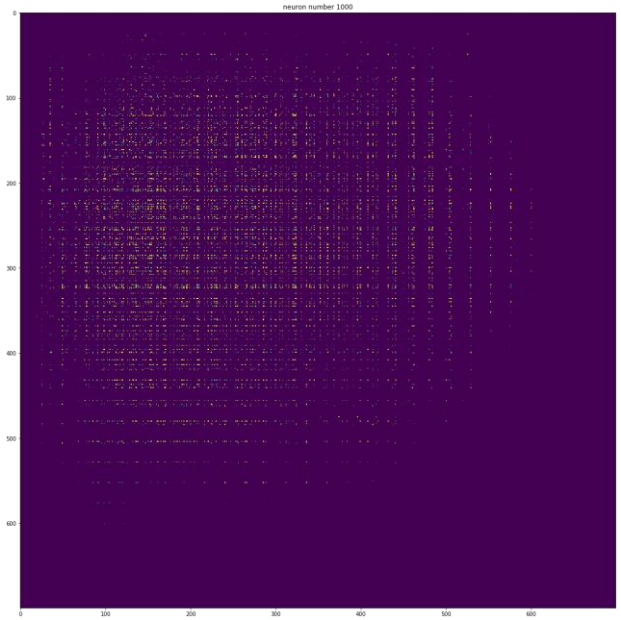
	Numbers
0	13
1	11
2	10
3	10
4	10
5	9
6	9
7	9
8	8
9	8
10	7
11	7
12	7
13	7
14	6
15	6
16	6
17	6
18	6
19	6

حال به تشکیل جدول 10 در 20 می‌پردازیم.

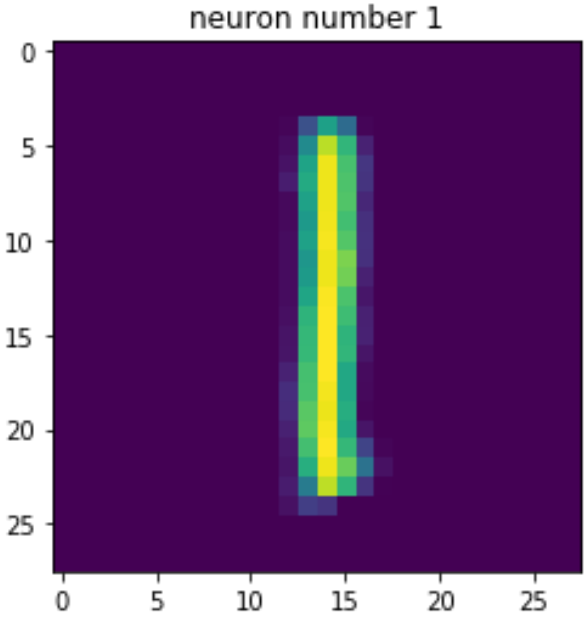
	1	2	3	4	5	6	7	8	9	10
0	1	0	2	1	1	2	1	2	2	1
1	1	1	2	1	1	0	0	2	2	1
2	2	2	2	1	1	0	0	0	0	2
3	0	2	1	1	2	2	0	0	0	2
4	0	2	2	1	1	1	0	1	1	1
5	0	0	1	1	1	1	0	1	2	2
6	1	2	2	0	0	1	0	1	0	2
7	1	0	0	0	0	0	3	0	2	3
8	0	1	0	0	3	1	3	0	0	0
9	0	0	0	1	2	2	1	0	1	1
10	0	1	0	2	0	1	0	1	2	0
11	1	2	0	1	0	0	1	1	0	1
12	0	1	1	1	2	0	1	1	0	0
13	2	0	0	0	0	1	1	1	2	0
14	1	1	1	1	0	1	0	0	0	1
15	0	0	1	2	0	0	2	0	1	0
16	1	0	0	0	2	1	0	1	0	1
17	0	0	1	1	2	0	0	1	0	1
18	0	1	0	0	1	1	1	0	0	2
19	1	2	1	0	1	0	0	1	0	0
20	0	1	0	2	1	0	0	1	0	1

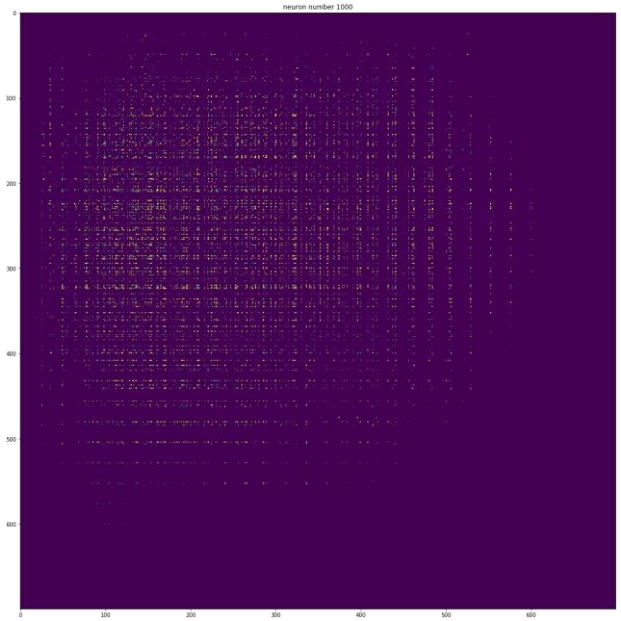
در انتها به سراغ حرکت وزن‌ها می‌رویم. پراکندگی وزن‌ها چشم گیر است زیرا خیلی از وزن‌ها شبیه به اطرافیان خود عمل کرده اند. در این حالت شرط در ایپاک 28 برقرار گردید.





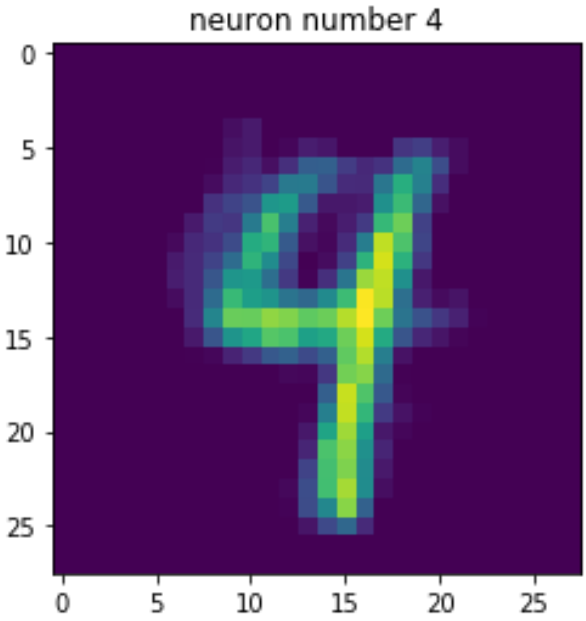
ایپاک 15

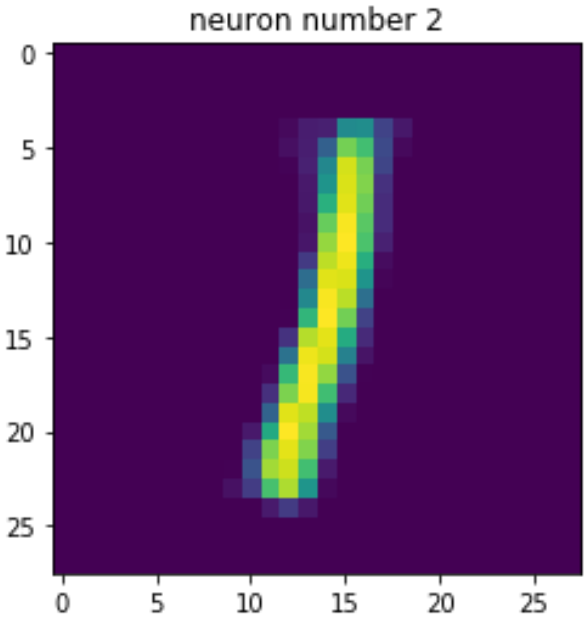


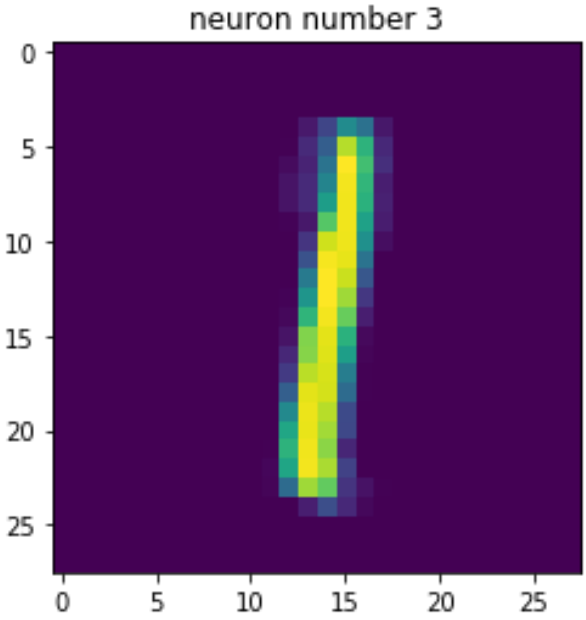


ایپاک 20

در نهایت به کشیدن 5 کلاستر برتر می‌پردازیم.







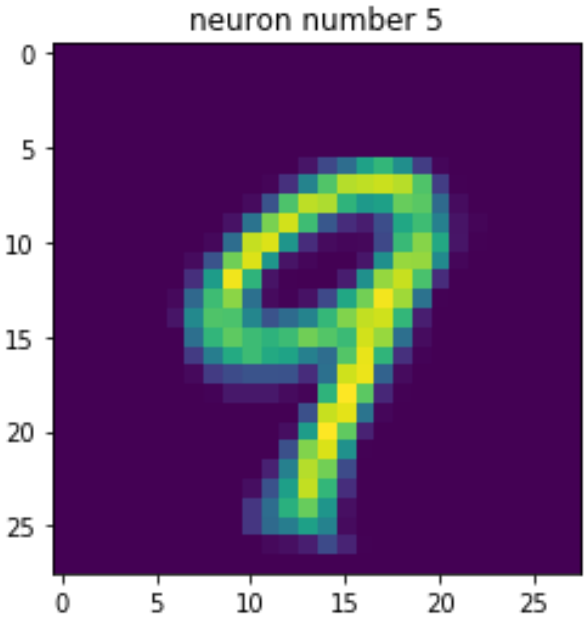
2-2- شعاع مجاورت نرون‌ها را 1 گرفته و نرون‌ها بر روی شبکه 25*25 باشند :

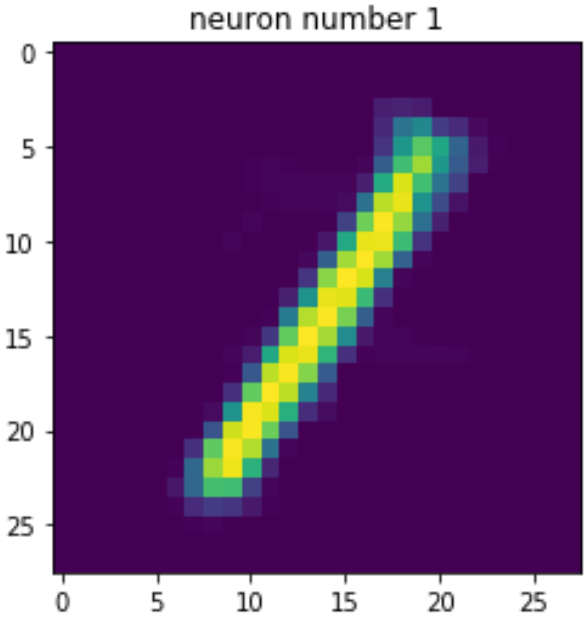
برای به روز رسانی وزن های متناظر با نرون هایی که در همسایگی نرون برنده قرار دارند نیز، یک تابع تعریف کرده ایم (تابع neighbors) که آرگومان متناظر با نرون برنده را به عنوان ورودی دریافت می کند و سپس بر اساس آرگومان متناظر با نرون برنده، آرگومان متناظر با نرون های همسایه با این نرون برنده را مشخص خواهد کرد. سپس، با در اختیار داشتن آرگومان این نرون ها، می توانیم وزن های متناظر با آن ها را به روز رسانی کنیم

البته همانند قسمت قبل عملیات به روز رسانی وزن های متناظر با نرون های مجاور با نرون برنده را تنها تا epoch دهم ادامه می دهیم و پس از آن دیگر نرون هایی که در همسایگی نرون برنده قرار دارند را به روز رسانی نخواهیم کرد

تعداد داده های مربوط به هر نرون برای 50 ایپاک :

در ایپاک 40 الگوریتم متوقف می‌شود :





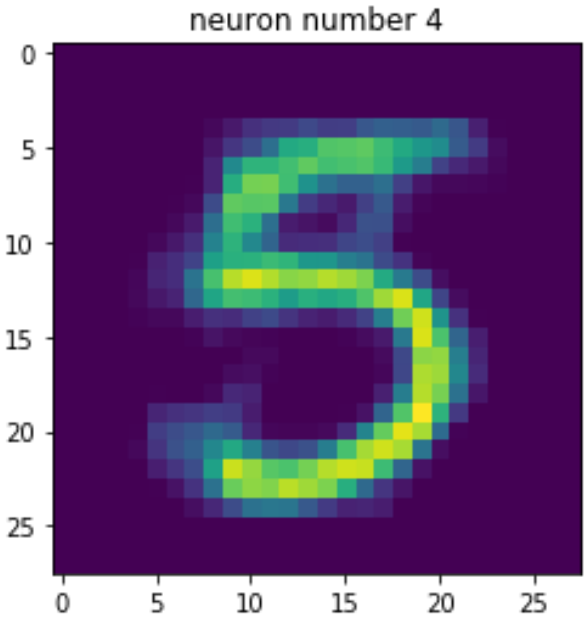
	Numbers
0	9
1	9
2	8
3	8
4	8
5	7
6	7
7	7
8	7
9	6
10	6
11	6
12	6
13	6
14	6
15	6
16	6
17	5
18	5
19	5

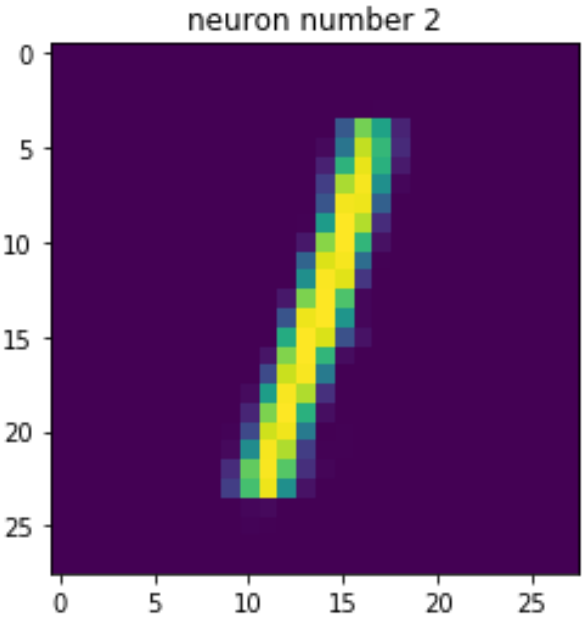
با افزایش epoch ها پراکندگی داده‌ها بر روی نرون‌ها بیشتر و بیشتر می‌شود. این حالت در بین سه روش بررسی شده در سوال بیشترین پراکندگی داده ها در بین نرون‌ها را دارد. این‌بار از نرون‌هایی که اکثر داده‌ها را به خود اختصاص دهند خبری نیست.

جدول 10 در 20 برای 20 نورون برتر:

	1	2	3	4	5	6	7	8	9	10
0	0	0	1	0	2	0	1	2	2	1
1	0	1	0	0	2	1	2	2	1	0
2	1	0	0	0	1	1	1	3	0	1
3	0	2	1	1	1	1	1	0	0	1
4	0	1	0	2	1	0	1	0	2	1
5	0	0	1	2	1	1	0	2	0	0
6	1	1	1	1	1	1	0	1	0	0
7	0	1	1	1	0	2	1	1	0	0
8	1	0	1	1	2	0	0	1	0	1
9	0	1	0	2	1	0	0	0	1	1
10	0	3	0	0	0	0	1	1	1	0
11	1	2	0	1	0	0	1	1	0	0
12	0	2	0	1	0	0	1	0	0	2
13	1	0	1	0	2	0	0	2	0	0
14	1	1	1	0	0	2	0	0	1	0
15	0	0	2	1	0	0	1	2	0	0
16	1	1	1	0	0	0	1	1	0	1
17	0	0	2	0	0	1	1	0	0	1
18	1	0	0	0	1	0	0	1	1	1
19	1	0	0	0	1	0	0	1	0	2
20	1	0	0	1	2	1	0	0	0	0

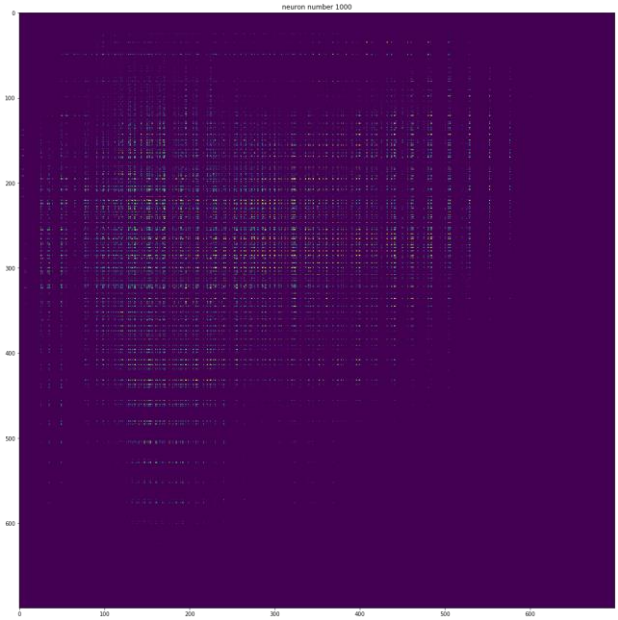
خروجی وزن 5 نرون اول در این حالت :

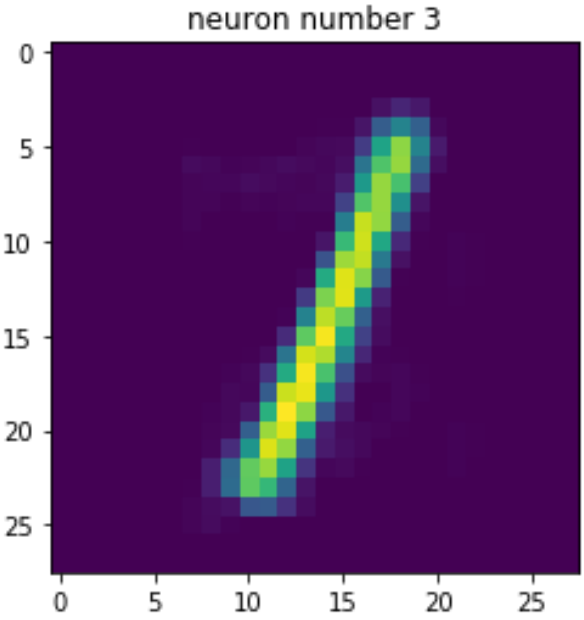


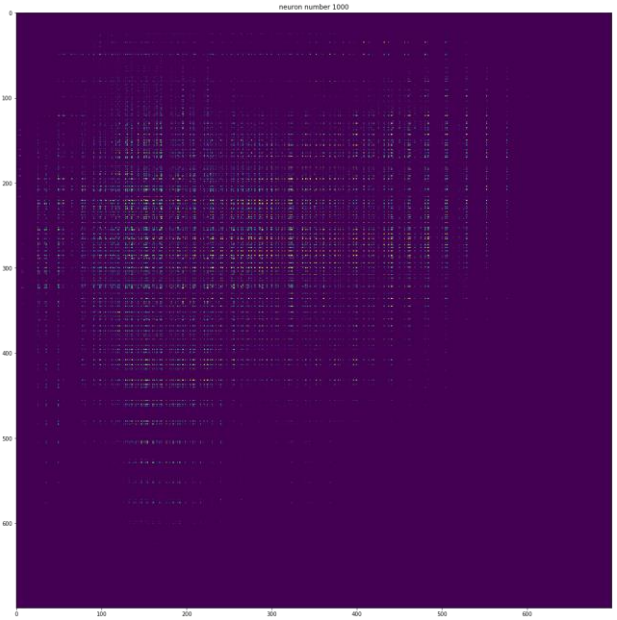


خروجی وزن‌ها نشان می‌دهد نرون به چه کلاسی نزدیک‌تر است.

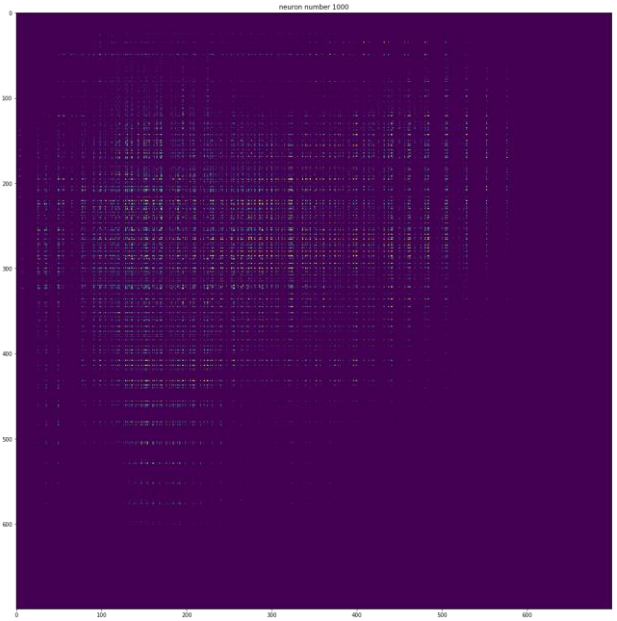
حال به تشکیل شکل 700 در 700 می‌پردازیم.



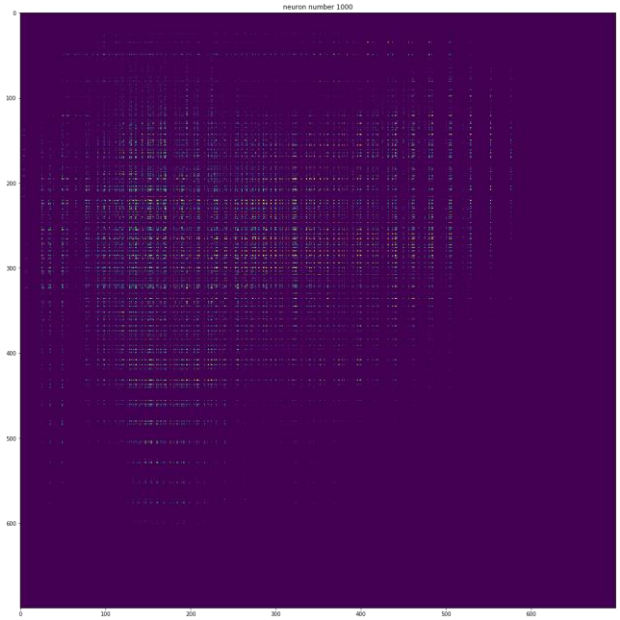




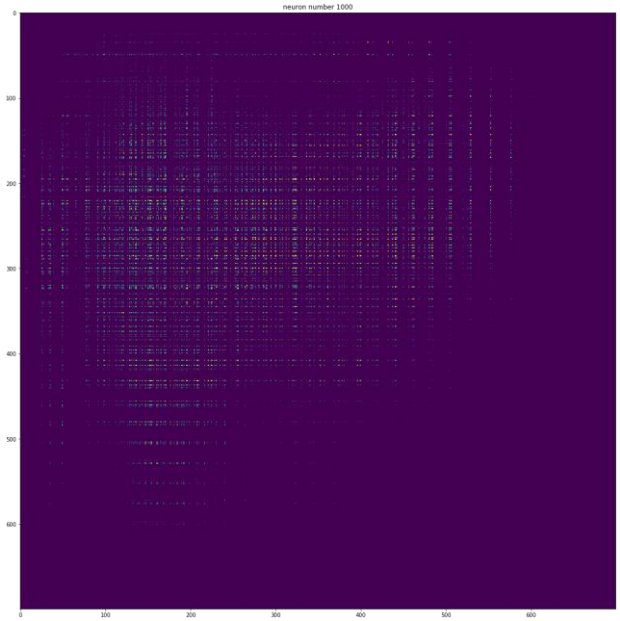
ایپاک 5



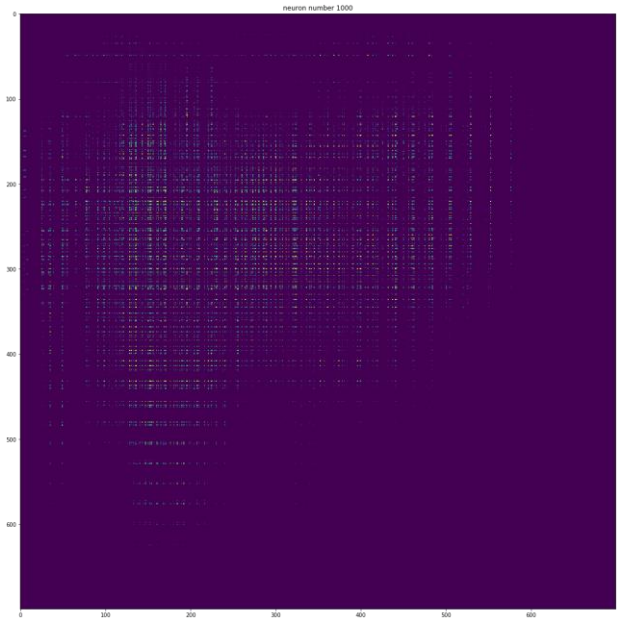
ایپاک 20



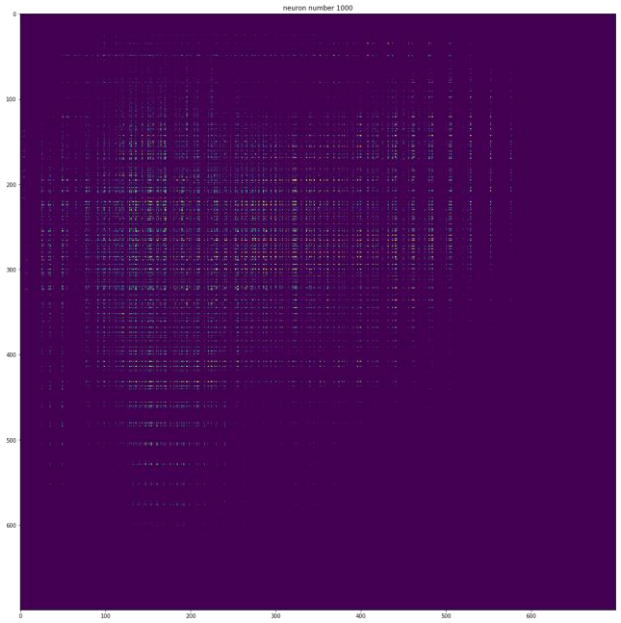
ایپاک 15



ایپاک 25



ایپاک30



ایپاک 35

در ادامه وزن‌های ورودی را از ورودی‌های Mnits به طور رندوم انتخاب می‌کنیم و تمامی فاکتورهای بالا را انجام می‌دهیم و محاسبه و نشان می‌دهیم. در این حالت شرط در ایپاک40 ارضا می‌شود.

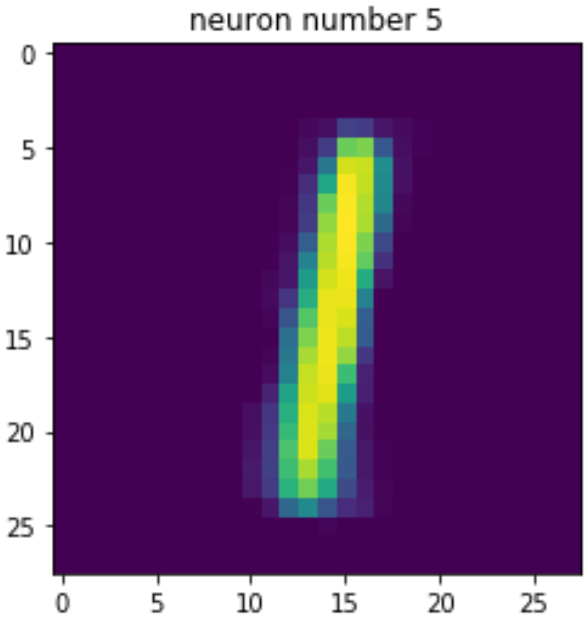
20 نورون برتر:

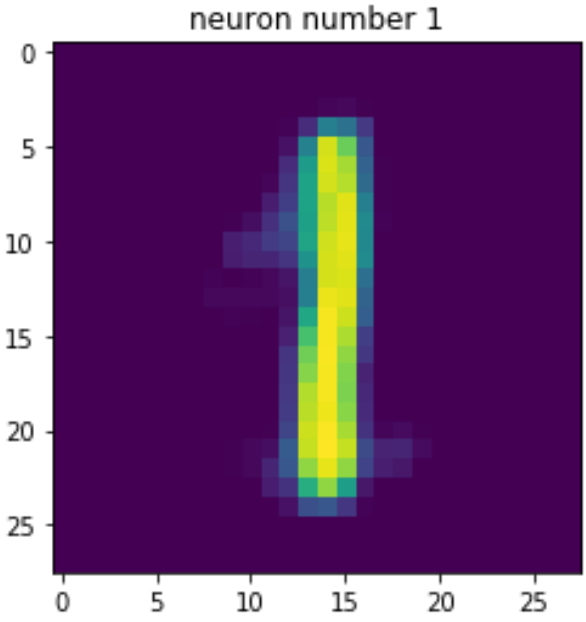
	Numbers
0	9
1	9
2	7
3	7
4	6
5	6
6	6
7	6
8	6
9	6
10	6
11	6
12	6
13	6
14	6
15	5
16	5
17	5
18	5
19	5

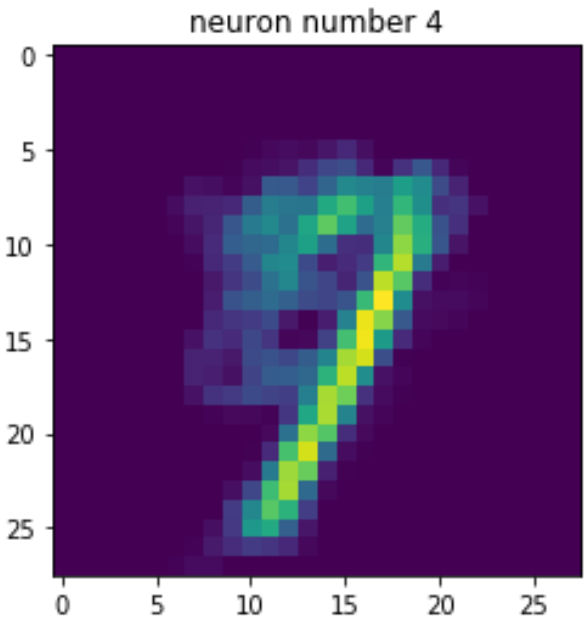
حدول 20 در 10:

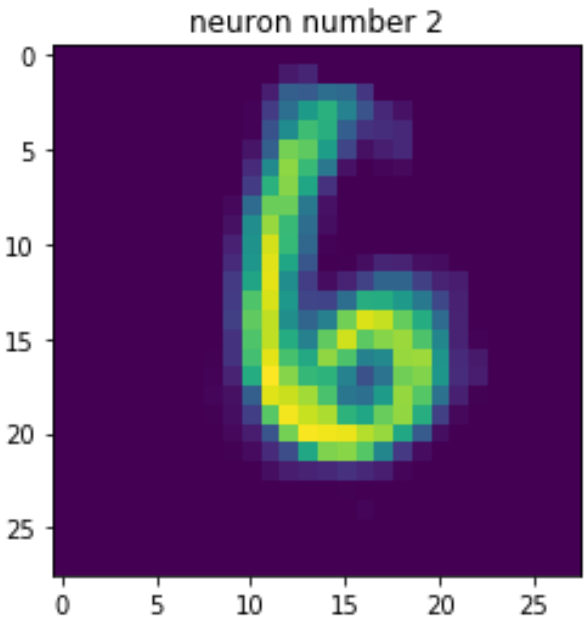
	1	2	3	4	5	6	7	8	9	10
0	0	1	2	0	1	1	1	2	0	1
1	1	0	2	1	2	1	1	0	0	1
2	1	1	1	0	0	1	1	1	1	0
3	0	0	1	1	1	0	0	2	1	1
4	0	2	0	0	1	0	1	0	0	2
5	0	0	0	1	0	1	0	3	0	1
6	1	1	1	0	1	1	1	0	0	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	1	0	0	0	1	1	1	1
9	0	1	0	3	0	0	0	0	1	1
10	1	1	0	1	0	0	1	1	1	0
11	2	0	0	1	1	0	1	1	0	0
12	0	2	0	1	0	0	1	0	0	2
13	1	1	0	0	1	0	1	1	1	0
14	1	0	1	2	0	2	0	0	0	0
15	1	0	0	0	0	2	0	0	1	1
16	1	1	1	1	0	0	0	1	0	0
17	0	1	0	0	1	0	0	1	1	1
18	1	0	1	0	0	1	0	0	0	2
19	0	0	2	0	0	0	0	1	1	1
20	0	0	0	0	0	2	0	2	0	1

5 نورون انتخابی:

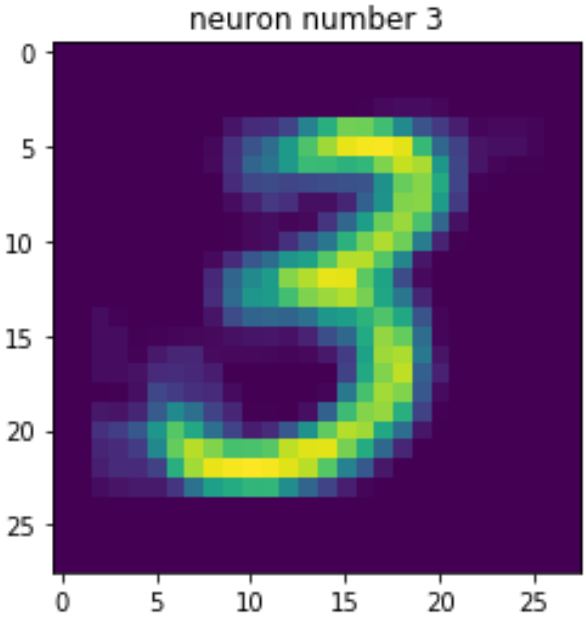

neuron number 5
0
5
10
15
20
25
0
5
10
15
20
25

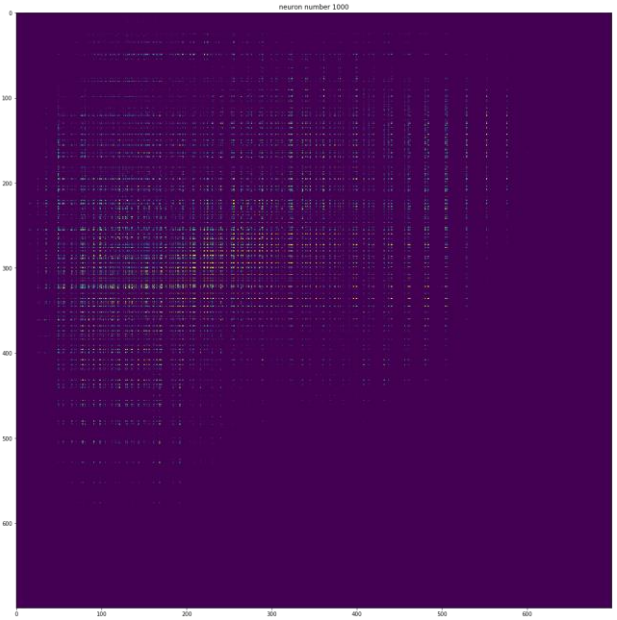

neuron number 1
0
5
10
15
20
25
0
5
10
15
20
25

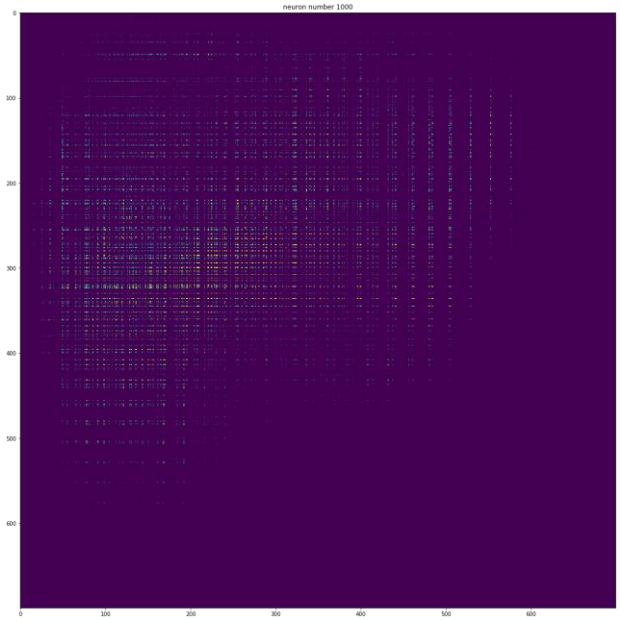


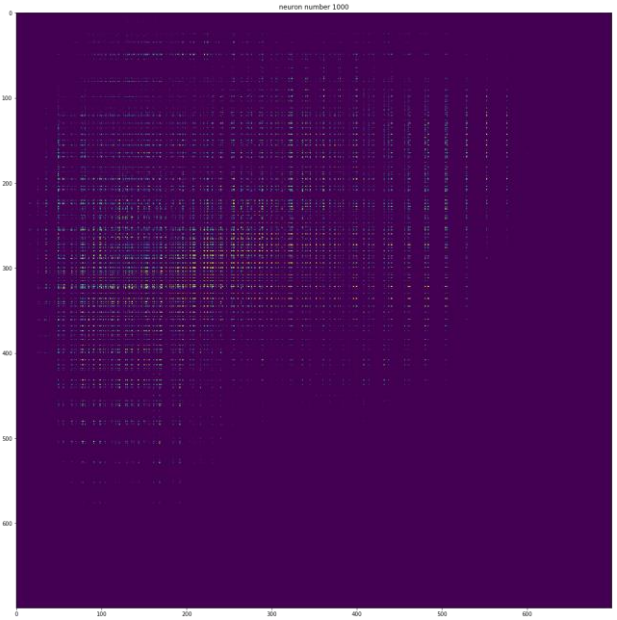


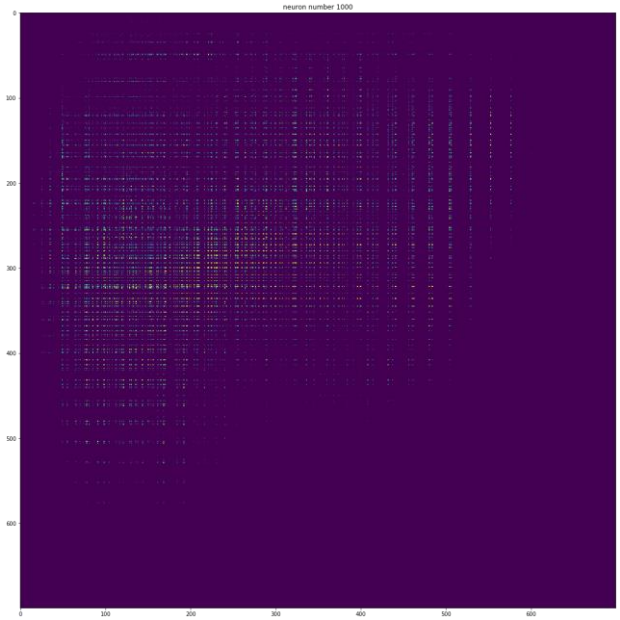
اشکال 700 در 700:(به ترتیب از ایپاک 5 تا 40)

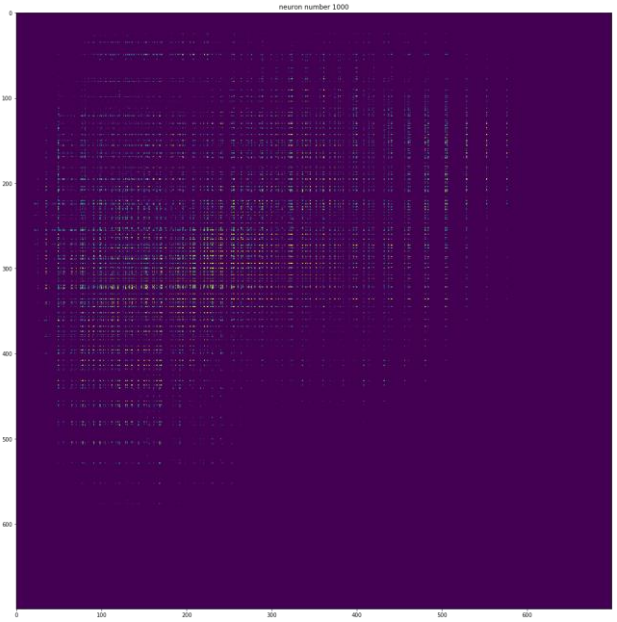


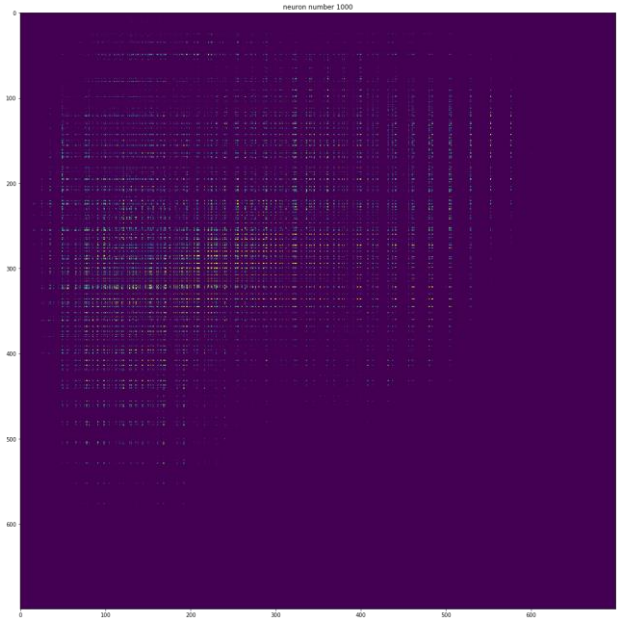

neuron number 1000

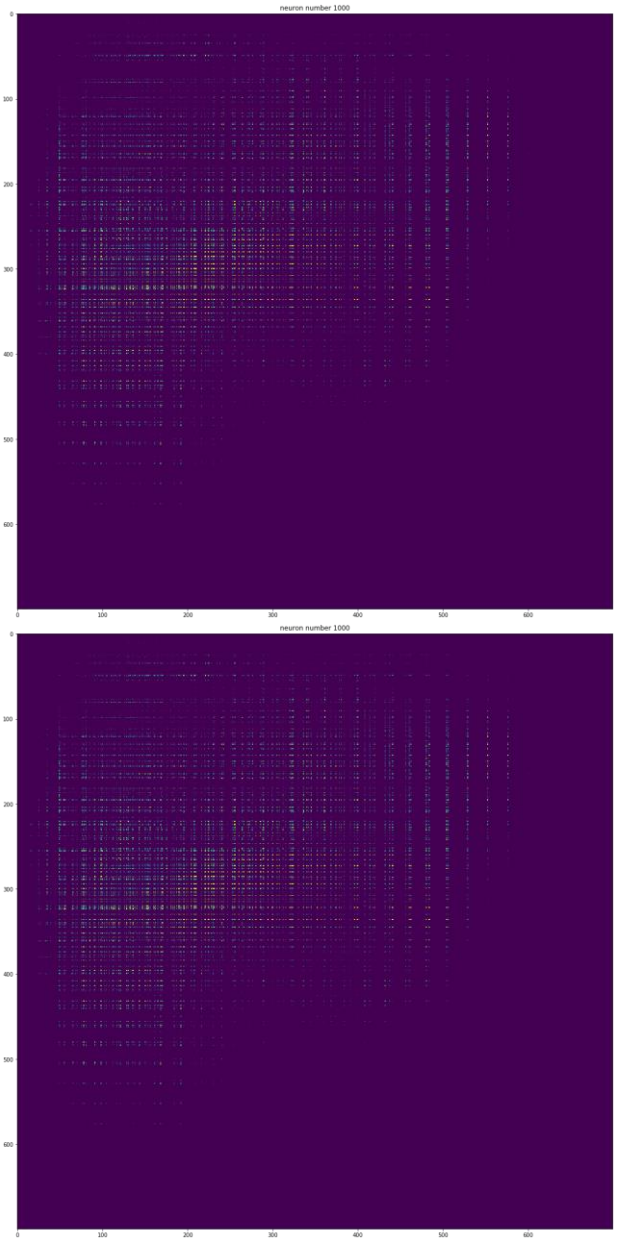

neuron number 1000


neuron number 1000


neuron number 1000


neuron number 1000


neuron number 1000



2-3- قسمت4: مقایسه چهار روش ارایه شده

از لحاظ maximum تعداد داده اختصاص یافته به هر نرون در سه روش مختلف به جدول زیر می‌رسیم:

25*25 network R=1 Deterministic	25*25 network R=1 Random	R=0 Deterministic	R=0 Random
7	6	9	10

مشاهده می‌شود تعداد داده اختصاص یافته به هر نرون در حالت r=0 بیشتر از همه و در حالت شبکه کمتر از همه می‌باشد. در حالت شبکه داده‌ها بر روی تعداد بیشتری از نرون‌ها پخش شده‌اند.

دلیل این امر این است که در صورتی که شعاع مجاورت را برابر با صفر قرار دهیم، در هر بار به روز رسانی، تنها وزن های متناظر با یکی از نرون ها که به عنوان نرون برنده انتخاب می شود، update خواهد شد. همین امر باعث می شود تا به هنگام یادگیری در epoch اول، وزن های متناظر با نرون هایی که به عنوان نرون های برنده انتخاب شده اند، به الگوی داده های اختصاص یافته به آن ها بسیار نزدیک شود، در حالی که وزن های متناظر با سایر نرون هایی که برنده نشده اند دست نخورده باقی می ماند. همین امر باعث می شود تا در epoch های بعدی در فرآیند یادگیری، همان نرون هایی که در epoch اول به عنوان نرون های برنده انتخاب شده اند، باز هم برنده شوند (چون وزن های متناظر با این نرون ها نسبت به سایر نرون هایی که وزن هایشان به روز رسانی نشده است، به الگوی متناظر با داده های مورد بررسی نزدیک تر است). این روند تا زمان اتمام فرایند یادگیری ادامه خواهد یافت و در نهایت پس از اتمام یادگیری، تعدادی نرون داریم که وزن های متناظر با آن ها به الگوی داده های مورد بررسی بسیار نزدیک است، اما بیشتر نرون های شبکه، نرون هایی هستند که در فرایند یادگیری، هیچگاه به عنوان نرون برنده انتخاب نشده اند و در نتیجه وزن متناظر با آن ها، همان وزن های تصادفی است که در ابتدا به آن ها assign کرده ایم. در نتیجه وزن های متناظر با این نرون ها از الگوهای متناظر با داده های مورد بررسی فاصله زیادی خواهند داشت. بنابراین نرون‌هایی هستند که در طی مراحل با update شدن داده‌های زیادی را به خود اختصاص خواهند داد.

در طرف مقابل، هنگامی که نرون ها را به فرم خطی و با شعاع مجاورت R=1 قرار می دهیم، در هر بار به روز رسانی وزن های متناظر با نرون ها، علاوه بر نرونی که به عنوان نرون برنده انتخاب می شود، وزن های متناظر با نرون هایی که در مجاورت با این نرون قرار دارند نیز update خواهند شد. این امر باعث می شود تا در صورتی که یکی از نرون ها به عنوان نرون برنده انتخاب شود، علاوه بر وزن های متناظر با خود نرون برنده، وزن های متناظر با نرون های مجاور نیز به الگوی داده‌ی اختصاص داده شده به نرون برنده نزدیک شود. در نتیجه، در epoch های بعدی در فرایند یادگیری، هر کدام از این نرون ها امکان آن که به عنوان نرون برنده انتخاب شوند را خواهند داشت. همین امر باعث می شود تا پس از اتمام فرایند یادگیری، تعداد نرون هایی که وزن های متناظر با آن ها به روز رسانی شده اند (و به الگوهای موجود در داده های مورد بررسی نزدیک شده اند)، بیشتر از حالتی باشد که شعاع مجاورت برابر با صفر بود. در

نتیجه، به هنگام خوشه بندی نهایی، تمامی داده های متناظر با یک الگو، به یک نرون اختصاص داده نمی شود و به تعدادی از نرون ها که وزن های متناظر با آن ها به الگوی مورد نظر نزدیک است، اختصاص داده خواهند شد. بنابرین، تعداد داده هایی که به هر کدام از نرون های برنده اختصاص می یابد، در مقایسه با حالت قبلی کمتر خواهد بود.

همین استدلال در مورد حالتی که نرون ها را روی نود های یک شبکه 25×25 قرار می دهیم و با فرم مجاورت مربعی با R=1، الگوریتم را پیاده سازی و اجرا می کنیم نیز برقرار است. البته در این حالت، تعداد نرون هایی که در همسایگی با نرون برنده قرار می گیرند و به واسطه این همسایگی، وزن های متناظر با آن ها به روز رسانی می شود، نسبت به حالت قبل بیشتر است. در نتیجه، در مقایسه با حالت قبل، تعداد نرون هایی که وزن های متناظر با آن ها پس از اتمام فرایند یادگیری، به الگو های موجود در داده های مورد بررسی نزدیک است، نسبت به حالت قبل بیشتر خواهد بود و در نتیجه انتظار خواهیم داشت تا در این حالت، تعداد داده های اختصاص یافته به هر کدام از نرون های برنده نسبت به حالت قبلی نیز، کمتر باشد.

از لحاظ مجموع داده‌های اختصاص یافته به 20 نرون اول :

25*25 network R=1 Deterministic	25*25 network R=1 Random	R=0 Deterministic	R=0 Random
9	9	13	62

مشاهده می‌شود با افزایش همکاری‌ها تعداد داده‌های اختصاص یافته به نرون‌های برتر کم می شود و به سمتی پیش می‌رویم که تعداد محدودی نرون غالب وجود نداشته باشد.

از لحاظ epoch تا رسیدن به شرط توقف :

25*25 network R=1 Deterministic	25*25 network R=1 Random	R=0 Deterministic	R=0 Random
40	39	28	29

مشاهده می‌شود با افزایش همکاری‌ها در سطح شبکه تعداد epoch ها به صورت ملموسی کاهش داشته است.

اگر بخواهیم به صورت کلی نتیجه گیری کنیم، می توانیم بگوییم در صورتی که آموزش شبکه عصبی به صورتی باشد که در به روز رسانی وزن های متناظر با نرون ها شعاع مجاورت را در نظر نگیریم، خوشه بندی نهایی شامل تعداد کمی خوشه با سایز بزرگ خواهد شد. اما در صورتی که آموزش شبکه عصبی به صورتی باشد که در به روز رسانی وزن های متناظر با نرون ها شعاع مجاورت را در نظر بگیریم، خوشه بندی نهایی شامل تعداد زیادی خوشه با سایز کوچک خواهد شد. هر چقدر شعاع مجاوت در نظر گرفته شده بزرگ تر باشد (یا به عبارت دیگر تعداد نرون هایی که به عنوان همسایه نرون برنده محسوب می شوند بیشتر باشد)، خوشه بندی نهایی شامل تعداد بیشتری خوشه با سایز کوچکتر خواهد بود.

3- پیاده‌سازی یک شبکه RNN جهت تشخیص یک بیماری بر اساس انتخاب 6 معیار از 11 معیار

در این قسمت می‌خواهیم یک بیماری را توسط یک شبکه رقابتی بر اساس انتخاب 6 نشانه از 11 نشانه پیاده‌سازی کنیم.

اینگونه به نظر می‌رسد که با توجه به نوع ورودی‌ها که شامل یک آرایه 11 بیتی است و عملیات مقایسه انجام می‌شود، شبکه Hamming Net می‌تواند گذینه مناسبی برای این تمرین باشد. برای معماری این شبکه با توجه به 11 بیتی بودن ورودی-ها در ابتدا یک لایه ورودی شامل 11 نورون Input داریم. سپس به دلیل دو مرجع بیمار و سالم ، یک لایه میانی شامل دو نورون داریم که این لایه لایه fully connected نیست و در انتها یک ساختار Maxnet برای ایجاد رقابت قرار می‌دهیم. شکل شبکه پیشنهادی ما همانند شکل زیر ولی با تفاوت لایه اول 11 تایی می‌باشد.

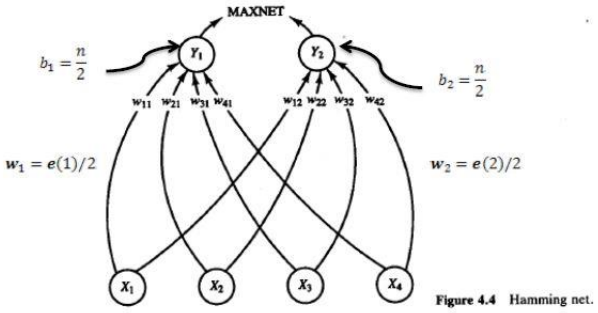


Figure 4.4 Hamming net.

نحوه فرموله کردن نمونه‌های خود را به صورتی که هر کدام یک آرایه 11 تایی هستند در نظر می‌گیریم که با توجه به فاصله همینگ 6 تایی مورد نظر سوال این دو باید حداقل در 6 بیت با یکدیگر تفاوت داشته باشند برای مثال دو نمونه زیر:

مریض : [1,1,-1,1,-1,1,1,1,-1,-1,1]

سالم : [-1,-1,1,1,-1,-1,-1,1,1,-1,-1]

سپس به تشکیل ماتریس وزن‌های این دو می‌رویم. این ماتری شامل 22 وزن است که از روابط موجود در شکل بالا به صورت زیر محاسبه می‌شود.

$$\begin{bmatrix} 0.5 & -0.5 \\ 0.5 & -0.5 \\ -0.5 & 0.5 \\ 0.5 & 0.5 \\ -0.5 & -0.5 \\ 0.5 & -0.5 \\ 0.5 & -0.5 \\ 0.5 & 0.5 \\ -0.5 & 0.5 \\ -0.5 & -0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

و b1=b2=2

پس از نوشتن کد این برنامه در python به صـورت تصـادفی 5 داده را به سیستم می‌دهیم. این داده‌ها عبارتند از :

person1 = [1,-1, 1, 1, 1,-1,-1, 1,-1,-1, 1]
person2 = [1,-1, 1, 1, 1,-1,-1, 1,-1,-1, 1]
person3 = [-1,-1, 1,-1, 1,-1,-1,-1,-1,-1, 1]
person4 = [1, 1, 1,-1, 1,-1, 1, 1,-1, 1, 1]
person5 = [1,-1, 1, 1,-1, 1,-1, 1, 1,-1, 1]

با توجه به فاصله همینگ این داده‌ها با دو معیار سالم و مریض تنها شخص 4 مریض تشخیص داده می‌شود و باقی سالمند. فاصله همینگ داده‌های بالا با دو مرجع ما در جدول زیر موجود است.

فاصله همینگ با معیار سالم	فاصله همینگ با معیار مریض
5	4
5	4
8	5
5	8
4	3

مراجع

[1] http://hadifar.net