دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

گزارش تمرین سری سوم درس شبکه عصبی

یلدا فروتن

۸۱۰۱۹۶۲۶۵

استاد درس:

جناب آقای دکتر کلهر

پاییز ۹۸

۱. پیاده‌سازی الگوریتم Self-Organizing Map

۱,۱ مقدمه‌ای بر شبکه SOM

یکی از شبکه‌های برپایه مکانیزم شبکه تک‌لایه SOM است که وزن‌های آن ثابت نیست. درواقع SOM اجازه می‌دهد که در یک فضای n بعدی، بردارها به m بردار cluster بشوند. الگوریتم SOM شبیه به K-means است چون اجازه می‌دهد تعداد زیادی بردار هم‌بعد، دسته‌بندی شوند؛ به‌گونه‌ای‌که نمونه‌هایی که به یکدیگر نزدیک‌تر هستند در یک کلاستر قرار بگیرند. درواقع شبکه SOM دارای m نورون رقابت‌کننده است و هر نورون می‌خواهد در فضای بین نورون‌ها رسوخ کرده و بیش‌ترین سمپل‌ها را از آن خود کند تا هر نورون مرکز یک دسته شود. از طرفی نورون‌های همسایه در ابتدا ممکن است با یکدیگر همکاری بکنند تا مرکز دسته‌ها را فتح کنند. البته شعاع همسایگی می‌تواند صفر باشد و نورون‌ها از همان ابتدا فقط با یکدیگر رقابت کنند. میزان شعاع همسایگی درواقع به تعداد نورون‌ها بستگی دارد، به گونه‌ای‌که اگر تعداد نورون‌ها کم باشد، از همان ابتدا شعاع همسایگی صفر خواهد بود. لازم به ذکر است همکاری بین نورون‌ها، برخلاف الگوریتم Mexican Hat، کاهش می‌یابد. از طرفی چینش نورون‌ها نیز حائز اهمیت است؛ نورون‌ها می‌توانند بر روی یک خط باشند یا چینش دوبعدی و ... داشته باشند. بنابراین همسایگی نورون‌ها با چنیش آن‌ها رابطه مستقیم خواهد داشت. در ادامه یک شبکه SOM برای داده‌های MNIST طراحی می‌شود.

۲,۱ طراحی یک شبکه SOM

هدف از این بخش کلاستر کردن ۱۰۰۰ نمونه از داده‌های دیتاست MNIST است.

۱,۲,۱ فراخوانی داده‌های MNIST

دیتاست MNIST شامل هفتادهزار تصویر از اعداد دست‌نویس ۰ تا ۹ بوده و هر تصویر به صورت یک ماتریس ۲۸*۲۸ است. در ابتدا دیتاست توسط فریم‌ورک keras فراخوانی شده و به منظور نرمالایز کردن بر ۲۵۵ تقسیم شده‌اند. برای انتخاب ۱۰۰۰ داده به صورت رندوم، داده‌ها shuffle شده و ۱۰۰۰ تای ابتدایی آنها برداشته شده است. بنابراین سمپل‌ها به‌صورت یک ماتریس ۷۸۴*۱۰۰۰ خواهند بود و بدیهی است که ۷۸۴ همان تعداد پیکسل‌ها یعنی ۲۸*۲۸ می‌باشد.

۲,۲,۱ پیاده‌سازی الگوریتم

در این بخش الگوریتم یادگیری SOM طراحی می‌شود. پیش‌تر گفته شد که وزن‌های این شبکه غیرثابت هستند. درواقع می‌توان یک الگوریتم برای یادگیری شبکه پیاده کرد که در راستای کلاستر کردن دیتاست باشد. ازآنجایی که شبکه تک‌لایه است ماتریس وزن‌ها بین لایه ورودی و خروجی تعریف می‌شود. لایه ورودی به تعداد ویژگی‌ها نورون داشته و شبکه به تعداد کلاسترها نورون خروجی خواهد داشت. بنابراین در این مسئله ماتریس وزن دارای ابعاد ۶۲۵*۷۸۴ خواهد بود. برای مقداردهی اولیه به ماتریس وزن از در روش استفاده شده است:

- مقداردهی ماتریس وزن به صورت رندوم
- مقداردهی ماتریس وزن با استفاده از مقادیر سمپل‌های ورودی

از طرفی دو چینش متفاوت برای نورون‌ها درنظر گرفته شده است:

- نورون‌ها بر روی یک خط باشند و بدون همسایگی
- نورون‌ها در یک فضای ۲۵*۲۵ بوده و با شعاع یک نورون همسایگی داشته باشند

برای ۴ حالت فوق، ۴ نمونه اجرا بررسی شده است که در ادامه هر کدام بررسی خواهد شد. این حالت‌ها در ادامه آمده است.

حالت اول: ماتریس وزن به صورت رندوم مقداردهی اولیه شوند و بر روی یک خط باشند.

حالت دوم: ماتریس وزن با استفاده از سمپل‌ها مقداردهی شوند و بر روی یک خط باشند.

حالت سوم: ماتریس وزن به صورت رندوم مقداردهی اولیه شوند و با همسایگی در فضای دوبعدی ۲۵*۲۵ باشند.

حالت چهارم: ماتریس وزن با استفاده از سمپل‌ها مقداردهی شوند و با همسایگی در فضای دوبعدی ۲۵*۲۵ باشند.

در الگوریتم SOM نرخ یادگیری ابتدا بزرگ فرض می‌شود تا نورون‌ها قدرت جابه‌جایی داشته باشند؛ زیرا در ابتدا نورون‌ها در جاهای پرت می‌افتند. در طی iteration، نرخ یادگیری کاهش می‌یابد تا گرادیان کاهشی بتواند نورون‌ها را به نقاط پرچگال‌تر همگرا کند. درابتدا نرخ یادگیری ۰,۶ انتخاب شده است. کاهش نرخ یادگیری نیز به صورت هندسی بوده و هربار ۰,۹ برابر می‌گردد.

در شبکه SOM، سمپل‌های ورودی در ماتریس وزن ضرب نمی‌شوند بلکه فاصبه اقلیدسی آن‌ها با ماتریس وزن محاسبه می‌گردد. حال نورونی برنده خواهد بود که فاصله اقلیدسی آن کمترین مقدار را داشته باشد. بدین منظور یک تابع نوشته شده است که مجموع فواصل هر ورودی و با هر ستون ترانهاده ماتریس وزن را حساب می‌کند. درنهایت ایندکس ستونی از ترانهاده ماتریس که برنده شد (کمترین فاصله را از ورودی داشت) به عنوان خروجی تابع داده می‌شود. درواقع این ستون از ترانهاده ماتریس وزن (و همسایگی‌های آن) آپدیت خواهند شد.

حالت اول: مقداردهی اولیه رندوم و چینش خطی نورون‌ها

در این بخش ماتریس وزن به صورت رندوم مقداردهی شده است و تقسیم بر ۶ شده است. کوچک بودن مقادیر ماتریس وزن منجر می‌شود شانس همگرایی افزایش یابد. درادامه راجع‌به این موضوع صحبت خواهد شد. سپس به ازای ۱۰۰۰ نمونه فاصله هر نمونه با ستون‌های ترانهاده ماتریس وزن محاسبه گردید و ایندکس کمترین فاصله ذخیره شد. حال ایندکس ذخیره‌شده برای هر نمونه منجر به آپدیت شدن ستون مربوطه ماتریس وزن گردید. در

ایپاک آخر نیز لیبل‌های هر نمونه ذخیره گردید. در انتها نیز نرخ یادگیری ۰,۹ گردید. درنهایت الگوریتم برای ۲۰ ایپاک پیاده شد و runtime ۱۲۹ ثانیه طول کشید.

تا اینجا ماتریس وزن آپدیت شده است. حال لازم است تعداد سمپل‌هایی که هر نمونه به خود گرفته به همراه کلاس آن‌ها مشخص شود. بدین‌صورت یک ماتریس ۱۰*۶۲۵ ساخته شده که هر سطر آن نشان‌دهنده نورون‌ها و ستون‌های آن نشان‌دهنده کلاس‌های هر سمپل است. پس برای هر نمونه چک می‌شود که مربوط به کدام کلاس بوده و در ماتریس ساخته شده ذخیره می‌گردد. سپس یک ستون به ماتریس افزوده می‌شود که مجموع تمام لیبل‌هایی که هر نورون گرفته را محاسبه می‌کند. از طرفی یک ستون دیگر برای ایندکس‌های نورون‌ها به ماتریس ساخته شده افزوده می‌شود و در نهایت ماتریس به صورت ۱۲*۶۲۵ می‌شود که براساس ستون مجموع لیبل‌های گرفته شده سورت می‌گردد و ۲۰ نورون برنده که بیش‌ترین سمپل را به خود اختصاص داده‌اند انتخاب می‌شوند. درنهایت با استفاده از کتابخانه Pandas جدول نورون‌های برنده ساخته شده است. لازم به ذکر است منظور از cluster نورون بوده و لیبل‌ها، کلاس‌هایی را که به خود اختصاص داده‌اند نشان می‌دهد. همچنین ستون sum نشان‌دهنده تعداد نمونه‌هایی است که هر کلاس به خود اختصاص داده است. به‌عنوان مثال نورون ۱۸۲، ۳۲ نمونه از کلاس ۱ و دو نمونه از کلاس ۷ را به خود اختصاص داده که در مجموع با ۳۴ نمونه برنده شده است. در مجموع برای اجرا اول ۴۰۸ نمونه از ۱۰۰۰ نمونه در ۲۰ نورون برتر ذخیره شده است.

جدول ۱ – ۲۰ نورون برنده برای حالت اول: مقداردهی رندوم و چینش خطی

cluster	label=0	label=1	label=2	label=3	label=4	label=5	label=6	label=7	label=8	label=9	sum
182	0	32	0	0	0	0	0	2	0	0	34
73	2	0	0	1	0	0	0	0	23	0	26
52	0	23	1	0	1	0	0	1	0	0	26
47	1	0	1	1	13	0	0	3	0	6	25
204	0	23	0	1	0	0	0	1	0	0	25
594	0	0	0	22	0	1	0	0	1	0	24
297	0	0	0	0	0	0	0	16	1	4	21
343	0	19	1	0	0	0	0	1	0	0	21
552	0	19	0	0	0	0	0	0	1	0	20
512	19	0	0	0	0	0	0	0	0	0	19
34	0	0	0	6	0	2	0	0	9	1	18
156	0	0	1	0	1	0	0	12	2	2	18
522	0	0	0	0	1	0	0	1	0	15	17
220	0	0	0	0	0	0	0	17	0	0	17
285	0	0	0	0	0	0	16	0	1	0	17
421	0	0	0	17	0	0	0	0	0	0	17
571	0	0	0	0	0	1	15	0	0	0	16
579	0	0	0	0	1	0	0	0	3	12	16
202	0	0	0	0	11	0	0	1	0	4	16
304	15	0	0	0	0	0	0	0	0	0	15

حالت دوم: مقداردهی اولیه با استفاده از نمونه‌های ورودی و چینش خطی

در این بخش ماتریس وزن به صورت تصادفی مقدار دهی نمی‌شود بلکه از ویژگی‌های نمونه‌ها استفاده شده است. در حالت قبل از ۱۰۰۰ نمونه برای یادگیری som استفاده شد و ماتریس وزن به‌صورت ۶۲۵*۷۸۴ بود. حال لازم است تا از ۱۰۰۰ نمونه ورودی ۶۲۵ تا به عنوان ورودی انتخاب شود و ماتریس وزن مقداردهی شود. در این حالت زمان runtime ۱۴۳ ثانیه گزارش شده است. لازم به ذکر است الگوریتم و سایر پارامترها یکسان خواهد بود. در این حالت جدول ۱۱*۲۰ برای مقداردهی ماتریس وزن به‌صورت گفته شده به صورت زیر شد. در مجموع برای اجرا دوم ۹۴ نمونه از ۱۰۰۰ نمونه در ۲۰ نورون برتر ذخیره شده است.

جدول ۱ – ۲۰ نورون برنده برای حالت دوم: مقداردهی با استفاده از نمونه‌ها و چینش خطی

cluster	label=0	label=1	label=2	label=3	label=4	label=5	label=6	label=7	label=8	label=9	sum
498	7	0	0	0	0	0	0	0	0	0	7
479	0	0	0	0	0	0	0	7	0	0	7
276	0	0	0	6	0	0	0	0	0	0	6
109	0	0	0	0	0	0	6	0	0	0	6
377	0	6	0	0	0	0	0	0	0	0	6
530	0	0	0	0	0	0	6	0	0	0	6
428	0	0	0	0	4	0	0	0	0	1	5
501	0	5	0	0	0	0	0	0	0	0	5
51	0	0	0	0	0	0	0	0	5	0	5
228	0	0	0	0	0	0	5	0	0	0	5
517	0	0	0	0	0	0	0	0	5	0	5
423	0	0	0	0	0	0	0	4	0	0	4
541	0	0	0	0	0	0	0	0	4	0	4
281	4	0	0	0	0	0	0	0	0	0	4
282	4	0	0	0	0	0	0	0	0	0	4
89	0	0	4	0	0	0	0	0	0	0	4
132	0	0	4	0	0	0	0	0	0	0	4
415	0	4	0	0	0	0	0	0	0	0	4
127	0	0	1	0	2	0	0	0	0	1	4
298	0	0	0	0	0	4	0	0	0	0	4

حالت سوم: مقداردهی اولیه رندوم و با همسایگی ۱ در فضای دوبعدی ۲۵*۲۵

در این بخش برای نورون‌ها چینش دوبعدی درنظر گرفته شده است. به گونه‌ای که ۶۲۵ نورون به صورت یک فضای دوبعدی ۲۵ در ۲۵ چیده شده‌اند. همچنین همسایگی با شعاع یک نیز درنظر گرفته شده است. به گونه‌ای‌که اگر یک نورون برنده شود، علاوه‌بر خود نورون، همسایگان آن نیز تا شعاع یک برنده می‌شوند. یعنی برای هر نورون، نورون سمت راست، چپ، بالا و پایین آن نیز آپدیت خواهد شد. بنابراین فقط بخش به‌روزرسانی ماتریس وزن در الگوریتم متفاوت خواهد شد. هنگامی‌که تابع فاصله تعریف شده یک ایندکس به عنوان خروجی می‌دهد، علاوه بر ستون متناظر با آن ایندکس در ماتریس وزن، نورنوهای همسایه نیز باید به‌روزرسانی شوند. بدین‌منظور، نورون‌های کناری با ایندکس یکی کمتر و یکی بیشتر، همچنین ۲۵ ستون قبلی (برای نورون بالا) و ۲۵ ستون بعدی (برای نورون پایین) آپدیت شده‌اند. لازم به ذکر است که برای نورون‌هایی که در گوشه قرار می‌گیرند باید شرط‌هایی گذاشته شود. به عنوان مثال اگر نورون در ردیف اول باشد ۲۵ نورون قبل برای آن تعریف نشده بوده و باید تنها

نورون‌هایی که در ردیف دوم به بعد هستند، نورون بالایی در نظر گرفته شود. برای ۳ جهت دیگر نیز شرط گذاشته شده است. درنهایت زمان runtime ۲۴۸ ثانیه گزارش شده است. در مجموع برای اجرا اول ۱۲۱ نمونه از ۱۰۰۰ نمونه در ۲۰ نورون برتر ذخیره شده است.

جدول ۳ – ۲۰ نورون برنده برای حالت سوم: مقداردهی رندوم و چینش دوبعدی

cluster	label=0	label=1	label=2	label=3	label=4	label=5	label=6	label=7	label=8	label=9	sum
589	0	0	0	0	0	0	0	0	8	0	8
543	0	7	0	0	0	0	0	1	0	0	8
610	0	0	0	8	0	0	0	0	0	0	8
480	0	7	0	0	0	0	0	0	0	0	7
434	0	7	0	0	0	0	0	0	0	0	7
231	0	0	0	0	1	0	0	0	0	5	6
539	0	0	0	0	0	0	0	0	6	0	6
414	0	0	0	0	0	0	0	6	0	0	6
432	0	6	0	0	0	0	0	0	0	0	6
506	0	6	0	0	0	0	0	0	0	0	6
462	0	0	0	0	0	0	0	6	0	0	6
253	0	0	0	0	0	0	6	0	0	0	6
130	0	0	0	0	0	0	6	0	0	0	6
465	0	0	0	0	0	0	0	5	0	0	5
229	0	0	0	0	1	0	4	0	0	0	5
362	0	0	0	0	0	0	0	0	0	5	5
356	0	0	0	0	1	0	0	0	0	4	5
386	0	0	0	0	2	0	0	0	0	3	5
210	0	0	0	0	5	0	0	0	0	0	5
406	0	5	0	0	0	0	0	0	0	0	5

حالت چهارم: مقداردهی اولیه با استفاده از نمونه‌های ورودی و با همسایگی ۱ در فضای دوبعدی ۲۵*۲۵

این بخش همانند قسمت سوم است با این تفاوت که ماتریس وزن، درابتدا با استفاده از مقادیر نمونه‌ها مقداردهی شده است. زمان runtime ۲۴۸ ثانیه گزارش شده است. درادامه جدول مربوط به نورون‌های برنده به همراه کلاس‌های مربوطه آمده است. در مجموع برای اجرا اول ۱۱۹ نمونه از ۱۰۰۰ نمونه در ۲۰ نورون برتر ذخیره شده است.

جدول ۴ – ۲۰ نورون برنده برای حالت چهارم: مقداردهی توسط نمونه‌ها و چینش دوبعدی

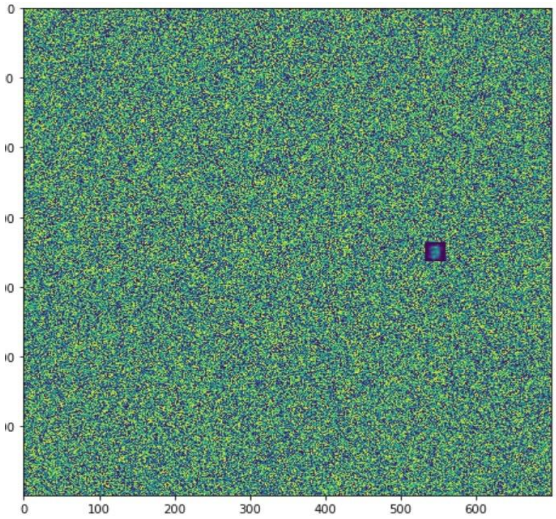
cluster	label=0	label=1	label=2	label=3	label=4	label=5	label=6	label=7	label=8	label=9	sum
473	0	8	0	0	0	0	0	0	0	0	8
120	0	0	0	0	6	0	0	0	0	1	7
157	0	6	0	0	0	0	0	0	0	0	6
514	0	0	0	0	1	0	0	0	0	5	6
449	0	6	0	0	0	0	0	0	0	0	6
115	0	0	0	0	4	0	0	0	0	2	6
131	0	6	0	0	0	0	0	0	0	0	6
535	0	0	0	0	6	0	0	0	0	0	6
601	0	0	0	0	6	0	0	0	0	0	6
198	6	0	0	0	0	0	0	0	0	0	6
571	0	0	0	0	0	0	0	0	6	0	6
618	0	0	0	0	0	0	0	0	6	0	6
284	6	0	0	0	0	0	0	0	0	0	6
355	0	0	0	6	0	0	0	0	0	0	6
325	0	6	0	0	0	0	0	0	0	0	6
416	0	0	0	0	0	0	0	0	0	6	6
103	0	0	0	0	1	0	0	1	1	2	5
239	0	0	0	0	0	5	0	0	0	0	5
98	0	0	0	0	0	0	0	5	0	0	5
107	5	0	0	0	0	0	0	0	0	0	5

۳,۲,۱ تصویرسازی روند Clustring

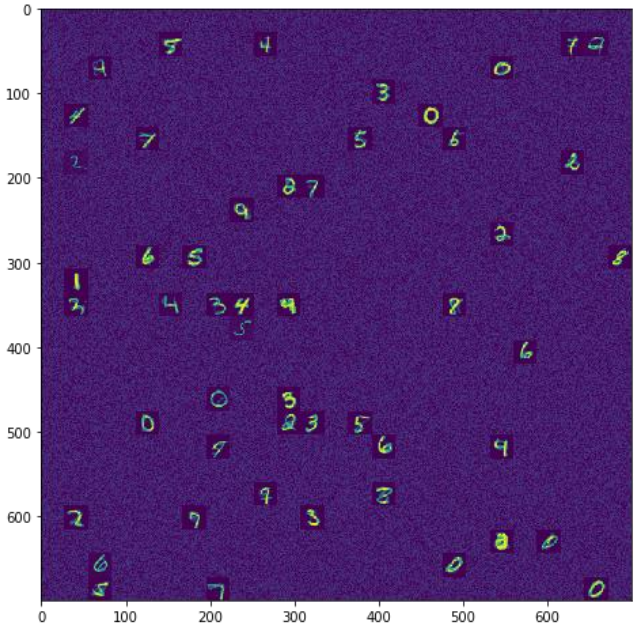
در این بخش خواسته شده تا روند clustering دیتاست MNIST تصویرسازی شود. درواقع ماتریس وزن دارای ابعاد ۶۲۵*۷۸۴ است و حاصل آن ۴۹۰۰۰۰ می‌شود که همان ۷۰۰*۷۰۰ خواهد بود. درواقع لازم است یک جابه‌جایی در ماتریس وزن صورت گیرد تا تصاویر هر عدد به صورت ۲۸*۲۸ (مربعی) نشان داده شود نه خطی! بدین‌صورت ترانهاده ماتریس وزن به‌صورت ۲۸*۲۸*۶۲۵ شده و بعد از آن، ۲۸*۲۸*۲۵*۲۵ شده است. حال با جابه‌جایی ستون بعد دوم و سوم ماتریس به صورت ۲۸*۲۵*۲۸*۲۵ درآمده است که پس از تغییر سایز آن به ماتریس ۷۰۰ در ۷۰۰ و ویژالایز کردن آن، تصاویر زیر برای ۴ اجرا مختلف مشاهده شده است.

حالت اول: مقداردهی اولیه رندوم و چینش خطی نورون‌ها

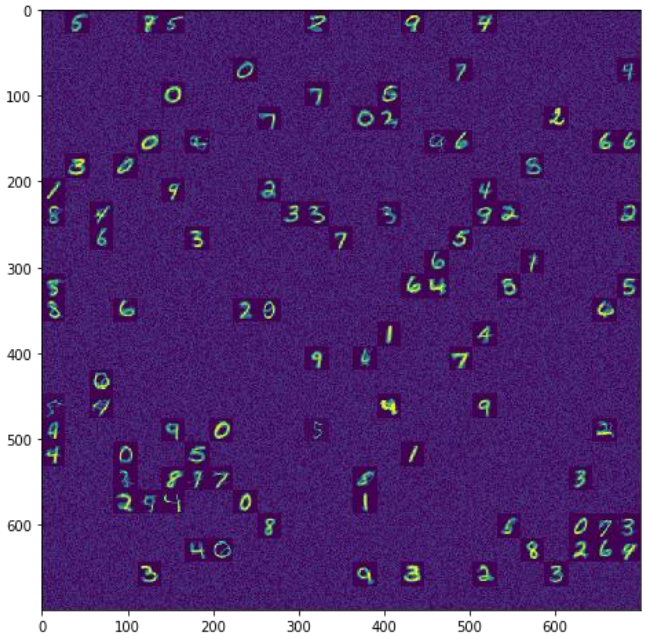
لازم به ذکر است همانطور که پیش‌تر گفته شد ماتریس وزن پس از initialize شدن به صورت تصادفی، بر عدد ۶ نیز تقسیم شد. اگر ماتریس وزن بر عدد ۶ تقسیم نمی‌شد پس از ۲۰ ایپاک خروجی زیر مشاهده می‌گردید که نشان می‌دهد به دلیل پرت بودن مقادیر ماتریس وزن، فقط به یک نمونه همگرایی صورت گرفته است.



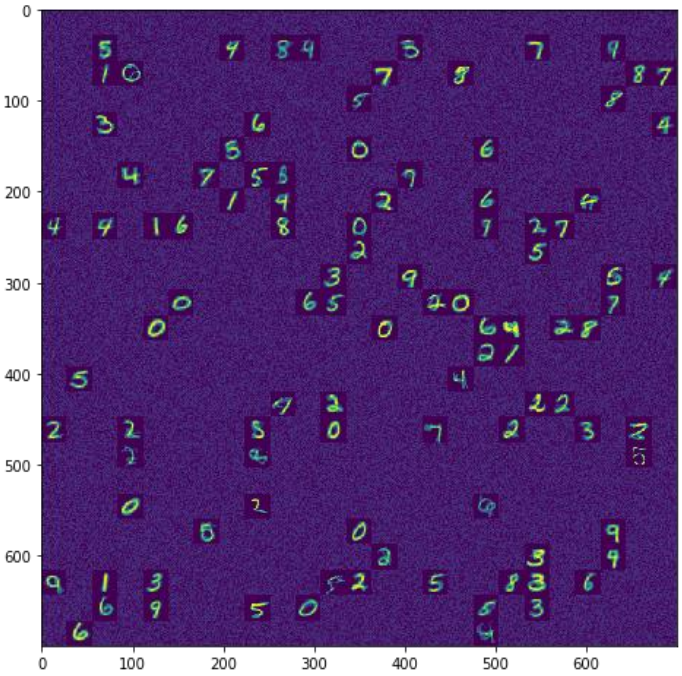
شکل ۱ - برای حالت اول: مقداردهی رندوم (بدون تقسیم بر ۶) و چینش خطی در ایپاک بیستم



شکل ۲ - برای حالت اول: مقداردهی رندوم و چینش خطی در ایپاک اول

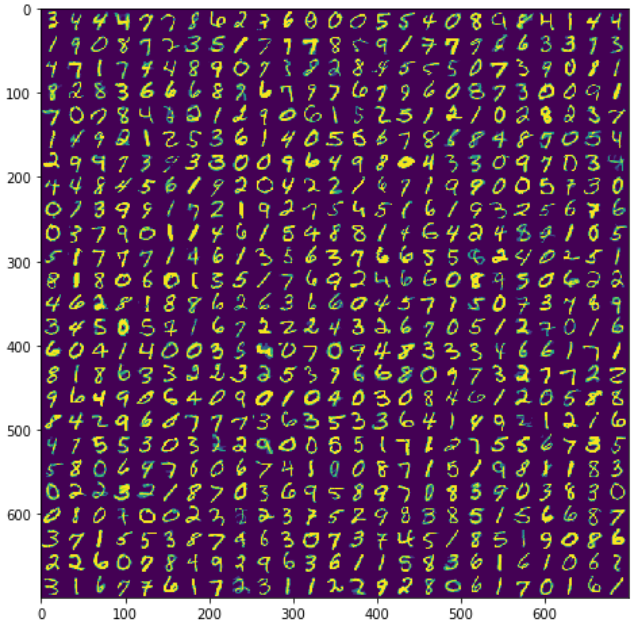


شکل ۳ - برای حالت اول: مقداردهی رندوم و چینش خطی در ایپاک دهم

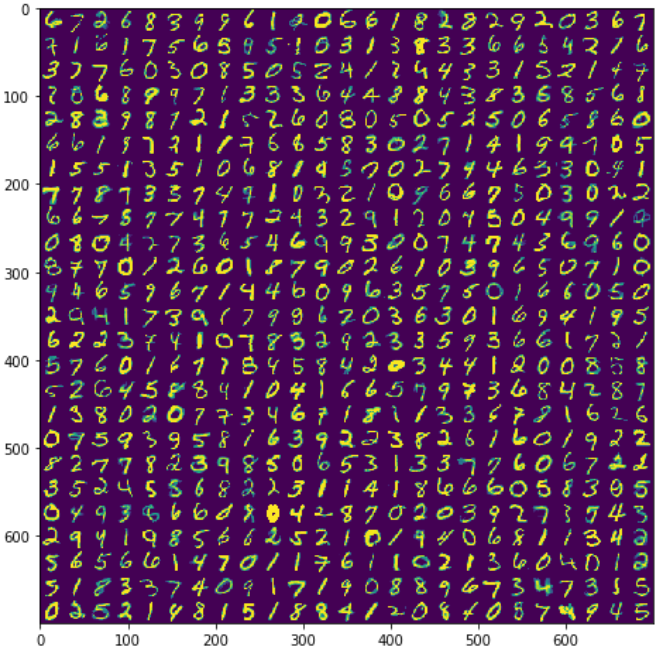


شکل ۴ - برای حالت اول: مقداردهی رندوم و چینش خطی در ایپاک بیستم

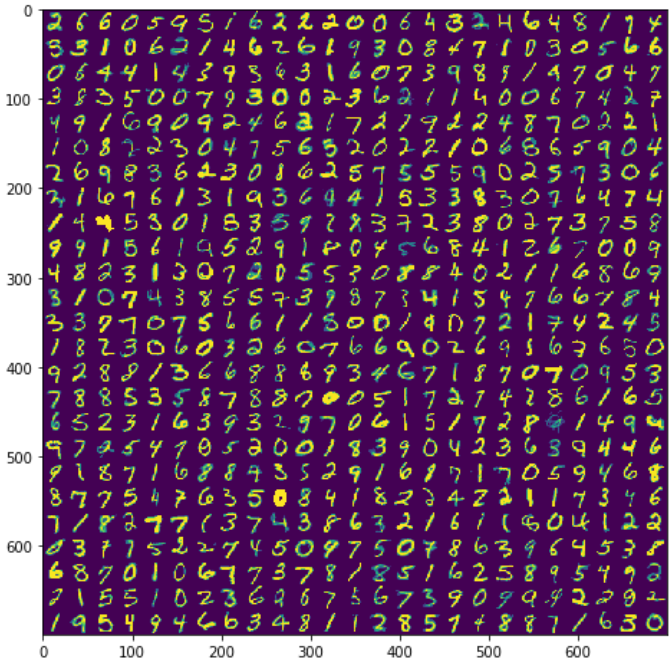
حالت دوم: مقداردهی اولیه با استفاده از نمونه‌های ورودی و چینش خطی



شکل ۵ – برای حالت دوم: مقداردهی با استفاده از نمونه و چینش خطی در ایپاک اول

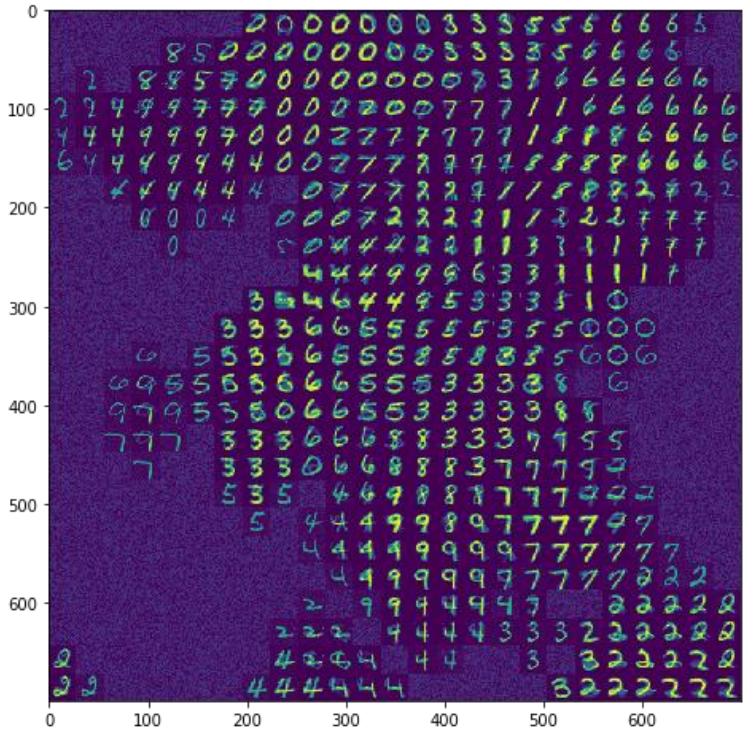


شکل ۶ – برای حالت دوم: مقداردهی با استفاده از نمونه و چینش خطی در ایپاک دهم

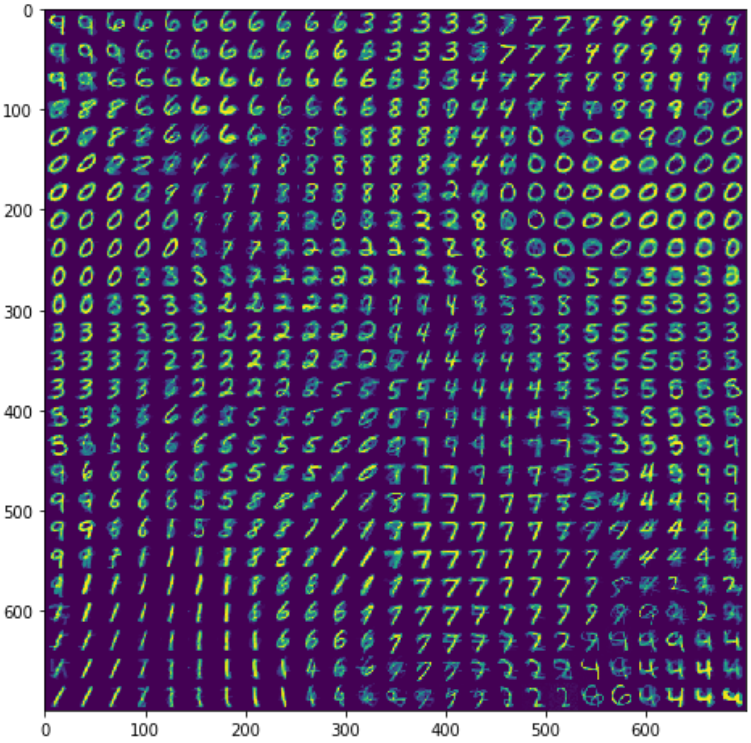


شکل ۷ – برای حالت دوم: مقداردهی با استفاده از نمونه و چینش خطی در ایپاک بیستم

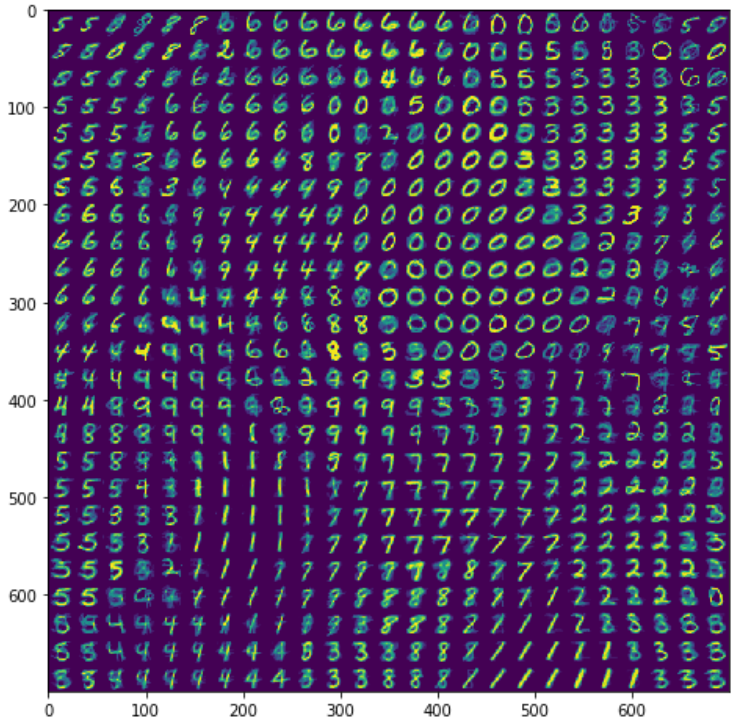
حالت سوم: مقداردهی اولیه رندوم و با همسایگی ۱ در فضای دوبعدی ۲۵*۲۵



شکل ۸ – برای حالت سوم: مقداردهی رندوم و چینش دوبعدی در ایپاک اول

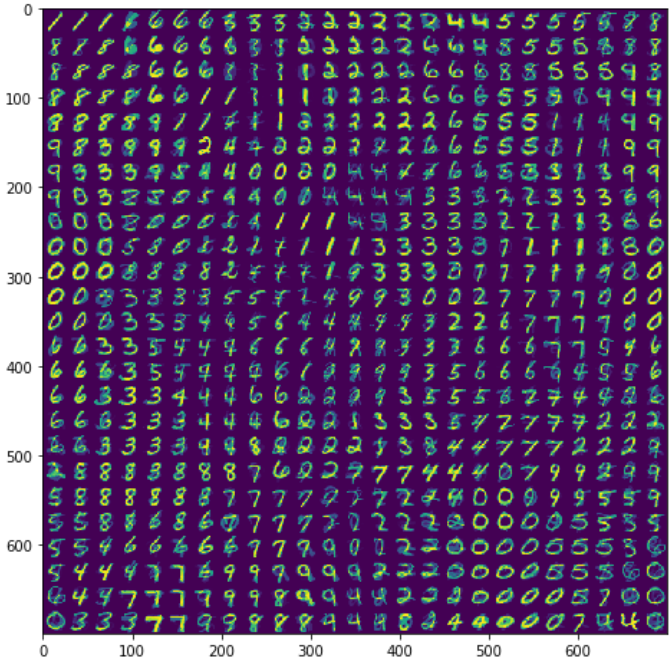


شکل ۹ – برای حالت سوم: مقداردهی رندوم و چینش دوبعدی در ایپاک دهم

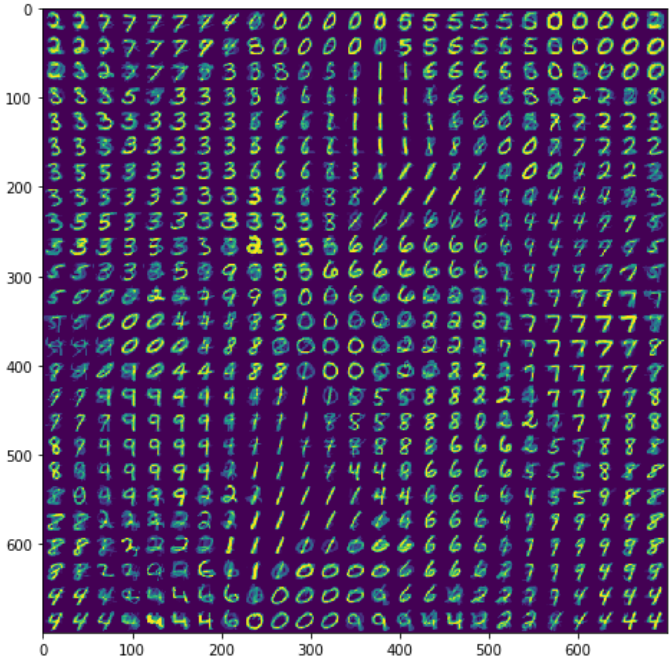


شکل ۱۰ – برای حالت سوم: مقداردهی رندوم و چینش دوبعدی در ایپاک بیستم

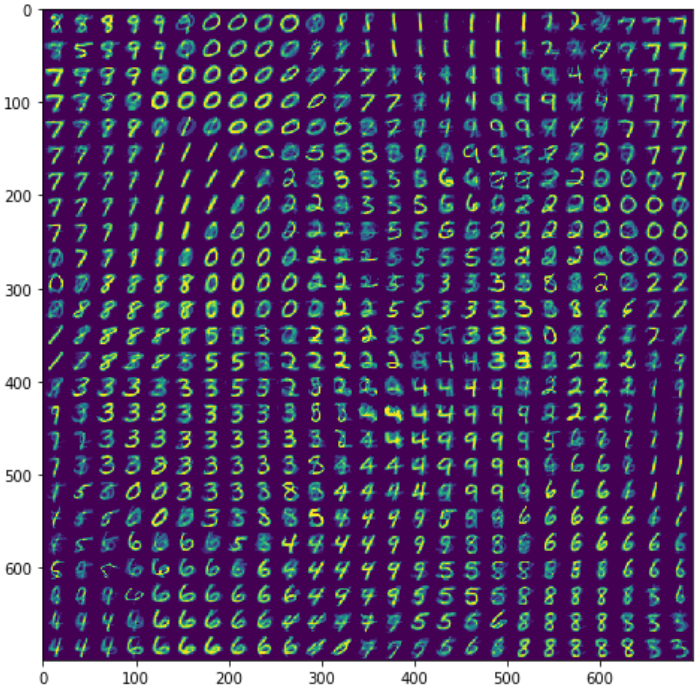
حالت چهارم: مقداردهی توسط نمونه‌های ورودی و چینش دوبعدی



شکل ۱۱ – برای حالت چهارم: مقداردهی توسط نمونه‌ها و چینش دوبعدی در ایپاک یکم



شکل ۱۲ - برای حالت چهارم: مقداردهی توسط نمونه‌ها و چینش دوبعدی در ایپاک دهم



شکل ۱۳ - برای حالت چهارم: مقداردهی توسط نمونه‌ها و چینش دوبعدی در ایپاک بیستم

۳,۱ مقایسه عملکرد ۴ اجرا

تا اینجا یک شبکه SOM برای دیتاست MNIST طراحی‌شده است و دیده شد که از ۴ اجرا متفاوت برای بررسی عملکرد شبکه استفاده شده است. درواقع دو مدل برای نحوه مقداردهی به ماتریس وزن (به‌صورت تصادفی و با استفاده از ویژگی‌های نمونه‌های ورودی) و دو مدل برای چینش نورون‌ها (به‌صورت خطی یا دوبعدی)

با جداول مربوط به هر اجرا، مشاهده می‌شود هنگامی‌که ماتریس وزن به صورت رندوم مقداردهی می‌شود، نورون‌ها نمونه‌های بیش‌تری را به خود جذب می‌کنند و این نمونه ممکن است دسته‌های متفاوتی هم باشد. حال هنگامی‌که برای مقداردهی اولیه ماتریس وزن از ویژگی نمونه‌های ورودی استفاده می‌شود، نورون‌ها نمونه‌های کمتری را گرفته اما این نمونه‌ها بیشتر مربوط به یک کلاس مشخص هستند تا چند کلاس. بنابراین درحالتی که ماتریس وزن با نمونه‌ها initialize می‌شود، همگرایی سریعتر و clustering بهتر صورت می‌گیرد.

هنگامی که چینش نورون‌ها به صورت دوبعدی می‌شود، هر نورون‌ها و همسایه‌های آن در یک کلاس خاص تخصص پیدا می‌کنند و ازآنجایی که یا خود نورون و یا همسایگانش نمونه‌ها را می‌گیرند، تعداد سمپل‌هایی که هر نورون به خود می‌گیرد کاهش می‌یابد. به طور کلی همانطور که در مقدمه توضیح داده شد، هنگامی که تعداد نورون‌ها زیاد است، چینش‌های چندبعدی ارجعیت داشته و افزایش شعاع همسایگی به clustering بهتر کمک می‌کند.

۲,۱ تشخیص بیماری با استفاده از شبکه همینگ

در ابتدا یک سمپل سالم درنظر گرفته شده است و به عنوان ورودی به شبکه اعمال می‌گردد. ازطرفی دو مرجع به عنوان بیمار و سالم به صورت بردارهایی با ۱۱ درایه ساخته شده است. بدین صورت که برای مرجع سالم، همه درایه‌ها -۱ بوده و برای مرجع بیمار، همه درایه‌ها +۱ است. در نهایت یک ماتریس وزن ۲*۱۱ برای این شبکه با استفاده از دو بردار مرجع ساخته شده است. در شبکه همینگ، مقادیر وزن ثابت بوده و برابر است با نصف بردارهای مرجع. از طرفی یک بایاس نیز تعریف می‌گردد که برای هر دو مرجع مقدار آن ۱۱/۲ است. خروجی شبکه در واقع یک بردار دو درایه‌ای خواهد بود که درایه اول آن نشان‌دهنده سالم بودن و درایه دوم آن نشان مریض بودن فرد خواهد بود. حال مقادیر خروجی بدین صورت تشکیل می‌شود که نمونه در ماتریس وزن ضرب می‌شود و مقدار آن با بایاس جمع می‌گردد. حال هر کدوم از درایه‌های خروجی که بزرگ‌تر بود، بردار ورودی به آن نزدیک‌تر بوده و خروجی متناظر با آن فعال خواهد بود. برای این منظور از یک شبکه MaxNet نیز استفاده شده است تا خروجی شبکه رو تعیین کند.

برای طراحی شبکه MaxNet در ابتدا یک متغیر اپسیلون تعریف شده است. از آنجایی که بردار خروجی دو درایه داره و مقدار اپسیلون باید کوچکتر از ½ باشد، مقدار ۰,۴ برای آن انتخاب شده است. حال خروجی این بخش بدین صورت تعریف می‌گردد که از مقدار ورودی متناظر با خروجی اپسیلون برابر مجموع مقادیر دیگر کاسته می‌شود. حال این خروجی باید از تابع فعال‌ساز عبور کند تا به ازای مقادیر منفی صفر شود که همان تابع relu است. درنهایت خروجی شبکه MaxNet محاسبه می‌شود که درواقع خروجی شبکه HammingNet نیز

می‌باشد. بنابراین با اعمال یک نمونه و مقایسه با دو مرجع سالم یا بیمار، یک بردار به عنوان میزان شباهت سمپل با هر مرجع می‌دهد. حال با مرجعی که بیش‌ترین شباهت را داشته باشد متعلق به آن دسته بود و MaxNet آنرا مشخص می‌کند. در ادامه دو نمونه بیمار و سالم به شبکه داده شده است و خروجی‌های زیر مشاهده گردید.

```
sample is : [[-1  1 -1 -1 -1  1 -1  1 -1  1 -1]]
MaxNet input is: [[7. 4.]]
MaxNet output/HammingNet output is [4.92 0.  ]
she/he is ok.
```

شکل ۱۴ – نمونه سالم اعمال شده به شبکه و خروجی آن

```
sample is : [[ 1  1 -1  1 -1  1 -1  1  1  1  1]]
MaxNet input is: [[3. 8.]]
MaxNet output/HammingNet output is [0.  6.8]
she/he is sick.
```

شکل ۱۵ – نمونه بیمار اعمال شده به شبکه و خروجی آن