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

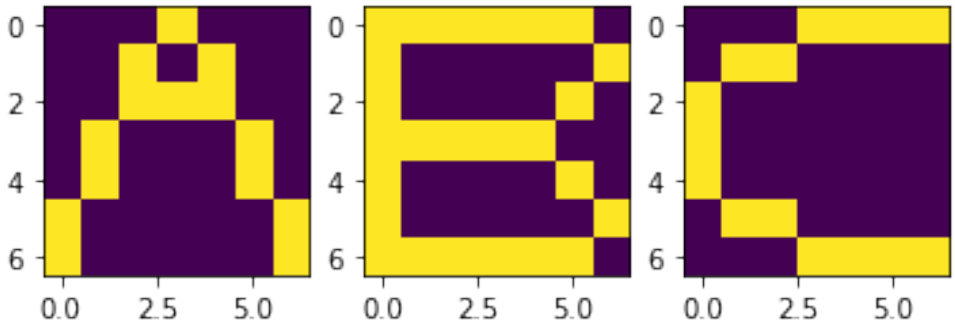
پاییز ۹۸

۱. بازشناسی به روش هب

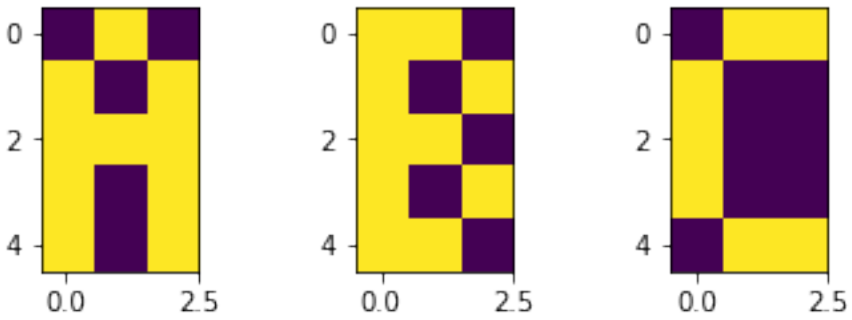
در طراحی شبکه‌های عصبی با هدف تداعی کردن یک پترن ورودی نیاز به حافظه است و خروجی مدنظر را می‌دهد. حال پترن ورودی می‌تواند به صورت explicit یا implicit باشد. منظور از ورودی implicit آن است که ورودی‌ها توسط یک اردر زمانی یا مکانی به شبکه داده می‌شوند. درغیراینصورت ورودی‌ها explicit است. شاید بتوان گفت شبکه‌ای که برای ورودی‌های explicit طراحی می‌گردد، از جنس مسائل classification است با این تفاوت که تعداد کلاس‌ها بیشتر می‌باشد. درواقع شبکه‌های عصبی با کاربرد حافظه درصورتی که ورودی‌ها explicit باشند نیاز به یک حافظه استاتیکی دارند. بدیهی است که طراحی یک شبکه عصبی برای ورودی‌های explicit ساده‌تر از طراحی شبکه برای داده‌های implicit است. یکی از الگوریتم‌های یادگیری متداول برای شبکه‌های عصبی با حافظه استاتیکی، قانون یادگیری هب است. در ادامه یک شبکه عصبی تک‌لایه با قانون هب طراحی می‌گردد.

۱.۱ طراحی یک شبکه با قانون هب

هدف از این قسمت کاهش ابعاد سه حرف A، B و C است. در ابتدا حروف مذکور به صورت ماتریس ۷*۷ داده شده‌اند و انتظار می‌رود که به ابعاد ۳*۵ تداعی شوند. در ادامه هر حرف و تارگت مدنظر آن آمده است. لازم به ذکر است رنگ زرد نماد درایه ۱ و رنگ بنفش نماد درایه ۱- است.



شکل ۱ – ورودی‌های explicit شبکه (Sample) به صورت ماتریس ۷*۷

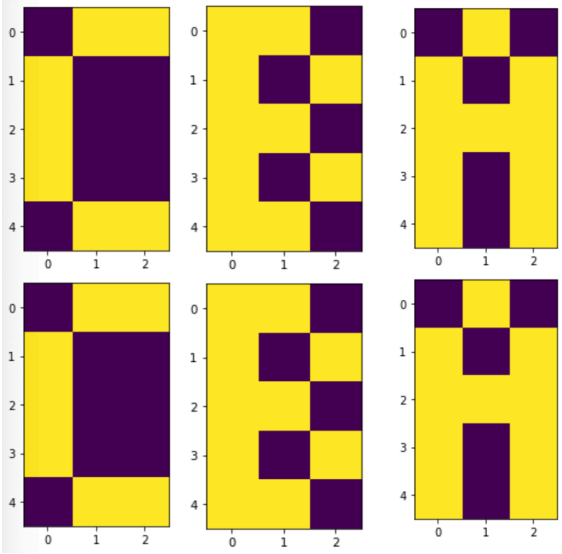


شکل ۲ - خروجی‌های explicit شبکه (Target) به صورت ماتریس ۳*۵

درابتدا هر حرف ۷*۷ به صورت یک ماتریس با ابعاد ۱ و ۱- در شبکه به صورت یک بردار ۴۹ درایه‌ای ذخیره شده‌اند و همه به عنوان ورودی شبکه کانکت شده و ماتریس ورودی با ابعاد 49*3 ساخته شد. بدیهی است سه تعداد نمونه‌های ورودی است. ازطرفی ماتریس تارگت نیز ساخته شد. به صورتی که هر حرف خروجی با ابعاد 3*5 به یک بردار ۱۵ درایه‌ای تبدیل شده و پس از کانکت شدن به یه بردار تارگت با ابعاد 3*15 تبدیل گردید.

برای محاسبه ماتریس وزن، بردار نمونه‌ها یا همان ورودی به صورت ترنسپوزشده در ماتریس تارگت ضرب شد. از آنجایی که ماتریس‌های ورودی و تارگت برای هر نمونه به ترتیب دارای ۴۹ و ۱۵ درایه بوده‌اند، ابعاد ماتریس وزن به صورت 15*49 درآمد.

برای آزمودن آنکه آیا به ازای هر سمپل، تارگت متناظر تداعی می‌شود یا نه کافی است نمونه مدنظر در ماتریس وزن ضرب گردد و از یک تابع فعال‌ساز عبور کند. بدیهی است بدون استفاده از تابع فعال‌ساز، شبکه خطی بوده و فاقد نقاط اکسترمم خواهد بود اما شبکه یک تله می‌خواهد که به ازای ورودی‌های disturbed شده نیز در آن گیر کند. درواقع شبکه کاملا خطی نمی‌تواند حافظه ایجاد کند. تابع فعال‌ساز استفاده شده در این بخش تابع sign به صورت bipolar است. به‌گونه‌ای که اگر هر یک از درایه‌های بردار حاصل از ضرب سمپل در ماتریس وزن بزرگتر از صفر بود، درایه ۱ و درغیر اینصورت درایه ۱- گردد. در نهایت پس از اعمال هر یک از نمونه‌ها به شبکه خروجی متناظر دیده شد. همچنین از مفهوم Hamming Distance نیز استفاده شده است. به گونه‌ای که خروجی هر شبکه به ازای هر ورودی با تارگت‌ها مقایسه گردید که برای هر ۳ نمونه این فاصله صفر بوده است. برای حروف A، B و C ماتریس تارگت و خروجی شبکه به صورت زیر مشاهده گردید. ملاحظه می‌گردد تارگت‌ها و خروجی‌های شبکه کاملا مطابق یکدیگر می‌باشند.



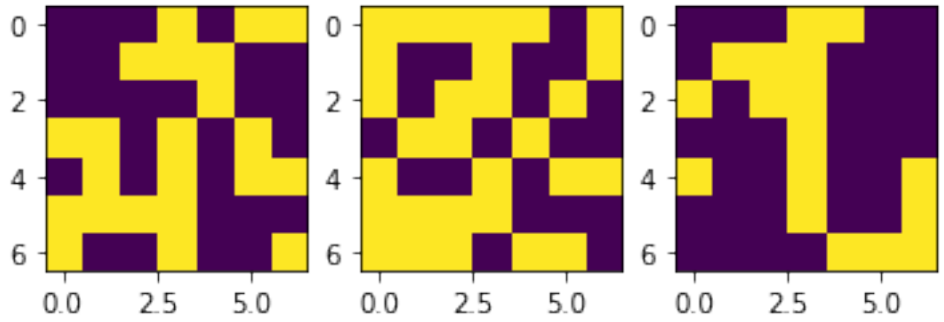
شکل ۳ – تارگت‌های مدنظر و خروجی‌های مشاهده‌شده

۲.۱ افزود اغتشاش به شبکه طراحی‌شده

در قسمت قبل پس از ساخت ماتریس وزن، همون نمونه‌های که در ساخت ماتریس وزن نقش داشتند به شبکه اعمال گردید. هدف از این بخش آزمودن قدرت شبکه برای داده‌هایی متفاوت از سمپل‌های ورودی است به‌گونه‌ای که برخی از داده‌ها با نویز ترکیب و یا برخی اطلاعات حذف می‌شوند. در ادامه شبکه به دو صورت‌گفته شده آزمایش می‌شود.

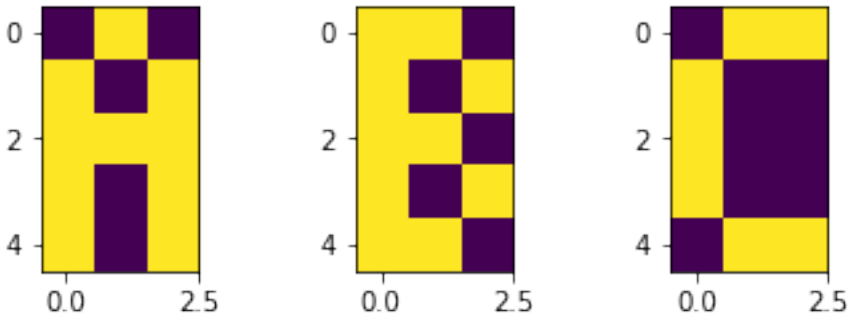
- **افزودن نویز به سمپل‌های ورودی**

بدیهی است افزودن نویز به نمونه‌ها باید به صورت رندوم باشد. برای تحقق این امر ابتدا ۳۰ درصد از خانه‌های ورودی به صورت رندوم انتخاب گردید اما درنهایت شبکه برای ۲۸ درصد نویز مقاوم بود. بنابراین از ۴۹ خانه هر سمپل، ۱۴ خانه به صورت رندوم با نویز پر شد. بدین صورت که اگر ۱۴ خانه انتخاب‌شده حاوی ۱ باشند ۱- گردند و اگر حاوی ۱- هستند ۱ شوند. در اینصورت نمونه‌های جدید ساخته شد. در بخش قبل برای ماتریس‌های هر حرف، خروجی متناظر داده شد. اما در این بخش نمونه‌ها به صورت زیر درآمده است.



شکل ۴ – سمپل‌های ورودی با ۲۸ درصد نویز

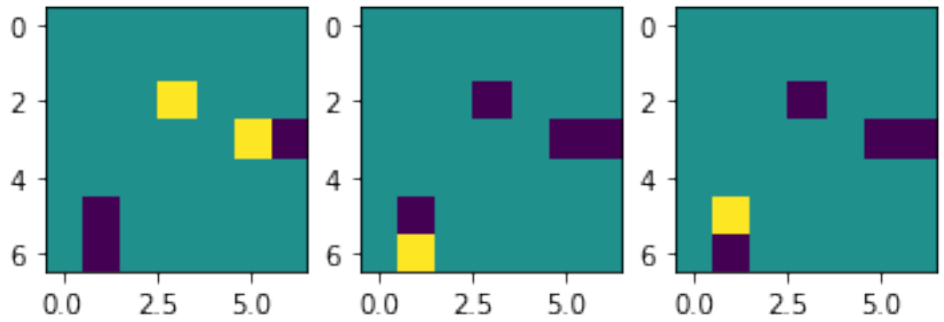
حال با محاسبه حاصلضرب هر نمونه در ماتریس وزن خروجی شبکه و گذراندن از تابع‌فعال‌ساز، خروجی شبکه به صورت زیر مشاهده گردید. بنابراین همانطور که پیش‌تر گفته شد، اگه ورودی‌های شبکه طراحی شده تا ۲۸.۶ درصد دچار نویز شوند، شبکه مقاوم بوده و خروجی‌های مدنظر را می‌دهد.



شکل ۵ – خروجی شبکه برای سمپل‌های ترکیب‌شده با نویز

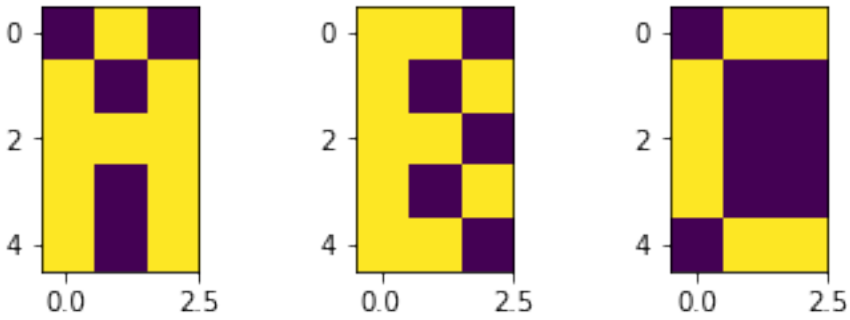
- **افزودن Missing Point به ورودی‌های شبکه**

در این بخش پس از انتخاب خانه‌های ماتریس‌های ورودی، برای ایجاد Missing Point، خانه‌های انتخاب‌شده فارغ از مقادیری که دارند به صفر تبدیل‌شده اند که در ادامه آمده است.



شکل ۶ – سمپل‌های شبکه با حذف مقادیر برخی خانه‌ها

در این حالت نیز پس از محاسبه حاصلضرب ماتریس وزن و نمونه‌ها و عبور از تابع فعال‌ساز خروجی شبکه به صورت زیر مشاهده گردید. لازم به ذکر است در این حالت حدود ۹۰ درصد از خانه‌های هر سمپل (۴۴ خانه از ۴۹ خانه) حذف گردید و شبکه قدرت بازشناسی را داشت.



شکل ۷ – خروجی شبکه برای سمپل‌هایی که ۹۰ درصد اطلاعات آنها حذف شده بود

۲. شبکه خودانجمنی با استفاده از هب

در شبکه‌های خودانجمنی یا Auto-associative ورودی و خروجی هم جنس بوده و هدف تداعی کردن سمپل s(p) از همان سمپل ورودی s(p) است. درواقع فایده این شبکه‌ها آن است که بتوان پس از disturb شدن ورودی، آنرا بازیابی کرد. شبکه خودتداعی‌گر شبیه به آتوانکودرها می‌باشد با این تفاوت که هدف در آتوانکودرها آن است که با dimension reduction، correlation ها شکسته شود؛ ولی در شبکه‌های خودتداعی‌گر ابعاد کاهش نمی‌یابد و هدف disturbance rejection است. بنابراین در ای شبکه‌ها حافظه‌ای ساخته می‌شود که نسبت به disturbance مقاوم هستند. درادامه یک شبکه خودتداعی‌گر ساخته می‌شود.

۱.۲ ساخت ماتریس وزن برای یک شبکه خودتداعی‌گر

هدف از این بخش طراحی یک شبکه خودتداعی‌گر در راستای ایجاد حافظه برای ذخیره‌سازی بردار داده‌شده S می‌باشد. برای طراحی این شبکه در ابتدا بردار s به صورت bipolar ذخیره می‌گردد. حال لازم است ماتریس وزن ساخته شود. از آنجایی که شبکه خودتداعی‌گر است، ورودی و خروجی یکسان بوده و برای محاسبه ماتریس وزن، ورودی به صورت ترنسپوز در همان بردار ورودی ضرب می‌شود. به نظر می‌رسد که در هنگامی که یک بردار و ترنسپوز آن در هم ضرب می‌شوند باعث میشود قطر ماتریس وزن یک شود. (ضرب یک در یک یا یک در منفی یک، یک می‌شود.) بنابراین برای تعداد سمپل بالا، ماتریس به سمت همانی شدن رفته و disturbance rejection انجام نمی‌شود؛ زیرا ورودی‌های disturbed شده در یک ماتریس همانی ضرب می‌شوند؛ درنتیجه همان ورودی‌های disturbed شده در خروجی مشاهده می‌گردند. بدین صورت قطر ماتریس وزن صفر می‌شود تا به ازای تعداد سمپل زیاد نیز، حذف disturbance حتمی باشد. به این روش قانون یادگیری هب اصلاح‌شده می‌گویند. درنهایت ماتریس وزن به صورت زیر بدست آمد.

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

شکل ۸ – ماتریس وزن برای شبکه خودتداعی‌گر طراحی شده

۲.۲ تست قوام شبکه طراحی‌شده

در این بخش برای افزودن اشتباه، دو خانه به صورت رندم انتخاب گردید و مقادیر آن‌ها عوض شد. بدین صورت که اگر هر خانخ حاوی یک بود، منفی یک شد و برعکس. شبکه برای دو اشتباه قادر به ساخت ورودی غیرنویزی شده نشد. اما به ازای یک خطا، توانست به بردار سمپل اولیه تداعی شود. بنابراین شبکه قدرت تحمل ۲۵ درصد (یک خانه از ۴ خانه) نویزی شدن را دارد. درنهایت توسط Hamming Distance نشان داده شد که اگر ورودی تا ۲۵ درصد نویزی شود، شبکه همان بردار را به صورت تمیز تداعی می‌کند. در ادامه خروجی شبکه به ازای داده نویزی و غیر نویزی و همچنین فاصله همینگ این دو بردار آمده است.

```
Output is [[ 1.  1.  1. -1.]] .
Target is [[ 1  1  1 -1]] .
Hamming distance between output and target is 0 .
```

شکل ۹ – فاصله همینگ دو بردار خروجی و تارگت

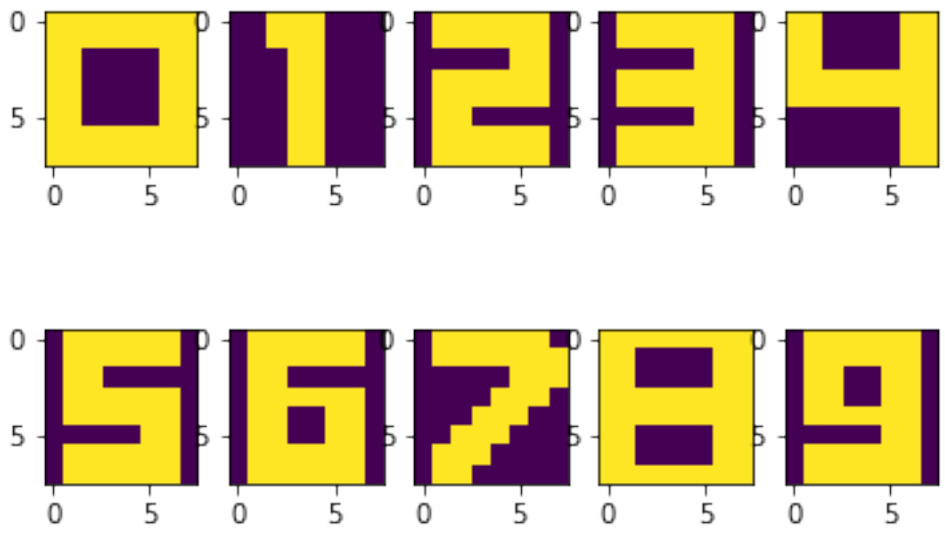
۳. شبکه هاپفیلد

در قسمت‌های قبل شبکه‌های عصبی با کاربرد حافظه طراحی گردید. شبکه‌های طراحی‌شده در بخش اول توسط قانون هب و در بخش دوم یک شبکه خودتداعی‌گر توسط قانون هب اصلاح‌شده آموزش داده شدند. درواقع یادگیری به روش هب یک قابلیت برای به حافظه سپاری بردارهای ورودی نسبت به بردار خروجی می‌دهد. برای قوی‌تر کردن شبکه‌ها طراحی‌شده می‌توان خروجی را مجدد در ورودی اعمال کرد؛ بنابراین شبکه‌ها به صورت recurrent می‌شوند. به عنوان مثال شبکه هاپفیلد از خروجی‌ها نیز در ساخت شبکه بهره می‌برد. لازم به ذکر است ماتریس وزن شبکه هاپفیلد همان ماتریس وزن هب است. شبکه‌های recurrent قدرت به حافظه‌سپاری را افزایش می‌دهند و با تعدا محدودی تکرار همگرا خواهند شد. در ادامه یک شبکه هاپفیلد برای ذخیرسازی اعداد ۰ تا ۹ طراحی می‌گردد.

۱.۳ طراحی یک شبکه هاپفیلد برای ذخیره‌سازی ده عدد ۰ تا ۹

هدف از این بخش، همانطور که پیش‌تر گفته شد، طراحی یک شبکه عصبی recurrent توسط الگوریتم هاپفیلد است. در این شبکه قرار است اعداد ۰ تا ۹ که به صورت ماتریس ۸*۸ هستند ذخیره گردد. لازم به ذکر تارگت این شبکه همان سمپل‌های ورودی بوده و شبکه به صورت خودتداعی‌گر است.

درابتدا سمپل‌های ورودی به صورت ماتریس‌های ۸*۸ و سپس به صورت یک بردار ۶۴*۱ ذخیره شده و پس از کانکت کردن همه نمونه‌ها بردار سمپل با ابعاد ۶۴*۱۰ ساخته شد. در ادامه ماتریس‌ها ساخته شده با ابعاد ۸*۸ آمده است.

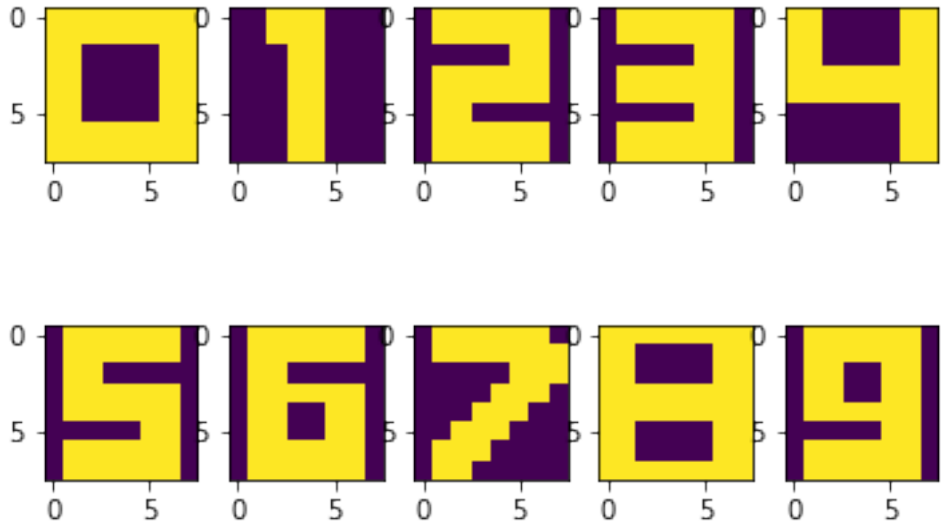


شکل ۱۰ – سمپل‌های ذخیره‌شده در شبکه هاپفیلد طراحی‌شده

ازآنجایی که شبکه خودتداعی‌گر بوده، لازم است برای ساخت ماتریس وزن، قطر ماتریس ۰ گردد تا از تبدیل ماتریس وزن به ماتریس همانی جلوگیری شود. بنابراین برای ساخت ماتریس وزن ماتریس سمپل‌های ورودی به صورت ترنسپوز در خودش ضرب می‌شود و قطر آن صفر می‌گردد. برای حذف قطر ماتریس وزن، از ۱۰ برابر یک ماتریس همانی استفاده شده است. در نهایت ماتریس وزن بر عدد ۲۵۰ تقسیم گردید تا استپ‌های بین ورودی و خروجی زیاد نشود و شبکه بتواند مقادیر تارکت را با ورودی‌ها یک و منفی یک تداعی کند.

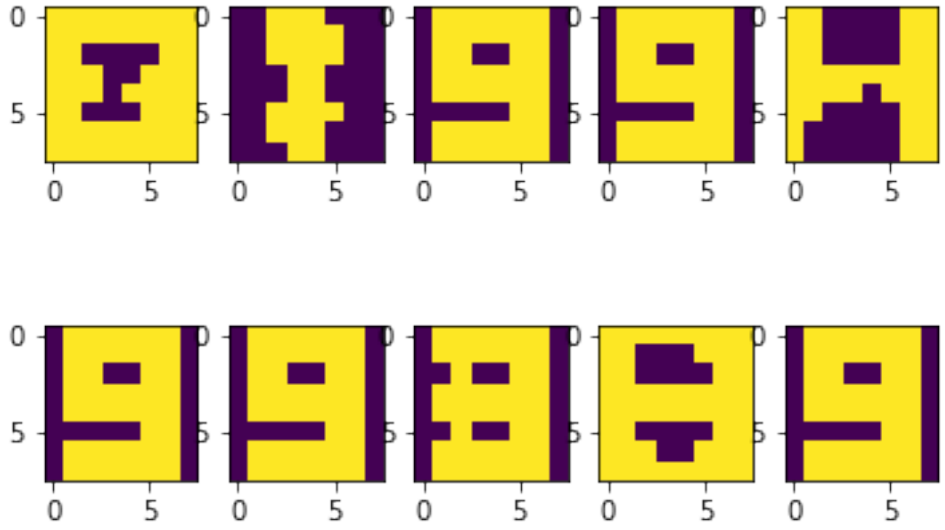
برای این بخش یک تابع فعال‌ساز شبیه به تابع sign طراحی گردید. بدین صورت که اگر ورودی بزرگ‌تر از صفر بود، خروجی یک شود و اگر کمتر از صفر بود منهای یک را به عنوان خروجی دهد. برای حالتی که ورودی صفر می‌شود نیز همان خروجی را بدهد. درنهایت الگوریتم هاپفیلد به صورت زیر پیاده شد:

به ازای هر سمپل، تا هنگامی همگرایی صورت نگرفته باشد، عملیات را ادامه می‌دهد. در ابتدا یک random order تعریف می‌شود که خروجی هر سمپل به ازای آن به‌روزرسانی گردد. حال برای هر رندوم اردر، ماتریس خروجی در ستون nام از ماتریس وزن ضرب می‌گردد. منظور از ستون nام ماتریس وزن، ستون مربوط به رندوم اردر است. البته ماتریس خروجی و وزن به صورت درایه‌ای در هم ضرب شده‌اند و درنهایت یک عدد اسکالر می‌دهند. درنهایت خروجی به تابع فعال‌ساز تعریف شده داده می‌شود و خروجی مدنظر ساخته می‌شود. دراین‌حالت شبکه به ازای همه ورودی‌ها خودش را تداعی می‌کند که در ادامه آمده است.



شکل ۱۱ – خروجی شبکه خودتداعی‌گر به ازای سمپل و ماتریس وزنی که تقسیم بر ۲۵۰ شده است

لازم به ذکر است اگر ماتریس وزن بر عدد ۲۵۰ تقسیم نمی‌شد، خروجی شبکه به ازای هر سمپل به صورت زیر مشاهده می‌گردید.

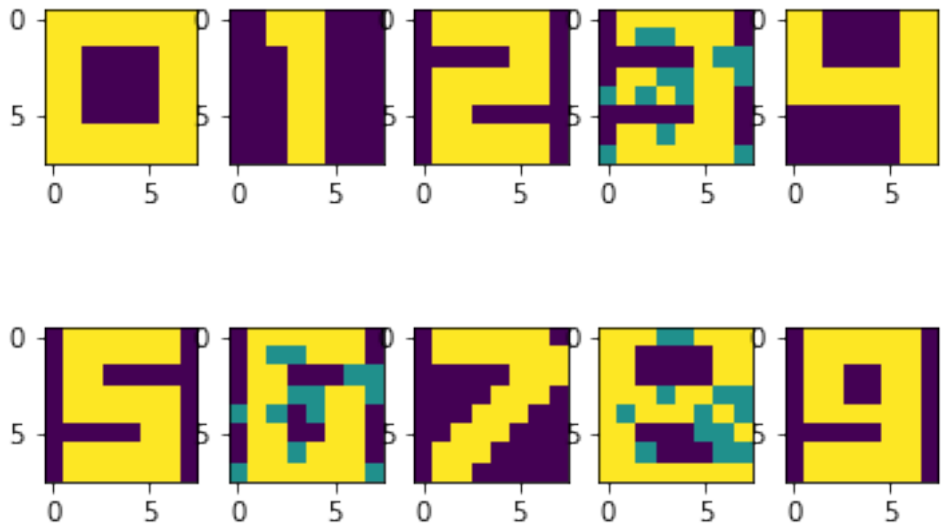


شکل ۱۲ – خروجی شبکه به ازای ماتریس وزن اولیه

۲.۳ افزودن نویز به اعداد ۳، ۶ و ۸

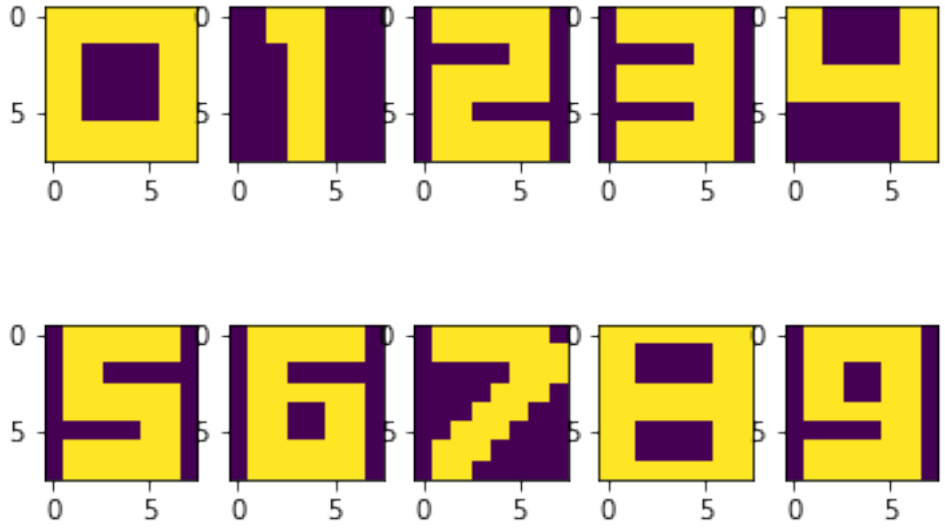
در بخش قبل، یک شبکه خودتداعی‌گر برای ذخیره اعدا ۰ تا ۹ و بازیابی آن‌ها با دادن سمپل مربوطه طراحی گردید. در این بخش هدف آن است که سمپل‌های ورودی نویزی شوند و قدرت شبکه برای آنها آزمایش گردد. در ادامه به صورت رندوم ۱۳ خانه از ۶۴ خانه مربوط به اعداد ۳، ۶ و ۸نویزی گردید (در حدود ۲۰

درصد). لازم به ذکر است برای هر عدد، خانه‌های مختلفی برای نویزی کردن انتخاب شده است. درنهایت خانه مربوط به این خانه‌ها صفر گردید و نمونه‌های نویزی به صورت زیر مشاهده شد.



شکل ۱۳ – سمپل‌های نویزی‌شده

با اعمال سمپل‌های نویزی به شکبکه طراحی شده خروجی به صورت زیر مشاهده گردید. البته از آنجایی که خانه‌هایی که نویزی می‌شوند به صورت رندوم انتخاب می‌شوند، در برخی موارد سمپل‌ها به صورت تارگت‌های مربوطه تداعی نمی‌شوند.



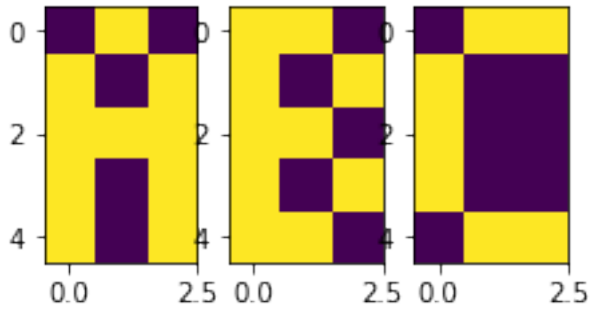
شکل ۱۴ – خروجی شبکه خودتداعی‌گر برای سمپل‌های نویزی‌شده

۴. شبکه BAM

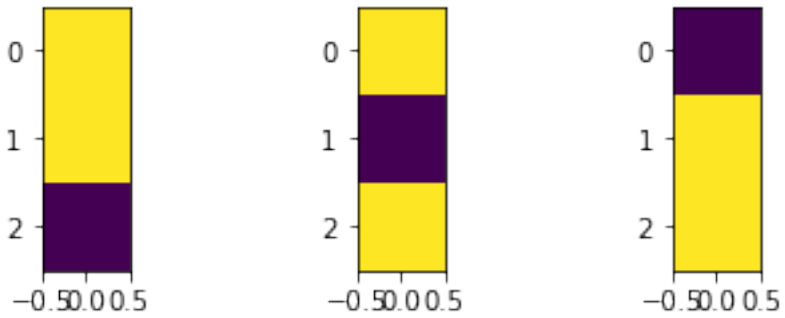
پیش‌تر گفته شد که recurrentکردن شبکه‌های عصبی خودتداعی‌گر قدرت شبکه را برای تداعی کردن افزایش می‌دهد. سوالی که مطرح می‌شود آن است که آیا میتوان یک شبکه auto-associative را recurrent کرد؟ درواقع recurrent کردن این شبکه‌ها به صورت bidirectional بوده و ورودی از هر طرف می‌آید؛ به‌گونه‌ای‌که سمپل‌ها می‌خواهند به تارگت‌ها تداعی شوند و تارگت‌ها می‌خواهند به سمپل‌ها تداعی گردند. دراین حالت درواقع شبکه براساس یک تابع خطا تعریف می‌گردد. تضمین می‌کند که اگر سمپل‌ها و تارگت‌های متناظر با ماتریس هب ذخیره گردند، پس از disturbed شدن هم به یکدیگر تداعی خواهند شد. در ادامه یک شبکه BAM طراحی می‌گردد.

۱.۴ طراحی یک شبکه BAM

هدف از این بخش، ذخیره‌سازی حروف A، B و C و تارگت‌های متناظر با آنها است. سمپل‌ها ماتریس‌های ۵*۳ بوده و تارگت متناظر آنها یک بردار سه‌درایه‌ای است. در ابتدا سه نمونه گفته‌شده ذخیره و کانکت شده‌اند تا ماتریس سمپل با ابعاد ۱۵*۳ ساخته شود. همچنین ماتریس تارگت با ابعاد ۳*۳ ساخته شد تا خروجی متناظر به هر حرف را تداعی کند. در ادامه سمپل‌های مدنظر و خروجی متناظر با آنها نشان داده شده است. رنگ زرد نشان‌دهنده ۱ و رنگ بنفش نشان‌دهنده ۱- است.



شکل ۱۵ – سمپل‌های شبکه BAM



شکل ۱۶ – تارگت‌های شبکه BAM

بدیهی است که در یک شبکه BAM از آنجایی که سمپل و تارگت هم‌جنس نیستند، نمی‌توان از ماتریس وزن هب اصلاح‌شده استفاده کرد. بنابراین در شبکه BAM از همان ماتریس هب استفاده می‌گردد. در این مساله برای ساخت ماتریس وزن، ماتریس سمپل به صورت ترنسپوز در ماتریس تارگت ضرب شده است. درنهایت ماتریس وزن این شبکه به صورت زیر مشاهده گردید.

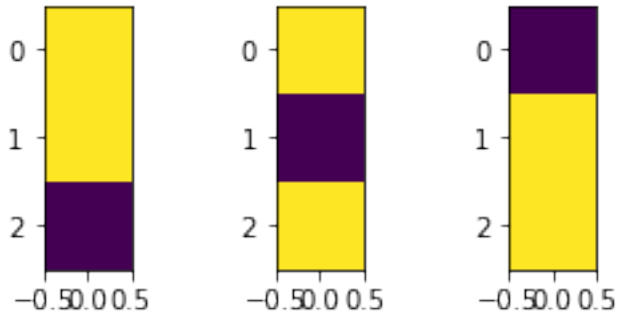
```
array([[ 1., -3.,  1.],
       [ 1.,  1.,  1.],
       [-3.,  1.,  1.],
       [ 1.,  1.,  1.],
       [-1., -1., -1.],
       [ 3., -1., -1.],
       [ 1.,  1.,  1.],
       [ 3., -1., -1.],
       [ 1.,  1., -3.],
       [ 1.,  1.,  1.],
       [-1., -1., -1.],
       [ 3., -1., -1.],
       [ 3., -1., -1.],
       [-1., -1.,  3.],
       [-1.,  3., -1.]])
```

شکل ۱۷ – ماتریس وزن ساخته‌شده برای شبکه BAM

۲.۴ آزمودن شبکه طراحی‌شده

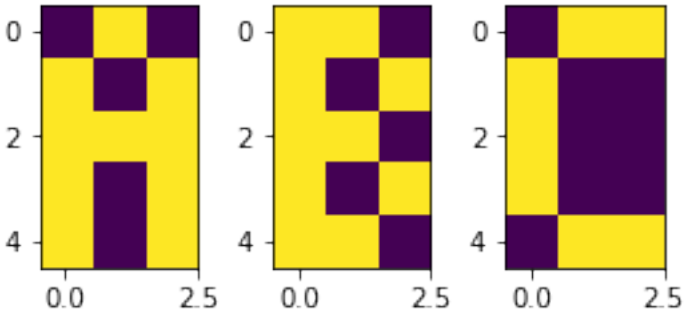
برای آزمودن شبکه لازم است سمپل‌های ورودی از یک طرف و تارگت‌ها از طرف دیگر به شبکه اعمال شوند. برای این امر کافی است الگوریتم هاپفیلد پیاده‌سازی گردد. در ابتدا سمپل‌ها به شبکه اعمال شده‌اند و انتظار می‌رود که تارگت‌ها متناظر مشاهده گردد. در این حالت الگوریتم هاپفیلد تا جایی ادامه خواهد داشت

که تداعی صورت گیرد. در ابتدا برای ستون هر سمپل در سطر ماتریس وزن ضرب می‌شود و یک عدد اسکالر بدست می‌آید که به تابع فعال‌ساز اعمال می‌شود تا خروجی شبکه ساخته شود. در نهایت برای هر سمپل خروجی متناظر مشاهده گردید. لازم به ذکر است در الگوریتم BAM از رندوم اردر استفاده نشده است.



شکل ۱۸ – خروجی شبکه BAM به ازای سمپل‌ها

همین روند برای تارگت‌ها طی می‌شود تا سمپل‌ها تداعی گردند. در آن حالت خروجی شبکه به صورت زیر مشاهده گردید که با سمپل‌های ورودی یکسان هستند.

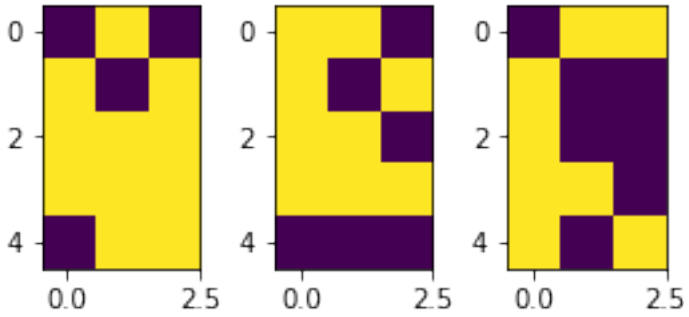


شکل ۱۹ – خروجی شبکه BAM به ازای اعمال تارگت‌ها

۳.۴ افزودن نویز به شبکه

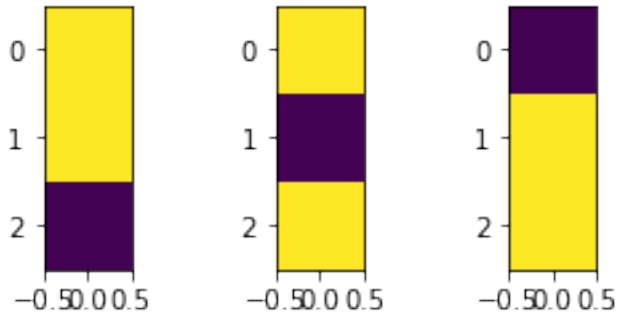
در این بخش به شبکه BAM طراحی شده، نویز افزوده شده تا قدرت آن آزمایش گردد. بدین منظور به سمپل‌های ورودی نویز اضافه شده تا مشاهده گردد که آیا به خروجی مدنظر تداعی می شوند یا نه. در ابتدا خانه‌هایی از هر سمپل به صورت رندوم انتخاب گردید و به ازای آن خانه‌ها، سمپل ها تغییر علامت یافتند. به‌گونه‌ای‌که اگر حاوی یک بودند به منفی یک تبدیل شدند و اگر منفی یک بودند به یک تبدیل شدند. این روند برای تعداد مختلف از خانه‌های سمپل تست گردید و درنهایت شبکه به ازای ۳ تغییر علامت توانست به

تارگت مدنظر تداعی گردد. در ادامه سمپل‌های اعمال‌شده به شبکه آورده شده است. لازم به ذکر است در هر سمپل ۳ خانه دچار نویز شده است.



شکل ۲۰ – سمپل‌های نویزی

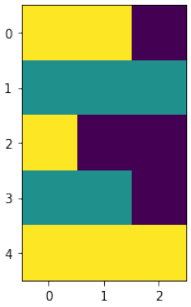
درنهایت هر سمپل نویزی به شبکه اعمال گردید. به‌گونه‌ای‌که بردار متناظر با هر سمپل در ماتریس وزن ضرب شدند و تابع فعال‌ساز بر خروجی اعمال گردید. درنهایت خروجی‌های زیر مشاهده شد. البته در هر بار ران گرفتن، ممکن است خروجی به درستی تداعی نشوند؛ زیرا خروجی شبکه به خانه‌هایی از سمپل‌ها که نویزی می شود حساس بوده و ممکن است حرف مدنظر را تشخیص ندهد. درنهایت شبکه با ۲۰ درصد خطا به خروجی‌های زیر تداعی گردید.



شکل ۲۱ – خروجی شبکه به ازای سمپل‌های نویزی

۴.۴ اعمال تارگت جدید به شبکه

در این قسمت یک بردار تارگت فارغ از بردارهای تارگت اولیه به شبکه اعمال شد. متاسفانه بردار تارگت جدید نتوانست به ماتریس‌های حروف اولیه تداعی شود و خروجی شکل زیر را داد.



شکل ۲۲ – خروجی تداعی‌شده به ازای تارگت اعمالی

۵.۴ محاسبه فاصله همینگ

هدف از این بخش محاسبه فاصله همینگ بین هر دو بردار ورودی است. درواقع فاصله همینگ تعداد بیت‌هایی غیریکسان برای دو بردار هم‌بعد است. در اینجا با استفاده از دستور count_nonzero از کتابخانه numpy، تعداد بیت‌های غیریکسان بردارهای سمپل مشاهده گردیده است. لازم به ذکر است فاصله همینگ برای هر دو بردار تعریف شده است. حال هرچه فاصله همینگ بین هر دو بردار کمتر باشد، احتمال آنکه با افزودن نویز خروجی شبکه به تارگت‌های مدنظر تداعی نشود، بیشتر خواهد شد. به عنوان مثال برای حروف A و B احتمال به خطا افتادن شبکه بیشتر خواهد بود.

```
Hamming distance between A and B is 4 .
Hamming distance between A and c is 7 .
Hamming distance between B and C is 7 .
```

شکل ۲۳ – فاصله همینگ بین هر دو بردار