|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر**  **شبکه های عصبی و یادگیری عمیق**  **تمرین سری سوم** | | |

|  |  |
| --- | --- |
| بهراد محمودی | نام و نام خانوادگی |
| 810100464 | شماره‌ دانشجویی |
| 23/2/1401 | تاریخ ارسال گزارش |

­

**فهرست گزارش سوالات**

[سوال 1 –Pattern Association using Hebbian Learning Rule 3](#_Toc32834855)

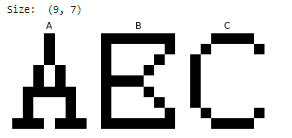
[سوال ۲ – Auto-associative Net 4](#_Toc32834856)

[سوال 3 – Discrete Hopfield Network 5](#_Toc32834857)

[سوال 4 – Bidirectional Associative Memory 6](#_Toc32834858)

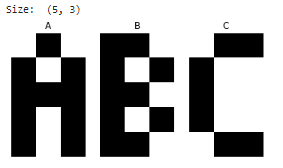
# سوال 1 – شناسایی حروف با استفاده از شبکه هب

در این سوال می­خواهیم با استفاده از شبکه هب الگوهای 9\*7 را به الگو­های 5\*3 تبدیل کنیم. ابتدا الگوهای مورد نظر را به صورت زیر رسم می­کنیم.



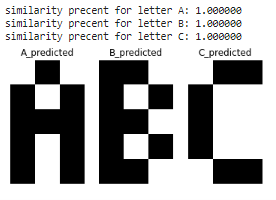
شکل 1.1- الگوی ورودی­های شبکه مورد نظر

انتظار داریم وقتی الگوهای شکل 1.1 را به عنوان ورودی به شبکه طراحی شده می­دهیم خروجی­های زیر حاصل گردند.



شکل 1.2- الگوی خروجی­های قابل انتظار با ورودی­های شکل 1.1

برای ساخت شبکه از یک کلاس با نام patternAssociatingNetاستفاده می­کنیم که متدهای hebbianLearning، که برای آموزش شبکه و تنظیم وزن­ها مورد استفاده قرار می­گیرد، و recallLetter، که برای تعیین خروجی می­باشد، متد­های اصلی آن می­باشند. با استفاده از این شبکه و الگو­های فوق برای آموزش شبکه وقتی الگوهای شکل 1.1 را به عنوان ورودی به این شبکه اعمال می­کنیم خروجی­ها به شکل زیر خواهند بود. با توجه به شکل زیر مشاهده می­کنیم که خروجی­ها کاملا منطبق با آنچه انتظار داشتیم می­باشند.



شکل 1.3- خروجی­های شبکه با وروی­های شکل 1.1

ب) حال می­خواهیم حساسیت شبکه را به ورودی نویزی و همچنین ورودی که اطلاعات آن از بین رفته است را بسنجیم. برای این کار ابتدا به هر یک از حروف به ترتیب 10 و 25 درصد نویز اضافه می­کنیم و حروف نویزی را به شبکه می­دهیم. برای اینکه یک دید جامع نسبت به این موضوع داشته باشیم به تعداد 100 مرتبه داده­ها را نویزی کرده و به ورودی شبکه اعمال می­کنیم. در صورتی که شبکه به صورت کاملا صحیح هر سه حرف را بازیابی کرده باشد، شمارنده را یکی اضافه می­کنیم. در نهایت نتیجه به صورت زیر خواهد شد.

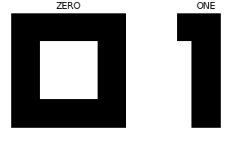
مشاهده می­شود که هرچه نویز بیشتر باشد، تاثیر آن روی حرف بازیابی شده توسط شبکه بیشتر خواهد بود. حال به جای اعمال نویز اطلاعات نمونه­ها را حذف می­کنیم. با همان توضیحات قسمت قبل خروجی­ها در این قسمت به صورت زیر خواهند بود.

مشاهده می­شود که از دست رفتن اطلاعات تا 25 درصد تاثیری در خروجی بازیابی شده توسط شبکه نداشته است اما با افزایش این حذف اطلاعات نیز خروجی تحت تاثیر قرار خواهد گرفت. با توجه به نتایج بدست آمده می­توان دریافت که خروجی شبکه طراحی شده نسبت به حذف اطلاعات ورودی حساسیت کمتری نسبت به ورودی نویزی دارد، چرا که با افزایش نویز به 25 درصد دقت کاهش پیدا کرده اما با افزایش خذف اطلاعات ورودی به همین مقدار دقت خروجی نسبت به حالت قبلی تغییری نکرده است.

# سوال ۲ – شبکه خودانجمنی

# سوال 3 – شبکه هاپفیلد

در این سوال خواسته شده که یک شبکه هاپفیلد طراحی کنیم تا قابلیت ذخیره­سازی دو عدد صفر و یک به صورت زیر را داشته باشد.

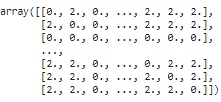


شکل 3.1- الگوهایی که باید در این سوال ذخیره گردند

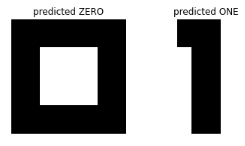
الف) حال باید شبکه را طبق الگوریتم آموزش هب طراحی نمائیم. این الگوریتم به صورت زیر می­باشد.



شکل 3.2- الگوریتم یادگیری هب

با پیاده­سازی الگوریتم فوق ماتریس وزن­های زیر که با توجه به سایز ورودی­ها یک ماتریس 64\*64 می­باشد، بدست خواهد آمد.

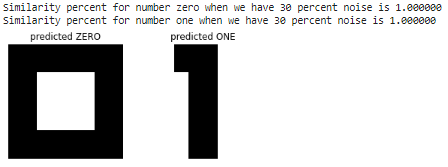
برای تست شبکه به دست آمده می­توانیم ماتریس تصاویر شکل 3.1 را به عنوان ورودی به شبکه بدهیم و خروجی را رسم کنیم. نتایج به صورت زیر خواهند بود.



شکل 3.3- اعداد بازیابی شده توسط شبکه با ورودی­های شکل 3.1

مشاهده می­شود که اعداد به صورت کاملا صحیح بازیابی شده­اند و عملکرد شبکه برای ورودی بدون نویز بی نقص می­باشد.

ب) حال می­خواهیم به ورودی­ها نویز اضافه کنیم و حساسیت شبکه به ورودی نویزی را بسنجیم.



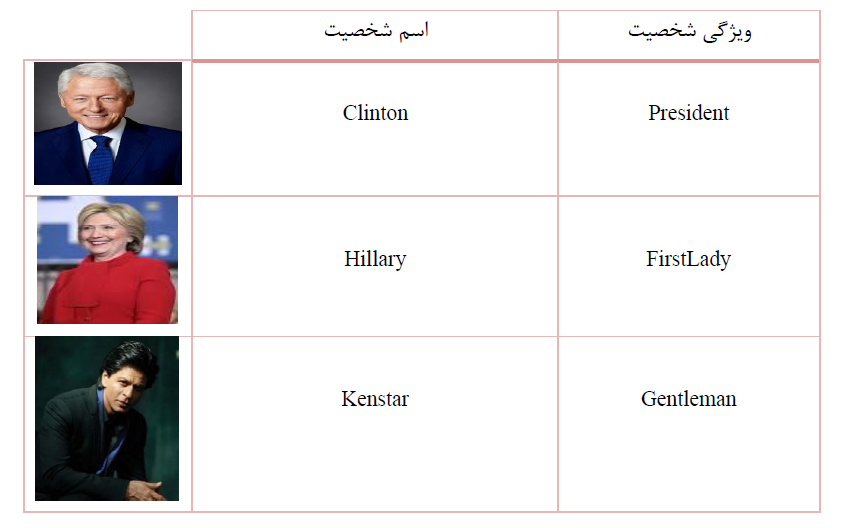
شکل 3.4- اعداد بازیابی شده در صورت اعمال ورودی نویزی

مشاهده می­شود در صورت وجود نویز هم شبکه این توانایی را دارد که به صورت کامل اطلاعات را بازیابی کند. همانطور که مشاهده می­شود اعداد بازیابی شده با ورودی نویزی کاملا منطبق با اعداد اصلی بدون نویز می­باشند.

ج) در واقع فاصله hamming معیاری است که بیانگر تفاوت میان دو ماتریس می­باشد. هرچه این مقدار بزرگتر باشد دو ماتریس از هم قابل تفکیک­تر خواهند بود. با اندازه­گیری فاصله hamming ماتریس دو تصویر شکل 3.1 با استفاده از توابع کتابخانه­ای به عدد 0.71875 می­رسیم که به نسبت بزرگ است (ماکسیمم فاصله hamming می­تواند عدد یک باشد). پس این دو تصویر وجه تمایز زیادی دارند و به راحتی قابل تفکیک هستند به همین دلیل در قسمت قبل با اضافه کردن نویر باز هم توانستیم اطلاعات را به صورت کامل بازیابی کنیم.

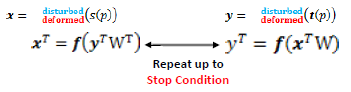
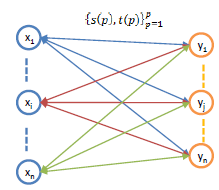
# سوال 4 – شبکه BAM

طبق صورت سوال می­خواهیم شبکه BAM را به گونه­ای طراحی کنیم تا شخصیت های زیر را به همراه ویژگی های منتسبشان ذخیره نماید.



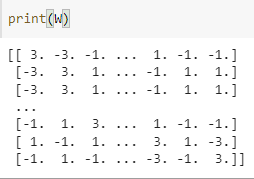
شکل 4.1- الگو های یادگیری برای سوال 4

1) مطابق با مطالب اسلایدها وزن­های شبکه مورد نظر به صورت زیر به دست می­آیند.



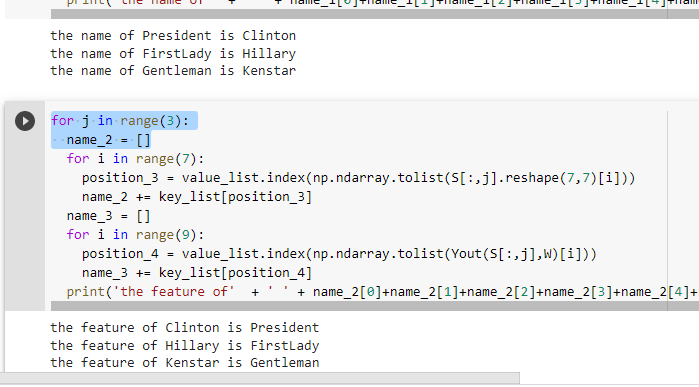
شکل 4.2- شبکه BAM و الگوریتم تنظیم وزن­های آن

با اعمال الگوریتم فوق، وزن­های شبکه مورد نظر به صورت زیر خواهد بود.



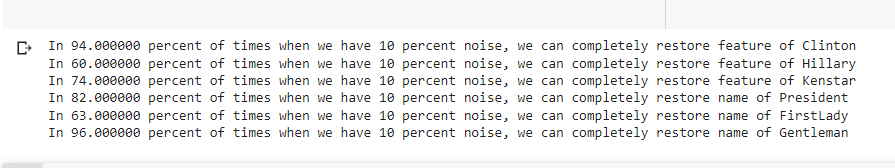
همچنین توجه داریم که تابع فعال­ساز مورد استفاده، تابع فعال­ساز باینری با سطح آستانه صفر می­باشد.

2) حال می­خواهیم شبکه بدست آمده را ارزیابی کنیم. این ارزیابی را از هر دو طرف انجام می­دهیم، یعنی یک بار با دادن شخص مورد نظر به شبکه تست می­کنیم که آیا ویژگی مربوطه را به یاد خواهد آورد؟ بار دیگر با دادن ویژگی به عنوان ورودی به شبکه تست می­کنیم که آیا به شخصیت مورد نظر می­رسیم یا خیر. خروجی این قسمت به صورت زیر می­باشد.

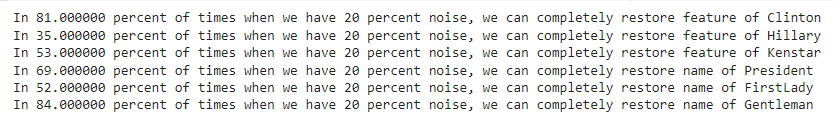


در شکل فوق، می­بینیم که نتیجه تست شبکه از هر دو وجه مطلوب می­باشد، یعنی هم با استفاده از نام شخص توانستیم به ویژگی متناظر برسیم هم با استفاده از ویژگی توانستیم به شخصیت درست برسیم. پس شبکه به خوبی اطلاعات را بازیابی می­کند.

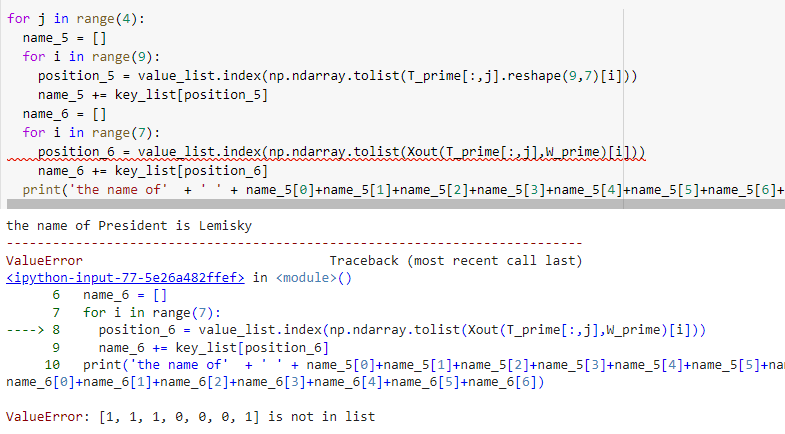
3) با اضافه نمودن نویز به نمونه­ها (تغییر تصادفی 1 به 1- و بالعکس) و تست شبکه با نمونه­های نویزی حاصل به تعداد 100 مرتبه، به نتایج زیر می­رسیم.



مشاهده می­شود که نویز 10 درصد تاثیر خیلی بدی در عملکرد شبکه نداشته است، اما مشاهده می­شود که وقتی 20 درصد از درایه­های یک نمونه را نویزی کنیم، عملکرد شبکه به صورت قابل ملاحظه­ای کاهش می­یابد (یعنی شبکه نسبت به نویز 10 درصد مقاوم است اما به نویز 40 درصد اینگونه نیست) که دلیل بسیار واضحی هم دارد. هرچه نمونه­های ورودی با نمونه­هایی که وزن­های شبکه را با آن­ها تعیین کردیم متفاوت باشند، عملکرد شبکه بیشتر افت خواهد کرد.



4) من با درست کردن یک دیکشنری از تمامی حروف استفاده شده در کاراکتر ها و فیچر آنها در قسمت های قبلی خروجی گرفته بودم، منتها در این بخش به نتایج زیر رسیدم:



همانطور که مشاهده میشود خروجی اول که کاملا اشتباه است و در خروجی های بعدی نیز با توجه به اینکه کد اسکی تمام حروف در دیکشنری من موجود نبود با ارور مواجه شدم. میتوان نتیجه گرفت که پیش بینی شبکه در این حالت حرفی بوده که اصلا داخل دیکشنری نبوده و پیش بینی های کاملا اشتباهی را به همراه داشته است.

با توجه به اینکه تعداد نورون ها در لایه ورودی 49 عدد و در لایه خروجی 63 عدد میباشد، دلیل این مقدار بی دقتی برای خود بنده نیز ناواضح است.