به نام خدا

**دانشگاه تهران**

**دانشکده مهندسی برق و کامپیوتر**

# شبکه های عصبی و یادگیری عمیق

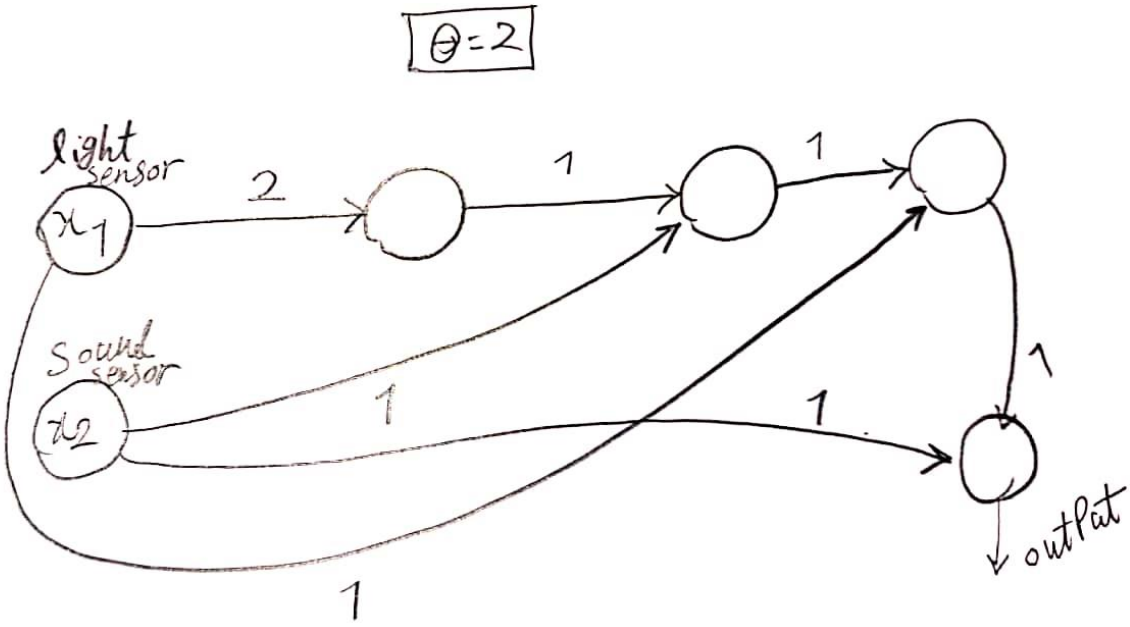
## تمرین سری اول

| | |
|---|---|
| نام و نام خانوادگی | امیر محمد رنجبر پازکی |
| شماره دانشجویی | ۸۱۰۱۹۹۳۴۰ |
| تاریخ ارسال گزارش | ۱۴۰۰/۱/۱۱ |

## فهرست گزارش سوالات

## سوال 1 – McCulloch-Pitts

**الف)** شکل این شبکه به صورت زیر است.




شکل ۱ - شبکه **McCulloch-Pitts** تشخیص دو بمب

در این ساختار هر دایره یک نورون McCulloch-Pitts است که یک یا دو ورودی دارد. اعداد روی یال‌ها وزن ورودی متناظر هستند. این شبکه انفجار دو بمب به صورت متوالی پس از دیده‌شدن نور و شنیده‌شدن صدای هر دو توسط فرد را گزارش می‌کند و در غیر این صورت، خروجی صفر می‌دهد.

**ب)** معادله منطقی این شبکه به صورت زیر است.

$$output = L_t \wedge S_{t+1} \wedge L_{t+2} \wedge S_{t+3}$$

عملگر ∧ عملگر AND منطقی است. L نمایشگر خروجی سنسور نور و S نمایانگر خروجی سنسور صداست. پایین‌نویس این نمادها نشانگر زمانی است که این خروجی تولید شده‌است.

**ج)** در این شبکه ابتدا سنسور نور در صورت مشاهده نور خروجی یک می‌دهد.(چپ بالا) حد آستانه فعال‌شدن هر نورون ۲ است. یعنی اگر مجموع ضرب ورودی‌ها یک نورون در وزن‌های متناظر بیش از ۲ باشد، آن نورون فعال می‌شود و خروجی یک تولید می‌کند. پس از تولید خروجی نور توسط نورون بالا سمت چپ، اگر در واحد زمانی بعد سنسور صدا صدایی را بشنود، ورودی صدا یک می‌شود و به نورون بالا وسط می‌رود و در این صورت، توسط نورون بالا وسط با خروجی سنسور نور در زمان قبل AND می‌شود و اگر هر دو یک باشند، یعنی بمب اول ترکیده‌است و خروجی این نورون یک می‌شود و در غیر این، صورت صفر می‌ماند. ورودی نورون بالا سمت چپ از طرفی خروجی این نورون است که نمایانگر ترکیدن بمب اول است و همچنین ورودی نور است که حساس به نور بمب دوم است و اگر در واحد زمانی سوم بمب بترکد

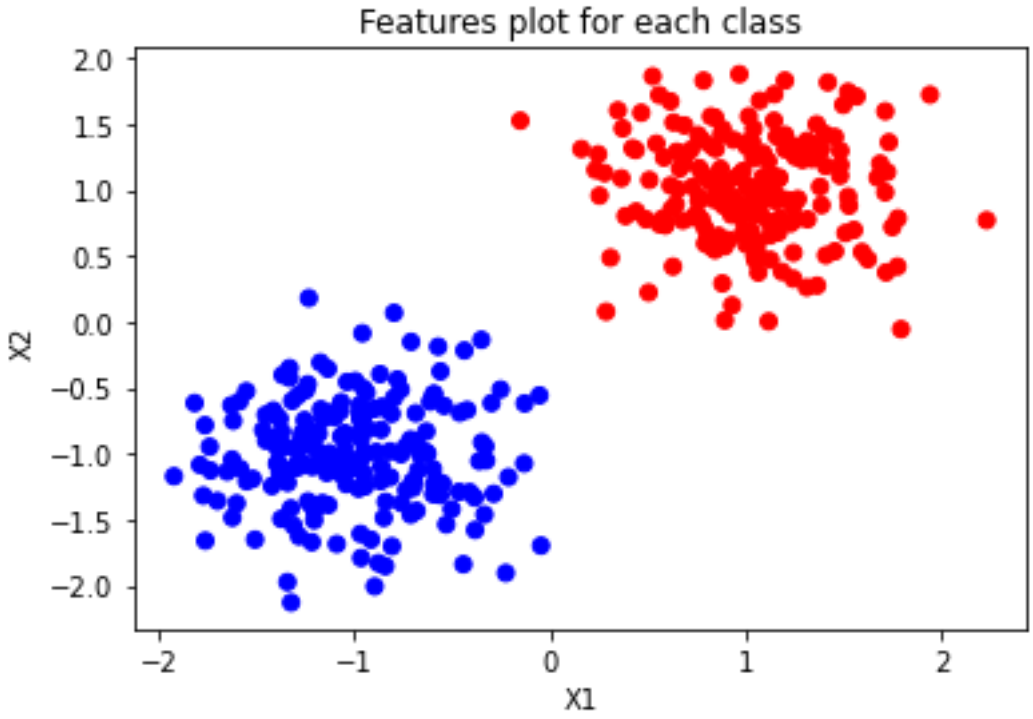
این ورودی نیز یک می‌شود و خروجی کل نورون یک می‌شود که حاکی از ترکیدن بمب اول و دیدن نور بمب دوم است.

در نهایت، خروجی این نورون به همراه ورودی صدا که منتظر صدای بمب دوم است به عنوان ورودی به نورون پایین سمت چپ یعنی همان نورون خروجی می‌روند. اگر تا به این جا ترکیدن بمب اول و نور بمب دوم خروجی نورون قبلی را یک کرده‌باشند و صدای بمب دوم نیز حس بشود، خروجی نورون آخر نیز یک می‌شود که به معنی تشخیص ترکیدن دو بمب متوالی در چهار گام زمانی است. اگر هر یک از اتفاقات بالا به ترتیب و به موقع رخ ندهد، خروجی صفر می‌ماند؛ یعنی، دو بمب متوالی نترکیده‌است.

## سوال ۲ – Perceptron

داده‌های مربوطه از فایل perceptron.csv با استفاده از کتابخانه pandas خوانده‌شد و سپس، ویژگی‌ها از برچسب‌های داده جدا شدند و با استفاده از تابع train_test_split کتابخانه sklearn به نسبت ۲۵ درصد به ۷۵ درصد به داده‌ی تست و آموزش تقسیم شدند.

**الف)** نمودار داده‌ها به صورت scatterplot با استفاده از scatter در کتابخانه matplotlib رسم شد که به صورت زیر است. محور عمودی ویژگی اول، محور افقی ویژگی دوم و رنگ نقطه‌ها نشانگر کلاس آن‌هاست.




شکل ۲- **scatterplot** داده‌ها به تفکیک کلاس

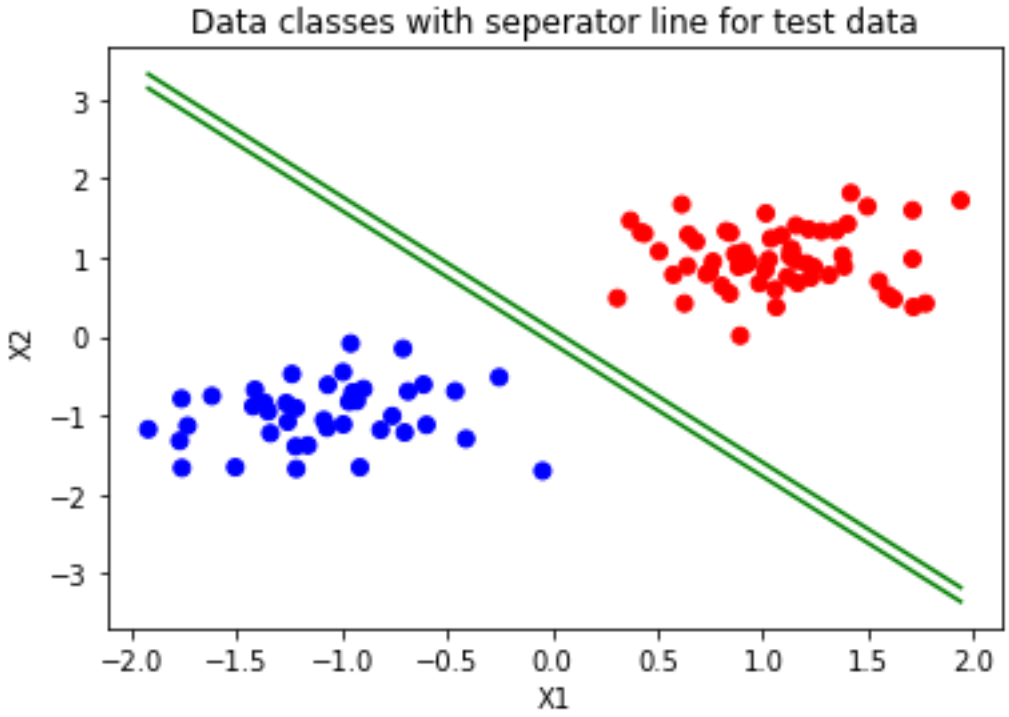
**ب)** یک کلاس Nueron زده‌شده‌است که پایه کلاس‌های دیگر است و توابع مشترک در آن است. کلاس PerceptronNueron برای این قسمت توسعه داده‌شده‌است که بخش‌های مورد نیاز و خاص این نوع از نورون‌است.

**ج)** میزان آستانه 0.001 است. نتایج برای داده‌ی تست به صورت زیر است.

تعداد epoch: ۱

دقت: ۱۰۰ درصد

نمودار دو خطی ایجادشده به همراه داده‌های تست به صورت زیر است.




شکل ۳- نمودار داده‌ها با دو خطی جداکننده با آستانه ۰.۰۰۱

**د)** دو قسمت قبلی با استفاده از دو آستانه‌ی ۰.۱ و ۰.۵ مجددا انجام شد که نتایج آن به صورت زیر است.

- آستانه: ۰.۱

تعداد epoch: ۱۹

دقت: ۱۰۰ درصد

نمودار:


شکل ۴- دو خط جدید با آستانه ۰.۱

- آستانه: ۰.۵
- تعداد epoch: ۱۰۰
- دقت: ۱۰۰ درصد
- نمودار:


شکل ۵- دو خط جدید با آستانه ۰.۵

همانطور که مشاهده‌می‌شود، با افزایش میزان آستانه دو خطی جداکننده از هم بیشتر فاصله می‌گیرد. در حقیقت، می‌توان گفت میزان آستانه فاصله دو خطی را مشخص می‌کند. این افزایش عرض دو خطی باعث می‌شود که شبکه robust تر شود و در مقابل خطا مقاوم تر شود و داده‌های میانه را به راحتی label نزند. البته داده‌های این محدوده بدون تصمیم می‌مانند.

هر چه میزان آستانه بیشتر شود، شبکه سعی می‌کند فاصله بیشتری بین دو دسته داده بیاندازد و خط وسط بیشترین فاصله از دو کلاس را داشته‌باشد که به همین دلیل، تعداد epoch بیشتری برای آموزش نیاز دارد. همانطور که مشاهده می‌کنید از حدی بزرگ‌تر شدن آستانه تفاوتی ایجاد نمی‌کند چراکه اگر خطها بیشتر از هم فاصله بگیرند، دقت شبکه پایین می‌آید و بنابراین، افزایش آستانه تاثیری ندارد. این مورد در افزایش از ۰.۱ به ۰.۵ مشاهده‌می‌شود.

نکته جالب این است که در صورت صفر کردن آستانه دو خط به هم می‌چسبند، چرا که فاصله این دو خط ۲ برابر آستانه است و در این صورت robustness شبکه پایین می‌آید و خط مرز تعیین دو کلاس از هم می‌شود.


شکل ۶- دو خط جدید با آستانه صفر

## سوال 3 – AdaLine

**الف)** از تفاوت‌های شبکه‌ی عصبی AdaLine و پرسپترون خطی می‌توان به موارد زیر اشاره کرد.

+ در آدلاین تابع فعالسازی دو ضابطه‌ای شده‌است و به همین دلیل، یک خط بین دو کلاس می‌کشد؛ بر خلاف شبکه‌ی پرسپترون خطی که دو خطی بین دو کلاس ایجاد می‌کند که این به دلیل تابع فعالسازی سه ضابطه‌ای است.

+ قاعده‌ی به روزرسانی این دو شبکه با هم تفاوت اساسی دارد که در زیر قابل مشاهده‌است. در آدلاین تلاش بر این است که مقدار net به target نزدیک‌شود. در صورتی که در پرسپترون این‌گونه نیست.

$$w_i(new) = w_i(old) + \alpha x_i t$$

$$b(new) = b(old) + \alpha \cdot t$$

روابط به‌روز رسانی مربوط به perceptron

$$w_i^+ = w_i^- + \alpha(t - net)x_i$$

$$b^+ = b^- + \alpha(t - net)$$

روابط به‌روز رسانی مربوط به AdaLine

+ بر خلاف پرسپترون، در آدلاین خبری از آستانه نیست و آستانه صفر در نظر گرفته‌شده‌است.

شباهت اصلی این دو شبکه مربوط به معماری آن‌هاست که بسیار مشابه هم است. هر دو دارای معماری بسیار ساده‌هستند و صرفا تفاوت آن‌ها در موارد بالاست.

**ب)** برای این بخش ابتدا داده‌های موردنظر با استفاده از تابع normal در کتابخانه np.random به تعداد موردنظر و با ویژگی‌های گفته‌شده تولید شدند.

سپس، یک نورون آدلاینی در کلاس AdaLineNueron تعریف شد. این کلاس ویژگی‌ها و توابع اصلی خود را از کلاس Nueron به ارث برده‌است. توابع متفاوت و مختص این کلاس در خود این کلاس پیاده‌سازی شده‌اند.

در این الگوریتم ابتدا وزن‌ها و مقدار bias را به یک مقدار تصادفی کوچک مقداردهی می‌کنیم. یک مقدار کوچک نیز برای learning rate انتخاب می‌کنیم. (0.01)

برای آموزش، حال به ازای هر زوج ورودی خروجی مراحل زیر را انجام می‌دهیم. (تابع train)

ورودی موردنظر را روی شبکه قرار می‌دهیم. مقدار جمع وزن‌دار ورودی‌ها(net) را محاسبه می‌کنیم. وزن‌ها و مقدار bias را طبق معادله‌ی زیر به‌روزرسانی می‌کنیم.

$$w_i^+ = w_i^- + \alpha(t - net)x_i$$

$$b^+ = b^- + \alpha(t - net)$$

روابط به‌روز رسانی مربوط به AdaLine

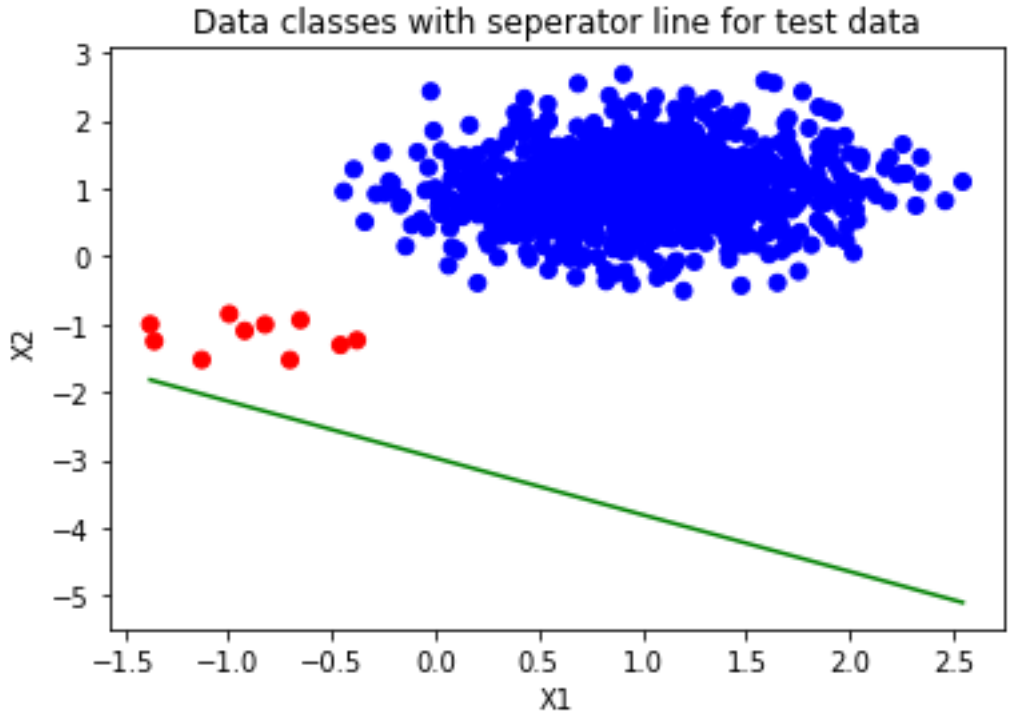
یک تابع هزینه به صورت نصف مجموع مربعات خطا(فاصله net با target) تعریف می‌شود که سعی این شبکه کمینه‌کردن این تابع هزینه است.در حقیقت، این شبکه سعی دارد به ازای هر داده net شبکه را به متغیر هدف نزدیک کند. اگر این مقدار برای همه داده‌ها کمتر مرز خطا باشد، متوقف می‌شویم و در غیر این صورت، به آموزش ادامه می‌دهیم. (اسم توابع متناظر مراحل است و گویاست.)

خروجی برای حالت یک به صورت زیر است.

تعداد epoch: ۱۰۰ (متوقف نشده‌است.)

دقت: ۹۹.۰۰۹ درصد

نمودار:



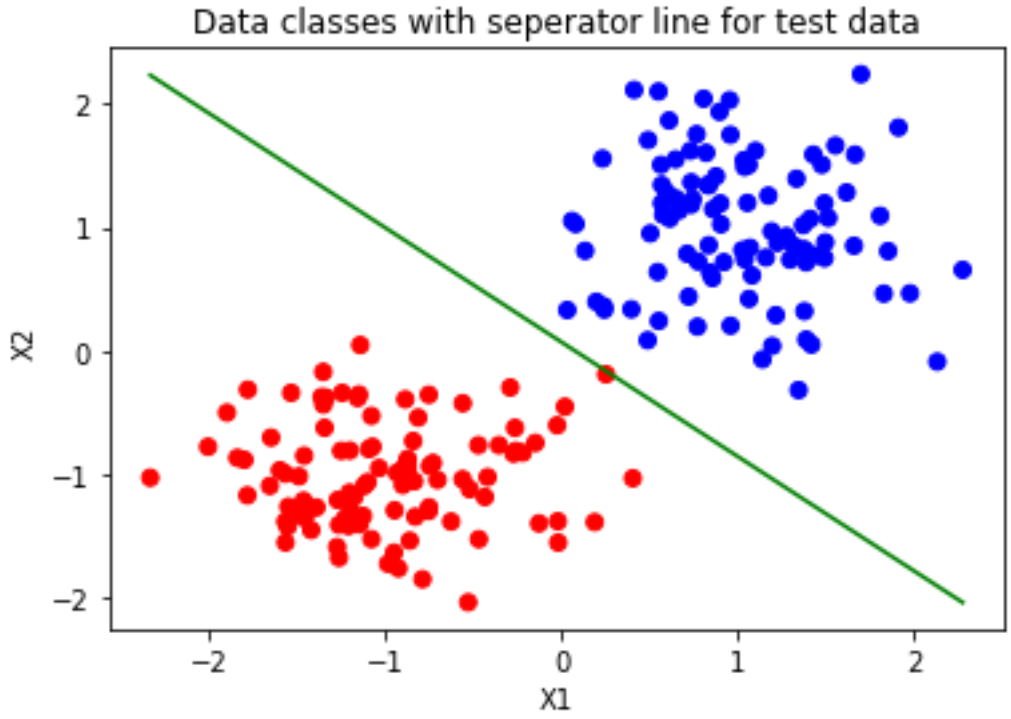

شکل ۷ - عملکرد شبکه در مورد یک

خروجی برای حالت دوم به صورت زیر است.

تعداد epoch: ۱۰۰

دقت: ۱۰۰ درصد

نمودار:




شکل ۷ - عملکرد شبکه در مورد دو

**ج)** خیر. هیچ تضمینی وجود ندارد. همانطور که می‌بینید هم اکنون نیز این دو کلاس را از یکدیگر درست جدا نکرده‌است. دلیل این موضوع عدم توازن تعداد داده‌هاست. این موضوع باعث شده‌است تا داده‌ها با تعداد بیشتر اهمیت بیشتری در طبقه‌بندی پیدا کنند. البته مشکل اساسی مربوط به این شبکه‌است چراکه سعی دارد تا **net** را به **target** نزدیک کند که این برای ما مهم نیست. مهم این است که خروجی شبکه به **target** نزدیک شود. به همین دلیل، این شبکه در مواردی که داده در دو کلاس توزیع مشابهی دارد، عملکرد خوبی دارد و در غیر این صورت، همانطور که مشاهده شد عملکرد مناسب ندارد.

برای اصلاح این مشکل باید تابع sign به عنوان تابع فعالساز را با یک تابع نرم مانند tanh جایگزین کرد. بر خلاف تابع sign این تابع نرم و مشتق پذیر است و روش سیستماتیک gradient descent را می‌توان برای به‌روزرسانی وزن‌ها مورد استفاده قرار داد و در این حالت، خروجی شبکه را به target نزدیک کرد و نه net را. (با قرار دادن تفاضل خروجی شبکه و target در تابع فعالساز) با استفاده از این کار مستقل از شکل و توزیع کلاس‌ها، به یک خط با بیشترین فاصله در بین دو کلاس(robustness کافی) می‌رسیم.

**د)** با انتخاب نرخ یادگیری بالا ممکن است از روی کمینه مورد نظر بپریم یا همگرایی در انتها پدید نیاید و تناوب کنیم. البته نرخ یادگیری بالا امکان فرار از کمینه‌های محلی را برای ما فراهم می‌کند. نرخ یادگیری خیلی کم سرعت یادگیری را بسیار پایین می‌آورد و امکان فرار از کمینه‌های محلی را ندارد. اما به سمت نقطه جذب محلی خود به خوبی پیش می‌رود. به دلایل گفته‌شده، نرخ یادگیری نباید چندان کم یا خیلی بزرگ باشد و باید مقدار میانه‌ی کوچکی داشته‌باشد.(0.01)

## سوال 4 – MAdaLine

**الف و ب)** در این قسمت با استفاده از منطق AdaLine یک کلاس MAdaLine نوشته‌شده‌است که این کلاس پیاده‌سازی موردنظر برای MAdaLine را در خود دارد. برای یادگیری MAdaLine از الگوریتم MRI که در کتاب آمده‌است استفاده شده‌است. این الگوریتم فقط وزن‌های لایه‌ی مخفی را یاد می‌گیرد. در این الگوریتم مشابه حالت قبلی خروجی هر یک از نورون‌های آدلاین لایه مخفی را حساب می‌کنیم. سپس، این خروجی‌ها را با وزن 1/n (n: تعداد نورون‌های لایه‌ی مخفی) به نورون خروجی متصل می‌کنیم. مقدار بایاس برای این نورون برابر n-1/n است. این مقادیر به این معناست که در صورتی که همه ۱- باشند خروجی شبکه ۱- است و این منطق OR است. به این صورت هر آدلاین یک خط می‌سازد و با OR کردن این خطها داخل خطوط ۱- می‌دهد و به محض خروج از هر خط(مثبت شدن آدلاین موردنظر) خروجی ۱+ می‌شود و کلاس دیگر مشخص می‌شود. به این صورت این پترن که در داخل یک چندضلعی محدب است، می‌تواند طبقه‌بندی شود. پس از حساب کردن خروجی شبکه در صورتی خروجی با مقدار هدف متفاوت باشد، عملیات به‌روزرسانی روی برخی وزن‌ها اعمال می‌شود. در غیر این صورت، به‌روزرسانی انجام نمی‌شود. اگر هدف ۱ باشد، به‌روزرسانی صرفا روی نورون با کوچک‌ترین net انجام می‌شود. اگر هدف ۱- باشد، روی همه نورون‌های مثبت انجام می‌شود تا خروجی شبکه به مقدار هدف نزدیک شود. پس از یادگیری، خطهای موردنظر را می‌کشیم. هر خط متعلق به یک AdaLine است.

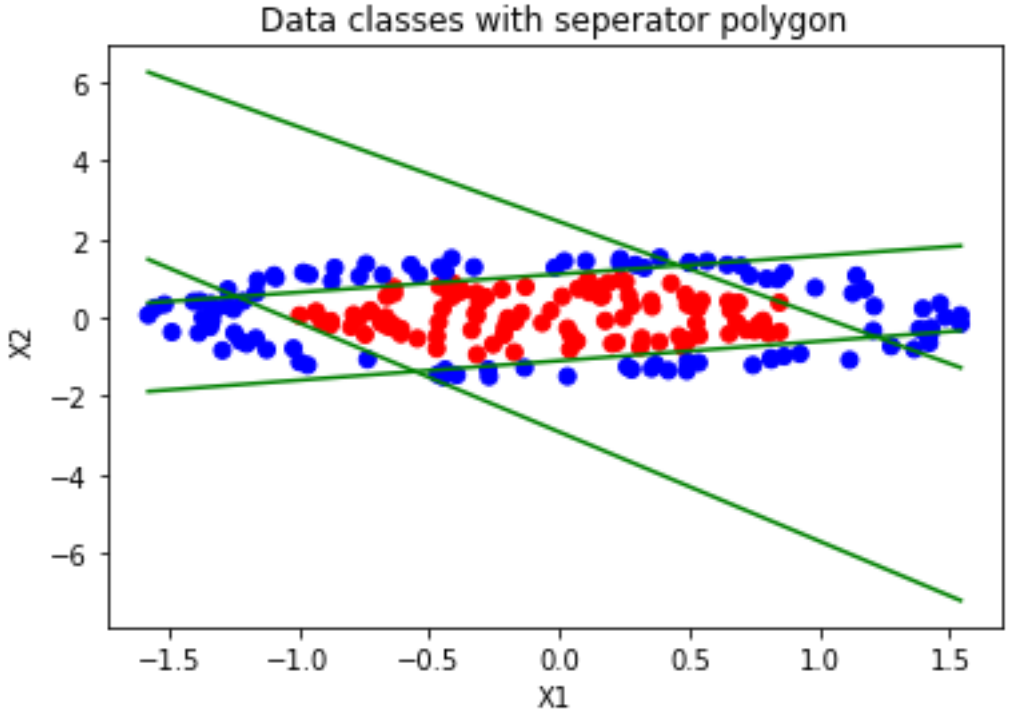
**ج)** نمودارها، تعداد epochها و دقت در سه حالت گفته‌شده به صورت زیر به‌دست آمد.

- ۴ نورون لایه‌ی مخفی:

دقت: ۱۰۰ درصد

تعداد epoch: ۲۷

نمودار:



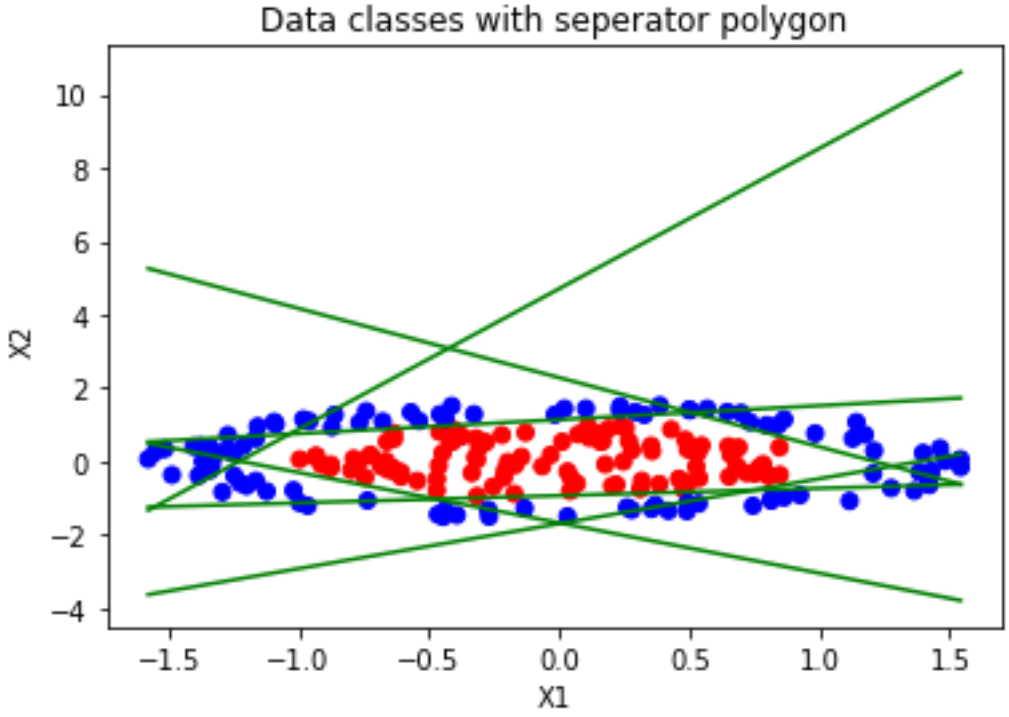

شکل ۸ ـ شبکه با ۴ نورون

- ۶ نورون لایه‌ی مخفی:

دقت: ۱۰۰ درصد

تعداد epoch: ۴

نمودار:



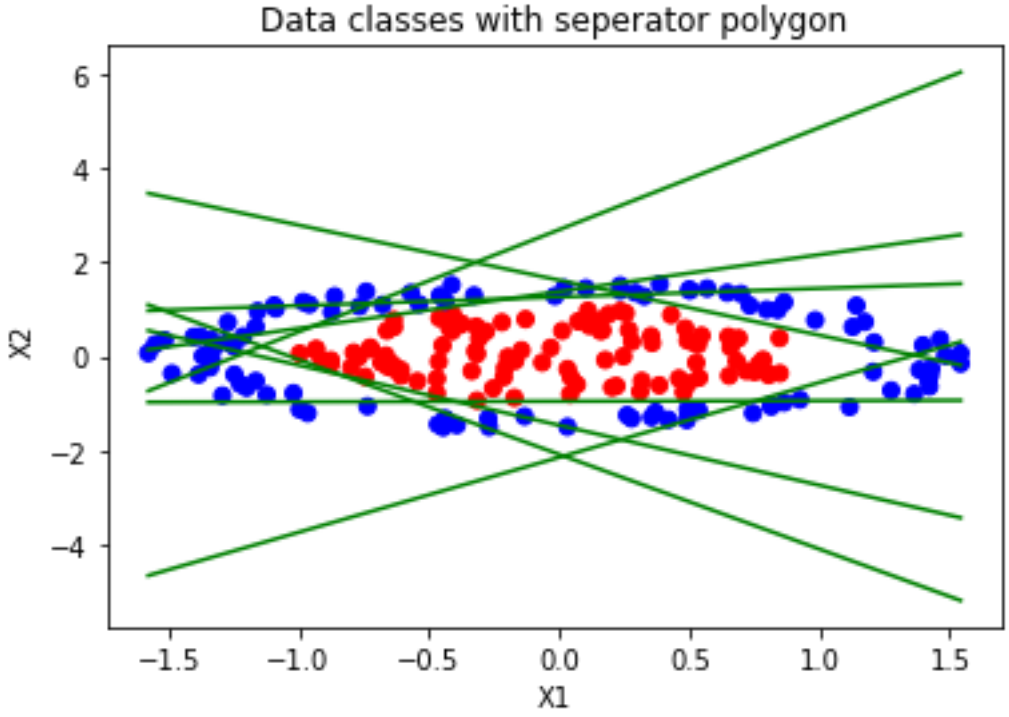

شکل ۹ - شبکه با ۶ نورون

- ۸ نورون لایه‌ی مخفی:

دقت: ۱۰۰ درصد

تعداد epoch: ۵

نمودار:




شکل ۱۰ ـ شبکه با ۸ نورون

همانطور که مشاهده می‌کنید، ۴ خط برای جداسازی این داده کافی‌است و به همین دلیل، افزایش تعداد خطوط کمکی به دقت ما نمی‌کند و همان دقت بیشینه را می‌دهد.

تعداد epochها با افزایش تعداد خطوط کاهش می‌یابد و دلیل آن این است که خطهای بیشتری هستند که می‌توانند این شکل را محصور کنند و هر کدام در حالت نزدیک شدن به جواب هستند و به همین دلیل، سریع‌تر این اتفاق رخ می‌دهد.

با مشاهده‌ی نمودارها می‌توان به این نتیجه رسید که با افزایش تعداد نورون‌ها و خطوط می‌توان به شکل نرم‌تری برای مرز رسید و این می‌تواند میزان robustness این شبکه را بالاتر ببرد.