به نام خداوند جان و خرد

نیمسال دوم 1402-1401

استاد درس: دکتر منصور رزقی آهق

دانشجو: بهار مهدوی
40152521337

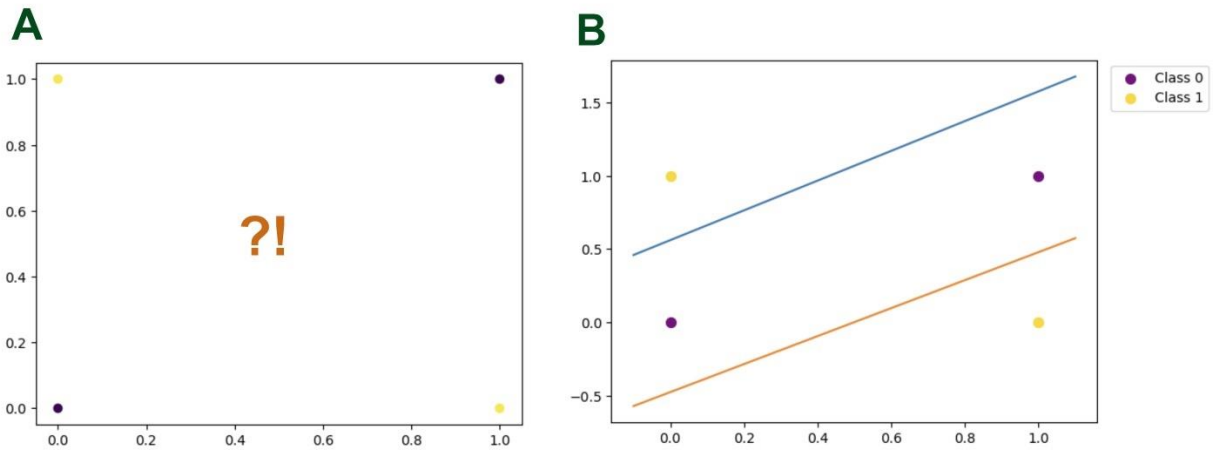
Deep Learning
Homework-1

Tarbiat Modares University

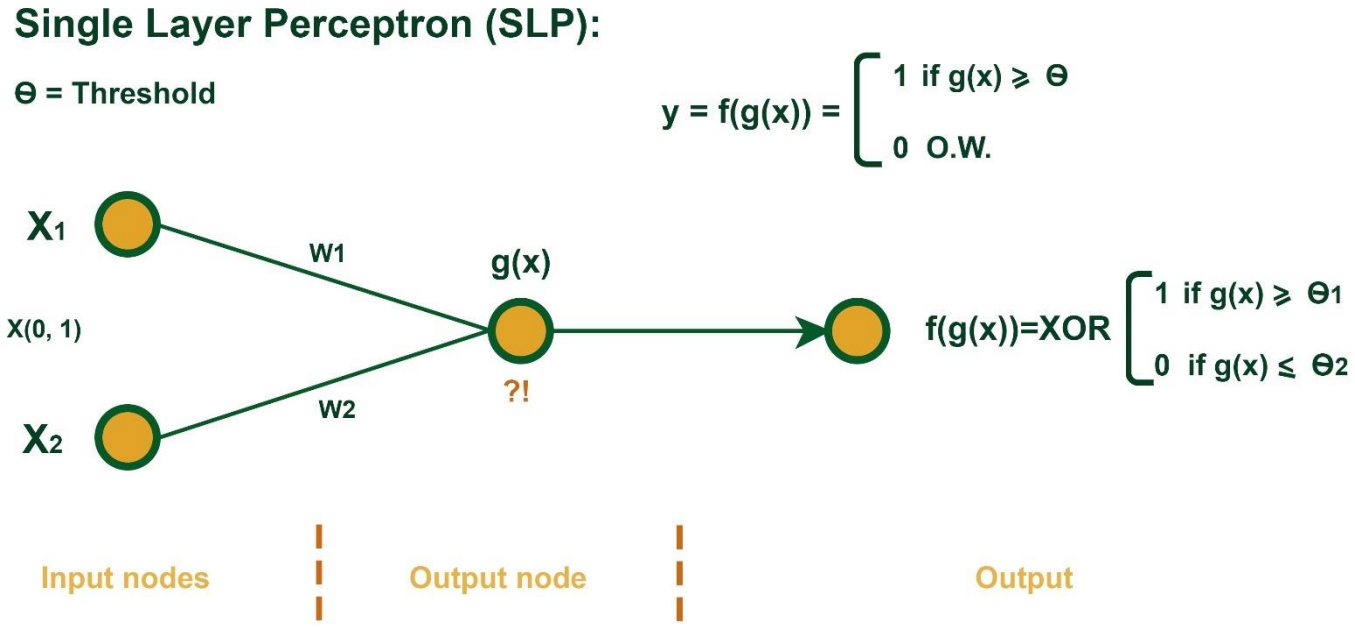
تمرین 1:

الف) نشان دهید تابع XOR بدون لایه پنهان پرسپترون قابل شبیه سازی نیست.

Marvin Lee Minsky، بنیانگذار آزمایشگاه هوش مصنوعی MIT، در سال 1969 تردید خود را در قالب تحلیل دقیق محدودیت های Perceptron اولیه، منتشر کرد. او نتیجه گرفت که رویکرد Perceptron در هوش مصنوعی یک بن بست است. مهمترین عنصر مورد بحث در این تحلیل، روشن کردن حدود یک Perceptron بدون لایه پنهان است. برای مثال، آنها نمی‌توانند یاد بگیرند تا تابع ساده boolean function XOR را پیشبینی کنند، زیرا عملگرهای XOR به صورت خطی جداپذیر نیستند و Perceptron منفرد (بدون لایه های پنهان و تابع فعال سازی) تنها می‌تواند داده های linearly separable را تفکیک کنند. به این ترتیب با اجرای Perceptron بدون لایه پنهان بر روی عملگر XOR (قسمت A تصویر زیر) تابع در یک loop تا بینهایت بار می‌گردد و هرگز به جواب نمیرسد و جهت شبیه سازی تابع XOR، مطابق قسمت B تصویر، به دو خط برای جداسازی 2 کلاس 0 و 1 آن نیاز خواهد بود که با اعمال شبکه های عصبی پیچیده تر غیر خطی مثل Perceptron چند لایه (مثلا لایه ای با تابع فعال سازی غیرخطی sigmoid) قابل اجرا است.



به این ترتیب برای پیدا کردن θ در ضابطه Perceptron برای حل تابع XOR نیاز است تا مطابق جدول رسم شده در قسمت ب سوال، g(x) ی را پیدا کنیم که بتواند بر روی ترکیب خطی X1 و X2 با وزن یک اعمال شود و نتیجه X1 XOR X2 را دهد، ولی با استفاده از روش SLP، هیچ θ ی را نمی‌توان در نظر گرفت تا این شبکه رو شبیه سازی کند و برای اینکار نیاز است تا g(x) بزرگتر مساوی θ و کوچکتر مساوی θ را بدست بیاوریم که شامل دو خط مجزا می‌شود:

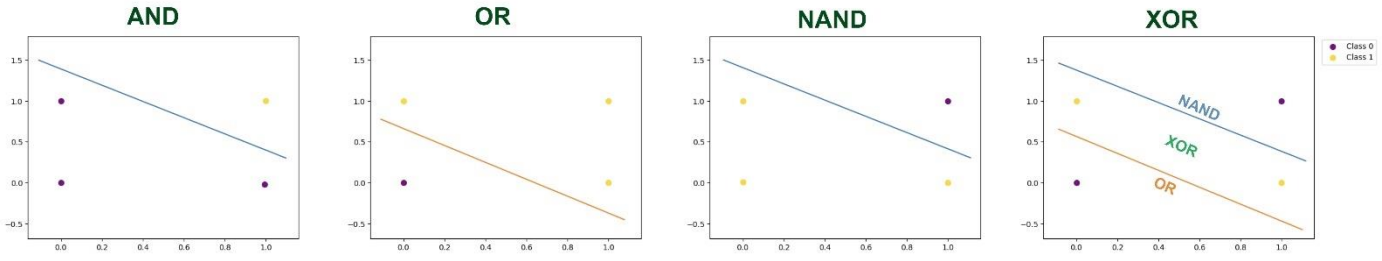


ب) XOR را با یک MLP با کمترین لایه شبیه سازی کنید.

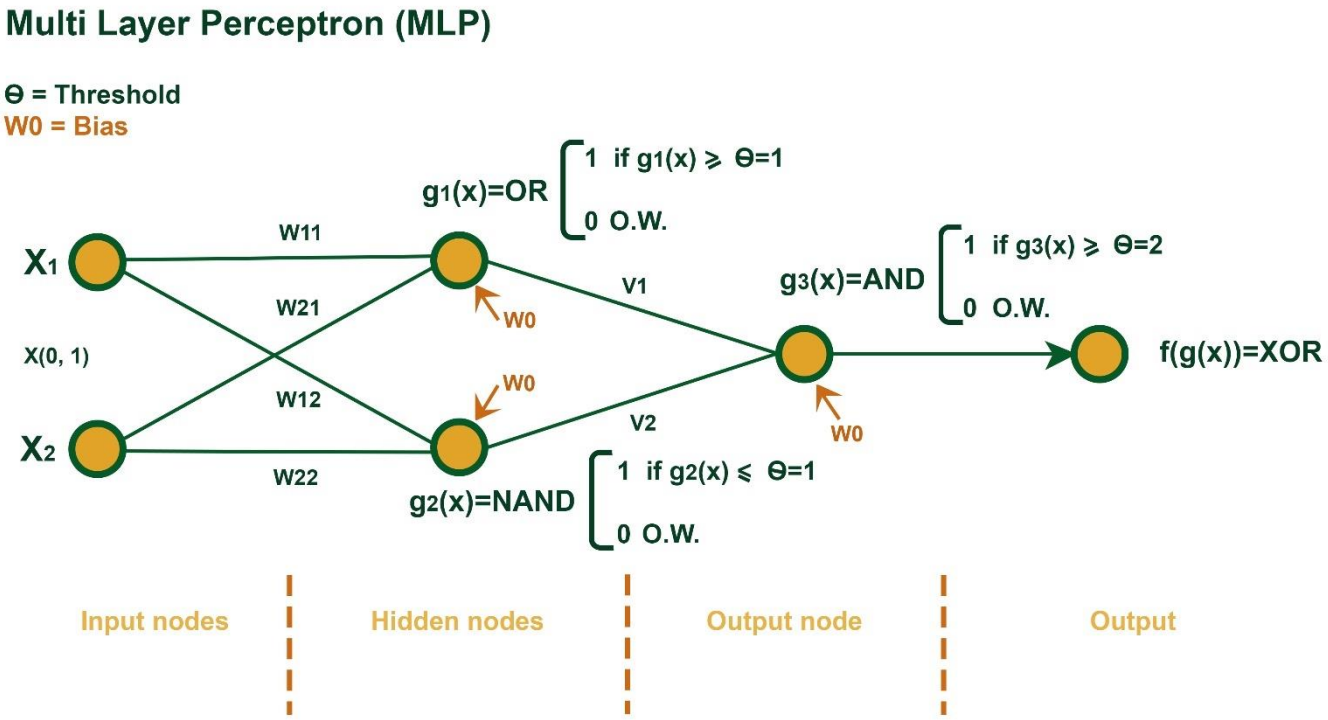
XOR حاصل AND دو عملگر OR و NAND است که جدول حقیقت این boolean function ها به شرح زیر است:

X_1	X_2	$g(x)=X_1, X_2$ Com.	X_1 AND X_2	X_1 OR X_2	X_1 NAND X_2	X_1 XOR X_2
T	T	2	1	1	0	0
T	F	1	0	1	1	1
F	T	1	1	1	1	1
F	F	0	0	0	1	0
T=1	F=0					

این عملگرها در فضای دو بعدی به صورت زیر نمایش داده می‌شوند که همانطور که مشخص است عملگرهای AND، OR و NAND به صورت خطی جداپذیر هستند ولی همانطور که در قسمت قبل توضیح داده شد XOR linearly separable نیست.



برای شبیه سازی تابع XOR می‌توان از شبکه رسم شده در زیر استفاده نمود که حاصل ترکیب دو عملگر OR و NAND بصورت AND است که خروجی XOR را ایجاد می‌کند:



در این شبکه $g_1(x)$، $g_2(x)$ و $g_3(x)$، توابع فعال ساز یا Active function های ما هستند که به ترتیب نمایانگر توابع OR، NAND و AND اند که به عنوان hidden layer شبکه MLP ما بکار می‌روند. این لایه ها برای عبور داده ها مجموع Weightها (W) و Bias ها (W0) را در هر پرسپترون ایجاد می کنند. با در نظر گرفتن θ صحیح شبکه تابع ما را محاسبه خواهد کرد.

در بخش پیاده سازی تمرین، از Active function سیگموئید به عنوان hidden layer استفاده گردید. برای آموزش مدل، نمونه های آموزشی زیادی به شبکه منتقل می شود. شبکه با Weightها و Bias های تصادفی راه‌اندازی می‌شود. پس از هر آموزشی که از طریق شبکه عبور داده می شود، مقداری را خروجی می دهد که پیش بینی آن برای خروجی (y) با توجه به ورودی (x) است. این پیش بینی با مقدار واقعی 1 یا 0 مقایسه می شود. مقایسه بین خروجی پیش‌بینی شده و واقعی در تابع هزینه انجام می‌شود. صرفا جهت مقایسه و تمرین در پیاده سازی از دو تابع هزینه MSE و BCE به طور جداگانه برای این کار استفاده گردید. پس از محاسبه Cost / Loss function، Weight ها و Bias ها با استفاده از گرادیان (SGD) تابع هزینه به روز می شوند تا به local یا global minimum نزدیک تر شوند. هدف از بهینه سازی به حداقل رساندن تابع هزینه است. وقتی تابع هزینه صفر باشد، به این معنی است که تمام مقادیر پیش بینی شده با مقادیر واقعی یکسان هستند و هر چه مقدار آن به صفر نزدیکتر باشد به این معنی است که شبکه ما بهتر عمل کرده است.

کد مرتبط با تمرین 1:

- DL-HW1-Q1-Mahdavi.ipynb

تمرین 2:

روش کلاس بندی لاجستیک رگرسیون برای حالت چندکلاسه را با یک شبکه نمایش دهید و جزییات آنرا شرح دهید.

لاجستیک رگرسیون چند کلاسه به صورت زیر شرح داده می‌شود:

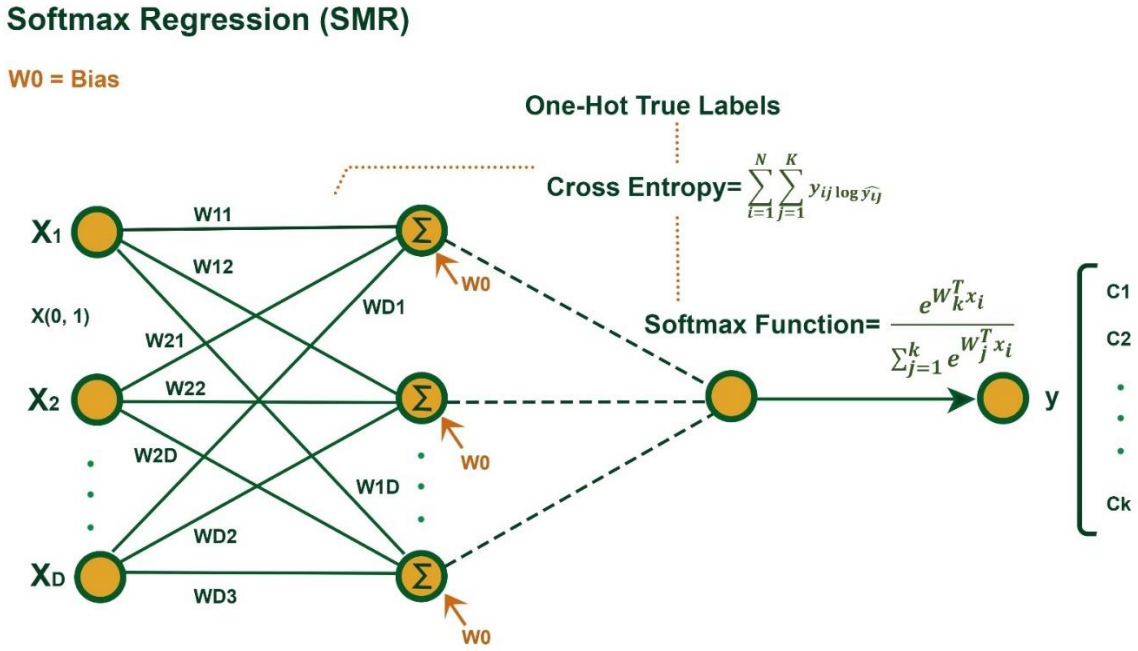
$$max\ loss(y, \hat{y}) = \prod_j \hat{y}_j^{y_j} = \sum_j y_j \ln \hat{y}_j$$

$$max \sum_i loss(y_i, \hat{y}_i) = \sum_i \sum_j y_{ij} \ln \hat{y}_{ij}$$

که معادل Cross Entropy است:

$$\text{Cross Entropy} = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \log \hat{y_{ij}}$$

اگر لاجستیک رگرسیون دو کلاسه داشتیم میتوانستیم از تابع فعال‌سازی sigmoid استفاده کنیم ولی sigmoid برای لاجستیک رگرسیون چند کلاسه جواب خوبی نمیدهد به این ترتیب برای این چنین مسائل از تابع فعال‌سازی Softmax استفاده میگردد که در آن مقادیر تابع هدف با کمک One-Hot True Labels به شکل برداری در می آیند و با یک loss function Cross Entropy آموزش میبینند و یک نوع رابطه غیر خطی از لاجستیک رگرسیون چند جمله ای را ارائه میدهند:



Softmax regression (یا لاجستیک رگرسیون چند جمله‌ای) تعمیم لاجستیک رگرسیون به مواردی است که می خواهیم k کلاس را در حالت چند بعدی مدیریت کنیم و از آن به عنوان آخرین تابع فعال‌سازی (Activation Function) برای شبکه عصبی مان جهت نرمال سازی خروجی شبکه و تبدیل آن به توزیع احتمال بهره می‌بریم. نرمال سازی در این حالت نسبت به کلاس‌های خروجی پیش بینی شده، صورت می‌گیرد. پس قبل از استفاده از تابع Softmax، بعضی از اجزای برداری ممکن است منفی یا بیشتر از یک باشند، اما بعد از استفاده از این تابع، هر مولفه در بازه صفر و یک قرار می‌گیرد بطوری که مجموع آن‌ها برابر با یک باشد.

این تابع با یک تحلیل آماری و به صورت زیر بدست می‌آید:

طبق قضیه Bayes برای xi اگر عضو کلاس ci باشد، شانس این کلاس به شرط رخداد آن به این صورت است:

$$P(ci \mid xi) = \frac{P(ci).P(xi \mid ci)}{P(xi)} = \frac{P(ci).P(xi \mid ci)}{\sum_{i=1}^{k} P(ci).P(xi \mid ci)}$$

برای این که مولفه ها در بازه صفر و یک قرار گیرند خواهیم داشت:

$$P(ci \mid xi) = \frac{e^{ln P(ci).P(xi \mid ci)}}{\sum_{i=1}^{k} e^{ln P(ci).P(xi \mid ci)}}$$

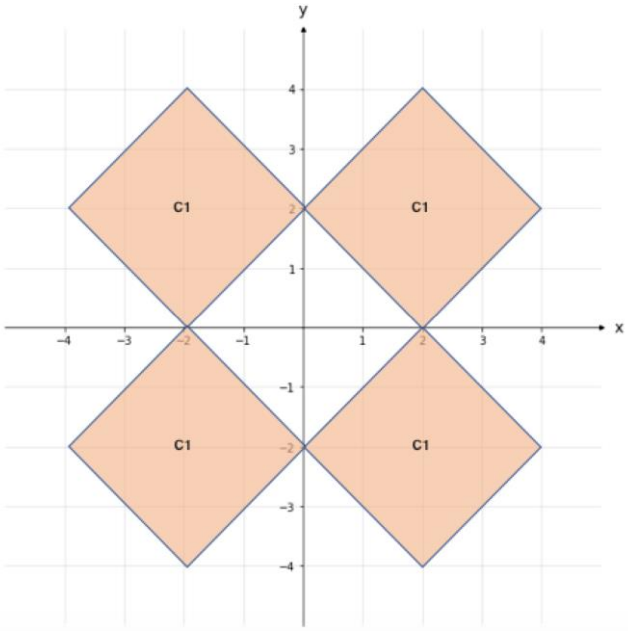
با تقریب زدن $ln\ P(ci).P(ci \mid xi)$ با $W_k^T x_i$ تابع Softmax به صورت زیر تعریف می‌شود:

$$softmax(x)_i = \frac{e^{W_k^T x_i}}{\sum_{j=1}^{k} e^{W_j^T x_i}}$$

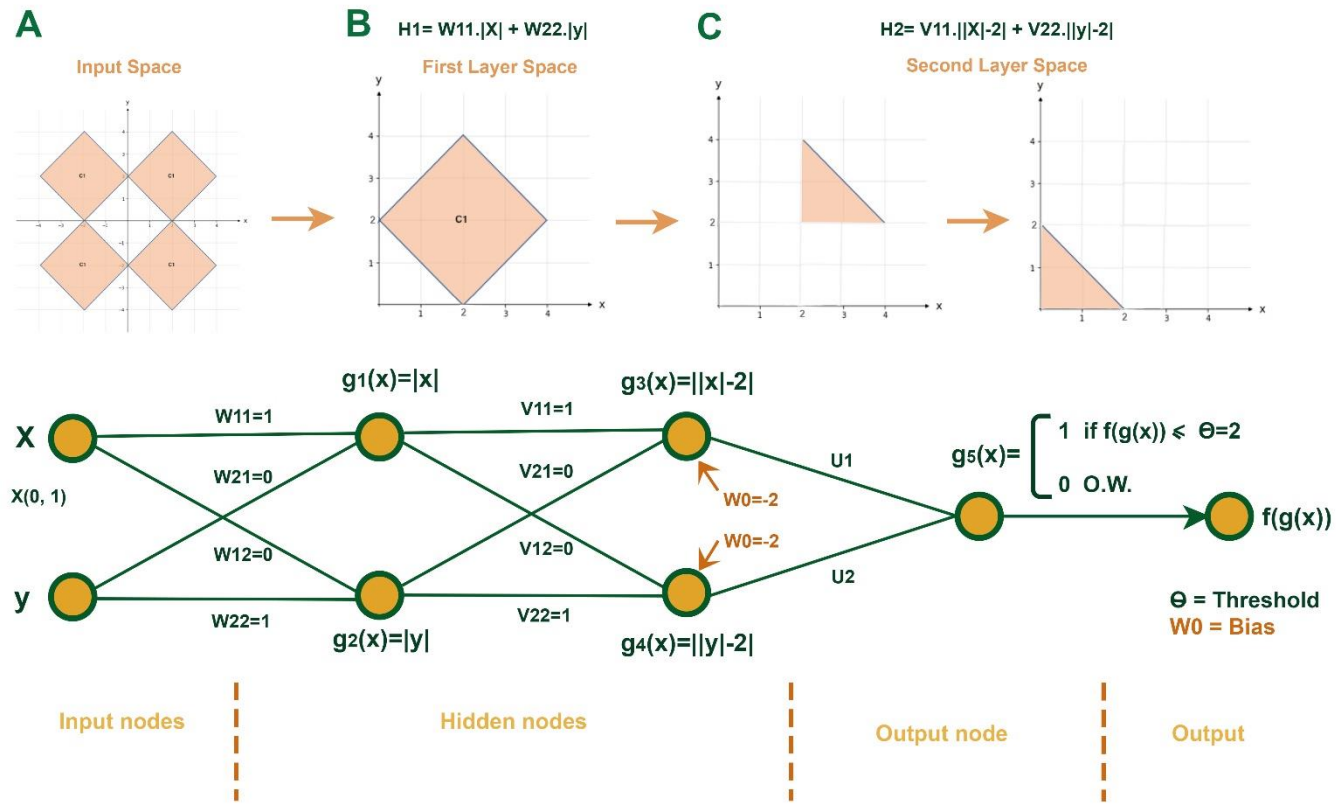
ایده پشت این روش به این صورت است که این مدل با دریافت یک نمونه، اول برای همه کلاس ها یک امتیاز رو محاسبه میکنه و بعد با دادن این امتیاز ها به تابع Softmax احتمال بودن در هر کلاس رو حدس میزنه. بعد از محاسبه امتیاز هر کلاس برای نمونه Xi ، میتوان احتمال تعلق Xi به کلاس k رو به دست آورد. پس این تابع اول exp همه امتیاز ها رو به دست میاره بعد با تقسیم این مقدار به جمع همه exp ها، آن را نرمال میکنه.

تمرین 3:

در شکل زیر کلاس C1 مشخص شده است. خارج از آن کلاس C2 است. با یک شبکه با کمترین لایه این دو کلاس را جدا کنید.



Absolute Value Active Function



برای جدا کردن دو کلاس C1 و C2 باید رویکردی را در نظر گرفت تا این تصویر را به تصویر ساده تری تبدیل کرد تا بتوان آن را بصورت خطی جداپذیر کرد. برای این منظور در ابتدا و در لایه پنهان اول، صفحه یک بارحول محور x و بار دیگر حول محور y تا میخورد تا نتیجتا یک ربع از تصویر که تصویر یک لوزی-مربع است بدست آید (قسمت B). این کار با استفاده از Activation Function قدر مطلق |X| و |y| انجام میگیرد که جزئیات نودها و وزن های ورودی در شکل نمایش داده شده است. در مرحله بعد و لایه پنهان دوم دوباره تصویر حول محورهای فرضی x و y مجددا با استفاده از توابع فعال‌سازی قدر مطلق |2-|X|| و |2-||y|| تا میخورد تا تصویر مثلث نمودار اول بخش C بدست آید و در نهایت با یک شیفت به مرکز محور x و y از طریق Bias منفی دو به فضای دوبعدی ای خواهیم رسید که به راحتی با یک خط می‌توان دو کلاس C1 (ناحیه رنگی) و C2 (خارج از ناحیه رنگی) را کلاسه بندی و از هم تفکیک کرد.