تمرین اول درس مبانی هوش محاسباتی

مبانی شبکه های عصبی

آرش حریرپوش
9731505

آرش حریرپوش
9731505

❖ بخش اول – مباحث تئوری و مسائل تشریحی

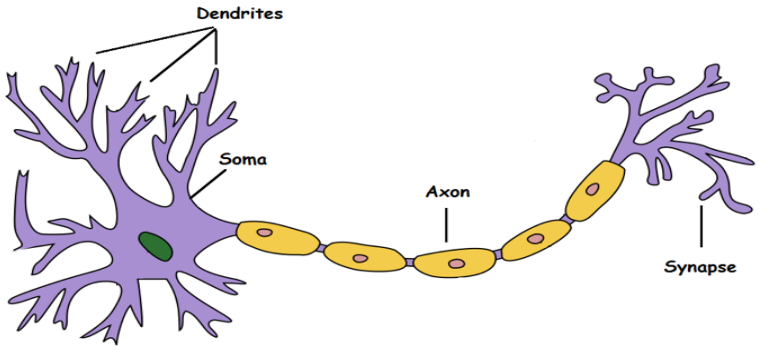
1. ساختار یک نورون (سلول) عصبی انسان همانند شکل 1 می باشد.

- نقش هر یک از قسمت های مشخص شده را بیان کنید و توضیح دهید هر کدام در یک نورون عصبی مصنوعی چگونه مدل سازی می شود.

در نورون عصبی اطلاعات از طریق دندریت ها (dendrites) وارد نورون می شوند، soma اطلاعات را پردازش می کند و از طریق axon آن را منتقل می کند، و سیناپس (synapse) نیز از با دریافت اطلاعات از آکسون (axon) فعالیت نورون های متصل را مهار (inhibition) یا تحریک (stimulation) می کند. در نورون های مصنوعی اطلاعات از طریق ورودی های (وزن گذاری شده، هر ورودی در وزن متناظر خود ضرب می شود، هر چه میزان یک وزن بالاتر باشد تاثیر یک واحد بر دیگری بیشتر می شود، همانند کاری است که سیناپس انجام می دهد و باعث برانگیختگی می شود.) وارد نورون می شوند (همانند دندریت). بدنه نورون مصنوعی ورودی های وزن گذاری شده و بایاس را جمع می کند و آن را از طریق تابع فعالیت (انتقال) پردازش می کند (همانند soma). در نهایت اطلاعات پردازش شده را از طریق خروجی خارج می کند (همانند آکسون).

- سیگنال های بین نورون ها از چه جنسی هستند و چگونه منتقل می شوند.

سیگنال های بین نورون ها از جنس تکانه های الکتریکی هستند و توسط آکسون منتقل می شوند؛ یک نورون عصبی پس از دریافت اطلاعات نورون های دیگر از طریق دندریت ها یک فعالیت الکتریکی را آغاز می کنند و از طریق آکسون به هزاران شاخه می فرستد. در انتهای هر شاخه یک ساختار به نام سیناپس وجود دارد که فعالیت های دریافتی از آکسون را به اثرات الکتریکی تبدیل می کند که فعالیت در نورون های متصل را مهار (inhibition) یا تحریک (stimulation) می کند.



شکل1- ساختار و بخش های یک نورون عصبی

آرش حریرپوش
9731505

2. در شبکه پرسپترون چندلایه : (در تمامی موارد پاسخ خود را توضیح دهید.)

- یکی از پارامتر های موجود، نرخ یادگیری می باشد. مقدار این پارامتر چه تاثیری در آموزش شبکه دارد؟ به نظر شما نحوه مقدار دهی مناسب این پارامتر چگونه باید باشد؟

نرخ یادگیری اندازه گام های یادگیری را مشخص می کند؛ بنابراین هر چه مقدار آن را کوچک تر تنظیم کنیم، شبکه دیرتر یاد می گیرد و به گام های بیشتری برای حل مسئله نیاز دارد اما نتیجه دقیق تر می شود و هرچه میزان این پارامتر را بزرگ تر کنیم مسئله زودتر حل می شود اما پس از مدتی plateau می شود، یعنی دقت مسئله از یک مقداری پایین تر نمی رود و تغییر دقت در مسئله تقریبا خط صاف می شود چراکه ممکن است گام آخر مورد نیاز برای مسئله از گام های فعلی کوچکتر باشد و ما با این اندازه گام ها نتوانیم به نتیجه مطلوب نهایی برسیم، و دقت نتایج آن نسبت به حالت قبل کاهش می یابد.
با توجه به مطالب عنوان شده بهتر است ابتدا نرخ یادگیری را بزرگ بگذاریم تا با گام های بلندتری یادگیری را انجام دهیم و زمانی که به گام های آخر و حل مسئله رسیدیم مقدار نرخ یادگیری را کاهش دهیم تا با دقت بیشتری به نقطه مطلوب برسیم.

- عمق یا تعداد لایه های این شبکه ها نیز هایپر پارامتری است که باید به آن توجه کرد. افزایش عمق شبکه چه تاثیری دارد؟

افزایش عمق شبکه باعث افزودن ویژگی غیر خطی می شود، هر چه عمق یک مسئله را بیشتر کنیم خطوط غیر خطی بهتری می توانیم رسم کنیم، اما برای آموزش آن شبکه به داده بیشتری نیز نیاز خواهیم داشت.

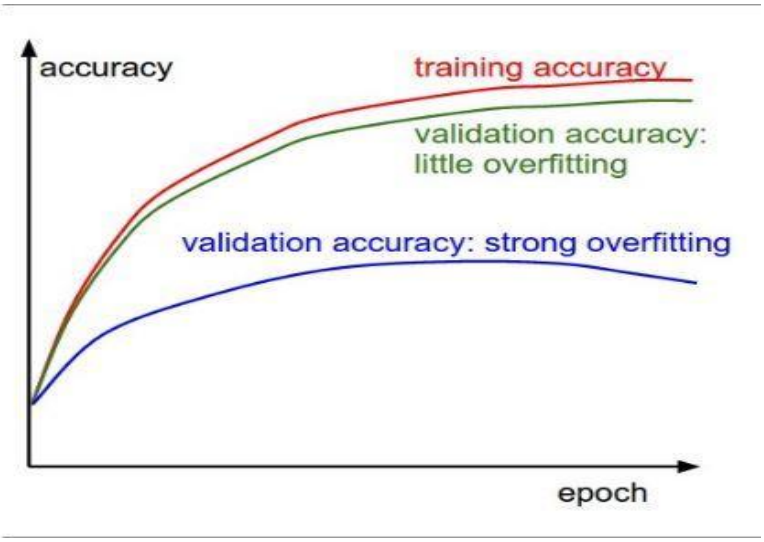
- تعداد نورون های یک لایه چه تاثیری در کارایی شبکه دارد؟

نورون های یک لایه برابر تعداد خطوطی است که در آن لایه قرار است رسم شود، بهتر است بجای اینکه تعداد نورون های یک لایه را خیلی زیاد کنیم عمق شبکه را بیشتر کنیم.

- افزایش تعداد لایه ها و تعداد نورون های هر لایه یا به طور کلی افزایش تعداد وزن های این شبکه می تواند مدل را دچار بیش برازش کند. به نظر شما چطور می توان متوجه شد که مدل آموزش دیده دچار بیش براز شده است؟ چه روشی برای جلوگیری بیش برازش در شبکه های عصبی عمیق وجود دارد و پیشنهاد می کنید؟

با تقسیم داده های موجود به سه دسته آموزش (train)، split و آزمایش (test) می توانیم متوجه نحوه عملکرد شبکه عصبی شویم، به این صورت که هنگامی که در حال آموزش داده های train هستیم پس از هر epoch دقت داده های split و train را بررسی کنیم اگر دقت در داده های train افزایش پیدا کند اما دقت در داده های split افزایش پیدا نکند و فاصله دقت بین این دو نمودار همانند شکل زیر زیاد شود (یا خطا در داده validation افزایش یابد اما در داده های test کاهش یابد)، یعنی بیش برازش رخ داده است.

آرش حریرپوش
9731505



برای جلوگیری از بیش برازش در شبکه های عصبی می توان کار های زیر را انجام داد:

1. Batch normalization
2. Drop out: در این روش بخشی از نورون های یک لایه را غیر فعال می کنیم و یادگیری را به کمک بقیه نورون ها انجام می دهیم.
3. افزایش داده ورودی: می توان داده بیشتری جمع آوری کرد که کار هزینه بری خواهد بود، وابسته به نوع داده می توان با انجام تغییراتی در داده های فعلی (data Augmentation) داده ها را افزایش داد.
4. محدود پیچیدگی شبکه عصبی: مزایای شبکه عصبی پیچیده این است که هر چه داده بیشتری به آن بدهیم عملکرد آن بهتر می شود. یک مدل با تعداد نزدیک به بینهایت مثال در نهایت plateau می شود با توجه به این که شبکه چقدر قدرت یادگیری دارد.
یک مدل می تواند دچار بیش برازش شود به این دلیل که توانایی مناسب آن را دارد.
کاهش پیچیدگی مدل احتمال بیش برازش مدل برای آن dataset را کاهش می دهد، تا نقطه ای که دیگر بیش برازش وجود نداشته باشد.
توانایی یک مدل شبکه عصبی، پیچیدگی آن، توسط ساختار مربوط به نود ها و لایه های آن و پارامتر های مربوط به وزن آن ها تعریف می شود. بنابراین، می توانیم از طریق یکی از روش های زیر پیچیدگی مدل را کاهش دهیم:
   1. تغییر پیچیدگی شبکه به وسیله تغییر ساختار شبکه (تعداد وزن ها).
   2. تغییر پیچیدگی شبکه به وسیله تغییر پارامتر های شبکه (مقدار وزن ها).

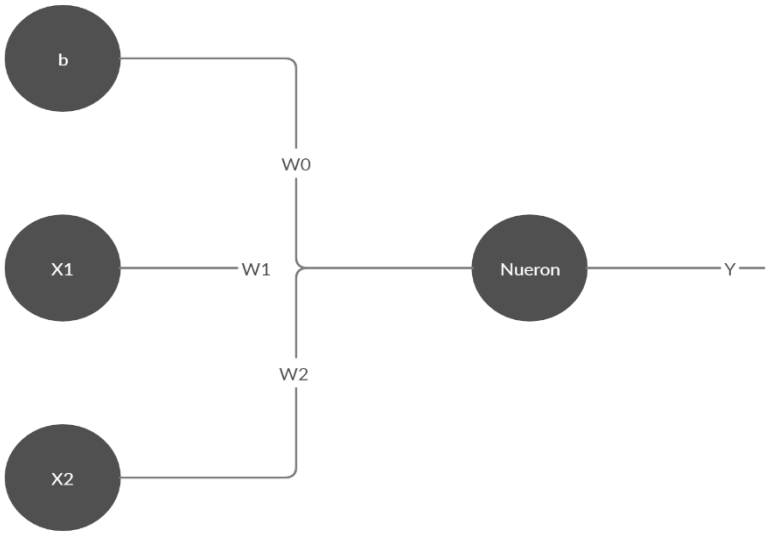
برای جلوگیری از بیش برازش معمولا drop out موثر عمل می کند.

(منبع: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/)

آرش حریرپوش
9731505

3. برای هر مورد زیر ، مدل پرسپترونی ارائه کنید که بتواند آن عملیات ر ا به عنوان تابع فعالیت مدل پیاده کند. در هر حالت ، وزن ها و بایاس ها را تعیین کرده و مرز تصمیم گیري را نمایش دهید.

همه این مدل ها دارای دو ورودی و یک بایاس می باشند که به صورت زیر نمایش داده می شود:



که در آن b برابر بایاس و X1 و X2 به ترتیب ورودی های نورون و W0، W1 وW2 به ترتیب وزن های مربوط به بایاس، ورودی اول و دوم و Y خروجی نورون می باشد ؛ سپس با توجه به جدول و رابطه زیر برای هر یک از این موارد وزن ها و بایاس را تنظیم می کنیم و مرز تصمیم گیری را مشخص می کنیم(به دلیل اینکه خروجی نورون ها باید یک یا صفر باشد تابع فعالیت برابر پله می باشد).

فرمول شماره 1:

$$W0 * b + X1 * W1 + X2 * W2$$

و از قوانین پرسپترون هم می دانیم:

فرمول شماره 2:

$$if\ W * X + b \leq 0\ then\ Y^* = 0$$

که در آن Y^* برابر خروجی نورون می باشد.

در دو مورد اول مقدار W0 برابر 1 در نظر گرفته شده و مقدار بایاس را فقط تغییر می دهیم.

- NOR

X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	0

آرش حریرپوش
9731505

با توجه به جدول بالا و فرمول 1 وزن های ورودی و بایاس را برای محاسبه NOR به صورت زیر محاسبه می کنیم:
ابتدامقادیر اولیه b و W1 و W2 را به ترتیب برابر 1- و 1 و1 قرار می دهیم:

$$-1 + X1 * 1 + X2 * 1 = -1 + X1 + X2$$

سپس مقادیر ردیف اول را در فرمول بالا قرار می دهیم:

$$-1 + 0 + 0 = -1$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای NOR برابر 1 می باشد.
بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 1 کند. برای این کار مقدار b را برابر 1 قرار می دهیم:

$$1 + 0 + 0 = 1$$

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف 1 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 2 را جایگذاری می کنیم:

$$1 + X1 + X2 = 1 + 0 + 1 = 2$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای NOR برابر 0 می باشد.
بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 0 کند. برای این کار مقدار W2 را برابر 1- قرار می دهیم:

$$1 + X1 - X2 = 1 + 0 - 1 = 0$$

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 3 را جایگذاری می کنیم:

$$1 + X1 - X2 = 1 + 1 + 0 = 2$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای NOR برابر 0 می باشد.
بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 0 کند. برای این کار مقدار W1 را برابر 1- قرار می دهیم:

$$1 - X1 - X2 = 1 - 1 + 0 = 0$$

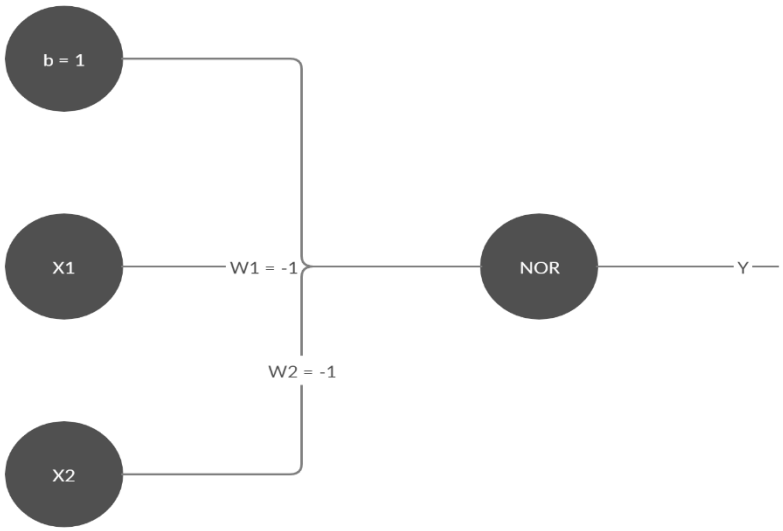
در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 و 3 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 4 را جایگذاری می کنیم:

$$1 - X1 - X2 = 1 - 1 - 1 = -1$$

با توجه به قانون شماره 2 مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف درست می باشد.

در نتیجه مقدار وزن ها و بایاس برای همه 4 ردیف ورودی درست می باشد؛ بنابراین نتیجه می گیریم که مدلی که گیت NOR را پیاده سازی کند به صورت زیر می باشد:

$$1 - X1 - X2$$



- NAND

X1	X2	Y
0	0	1
0	1	1
1	0	1
1	1	0

با توجه به جدول بالا و فرمول 1 وزن های ورودی و بایاس را برای محاسبه NAND به صورت زیر محاسبه می کنیم:

ابتدامقادیر اولیه b و W1 و W2 را به ترتیب برابر 1- و 1 و1 قرار می دهیم:

$$-1 + X1 * 1 + X2 * 1 = -1 + X1 + X2$$

سپس مقادیر ردیف اول را در فرمول بالا قرار می دهیم:

$$-1 + 0 + 0 = -1$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای NAND برابر 1 می باشد.

بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 1 کند. برای این کار مقدار b را برابر 1 قرار می دهیم:

$$1 + X1 + X2 = 1 + 0 + 0 = 1$$

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف 1 درست است.

آرش حریرپوش
9731505

حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 2 را جایگذاری می کنیم:

$$1 + X1 + X2 = 1 + 0 + 1 = 2$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد، که برای ورودی این ردیف درست می باشد.

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 درست است.

حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 3 را جایگذاری می کنیم:

$$1 + X1 + X2 = 1 + 1 + 0 = 2$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد که برای ورودی این ردیف درست می باشد.

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 و 3 درست است.

حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 4 را جایگذاری می کنیم:

$$1 + X1 + X2 = 1 + 1 + 1 = 3$$

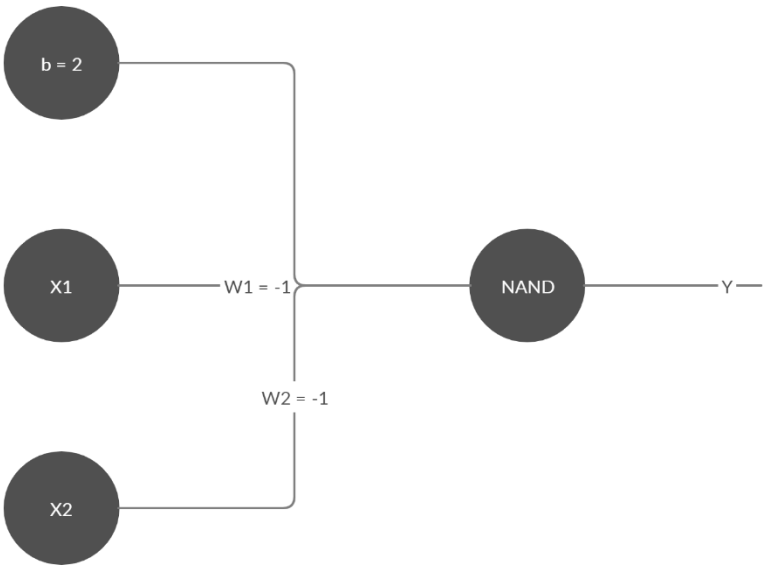
با توجه به قانون شماره 2 مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای NAND برابر 0 می باشد.

بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 0 کند. برای این کار مقدار b را برابر 2 و W1 و W2 را برابر 1- قرار می دهیم:

$$2 - X1 - X2 = 2 - 1 - 1 = 0$$

در نتیجه مقدار وزن ها و بایاس برای همه 4 ردیف ورودی درست می باشد؛ بنابراین نتیجه می گیریم که مدلی که گیت NAND را پیاده سازی کند به صورت زیر می باشد:

$$2 - X1 - X2$$

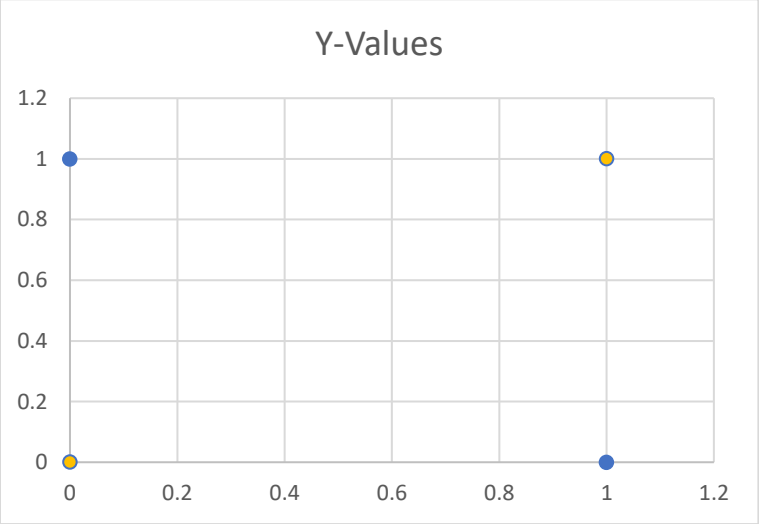


آرش حریرپوش
9731505

- XNOR

X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	1

با توجه به جدول این گیت، نمودار آن را رسم می کنیم:



در شکل بالا نقاط زرد رنگ برابر مقدار 1 و نقاط آبی رنگ برابر مقدار0 می باشند، همانطور که از شکل بالا مشخص است برای تفکیک نقاط زرد از آبی به دو خط نیاز داریم (این مسئله برخلاف مسائل قبلی غیرخطی می باشد به همین دلیل به بیش از یک پرسپترون نیاز داریم) به همین دلیل این مورد همانند موارد قبلی به کمک یک پرسپترون نمی تواند انجام شود بلکه به 2 پرسپترون برای ورودی و یک پرسپترون برای خروجی نیاز دارد؛ به دلیل اینکه محاسبه وزن های پرسپترون چند لایه مشکل می باشد، به کمک توابع AND و OR و NOR این مورد را محاسبه می کنیم:

$$XNOR = X1X2 + X1'X2' = OR(AND(X1, X2), NOR(X1, X2))$$

بنابراین ابتدا تابع AND را به کمک یک پرسپترون پیاده سازی می کنیم(در این قسمت به بعد مقدار بایاس را 1 در نظر می گیریم و در صورت نیاز برای تغییر در وزن آن ضرب می کنیم):

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

آرش حریرپوش
9731505

با توجه به جدول بالا و فرمول 1 وزن های ورودی و بایاس را برای محاسبه AND به صورت زیر محاسبه می کنیم:
ابتدامقادیر اولیه W0 و W1 و W2 را به ترتیب برابر 1- و 1 و1 قرار می دهیم:

$$-1 + X1 * 1 + X2 * 1 = -1 + X1 + X2$$

سپس مقادیر ردیف اول را در فرمول بالا قرار می دهیم:

$$-1 + 0 + 0 = -1$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف درست می باشد.
در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف 1 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 2 را جایگذاری می کنیم:

$$-1 + X1 + X2 = -1 + 0 + 1 = 0$$

با توجه به قانون شماره 2 مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف درست می باشد.
در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 3 را جایگذاری می کنیم:

$$-1 + X1 + X2 = -1 + 1 + 0 = 0$$

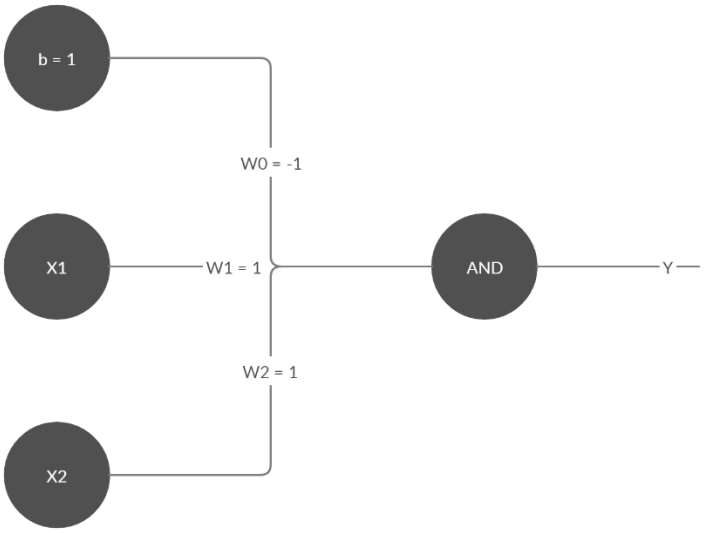
با توجه به قانون شماره 2 مقدار $Y^* = 0$ می باشد که برای ورودی این ردیف درست می باشد.
در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 و 3 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 4 را جایگذاری می کنیم:

$$-1 + X1 + X2 = -1 + 1 + 1 = 1$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد، که برای ورودی این ردیف درست می باشد.
در نتیجه مقدار وزن ها و بایاس برای همه 4 ردیف ورودی درست می باشد؛ بنابراین نتیجه می گیریم که مدلی که گیت AND را پیاده سازی کند به صورت زیر می باشد:

$$-1 + X1 + X2$$

آرش حریرپوش
9731505



سپس تابع OR را به صورت زیر پیاده سازی می کنیم:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

با توجه به جدول بالا و فرمول 1 وزن های ورودی و بایاس را برای محاسبه OR به صورت زیر محاسبه می کنیم:

ابتدامقادیر اولیه W0 و W1 و W2 را به ترتیب برابر 1- و 1 و1 قرار می دهیم:

$$-1 + X1 * 1 + X2 * 1 = -1 + X1 + X2$$

سپس مقادیر ردیف اول را در فرمول بالا قرار می دهیم:

$$-1 + 0 + 0 = -1$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف درست می باشد.

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف 1 درست است.

حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 2 را جایگذاری می کنیم:

$$-1 + X1 + X2 = -1 + 0 + 1 = 0$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای OR برابر 1 می باشد.

بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 1 کند. برای این کار مقدار W2 را برابر 2 قرار می دهیم:

$$-1 + X1 + 2 * X2 = -1 + 0 + 2 = 1$$

آرش حریرپوش
9731505

در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 3 را جایگذاری می کنیم:

$$-1 + X1 + 2 * X2 = -1 + 1 + 0 = 0$$

از طریق قانون پرسپترون، قانون 2، مقدار $Y^* = 0$ می باشد، که برای ورودی این ردیف اشتباه می باشد، چراکه خروجی این ردیف برای OR برابر 1 می باشد.
بنابراین، ما به مقادیری نیاز داریم تا مقدار Y^* را برابر 1 کند. برای این کار مقدار W1 را برابر 2 قرار می دهیم:

$$-1 + 2 * X1 + 2 * X2 = -1 + 2 + 0 = 1$$

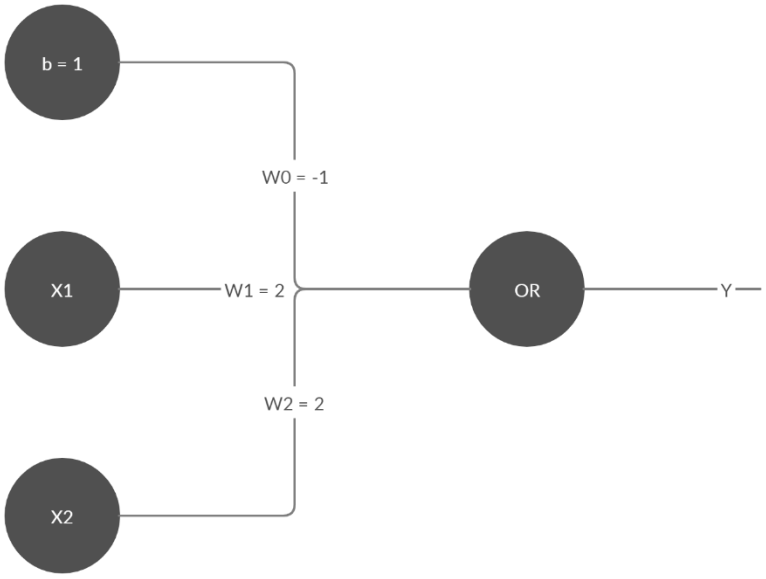
در نتیجه به وسیله قانون پرسپترون (قانون 2) این وزن ها برای ورودی ردیف های 1 و 2 و 3 درست است.
حال با توجه به مقادیر وزن ها و بایاس مقدار ردیف 4 را جایگذاری می کنیم:

$$-1 + 2 * X1 + 2 * X2 = -1 + 2 + 2 = 3$$

با توجه به قانون شماره 2 مقدار $Y^* = 1$ می باشد، که برای ورودی این ردیف درست می باشد.

در نتیجه مقدار وزن ها و بایاس برای همه 4 ردیف ورودی درست می باشد؛ بنابراین نتیجه می گیریم که مدلی که گیت OR را پیاده سازی کند به صورت زیر می باشد:

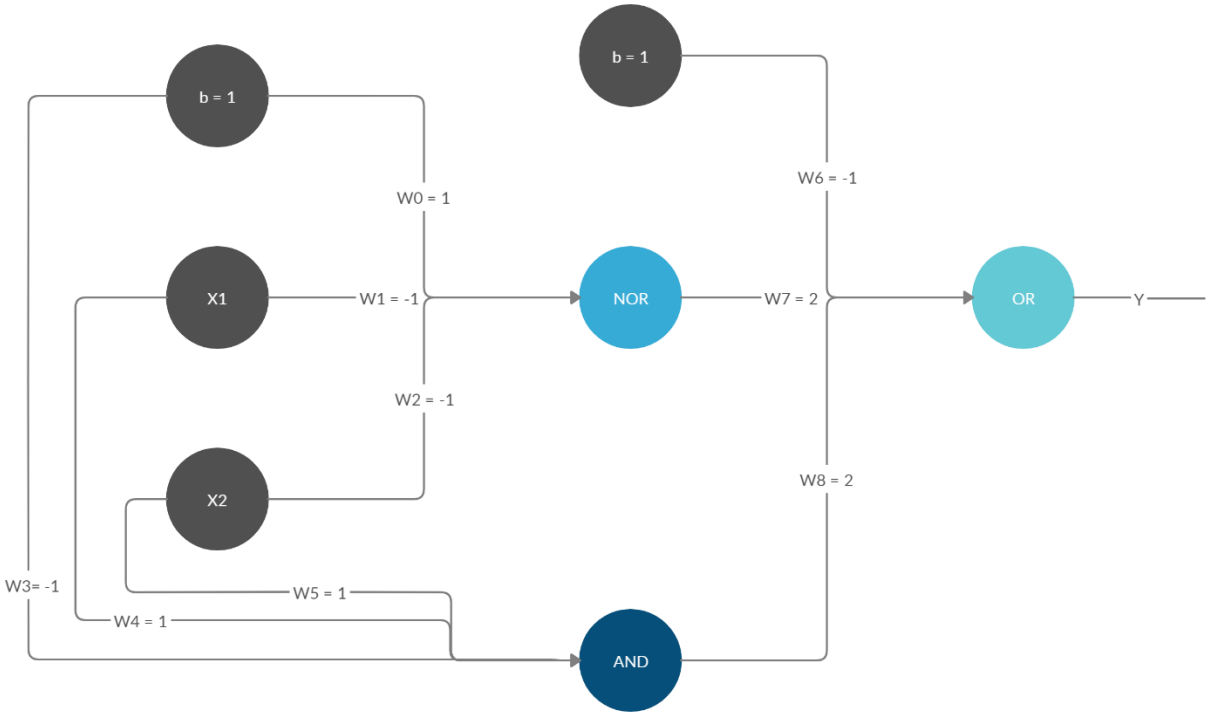
$$-1 + 2 * X1 + 2 * X2$$



آرش حریرپوش
9731505

حال به کمک مدل های NOR و AND و OR مدل XNOR را به صورت زیر پیاده سازی می کنیم:

$$OR(AND(X1, X2), NOR(X1, X2))$$



4. یادگیری بانظارت، بدون نظارت و تقویتی یکی از روش های تقسیم بندی مسائل مختلف یادگیری ماشین می باشد. این سه دسته را شرح دهید و کارآیی و نحوه استفاده شبکه های عصبی را در این مسائل توضیح دهید.

ابتدا روش های تقسیم بندی مسائل مختلف یادگیری ماشین را به صورت زیر شرح می دهیم:

- یادگیری با نظارت:

در این روش یک مجموعه داده که label های(خروجی های آن ، دسته بندی هر یک از داده های آموزشی مشخص شده است) آنها مشخص است برای آموزش شبکه استفاده می کنیم. و سپس پارامتر های شبکه عصبی را با آموزش آن داده ها تنظیم می کنیم. وظیفه یادگیری شبکه عصبی مصنوعی تنظیم مقادیر پارامتر های خودش برای هر مقدار ورودی معتبر پس از دیدن خروجی آن می باشد. داده آموزشی شامل جفت های ورودی و مقدار خروجی مورد نظر است که در بردار (vector) های داده نمایش داده شده اند (در این روش یادگیری بر اساس اصلاح و تنظیم وزن ها و بایاس برای نزدیکی به مقادیر پاسخ انجام می شود). همچنین یادگیری نظارت شده به classification نیز شناخته شده اند، که ما classifier ها زیادی داریم، که هر کدام نقاط ضعف و قدرت خود را دارند.

برای حل مسئله یادگیری نظارت شده داده شده قدم های گوناگون زیر را باید در نظر بگیریم. در قدم اول باید نوع مثال های آموزشی را مشخص کنیم. در گام دوم نیاز داریم تا مجموعه داده آموزشی که مشکل داده شده را به صورت میانه توصیف می کند جمع آوری کنیم. در قدم سوم نیاز داریم تا مجموعه داده آموزشی جمع آوری شده را به صورتی که برای شبکه عصبی مصنوعی که انتخاب کرده ایم قابل یادگیری باشد توصیف کنیم. در گام چهارم یادگیری را انجام می دهیم و پس از آن عملکرد یادگیری شبکه عصبی مصنوعی را به کمک مجموعه داده های test (validation) ارزیابی می کنیم. مجموعه داده test شامل داده هایی می باشد که در زمان یادگیری شبکه عصبی مصنوعی به آن نمایش داده نشده است.

- یادگیری بدون نظارت:

در این روش یک مجموعه داده که دارای label نمی باشند برای آموزش شبکه استفاده می کنیم. و سپس پارامتر های شبکه عصبی مصنوعی براساس داده های داده شده و تابع هزینه که باید minimized باشد تنظیم می شود. تابع هزینه هر تابعی می تواند باشد و توسط task formulation مشخص می شود. این روش یادگیری معمولا در برنامه هایی که در دامنه تخمین مشکلات قرار می گیرند، همانند مدل سازی آماری، فشرده سازی، فیلتر کردن، جداسازی منبع کور (blind source separation) و clustering استفاده می شود (وزن ها و بایاس شبکه فقط بر اساس ورودی ها اصلاح می شوند). در این نوع یادگیری ما به دنبال نحوه سازماندهی داده ها هستیم. این روش با یادگیری نظارت شده و یادگیری تقویتی که در آن نیز داده

ها دارای label نمی باشند متفاوت است. یکی از روش های معمول این یادگیری clustering می باشد که در آن سعی می کنیم تا داده را در cluster های مختلف به وسیله شباهت هایشان دسته بندی کنیم.

- یادگیری تقویتی:

در این روش یادگیری در شبکه ها توسط مشاهده و سعی و خطا صورت می پذیرد. و سپس پارامتر های این شبکه عصبی مصنوعی، که داده معمولا به آن ها داده نشده است، اما به وسیله تعامل با محیط جمع آوری می شود تنظیم می شود. در این روش نگرانی در مورد این است که یک شبکه عصبی مصنوعی چگونه در محیط اقدام کند تا برخی از ایده های پاداش بلند مدت در آن شبکه maximize (بیشینه) شود. (با توجه به تعاملی که با محیط پیرامون خود دارد عملی انجام داده و نتیجه عمل را به صورت پاداش و تنبیه برای نزدیکی و دوری به هدف مشخص می شود. شبکه کارهایی انجام می دهد که منجر به کسب پاداش بیشتری می شود) یادگیری تقویتی اغلب به عنوان بخشی از الگوریتم های یادگیری شبکه عصبی استفاده می شود.
این روش به ویژه برای مسائلی که شامل trade-off بین پاداش های طولانی مدت در مقابل کوتاه مدت مناسب می باشد. در مسلائل گوناگونی به موفقیت استفاده شده است، شامل کنترل ربات، ارتباطات از راه دور، و بازی هایی مثل شطرنج و بقیه تصمیم های پی در پی برای انجام وظایف.

❖ بخش دوم – مسائل برنامه نویسی و پیاده سازی

1. فایلی برای داده ها همراه این تمرین آپلود شده است. داده ها را در یک نمودار scatter نشان دهید و داده های هر دسته را با رنگی جداگانه نشان دهید. آن را به دو دسته آموزش و آزمایش جدا کنید. داده ها نقاطی دو بعدی هستند که برچسب 0 و 1 دارند.

ابتدا کتابخانه های لازم برای خواندن و رسم نمودار در پایتون را به برنامه اضافه می کنیم:

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
```

سپس به کمک کتابخانه های اضافه شده داده ها را از فایل CSV می خوانیم:

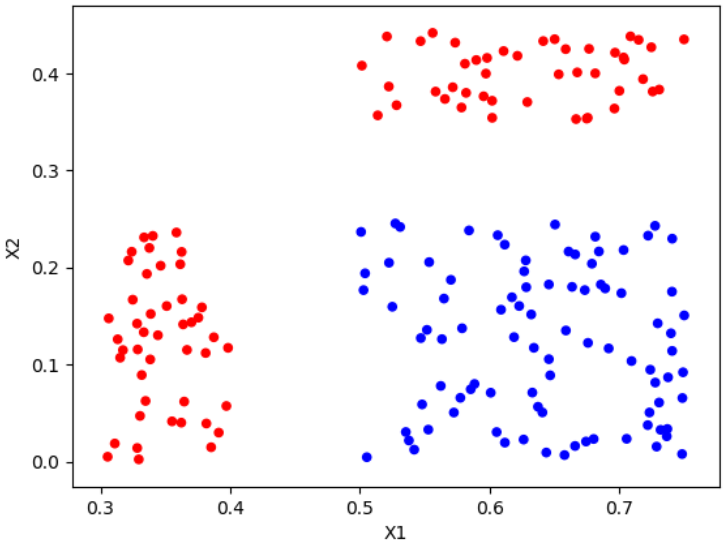
```
# reading csv file
data = pd.read_csv("dataset.csv")

df = pd.DataFrame(data, columns=['X1','X2', 'Label'])
```

سپس داده های با برچسب 0 را آبی و داده های با برچسب 1 را قرمز در نظر می گیریم و به کمک قطعه کد زیر نمودار scatter آن را رسم می کنیم:

```
#set the colors of data with label 0 as blue and label 1 as red
colors = np.where(df.Label > 0, 'r', 'b')
#plot the scatter of the data
scatter = df.plot(kind='scatter', x='X1', y='X2', c=colors)
#show the plotted scatter
plt.show()
```

در نتیجه نمودار scatter داده های داده شده به کمک قطعه کد بالا به صورت زیر رسم می شود:



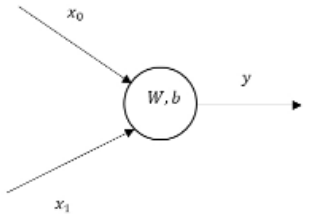
سپس به کمک قطعه کد زیر داده های داده شده را به دو دسته train و test تقسیم می کنیم، بدین صورت که 150 داده اول جدول را به عنوان train و بقیه داده ها (30 داده آخر) را به عنوان test در نظر می گیریم:

```
# split data to train and test
x_train = df[['X1', 'X2']][0:150]
y_train = df[['Label']][0:150]
x_test = df[['X1', 'X2']][150:]
y_test = df[['Label']][150:]
```

آرش حریرپوش
9731505

2. شبکه عصبی شکل 2 را در نظر بگیرید.

$$y = S(W \cdot X + b), W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, b \in R, X = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$
$$cost = -\sum_{\forall x} [y_T * \log(y) + (1 - y_T) * \log(1 - y))]$$
$$X, y_T \in data$$
$$S(x) = \frac{1}{1 + e^{-x}}, (Sigmoid\ function),$$
$$\frac{dS(x)}{dx} = S(x) * (1 - S(x))$$



شکل ۲- شبکه عصبی با یک نرون

- در این شبکه مقادیر زیر را محاسبه کنید.
راهنمایی: از قاعده زنجیره ای مشتق استفاده کنید.

$$\frac{dcost}{dW} = \frac{dcost}{dy} * \frac{dy}{dW}$$

ابتدا مقدار مشتق های زیر را محاسبه می کنیم:

$$\frac{dcost}{dy} = -\sum_{\forall x} \frac{y_T}{y} - \frac{1 - y_T}{(1 - y)} = -\sum_{\forall x} \frac{y_T - (y * y_T) - y + (y * y_T)}{y * (1 - y)}$$

$$= -\sum_{\forall x} \frac{y_T - y}{y * (1 - y)}$$

$$y = S(W.X + b) = \frac{1}{e^{-(W.X+b)}} = \frac{1}{e^{-W.X-b}}$$

$$\frac{dy}{dW} = \frac{-(-X * e^{-WX-b})}{(1 + e^{-WX-b})^2} = \frac{X * e^{-WX-b}}{(1 + e^{-WX-b})^2}$$
$$\frac{dy}{db} = \frac{-(-e^{-WX-b})}{(1 + e^{-WX-b})^2} = \frac{e^{-WX-b}}{(1 + e^{-WX-b})^2}$$

از رابطه $\frac{dS(x)}{dx}$ = S(x) * (1-S(x)) می توانیم دو مشتق بالا را به صورت زیر بازنویسی کنیم:

$$\frac{dy}{dW} = X * (S(W.X + b) * (1 - S(W.X + b)))$$
$$\frac{dy}{db} = S(W.X + b) * (1 - S(W.X + b)))$$

آرش حریرپوش
9731505

حال به کمک مشتق های بالا مشتق های زیر را محاسبه می کنیم:

$$\frac{dcost}{dW} = \frac{dcost}{dy} * \frac{dy}{dW} = -\sum_{\forall x} \frac{y_T - y}{y * (1-y)} * \frac{X_{Transpose} * e^{-WX-b}}{(1+e^{-WX-b})^2}$$

$$= -\sum_{\forall x} \frac{y_T - y}{y * (1-y)} * (X_{Transpose} * (S(W.X+b)$$

$$* \big(1 - S(W.X+b)\big))$$

$$= -X_{Transpose} * \sum_{\forall x} \frac{y_T - y}{y * (1-y)} * (S(W.X+b)$$

$$* (1 - S(W.X+b))) = -X_{Transpose} * \sum_{\forall x} y_T - y$$

$$\frac{dcost}{db} = \frac{dcost}{dy} * \frac{dy}{db} = -\sum_{\forall x} \frac{y_T - y}{y * (1-y)} * \frac{e^{-WX-b}}{(1+e^{-WX-b})^2}$$

$$= -\sum_{\forall x} \frac{y_T - y}{y * (1-y)} * \Big(S(W.X+b) * \big(1 - S(W.X+b)\big)\Big)$$

$$= -\sum_{\forall x} y_T - y$$

آرش حریرپوش
9731505

3. با استفاده از داده های آموزشی و شبه کد شکل3، این شبکه را آموزش دهید. مقادیر نرخ یادگیری(lr) و تعداد گام (n_epoch) مناسب را پیدا و گزارش کنید.

```
initialize W and b from N(0,1)
n_epoch = ؟
lr = ؟
n = number of train records
for i from 0 to n_epoch:
        for each W or b:
                grad[W] = 0 //  grad[b] for biases
                for x0,x1,y0 in train_data:
                        compute y
                        compute cost
                        grad[W] += dcost/dW // grad[b]  and db for biases
        for each W or b:
                W = W – (lr * grad[w])/n // b and grad[b] for biases
```

شکل۳- شبه کد آموزش وزن‌های شبکه عصبی

ابتدا داده ها آموزشی را به کمک قطعه کد زیر آموزش می دهیم:

```python
def S(x):
    return 1/(1+np.exp(-x))

def fit(x_train, y_train, n_epoch, lr):
    train_data = list(zip(x_train['X1'], x_train['X2'],y_train['Label']))

    n = len(train_data)

    weights = np.random.normal(0, 1, size= 2)
    b = np.random.normal(0, 1, size=1)[0]

    grad = [0, 0, 0]
    for epoch in range(n_epoch):
        cost = 0
        # grad[0] for bias
        for w in range(len(weights)):
            grad[w+1] = 0
            #grad[0] stands for bias grad
            grad[0] = 0
            for data in train_data:
                # compute y
                x1 = data[0]
                x2 = data[1]
                yt = data[2]
                x = np.array([x1, x2], dtype=float)
                W = np.array([weights[:]], dtype=float)
                inp_y = sum(i for i in np.multiply(x, W)[0])
                inp_y+= b
                y = S(inp_y)

                # compute cost
                cost += -(yt*np.log(y) + (1-yt)*np.log(1-y))
                # compute grad of the weights
                grad[w+1] += -data[w]*(yt - y)
```

```
                # compute grad[0] for biases
                grad[0] += -(yt - y)

        #update weights
        for w in range(len(weights)):
            weights[w] = weights[w] - (lr * grad[w+1]) / n
        #update bias
        b = b - (lr * grad[0])/n
    return b, weights
```

سپس به کمک قطعه کد زیر به کمک داده های تست دقت مدل را ارزیابی می کنیم:

```
def test(x_test, y_test, weights, scatter = 0):
    test_data = pd.DataFrame({'X1':x_test['X1'], 'X2':x_test['X2'],
'Label':y_test['Label']})
    X1 = []
    X2 = []
    Label = []
    n = len(x_test)
    true_predicts = 0

    for row, data in test_data.iterrows():
        x1 = data['X1']
        x2 = data['X2']
        y = data['Label']
        yp = S(weights[0] + x1*weights[1][0] + x2*weights[1][1])
        if yp > 0.5:
            yp = 1
        else:
            yp = 0
        if y == yp:
            true_predicts += 1
        X1.append(x1)
        X2.append(x2)
        Label.append(yp)
    acc = true_predicts/n
    if scatter:
        # set the colors of data with label 0 as blue and label 1 as red
        colors = np.where(test_data.Label > 0, 'r', 'b')
        # plot the scatter of the data
        inp_scatter = test_data.plot(title= 'Test Dataset',
kind='scatter', x='X1', y='X2', c=colors)
        # show the plotted scatter
        plt.show()

        res_data = pd.DataFrame({'X1': X1, 'X2': X2, 'Label': Label})
        # set the colors of data with label 0 as blue and label 1 as red
        colors = np.where(res_data.Label > 0, 'r', 'b')
        #plot the scatter of the data
        predicted_scatter = res_data.plot(title= 'Predicted Test
Dataset', kind='scatter', x='X1', y='X2', c=colors)
        #show the plotted scatter
        plt.show()
    return acc
```

سپس به کمک قطعه کد زیر شبکه را آموزش و دقت آن را به ازای هایپر پارامتر های مختلف ارزیابی می کنیم:

```
#train the model
weights = fit(x_train, y_train, 1500, 0.5)

#test the model
acc = test(x_test, y_test, weights, 1)
acc_train = test(x_train, y_train, weights, 1)
print("Accuracy test of model is: ", acc)
print("Accuracy train of model is: ", acc_train)
```

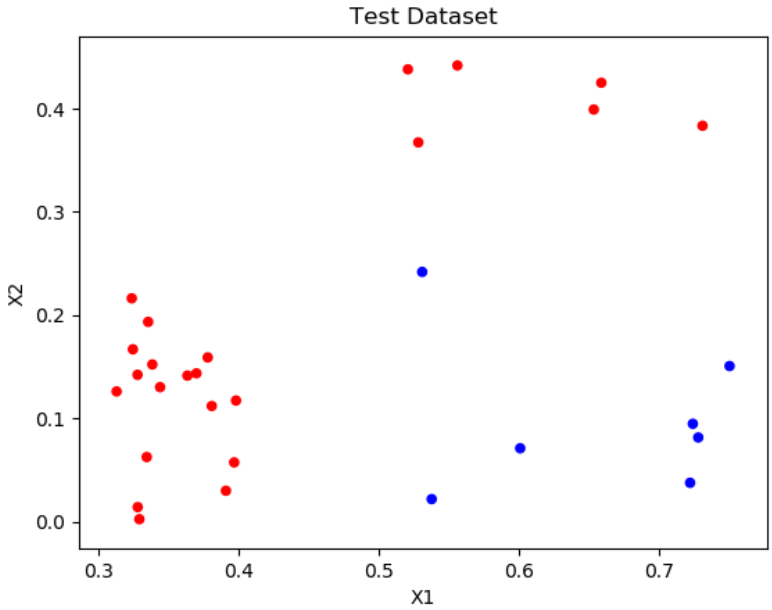
در نهایت مقدار دقت مدل به ازای هایپر پارامتر های مختلف را در جدول زیر مشاهده می کنید:

تعداد گام (n_epoch)	نرخ یادگیری (lr)	دقت داده تست (Accuracy test)	دقت داده آموزش (Accuracy train)
5000	0.01	0.43333333333333335	0.7866666666666666
700	0.1	0.7	0.9266666666666666
1000	0.2	0.8	0.92
2000	0.15	0.8333333333333334	0.9333333333333333
1700	0.25	0.8333333333333334	0.9266666666666666
1500	0.3	0.8333333333333334	0.9266666666666666
1500	0.4	0.8333333333333334	0.9133333333333333
1500	0.5	0.8333333333333334	0.94
1400	0.6	0.8333333333333334	0.94
1200	0.7	0.8333333333333334	0.94
1000	0.8	0.8333333333333334	0.94
1000	0.88	0.8333333333333334	0.94
800	1	0.8333333333333334	0.94

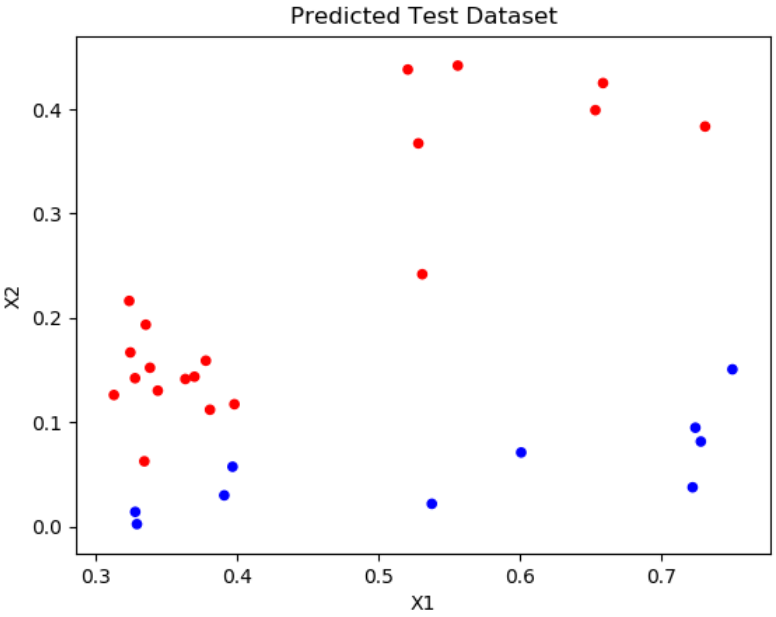
همانطور که از جدول بالا مشاهده می کنید، با افزایش مقدار n_epoch و lr میزان دقت مدل در داده های تست و آموزش افزایش یافته است، به طوری که پس از ردیف سوم در جدول بالا مقدار دقت ثابت مانده چرا که به دلیل بالا بودن تعداد گام ها n_epoch مدل تا حداکثر توان خود در داده های تست آموزش یافته است، اما نکته جالبی که می توان در این جدول مشاهده کرد افزایش مقدار دقت داده آموزش است که همواره تا مقدار ردیف حاوی n_epoch = 1500 و lr = 0.4 متغیر بوده اما سپس پس از این ردیف، ردیف 7 به بعد مقدار آن به بیشینه رسیده و تا انتها ثابت مانده است، که این الگو می تواند نشان دهنده overfit باشد، چرا که دقت داده های تست که مدل آن ها را مشاهده نکرده تغییری نمی کند اما دقت داده های آموزش که مدل بر اساس آن ها آموزش دیده افزایش پیدا می کند، این یعنی مدل می تواند داده هایی که قبلا دیده است را بهتر پیشبینی کند اما داده هایی که تا کنون ندیده است را نمی تواند بهتر پیشبینی کند.

آرش حریرپوش
9731505

نمودار داده آزمایش را به صورت زیر رسم می کنیم:

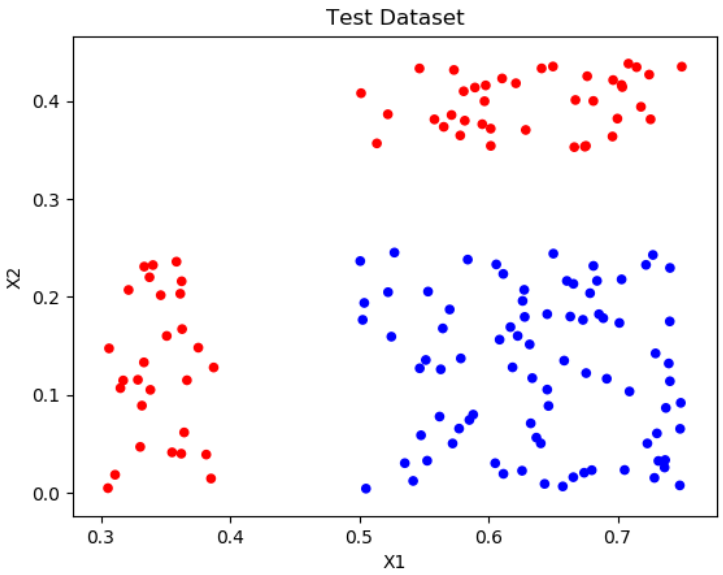


سپس نمودار داده پیشبینی شده توسط مدل آموزش دیده برای n_epoch = 1500 و lr = 0.5 به صورت زیر رسم می کنیم:

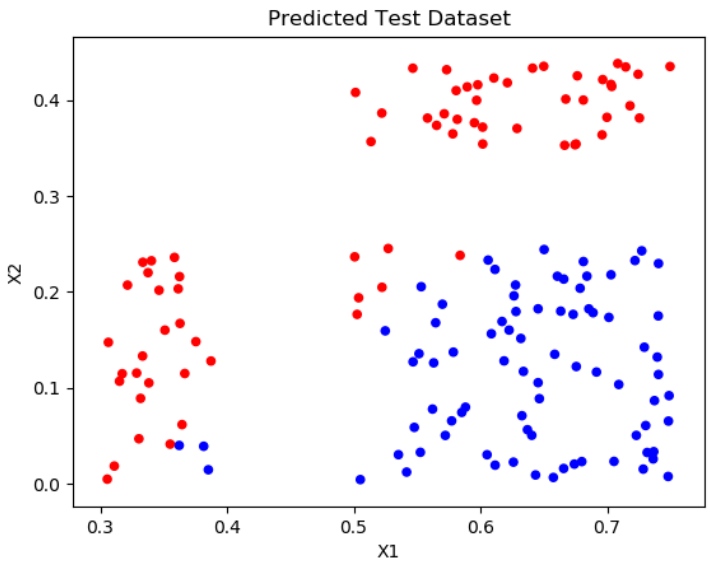


4. حالا داده های آموزش را به شبکه بدهید و دقت دسته بندی را ارزیابی کنید. نمودار scatter برای داده های آموزش را رسم کنید (**رنگ هر نقطه بر اساس خروجی شبکه برای آن نقطه باشد**).

در نمودار های رسم شده، داده های با برچسب 0 را آبی و داده های با برچسب 1 را قرمز در نظر می گیریم، ابتدا نمودار داده های آموزش ورودی را به صورت زیر رسم می کنیم:



سپس پس از آموزش داده و تست کردن داده های آموزش در مدل آموزش داده شده با هایپر پارامتر های n_epoch = 1500 و lr = 0.5 به دقت 0.94 رسیده ایم، و نمودار Label های پیشبینی شده داده آموزش در مدل آموزش داده شده به صورت زیر می باشد:

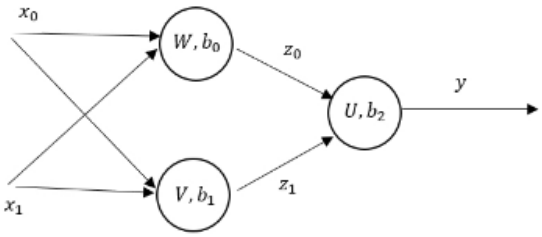


(کد این بخش در فایل های ANN_HW01.ipynb و ANN_HW01.py به نام به پیوست ارائه گردیده است.)

آرش حریرپوش
9731505

5. حال شبکه شکل 4 را در نظر بگیرید.

$$z_0 = S(X \cdot W + b_0)$$
$$z_1 = S(X \cdot V + b_1)$$
$$y = S(Z \cdot U + b_2)$$
$$Z = \begin{bmatrix} z_0 \\ z_1 \end{bmatrix}, W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, V = \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}, U = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}$$
$$cost = (y - y_T)^2$$
$$X, y_T \in data$$
$$S(x) = \frac{1}{1 + e^{-x}} \quad (Sigmoid\ function)$$
$$\frac{dS(x)}{dx} = S(x) * (1 - S(x))$$



شکل ۴- شبکه عصبی با سه لایه ورودی، میانی و خروجی

- در این شبکه مقادیر زیر را محاسبه کنید.

ابتدا مقدار مشتق های زیر را محاسبه می کنیم:

$$\frac{dcost}{dy} = 2 * (y - y_T)$$

$$\frac{dy}{dz_0} = u_0 * \Big(S(Z.U + b_2) * \big(1 - S(Z.U + b_2)\big)\Big) = u_0 * (y * (1 - y))$$

$$\frac{dz_0}{dW} = X_{Transpose} * \Big(S(X.W + b_0) * \big(1 - S(X.W + b_0)\big)\Big)$$
$$= X_{Transpose} * (z_0 * (1 - z_0))$$

$$\frac{dy}{dz_1} = u_1 * \Big(S(Z.U + b_2) * \big(1 - S(Z.U + b_2)\big)\Big) = u_1 * (y * (1 - y))$$

$$\frac{dz_1}{dV} = X_{Transpose} * \Big(S(X.V + b_1) * \big(1 - S(X.V + b_1)\big)\Big)$$
$$= X_{Transpose} * (z_1 * (1 - z_1))$$

$$\frac{dy}{dU} = Z_{Transpose} * \Big(S(Z.U + b_2) * \big(1 - S(Z.U + b_2)\big)\Big)$$
$$= Z_{Transpose} * (y * (1 - y))$$

$$\frac{dz_0}{db_0} = S(X.W + b_0) * \big(1 - S(X.W + b_0)\big) = z_0 * (1 - z_0)$$

$$\frac{dz_1}{db_1} = S(X.V + b_1) * \big(1 - S(X.V + b_1)\big) = z_1 * (1 - z_1)$$

$$\frac{dy}{db_2} = S(Z.U + b_2) * \big(1 - S(Z.U + b_2)\big) = y * (1 - y)$$

آرش حریرپوش
9731505

حال به کمک مشتق های بالا مشتق های زیر را محاسبه می کنیم:

$$\frac{dcost}{dW} = \frac{dcost}{dy} * \frac{dy}{dz_0} * \frac{dz_0}{dW}$$
$$= 2 * (y - y_T) * u_0 * \big(y * (1 - y)\big) * X_{Transpose}$$
$$* \big(z_0 * (1 - z_0)\big)$$
$$= 2 * u_0 * X_{Transpose} * (y - y_T) * \big(y * (1 - y)\big)$$
$$* \big(z_0 * (1 - z_0)\big)$$

$$\frac{dcost}{dV} = \frac{dcost}{dy} * \frac{dy}{dz_1} * \frac{dz_1}{dV}$$
$$= 2 * (y - y_T) * u_1 * \big(y * (1 - y)\big) * X_{Transpose}$$
$$* \big(z_1 * (1 - z_1)\big)$$
$$= 2 * u_1 * X_{Transpose} * (y - y_T) * \big(y * (1 - y)\big)$$
$$* \big(z_1 * (1 - z_1)\big)$$

$$\frac{dcost}{dU} = \frac{dcost}{dy} * \frac{dy}{dU} = 2 * (y - y_T) * Z_{Transpose} * \big(y * (1 - y)\big)$$
$$= 2 * Z_{Transpose} * (y - y_T) * \big(y * (1 - y)\big)$$

$$\frac{dcost}{db_0} = \frac{dcost}{dy} * \frac{dy}{dz_0} * \frac{dz_0}{db_0}$$
$$= 2 * (y - y_T) * u_0 * \big(y * (1 - y)\big) * z_0 * (1 - z_0)$$
$$= 2 * u_0 * z_0 * (y - y_T) * \big(y * (1 - y)\big) * (1 - z_0)$$

$$\frac{dcost}{db_1} = \frac{dcost}{dy} * \frac{dy}{dz_1} * \frac{dz_1}{db_1}$$
$$= 2 * (y - y_T) * u_1 * \big(y * (1 - y)\big) * z_1 * (1 - z_1)$$
$$= 2 * u_1 * z_1 * (y - y_T) * \big(y * (1 - y)\big) * (1 - z_1)$$

$$\frac{dcost}{db_2} = \frac{dcost}{dy} * \frac{dy}{db_2} = 2 * (y - y_T) * y * (1 - y)$$
$$= 2 * y * (y - y_T) * (1 - y)$$

- قسمت 3 و 4 را برای این شبکه نیز انجام دهید.

ابتدا داده ها آموزشی را به کمک قطعه کد زیر آموزش می دهیم:

```
def S(x):
    return 1/(1+np.exp(-x))

def fit(x_train, y_train, n_epoch, lr):
    train_data = list(zip(x_train['X1'], x_train['X2'],
y_train['Label']))
    weights = np.random.normal(0, 1, size= 6)
    # b = abs(np.random.normal(0, 0.1, size=1))
    bias = np.random.normal(0, 1, size=3)
    z = [0, 0]
    grad = np.zeros(9)

    for epoch in range(n_epoch):
        for data in train_data:
            # compute y
            yt = data[2]
            z[0] = S(data[0]*weights[0] + data[1]*weights[1] + bias[0])
            z[1] = S(data[0]*weights[2] + data[1]*weights[3] + bias[1])
            y = S(z[0]*weights[4] + z[1]*weights[5] + bias[2])
            loss = y - yt
            dis_y = y*(1-y)
            dis_z0 = z[0]*(1-z[0])
            dis_z1 = z[1]*(1-z[1])
            #dcost/db0
            grad[0] = 2*weights[4]*loss*dis_y*dis_z0
            #dcost/db1
            grad[1] = 2*weights[5]*loss*dis_y*dis_z1
            #dcost/db2
            grad[2] = 2*loss*dis_y
            #dcost/dW
            grad[3] = data[0]*grad[0]
            grad[4] = data[1]*grad[0]
            #dcost/dV
            grad[5] = data[0]*grad[1]
            grad[6] = data[1]*grad[1]
            #dcost/dU
            grad[7] = z[0]*grad[2]
            grad[8] = z[1]*grad[2]
            #update weights
            num_of_weight_group = int(len(weights)/2)
            num_of_w_in_group = int(len(weights)/num_of_weight_group)
            for w in range(num_of_weight_group):
                for i in range(num_of_w_in_group):
                    weights[2*w+i] = weights[2*w+i] - (lr *
grad[3+2*w+i])
            #update bias
            for b in range(len(bias)):
                bias[b] = bias[b] - (lr * grad[b])
    return bias, weights
```

سپس به کمک قطعه کد زیر به کمک داده های تست دقت مدل را ارزیابی می کنیم:

```
def test(x_test, y_test, params, scatter = 0):
    test_data = pd.DataFrame({'X1':x_test['X1'], 'X2':x_test['X2'],
'Label':y_test['Label']})
    bias = params[0]
    weights = params[1]
    z = [0, 0]

    X1 = []
    X2 = []
    Label = []
    n = len(x_test)
    true_predicts = 0

    for row, data in test_data.iterrows():
        x1 = data['X1']
        x2 = data['X2']
        y = data['Label']
        z[0] = S(x1 * weights[0] + x2 * weights[1] + bias[0])
        z[1] = S(x1 * weights[2] + x2 * weights[3] + bias[1])
        yp = S(z[0] * weights[4] + z[1]*weights[5] + bias[2])

        if yp > 0.5:
            yp = 1
        else:
            yp = 0
        if y == yp:
            true_predicts += 1
        X1.append(x1)
        X2.append(x2)
        Label.append(yp)
    acc = true_predicts/n
    if scatter:
        # set the colors of data with label 0 as blue and label 1 as red
        colors = np.where(test_data.Label > 0, 'r', 'b')
        # plot the scatter of the data
        inp_scatter = test_data.plot(title= 'Test Dataset',
kind='scatter', x='X1', y='X2', c=colors)
        # show the plotted scatter
        plt.show()

        res_data = pd.DataFrame({'X1': X1, 'X2': X2, 'Label': Label})
        # set the colors of data with label 0 as blue and label 1 as red
        colors = np.where(res_data.Label > 0, 'r', 'b')
        #plot the scatter of the data
        predicted_scatter = res_data.plot(title= 'Predicted Test
Dataset', kind='scatter', x='X1', y='X2', c=colors)
        #show the plotted scatter
        plt.show()
    return acc
```

سپس به کمک قطعه کد زیر شبکه را آموزش و دقت آن را به ازای هایپر پارامتر های مختلف ارزیابی می کنیم:

```
#train the model
weights = fit(x_train, y_train, 1000, 0.1)

#test the model
acc = test(x_test, y_test, weights, 1)
acc_train = test(x_train, y_train, weights, 1)
print("Accuracy test of model is: ", acc)
print("Accuracy train of model is: ", acc_train)
```

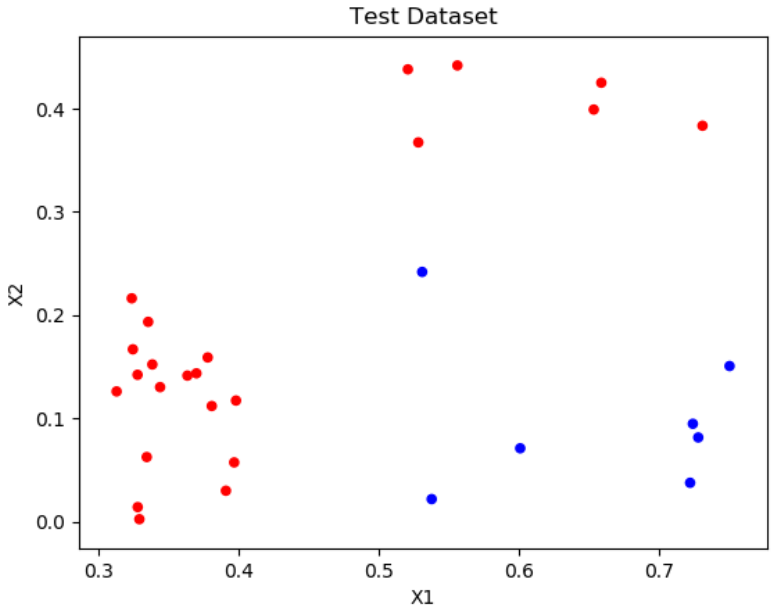
در نهایت مقدار دقت مدل به ازای هایپر پارامتر های مختلف را در جدول زیر مشاهده می کنید:

تعداد گام (n_epoch)	نرخ یادگیری (lr)	دقت داده تست (Accuracy test)	دقت داده آموزش (Accuracy train)
2000	0.01	0.9	0.9466666666666667
2000	0.07	0.9333333333333333	0.98
1500	0.1	0.8666666666666667	0.9466666666666667
1500	0.2	0.9	0.9466666666666667
1500	0.3	1.0	1.0
1400	0.4	1.0	1.0
1400	0.5	1.0	1.0
1200	0.6	1.0	1.0
1200	0.7	0.9	0.9533333333333334
1200	0.77	1.0	1.0
1000	0.8	1.0	1.0
800	0.9	0.9	0.9466666666666667
300	1	1.0	1.0

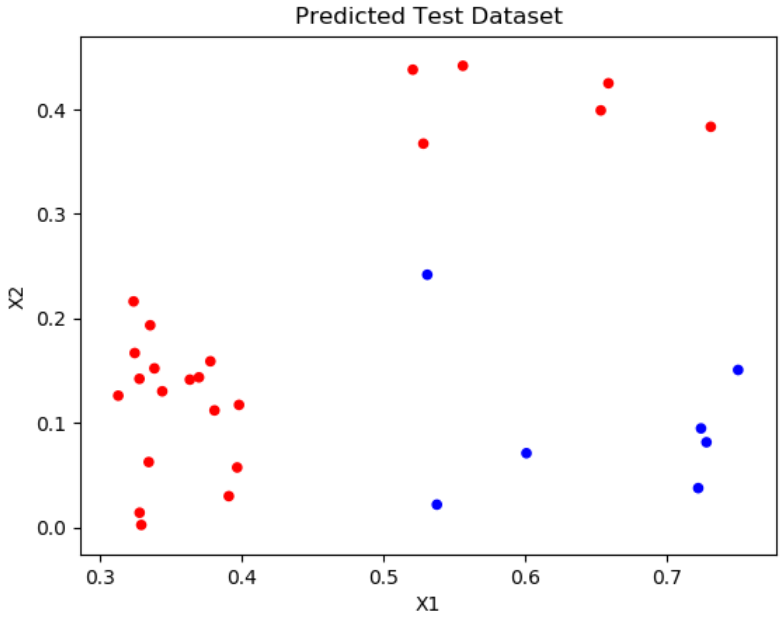
همانطور که از جدول بالا مشاهده می کنید، n_epoch = 1500 و lr = 0.3 برای اولین بار در هایپر پارامتر های تست شده به مقدار 1 رسیده ایم، که برابر مقدار بیشینه می باشد، هر یک از هایپر پارامتر های دیگری که به این مقدار رسیده اند هم مناسب می باشند؛ به دلیل اینکه مرز داده های ما به صورت غیر خطی از یک دیگر تمایز داده می شوند، بنابرایم در روش مسئله 3 با یک نورون به صورت کامل قابل تفکیک نمی باشند، در صورتی که در سوال 5 با افزایش تعداد لایه ها و همچنین نورون ها توانسته ایم مسئله را در بسیاری از هایپر پارامتر ها به طور کامل تفکیک کنیم، همچنین به دلیل اینکه در این مسئله وزن ها را به صورت Stochastic Gradient Descent به روز رسانی کرده ایم، و این روش با نوسان به جواب بهینه می رسد، به همین دلیل در ردیف هایی که به دقت 1.0 نرسیده ایم با اجرا های متفاوت دقت کمی کاهش یا افزایش می یابد، همچنین با افزایش تعداد گام ها (n_epoch) در ردیف هایی که به دقت 1.0 نرسیده ایم می توانیم دقت را به 1.0 برسانیم.

آرش حریرپوش
9731505

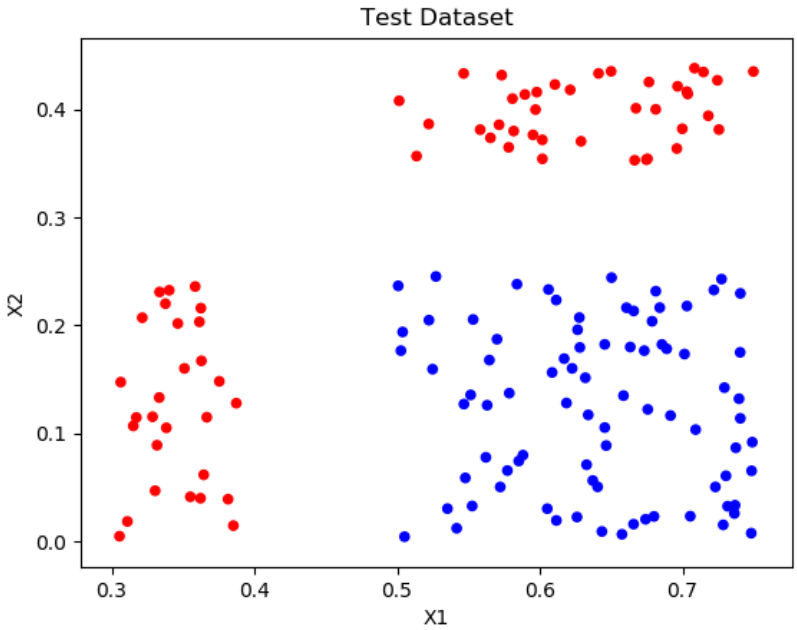
نمودار داده آزمایش را به صورت زیر رسم می کنیم:



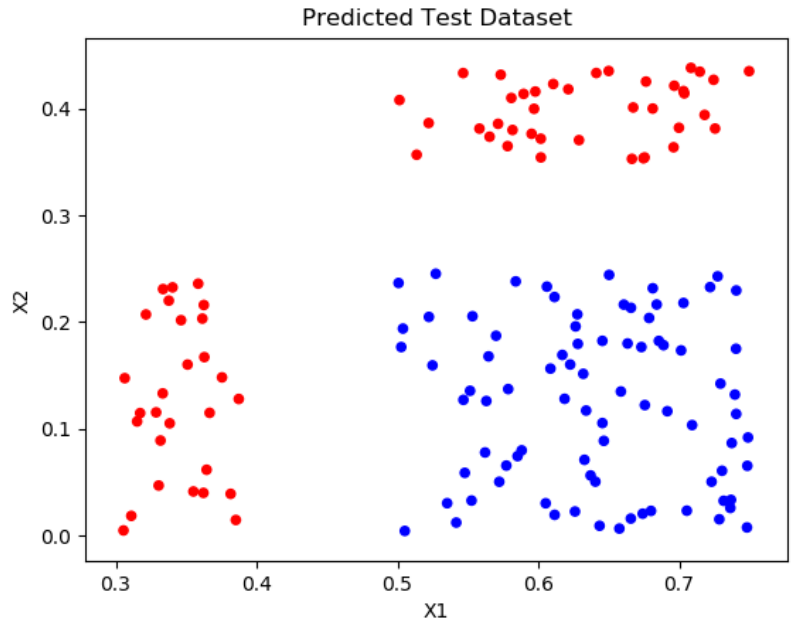
سپس نمودار داده پیشبینی شده توسط مدل آموزش دیده برای n_epoch = 1500 و lr = 0.3 به صورت زیر رسم می کنیم:



در نمودار های رسم شده، داده های با برچسب 0 را آبی و داده های با برچسب 1 را قرمز در نظر می گیریم، ابتدا نمودار داده های آموزش ورودی را به صورت زیر رسم می کنیم:



سپس پس از آموزش داده و تست کردن داده های آموزش در مدل آموزش داده شده با هایپر پارامتر های n_epoch = 1500 و lr = 0.3 به دقت 1.0 رسیده ایم، و نمودار Label های پیشبینی شده داده آموزش در مدل آموزش داده شده به صورت زیر می باشد:

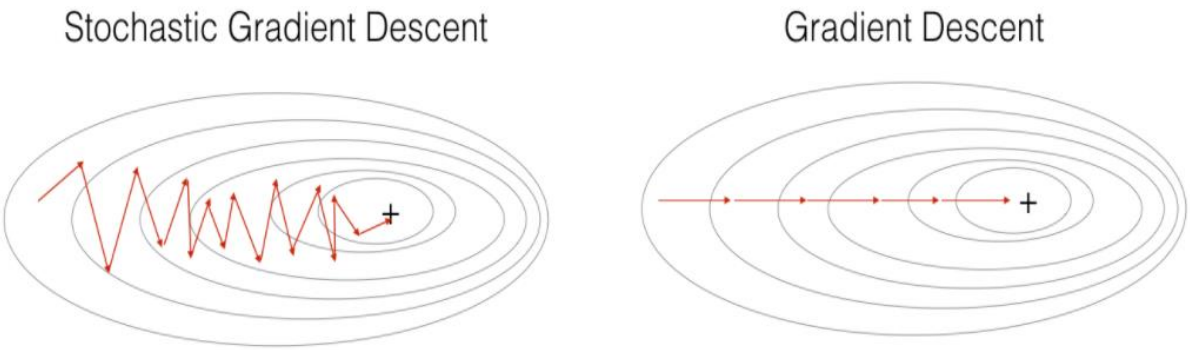


(کد این بخش در فایل های ANN_HW01_5.ipynb و ANN_HW01_5.py به نام به پیوست ارائه گردیده است.)

6. کارایی این شبکه نسبت به قبلی چه تفاوتی دارد؟ **چرا؟**

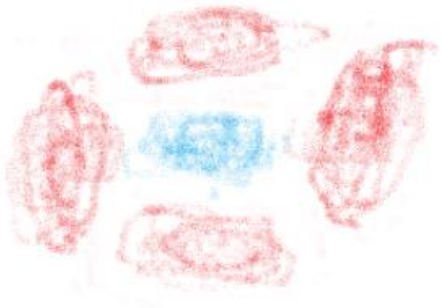
در روش قبلی برای به روز رسانی وزن ها از روش batch gradient descent استفاده کرده ایم، که در آن پس از مشاهده همه داده ها تصمیم می گیریم که چه مقدار (به اندازه میانگین گرادیان داده های مثال) وزن ها را تغییر دهیم؛ در این روش به طور تقریبا مستقیمی به مقدار بهینه مدل نزدیک می شویم؛ این در حالی است که در شبکه سوال 5 برای به روز رسانی وزن ها از روش Stochastic Gradient Descent استفاده کرده ایم، که در آن پس از مشاهده هر داده (به اندازه گرادیان آن داده) وزن ها را تغییر می دهیم. این روش هر چه داده های آموزشی بیشتری داشته باشیم شانس اینکه به مدل بهتری برسیم بیشتر می شود؛ در این روش به دلیل اینکه پس از هر بار مشاهده یک داده وزن ها را به روز رسانی می کنیم مقدار cost نوسان می کند و لزوما این نوسان به صورت کاهشی نمی باشد. اما پس از مدت طولانی اجرا، مقدار cost به صورت نوسانی کاهش یافته است.

به صورت کلی همان طور که در شکل زیر مشاهده می کنید، در روش batch gradient descent گام های کمتر اما بهینه تر و در جهت رسیدن به مسئله طی می کنیم، این در حالی است که در روش Stochastic Gradient Descent گام های بیشتر اما نه لزوما در جهت مسیر بهینه طی می کنیم اما در نهایت با مشاهده کلی گام ها در این روش مشاهده می شود که در جهت حل مسئله پیش رفته ایم.



با توجه به توضیحات داده شده در بخش بالا، مقدار دقت در روش Stochastic Gradient Descent در اجرا های مختلف نوسان می کند که این نوسان را در جدول سوال قبل نیز مشاهده کرد، همچنین زمانی که مقدار دقت مدل به ازای هایپر پارامتر های داده شده برابر 1.0 نشده مقدار دقت به ازای اجرا های متفاوت کمی نوسان می کند و کم و زیاد می شود، این در حالی است که در روش batch gradient descent این اتفاق به ندرت یا هرگز مشاهده نمی شد؛ دقت در شبکه سوال 5 از شبکه سوال 3 بیشتر است که به این دلیل است که این مسئله یک مسئله غیر خطی می باشد و به همین دلیل با یک نورون قابل حل نمی باشد، به همین دلیل با استفاده از تعداد نورون های بیشتر و افزایش لایه های شبکه در سوال 3 دقت مدل آموزش دیده در این مدل بهتر از مدل مسئله شده است.

7. فرض کنید داده های مسئله همانند شکل 5 باشد. آنگاه چه معماری یا معماری هایی برای شبکه مناسب بود. چرا؟



شکل۵- نمونه داده با دو برچسب به رنگ‌های آبی و قرمز در یک فضای دو بعدی

همانطور که از شکل بالا مشخص است، این نمونه داده به کمک یک خط قابل تفکیک نمی باشد به همین دلیل یک مسئله خطی نمی باشد و نیاز است تا برای حل آن از بیش از یک نورون عصبی مصنوعی استفاده کنیم، برای بدست آوردن این که این نمونه داده را با چه معماری می توان مدل سازی کرد که مناسب باشد، از روش زیر استفاده می کنیم:

هر شبکه عصبی یک لایه ورودی و یک لایه خروجی دارد،انتخاب تعداد نورون ها برای لایه های ورودی و خروجی ساده می باشد و تعداد نورون های لایه ورودی برابر تعداد متغیر های ورودی مسئله می باشد و تعداد نورون های لایه خروجی برابر تعداد خروجی هاییست که باید برای هر ورودی بدست آوریم (تعداد دسته های دسته بندی، اگر دو دسته داشته باشیم می توانیم از یک نورون برای خروجی داشته باشیم که تابع فعالیت آن Sigmoid می باشد یا دو نورون با تابع فعالیت softmax استفاده کنیم)

برای بدست آوردن تعداد نورون های لایه های مخفی و تعداد لایه های مخفی به صورت زیر عمل می کنیم:

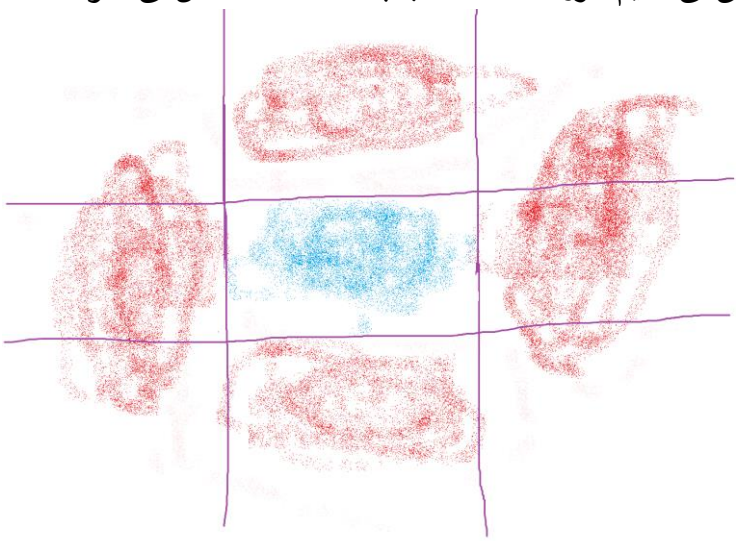
1. بر مبنای داده، تعداد خط هایی که به کمک آن می توانیم مرز های مسئله را مشخص کنیم رسم می کنیم.
2. مرز های مسئله را به صورت مجموعه ای از خط ها جدا می کنیم. توجه داشته باشید که ترکیب این خط ها باید مرز مسئله را مشخص کند.
3. تعداد خط های انتخاب شده بیانگر تعداد نورون های اولین لایه مخفی می باشد.
4. برای متصل کردن خط هایی که قبلا رسم شده است، لایه های مخفی جدیدی اضافه می کنیم. در نظر داشته باشید که یک لایه مخفی جدید هر دفعه که بین خط ها ارتباط ایجاد می کنید اضافه می شود.
5. تعداد نورون های هر لایه مخفی که اضافه می کنید برابر تعداد اتصالاتی است که در لایه مخفی جدید می خواهید ایجاد کنید.

(منبع: https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e)

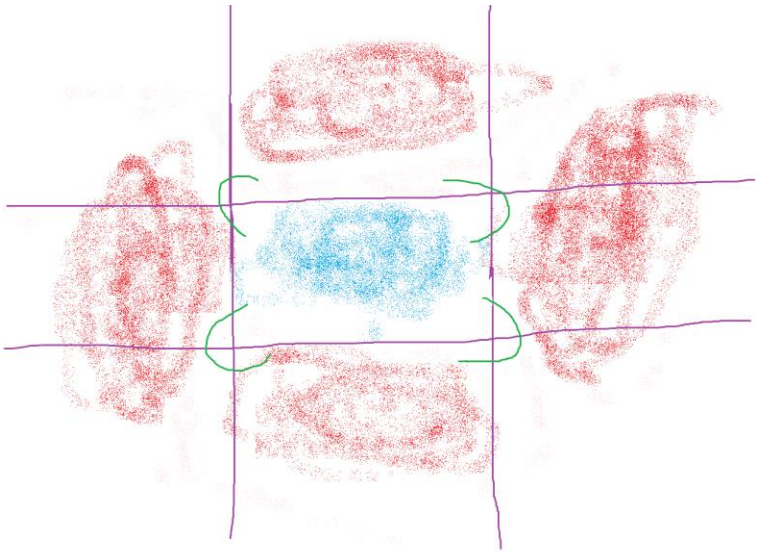
حال با توجه به گزاره های بالا می دانیم که مسئله ما دو ورودی دارد به دلیل این که در شکل عنوان شده که مسئله دو بعدی است، بنابراین به دو نورون برای لایه ورودی نیاز داریم؛ همچنین میدانیم که این مسئله دارای دو دسته می باشد که با رنگ های آبی و قرمز نمایش داده شده است، بنابراین به یک نورون با تابع فعالیت sigmoid یا دو نورون با تابع فعالیت softmax برای لایه خروجی نیاز داریم.

حال با دانستن تعداد نورون ها در لایه ورودی و خروجی تعداد لایه های مخفی و تعداد نورون های آن را بررسی می کنیم:

1. ابتدا بررسی می کنیم مرزهای مسئله با چند خط مشخص می شود:



همانطور که از شکل بالا مشخص است به چهار خط برای مشخص کردن مرز داده ها نیاز داریم، بنابراین در لایه اول مخفی 4 نورون نیاز داریم، سپس باید تعداد اتصالات این 4 خط را بدست آوریم:



از شکل بالا مشخص است که برای مشخص کردن مرز این داده ها باید این نقاط را از طریق 4 اتصال نشان داده شده به یک دیگر متصل کنیم، می توانیم این کار را با 3 نورون انجام داده و اتصال آخر در لایه خروجی انجام شود یا با 4 نورون انجام دهیم، بدین ترتیب معماری مدل مورد نظر برای این مسئله برابر شکل های زیر می باشد:

که در معماری شکل بالا تابع فعالیت لایه ورودی و لایه مخفی ReLU و لایه خروجی Sigmoid می باشد.