به نام خدا

دانشگاه صنعتی امیرکبیر

Amirkabir University
of Technology

پروژه اول درس یادگیری عمیق

آشنایی با مفاهیم اولیه واحدهای پرسپترون و آدالاین

استاد درس: دکتر صفابخش

نام: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

پاییز ۱۴۰۲

فهرست مطالب

سوال اول

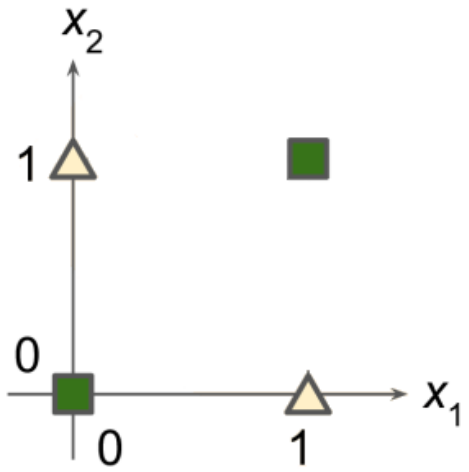
برای نگاشت یک پرسپترون منفرد برای تولید خروجی یک در زمانی که x1 و x2 متفاوت هستند (یعنی زمانی که y نزدیک به صفر است) و خروجی صفر در سایر موارد، باید وزن‌ها و بایاس را در پرسپترون تنظیم کنیم.

در مسئله داده شده می توان y را به عنوان ورودی در نظر بگیریم و اگر y از حد آستانه بزرگتر بود خروجی برابر صفر و در غیر اینصورت شبکه با دو ورودی x1, x2 آموزش می دهیم و برای این کار میتوانیم x1,x2 را به فضایی 0,1 تبدیل کنیم(اگر x1,x2 با یکدیگر برابر بودند هر دو را صفر و یا ۱ در نظر بگیریم و در غیر اینصورت یکی را صفر و دیگری را یک). برای آموزش شبکه با دو ورودی x1, x2 و تعیین وزن‌ها و بایاس میتوانیم با آن مانند گیت xor رفتار کنیم:

	عدد ۱ (ویژگی اول)	عدد ۲ (ویژگی دوم)	خروجی (برچسب)
#1	0	0	0
#2	0	1	1
#3	1	0	1
#4	1	1	0

همانطور که در بالا مشخص است در گیت xor به ازای x1, x2 نامساوی خروجی برابر ۱ و اگر x1, x2 برابر باشند خروجی برابر صفر خواهد بود. بنابراین شبکه عملکرد مدنظر را خواهد داشت یعنی اگر y از حد آستانه کوچکتر باشد، میزان شباهت x1, x2 در نظر گرفته می‌شوند و در صورتی که x1 , x2 شبیه به یکدیگر باشند خروجی برابر ۰ و در غیر اینصورت برابر با ۱ خواهد بود.

برای آموزش پرسپترون با دو مقدار x1, x2 با رسم آنها در فضای دو بعدی متوجه میشویم که امکان جداسازی آنها با استفاده از یک خط وجود ندارد.

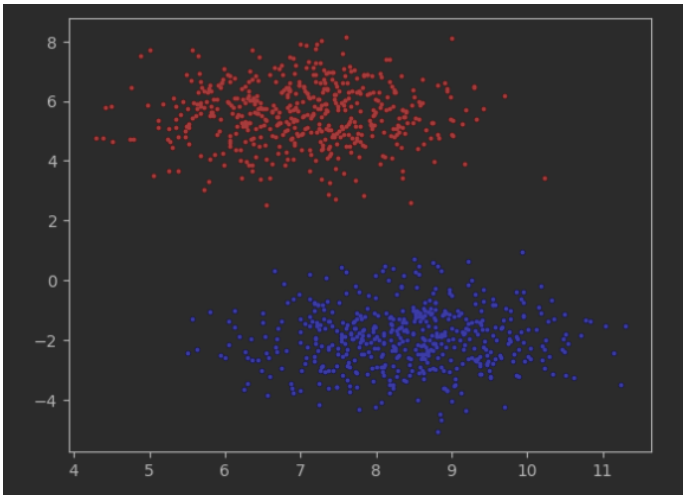


بنابراین برای آموزش پرسپترون باید به عنوان ورودی x1,x2 را با ضرایب بالاتر نیز اعمال کنیم.

در این روش x1,x2, y به عنوان ورودی در نظر گرفته می‌شود و در صورت بیشتر بودن y از حد آستانه خروجی برابر ۰ و در غیر اینصورت میزان شباهت x1, x2 مانند تابع xor در نظر گرفته میشود و در صورت متفاوت بودن خروجی برابر با ۱ و در صورت شبیه بودن خروجی برابر با ۰ است، این شبکه عملکرد مدنظر را دارد.

سوال دوم

نمایش دیتاست:



پرسپترون مجموعه‌ای از سیگنال‌های ورودی را دریافت می‌کند و اگر ترکیب خطی این ورودی‌ها از مقدار آستانه بیشتر شد فعال می‌شود، وگرنه غیرفعال باقی می‌ماند.

برای آنکه پرسپترون کار کند، نیاز است تا ضرایب w_j و سوگیری b برای آن مسئله مشخص معلوم باشند. این‌ها پارامترهای مدل هستند که باید محاسبه شوند. فرآیند آموزش با مقادیر اولیه‌ای برای پارامترها شروع می‌شود و در گام‌های بعدی به‌تدریج بر اساس ورودی‌های مختلف پارامترهای اولیه صلاح و به سمت پارامتر بهینه میل می‌کند. الگوریتم یادگیری برای حل مسائلی که هدف تشخیص تمایز دو رسته‌ای بود که به شکل خطی از هم جداشدنی هستند، کاربرد دارد.

در تابع net_input، ترکیب خطی ورودی با استفاده ازتابع dot در کتابخانه numpy محاسبه شده و با بایاس جمع میشود.

```
def net_input(self, X):
    return np.dot(X, self.w_[1:]) + self.w_[0]
```

در تابع predict به ازای مقادیر ورودی خروجی حاصل از perceptron محاسبه میشود و در آن تابع فعال ساز پله ای پیاده سازی شده است که به ازای مقادیر بزرگتر ۰ خروجی برابر با ۱ و در غیر اینصورت خروجی برابر با ۰ خواهد بود.

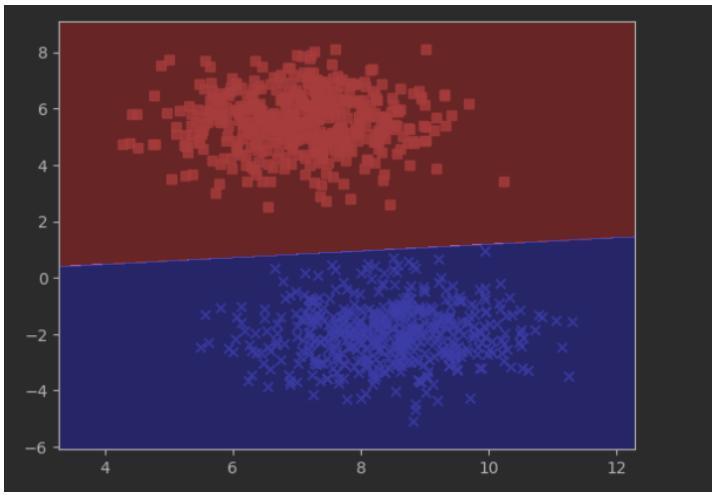
```
def predict(self, X):
    return np.where(self.net_input(X) >= 0, 1, 0)
```

در حلقه اول، به تعداد n_iter مراحل یادگیری اجرا می‌شود. در هر بار کل داده‌های آموزش یکی‌یکی وارد الگوریتم می‌شوند و خروجی حاصل از پرسپترون برای هر داده آموزشی محاسبه شده سپس به‌روزرسانی پارامترها صورت می‌پذیرد.

```
def fit(self, X, y):
    self.w_ = np.zeros(1 + X.shape[1])
    self.errors_ = []

    for _ in range(self.n_iter):
        errors = 0
        for Xi, target in zip(X, y):
            update = self.eta * (target - self.predict(Xi))
            self.w_[1:] += update * Xi
            self.w_[0] += update
            errors += int(update != 0.0)
        self.errors_.append(errors)
    return self
```

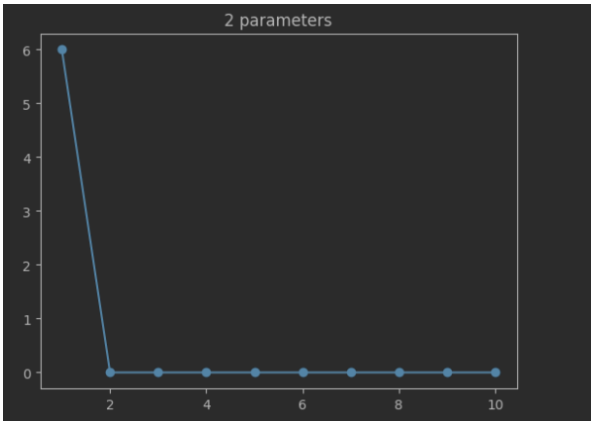
مرز تصمیم حاصل از آموزش مدل:

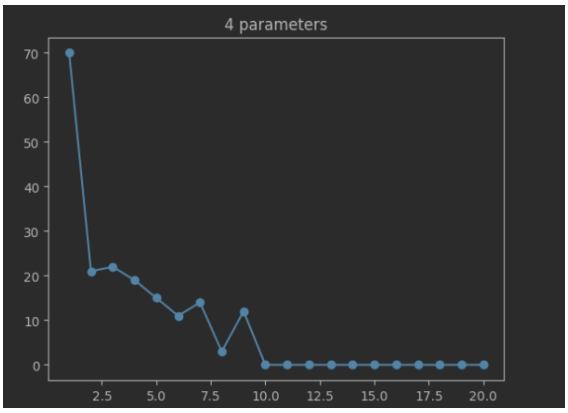


همانطور که در بالا مشخص است داده ها به صورت خطی قابلیت جداسازی دارند و تک پرسپترون به خوبی توانسته این جداسازی را انجام دهد (تعداد n_iter برابر با ۱۰ و نرخ یادگیری برابر با ۰.۰۱ در نظر گرفته شده).

با پیچیده تر کردن مدل و استفاده از پارامتر های بیشتر به عنوان ورودی، مدل با گذشت تعداد iter بیشتری تابع خطای آن برابر با صفر می‌شود و سرعت همگرایی آن کمتر میشود، همانطور که در شکل زیر مشخص است در دیتاست پیشنهادی مدل با ۲ اپک تابع خطای آن برابر با صفر می‌شود در ۴ پارامتر مدل با ۱۰ اپک تابع خطای آن صفر میشود، بنابراین افزایش پیچیدگی مدل و در نظر گرفتن حالات مختلف از x به عنوان ورودی در مواردی که میتوان آن را با تعداد کمتر مدل کرد، باعث پیچیده تر شدن و سرعت همگرایی کمتر میشود.

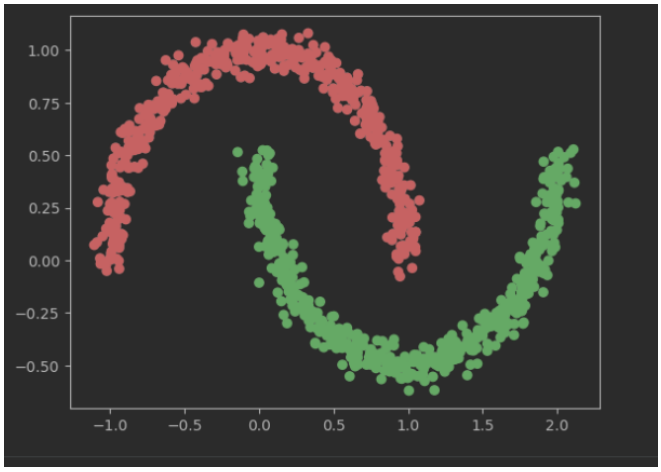
علت افزایش زمان همگرایی این است که مدل باید وزن های بیشتری را در حین فرآیند آموزش یاد بگیرد و برای آموزش آنها زمان بیشتری نیاز است در حالی که تعدادی از این وزنها برابر با صفر هستند و مدل با پارامتر های کمتری نیز عملکرد مناسبی دارد.





سوال سوم

دیتاست ورودی:



همانطور که از شکل بالا مشخص است، این مجموعه داده به صورت خط قابلیت جداسازی ندارند بنابراین برای آموزش صحیح مدل باید تعداد لایه ها بیشتر شود و یا ترکیبات بیشتری از x1,x2 به عنوان ورودی اعمال شود.

دیتاست ابتدایی به صورت زیر است:

5 rows × 2 columns

	0	1
0	-0.021371	0.406186
1	0.976700	-0.458323
2	0.904059	-0.376520
3	0.377363	-0.397037
4	-0.841926	0.530587

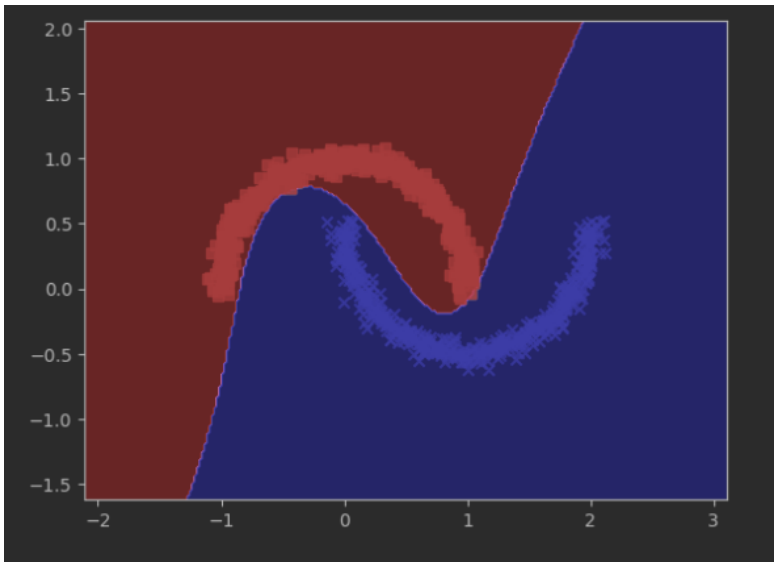
دیتاست که به عنوان ورودی برای آموزش پرسپترون استفاده میشود به صورت زیر است:

1-10 | 1,000 rows × 6 columns | np.ndarray

	0	1	2	3	4	5
0	-0.021371	0.406186	0.000457	0.164987	-0.000010	0.067015
1	0.976700	-0.458323	0.953944	0.210060	0.931717	-0.096275
2	0.904059	-0.376520	0.817322	0.141767	0.738907	-0.053378
3	0.377363	-0.397037	0.142403	0.157639	0.053738	-0.062588
4	-0.841926	0.530587	0.708839	0.281523	-0.596789	0.149372
5	0.589303	-0.321376	0.347278	0.103283	0.204652	-0.033193
6	0.292487	-0.206963	0.085549	0.042834	0.025022	-0.008865
7	-0.026378	0.447663	0.000696	0.200402	-0.000018	0.089712
8	1.620141	-0.287589	2.624856	0.082707	4.252637	-0.023786
9	0.682467	0.807121	0.465762	0.651444	0.317867	0.525794

ستون دوم (x0^2)، ستون سوم (x1^2)، ستون چهارم (x0^3)، ستون پنجم(x1^3).

در این سوال کلاس پرسپترون مانند سوال قبلی پیاده سازی شده است. مرز نهایی تصمیم گیری به صورت زیر است:



برای آموزش مرز تصمیم یر خطی به کمک یک پرسپترون، درجات بالاتری از x1,x2 به عنوان ورودی استفاده شده است.

سوال چهارم

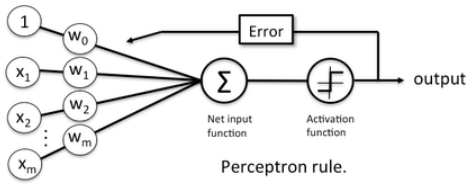
شباهت آدالاین و پرسپترون:

- دسته‌بندی کننده باینری هستند
- هر دو دارای یک مرز تصمیم گیری خطی هستند
- هر دو می توانند به صورت مکرر، یاد بگیرند (پرسپترون به طور طبیعی و آدالین از طریق نزول گرادیان تصادفی)
- هر دو از تابع آستانه استفاده می کنند

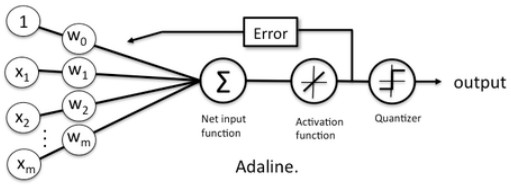
تفاوت بین پرسپترون و آدالاین:

- پرسپترون از برچسب های کلاس برای یادگیری ضرایب مدل استفاده می کند
- آدالاین از مقادیر پیش‌بینی‌شده پیوسته (از ورودی خالص) برای یادگیری ضرایب مدل استفاده می‌کند، که «قدرت‌مندتر» است زیرا نشان میدهد چقدر درست یا غلط بوده‌ایم.

بنابراین، در پرسپترون از برچسب‌های کلاس پیش‌بینی‌شده برای به‌روزرسانی وزن‌ها استفاده می‌کنیم و در Adaline، از یک پاسخ پیوسته استفاده می‌کنیم.

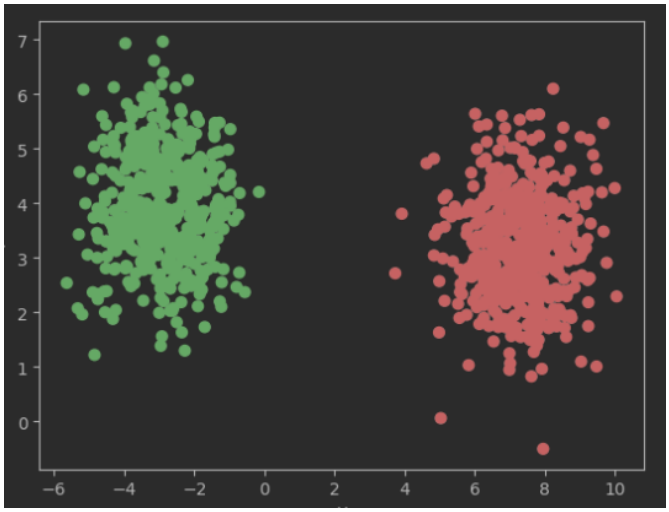


Perceptron rule.

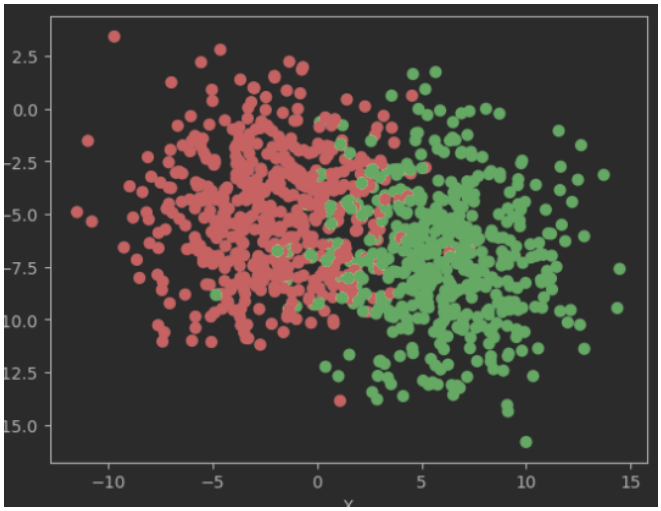


Adaline.

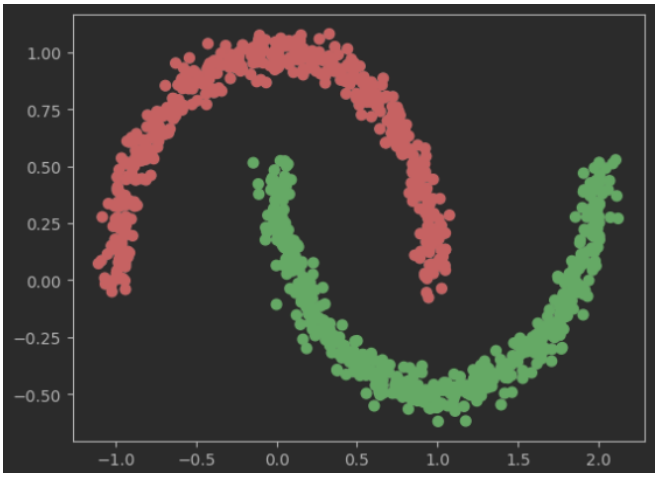
داده جداپذیر خطی (در تابعmake_blobs مقدار cluster_std برابر با ۱ در نظر گرفته شده)



داده جداپذیر شبه‌خطی (در تابعmake_blobs مقدار cluster_std برابر با ۳ در نظر گرفته شده)



جداناپذیر خطی



در این سوال کلاس پرسپترون همچون دو قسمت قبل پیاده سازی شده است، و در کلاس آدالاین در هر اپک کل مجموعه داده به عنوان ورودی در نظر گرفته می‌شود و در وزنها ضرب شده و با بایاس جمع می‌شود.

تابع fit در آدالاین به صورت زیر می‌باشد که در آن قبل از اعمال تابع فعال سازی، وزنهای شبکه به روز میشوند.

```
def fit(self, X, y):
    self.w_ = np.zeros(1 + X.shape[1])
    self.cost_ = []
    for i in range(self.n_iter):
        output = self.net_input(X)
        errors = (y - output)
        self.w_[1:] += self.eta * X.T.dot(errors)
        self.w_[0] += self.eta * errors.sum()
        cost = (errors ** 2).sum() / 2.0
        self.cost_.append(cost)
    return self
```

در شبکه داده شده دز سوال تابع fit به صورت زیر پیاده سازی شده است:

```
def fit(self, X, y):
    self.w_a = np.zeros(1 + X.shape[1])
    self.w_p = np.zeros(1 + X.shape[1])
    self.errors_a = []
    self.errors_p = []

    for _ in range(self.n_iter):
        errors = 0
        for Xi, target in zip(X, y):

            output_a = self.net_input_adaline(Xi)
            error_a = target - output_a
            self.w_a[1:] += self.eta_a * Xi.T.dot(error_a)



            self.w_p[0] = np.where(output_a >= 0, 1, -1)
            update = self.eta_p * (target - self.predict_perceptron(Xi))
            self.w_p[1:] += update * Xi

            errors += int(update != 0.0)
        self.errors_p.append(errors)
    return self
```

در تابع بالا ورودی ها به آدالاین اعمال میشود و وزنهای واحد آدالاین به روز میشود، خروجی آدالاین به عنوان بایاس پرسپترون استفاده میشود و در نهایت براساس خروجی تابع فعالسازی در واحد پرسپترون میزان خطا مشخص شده و وزنها به‌روز می‌شود.

میتوان در این شبکه وزن واحد آدالاین را نسبت به بایاس پرسپترون و با نسبت به هدف (y) بروز رسانی کرد، در به روز رسانی نسبت به بایاس پرسپترون در نتایج عملکرد بهتری نسبت به آدالاین و عملکرد بدتری نسبت به پرسپترون مشاهده شد ولی نتایج ارائه شده در ادامه به روز رسانی نسبت به مقدار هدف را بیان میکند.

با توجه به اینکه خروجی آدالاین تنها میتواند دو مقدار ۱ و -۱ را داشته باشد، خروجی این مدل میتواند به دو صورت زیر باشد:

$$w1*x1 + w2*x2 + 1$$

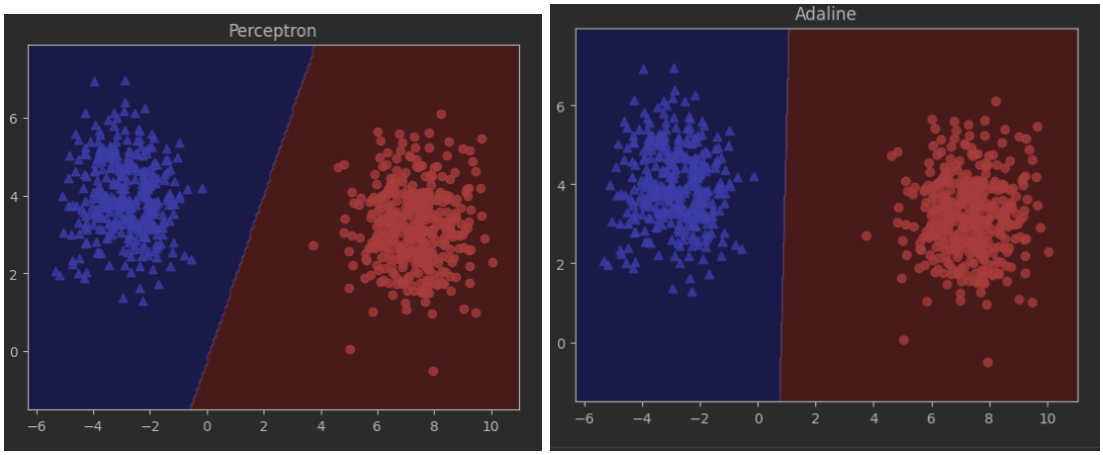
$$w1*x1 + w2*x2 - 1$$

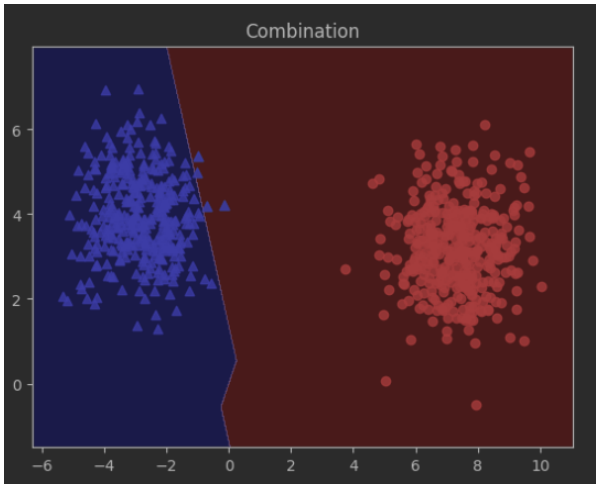
مقدار ۱ و -۱ را واحد آدالاین مشخص میکند، بنابراین مرز تصمیم میتواند از چند خط با عرض از مبدا ۱ و یا -۱ تشکیل شده باشد که این خط ها شیب یکسانی دارند.

با توجه به اینکه مدل ترکیبی در نهایت خروجی پرسپترون را ارائه میدهد عملکرد مشابهی با پرسپترون دارد و در مجموعه داده‌هایی که عرض از مبدا برای جدایی مناسب این پرسپترون میتواند دقت بالاتری ارائه دهد و زمانی که با چند خط بتوان داده ها را از یکدیگر جدا کرد مدل ترکیبی بهتر است.

مدل آدالاین به عنوان جزئی از واحد ترکیبی است و در این مجموعه داده ها همواره مدل ترکیبی دقت بالاتری دارد زیرا در درون مدل ترکیبی آموزش مدل آدالاین نیز وجود دارد.

داده‌های جداپذیر خطی





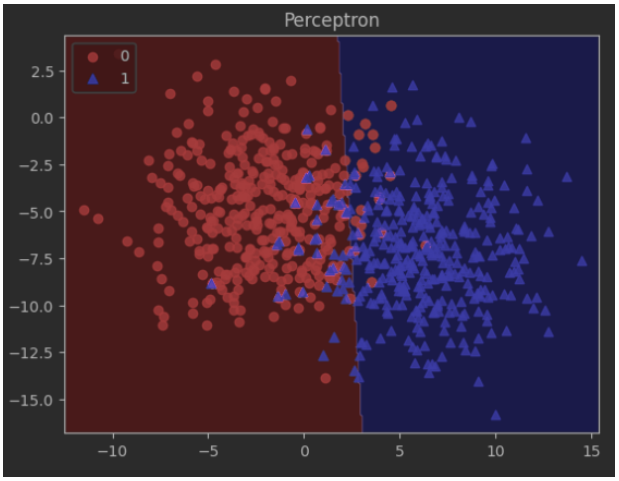
Perceptron Accuracy: 100.0%

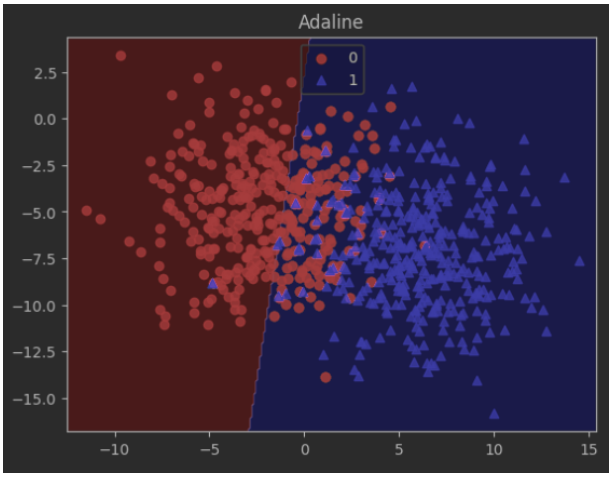
Adaline Accuracy: 100.0%

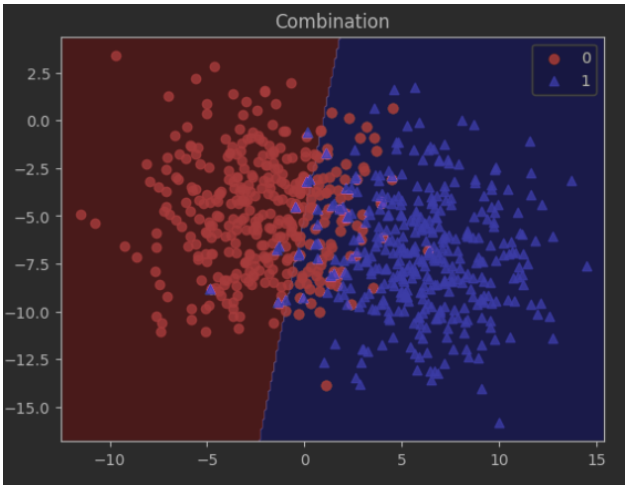
Combination Accuracy: 100.0%

دقت در هر سه مدل برابر با ۱۰۰ می باشد زیرا هر سه مدل قدرت جداسازی داده های جداپذیر خطی را دارند، البته در شکل بالا در مدل ترکیبی که برای داده های تست رسم شده بعضی از داده ها به درستی قرار نگرفته اند ولی با تعیین مناسب نرخ یادگیری میتوان این مشکل را برطرف کرد و هر سه مدل به درستی میتوانند این داده هی جداپذیر خطی را طبقه‌بندی کنند.

داده‌های جداپذیر شبه‌خطی







Perceptron Accuracy: 88.4%

Adaline Accuracy: 79.2%

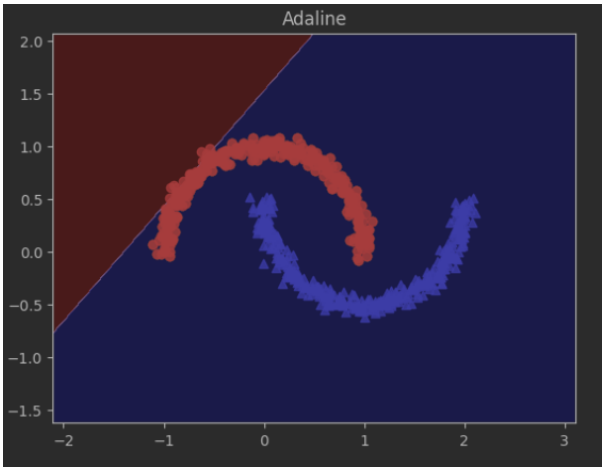
Combination Accuracy: 86.4%

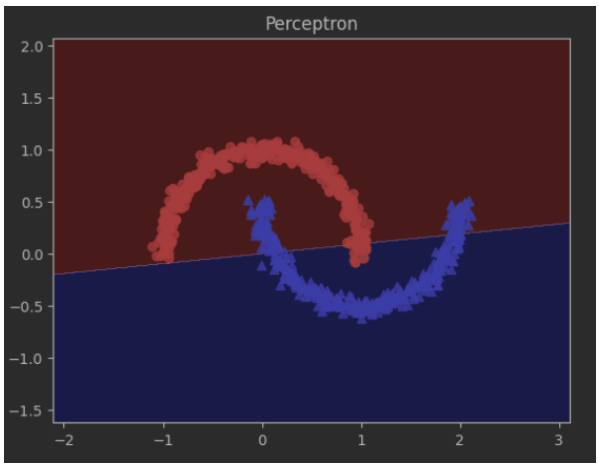
مدل ترکیبی نسبت به آدالاین عملکرد بهتری دارد زیرا در ساختار واحد ترکیبی، آدالاین است و از آن برای آموزش بایاس استفاده می‌شود و ساختار واحد ترکیبی میتواند حالت زیگزاگ داشته باشد و از خط هایی با شیب یکسان و عرض از مبدا متفاوت استفاده کند که این می تواند مرز تصمیم بهتری ارائه دهد.

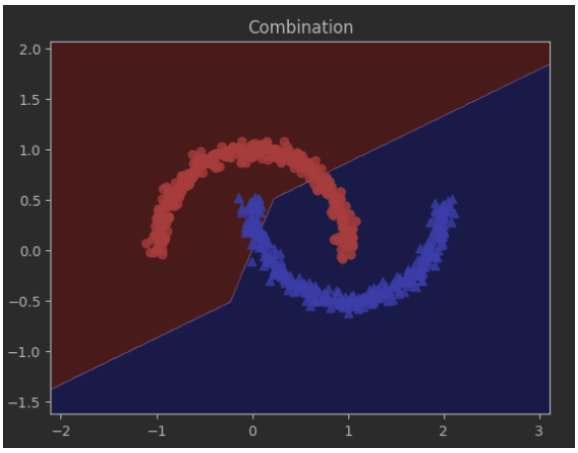
مدل ترکیبی عملکرد مشابهی نسبت به پرسپترون دارد، زیرا در نهایت خروجی واحد ترکیبی را پرسپترون مشخص میکند و این شباهت طبیعی است و برای داده های جداپذیر شبه‌خطی با توجه به اینکه عرض از مبدا پرسپترون میتواند مقادیر مختلفی را داشته باشد و مدل ترکیبی میتواند چندین خط با شیب یکسان و عرض از

مبدا ۱ و -۱ به عنوان مرز تصمیم مشخص کند، بستگی به داده و توزیع آنها عملکرد این دو مدل (ترکیبی و پرسپترون) میتواند مشابه ولی به نسبت کمی یکی از آنها با توجه به دیتاست بهتر باشد.

داده‌های جداناپذیر خطی







Perceptron Accuracy: 88.0%

Adaline Accuracy: 52.0%

Combination Accuracy: 83.2%

مدل ترکیبی در مقایسه با آدالاین عملکرد بهتری دارد و آدالاین تقریبا همه مجموعه داده را به عنوان یک کلاس در نظر گرفته و بنابراین عملکرد نزدیک به ۵۰ درصد دارد. آدالاین در مجموعه داده جداناپذیر خطی عملکرد مناسبی ندارد ولی مدل ترکیبی توانسته به خوبی کلاس داده آبی را آموزش ببیند و بقیه مجموعه داده را به عنوان قرمز

در نظر گرفته و دقت خوبی دارد و با توجه به اینکه مدل ترکیبی تنها میتواند چند خط با عرض مبدا متفاوت رسم کند در این مدل داده نتوانسته به خوبی کلاس قرمز را تشخیص دهد ولی ممکن است در داده های دیگر این قدرت را داشته باشد.

مدل ترکیبی دقت مشابهی نسبت به پرسپترون دارد و پرسپترون در این مجموعه داده توانسته با رسم یک خط با عرض از مبدا متفاوت دقت بالاتری نسبت به مدل ترکیبی (رسم چندین خط) داشته باشد.