درس یادگیری ماشین

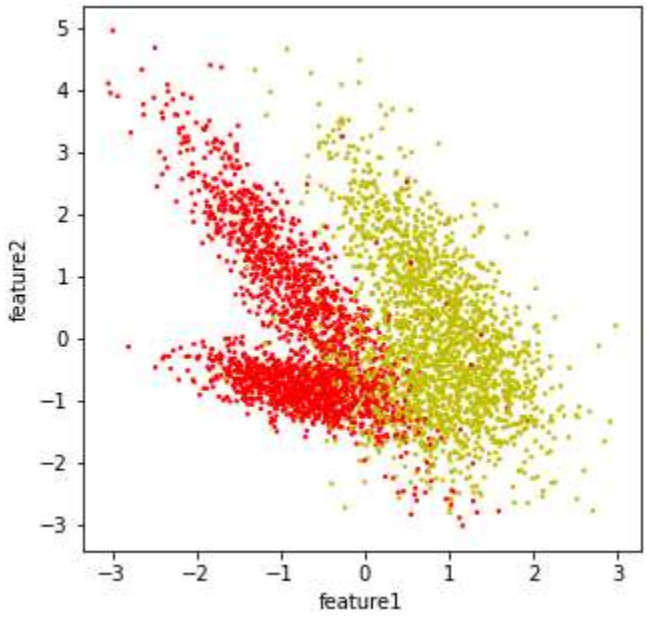
تکلیف برنامه نویسی: دسته بند پرسپترون

فایل ضمیمه : 02.ipynb

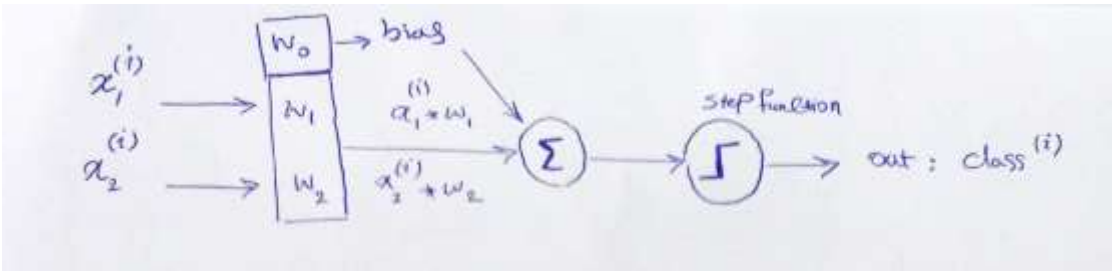
داده‌های آموزشی را از فایل train.csv به صورت یک دیتافریم کتابخانه ی pandas می‌خوانم. برای بخش کوچکی از محاسبه، کتابخانه ی numpy هم اضافه شد و از matplotlib.pyplot هم برای رسم نمودارها استفاده کردم. ماتریس داده های مساله شامل سه ستون است که دو ستون اول همان $x^{(i)}$ ها هستند و مقدار اسمی نظیر ستون سوم برچسب داده ها ست. ۴۰۰۰ داده ی اموزشی در اختیار هست. مساله دوکلاسه است و محتویات ستون label نظیر هر داده مقداری برابر ۰ یا ۱ دارد. برای راحت تر شدن روند کار این مقادیر را به ۱ و -۱ نظیر کردم.

نمودار نقطه ای داده های آموزشی را رسم می‌کنم. گروهی که برچسب -۱ دارند نقاط قرمز و نقاط زرد نظیر داده های با برچسب ۱ است.

محور عمودی نظیر مقدار ستون دوم داده ها یا feature2 یا همان $x_2^{(i)}$ روی محاسبات است و محور افقی هم نظیر مقدار ستون دوم داده ها یا feature1 یا همان $x_1^{(i)}$ در جریان محاسبات.



مدل خواسته شده یک پرسپترون به صورت زیر است : (تابع پله ی استفاده شده همان تابع علامت است)



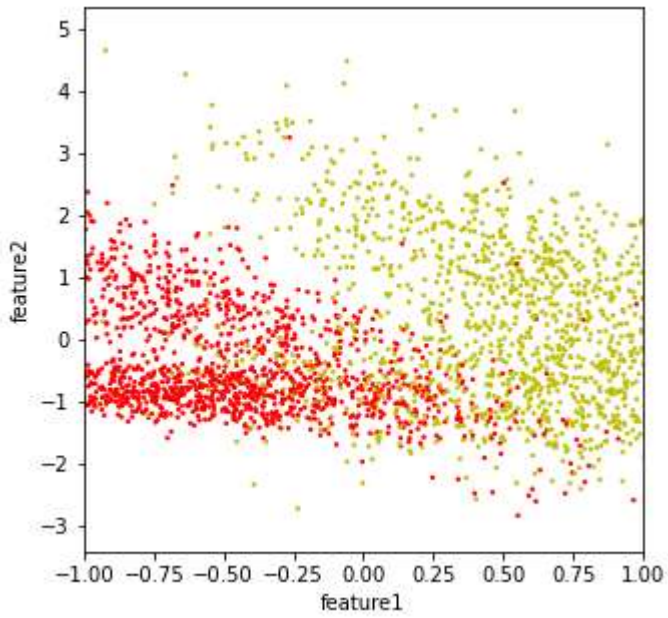
یعنی یک معادله ی خط به صورت زیر قرار است پیدا کنیم که اگر روی این نمودار رسم شود، دو کلاس داده ها را به صورت زیر از هم کاملا تفکیک کند.

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

$$\forall x = [x_1, x_2] \quad \text{if} \quad w_0 + w_1 x_1 + w_2 x_2 \geq 0 \quad \text{class yellow : 1}$$

$$\text{else} \quad w_0 + w_1 x_1 + w_2 x_2 < 0 \quad \text{class red : -1}$$

اگر به نمودار دوباره نگاه کنم و مخصوصا با دقت در برش زیر از دامنه ی feature1 به سادگی معلوم می‌شود که چنین خطی وجود ندارد. یعنی این داده ها به طور خطی تفکیک پذیر نیستند.



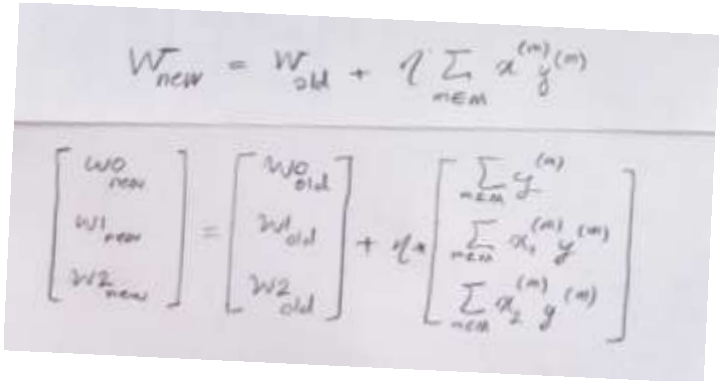
پس مدل خطی هرگز روی داده های آموزشی به دقت ۱۰۰ نمی‌رسد (همه را درست کلاس بندی کند) پس اگر الگوریتم تیوری پرسپترون را استفاده کنم که شرط توقفش تا زمانی است که هیچ داده ای در دسته ی اشتباهی نباشد، الگوریتم هرگز متوقف نمی‌شود. برای اینکه جواب نزدیکی برای مساله با مدل خطی پیدا کنم، قراردادی الگوریتم اپدیت وزن ها را تا مقدار مشخصی تکرار اجرا می کنم. اینجا من ۱۰۰ انتخاب کردم.

برای راحتی بیشتر محاسبات بایاس مدل، یک ستون ۱ به ماتریس داده ها اضافه کردم. که روی محاسبه مقدار $x_0^{(i)}$ را می‌گیرد.

حالا با استفاده از داده هایی که اماده شد، الگوریتم یادگیری پرسپترون را اجرا می کنم. حاصل کار، یک بردار سه مولفه ای w است. این همان خطی است که انتظار داشتیم داده ها را در دو طرف خود تفکیک کند.

یاداوری اینکه زمانی که خواستم روی داده های تست عملکرد مدل را بسنجم همین تغیرات باید روی ان ماتریس داده ها هم اعمال بشود.

حلقه ی اصلی آموزش در تابع train_perceptron پیاده شده که ماتریسی که آماده شد را می‌گیرد و تا زمانی که تمام داده ها درست طبقه بندی شوند یا تا حداکثر ۱۰۰ بار بردار وزن های پرسپترون را از رابطه ی زیر آپدیت می‌کند. که M مجموعه ی داده‌های اشتباه کلاس بندی شده است و y هم همان مقدار label هست.

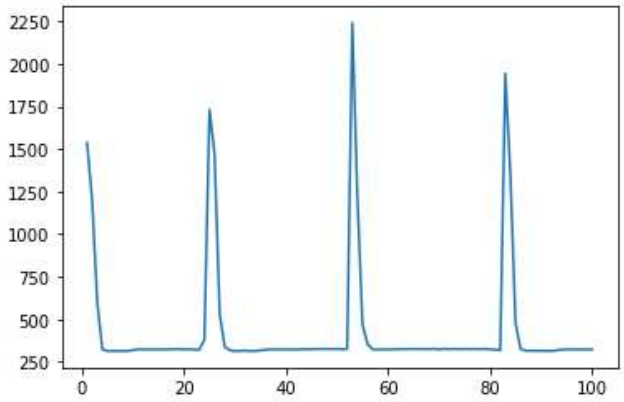


مقدار اولیه وزن ها را تصادفی انتخاب کردم. هر بار با بردار وزن به دست آمده از مرحله ی قبل، با تابع classify روی کل داده های آموزشی کلاس بندی انجام می‌دهد و نتیجه ی کلاس به دست آمده از رابطه‌ی زیر، روی یک ستون به نام assigned_label از دیتافریم اموزشی ذخیره می‌شود.

در قدم بعدی لازم است این ستون را با ستون label مقایسه کرده و نقیض نتیجه را روی ستون دیگری به نام missclass قرار دهد. این کار در تابع find_missed انجام می‌شود. کل داده هایی که روی این ستون مقدار صحیح دارند، برای آپدیت وزن در تکرار فعلی حلقه استفاده می‌شوند. در واثع دارم از الگوریتم دسته ای (batch mode) برای الگوریتم آموزش پرسپترون استفاده می‌کنم. برای ساختن مقداری که باید به مولفه های w اضافه و کم شود از یک دیتافریم موقتی استفاده می‌کنم و در نهایت با نرخ یادگیری که برابر 0.001 تعین شده w را آپدیت می‌کنم. برای کنترل روند آموزشی تعداد داده هایی که اشتباه کلاس بندی شده اند در هر تکرار نگهداری می‌شود. وزن هایی که به دست آمدند :

W0	W1	W2
0.063877	0.0834247	0.226945

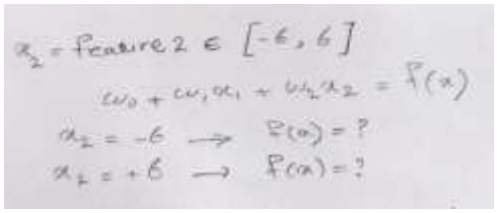
روی ۱۰۰ بار روند تکرار الگوریتم تعداد داده هایی که هربار اشتباه کلاس بندی شدند را رسم می‌کنم.



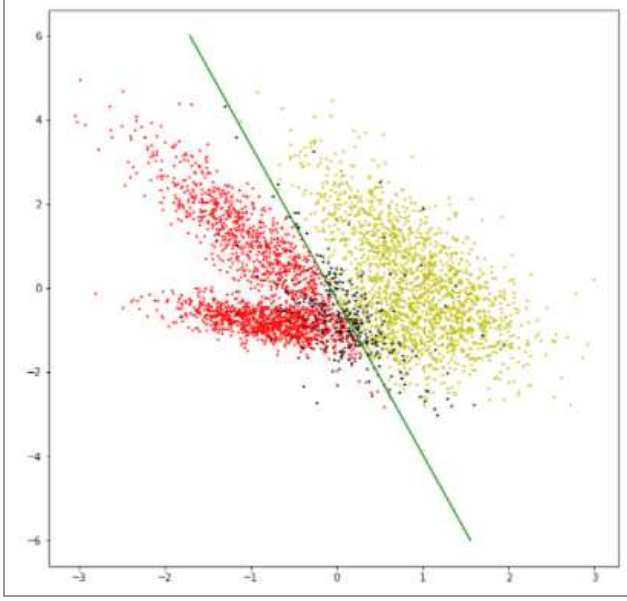
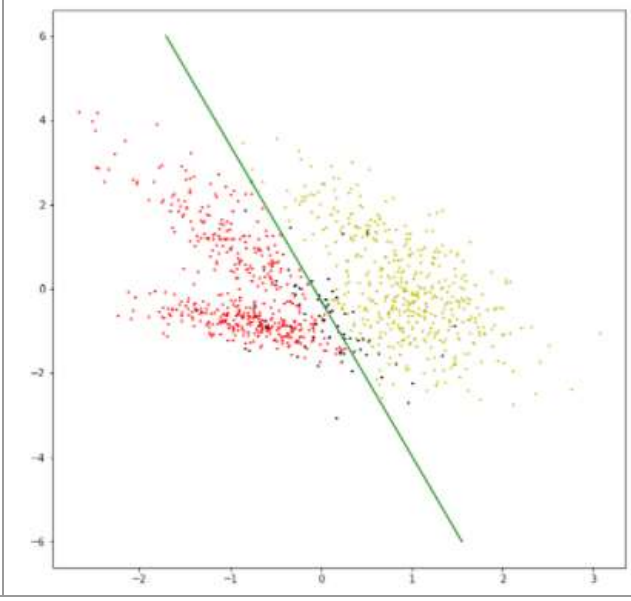
نمودار پرش ها و فرود های ناگهانی دارد. در تکرار های خاصی اگر اموزش متوقف شود خطای داده آموزش بسیار بالاست. این وضعیت به دلیل همان خطی تفکیک پذیر نبودن داده ها رخ می‌دهد. و در سایر نقاط نمودار با شیب خیلی کمی حول عرض کمتر از ۵۰۰ تا داده ی خطا می‌ماند. مدل در بهترین حالت روی داده‌های آموزشی این تعداد خطا دارد. این خطا ناشی از پیچیدگی کم مدل است و با این مدل قابل رفع نیست.

داده های تست را از روی فایل test.csv به صورت یک دیتا فریم خواندم و تغیرات لازم را انجام دادم.

برای ارزیابی مدل روی ماتریس داده های اموزش و تست، یک باز هر دو ماتریس را با مدل به دست آمده طبقه بندی می‌کنم. بعد از ساخته شدن ستون‌های assigned_label و missclass نتیجه ی طبقه بندی را با دو تابع measurments و plot_result ارزیابی می‌کنم.
تابع measurments اطلاعاتی از عملکرد مدل به دست می دهد. این تابع تعداد کل داده هایی که اشتباه دسته بندی شده اند و نسبت آن ها به کل داده ها را و میزان دقت مدل روی آن داده های خواسته شده را گزارش می‌کند. اینجا دقت را برابر نسبت داد هایی که درست کلاس بندی شدند به کل داده ها گرفتم (همان correct_ratio)
تابع plot_result برای نمایش داده های دو کلاس بر اساس رنگ کلاسشان روی ماتریس داده ی ورودی تعریف شده. داده هایی که اشتباه کلاس بندی شده اند را به رنگ سیاه نمایش داده. روی همین نمودار یک خط سبز رنگ هم رسم می‌شود که نشان دهنده ی مقدار w نهایی به دست آمده ست. (این خط را از اعمال وزن های w روی دو نقطه ی ابتدا و انتهای محدوده ی محور عمودی رسم کردم)



نتایج در ادامه آمده.

train	test
total = 4000 total_missclassed = 323 misclassed_ratio = 0.08075 correct_ratio = 0.91925	total = 1000 total_missclassed = 74 misclassed_ratio = 0.074 correct_ratio = 0.926
	

همانطور که از شکل نمودار اول گزارش انتظار می‌رفت بیشتر داده هایی که اشتباه کلاس بندی شده اند متعلق به همان بخش در هم رفتگی دو کلاس هستند که از رویش نتیجه گرفتیم مدل خطی نمی‌تواند جدا کننده ی کامل داده های آموزشی باشد. اگر به خاطر درهم رفتگی ذاتی این دو کلاس روی صفحه نبود، مدل می‌توانست طبقه بندی بهتری انجام دهد..

کل داده های تست و آموزش به تزتیب ۱۰۰۰ و ۴۰۰۰ تا هستند که روی هر کدام باز هم به ترتیب ۷ و ۸ درصد خطا داریم. که برای اسن داده‌ها عدد ثابل قبولی هم هست. همانطور که از نمودار أبی رنگ تعداد کلاس بندی های اشتباه در روند آموزش انتظار می‌رفت تعداد کل خطاهای آموزش عددی زیر ۵۰۰ برابر ۳۲۳ به دست آمده. اگر درصد کل های درست طبقه بندی شده روی تست را به عنوان دثت نهایی در نظر بگیریم، این مدل ۹۲ درصد داده ها ی این مساله را می‌تواند درست طبقه بندی کند که عدد قابل ثبولی است. اما اگر توزیع داده های تست طوری بود که بیشتر شامل نقاط درهم رفته ی دو کلاس میشد، عملکرد افت می‌کرد اما ما داده های تست و آموزش را همیشه از یک توزیع یکسان انتخاب می کنیم و علت نزدیک بودن نتیجه ی تست و آموزش هم در واقع همین است. اینجا مدل نهایی به لحاظ عددی هرچند عملکرد خوبی داشته ولی اگز به نمودار ها نگاه کنیم واضح است که از مدل درستی استفاده نکردیم و من نتیجه ی عددی نسبتا مطلوب را به این نسبت می‌دهم که هرچند دو کلاس درهم رفتگی دارند اما این درهم رفتگی روی توزیع داده ها عدد کوچکی است (نمودارهای اول گزارش) که ممکن است بتوان آن را حتی به زیاد بودن نویز داده ها نسبت داد اما چون نویز را یک فاکتور رندوم می‌گیریم، با حرف هایی که درباره تشابه توزیع (صرفا به لحاظ ظاهری و نه بررسی آماری) این داده های درهم (سیاه رنگ روی نمودار) روی دو دسته ی داده ها زدم، نظر من این است که مساله ی اصلی کم بودن پیچیدگی مدل پرسپترون خطی برای طبقه بندی و یادگیری این داده هاست. با ساختن مدل های درجه بالاتر می‌توان این فرض را بررسی کرد.