به نام خدا

دانشگاه امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه اول

شبکه های عصبی

خدیجه ساعدنیا

۹۴۱۳۱۰۵۹

فهرست مطالب

بخش اول:

تعاریف اولیه

۱-۱ تعریف پرسپترون:

پرسپترون یک نوع دسته بندی کننده دودویی است که ورودی خود x (یک بردار از نـوع اعـداد حقیقـی) را بـه مقدار خروجی g(x) که به صورت زیر حساب می‌شود، متناظر می‌کند:

$$g(z) = \begin{cases} 1 \ if \ z \geq 0 \\ -1 \ otherwise. \end{cases}$$

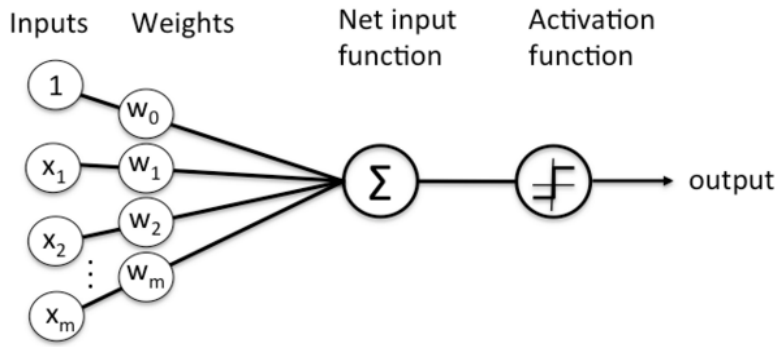
علامت g(z) برای دسته بندی z به نمونه مثبت یا منفی، در مسایل دسته بندی دودویی استفاده می‌شود.

$$\mathbf{z} = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{j=0}^{m} x_j w_j$$
$$= \mathbf{w}^T \mathbf{x}.$$

۱-۱-۱ الگوریتم یادگیری پرسپترون:

1. Initialize the weights to 0 or small random numbers.
2. For each training sample $x^{(i)}$:
    1. Calculate the *output* value.
    2. Update the weights.

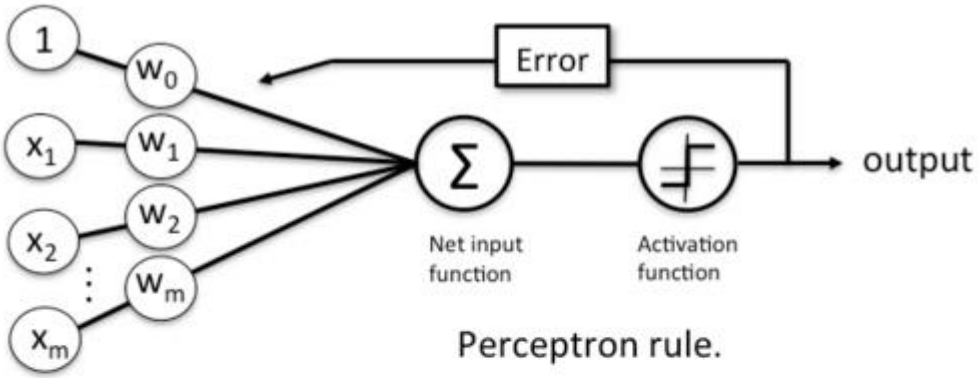
$$\Delta \mathrm{wj} = \eta(\mathrm{target}^{(i)} - \mathrm{output}^{(i)})\mathrm{x}^{(i)}\mathrm{j}$$



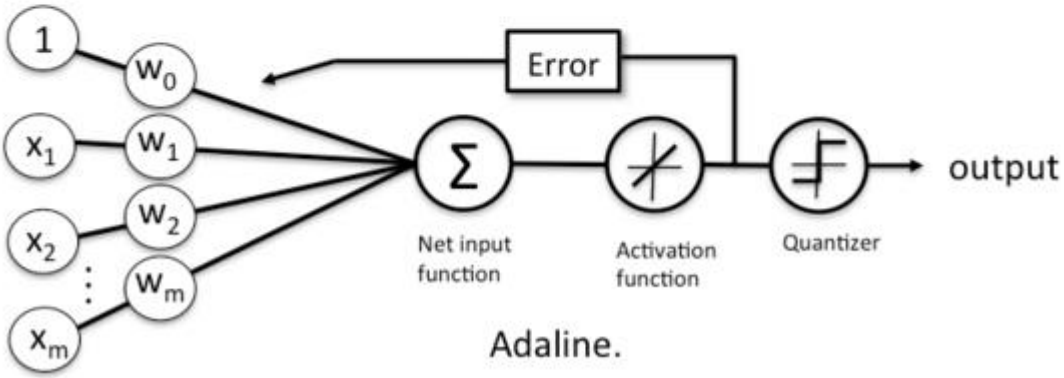
Schematic of Rosenblatt's Perceptron

۱-۲ تعریف آدالاین:

بر خلاف پرسپترون، قانون دلتا آدلاین, وزن ها را بر اساس تابع فعال ساز خطی آپدیت می کند.



Perceptron rule.



Adaline.

برای محاسبه خطا از رابطه زیر استفاده می کنیم:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i}(\text{target}^{(i)} - \text{output}^{(i)})^2 \qquad \text{output}^{(i)} \in \mathbb{R}$$

۱-۳ تعریف پرسپترون درجه دوم:

این الگوریتم مشابه الگوریتم پرسپترون خطی می باشد با این تفاوت که علاوه بر خود ورودی ها مرتبه دوم آنها نیز در معادله حضور دارند، به این ترتیب برای دو ورودی x_1 و x_2 ما نیاز به ۵ وزن اولیه داریم. یعنی معادله جدا کننده کلاس ها به صورت زیر در می آید:

$$I = w_0 b + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$$

سایر بخش های این الگوریتم مشابه پرسپترون خطی می باشد.

در واقع در این روش دسته بندی کننده ی ما مرتبه دوم می باشد، پس می توان به کمک آن کلاس هایی که قابل جدا سازی خطی نمی باشند را دسته بندی کرد.

معادله دسته بندی کننده بر اساس وزن های ایجاد شده می تواند معادله ی یک سهمی، هذلولی، دایره و یا بیضی باشد.

بخش دوم:

سوال اول

۲-۱ پرسپترون:

۲-۱-۱ پیاده سازی پرسپترون:

همانطور که بیان کردیم این الگوریتم توانایی جدا سازی دو کلاس را دارد، با توجه به اینکه در این مسئله سه کلاس داریم باید در فاز آموزش هر بار یک کلاس را از بقیه تفکیک کنیم. در این پیاده سازی شماره کلاس ها به ترتیب زیر قرار داده شده است:

Class 0 = [-1,-1,-1]

Class 1 = [-1,1,-1]

Class 2 = [-1,-1,1]

به این ترتیب با هر بار اجرای فاز آموزش می توان تابع خطی جداکننده بین هر دو کلاس را پیدا کرد. اما در این مسئله همه کلاسها قابل جداسازی خطی نمی باشند و تنها کلاس شماره ۰ را می توان با خطای ۰ از سایر کلاسها جدا کرد.

برای محاسبه خطا در این الگوریتم در انتهای هر ایپک آموزشی پس از آنکه خروجی نهایی توسط تابع علامت تعیین شد، تعداد خطاها به ازای هر کلاس شمارش می شود و در آرایه ای ذخیره می شود.

داده های ورودی به دو دسته ی Training و Validation تقسیم می شوند که در Ui می توان درصد هر کدام را تعیین کرد.

همزمان با فاز آموزش پس از هر ایپک، با استفاده از وزنهای جدیدی که تولید شده است، داده های Validation هم تست می شوند و خطای Validation نیز تولید می شود.

شرایط اتمام آموزش به قرار زیر می باشد:

۱- تعداد مشخصی ایپک آموزشی طی شود.
۲- خطای آموزشی به صفر و یا عددی که تعیین کرده ایم برسد.
۳- Over Fit رخ دهد.

Over Fit: زمانی که آموزش شروع می شود بـا هربـار اجـرای الگـوریتم خطـای Training و Validation خطاها رو به کاهش است تا به نقطه ای می رسیم که خطای Validation شروع به افزایش میکند کـه بـه آن Over Fit گفته می شود.

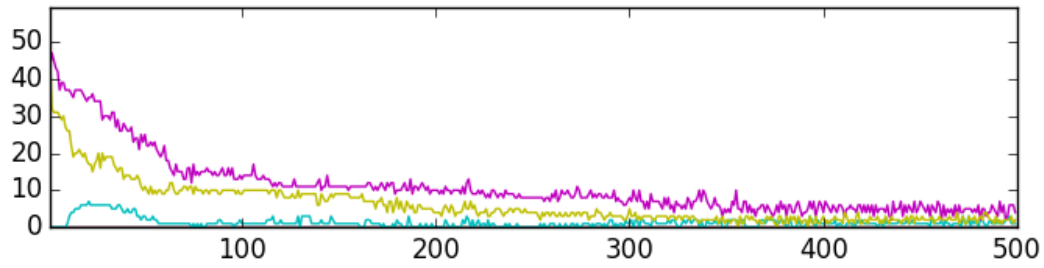
۲-۱-۲ نتایج:

با توجه به تعداد خطاهای شمارش شده در پایان هر ایپک می بینیم که تنها خطا برای کلاس شـماره ی صـفر، می تواند صفر بشود پس این کلاس قابل جدا سازی خطی از سایر کلاس ها می باشد.

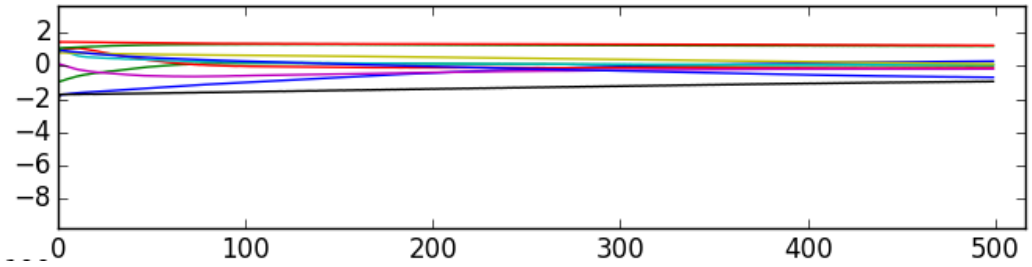
همانطور که مشاهده می شود با افزایش ضریب آلفا خطاها با سرعت بیشـتری کـاهش مـی یابنـد و همگـرا مـی شوند.

اجرای برنامه برای سه مقدار مختلف ضریب آلفا به صورت زیر می باشد.

Alfa = 0.001:

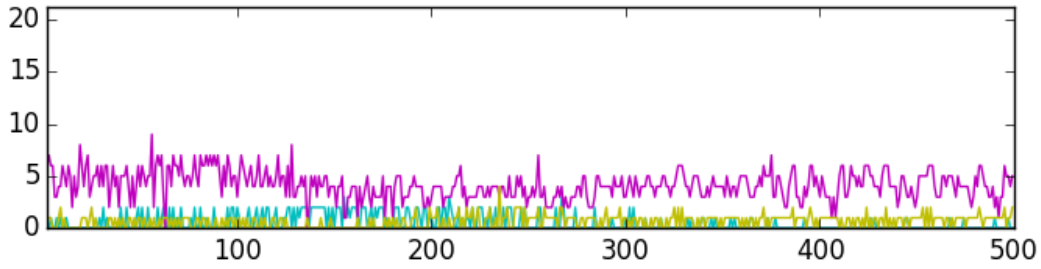


نمودار خطا در هر ایپک آموزشی

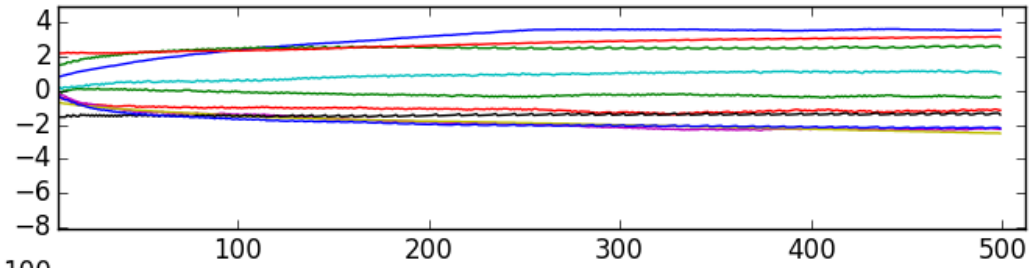


نمودار تغییر وزن در هر ایپک آموزشی

Alfa = 0.01:

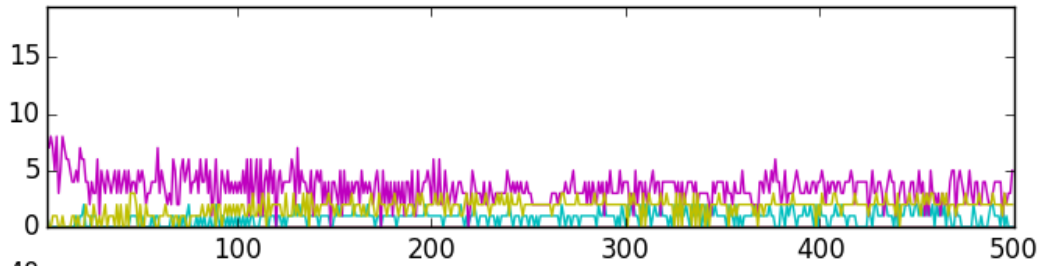


نمودار خطا در هر ایپک آموزشی

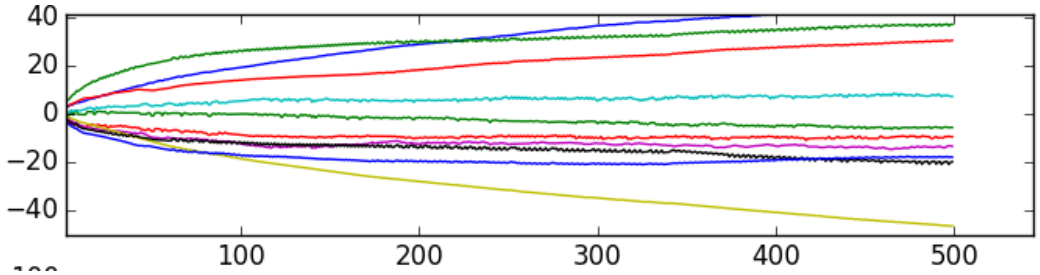


نمودار تغییر وزن در هر ایپک آموزشی

Alfa = 0.1:



نمودار خطا در هر ایپک آموزشی



نمودار تغییر وزن در هر ایپک آموزشی

۲-۲ آدلاین:

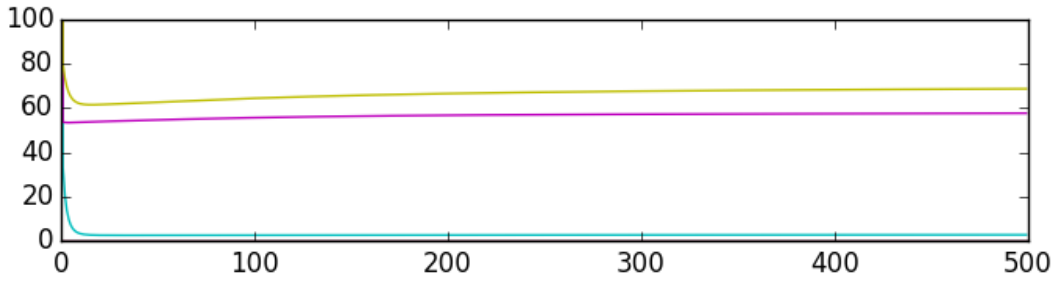
۲-۲-۱ پیاده سازی برنامه:

برای پیاده سازی این الگوریتم یک کلاس از کلاس پرسپترون ساخته می شود وتوابع Train و Validation آن Over Write می شود. در واقع تفاوت پرسپترون و آدلاین تنها در روش محاسبه خطا و اصلاح وزنها می باشد.

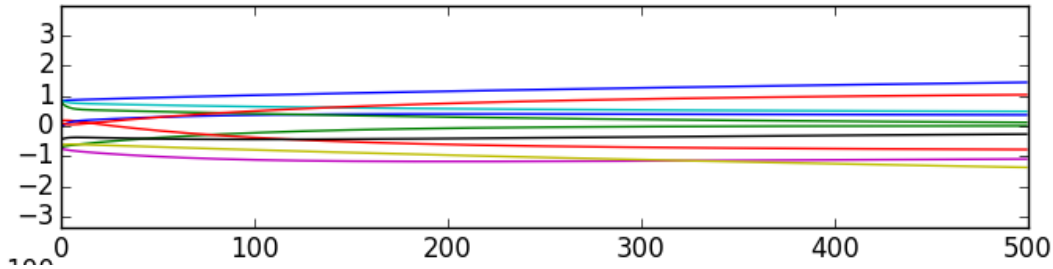
۲-۲-۲ نتایج:

همانطور که می دانیم آدلاین سرعت بسیار کمی دارد، با افزایش آلفا می توانیم سرعت آن را بهبود دهیم، از طرفی به علت مشکل مینیمم محلی اگر ضریب آلفا خیلی بزرگ شود دیگر همگرا نمی شود.

Alfa = 0.001:



نمودار خطا در هر ایپک آموزشی



نمودار تغییر وزن در هر ایپک آموزشی

بخش سوم:

سوال دوم

۳-۱ پیاده سازی برنامه:

برای این سوال با توجه به این که تنها دو کلاس داریم و امکان جداسازی خطی وجود دارد، به راحتی می توانیم از همان پیاده سازی سوال اول استفاده کرده و خط جدا ساز بین آنها را رسم کنیم.

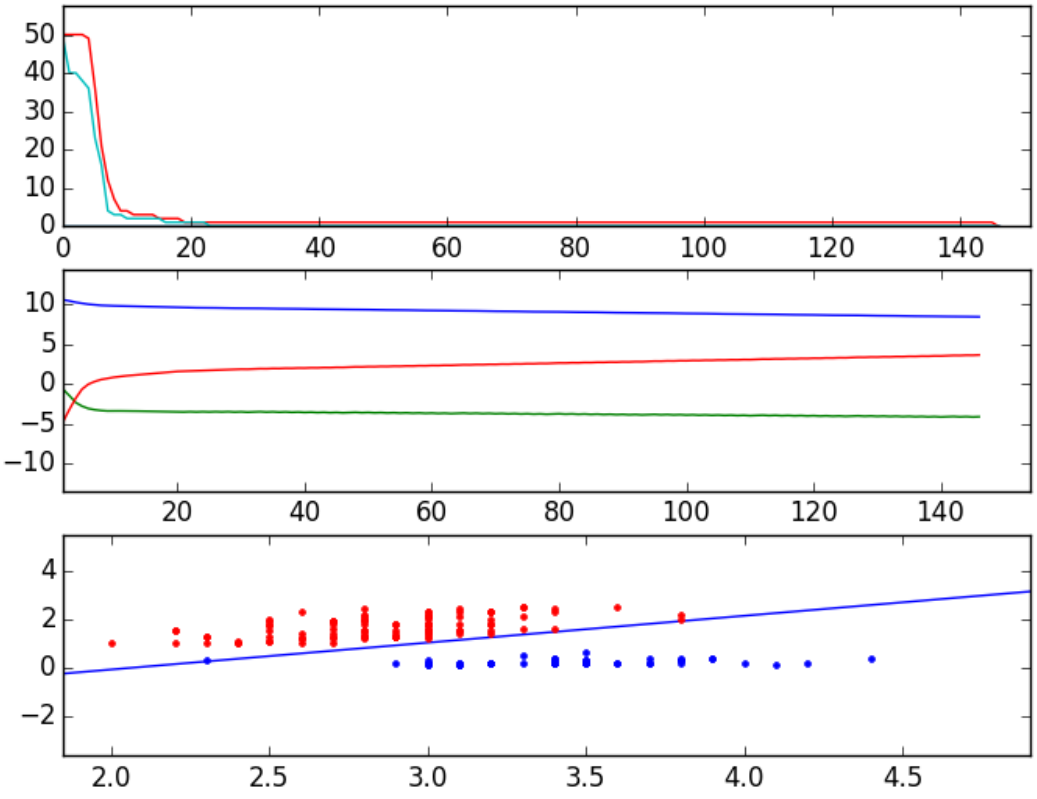
برای رسم نمودار در این مسئله پس از اتمام هر ایپک براساس وزن های ایجاد شده معادله خط آن را نوشته و رسم می کنیم.

۳-۲ نتایج:

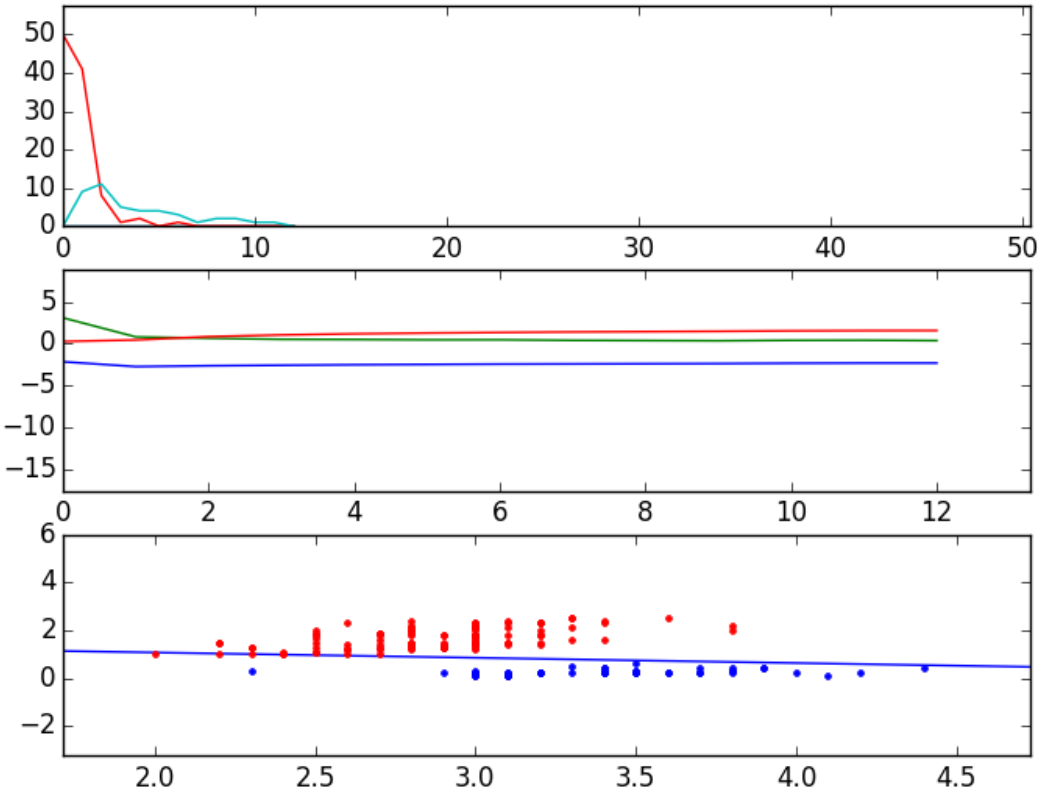
در شکل های زیر، نمودار اول تغییرات خطا در هر ایپک، نمودار دوم تغییرات وزن در هر ایپک و نمودار سوم داده های ورودی به همراه خط جدا ساز آنها را نمایش می دهند.

در حالت اول وزنهای اولیه خیلی با وزنهای مطلوب فاصله داشته به همین علت تعداد ایپک های آموزشی زیاد است، اما در حالت آخر به علت نزدیک بودن وزنها به مقدار مورد انتظار تنها در دو ایپک آموزش خاتمه یافته است.

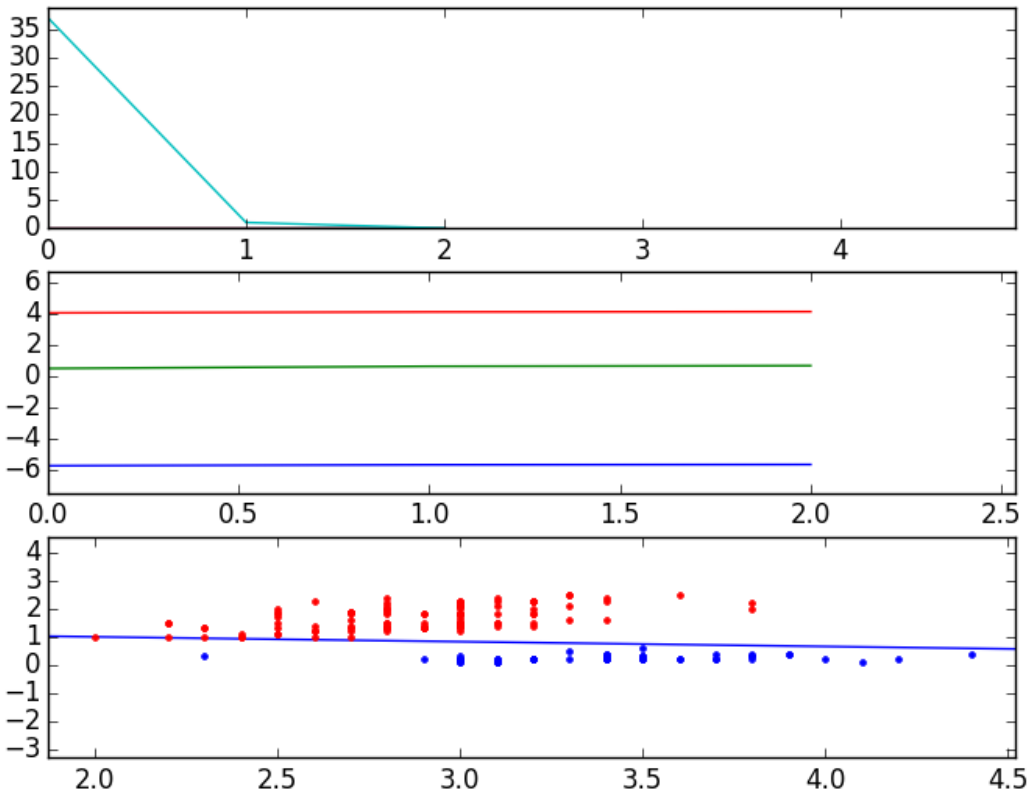
First Weights = [10.87861087, 1.36403839, -9.67000002]:



First Weights = [-1.18428246, 6.49017601, 0.48276697]:



First Weights = [-6.4429813, -1.5696077, 2.96128179]:



بخش چهارم:

سوال سوم

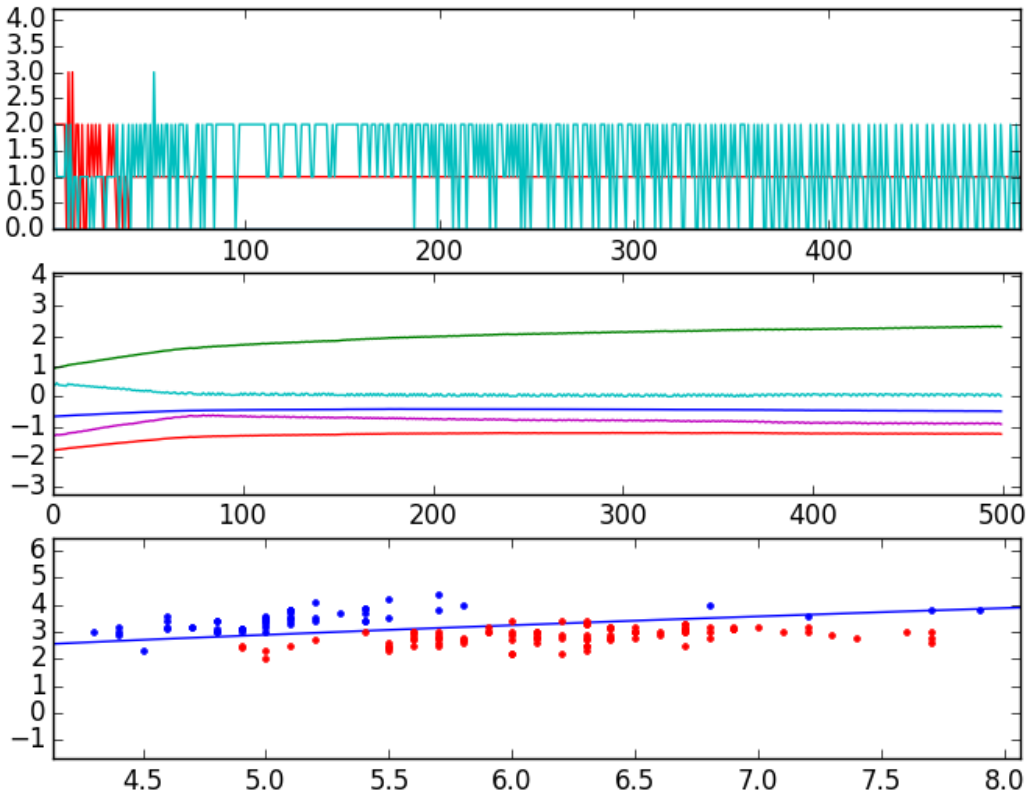
۴-۱ پیاده سازی برنامه:

برای پیاده سازی این الگوریتم از همان کلاس پرسپترون خطی استفاده میکنیم و تنها تابع Train آن را تغییر می دهیم.

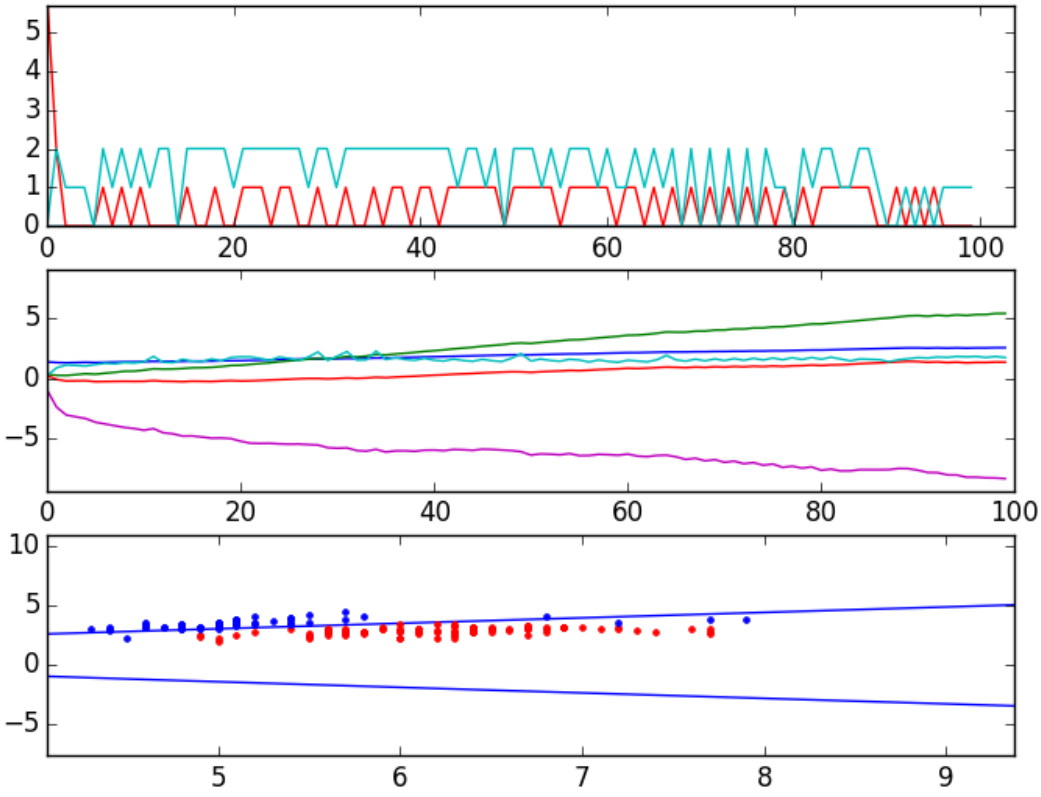
۴-۲ نتایج:

در شکل های زیر، نمودار اول تغییرات خطا در هر ایپک، نمودار دوم تغییرات وزن در هر ایپک و نمودار سوم داده های ورودی به همراه تابع جدا ساز آنها که به صورت هذلولی می باشد را نمایش می دهند.

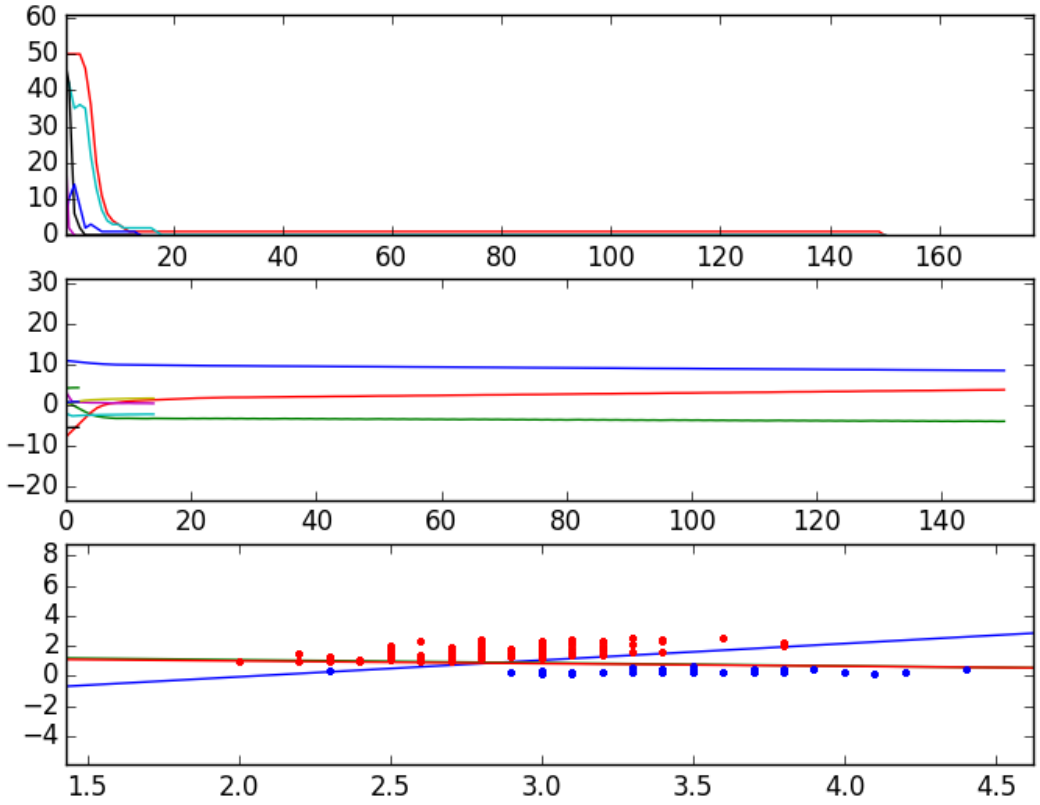
Alfa = 0.001:



Alfa = 0.01:



شکل زیر همزمان هر دو تابع جدا ساز به ازای ضرایب Alfa متفاوت را نمایش می دهد.



بخش پنجم:

واسط کاربری پروژه

این پروژه به زبان پایتون نوشته شده است . برای اجرا آن نیاز نصب برخی از کتابخانه های پایتون می باشد.

برای این پروژه نیاز به تنظیم پارامترهای متعددی می باشد که همه اینها در واسط کاربری در نظر گرفته شده است.

در واسط کاربری این برنامه تعدادی Radio Button در نظر گرفته شده است که می توان نوع الگوریتم را تعیین کرد، بعد از آن تعدادی Text Box قرار دارد که می توان پارامترهای ورودی را توسط آن تعیین کرد و Radio Button های پایین صفحه برای تعیین Data Set می باشد.

۵-۱ شرایط آزمایش:

برای اجرا برنامه باید ابتدا فایل ui.py اجرا شود، با اجرای این فایل همزمان دو پنجره باز می شود که یکی برای تنظیم پارامترها و دیگری برای رسم نمودارها می باشد.

در پنجره Hw1 برخی از مقادیر به صورت پیش فرض تعیین شده است که می توان آنها را تغییر داد. ابتدا از مجموعه Radio Button های بالای پنجره نوع الگوریتم باید تعیین شود، سپس پارامترها را باید تنظیم کرد.

alfa: همان نرخ یادگیری می باشد که به صورت پیش فرض 0.01 در نظر گرفته شده است.

epochs: تعیین کننده ی تعداد ایپک هایی است که در فاز آموزش می تواند طی شود که به صورت پیش فرض مقدار 100 است.

percent: این پارامتر نسبت نمونه های آموزشی به نمونه های validation را نشان میدهد، به طور مثال درصورتی که مقدار آن 0.7 قرار داده شود، 70% نمونه ها برای آموزش و 30% برای Validation در نظر گرفته می شود.

error_threshold: این پارامتر یکی از شرط های پایان آموزش می باشد. در صورتی که مقدار خطاها به این عدد برسد، آموزش متوقف می شود. به صورت پیش فرض مقدار آن برابر 1- قرار داده شده است به این معنا که این شرط در اتمام آموزش بی تاثیر باشد.

Weights address: اگر آدرس فایلی که وزنها در آن ذخیره شده اند را بدهیم، با زدن Train وزنها از فایل خوانده می شوند و با اجرای تنها یک ایپک کار خاتمه می یابد، در صورتی که آدرسی ذکر نشود و یا آدرس به اشتباه داده شود، برنامه با ایجاد وزن های رندم کار خود را آغاز می کند.

در هر مرحله از آموزش می توان آنها را save کرد و همچنین برای مقایسه چندین شبکه با یکدیگر کافی است آدرس وزنهای هرکدام را داده و Train زده شود.