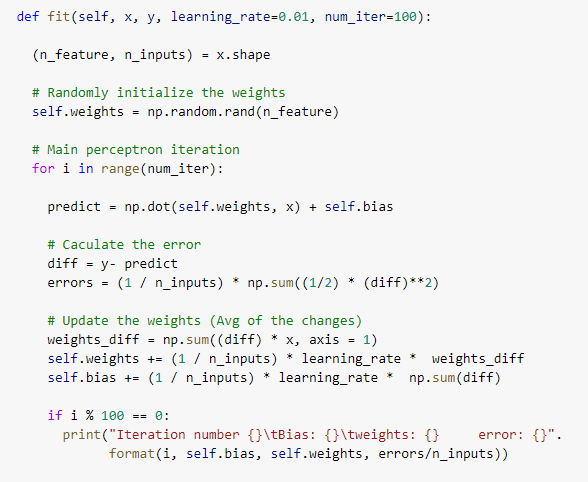
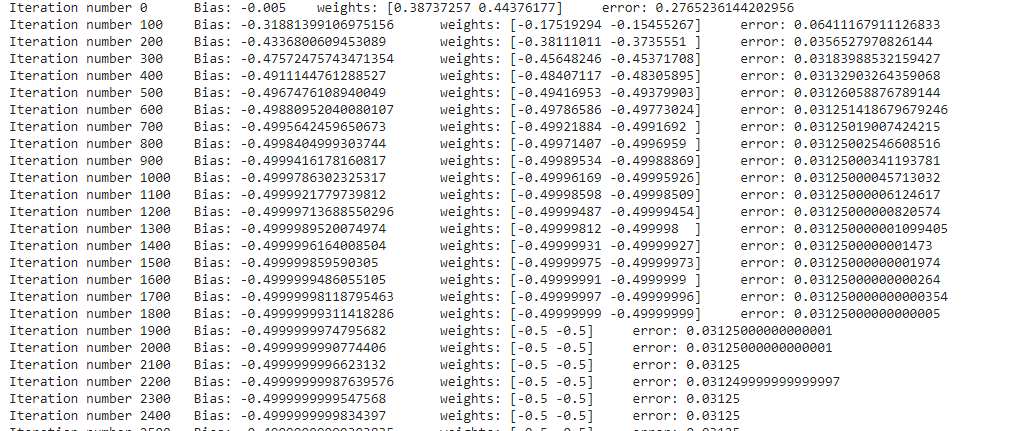
**محمد امین قسوری 97521432 CI HW 1**

توضیحات سوال اول:

سوال اول یک perceptron ساده بود و ما در حقیقت یک نورون داریم و دو داده برای train شدن. حال یک کلاس perceptron ساخته شده است که ابتدا مقادیر bias و وزن ها را بصورت رندوم ایجاد میکند. یک تابع fit هم خواهیم داشت که عملیات train شدن را برای ما انجام میدهد. ورودی های این تابع learning rate و تعداد iteration ها را به همراه داده ها هستند.

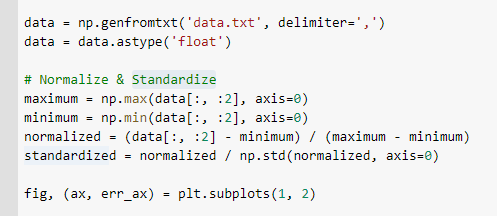


در هر حلقه Predict را بوسیله وزن ها حساب کرده و با استفاده از error ای که بدست میاید وزن ها و bias را برای این perceptron با توجه به learning rate آپدیت خواهیم کرد در هر مرحله میانگین error ها برای تمام داده های ما در نظر گرفته میشود و در نهایت به این وزن ها خواهیم رسید.

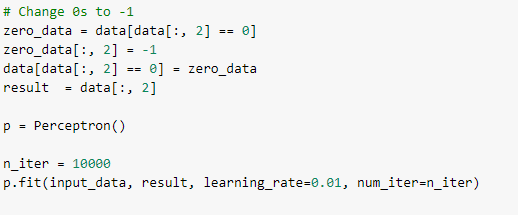


توضیحات سوال دوم:

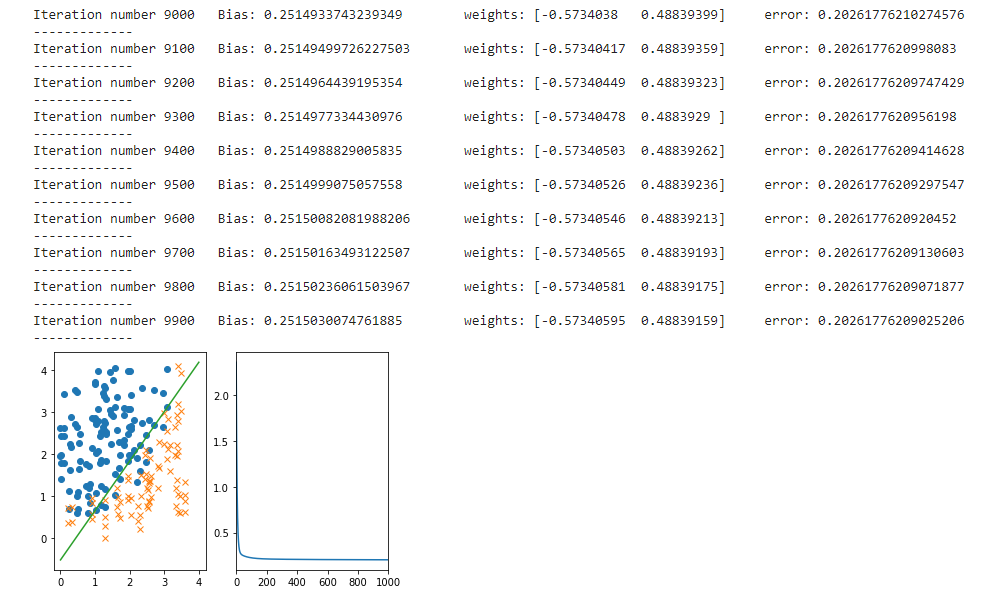
ابتدا فایل را به یک آرایه از نوع نامپای تبدیل میکنیم و در ادامه برای آن که بتوانیم سوال را به درستی حل کنیم در ابتدا باید داده های ورودی خود را Normalize و Standardize کنیم. با این کار محدوده دیتا های ما بین صفر و یک خواهد شد. برای آن که بتوانیم نقاط با مقادیر 0 و یک را در یک تصوی نشان دهیم figure را به شکل زیر ساختیم.



برای راحتی بیشتر داده های 0 را تبدیل به 1- میکنیم و در داخل همان کلاس سوال قبل پاس میدهیم.

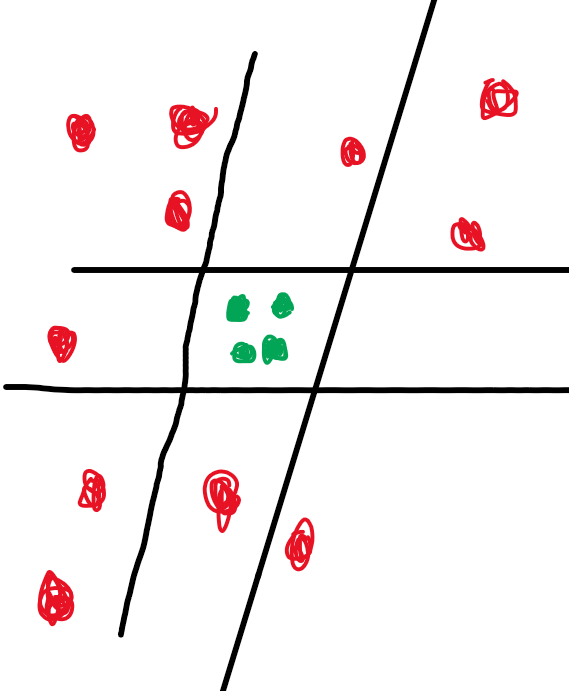


در این جا هم وزن ها را در مقادیر ضرب ماتریسی میکنیم و با بایس جمع خواهیم کرد. تنها نکته اضافی نگه داشتن error ها است تا بتوانیم در انتها با استفاده از آن نمودار را رسم کنیم.

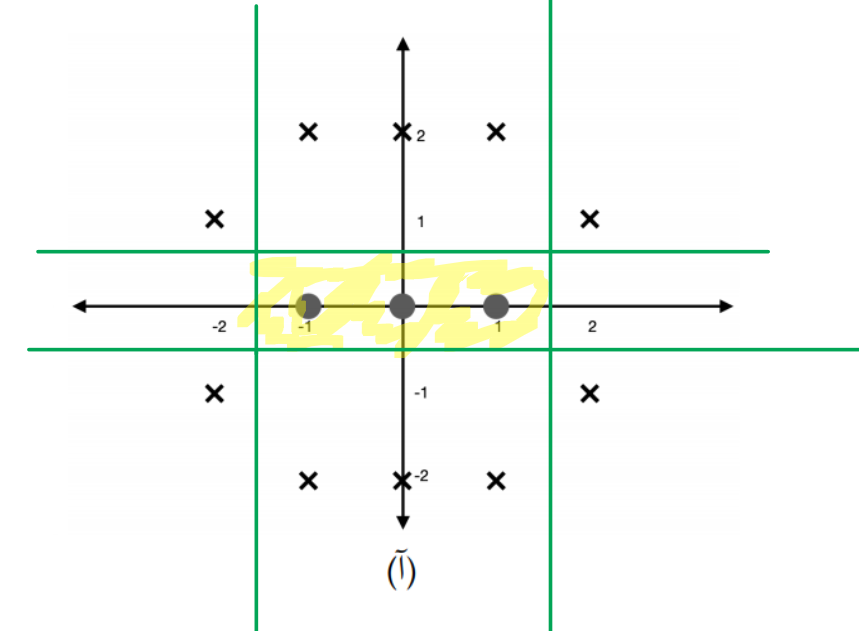


توضیحات سوال چهارم:

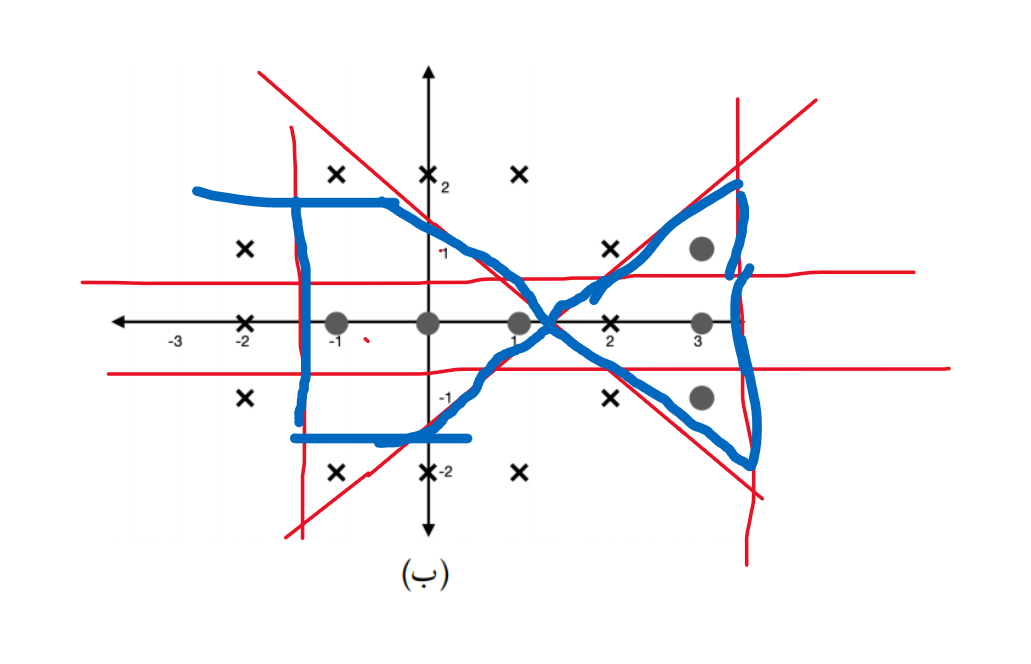
Madaline تعدادی واحد به نام Adaline وجود دارند که هر کدام میتواند مانند یک خط عمل کند و کلا در مدلاین ما میتوانیم چندین خط داشته باشیم و بوسیله شکلی که در بین آن ها و در اشتراک آن ها بوجود آمده است داده های خود را تفکیک کنیم. پس تفاوت با perceptron همین است. طبق نکته ای که سر کلاس هم اشاره شد و باید به آن توجه داشت مه باید ناحیه ما باید تشکیل یک چندضلعی محدب بدهد و در واقع داده های ما جدا پذیر باشند. در شکل زیر ناحیه سبز رنگ به وسیله یک convex یا چندضلعی محدب قابل پوشش است و هم چنین ناحیه ها از همدیگر جداپذیر هستند. در حقیقت هر کدام از این خط ها ادلاین های ما هستند که از and آن ها ناحیه بدست آمده است.



برای مورد الف میتوانیم ناحیه ای که در آن دایره ها هستند را بوسیله یک convex یا چند ضلعی محدب تفکیک نمود. در اینجا پس ما میتوانیم 4 ادلاین بصورت parallel داشته باشیم که یک مدلاین را تشکیل میدهند. و از همه آن ها به یک نرون دیگر میروند تا با هم and شوند.

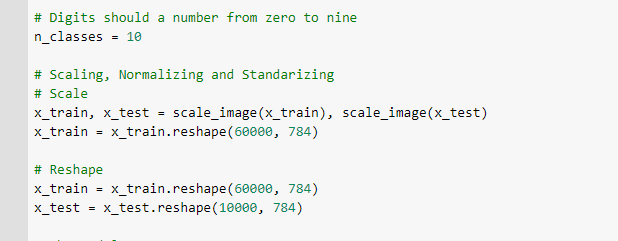


در مورد ب این مطلب صادق نیست و این دو ناحیه جدا ناپذیر هستند و نمیشود بوسیله یک مدلاین آن ها را از یک دیگر تفکیک کرد.

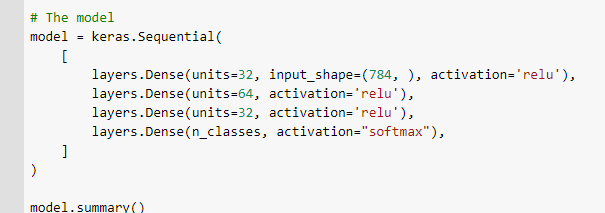


توضیحات سوال چهارم:

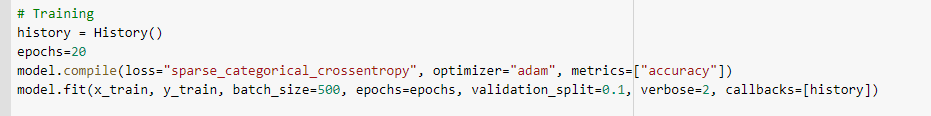
بیشتر سوال چهارم کار با keras بود وگرنه از بین بقیه سوالات پیچیدگی کمتری داشت.داده های mnist را لود کردیم. عکس ها 28 در 28 هستند و هر پیکسل بین 0 تا 255 است و دوباره اینجا از عملیات scaling استفاده شد تا نتایج درست به دست بیایند و داده های ما normalized باشند.

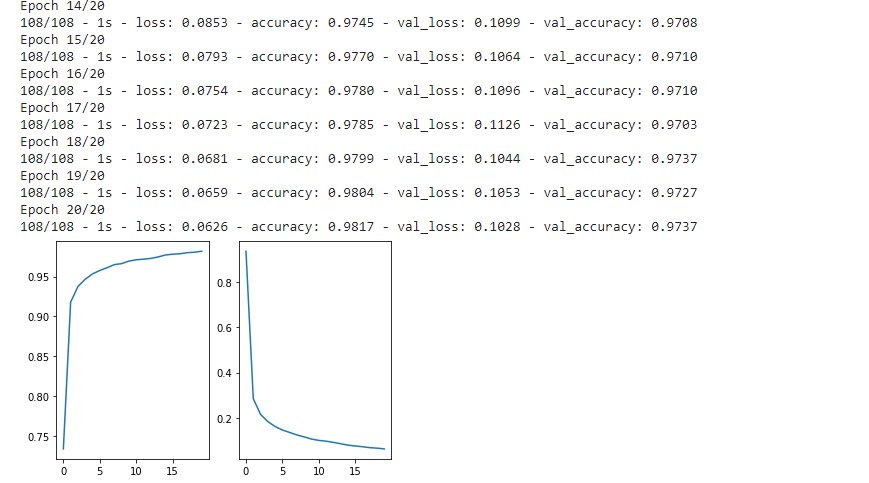


در keras به راحتی میتوانیم در داخل یک مدل لایه های خود را قرار دهیم. برای این سوال من یک مدل با سه لایه hidden و یک لایه پایانی درست کردم. لایه از نوع Dense چیزی بود که تا اینجا بلد هستیم و در هر لایه activation function را مشخص خواهیم کرد که در سه لایه اول relu هست که زیر 0 صفر و بالای صفرمقدار خود را دارد. لایه آخر هم که softmax است.

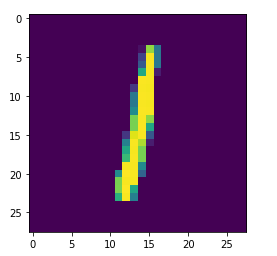
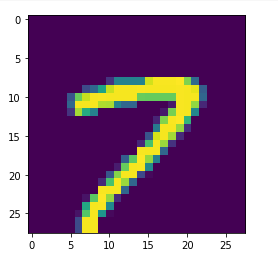
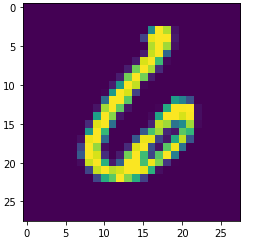
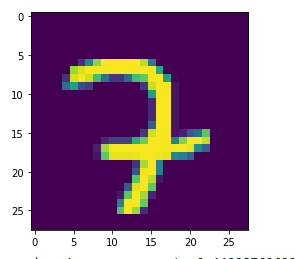


Epoch همان تعداد iteration های ماست و batch هم همان اندازه دسته دسته دیتا هایی است که در هر iteration بررسی میشوند. در اینجا mini batch داریم و در هر مرحله 500 مورد بررسی میشود. آن یک دهم هم برای validation هست که خوب حواسمون باشه overfit نشیم.



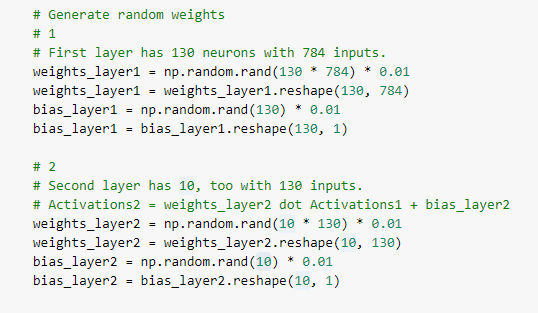


توضیحات سوال پنجم:

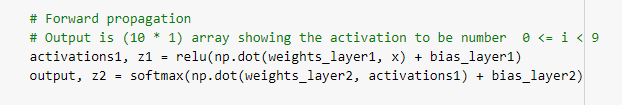


عکس های ما دوباره همان عکس ها 28 در 28 هستند دوباره آن ها را Normalize میکنیم و scale میکنیم. یک مدل 2 لایه در نظر گرفته شده است که لایه اول از تابع relu و لایه آخر از تابع softmax استفاده میکند و به ترتیب تعداد 130 و 10 تا نرون دارند. (دقیقا مانند سوال قبل) حال دیتا های train را به شکل (784, 60000) تبدیل کرده و همچنین نتایج را به شکل (1, 600000) تبدیل کرده ایم. این مقادیر ورودی شبکه ای میشود که ما بصورت scratch با استفاده از numpy طراحی کرده ایم. در این شبکه و در هر iteration خطا بروی همه داده های ترین سنجیده میشود و به عبارتی دیگر ما از batch gradient descent استفاده میکنیم.

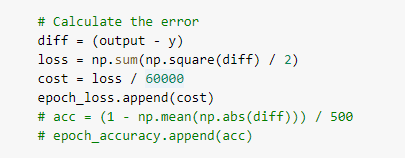
در ابتدا وزن ها و بایس ها را بصورت رندوم میسازیم و شکل آن ها را متناسب میکنیم. 0.01 برای کمتر کردن مقادیر بین 0 تا 100 تابع رندوم است.



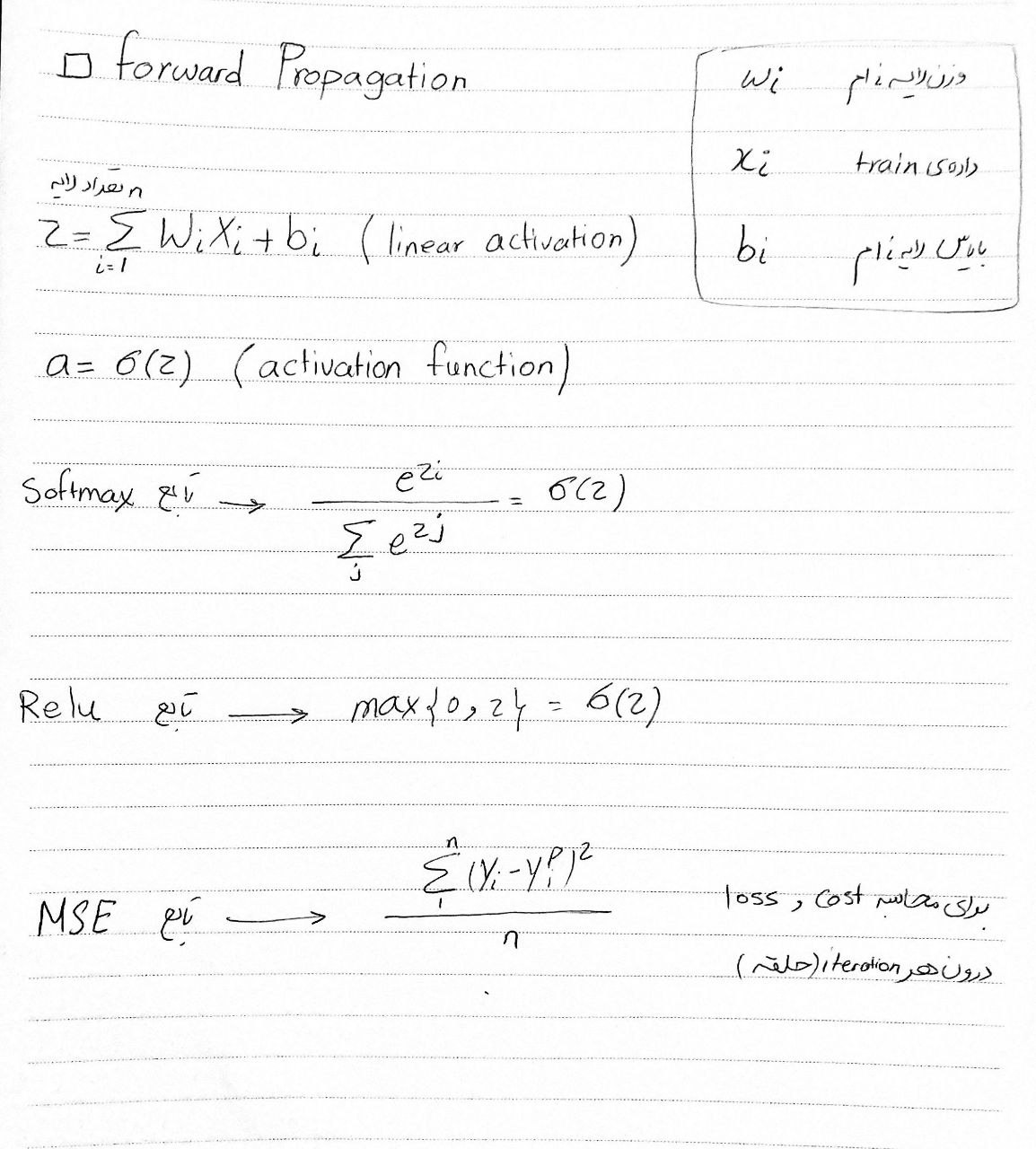
سپس در هر iteration ما ابتدا عملیات forward propagation را انجام خواهیم داد. با توجه به ابعاد ورودی کافی است وزن را در داده ها ضرب ماتریسی کرد و با بایس جمع کرد.

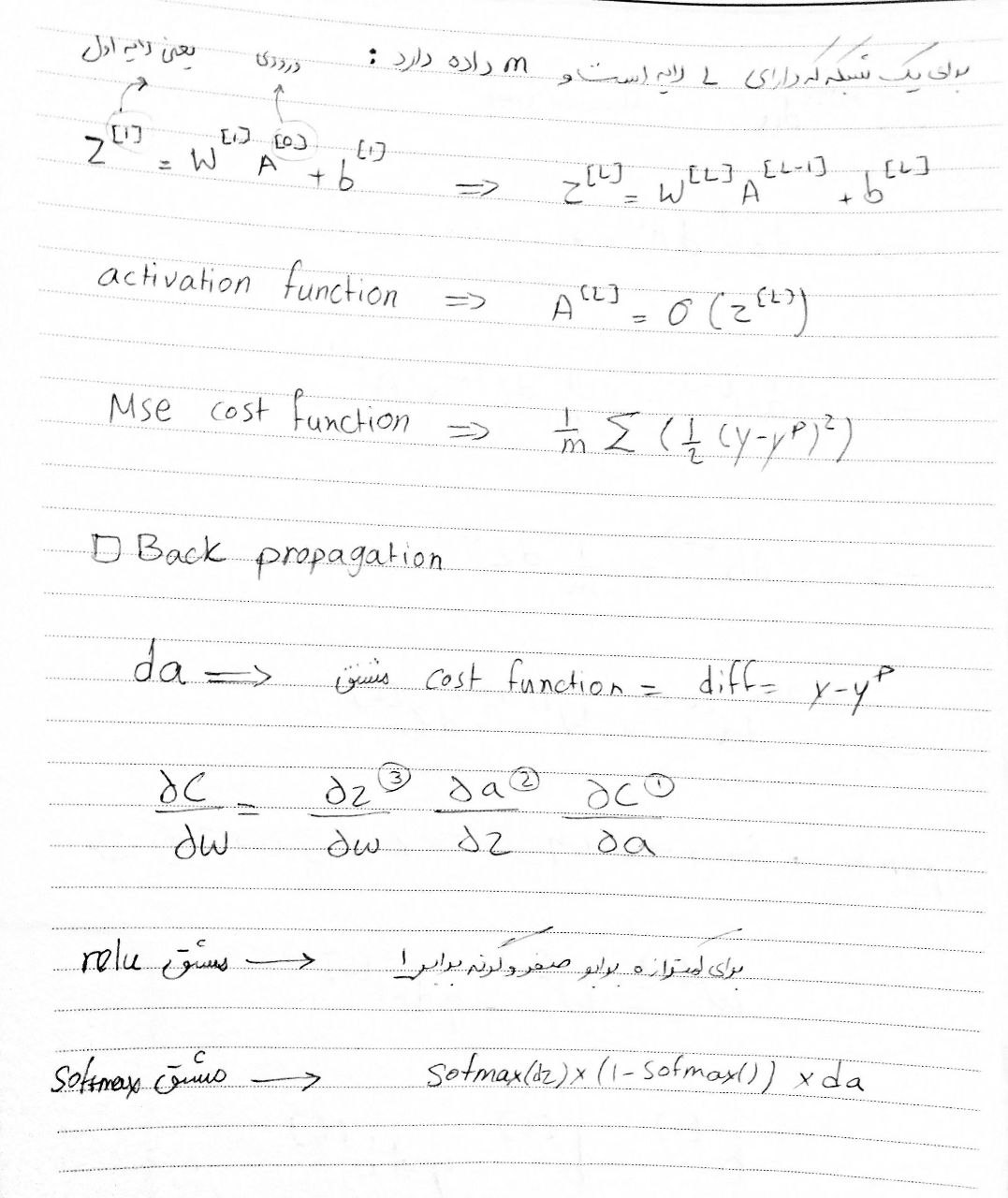


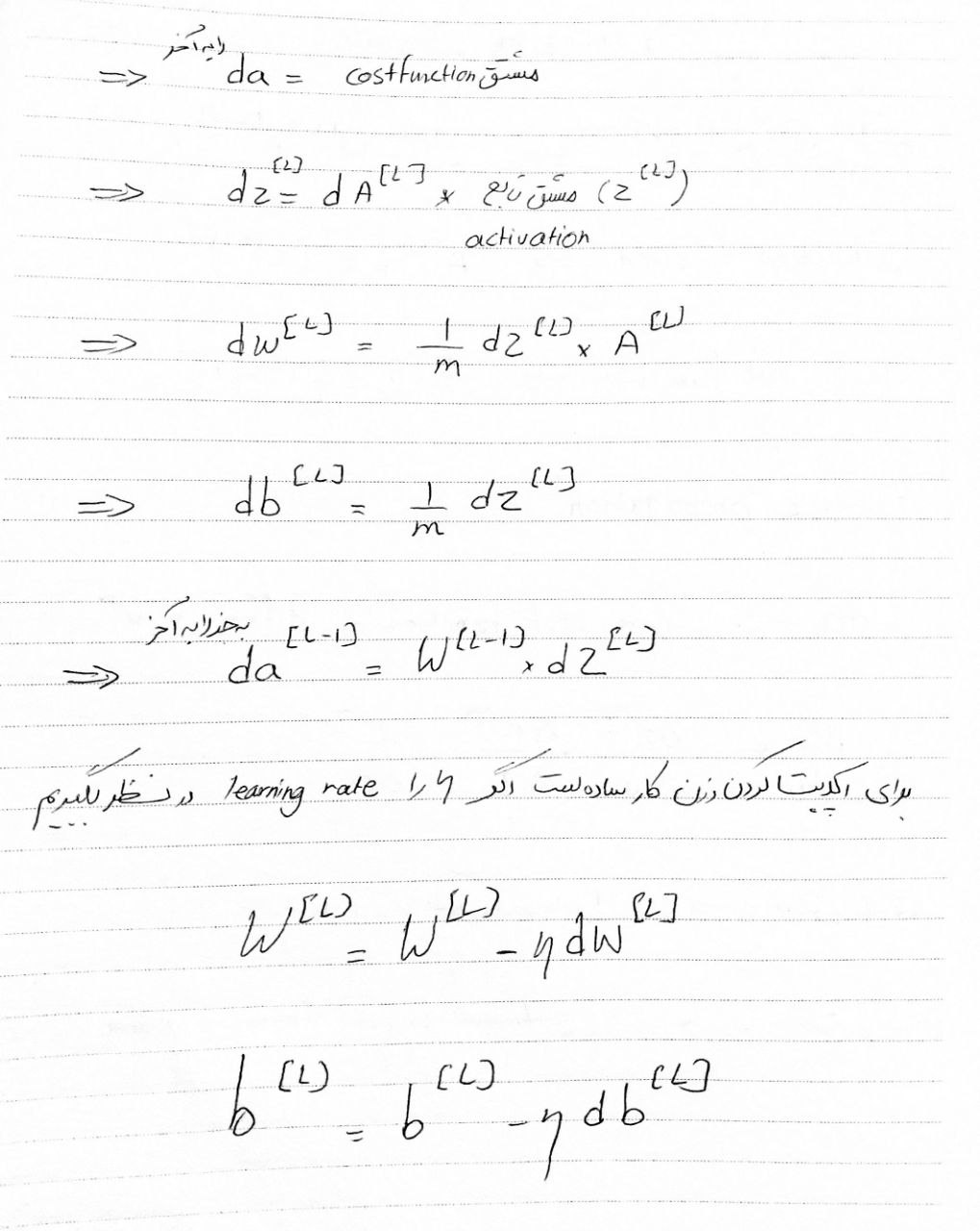
در مرحله بعدی باید خطا را حساب کنیم که من از همان MSE برای سنجش میزان ارور استفاده کرده ام.

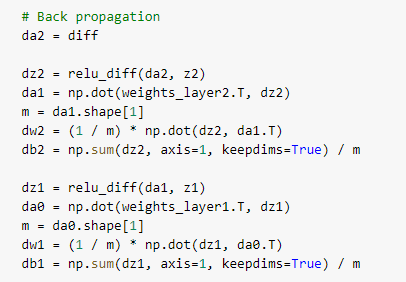


در ادامه باید عملیات back propagation را انجام بدهیم. در ابتدا کمی محاسبات لازم است.









همانطوری که محاسبات ریاضی انجام شد به شکل زیر آن ها را پیاده سازی میکنیم. در محاسبه کردن da ها بدلیل این که با ابعاد ابتدایی نمیتوانیم ضرب ماتریسی انجام بدیم مجبور هستیم که ترانهاده ماتریس وزن آن لایه را در dz ضرب کنیم تا ابعاد با هم ضرب پذیر باشند. در انتها نیز با توجه به learning rate و گرادیانت وزن در جهت مخالف وزن ها را تغییر می دهیم تا خطای ما مینیمم شود.

