هوش محاسباتی

پروژه شبکه عصبی

گام اول:

در گام اول ما کتاب خانه های numpy,matplotlib.pyplot,tensorflow.keras را import میکنیم.توجه شود که تنها دلیل استفاده از keras برای خواندن داده های mnist بوده است.

تابع read_data وظیفه خواندن داده های mnist را دارد.تابع های show_train_image,show test image برای نمایش داده های اموزشی و تست به همراه لیبل آن ها میباشد. و در نهایت تابع data_number تعداد داده های آموزشی و تست را نشان میدهد.

گام دوم:

در گام دوم کتاب خانه های مورد نظر را Import میکنیم.سپس داده های اموزشی و تست و لیبل های آن ها را در در لیست های مجزا نگهداری میکنیم.

وزن ها را مقدار دهی اولیه میکنیم بر اساس توزیع نرمال و آن ها را در ماتریس ۷۸۴×۱۰ نگهداری میکنیم.

سپس تابع سیگموید را نوشته و در نهایت بر اساس وزن های رندوم داده شده دقت را محاسبه میکنیم. که مشاهده میشد دقت ۱۰ درصد میباشد.

گام سوم:

توجه شود که در گام ۳ به علت کند بودن الگوریتم فقط دولایه(اول و اخر) را در نظر گرفتم.

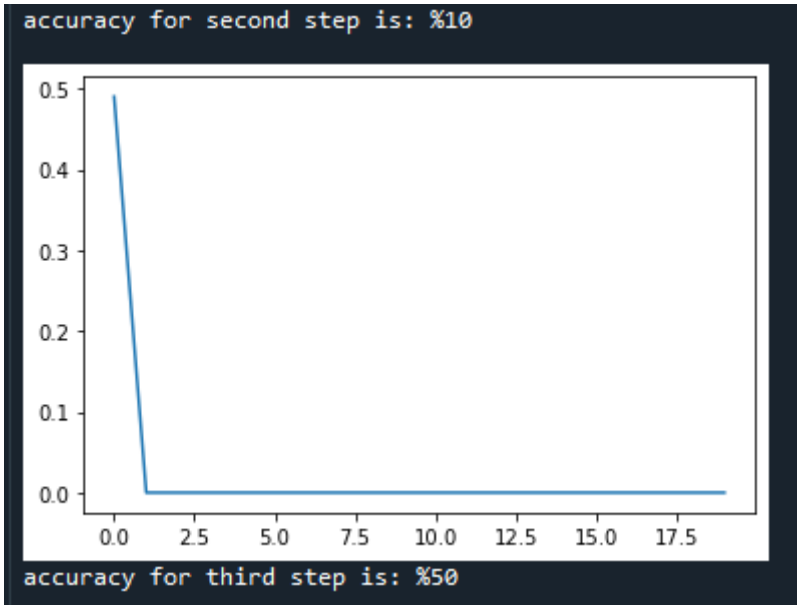
در ابتدا ماتریس گرادیان وزن و تابع گرادیان بایاس را به ماتریس های تماما صفر مقدار دهی اولیه انجام دادم.

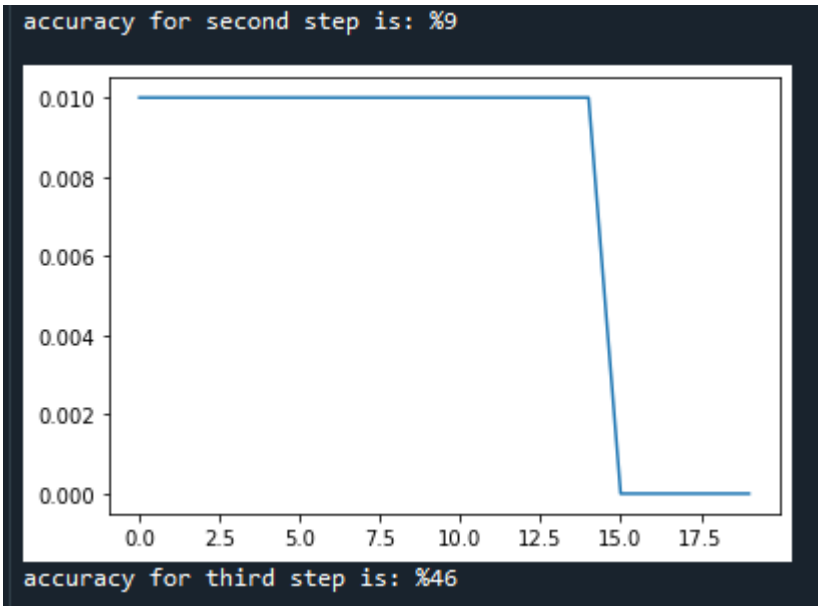
و تابع cost_gradint_calc را نوشتم که که گرادیان هزینه نسبت به وزن و بایاس را برمیگرداند.

در نهایت داده ها را با هایپر پارامتر های خواسته شده اموزش دادم تا دقت را برای ۱۰۰ داده اول از اول محاسبه کنم در نهایت دقت ۵۰ درصد بدست میاد و مشاهده میشود تابع هزینه

روند نزولی داشته.در یک ران دیگر از برنامه دقت اولیه ۹ درصد و دقت نهایی ۴۶ درصد شد.توجه شود به علت زمانبر بودن الگوریتم تابع cost را به صورت تقریبی محاسبه کردم اما اگر به صورت دقیق محاسبه میکردم شکل به صورت نزول تدریجی در میامد.

در شکل زیر خروجی برای قدم دوم و سوم را مشاهده میکنید:





گام چهارم:

در ابتدا کتابخانه های مورد نیاز را import میکنیم.و بعد از آن تابع فعالیت را تعریف میکنیم.(در اینجا از relu استفاده کردم.)

تابع find_max_element در یک ماتریس طول و عرض عضو ماکسیمم را برمیگرداند.

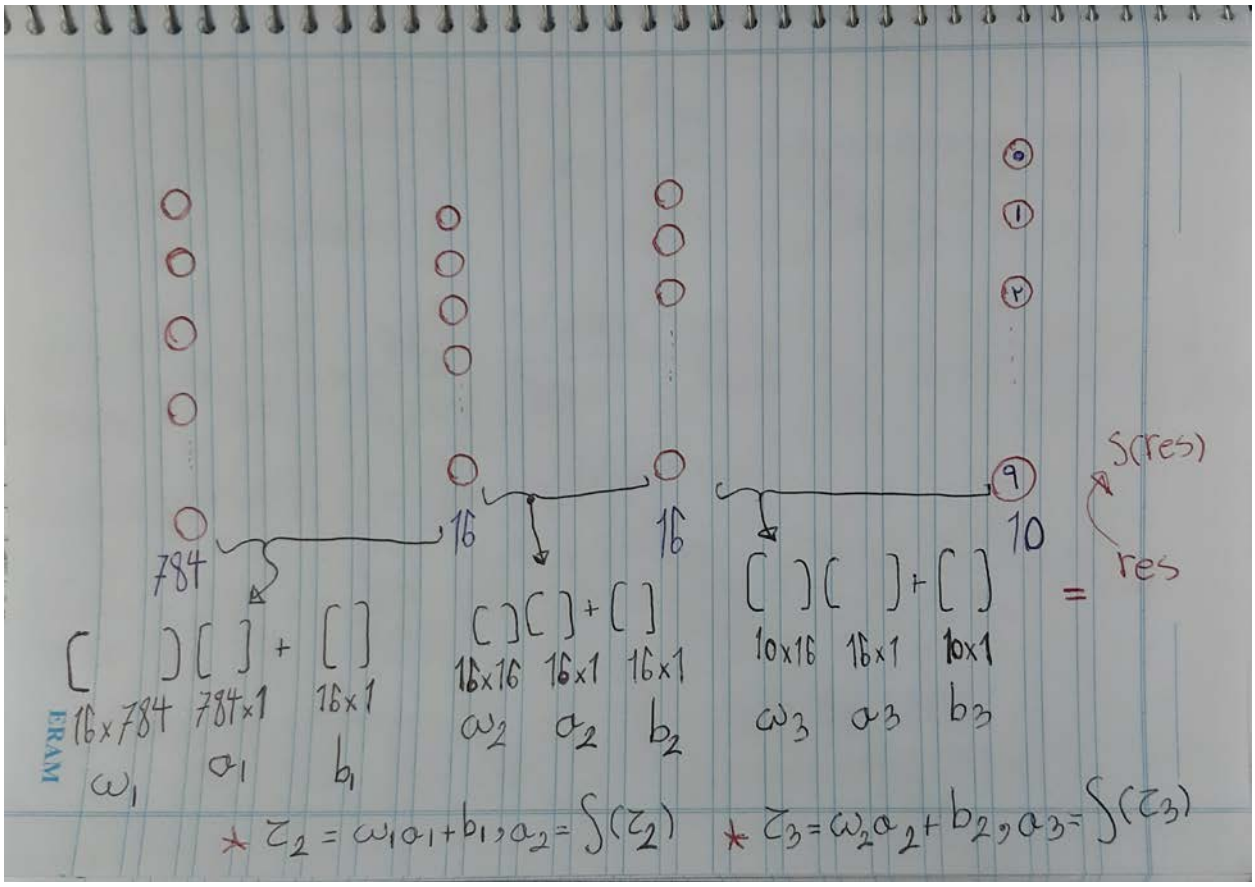
برای هر ۳ سطح ماتریس بایاس و وزن را تعریف کرده ایم.تابع هزینه که هزینه نهایی را محاسبه میکند.

تابعی نوشتم که بتواند a,z سطح ۲ و۳ را محاسبه کند.(level2_AZوlevel3_AZ)

تابع y_calculator ماتریسی را برمیگرداند که خانه سطری که متناظر با لیبل واقعی است را یک قرار میدهد ۹ تا سطر بعدی را صفر قرار میدهد.

در نهایت هم تابع learn با گرفتن پارامتر ها اموزش را انجام میدهد.

در شکل های زیر ساختار شبکه عصبی و مشتق cost نسبت به وزن های سطح ۱و۲و۳ به صورت vectorized آورده شده است.



$$\overset{O}{\overbrace{cost = \Sigma\left(\sigma(res) - y_i\right)^2}}$$

y_i → realdata

مثلاً اگر لیبل ورودی ۷ باشد.

$Y = [0,0,0,0,0,0,0,1,0,0]$ → البته matrix 10×1 است.

$$\frac{\partial cost}{\partial \omega_3} = \frac{\partial cost}{\partial O} \times \frac{\partial O}{\partial \sigma(res)} \times \frac{\partial \sigma(res)}{\partial res} \times \frac{\partial res}{\partial \omega_3} = \underbrace{2\left(\sigma(res) - y_i\right)}_{10\times1} \times \underbrace{\sigma'(res)}_{10\times1} \times \underbrace{a_3}_{16\times1}$$

$$= \left(2\left(\sigma(res) - y_i\right) \times \sigma'(res)\right) @\ a_3.transpose()$$

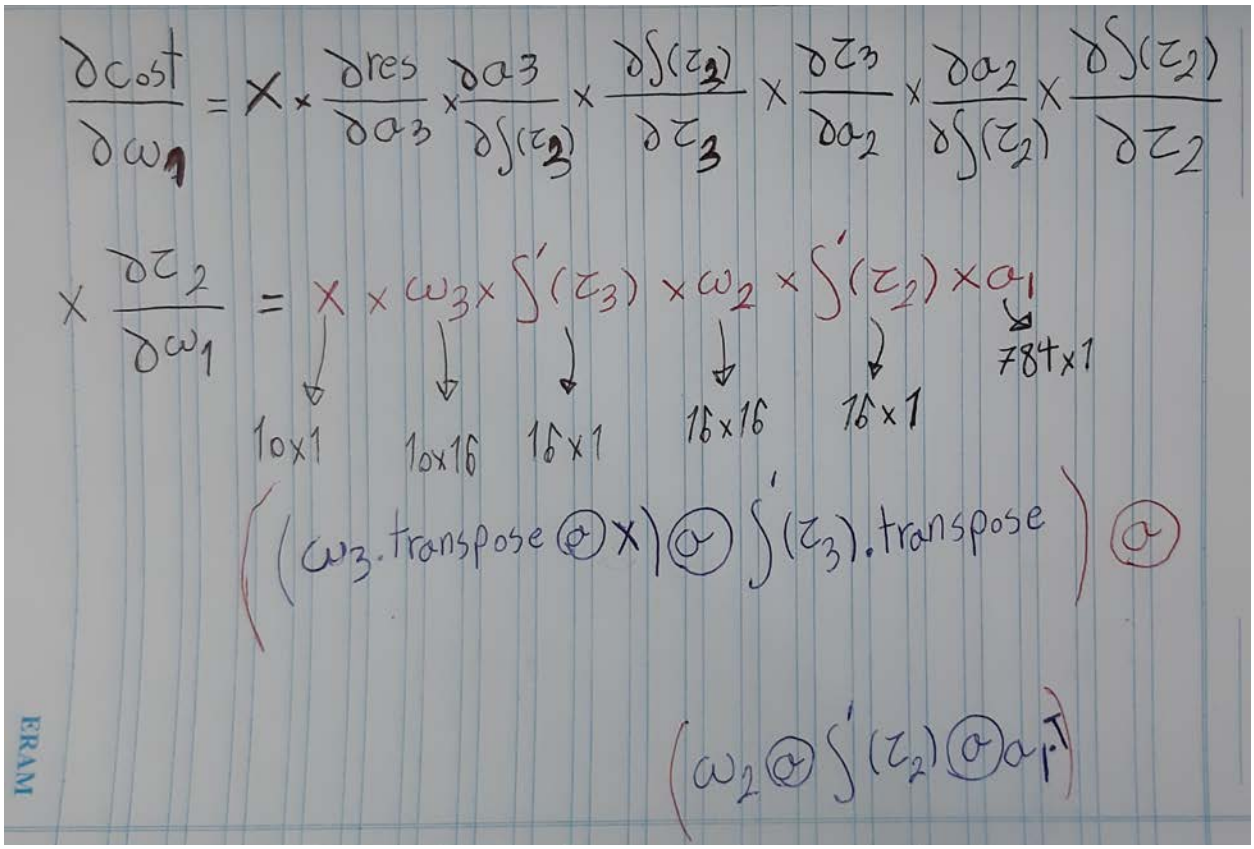
$$\frac{\partial cost}{\partial \omega_2} = \frac{\partial cost}{\partial O} \times \frac{\partial O}{\partial \sigma(res)} \times \frac{\partial \sigma(res)}{\partial res} \times \frac{\partial res}{\partial a_3} \times \frac{\partial a_3}{\partial \sigma(z_3)} \times \frac{\partial \sigma(z_3)}{\partial z_3} \times \frac{\partial z_3}{\partial \omega_2}$$

$$= \underbrace{2\left(\sigma(res) - y_i\right) \times \sigma'(res)}_{10\times1} \times \underbrace{\omega_3}_{10\times16} \times \underbrace{\sigma'(z_3)}_{16\times1} \times \underbrace{a_2}_{16\times1} =$$

ERR → X

$(\omega_3.transpose @ X) @ (\sigma'(z_3) \times a_2).transpose()$

$a_3 = sigmoid(z_3)$



گام پنجم:

تابع تست که در آن تابع learn را فراخوانی میکند و دقت نهایی را برحسب درصد میدهد.