به نام خدا

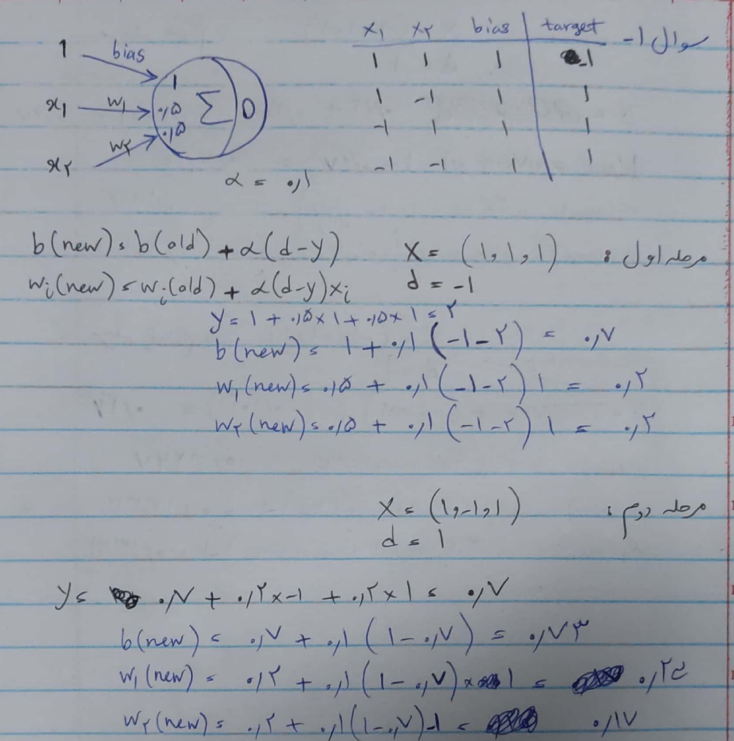
درس هوش محاسباتی

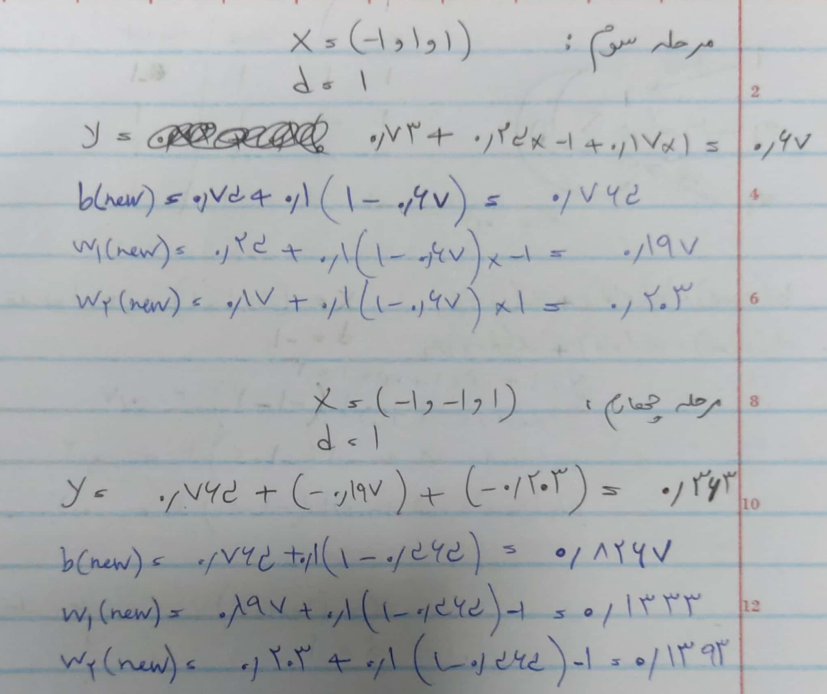
تمرین سری دوم

سینا علی‌نژاد

شماره دانشجویی: 99521469

سوال 1-





سوال 2-

الف) توابع فعالسازی خطی، یک رابطه خطی بین ورودی و خروجی تولید میکنند. یعنی خروجی یک ضرب وزن‌دار بر روی ورودی است، یا به عبارتی y = wx + b که x ورودی، y خروجی، w وزن‌ها و b همان bias است. این نوع توابع فعالسازی برای مسائل ساده رگرسیون مورد استفاده قرار میگیرند و برای مسائل پیچیده‌تر مناسب نمی‌باشند زیرا خیلی از مسائل را نمیتوان با یک خط مدل کرد.

برای حل مشکل بالا، توابع فعالسازی غیرخطی مورد استفاده قرار می‌گیرند. این توابع می‌توانند روابط پیچیده‌تر بین ورودی و خروجی را مدل کنند. بعضی از این توابع عبارتند از: Relu, Tanh, Sigmoid. این توابع باعث میشوند شبکه منعطف تر باشد و برای مسائل پیچیده‌تر مثل تشخیص تصویر و پردازش زبانهای طبیعی جواب بدهد.

ب)

روش اول: در این روش، به دلیل صفر بودن وزنها ممکن است دچار مشکل شود. بدین صورت که خروجی همه نورونهای یک لایه برای یک ورودی هنگام forward و backpropagation، یکسان خواهد بود و این باعث میشود وزنهای این لایه به یک میزان آپدیت شوند و شبکه نتواند ویژگی‌های مختلف را یاد بگیرد. به این مشکل، symmetry issue هم گفته میشود. این روش در مقداردهی اولیه توصیه نمیشود.

روش دوم: این روش، روشی است که معمولا در شبکه‌ها استفاده می‌شود. صفر بودن ابتدایی بایاس باعث می‌شود شبکه با استفاده از همان وزنها بهترین یادگیری را انجام دهد و از بایاس برای بهتر کردن آن کمک بگیرد. رندوم بودن مقدار اولیه وزنها باعث میشود، مدل سریعتر ویژگی‌های مختلف را یاد بگیرد و هنگام آموزش زودتر همگرا شود.

ج) کمترین قدرت تعمیم برای perceptron است. اگر بخواهیم این مدل را با Adaline مقایسه کنیم، در اینجا سیگنال ارور بر روی خروجی نهایی که yes/no هست، محاسبه میشه ولی در Adaline اختلاف همان خروجی عددی با مقدار واقعی مقایسه میشه و مدل تلاش میکنه این فاصله رو کمتر کنه. در واقع هردوی این مدل‌ها صرفا بر روی مسائل خطی میتوانند کار کنند ولی Adaline نسبت به perceptron خط بهتری را به مدل فیت میکند، خطی که تقریبا در وسط ناحیه مرز بین دو کلاس است. در حالیکه perceptron صرفا اگر آن خط به درستی دو کلاس را جدا کرده باشد، متوقف میشود و تلاشی برای بهتر کردن آن خط نمی‌کند.

همچنین در M-adaline میتوان مدل‌های کمی پیچیده‌تر نسبت به مدل خطی ایجاد کرد زیرا چندین Adaline را ترکیب میکند ولی ماهیت خطی بودن Adaline همچنان آن را محدود میکند.

و در نهایت، بیشترین قابلیت تعمیم‌دهی مربوط به MLP (Multi Layer Perceptron) میشود. این مدل شامل چندین لایه پنهان و توابع فعالساز غیرخطی میشود، در نتیجه میتواند بازنمایی مسئله را بارها عوض کند تا بتواند در نهایت با یک لایه خطی مسئله را حل کند. توابع فعالساز غیرخطی در تغییر این بازنمایی نقش مهمی دارند، در غیر اینصورت، چندین لایه خطی پشت سر هم با یک لایه خطی فرقی نداشت. قدرت تعمیم دهی بالای این مدل باعث میشود در مسائل پیچیده‌تر مثل پردازش تصویر و پردازش زبانهای طبیعی مورد استفاده قرار بگیرد.

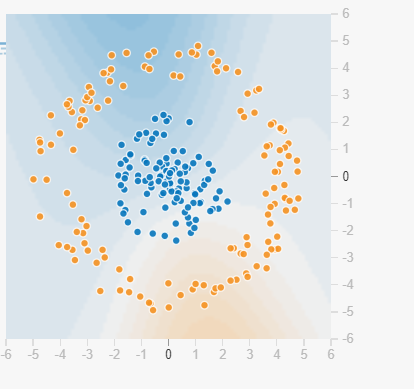
در نتیجه کمترین قدرت تعمیم دهی برای perceptron و بیشترین متعلق به MLP میباشد.

د) این روش یک مزیت مهم دارد. در روش معمول، از یک نرخ یادگیری آلفا برای همه‌ی وزن‌ها استفاده می‌شود و به این شکل است که یک خط در نزدیکی نمودار تابع ضرر نسبت به یک وزن مشخص در نظر گرفته شده و در جهت خلاف آن خط و به یک میزان مشخص که توسط مقدار مشتق تابع ضرر نسبت به این وزن و همچنین مقدار نرخ یادگیری مشخص میشود، حرکت میکند. اما در این روش که از ماتریس hessian که شامل مشتقهای مرتبه دوم است استفاده میشود، به جای فیت کردن یک خط، یک تابع درجه دو را در آن نقطه فیت میکند و نقطه مینیمم این تابع را بدست می‌آورد(محاسبه نقطه مینیمم تابع درجه دو راحت است) و به همان اندازه، این وزن را آپدیت میکند. این باعث میشود جایی که نیاز است، با مقدار بیشتری آپدیت انجام شود و این بر خلاف چیزی است که در استفاده از نرخ یادگیری داشتیم. در واقع انگار برای هر پارامتر از یک نرخ یادگیری متفاوت بسته به شرایط استفاده میکنیم. در نهایت، از همه گفته‌های بالا می‌توان این نتیجه را گرفت که شبکه با استفاده از این روش سریعتر همگرا میشود.

معایب: معایب این روش تقریبا مشخص است. یکی از آنها، محاسبه بیشتر است زیرا شبکه در هر backpropagation به غیر از مشتق اول، باید مشتق دوم را هم محاسبه کند. عیب دیگر حافظه زیادی است که میخواهد. اگر تعداد پارامترهای کل شبکه S باشد، تعداد اعضای ماتریس hessian برابر با S\*S خواهد بود که برای مدل‌های بزرگ کنونی، مقدار قابل توجهی است.

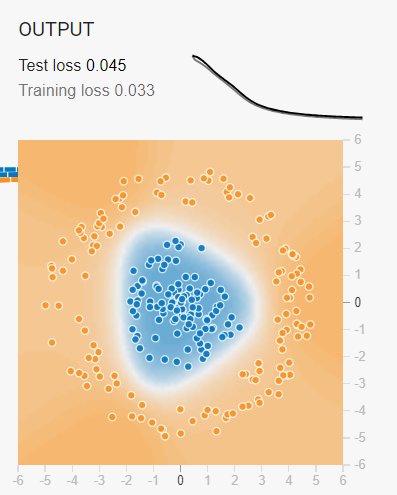
سوال 3-

دیتای اول:



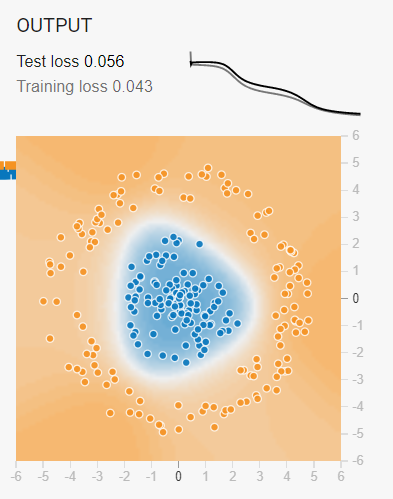
تابع Tanh:

با استفاده از این تابع، تنها پس از 75 epoch مدل همگرا شد.



تابع Sigmoid:

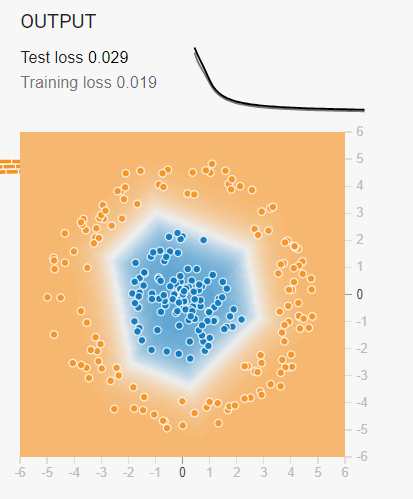
در این حالت، مدل پس از حدود 325 epoch همگرا شد که نسبت به Tanh خیلی تفاوت داشت.



اولین دلیل این است که خروجی تابع سیگموئید همواره مثبت است و این باعث می‌شود در فرایند backpropagation همه وزنهای یک نورون یا همزمان زیاد شوند یا همزمان کم شوند و این باعث میشود که همگرایی نسبت به Tanh دیرتر رخ دهد. دلیل دیگر، شیب بیشتر تابع Tanh نسبت به سیگموئید است که باز هم باعث سریعتر بودن همگرایی میشود.

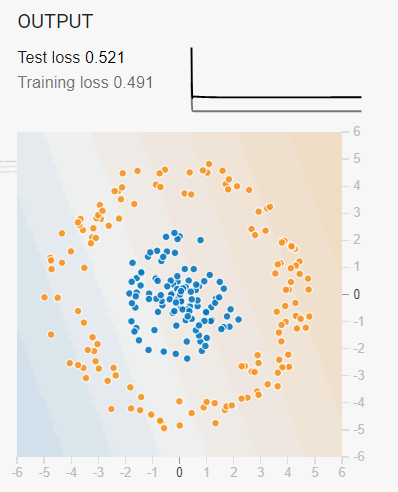
تابع Relu:

در این حالت، چون در قسمت مثبت، شیب تابع کم نمیشود، بنابراین اجازه میدهد مدل با دست بازتر آپدیت پارامترها را انجام دهد و همگرایی سریعتر میشود، به طوریکه تنها پس 40 epoch تقریبا همگرا میشود.

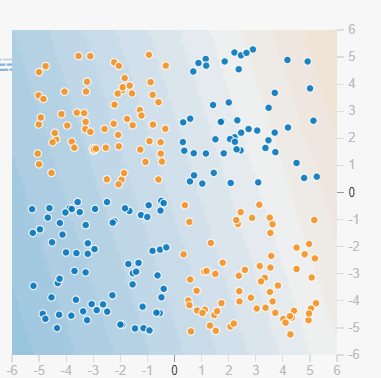


تابع Linear:

در این حالت همگرایی حتی پس از 500 epoch حاصل نمیشود زیرا این مسئله Linear Seperable نیست.

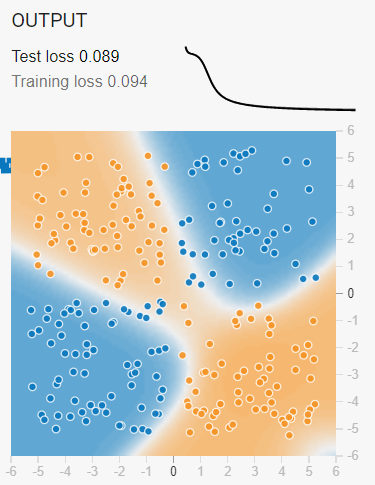


دیتای دوم:



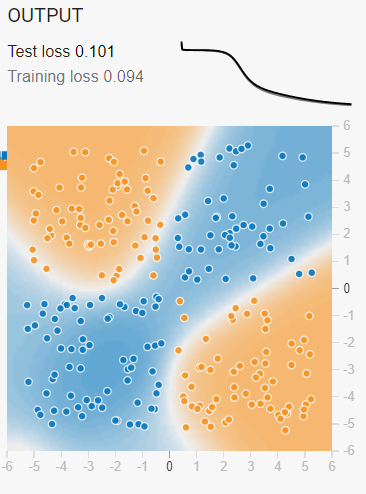
تابع Tanh:

پس از 130 epoch همگرا شد. نسبت به دیتاست قبلی با همین تابع فعالساز، دیرتر همگرا شد زیرا احتمالا مدل پیچیده‌تری برای این دیتاست نیاز است.



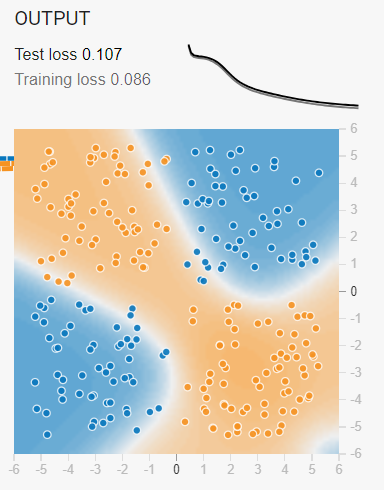
تابع Sigmoid:

به همان دلایلی که برای دیتاست قبلی گفتم، نسبت به Tanh دیرتر همگرا میشود. در این حالت پس از 310 epoch همگرایی رخ داد



تابع Relu:

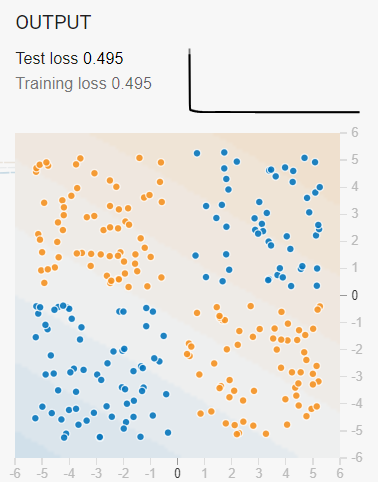
پس از 60 epoch همگرا شد و سریعتر بودن آن به همان دلایلی است که برای دیتاست اول گفتیم. نکته جالب این بود که توی بعضی از ران‌ها با این تابع فعالساز، مدل نمیتونست همگرا بشه که حدس میزنم بخاطر تاثیر مقداردهی اولیه وزن‌ها باشه که ماهیت رندوم بودنش باعث این موضوع میشه.



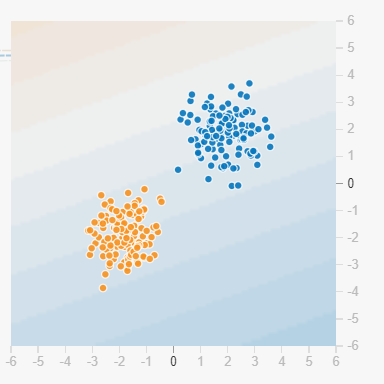
تابع Linear:

در این حالت، طبق انتظار، مدل همگرا نشد زیرا مسئله، یک مسئله خطی نبود.

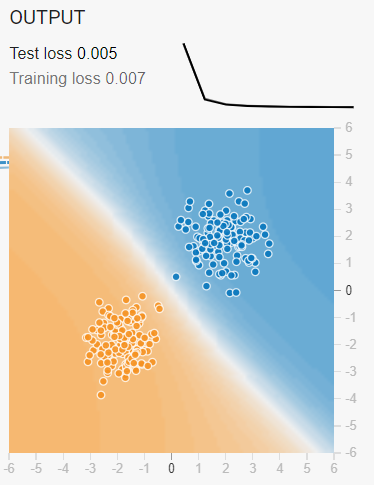
پس از 500 epoch:



دیتاست سوم:



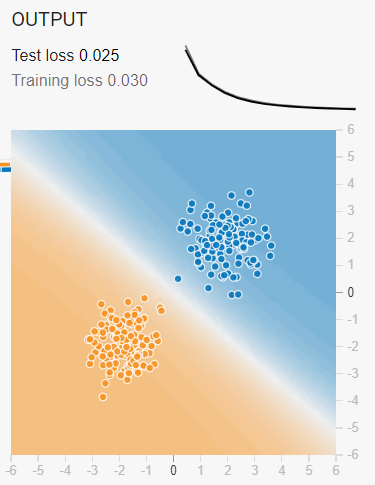
تابع Tanh:



تنها پس از 5 epoch همگرا شد، زیرا این یک مسئله خطی است و کمترین پیچیدگی را دارد.

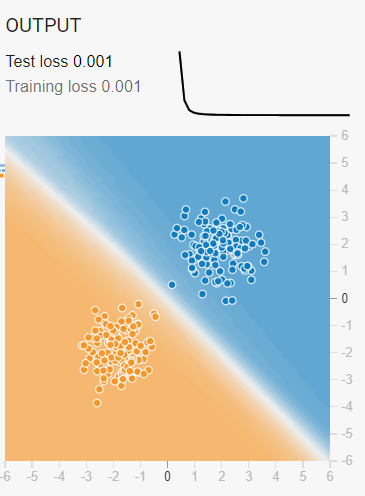
تابع Sigmoid:

پس از 10 epoch همگرا شد.



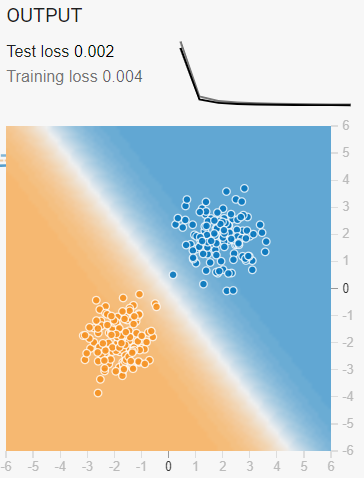
تابع Relu:

پس از 15 epoch همگرا شد.



تابع Linear:

نکته حائز اهمیت این است که در این حالت، تابع خطی نیز همگرا شد که به دلیل خطی بودن مسئله است.

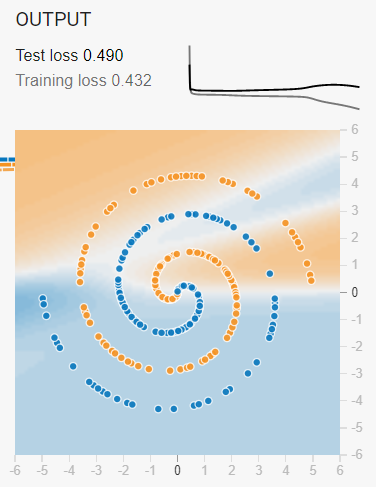


پس از تنها 6 epoch همگرا شد.

دیتاست چهارم:

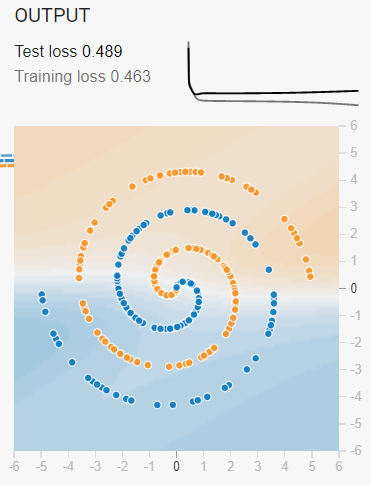
تابع Tanh:

از آنجا که مسئله خیلی پیچیده است، با این تعداد نورون و لایه نمیتوان به همگرایی رسید. میتوان با زیاد کردن تعداد لایه ها و تعداد نورونها به دقت خیلی بهتری رسید.



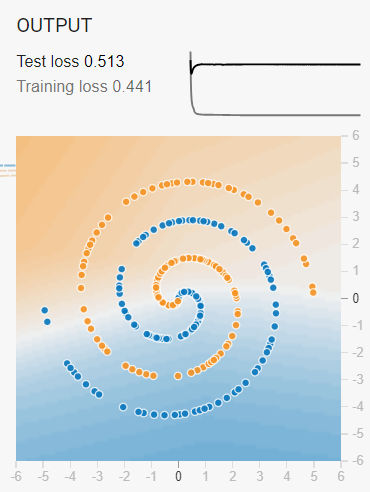
تابع Sigmoid:

همان نکاتی که برای تابع Tanh گفتیم، اینجا هم صدق میکند و مدل همگرا نمیشود.



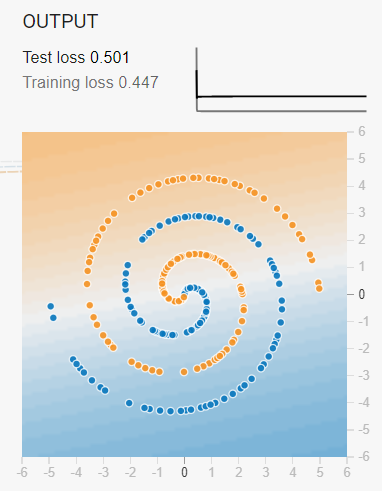
تابع Relu:

از توضیحات اضافه پرهیز میکنم. همان موارد قبل



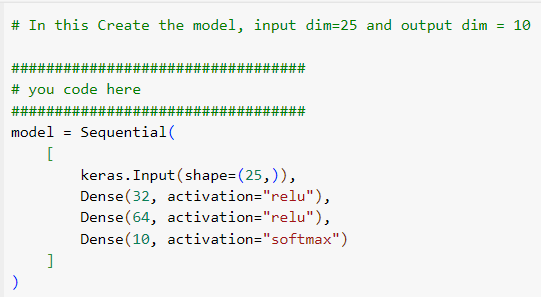
تابع Linear:

این مورد نیز طبق انتظار نخواهد توانست.



سوال 4-

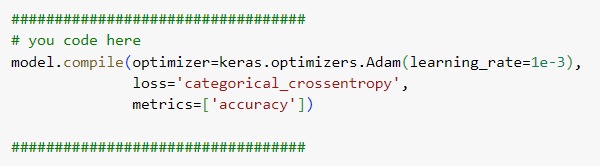
مدلی که من برای این مسئله انتخاب کردم:



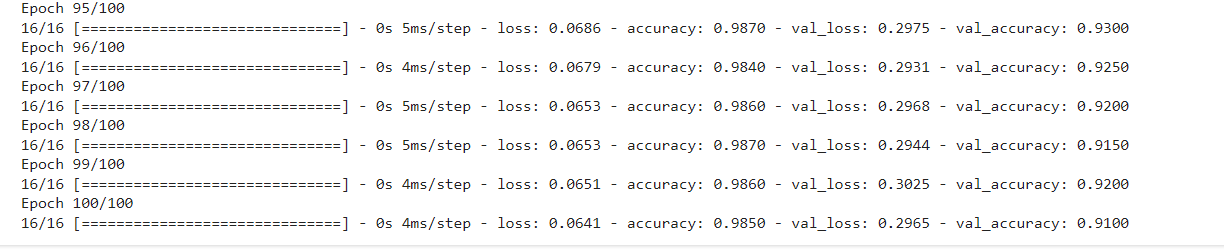
یک لایه ورودی (این لایه برای اینکه بتوانیم از مدل summary بگیریم لازم است) داریم و سه لایه خطی که دو تای اولی از تابع فعالساز relu و آخری از softmax استفاده میکند زیرا با یک مسئله دسته بندی مواجه هستیم و برای این لازم است مقادیر احتمالی برای هر کلاس داشته باشیم تا بیشترین احتمال را به عنوان کلاس نهایی انتخاب کنیم. به همین دلیل تعداد نورون لایه آخر به اندازه تعداد کلاسها یعنی 10 تاست.

از سه لایه بیشتر استفاده نکردم زیرا با دیتاست تقریبا ساده‌ای مواجه هستیم و نیازی به پیچیده کردن شبکه نیست. پیچیده کردن شبکه در اینجا صرفا باعث دیرتر همگرا شدن و در نهایت overfitting میشود زیرا شبکه با ظرفیت زیادی که دارد، همه جزئیات داده آموزش را حفظ میکند و در نتیجه تعمیم دهی را از دست میدهد.

از آنجا که تابع فعالساز معمول در شبکه های عصبی، relu است، از ابتدا از همین تابع استفاده کردم و به نتیجه‌ی خوبی هم رسیدم.

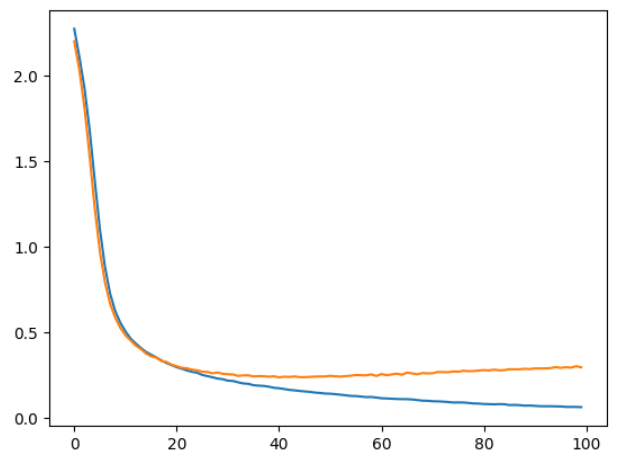


برای بهینه ساز از adam با نرخ یادگیری 0.001 استفاده کردم و برای تابع ضرر از categorical crossentropy استفاده کردم که برای classification معمولا از همین تابع استفاده میشود. برای متریک هم از accuracy استفاده کردم.



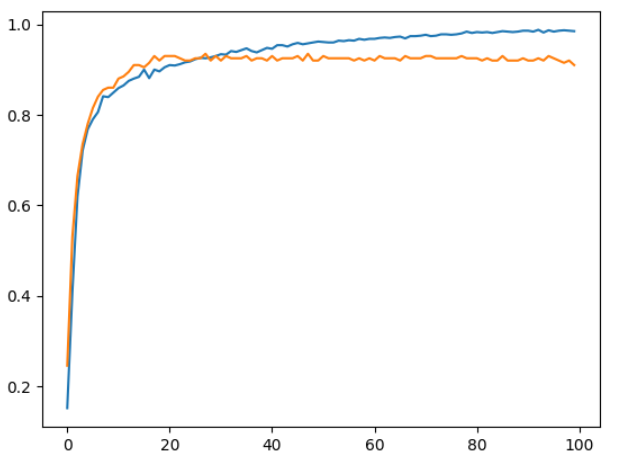
در چند epoch آخر، accuracy بر روی داده آزمایش در حال کاهش است و این نشان میدهد آموزش باید متوقف شود زیرا به سمت overfit شدن داریم حرکت میکنیم.

نمودار loss:



همانطور که مشخص است، از حدود epoch 40 به بعد، دیگر مدل قدرت تعمیم دهی خود را نمیتواند بیشتر کند و به سمت overfitting حرکت میکند. ولی نمودار بر روی داده آموزش همواره در حال کاهش است.

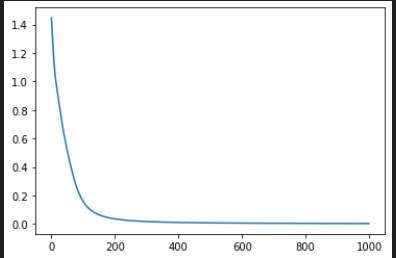
نمودار accuracy:



در اینجا هم همان قضیه قابل مشاهده است. از epoch 40 به بعد پیشرفتی روی داده آزمایش حاصل نمیشود.

سوال 5-

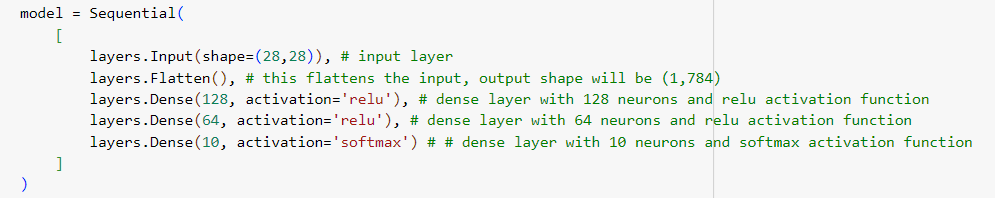
نمودار loss نهایی:



توضیح کد به صورت کامنت قرار داده شد.

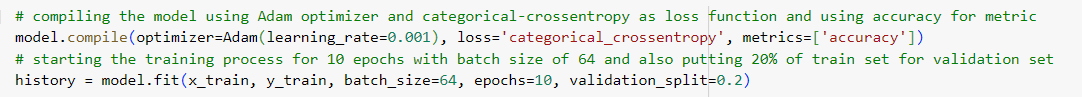
سوال 6-

توضیح کد به صورت کامنت در کد قرار داده شده است.



از سه لایه خطی استفاده کردم که دوتای اولی از تابع فعالساز relu استفاده کردم زیرا معمولا از همین تابع در لایه‌های میانی شبکه های عصبی استفاده می‌شود، به دلیل اینکه vanishing gradient را تا حد زیادی رفع میکند. و لایه آخر از softmax استفاده کردم زیرا یک مسئله دسته بندی داریم و یک راه معمول، گذاشتن نورون به تعداد کلاسها در لایه آخر و استفاده از تابع softmax برای بردن مقادیر خروجی به توزیع احتمالی است تا بیشترین احتمال به عنوان کلاس نهایی استفاده شود.

تعداد لایه‌ها را زیاد نکردم تا مدل پیچیده نشود و به سمت overfitting نرود. از طرفی تعداد نورونها را سعی کردم زیاد بگیرم تا بتواند در همان لایه ویژگی های مختلف را یاد بگیرد. همچنین تعداد نورون را در لایه بعدی کمتر کردم زیرا این ویژگیها به نوعی ترکیب ویژگیهای قبلی هستند و سطح بالاترند و تعدادشان کمتر است.



برای تابع ضرر، از categorical crossentropy استفاده کردم که معمولا برای دسته بندی از این استفاده میشود. برای بهینه ساز هم از adam با نرخ یادگیری 0.001 استفاده کردم.

