به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

درس یادگیری عمیق و کاربردها

تمرین شماره ۱

امیر محمد کریمی

۸۱۰۱۹۴۳۸۳

اسفند ماه 1397

فهرست

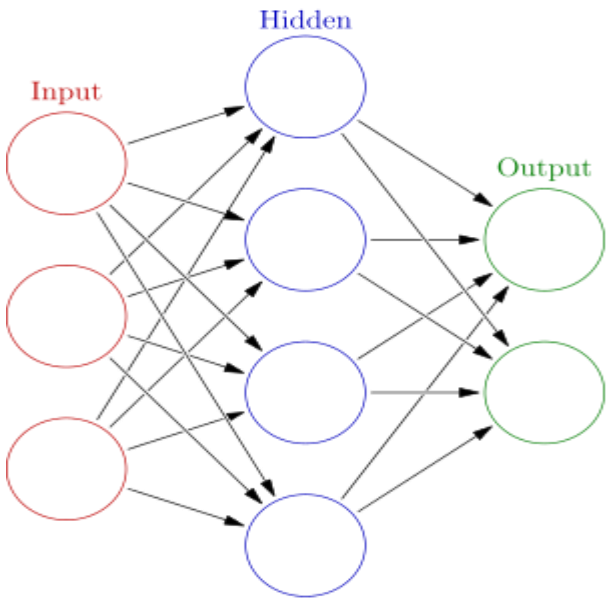
سوال ۱

در این قسمت، یک شبکه‌ی عصبی با یک لایه‌ی مخفی ۲۰۰ نورونی طراحی کردیم که تابع فعالسازی آن، relu و loss آن hinge loss می‌باشد. این شبکه، ۷۸۴ نورون ورودی، ۲۰۰ نورون پنهان و ۱۰ نورون خروجی (به تعداد کلاس‌های دیتاست مورد نظر) دارد.

این شبکه‌ی عصبی، روی داده‌های MNIST آموزش خواهد دید. دیتاست MNIST دیتاست عکس‌های دست نوشته از اعداد ۰ تا ۹ است که هر کدام از این عکس‌ها، ۲۸*۲۸ هستند. تعداد داده‌های آموزش آن، ۶۰۰۰۰ و تعداد داده‌های تست آن، ۱۰۰۰۰ می‌باشد. درساختن این شبکه، تعداد لایه‌ها و تعداد نورون‌ها به صورت پارامتری داده می‌شود و ساختار شبکه مستقل از تعداد لایه‌ها و تعداد نورون‌های هر لایه است.

برای پیاده‌سازی این شبکه، یک کلاس NeuralNetwork داریم که تعداد نورون‌های هر لایه، ضریب آموزش، batch size، تعداد ماکزیمم epochها و فایل وزن‌ها (در صورتی که نیاز به آموزش نباشد و قصد خواندن وزن‌ها از یک فایل را داشته باشیم) را به عنوان ورودی دریافت می‌کند. متد train، این شبکه را با دادن ورودی مورد نظر و خروجی مطلوب، آموزش می‌دهد. متد test نیز این شبکه را روی داده‌های ورودی داده‌شده به آن، تست می‌کند.

ساختار شبکه‌ی مورد نظر به صورت زیر است:



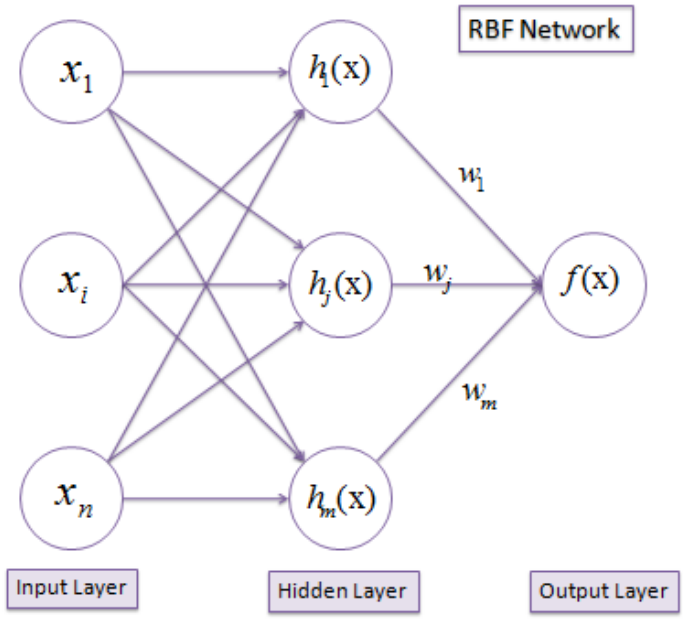
شکل ۱. ساختار شبکه‌ی Mutlilayer Perceptron

سوال ۲

در این بخش، یک شبکه‌ی Gaussian RBF می‌سازیم که همانند شبکه‌ی گفته‌شده در قسمت قبل، دارای ۲۰۰ نورون لایه‌ی پنهان می‌باشد. به طور کلی، شبکه های RBF بدین صورت هستند که بین لایه‌ی ورودی به لایه‌ی پنهان وزنی وجود ندارد. در عوض، هر نورون لایه‌ی پنهان همانند یک کلاسترعمل می‌کند (دارای یک مرکز و یک فاصله (sigma) می‌باشد) و در واقع سعی می‌کند تا نقاطی از ورودی که فاصله‌ی کمتری با مرکز خود دارند را بیابد و آن‌ها را در یک دسته‌بندی قرار دهد. برای این کار، هر نورون لایه‌ی پنهان یک تابع (معمولا تابع Gaussian) روی ورودی‌های خود اعمال می‌کند. سپس این خروجی‌ها را به صورت وزن دار به لایه‌ی بعدی می‌دهد. در نهایت در لایه‌ی خروجی، خروجی هر نورون ایجاد می‌شود و در استفاده‌ی classification این شبکه، بین خروجی‌ها ماکزیمم گرفته می‌شود و اندیس این نورون ماکزیمم به عنوان خروجی شبکه در نظر گرفته می‌شود.

در پیاده‌سازی این شبکه، از کلاس NeuralNetwork، یک فرزند را به ارث می‌بریم و بعضی از متدهای آن را مجددا پیاده‌سازی می‌کنیم.

ساختار این شبکه به صورت زیر است:



$$f(x) = \sum_{j=1}^{m} w_j h_j(x)$$

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

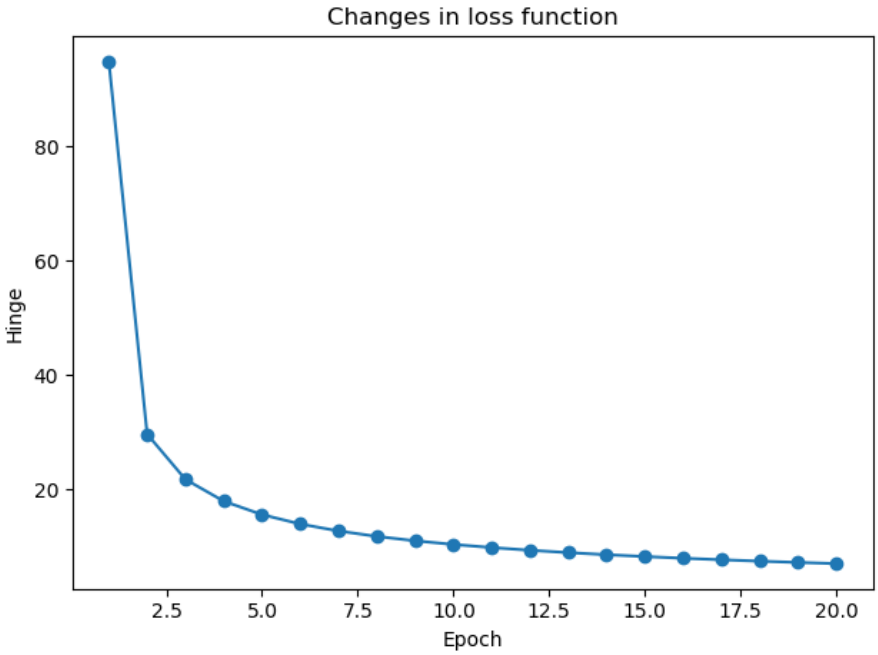
شکل ۲ . ساختار کلی شبکه‌ی Gaussian RBF

سوال ۳

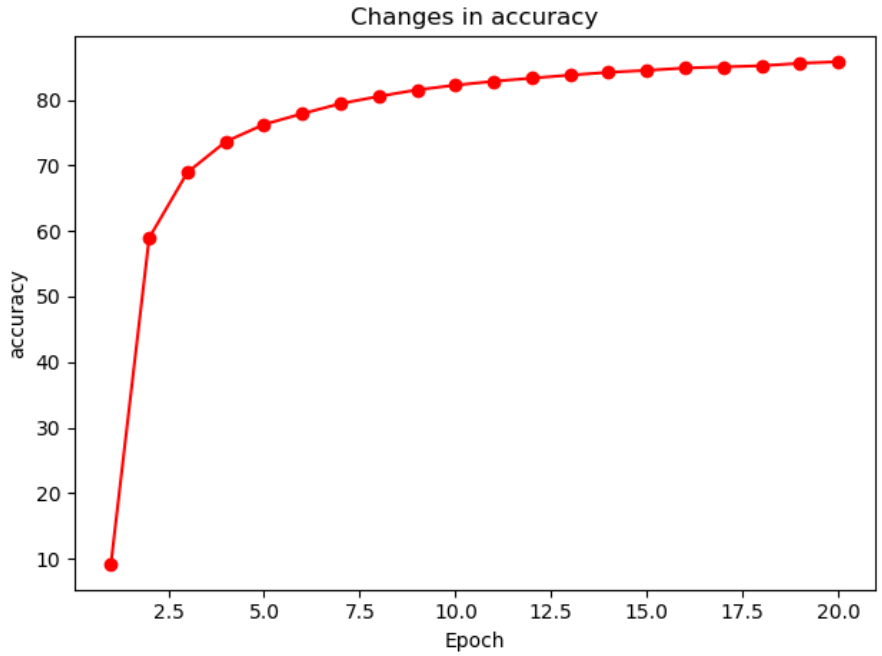
در این بخش، شبکه‌های گفته‌شده را با مقداردهی اولیه‌ی وزن‌ها به صورت رندوم و با روش Stochastic Gradient Descent، آموزش می‌دهیم. این روش بدین صورت شبکه را آموزش می‌دهد که در هر epoch، تعداد مشخصی داده (که با پارامتر batch size مشخص می‌شود) را به شبکه می‌دهیم، خطای متناظر با آن‌ها را محاسبه می‌کنیم و در نهایت این خطا را روی وزن‌ها (و مراکز در شبکه‌ی RBF) اعمال می‌کنیم.

شبکه‌ی MLP:

میزان خطای این شبکه (با ضریب آموزش ۰٫۰۱) و همچنین دقت این شبکه روی داده‌های تست به صورت زیر می‌باشد:



شکل ۳. میزان خطای شبکه‌ی MLP در ۲۰ epoch

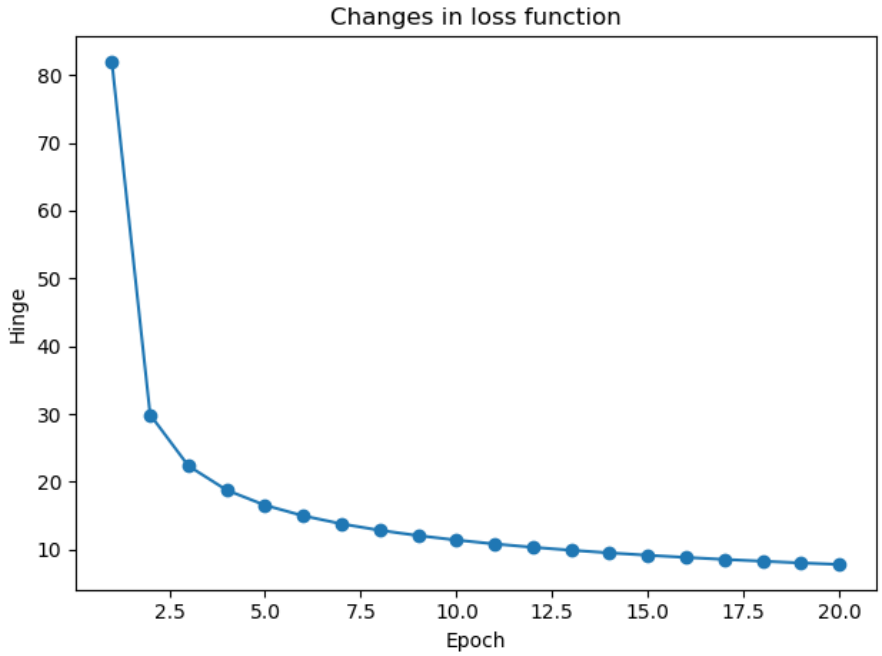


شکل ۴ . میزان دقت شبکه‌ی MLP در ۲۰ epoch

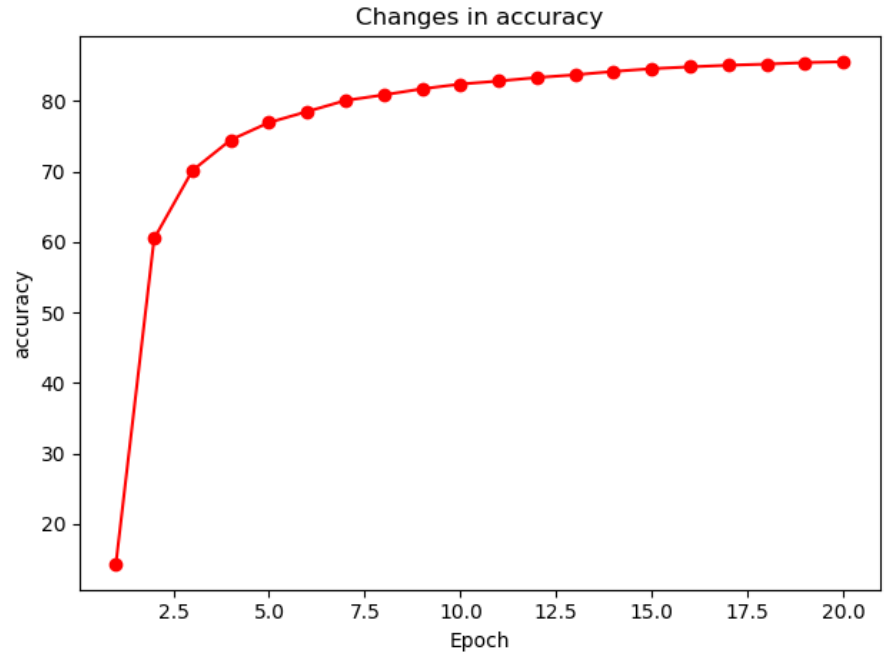
با بررسی نمودارهای بالا، مشاهده می‌کنیم که در ۲۰ epoch، دقت شبکه به ۸۵,۴۹ درصد و خطای آموزش به ۰,۸۲ می‌رسد.

شبکه‌ی RBF:

میزان خطای این شبکه (با ضریب آموزش ۰,۰۱) و همچنین دقت این شبکه روی داده‌های تست به صورت زیر می‌باشد: (متاسفانه به دلیل زمانبر بودن آموزش این شبکه، تنها خروجی این شبکه در این قسمت از تمرین به دست آمده است)



شکل ۵. میزان خطای شبکه‌ی RBF در ۲۰ epoch



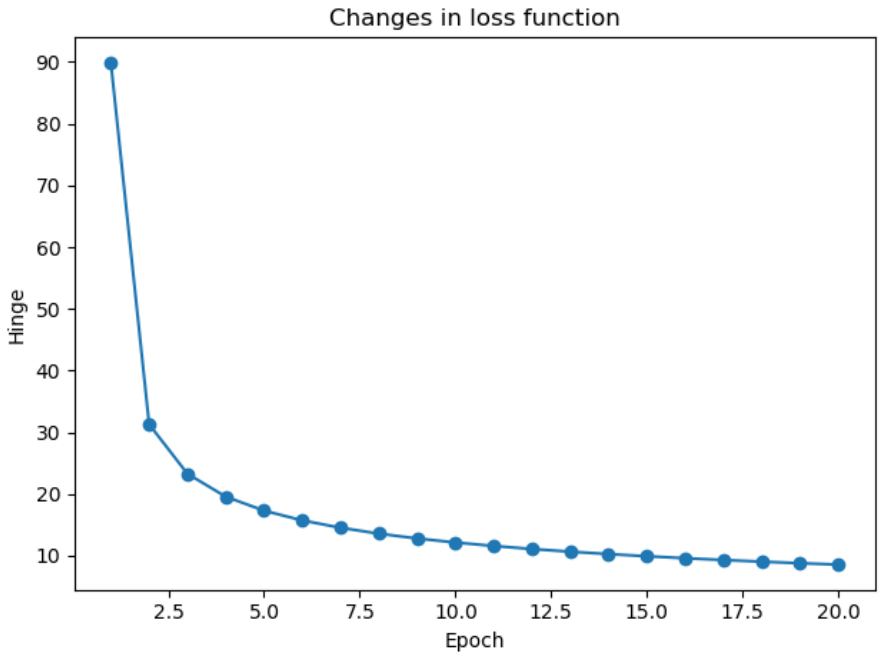
شکل ۶. میزان دقت شبکه‌ی RBF در ۲۰ epoch

با مقایسه‌ی دو شبکه‌ی گفته‌شده، مشاهده می‌کنیم که شبکه‌ی MLP، سریعتر آموزش می‌بیند اما دقت شبکه‌ی RBF در دراز مدت بیشتر خواهد بود.

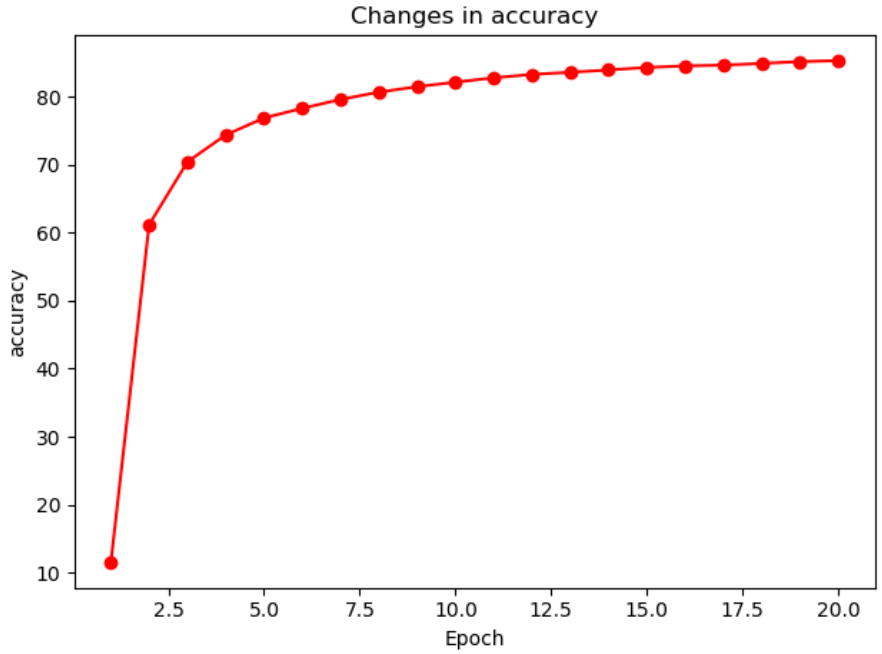
سوال ۴

آموزش با تابع فعالسازی softplus:

میزان خطای داده‌های آموزش و دقت شبکه در نمودارهای زیر بیان شده است:



شکل ۷. میزان خطای شبکه‌ی MLP در ۲۰ epoch با تابع فعالسازی softplus

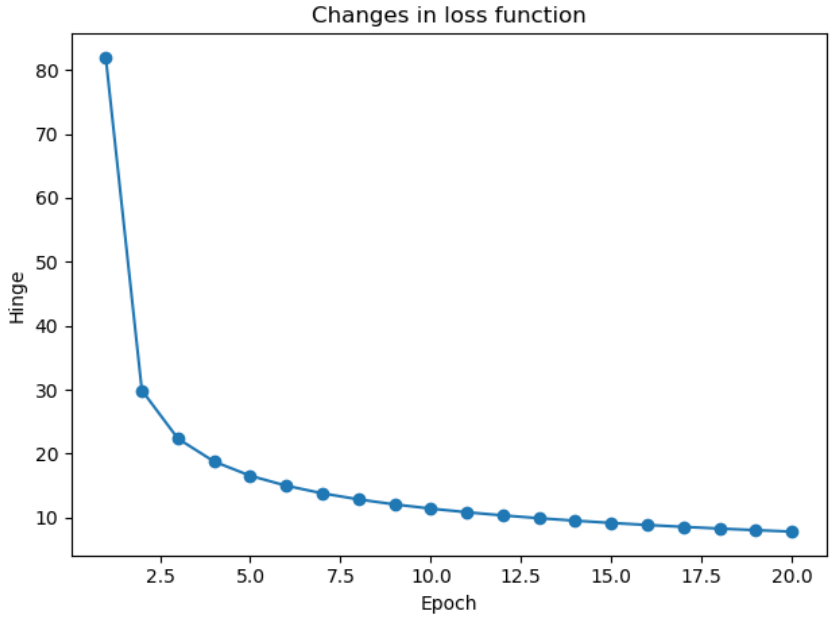


شکل ۸. میزان دقت شبکه‌ی MLP در ۲۰ epoch با تابع فعالسازی softplus

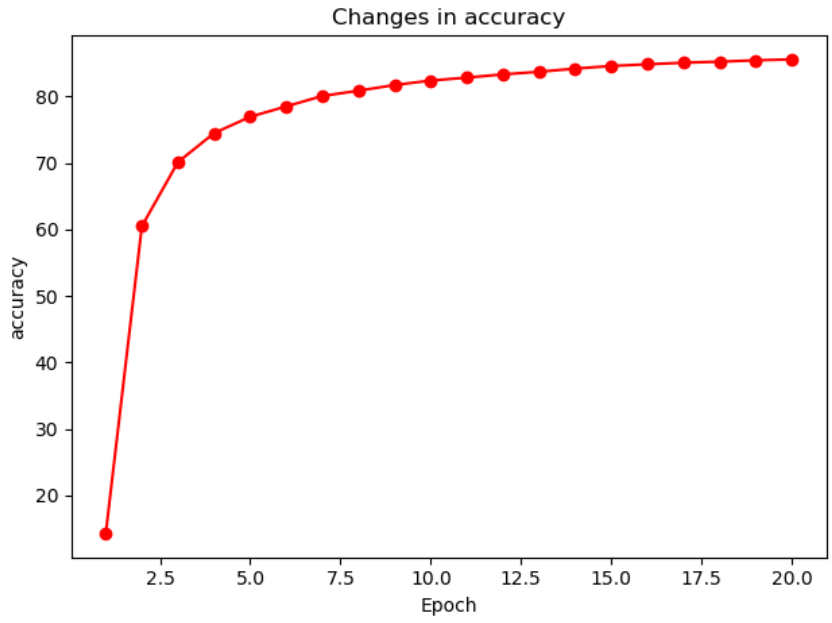
با استفاده از تابع فعالسازی softplus، ماکزیمم دقت شبکه به ۸۶ درصد و خطای شبکه به حدود ۵,۰۴ می‌رسد.

آموزش با تابع فعالسازی leaky relu:

میزان خطای داده‌های آموزش و دقت شبکه در نمودارهای زیر قابل مشاهده است:



شکل ۹ . میزان خطای شبکه‌ی MLP در ۲۰ epoch با تابع فعالسازی leaky relu



شکل ۹ . میزان خطای شبکه‌ی MLP در ۲۰ epoch با تابع فعالسازی leaky relu

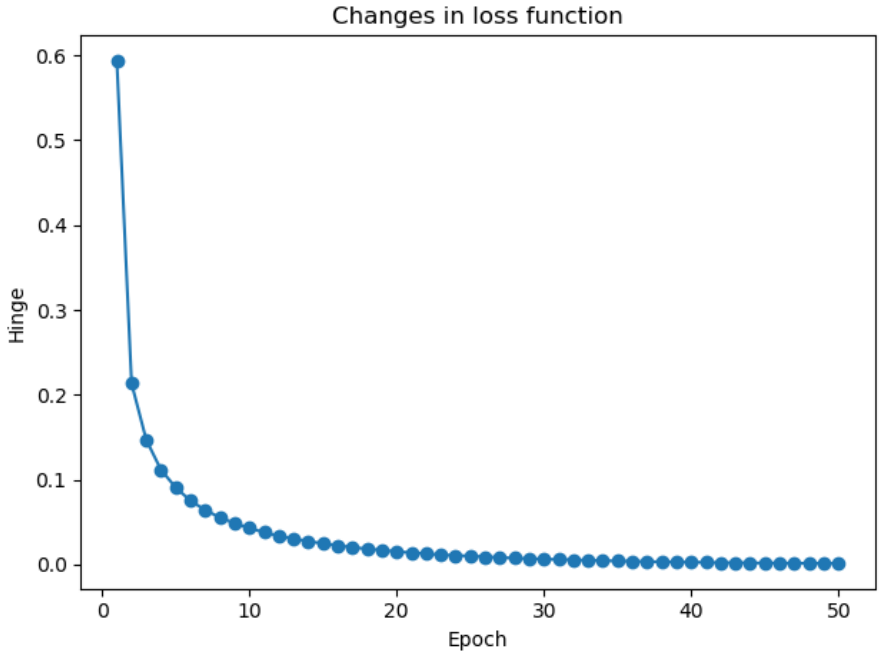
مقایسه نتایج:

با مقایسه خروجی شبکه عصبی با توابع فعالسازی مختلف به نتایج زیر می‌رسیم:

- آموزش در حالتی که از تابع softplus استفاده می‌شود، از سایر حالات شیب ملایم‌تری دارد و آرامتر آموزش می‌بیند. این تابع به دلیل ساختار خود (که در نقاط نزدیک به صفر، خروجی را ۰ نمی‌کند و به صورت ملایم کاهش می‌دهد)، سعی دارد هیچ نورونی را حذف نکند.
- تابع leaky relu، سعی دارد تا مشکل در نظر نگرفتن برخی از نورون‌ها در relu را حل کند و این کار را با در نظر گرفتن یک ضریب alpha برای نتایج کوچکتر از ۰ انجام می‌دهد.
- تابع relu تابعی ساده‌تر نسبت به سایر توابع می‌باشد و با مشکل vanishing gradient مقابله می‌کند. اما به دلیل فیلتر کردن خروجی در حالتی که ورودی آن کوچکتر از ۰ است، برخی از نورون‌ها را در بعضی حالات در نظر نمی‌گیرد.

سوال ۵

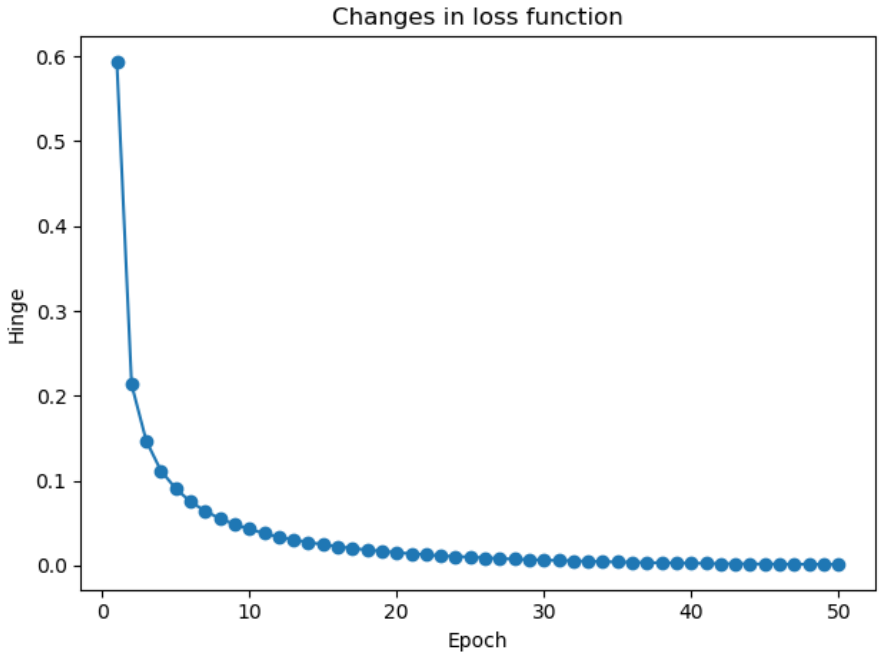
در حالت کلی برای overfit شدن شبکه، باید شبکه‌ی پیچیده‌ای بسازیم (در واقع تعداد پارامترها را زیاد کنیم) و خطای آموزش را تا حد ممکن به ۰ نزدیک کنیم. برای این کار، یک لایه‌ی مخفی دیگر با ۱۰۰ نورون اضافه کردیم و تعداد نورون‌های لایه‌ی اول پنهان را نیز به ۸۰۰ نورون افزایش دادیم. همچنین، تعداد epochها را نیز به ۵۰ افزایش دادیم. میزان خطا به صورت زیر تغییر کرد:



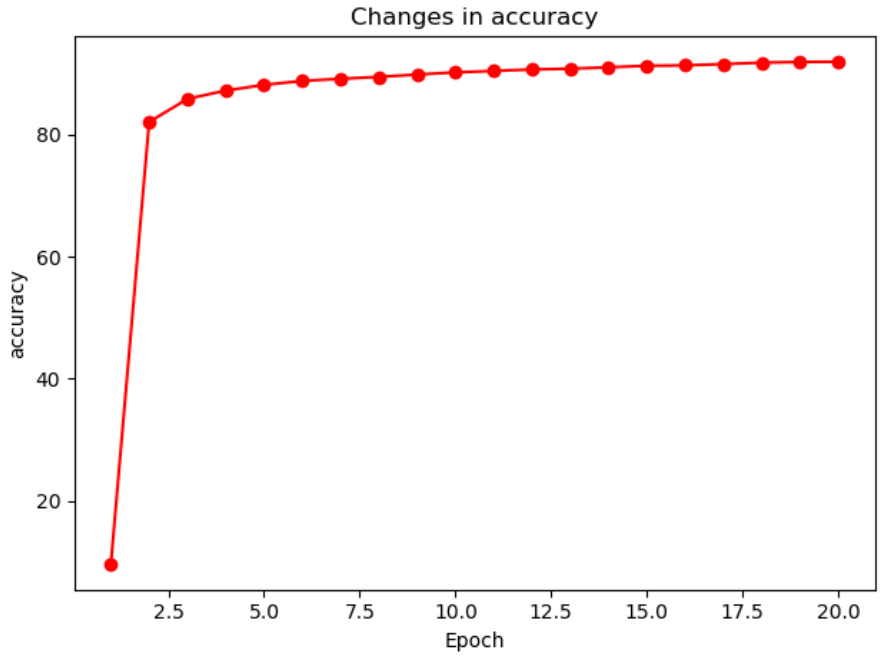
شکل ۱۰. میزان خطای شبکه‌ی MLP در ۵۰ epoch در حالت overfit شده

دقت داده‌های تست در حالتی که شبکه overfit شده است، به حدود ۷۰ درصد کاهش می‌یابد.

پس از اضافه کردن L2 Regularization، خطای شبکه با شیب کمتری کاهش می‌یابد اما دقت شبکه تا ۸۹ درصد افزایش می‌یابد. نمودار خطای شبکه و دقت آن در حالتی که از L2 Regularization استفاده شده است، در شکل‌های زیر آمده است:



شکل ۱۱ . میزان خطای شبکه‌ی MLP در ۵۰ epoch پس از استفاده از L2 Regularization



شکل ۱۲ . میزان دقت شبکه‌ی MLP در ۵۰ epoch پس از استفاده از L2 Regularization

سوال ۶

در روش اول، ابتدا یک شبکه با یک لایه را در ۲ epoch آموزش می‌دهیم (واریانس بهینه‌ی وزن‌های شبکه پس از بررسی‌های انجام شده، ۰/۰۱ به دست آمده است). سپس یک لایه‌ی دیگر نیز به آن اضافه می‌کنیم و شبکه‌ی حاصل را برای یک epoch دیگر آموزش می‌دهیم.

در این حالت، دقت شبکه ۷۳/۵۲ درصد می‌باشد.

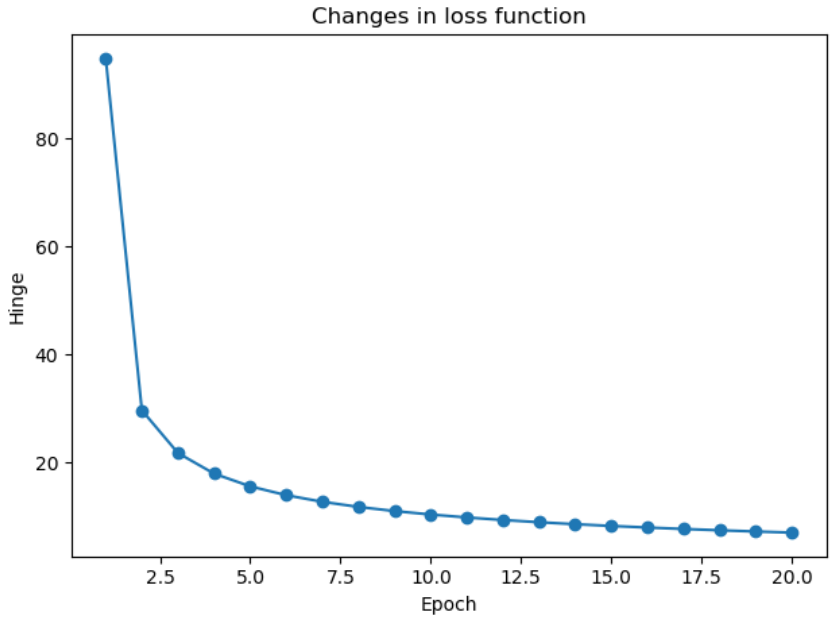
در حالت دوم، یک شبکه‌ی دو لایه را با واریانس ۰/۰۱ برای ۳ epoch آموزش می‌دهیم. در این حالت دقت شبکه ۷۲/۴۹ به دست آمد.

با مقایسه‌ی دقت این دو شبکه، در می‌یابیم که در حالت اول، دقت شبکه اندکی از حالت دوم بهتر است.

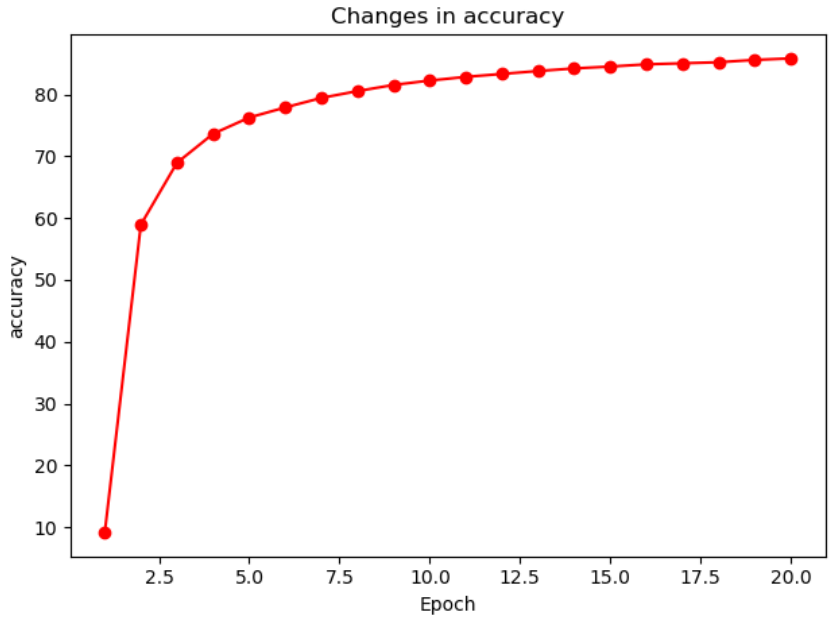
سوال ۷

ورودی به صورت خام:

نتایج این حالت به صورت زیر می‌باشد (مشابه سوال سوم):



شکل ۱۳ . میزان خطای شبکه‌ی MLP در ۲۰ epoch

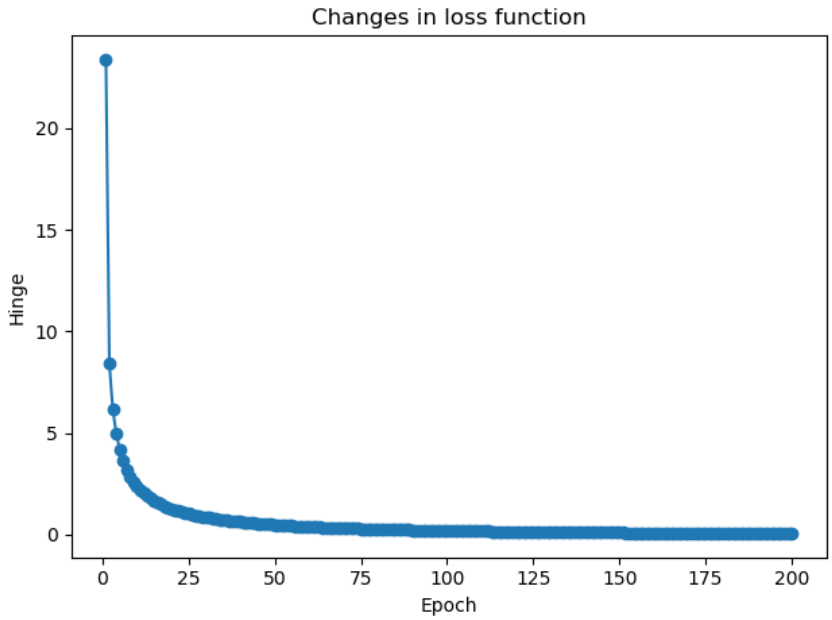


شکل ۱۴ . میزان دقت شبکه‌ی MLP در ۲۰ epoch

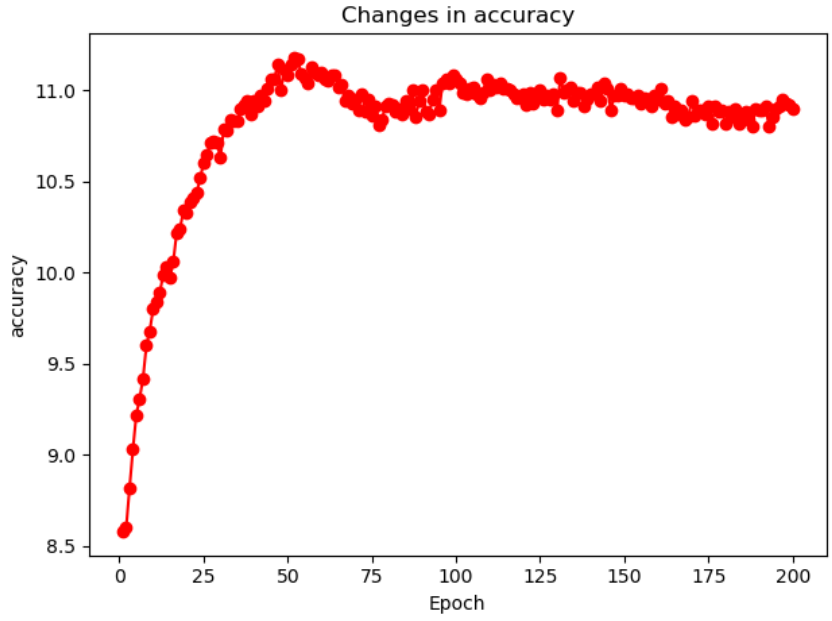
ورودی پس از اجرای PCA و کاهش بعد به ۱۲۸:

PCA، یک روش برای کاهش بعد داده است. هدف از آن این است که داده‌هایی در هر کلاس که بیشترین پراکندگی را دارند نگه داریم.

میزان دقت و خطای شبکه در این حالت به صورت زیر می‌باشد:



شکل ۱۵ . میزان خطای شبکه‌ی MLP در ۲۰۰ epoch پس از استفاده از PCA و کاهش بعد به ۱۲۸

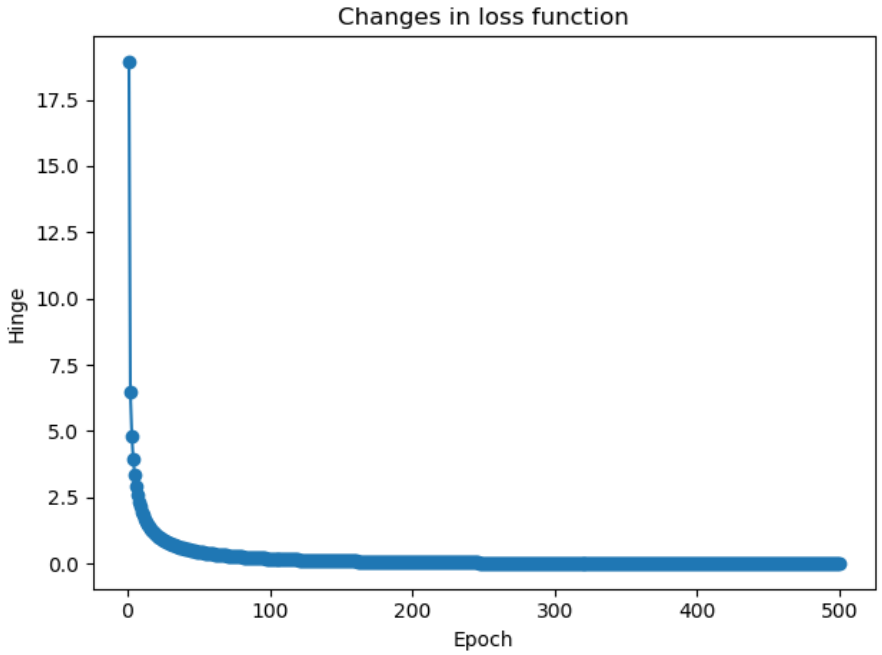


شکل ۱۶ . میزان خطای شبکه‌ی MLP در ۲۰۰ epoch پس از استفاده از PCA و کاهش بعد به ۱۲۸

با مشاهده‌ی این نمودارها، در می‌یابیم که سرعت همگرایی داده‌ها بسیار کم است و با ۲۰۰ epoch هم به دقت مطلوبی نرسیدیم و نیاز به تعداد بسیاری epoch داریم تا شبکه بتواند به خوبی آموزش ببیند.

ورودی پس از اجرای PCA و کاهش بعد به ۶۴:

میزان خطای شبکه در این حالت به صورت زیر می‌باشد:

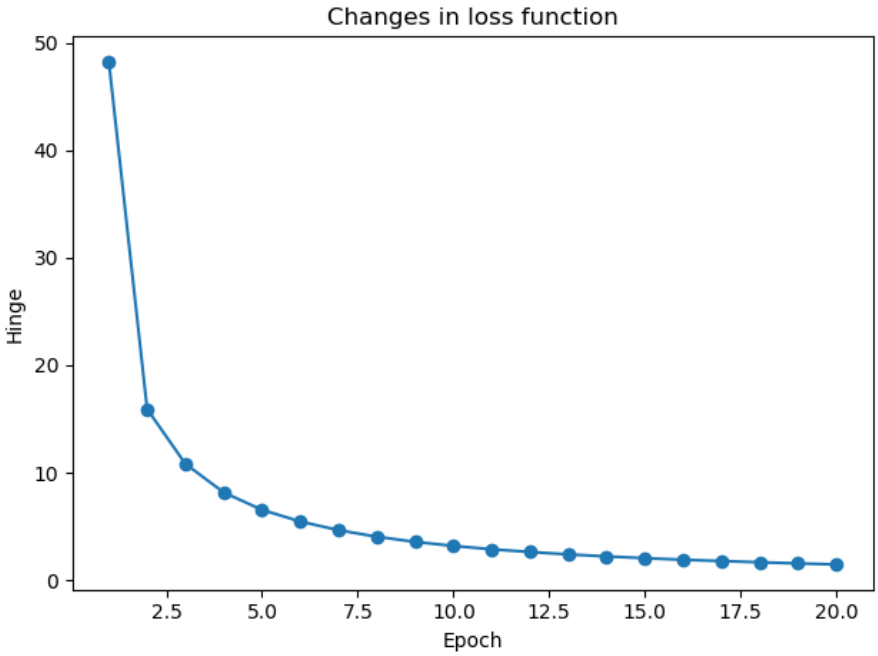


شکل ۱۷ . میزان خطای شبکه‌ی MLP در ۵۰۰ epoch پس از استفاده از PCA و کاهش بعد به ۶۴

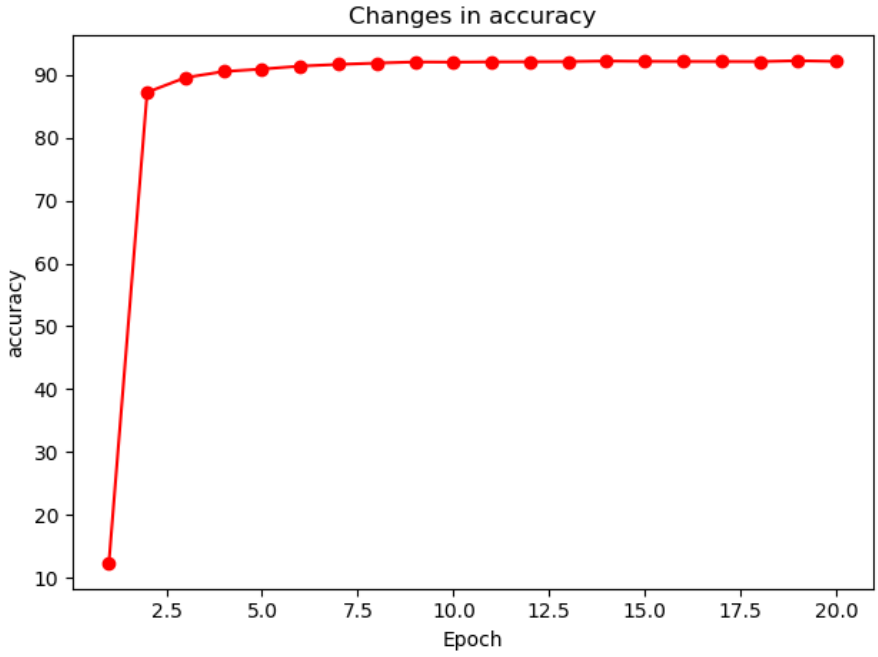
دقت در این حالت از ۱۰،۵ درصد بالاتر نمی‌رود و همگرایی آن نیاز به زمان زیادی دارد.

ورودی پس از نرمال کردن داده‌ها:

در این حالت نمودار خطا و دقت به صورت زیر است:



شکل ۱۸ . میزان خطای شبکه‌ی MLP در ۲۰ epoch در حالت نرمالایز شده‌ی داده‌های ورودی



شکل ۱۹ . میزان دقت شبکه‌ی MLP در ۲۰ epoch در حالت نرمالایز شده‌ی داده‌های ورودی

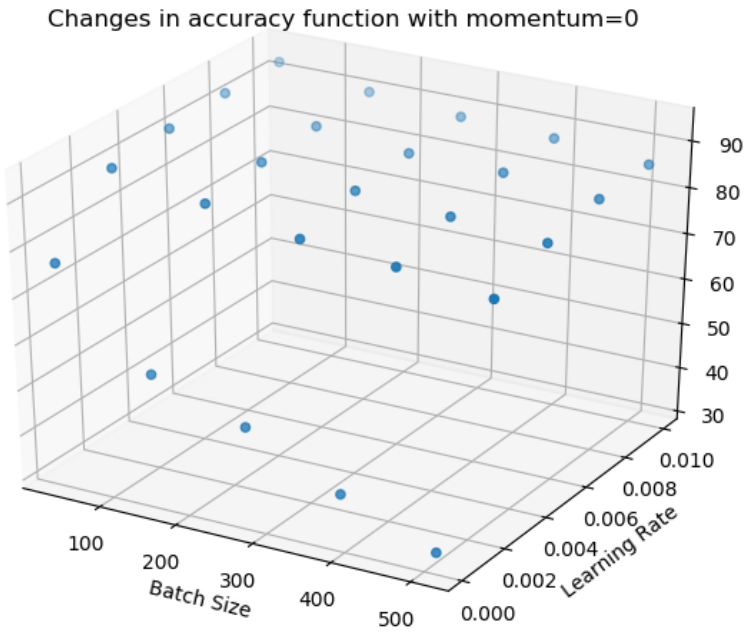
در این حالت، سرعت همگرایی نسبت به حالتی که داده‌ها به صورت خام به شبکه داده شده‌اند، اندکی بهتر است و شبکه با سرعت بیشتری همگرا می‌شود.

سوال ۸

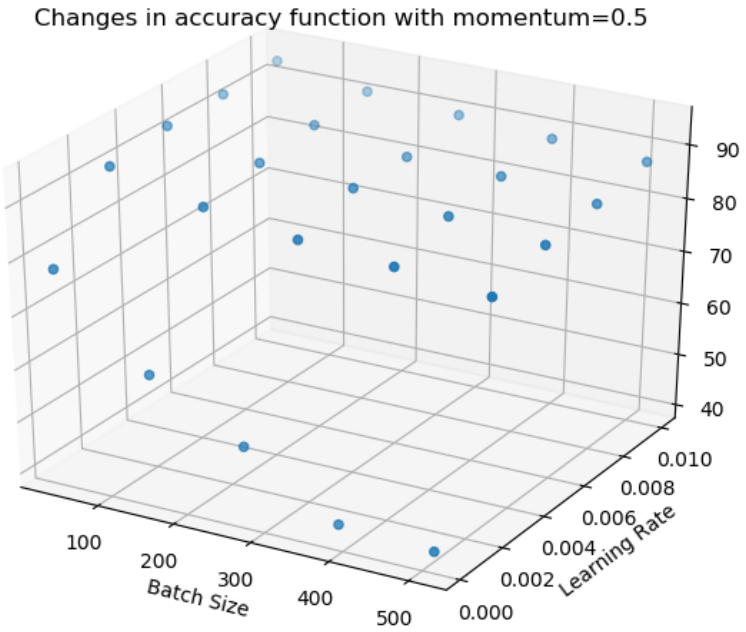
به دلیل زمان زیاد آموزش شبکه‌ی RBF و کمبود زمان، نتایج این بخش به دست نیامده است.

سوال ۹

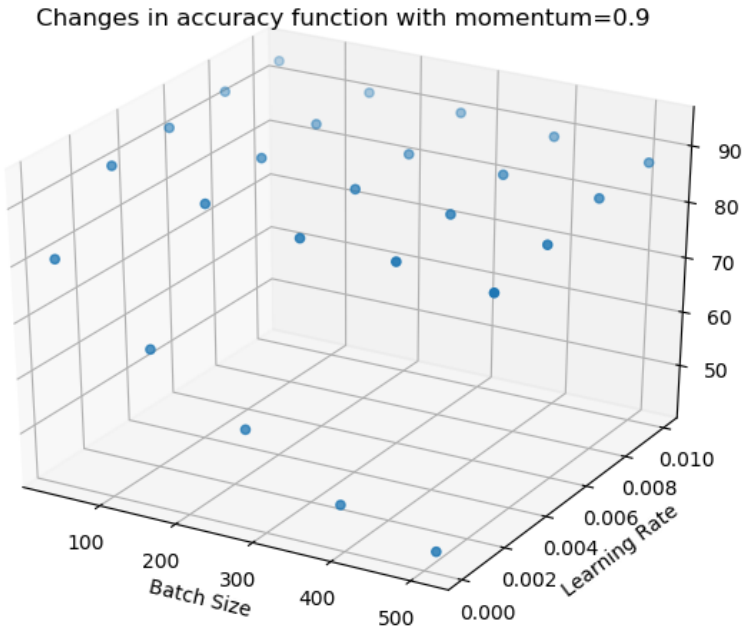
در این بخش، شبکه را با مومنتوم‌های ۰، ۰/۵، ۰/۹ و ۰/۹۹ و ۵ ضریب آموزش مختلف (بین ۰/۰۰۰۱ و ۰/۰۱) و ۵ batch size مختلف (بین ۳۲ تا ۵۱۲) آموزش می‌دهیم. نتایج به صورت زیر به دست آمده است:



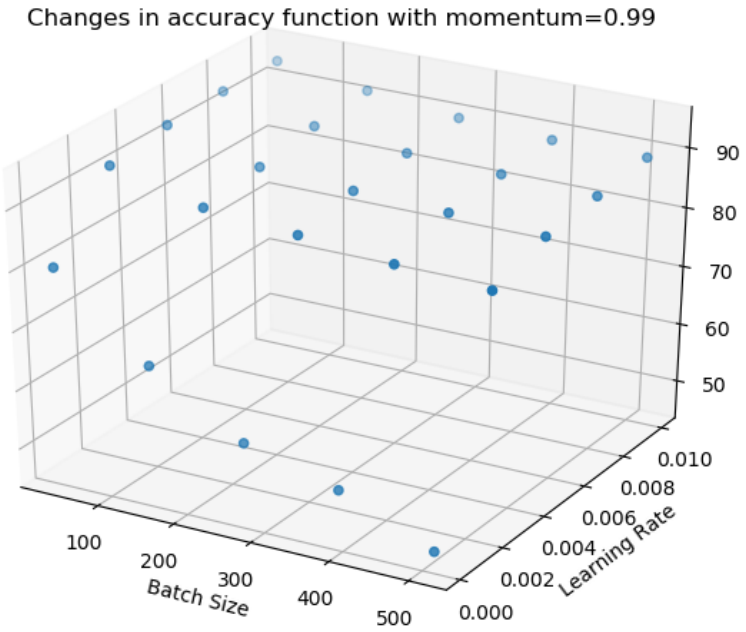
شکل ۲۰. نمودار تغییرات دقت نسبت به تغییر ضریب آموزش و batch size با مومنتوم ۰



شکل ۲۱ . نمودار تغییرات دقت نسبت به تغییر ضریب آموزش و batch size با مومنتوم ۰/۵



شکل ۲۲ . نمودار تغییرات دقت نسبت به تغییر ضریب آموزش و batch size با مومنتوم ۰/۹



شکل ۲۳ . نمودار تغییرات دقت نسبت به تغییر ضریب آموزش و batch size با مومنتوم ۰/۹۹

با مقایسه‌ی نمودارهای به دست آمده، به نتایج زیر می‌رسیم:

- در حالتی که مومنتوم نداریم، دقت شبکه به مراتب کمتر از حالاتی است که مومنتوم داریم.
- بین حالاتی که در شبکه مومنتوم داریم، دقت شبکه با افزایش مونتوم افزایش می‌یابد. این افزایش نیز در نقاطی که ضریب آموزشی کمتری دارند، به صورت واضحتری مشخص است.

سوال ۱۰

به دلیل زمان زیاد آموزش شبکه‌ی RBF و کمبود زمان، نتایج این بخش به دست نیامده است اما انتظار می‌رود نتایج این بخش نیز مشابه بخش قبل باشد.

پیوست 1: روند اجرای برنامه

برای اجرای برنامه، کافیست داده‌های MNIST را در فولدری به نام data در کنار فولدر codes قرار دهید. سپس، فایل main.py را اجرا کنید. بخش‌های مختلف این تمرین نیز به صورت کامنت در فایل main.py و network.py قرار گرفته است و با uncomment کردن آن‌ها می‌توان نتایج بخش‌های مختلف تمرین را مشاهده کرد.

مراجع

[1] Goodfellow, I., Bengio, Y. and Courville, A. (n.d.). Deep learning.

[2] Brownlee, J. (2019). How to Calculate Principal Component Analysis (PCA) from Scratch in Python. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/ [Accessed 4 Mar. 2019].