به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

پرسش ۱ و ۳	نام و نام خانوادگی	آرمان مجیدی
	شماره دانشجویی	۸۱۰۱۰۰۲۰۵
پرسش ۲ و ۴	نام و نام خانوادگی	آرین فیروزی
	شماره دانشجویی	۸۱۰۱۰۰۱۹۶
	مهلت ارسال پاسخ	**۱۴۰۳.۰۸.۱۵**

فهرست

پرسش ۱- تحلیل و طراحی شبکه‌های عصبی چندلایه (MLP) ۲

۱-۱. طراحی MLP ۲

۱-۲. آموزش دو مدل متفاوت ۵

۱-۳. الگوریتم بازگشت به عقب ۸

۱-۴. بررسی هایپرپارامترهای مختلف ۱۲

پرسش ۲ – آموزش و ارزیابی یک شبکه عصبی ساده ۲۲

۲-۱. آموزش شبکه عصبی ۲۲

۲-۱-۱. تابع فوروارد ۲۲

۲-۱-۲. تابع بکوارد ۲۲

۲-۲. آموزش ۲۲

پرسش ۳ – Madaline ۲۴

۳-۱. الگوریتم‌های **MRI** و **MRII** ۲۴

۳-۲. نمودار پراکندگی داده‌ها ۲۴

۳-۳. آموزش مدل ۲۵

پرسش ۴ – MLP ۲۸

۴-۱. تعداد NaN ۲۸

۴-۲. ماتریس همبستگی ۲۸

۴-۳. رسم نمودار ۲۸

۴-۴. پیش پردازش ۲۹

۴-۵. پیاده سازی مدل ۳۰

۴-۶. آموزش ۳۰

۴-۷. تحلیل نتایج ۳۱

شکل‌ها

پرسش ۱

۱) **شکل ۱.۱** معماری مدل در بخش ۱-۱ - صفحه ۲

۲) **شکل ۱.۲** نمودار تابع هزینه مدل ۱-۱ - صفحه ۳

۳) **شکل ۱.۳** ماتریس آشفتگی مدل ۱-۱ - صفحه ۳

۴) **شکل ۱.۴** معماری مدل اول در بخش ۲-۱ - صفحه ۵

۵) **شکل ۱.۵** معماری مدل دوم در بخش ۲-۱ - صفحه ۵

۶) **شکل ۱.۶** هیستوگرام لایه مخفی و لایه انتهایی برای مدل اول بخش ۲-۱ - صفحه ۶

۷) **شکل ۱.۷** هیستوگرام لایه مخفی و لایه انتهایی برای مدل دوم بخش ۲-۱ - صفحه ۶

۸) **شکل ۱.۸** معماری مدل بخش ۳-۱ - صفحه ۸

۹) **شکل ۱.۹** نمودار تابع هزینه مدل بخش ۳-۱ (بدون بهینه‌ساز) - صفحه ۸

۱۰)**شکل ۱.۱۰** نمودار تابع هزینه مدل بخش ۱-۳ (بهینه‌ساز Adam) - صفحه ۹

۱۱)**شکل ۱.۱۱** نمودار تابع هزینه مدل بخش ۳-۱ (بهینه‌ساز Nadam) - صفحه ۹

۱۲)**شکل ۱.۱۲** نمودار تابع هزینه مدل بخش ۳-۱ (بهینه‌ساز RMSprop) - صفحه ۹

۱۳)**شکل ۱.۱۳** معماری مدل اول بخش ۴-۱ - صفحه ۱۲

۱۴)**شکل ۱.۱۴** نمودار مدل اول بخش ۴-۱ با بهینه‌ساز Adam و نرخ یادگیری ۰.۰۱ - صفحه ۱۲

۱۵)**شکل ۱.۱۵** نمودار مدل اول بخش ۴-۱ با بهینه‌ساز Nadam و نرخ یادگیری ۰.۰۱ - صفحه ۱۳

۱۶)**شکل ۱.۱۶** نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری ۰.۰۱ - صفحه ۱۳

۱۷)**شکل ۱.۱۷** نمودار مدل اول بخش ۴-۱ با بهینه‌ساز Adam و نرخ یادگیری ۰.۰۰۵ - صفحه ۱۴

۱۸)**شکل ۱.۱۸** نمودار مدل اول بخش ۴-۱ با بهینه‌ساز Nadam و نرخ یادگیری ۰.۰۰۵ - صفحه ۱۴

۱۹)**شکل ۱.۱۹** نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری ۰.۰۰۵ - صفحه ۱۴

۲۰)**شکل ۱.۲۰** نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Adam و نرخ یادگیری ۰.۰۰۱ - صفحه ۱۵

۲۱)**شکل ۱.۲۱** نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Nadam و نرخ یادگیری ۰.۰۰۱ - صفحه ۱۵

۲۲)**شکل ۱.۲۲** نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری ۰.۰۰۱ - صفحه ۱۵

۲۳)**شکل ۱.۲۳** معماری مدل دوم بخش ۱-۴ - صفحه ۱۶

۲۴)**شکل ۱.۲۴** نمودار مدل دوم بخش ۱-۴ با بهینه ساز Nadam و نرخ یادگیری ۰.۰۰۵ و تعداد لایه ۲ - صفحه ۱۶

۲۵)**شکل ۱.۲۵** ماتریس آشفتگی الگوریتم - Random Search صفحه ۱۷

۲۶)**شکل ۱.۲۶** ماتریس آشفتگی الگوریتم - Grid Search صفحه ۱۷

پرسش ۲

۲۷)**شکل ۲.۱** نمودار خطای آموزش و تست با ضریب یادگیری ۰.۰۰۰۱ - صفحه ۲۲

۲۸)**شکل ۲.۲** نمودار خطای آموزش با ضریب یادگیری ۰.۰۰۱ - صفحه ۲۳

۲۹)**شکل ۲.۳** نمودار خطای آموزش با ضریب یادگیری ۰.۰۱ - صفحه ۲۳

۳۰)**شکل ۲.۴** نمودار خطای آموزش با ضریب یادگیری ۰.۰۰۰۰۱ - صفحه ۲۳

پرسش ۳

۳۱)**شکل ۳.۱** نمودار پراکندگی داده‌ها - صفحه ۲۴

۳۲)**شکل ۳.۲** نمودار پراکندگی مدل با ۳ نورون - صفحه ۲۵

۳۳)**شکل ۳.۳** نمودار پراکندگی مدل با ۴ نورون - صفحه ۲۵

۳۴)**شکل ۳.۴** نمودار پراکندگی مدل با ۸ نورون - صفحه ۲۵

پرسش ۴

۳۵)**شکل ۴.۱** ماتریکس همبستگی دیتاست سوال ۴ - صفحه ۲۸

۳۶)**شکل ۴.۲** نمودار توزیع قیمت - صفحه ۲۹

۳۷)**شکل ۴.۳** نمودار پراکندگی قیمت بر حسب sqft_living - صفحه ۲۹

۳۸)**شکل ۴.۴** نمودار خطا برای مدل با یک لایه پنهان - صفحه ۳۰

۳۹)**شکل ۴.۵** نمودار خطا برای مدل با دو لایه پنهان - صفحه ۳۰

جدول‌ها

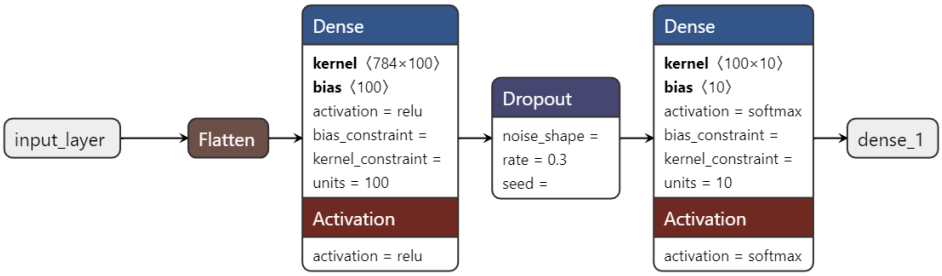
۱. **جدول ۱ :**هر کلاس و کلاسی که بیشتر با آن اشتباه گرفته می‌شوند - صفحه ۴

۲. **جدول ۲ :**بررسی اثر هایپرپارامترها بر روی ماتریس آشفتگی - صفحه ۱۷

پرسش ۱– تحلیل و طراحی شبکه‌های عصبی چندلایه (MLP)

۱-۱. طراحی MLP

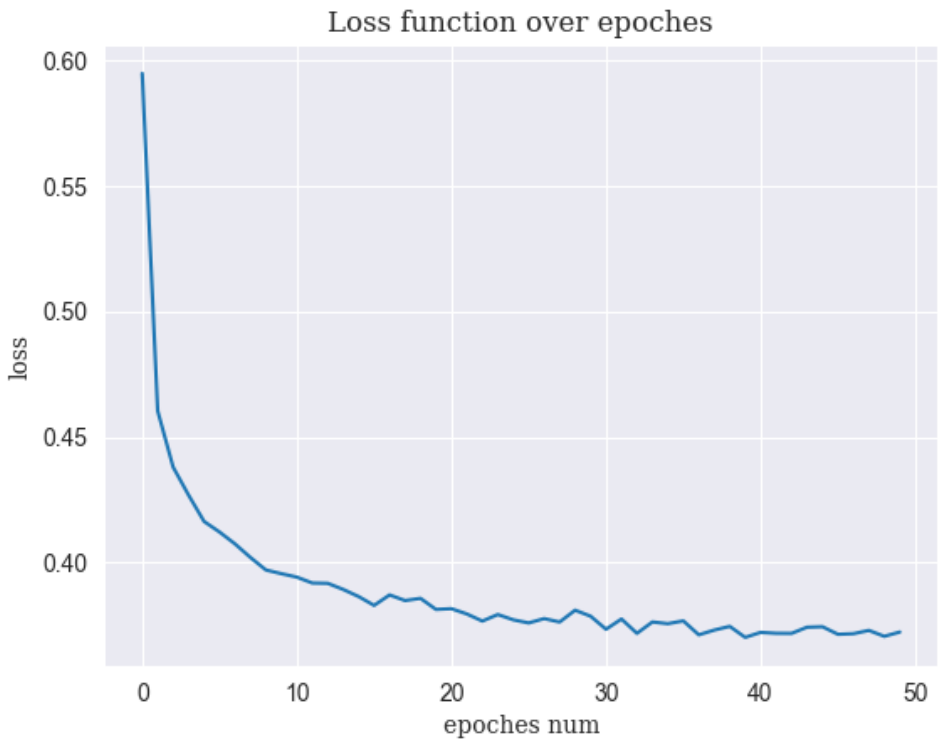
مدل این قسمت در شکل ۱ نشان داده شده‌است.



شکل۱.۱. معماری مدل در بخش ۱-۱

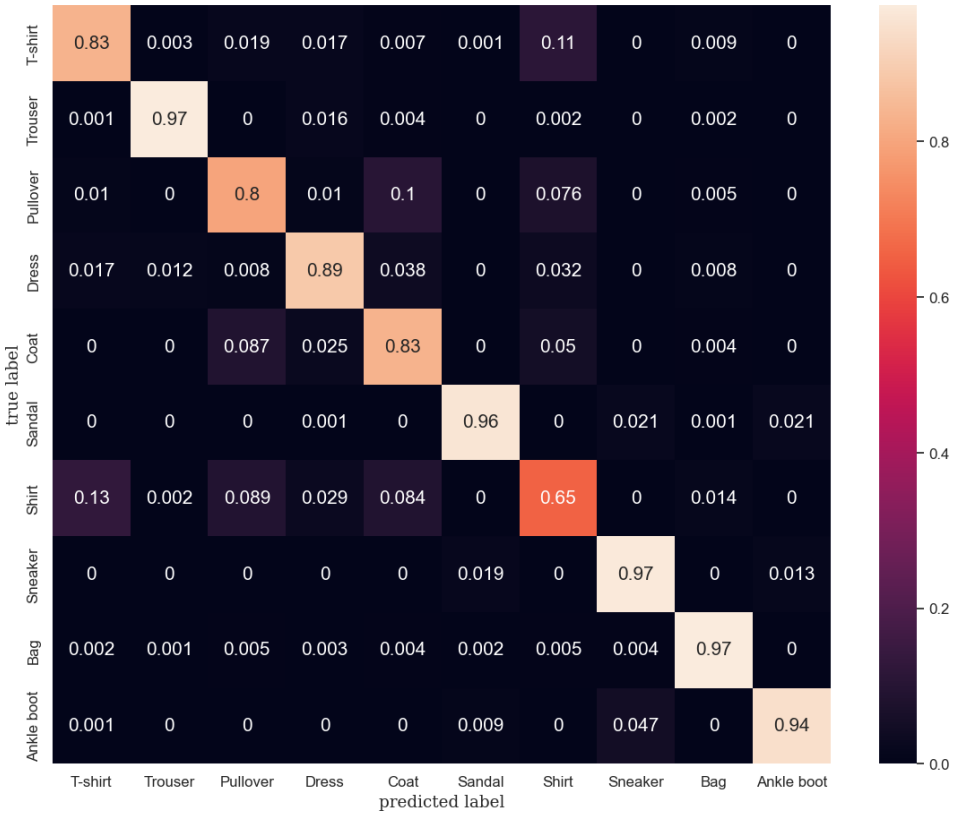
برای کامپایل مدل، از تابع هزینه SparseCategoricalCrossentropy استفاده شده‌است. همچنین، از بهینه‌ساز Adam با نرخ آموزش دیفالت (0.001) استفاده شده‌است. تنها معیار ما برای این مدل accuracy بوده‌است.

بعد از آموزش مدل به میزان 50 epoch، به دقت 0.8855 می‌رسیم. نمودار تابع هزینه برحسب شماره ایپاک در شکل ۱.۲ آمده‌است.



شکل ۱.۲. نمودار تابع هزینه مدل ۱-۱

- برای دریافت ماتریس آشفتگی، ابتدا داده تست را به مدل داده و کلاس‌های پیش‌بینی شده توسط مدل را دریافت می‌کنیم، سپس به کمک کتابخانه scikit-learn ماتریس آشفتگی را تولید می‌کنیم و به کمک کتابخانه seaborn آن را نمایش می‌دهیم. شکل ۱.۳، ماتریس آشفتگی بدست آمده از این مدل را نشان می‌دهد.



شکل ۱.۳. ماتریس آشفتگی مدل ۱-۱

- برای بدست آوردن کلاسی که بیشتر با هر یک از کلاس دیگر اشتباه گرفته می‌شود، از ماتریس آشفتگی کمک می‌گیریم. بدین ترتیب که درایه‌های قطری ماتریس آشفتگی را حذف می‌کنیم و سپس ماکزیمم هر سطر را بررسی می‌کنیم. اندیسی که در هر سطر ماکزیمم رخ داده، کلاسی که بیشتر با کلاس هدف ما اشتباه گرفته می‌شود را نشان می‌دهد. جدول ۱، کلاس‌ها و کلاسی که بیشتر با آن اشتباه گرفته می‌شوند را نشان می‌دهد.

جدول۱. هر کلاس و کلاسی که بیشتر با آن اشتباه گرفته می‌شوند

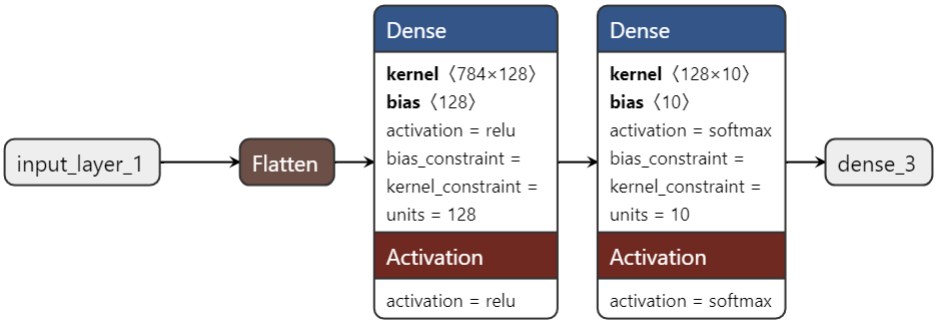
کلاس	کلاس اشتباه
T-shirt	Shirt
Trouser	Dress
Pullover	Coat
Dress	Coat
Coat	Pullover
Sandal	Sneaker
Shirt	T-shirt

Sandal	**Sneaker**
Pullover	**Bag**
Sneaker	**Ankle boot**

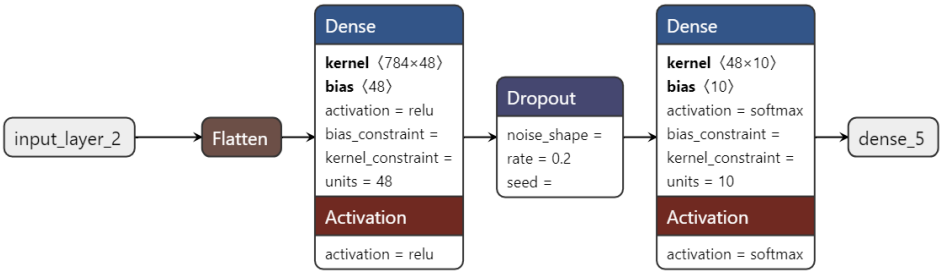
- برای بدست آوردن دو کلاسی که به طور کلی بیشتر با هم اشتباه گرفته می‌شوند، ابتدا باید ماتریس آشفتگی را با ترانهاده آن جمع کنیم. این کار بدین منظور است که مقدار درصد تشخیص کلاس i به جای کلاس j و مقدار درصد تشخیص کلاس j به جای کلاس i با یکدیگر جمع گردند. مجددا درایه‌های قطری ماتریس را صفر می‌کنیم. در انتها، به این نتیجه می‌رسیم که دو کلاس Shirt و T-shirt دو کلاسی هستند که بیشتر با یکدیگر اشتباه گرفته می‌شوند.
- با افزایش پیچیدگی مدل با استفاده از تعداد بیشتر لایه‌های مخفی یا نورون‌ها، این امکان را به مدل می‌دهد که الگوهای پیچیده‌تر دیتا را تشخیص دهد. با این حال، با افزایش پیچیدگی مدل، احتمال overfit شدن بیشتر می‌شود. چون حساسیت مدل بر نویزها و اطلاعات بی‌اهمیت داده نیز بیشتر شده و آن ها را نیز به عنوان feature استخراج می‌کند. در قسمت‌های بعد، میبینیم که چگونه با افزایش پیچیدگی مدل، بهبود عملکرد را به همراه دارد.
- برای انتخاب بهترین پیکربندی، معیارهای متفاوتی از جمله تعداد لایه‌ها، تعداد نورون‌های هر لایه، انتخاب بهینه‌ساز مناسب، انتخاب نرخ آموزش مناسب، تابع هزینه مناسب و ... وجود دارد. در کل به چنین پارامترهایی، هایپرپارامتر گفته می‌شود. برای انتخاب هایپرپارامترهای مناسب، از الگوریتم‌های tuning مانند GridSearch، RandomSearch و ... استفاده می‌شود. همچنین، یک سری معیارهای دیگر برای بررسی کیفیت مدل خود وجود دارد. accuracy، precision، recall، f1 score مثال‌هایی از این دسته هستند.

۱-۲. آموزش دو مدل متفاوت

معماری مدل اول در شکل ۱.۴ آمده‌است. همچنین، معماری مدل دوم در شکل ۱.۵ نشان داده شده‌است.



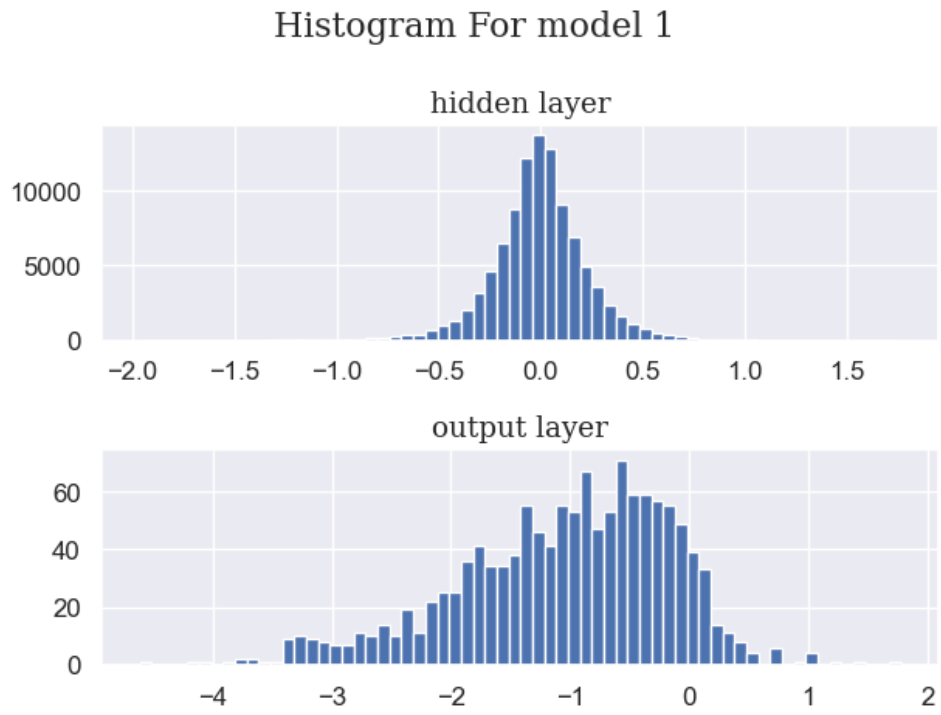
شکل ۱.۴. معماری مدل اول در بخش ۱-۲



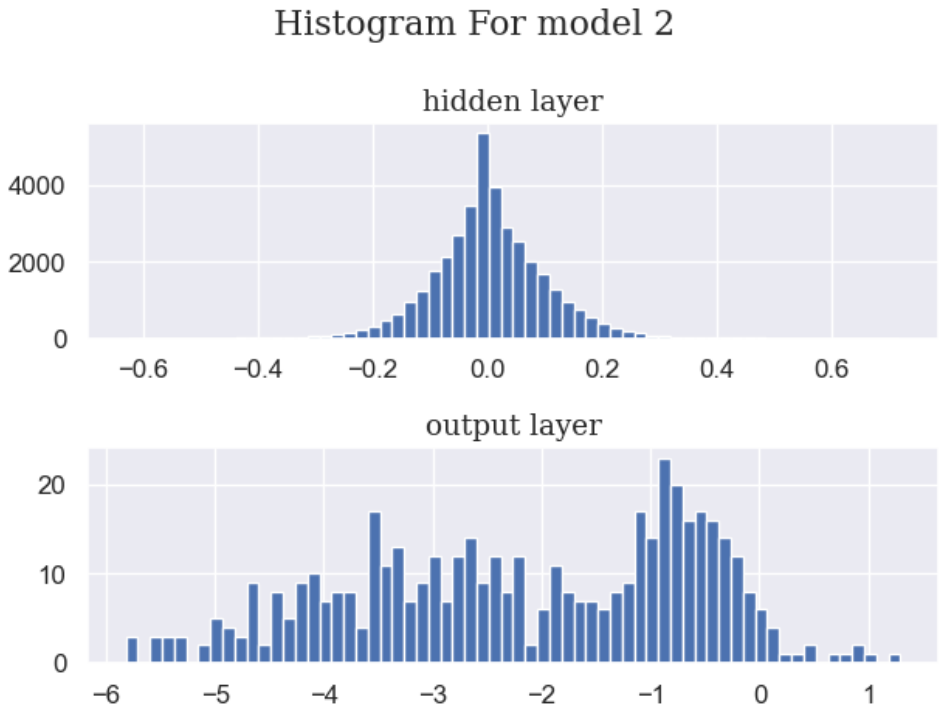
شکل ۱.۵. معماری مدل دوم در بخش ۱-۲

میزان دقت مدل اول بر روی داده‌های آموزش 0.9432 و بر روی داده‌های تست 0.8830 می‌باشد. این در حالی است که میزان دقت مدل اول بر روی داده‌های آموزش 0.8709 و بر روی داده‌های تست 0.8634 می‌باشد.

- هیستوگرام‌های مطلوب سوال برای مدل اول در شکل ۱.۶ و برای مدل دوم در شکل ۱.۷ آمده‌است.



شکل ۱.۶. هیستوگرام لایه مخفی و لایه انتهایی برای مدل اول بخش ۱-۲



شکل ۱.۷. هیستوگرام لایه مخفی و لایه انتهایی برای مدل دوم بخش ۱-۲

در مورد هیستوگرام‌ها میتوان گفت که در لایه انتهایی، توزیع وزن‌ها پخش‌تر می‌باشد و گوناگونی وزن‌های لایه آخر از وزن‌های لایه پنهان بیشتر می‌باشد. همچنین، در مقایسه هیستوگرام دو مدل می‌توان گفت که به دلیل وجود تمهیداتی برای جلوگیری از overfit شدن

مدل (وجود Dropout و L2-Regularizer)، وزن‌های لایه پنهان به یکسان شدن متمایل شده‌اند.

- به طور کلی می‌توان گفت که اضافه‌کردن روش‌های بهینه‌سازی پیشرفته عملکرد مدل را بهبود می‌بخشد.

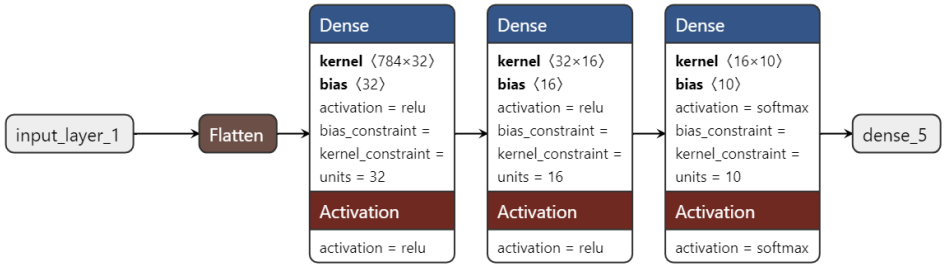
اگر مدل‌ها را توسط بهینه‌ساز Adam بهبود ببخشیم، پس از آموزش مدل، دقت مدل اول بر روی داده‌های آموزش برابر با 0.9689 و بر روی داده‌های تست برابر با 0.8828 می‌باشد. همچنین، دقت مدل دوم بر روی داده‌های آموزش برابر با 0.8826 و بر روی داده‌های تست برابر با 0.8732 می‌باشد.

اگر مدل‌ها را توسط بهینه‌ساز RMSprop بهبود ببخشیم، پس از آموزش مدل، دقت مدل اول بر روی داده‌های آموزش برابر با 0.9808 و بر روی داده‌های تست برابر با 0.8877 می‌باشد. همچنین، دقت مدل دوم بر روی داده‌های آموزش برابر با 0.8713 و بر روی داده‌های تست برابر با 0.8589 می‌باشد.

طبق نتایج بدست آمده، می‌توان گفت که توسط روش‌های بهینه‌سازی عملکرد مدل‌های ما بهبود یافته‌است.

۱-۳. الگوریتم بازگشت به عقب

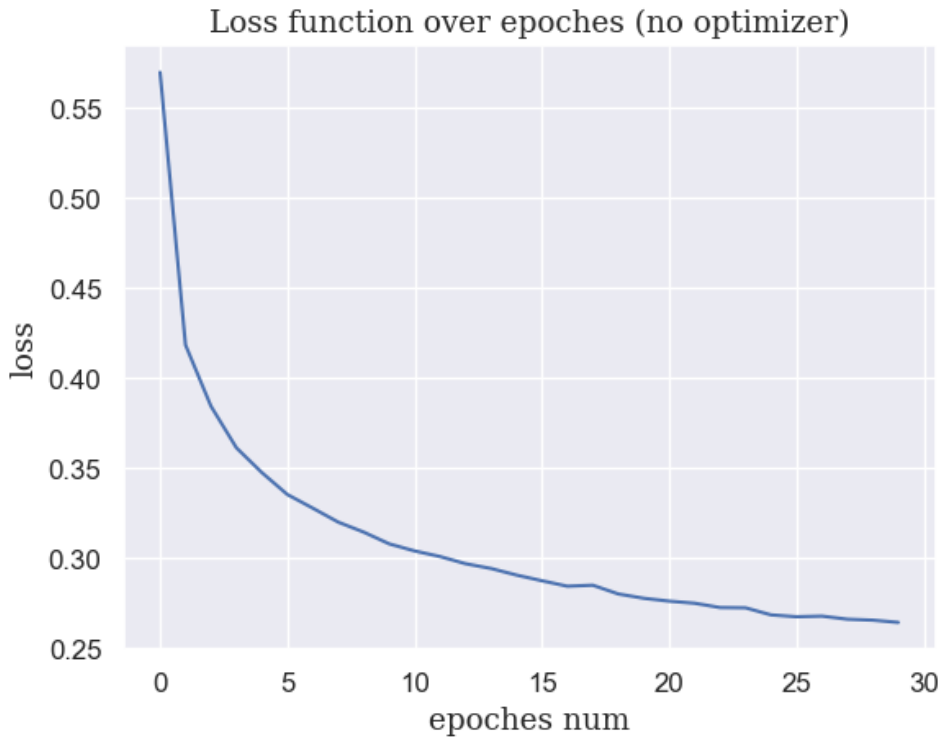
معماری این مدل در شکل ۱.۸ آمده‌است.



شکل۱.۸. معماری مدل بخش ۱-۳

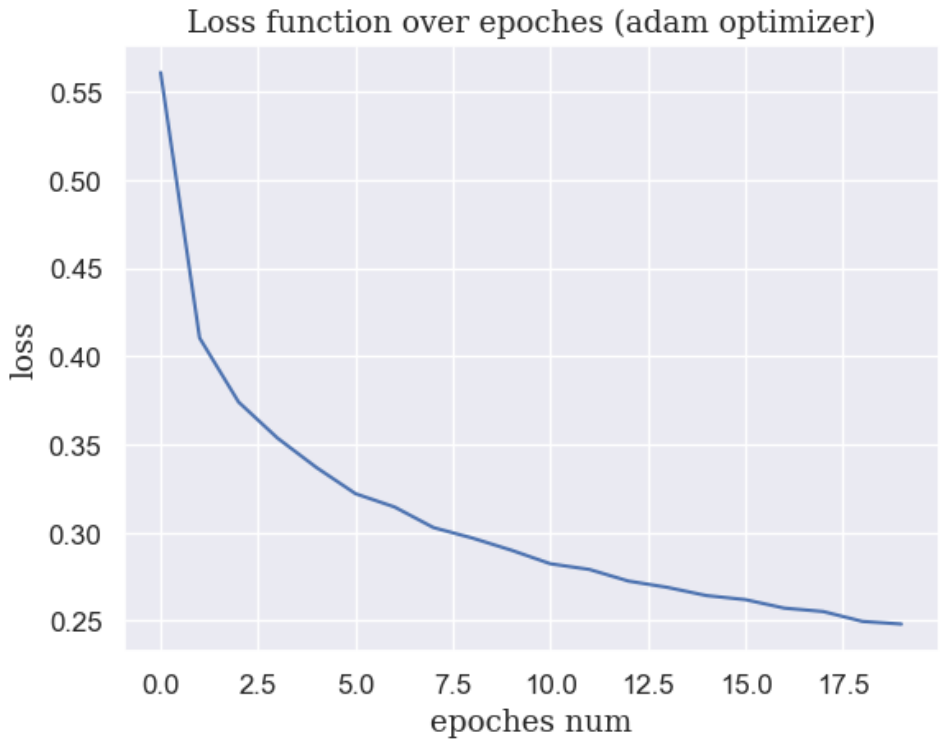
قبل از آموزش مدل، وزن‌های مدل را ذخیره می‌کنیم تا امکان بررسی آموزش‌های متفاوت این مدل را داشته‌باشیم.

ابتدا مدل را بدون هیچگونه بهینه‌ساز آموزش می‌دهیم. بعد از آموزش، دقت مدل بر روی داده‌های آموزش برابر با 0.9077 و بر روی داده‌های تست برابر با 0.8665 می‌باشد. همچنین، نمودار تابع هزینه بر حسب شماره ایپاک در شکل ۱.۹ آمده‌است.



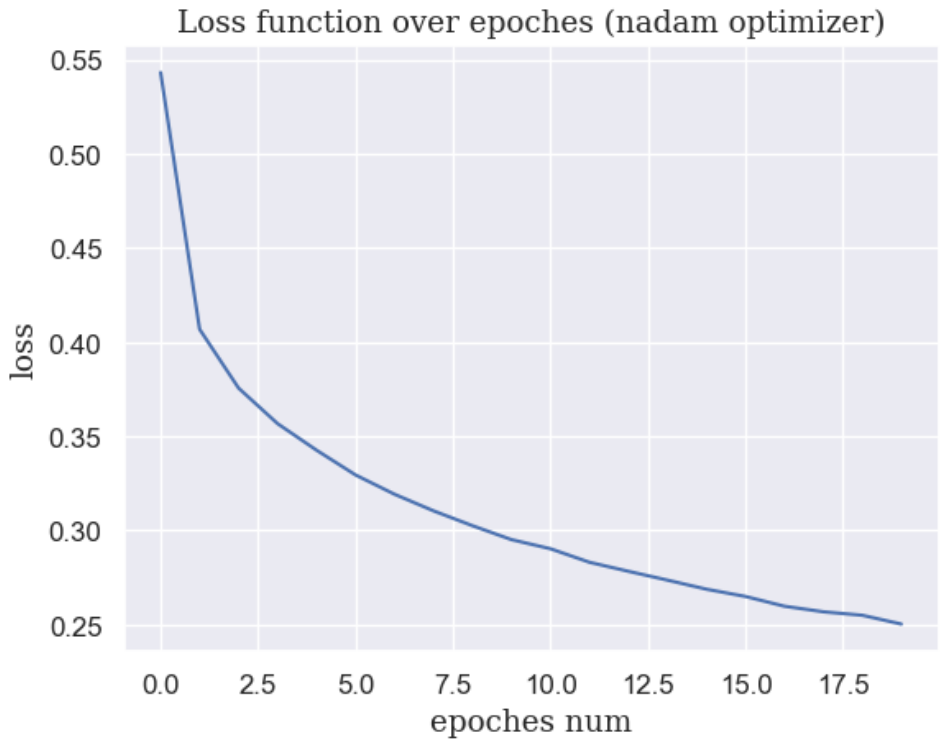
شکل۱.۹. نمودار تابع هزینه مدل بخش ۱-۳ (بدون بهینه‌ساز)

سپس مدل را با بهینه‌ساز Adam آموزش می‌دهیم. بعد از آموزش، دقت مدل بر روی داده‌های آموزش برابر با 0.9083 و بر روی داده‌های تست برابر با 0.8765 می‌باشد. همچنین، نمودار تابع هزینه بر حسب شماره ایپاک در شکل ۱.۱۰ آمده‌است.



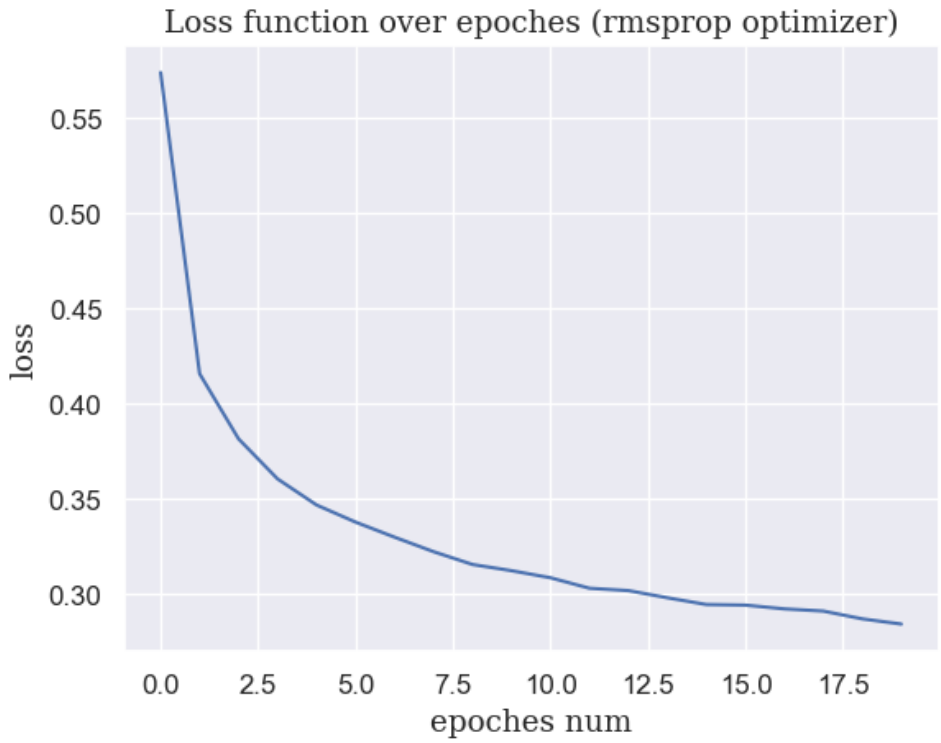
شکل۱.۱۰. نمودار تابع هزینه مدل بخش ۱-۳ (بهینه‌ساز Adam)

حال مدل را با بهینه‌ساز Nadam آموزش می‌دهیم. بعد از آموزش، دقت مدل بر روی داده‌های آموزش برابر با 0.9068 و بر روی داده‌های تست برابر با 0.8723 می‌باشد. همچنین، نمودار تابع هزینه بر حسب شماره ایپاک در شکل ۱.۱۱ آمده‌است.



شکل۱.۱۱. نمودار تابع هزینه مدل بخش ۱-۳(بهینه‌ساز Nadam)

در انتها مدل را با بهینه‌ساز RMSprop آموزش می‌دهیم. بعد از آموزش، دقت مدل بر روی داده‌های آموزش برابر با 0.8998 و بر روی داده‌های تست برابر با 0.8605 می‌باشد. همچنین، نمودار تابع هزینه بر حسب شماره ایپاک در شکل ۱.۱۲ آمده‌است.



شکل۱.۱۲. نمودار تابع هزینه مدل بخش ۱-۳ (بهینه‌ساز RMSprop)

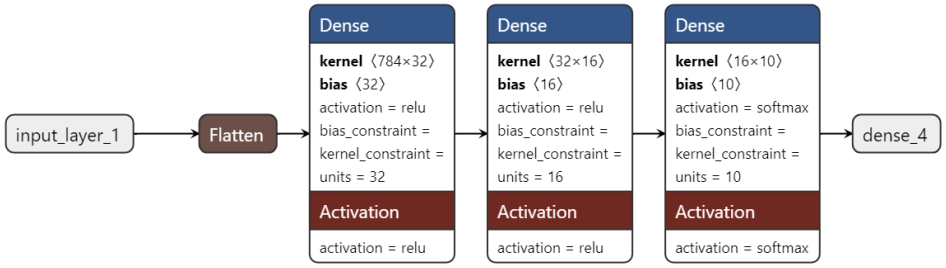
- با مقایسه نمودار تابع هزینه این 3 روش بهینه‌سازی و همچنین دقت کلی آن ها، متوجه می‌شویم که در مورد مقدار accuracy، بهترین بهینه‌ساز Adam و بدترین RMSprop می‌باشد. سرعت بهینه‌سازی این 3 روش بهینه‌سازی تقریبا یکسان می‌باشد. اما دو بهینه‌ساز Adam و Nadam نسبت به RMSprop به مقدار خطای کمتری میل می‌کنند.
- با اعمال الگوریتم جستجوی بیزی، به این نتیجه می‌رسیم که این الگوریتم به بهبود این فرآیند کمک شایانی می‌کند. توسط الگوریتم جستجوی بیزی، به دقت 0.9109 بر روی داده‌های آموزش می‌رسیم که از تمام نتایج بدست آمده در این بخش بهتر است.

۱-۴. بررسی هایپرپارامترهای مختلف

هایپرپارامترها در شبکه‌های عصبی، تنظیمات یا پارامترهایی هستند که قبل از شروع فرآیند آموزش مدل تعیین می‌شوند و برخلاف پارامترهای مدل، در طول فرآیند آموزش یاد گرفته نمی‌شوند. هایپرپارامترها بر ساختار شبکه و فرآیند یادگیری تاثیر زیادی دارند و انتخاب مناسب آن ها می‌تواند به افزایش دقت مدل و بهبود عملکرد آن منجر شود.

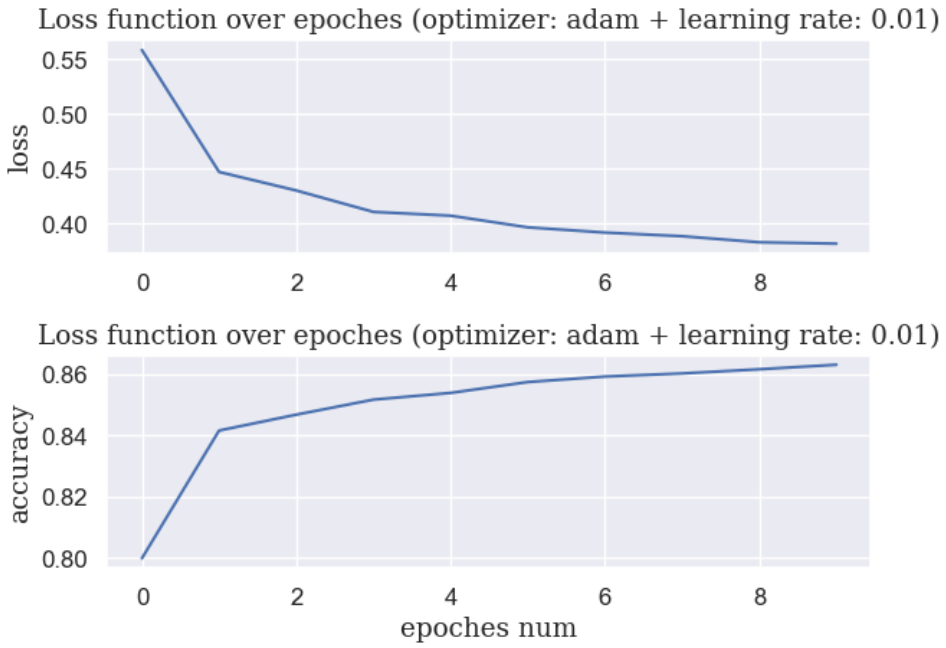
برای هایپرپارمترهای متفاوت سه مورد optimizer، learning_rate و loss_fumction انتخاب شده‌اند. در 9 حالت اول تنها دو هایپرپارامتر اول را تغییر می‌دهیم؛ اما در حالت آخر تعداد لایه‌ها را نیز تغییر داده و اثر آنرا بررسی می‌کنیم.

در شکل ۱.۱۳ معماری مدل چهارم برای 9 حالت ابتدایی نشان داده شده‌است. تمام 10 مدل با 10 ایپاک آموزش دیده‌اند.

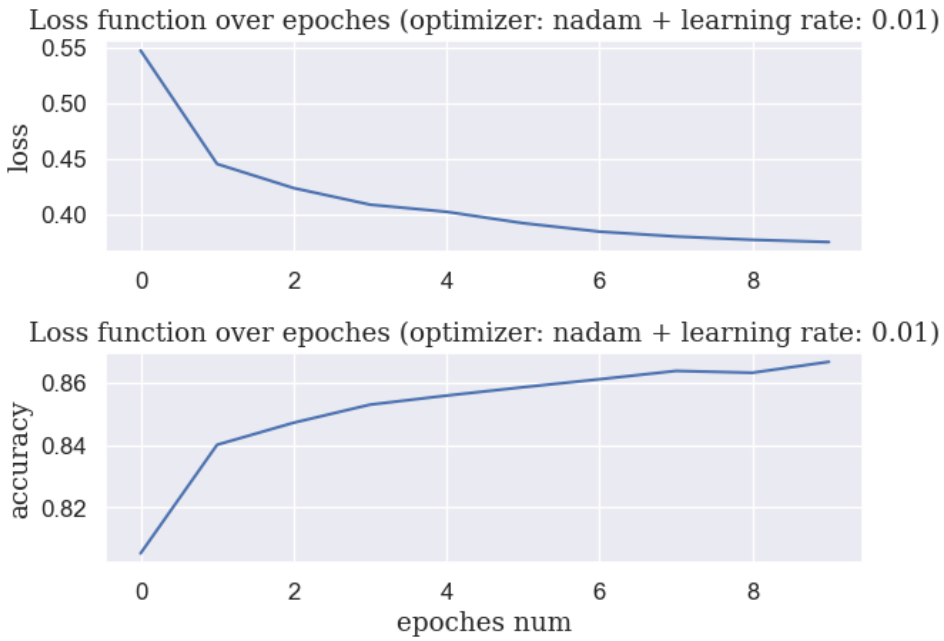


شکل۱.۱۳. معماری مدل اول بخش ۱-۴

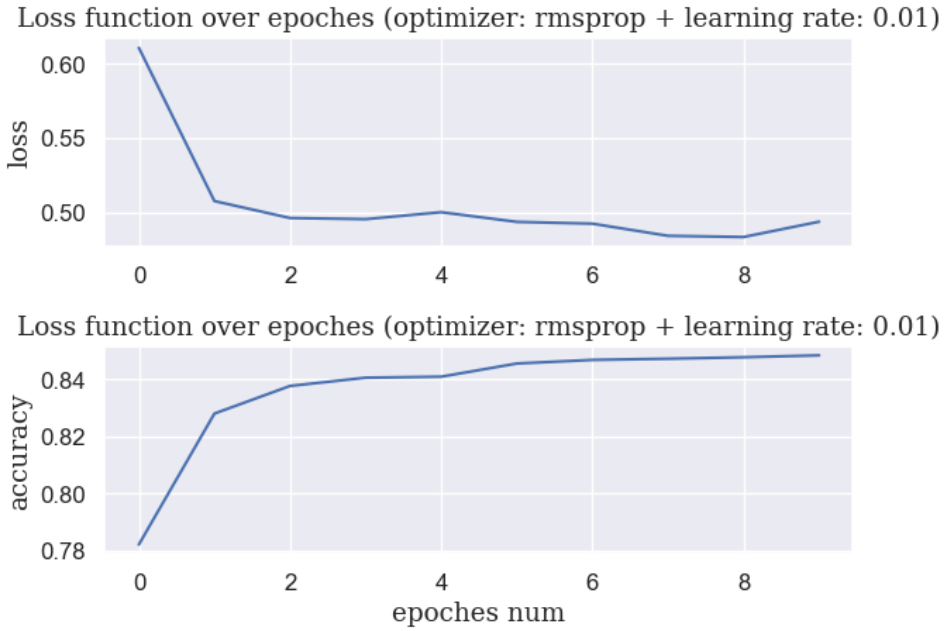
شکل ۱.۱۴ تا ۱.۲۲ نمودار دقت و تابع هزینه ۹ مدل ابتدایی را نشان می‌دهند.



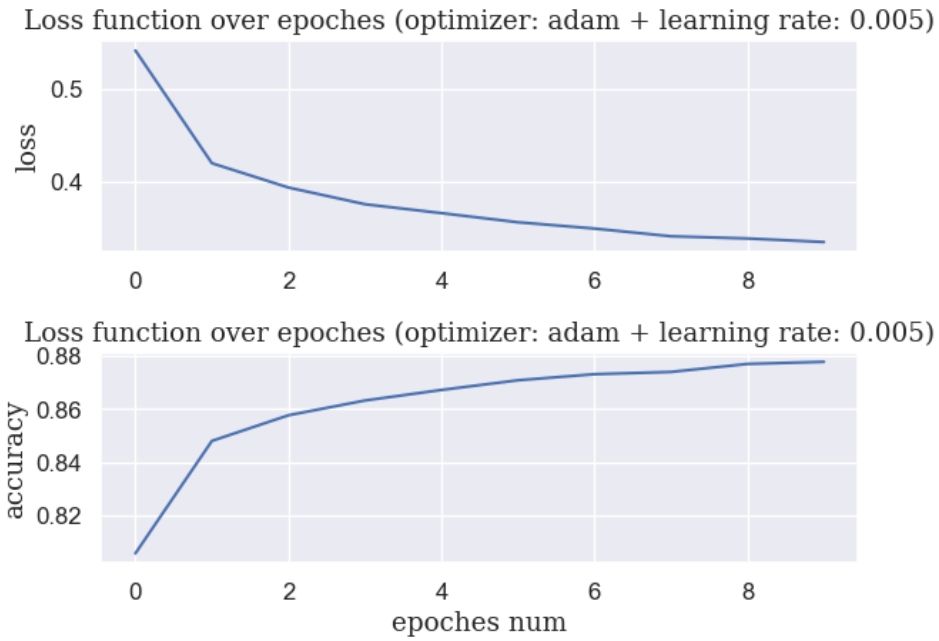
شکل۱.۱۴. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Adam و نرخ یادگیری 0.01



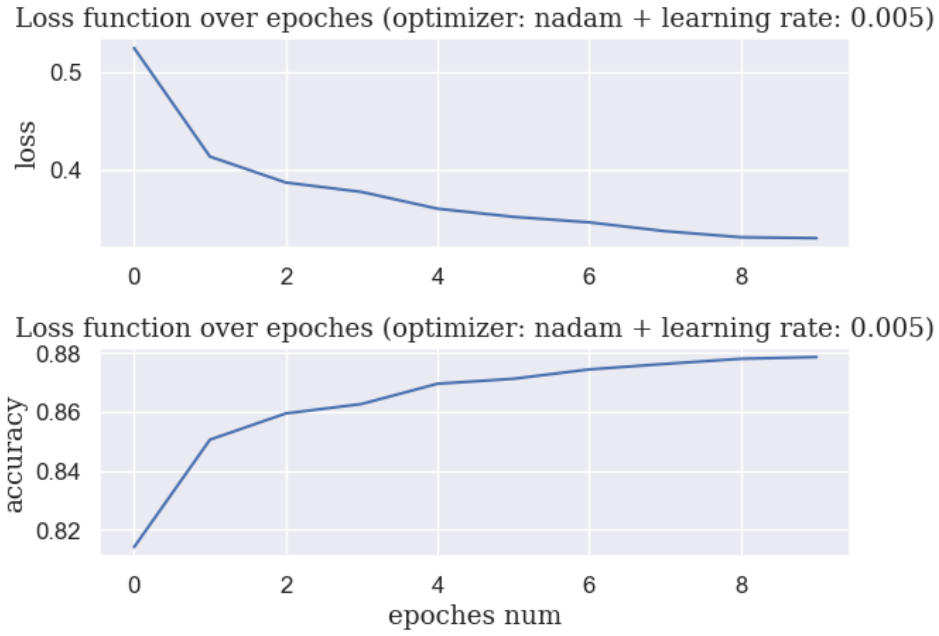
شکل۱.۱۵. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Nadam و نرخ یادگیری 0.01



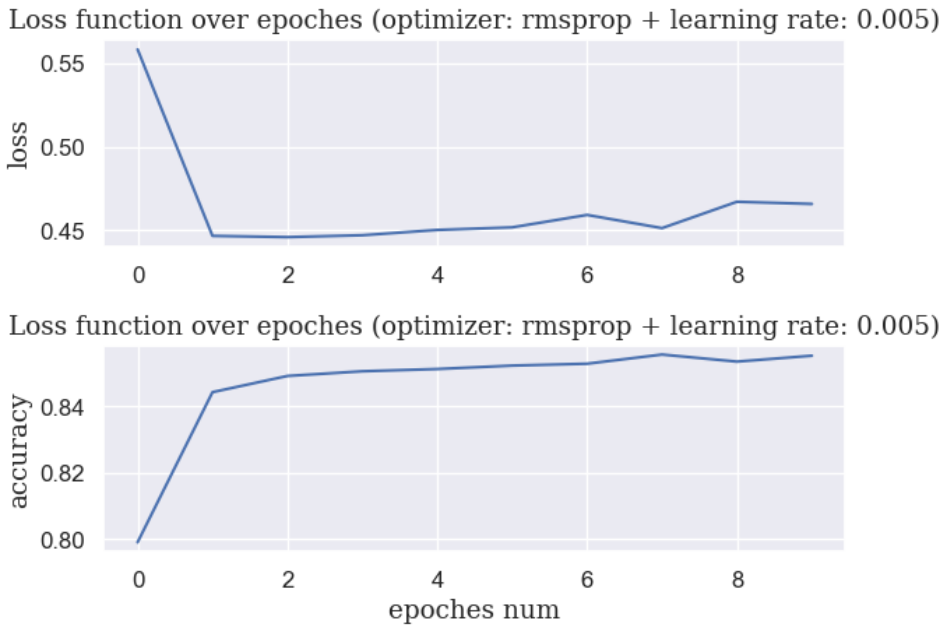
شکل۱.۱۶. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری 0.01



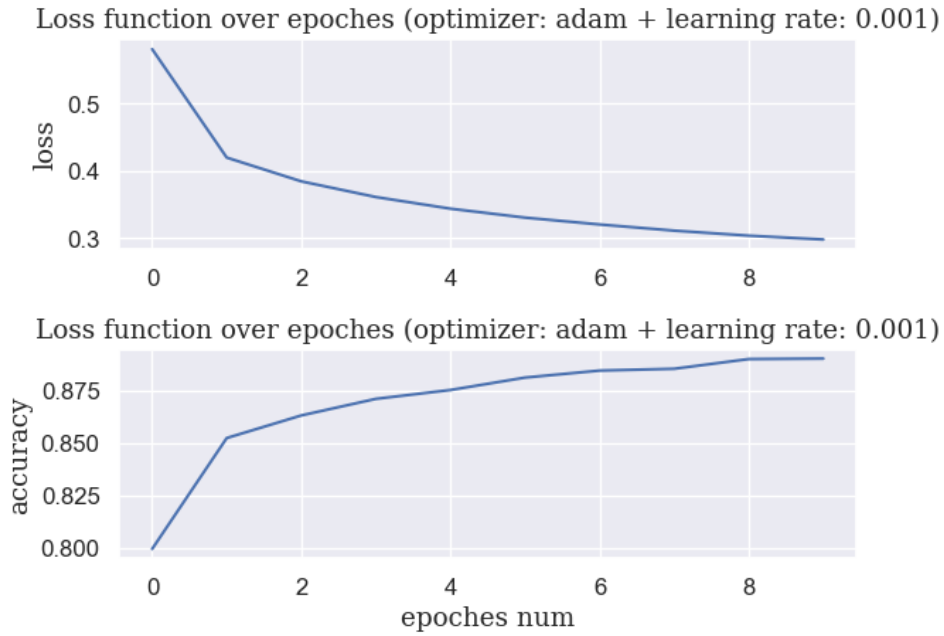
شکل ۱.۱۷. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Adam و نرخ یادگیری 0.005



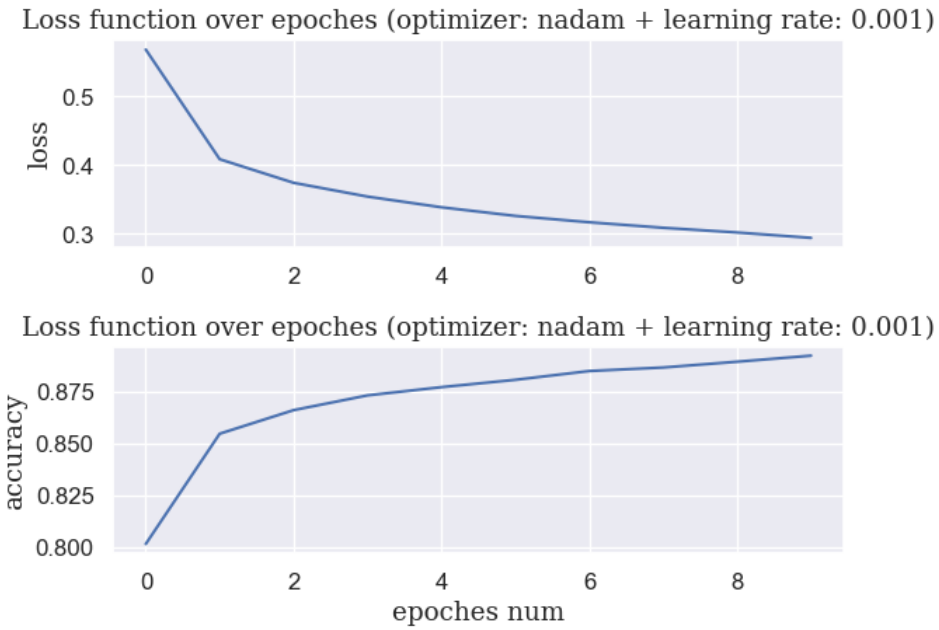
شکل ۱.۱۸. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Nadam و نرخ یادگیری 0.005



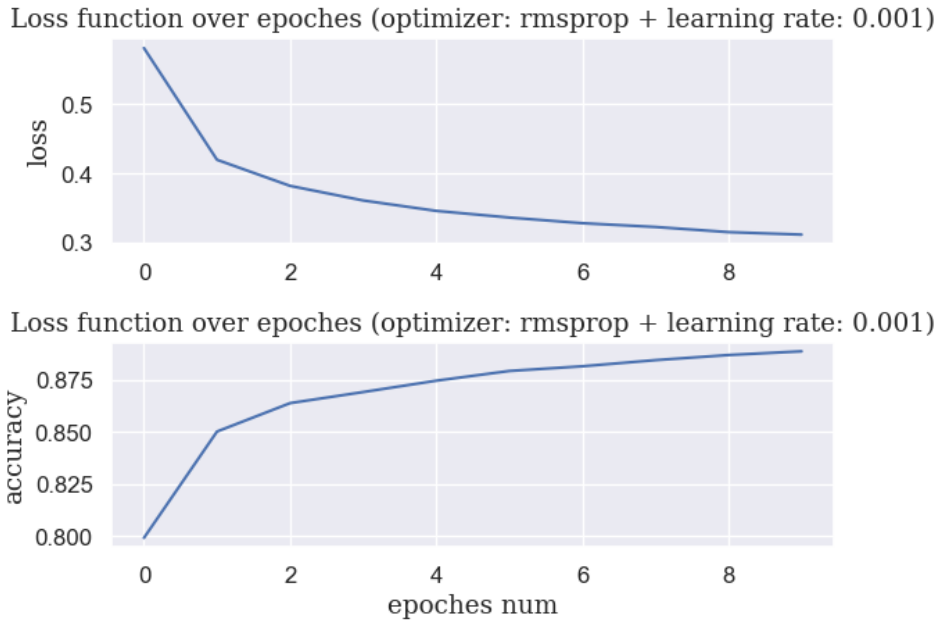
شکل۱.۱۹. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری 0.005



شکل۱.۲۰. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Adam و نرخ یادگیری 0.001

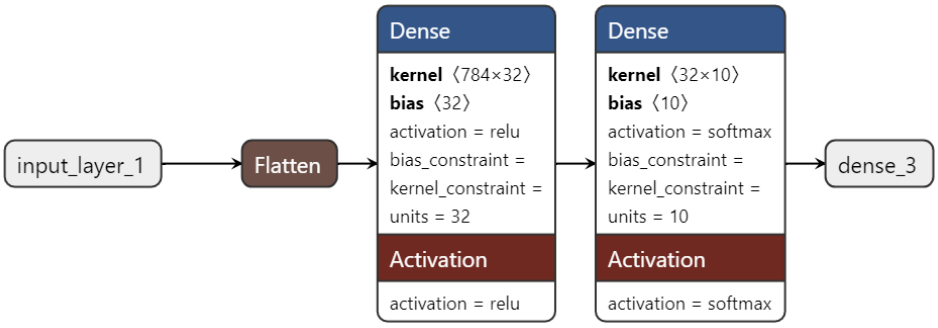


شکل۱.۲۱. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز Nadam و نرخ یادگیری 0.001



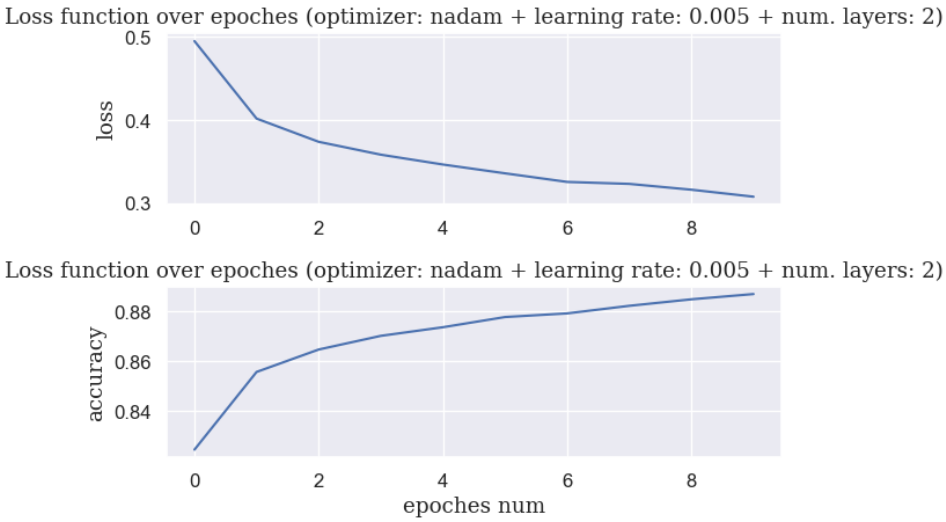
شکل۱.۲۲. نمودار مدل اول بخش ۱-۴ با بهینه‌ساز RMSprop و نرخ یادگیری 0.001

سپس، مدل را عوض می‌کنیم و با یک لایه پنهان مدل را آموزش می‌دهیم. معماری این مدل در شکل ۱.۲۳ آمده‌است.



شکل ۱.۲۳. معماری مدل دوم بخش ۱-۴

حال، این مدل را نیز آموزش می‌دهیم و نمودار دقت و تابع هزینه آن را رسم می‌کنیم. این نمودارها در شکل ۱.۲۴ آمده‌اند.



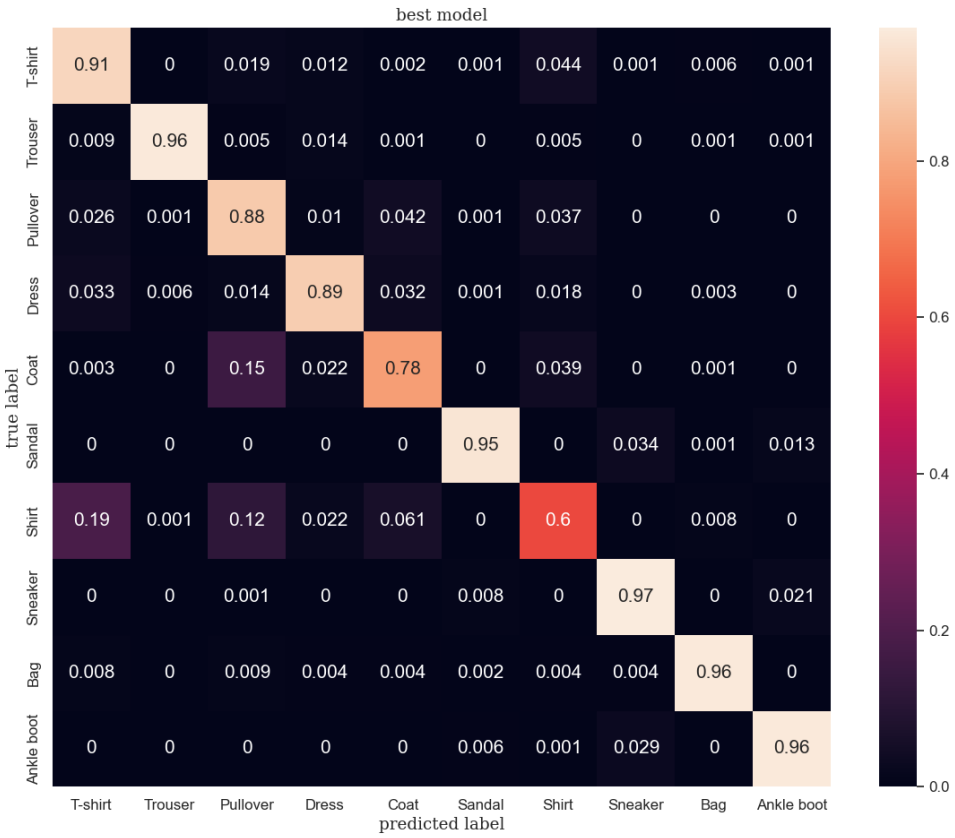
شکل ۱.۲۴. نمودار مدل دوم بخش ۱-۴ با بهینه ساز Nadam و نرخ یادگیری 0.005 و تعداد لایه 2

- برای این قسمت، از دو روش tuning استفاده شده‌است. این دو روش، Grid Search و Random Search می‌باشند. در روش Grid Search به همان ترتیبی که هایپرپارامترهای مورد نظر به الگوریتم داده‌شده، الگوریتم شروع به ساختن ترکیبات آن ها می‌کند و آن ها را جایگزین می‌کند و معیار هدف را مورد بررسی قرار می‌دهد. روش Random Search مشابه روش Grid Search می‌باشد با این تفاوت که این روش ترتیب خاصی به کار نمی‌گیرد و به صورت تصادفی ترکیبات را تشکیل می‌دهد. این دو روش در این بخش با 10 ترکیب از 3 هایپرپارامتر به کار گرفته شده‌اند تا تفاوت این دو روش و بهترین مدل بدست آید. 3 هایپرپارامتری که هدف هستند **تعداد نورون‌های لایه اول**، **تعداد نورون‌های لایه دوم** و **نرخ یادگیری** می‌باشند.

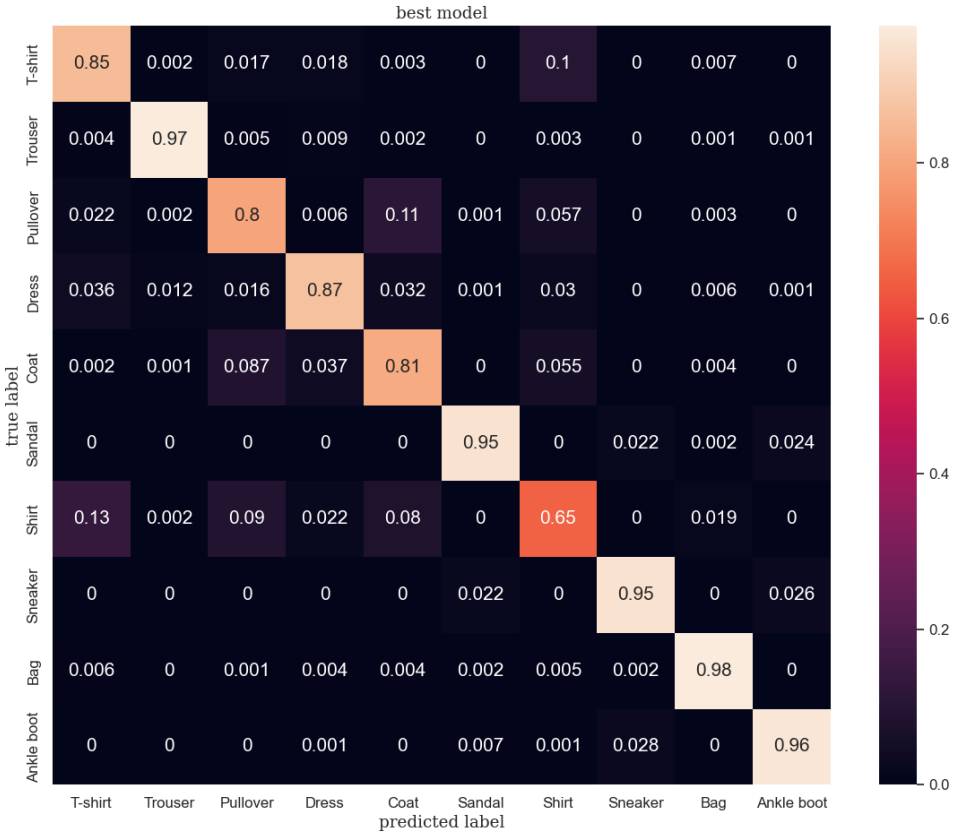
بهترین دقتی که توسط الگوریتم Grid Search حاصل می‌شود 0.9019 بوده‌است. این دقت با تعداد 64 نورون در لایه اول، تعداد 16 نورون در لایه دوم و نرخ یادگیری 0.001 حاصل شده‌است.

بهترین دقتی که توسط الگوریتم Random Search حاصل می‌شود 0.9104 بوده‌است. این دقت با تعداد 128 نورون در لایه اول، تعداد 64 نورون در لایه دوم و نرخ یادگیری 0.0005 حاصل شده‌است.

- دو ماتریس آشفتگی حاصل از Random Search و Grid Search به ترتیب در شکل‌های ۱.۲۵ و ۱.۲۶ آمده‌اند.



شکل۱.۲۵. ماتریس آشفتگی الگوریتم Random Search



شکل۱.۲۶. ماتریس آشفتگی الگوریتم Grid Search

بعد از بررسی دو ماتریس آشفتگی که یکی بهترین مدل بدست آمده از الگوریتم Random Search و دیگری بهترین مدل بدست آمده از الگوریتم Grid Search است، متوجه می‌شویم که هر هایپرپارامتر چه اثری بر روی ماتریس آشفتگی داشته‌است. جدول ۲ این بررسی را نشان می‌دهد.

جدول۲. بررسی اثر هایپرپارمترها بر روی ماتریس آشفتگی

هایپرپارامتر	اثر بر روی ماتریس آشفتگی
تعداد نورون‌های لایه اول و لایه دوم	با افزایش تعداد نورون‌های هر لایه، امکان یادگیری feature های پیچیده‌تر توسط مدل فراهم می‌گردد. با این حال، امکان حساس شدن مدل به feature های کم‌اهمیت و همچنین overfilt شدن مدل وجود دارد.

نرخ یادگیری

با افزایش نرخ یادگیری، سرعت همگرا شدن به مینیم افزایش می‌یابد. با این حال، افزایش نرخ یادگیری می‌تواند واگرا شدن یادگیری را به همراه داشته باشد و به طوری روی یادگیری اثر گذارد که ما دیگر به صورت پیوسته به مینیمم همگرا نشویم و در طول فرآیند گرادیان پرش‌های ناخواسته‌ای داشته‌باشیم.

پرسش ۲ – آموزش و ارزیابی یک شبکه عصبی ساده

۲-۱. آموزش شبکه عصبی

در این بخش، هدف تولید و آموزش یک شبکه عصبی بدون استفاده از کتابخانه های آماده (به جز numpy) میباشد.

۲-۱-۱. تابع فوروارد

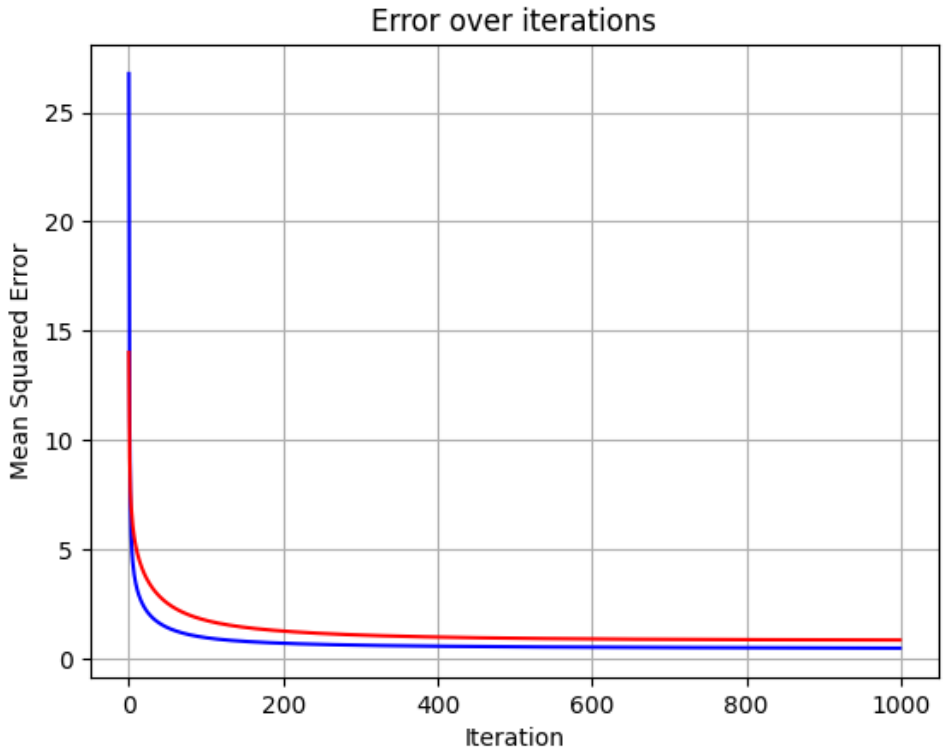
برای تولید شبکه عصبی خواسته شده در روی سوال، یک کلاس به اسم ANet طراحی شد. تابع فوروارد، یک تابع ساده با ورودی های x و وزن دو لایه است. کار این تابع تولید خروجی با توجه به وزن ها و ورودی شبکه عصبی است، بنابرین ورودی در وزن های ورودی تنها لایه این شبکه عصبی ضرب میشود، سپس برای اکتیویشن از تابع تانژانت هیپربولیک استفاده میشود تا خروجی Z به دست آید. در آخر، Z در وزن های خروجی لایه ضرب میشود و prediction شبکه عصبی به دست می آید.

۲-۱-۲. تابع بکوارد

برای این بخش، یک تابع بکوراد با ورودی های x, y ، تعداد نورون های لایه پنهان، تعداد iteration ها ، میزان ضریب یادگیری و دیتای تست طراحی شد. تابع بکوارد، ابتدا وزن های ورودی و خروجی لایه پنهان را با اعداد تصادفی مقداردهی میکند و سپس وارد بخش آموزش میشود. در بخش آموزش، از تابع فروارد که قبلا نوشته ایم استفاده میشود تا پیشبینی و Z به دست بیاید. سپس خطای شبکه را پیدا میکنیم و dw1 و dw2 با توجه به فرمول های الگوریتم backward در شبکه های عصبی تولید میشوند. از جایی که numpy مشتق تانژانت هیپربولیک را ارائه نمیدهد، این تابع به صورت دستی تعریف شد. پس از پیدا کردن مشتق ها، وزن ها با توجه به ضریب یادگیری اصلاح میشوند و این کار به تعداد iteration ها تکرار میشود. قابل توجه است که برای نشان دادن میزان پیشرفت شبکه عصبی، به جز خروجی های خواسته شده ، یک خروجی خطای تست نیز حساب می‌شود.

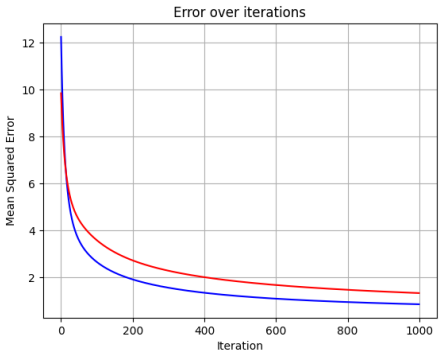
۲-۲. آموزش

قبل از آموزش، دیتای خواسته شده استاندارد شده و بایاس به آن اضافه میشود. سپس با استفاده از توابعی که قبلا نوشته شدند، عمل آموزش صورت میگیرد. نمودار خطا در این بخش با ضریب آموزش e-4 حساب شده و در شکل ۲.۱ قابل مشاهده است. همچنین خطای تست به صورت جداگانه محاسبه شده که به علت حجم زیاد، از بازنویسی آن در گزارش صرف نظر شد.

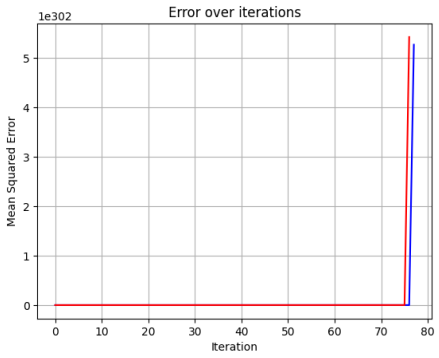


شکل ۲.۱. یادگیری با ضریب ۰.۰۰۰۱، خطای آموزش با آبی و خطای تست با قرمز نشان داده شده است

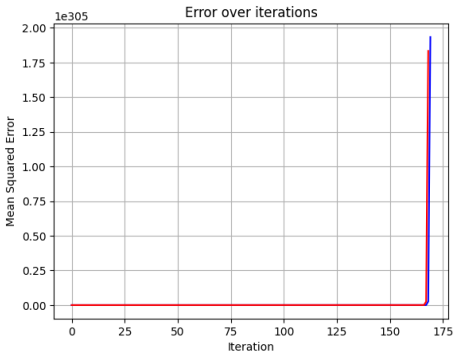
این آزمایش با ۳ ضریب مختلف ۰.۰۰۱ و ۰.۰۱ و ۰.۰۰۰۰۱ نیزتکرار شد که نتایج آن به ترتیب در تصاویر ۲.۲ و ۲.۳ و ۲.۴ قابل مشاهده است. از این مقایسه متوجه میشویم که ضرایب بالای یادگیری باعث میشوند که شبکه همگرا نشود و ضرایب پایین همگرایی را به تاخیر میاندازند و ضریب بهینه برای این شبکه با دیتاست استفاده شده، حدود ۰.۰۰۰۱ میباشد.



شکل ۲.۴. ضریب یادگیری ۰.۰۰۰۱



شکل۲.۳. ضریب یادگیری ۰.۰۱



شکل ۲.۴. ضریب یادگیری ۰.۰۰۱

پرسش ۳ – Madaline

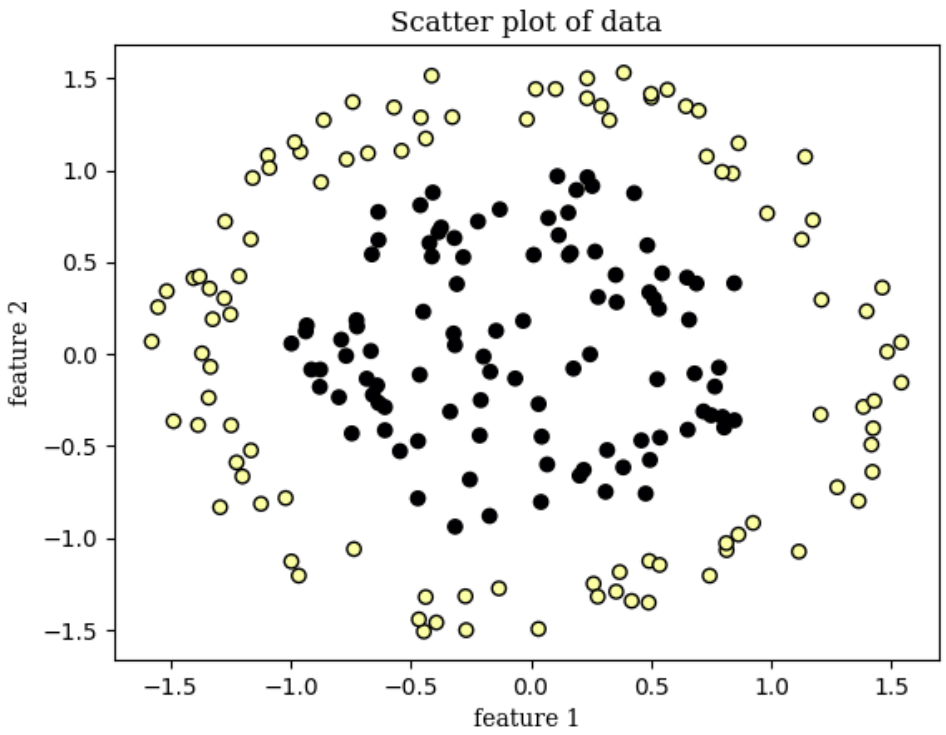
۳-۱. الگوریتم‌های MRI و MRII

الگوریتم قانون یادگیری Madaline 1 (Madaline Rule I) یک روش یادگیری نظارت شده برای شبکه‌های عصبی می‌باشد. این نوع شبکه معمولا از 3 لایه تشکیل شده‌است (یک لایه پنهان + لایه‌های ورودی و خروجی). این الگوریتم، به صورت Fully Connected می‌باشد. وزن‌ها و بایاس میان لایه پنهان و لایه آخر به صورت یک عدد ثابت بوده و آموزش دیده نمی‌شود. در کل لایه آخر MRI مشابه یک XOR عمل می‌کند. در یادگیری این روش، بسته به 1- یا 1+ بودن تارگت، رویکردهای متفاوتی به کار برده می‌شود.

۳-۲. نمودار پراکندگی داده‌ها

ابتدا توسط دستور read_csv() فایل مورد نظر را می‌خوانیم. به دلیل این که فایل ما header ندارد، آرگومان header=None را قرار می‌دهیم.

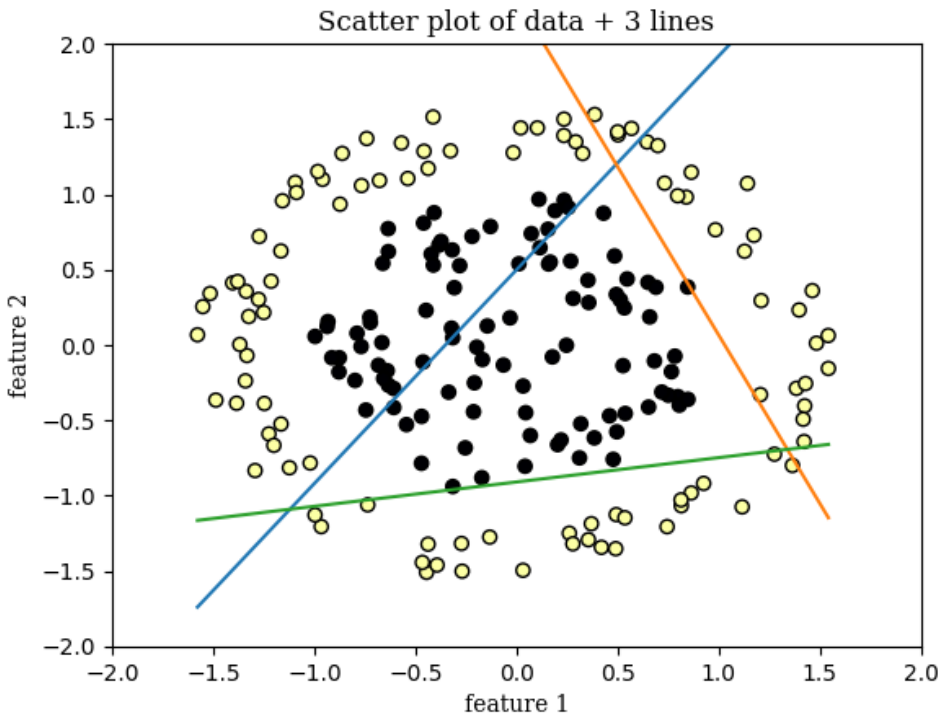
سپس داده دریافتی را به یک آرایه numpy تبدیل می‌کنیم تا کار با آن راحت‌تر شود. حال، scatter plot مورد نظر را رسم می‌کنیم. این نمودار در شکل ۳.۱ آمده است.



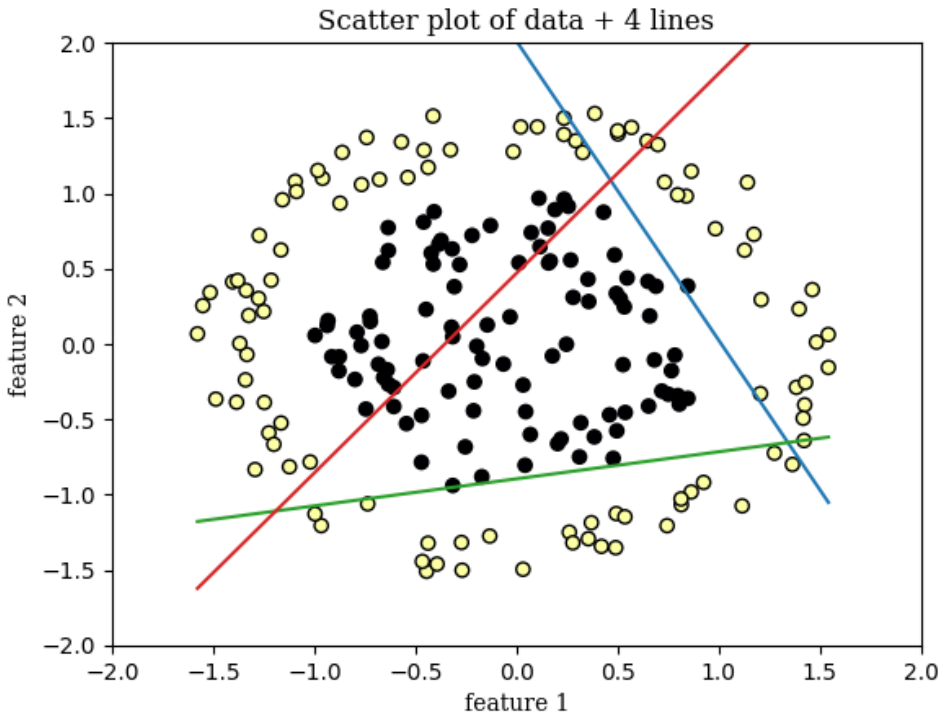
شکل۳.۱. نمودار پراکندگی داده‌ها

۳-۳. آموزش مدل

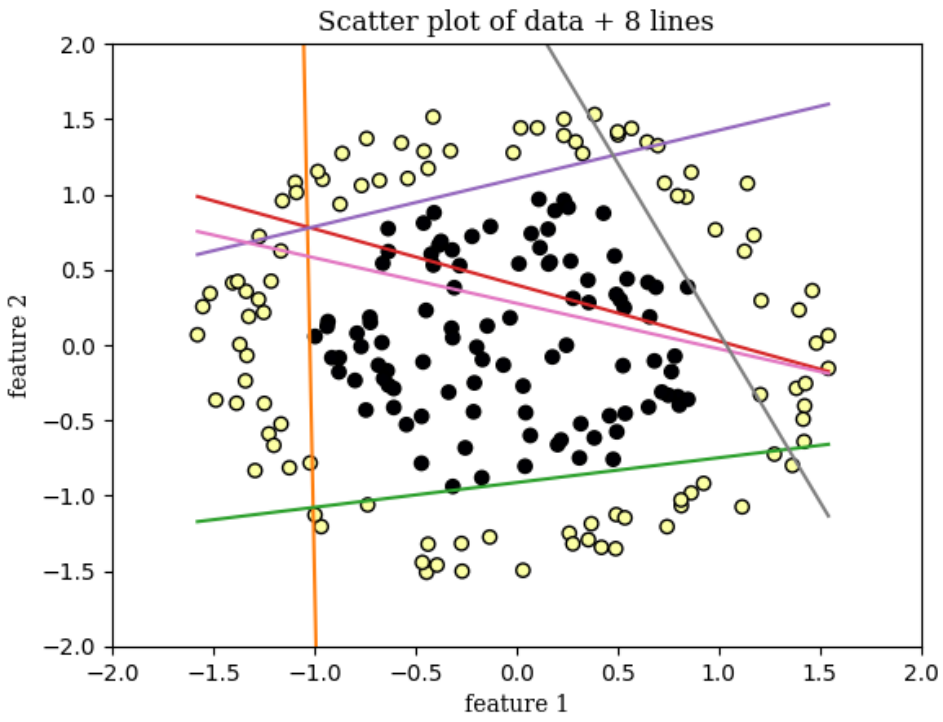
- بعد از آموزش مدل‌ها، مقدار خطای مدل با ۳ نورون برابر با 88، خطای مدل با ۴ نورون برابر با 82 و خطای مدل با 8 نورون برابر با 0 می‌باشد.
- شکل‌های ۳.۲، ۳.۳ و ۳.۴ به ترتیب نمودار پراکندگی مدل‌هایی با 3 نورون، 4 نورون و 8 نورون را نمایش داده‌اند.



شکل ۳.۲. نمودار پراکندگی مدل با 3 نورون



شکل ۳.۳. نمودار پراکندگی مدل با 4 نورون



شکل ۳.۴. نمودار پراکندگی مدل با 8 نورون

- همان طور که مشاهده می‌شود، می‌توان به صورت شهودی گفت که با افزایش نورون‌ها، تعداد خط‌های ما بیشتر می‌شود؛ خط‌های بیشتر این امکان را به ما می‌دهند که جداسازی راحت‌تر و بهتر صورت گیرد.

اگر خودمان به نمودار پراکندگی داده‌ها نگاه کنیم و تلاش کنیم که خودمان با خط‌هایی مستقیم اقدام به جداسازی کنیم، متوجه می‌شویم که امکان جداسازی داده‌ها با 3 خط امکان‌پذیر نمی‌باشد.

به نظر می‌رسد که با 4 خط مستقیم این امکان وجود دارد اما به دلیل ضعف الگوریتم و عدم توانایی آن در جداسازی، در این الگوریتم با 4 خط مستقیم جداسازی به طور صحیح انجام نشده‌است.

اما مشاهده می‌شود که با 8 خط مستقیم این جداسازی به طور احسن صورت گرفته و کلاس‌های ما از یگدیگر تمایز یافته‌اند.

در مورد تعداد ایپاک باید گفت که تمام این مدل‌ها به تعداد 1000 ایپاک آموزش دیده‌اند. اما خطای مدل با تعداد 8 نورون بعد از 300 ایپاک به مقدار صفر رسیده‌است.

پرسش ۴ – MLP

۴-۱. تعداد NaN

فایل خوانده شد و تعداد Nan ها نمایش داده شد. در فایل داده شده مقدار Nan وجود ندارد.

۴-۲. ماتریس همبستگی

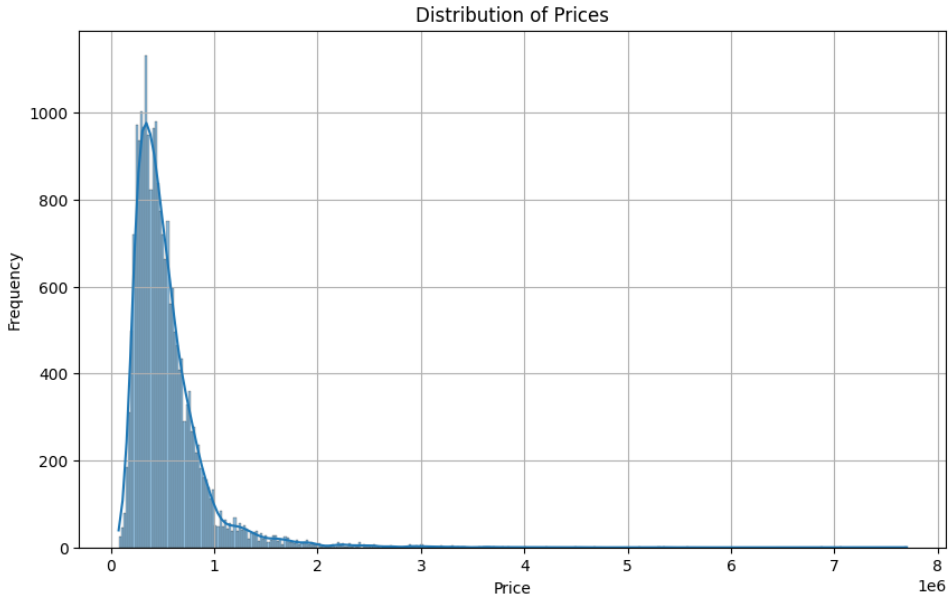
ماتریکس همبستگی توسط کتابخانه pandas در شکل ۴.۱ نشان داده شده است.

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
id	1.000000	-0.016762	0.001286	0.005160	-0.012258	-0.132109	0.018525	-0.002721	0.011592	-0.023783	0.008130	-0.010842	-0.005151	0.021380	-0.016907	-0.008224	-0.001891	0.020799	-0.002901	-0.138798
price	-0.016762	1.000000	0.308350	0.525138	0.702035	0.089661	0.256794	0.266369	0.397293	0.036362	0.667434	0.605567	0.323816	0.054012	0.126434	-0.053203	0.307003	0.021626	0.585379	0.082447
bedrooms	0.001286	0.308350	1.000000	0.515884	0.576671	0.031703	0.175429	-0.006582	0.079532	0.028472	0.356967	0.477600	0.303093	0.154178	0.018841	-0.152668	-0.008931	0.129473	0.391638	0.029244
bathrooms	0.005160	0.525138	0.515884	1.000000	0.754665	0.087740	0.500653	0.063744	0.187737	-0.124982	0.664983	0.685342	0.283770	0.506019	0.050739	-0.203866	0.024573	0.223042	0.568634	0.087175
sqft_living	-0.012258	0.702035	0.576671	0.754665	1.000000	0.172826	0.353949	0.103818	0.284611	-0.058753	0.762704	0.876597	0.435043	0.318049	0.055363	-0.199430	0.052529	0.240223	0.756420	0.183286
sqft_lot	-0.132109	0.089661	0.031703	0.087740	0.172826	1.000000	-0.005201	0.021604	0.074710	-0.008958	0.113621	0.183512	0.015286	0.053080	0.007644	-0.129574	-0.085683	0.229521	0.144608	0.718557
floors	0.018525	0.256794	0.175429	0.500653	0.353949	-0.005201	1.000000	0.023698	0.029444	-0.263768	0.458183	0.523885	-0.245705	0.489319	0.006338	-0.059121	0.049614	0.125419	0.279885	-0.011269
waterfront	-0.002721	0.266369	-0.006582	0.063744	0.103818	0.021604	0.023698	1.000000	0.401857	0.016653	0.082775	0.072075	0.080588	-0.026161	0.092885	0.030285	-0.014274	-0.041910	0.086463	0.030703
view	0.011592	0.397293	0.079532	0.187737	0.284611	0.074710	0.029444	0.401857	1.000000	0.045990	0.251321	0.167649	0.276947	-0.053440	0.103917	0.084827	0.006157	-0.078400	0.280439	0.072575
condition	-0.023783	0.036362	0.028472	-0.124982	-0.058753	-0.008958	-0.263768	0.016653	0.045990	1.000000	-0.144674	-0.158214	0.174105	-0.361417	-0.060618	0.003026	-0.014941	-0.106500	-0.092824	-0.003406
grade	0.008130	0.667434	0.356967	0.664983	0.762704	0.113621	0.458183	0.082775	0.251321	-0.144674	1.000000	0.755923	0.168392	0.446963	0.014414	-0.184862	0.114084	0.198372	0.713202	0.119248
sqft_above	-0.010842	0.605567	0.477600	0.685342	0.876597	0.183512	0.523885	0.072075	0.167649	-0.158214	0.755923	1.000000	-0.051943	0.423898	0.023285	-0.261190	-0.000816	0.343803	0.731870	0.194050
sqft_basement	-0.005151	0.323816	0.303093	0.283770	0.435043	0.015286	-0.245705	0.080588	0.276947	0.174105	0.168392	-0.051943	1.000000	-0.133124	0.071323	0.074845	0.110538	-0.144765	0.200355	0.017276
yr_built	0.021380	0.054012	0.154178	0.506019	0.318049	0.053080	0.489319	-0.026161	-0.053440	-0.361417	0.446963	0.423898	-0.133124	1.000000	-0.224874	-0.346869	-0.148122	0.409356	0.326229	0.070958
yr_renovated	-0.016907	0.126434	0.018841	0.050739	0.055363	0.007644	0.006338	0.092885	0.103917	-0.060618	0.014414	0.023285	0.071323	-0.224874	1.000000	0.064357	0.029398	-0.068372	-0.002673	0.007854
zipcode	-0.008224	-0.053203	-0.152668	-0.203866	-0.199430	-0.129574	-0.059121	0.030285	0.084827	0.003026	-0.184862	-0.261190	0.074845	-0.346869	0.064357	1.000000	0.267048	-0.564072	-0.279033	-0.147221
lat	-0.001891	0.307003	-0.008931	0.024573	0.052529	-0.085683	0.049614	-0.014274	0.006157	-0.014941	0.114084	-0.000816	0.110538	-0.148122	0.029398	0.267048	1.000000	-0.135512	0.048858	-0.086419
long	0.020799	0.021626	0.129473	0.223042	0.240223	0.229521	0.125419	-0.041910	-0.078400	-0.106500	0.198372	0.343803	-0.144765	0.409356	-0.068372	-0.564072	-0.135512	1.000000	0.334605	0.254451
sqft_living15	-0.002901	0.585379	0.391638	0.568634	0.756420	0.144608	0.279885	0.086463	0.280439	-0.092824	0.713202	0.731870	0.200355	0.326229	-0.002673	-0.279033	0.048858	0.334605	1.000000	0.183192
sqft_lot15	-0.138798	0.082447	0.029244	0.087175	0.183286	0.718557	-0.011269	0.030703	0.072575	-0.003406	0.119248	0.194050	0.017276	0.070958	0.007854	-0.147221	-0.086419	0.254451	0.183192	1.000000

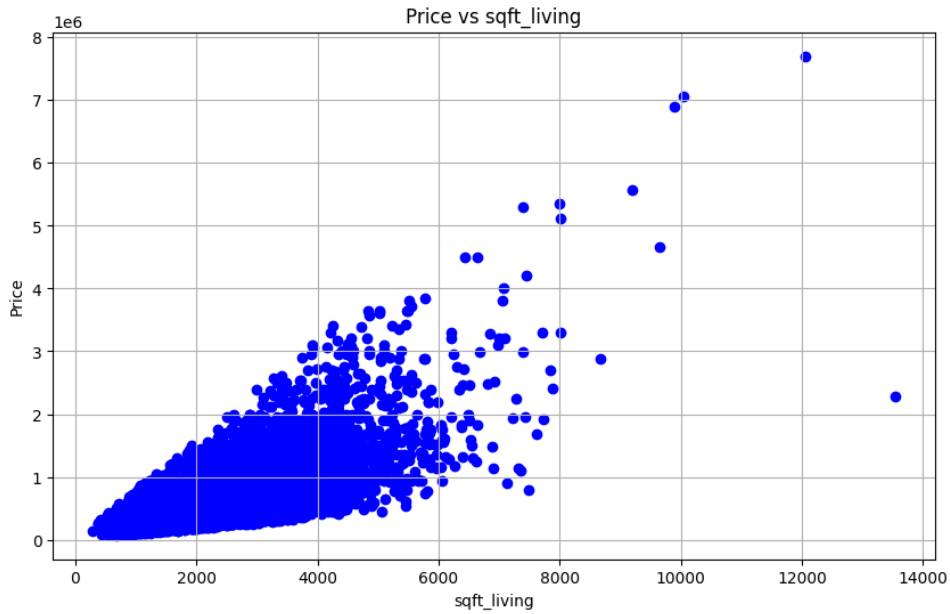
شکل ۴.۱. ماتریکس همبستگی دیتاست سوال ۴

۴-۳. رسم نمودار

نمودار توزیع قیمت در شکل ۴.۲ نشان داده شده است. طبق نتایج قسمت ۴-۱، بیشترین همبستگی قیمت با sqft_living بود که در شکل ۴.۳ نمودار پراکندگی آن نسبت به قیمت قابل مشاهده است.



شکل ۴.۲. نمودار توزیع قیمت



شکل ۴.۳. نمودار پراکندگی قیمت بر حسب sqft_living

۴-۴. پیش پردازش

در این بخش، ستون تاریخ در داده ها به دو بخش سال و ماه تقسیم شد و ستون قبلی از دیتاست drop شد. سپس، توسط توابع کتابخانه ی sklearn، داده ها با دو بخش تست و آموزش تقسیم شدند. در نهایت

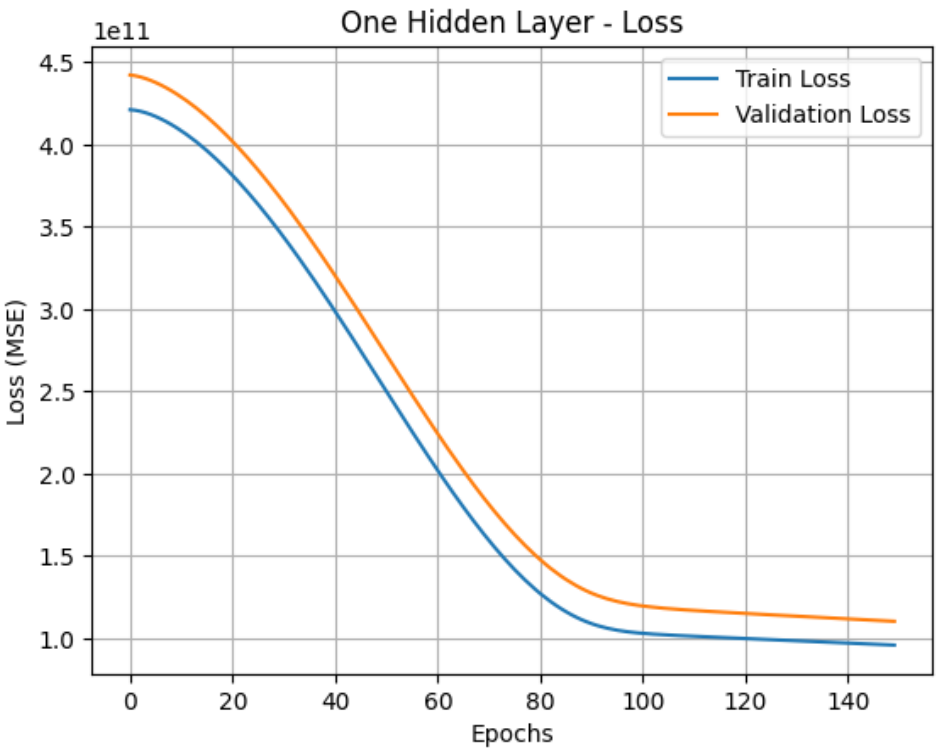
داده های بدست آمده توسط تابع MinMaxScaler همان کتابخانه به طور جداگانه اسکیل شدند. این تابع داده های موجود در یک دیتاست را به صورت جداگانه و به طور خطی به بازه ی دلخواه اسکیل میکند.

۴-۵. پیاده سازی مدل

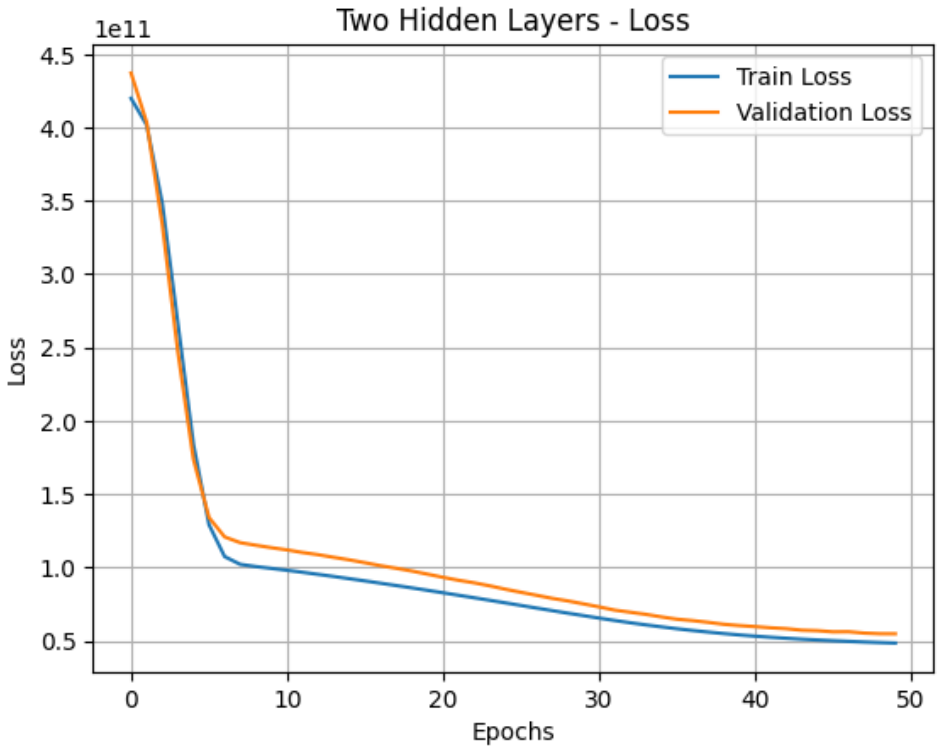
برای پیاده سازی مدل های خواسته شده، از کتابخانه ی Tensorflow استفاده شده است. برای اکتیویشن هر دو مدل از تابع ReLU استفاده شده و تعداد نورون های دو شبکه عصبی یکسان است، به طوری که برای مدل اول از یک لایه پنهان با ۶۰ نورون و برای مدل دوم از دو لایه با ۳۰ نورون استفاده شده است.

۴-۶. آموزش

هر دو مدل با دیتاست یکسان ترین شدند. نتایج بدست آمده از مدل های تک لایه و دو لایه به ترتیب در اشکال ۴.۴ و ۴.۵ مشاهده میشود. آموزش با استفاده از توابع آماده صورت گرفته و خطای Train و Validation با استفاده از تابع مربع خطاها (MSE) اندازه گرفته شده‌اند.



شکل ۴.۴. نمودارخطا برای مدل با یک لایه پنهان



شکل ۴.۵. نمودارخطا برای مدل با دو لایه پنهان

۴-۷. تحلیل نتایج

با مقایسه ی این دو مدل در می یابیم که مدلی که در آن از دو لایه پنهان استفاده شده، به طور قابل توجهی سریعتر به همگرایی رسیده است. مدل اول بعد از حدود ۱۲۰ ایتریشن به همگرایی رسیده و مدل دوم تنها ۵۰ ایپاک بعد از آموزش به همگرایی رسیده و تابع خطای آن نتیجه ی بهتری را گزارش میکند. این تفاوت ناشی از آن است که افزایش لایه ها باعث میشود شبکه عصبی بتواند محاسبات پیچیده تری برای مدل سازی از مسئله داشته باشد و به غیرخطی بودن نتیجه ی آموزش می افزاید. همچنین افزایش لایه ها در صورت ثابت ماندن تعداد نورون ها، زمان مورد نیاز برای آموزش را کاهش میدهد، به همین علت است که با اینکه یک لایه پنهان از لحاظ تئوری قادر به مدل کردن تمام توابع است، همچنان از شبکه های عصبی عمیق استفاده میکنیم.

برای پیش‌بینی قیمت از شبکه عصبی دو لایه استفاده شد و توسط کتابخانه‌ی numpy، 5 عدد را به طور رندم از دیتاست Validation انتخاب کردیم. خطای میانگین این داده ها به مقدار 0.0293 محاسبه شد که البته با توجه به رندم بودن و تعداد محدود مقادیر انتخاب شده، هر بار ممکن است نتایج متفاوتی به دست بیاید.