پاسخ تمرین سری ۱

محمدرضا عزیزی
۹۸۱۳۱۰۲۲
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)
mrazizi@aut.ac.ir

۱ راهکاری برای پیش‌پردازش داده‌ها

داده‌ها با استفاده از کتابخانه pandas بارگذاری کرده و از آن‌جایی که ستون‌های داده‌ها نام ندارد، ابتدا با توجه به توضیحات دیتاست، یک لیست برای نام ستون‌ها تعریف کرده و این نام‌ها را به ستون‌های دیتافریم بارگذاری شده اضافه می‌کنیم. لیست این نام‌ها به ترتیب برابر است با:

```
["age", "workclass", "fnlwgt", "education", "education-num",
"marital-status", "occupation", "relationship", "race", "sex",
"capital-gain", "capital-loss", "hours-per-week", "native-country", "label"]
```

ستون آخر که مربوط به برچسب داده‌است با با نام label نام‌گذاری کرده‌ایم.

راه حلی که در ابتدا ممکن است به ذهن برسد این است که یک دیکشنری تعریف کنیم که هر دوتایی کلید/مقدار آن، خود یک دیکشنری است. برای مثال به ازای کلید label، دیکشنری زیر را داریم:

```
'label': {'<=50K':0, '>50K': 1}
```

به ازای تمامی مقادیر تمامی ستون‌هایی که مقدار عددی ندارند، این دیکشنری را تعریف کرده و به هر مقدار اسمی، یک عدد نسبت دهیم. این اعداد از برای هر متغیر از ۰ شروع شده و یک واحد یک واحد افزایش می‌یابد.

در نهایت با استفاده از تابع replace از کتابخانه pandas، طبق دیکشنری تعریف شده، مقادیر اسمی را به مقادیر عددی تبدیل کنیم.

در ابتدا روش ذکر شده را پیاده‌سازی کردیم و بارگذاری داده به صورت زیر بود:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	label
0	39	5	77516	0	13	2	8	3	0	1	2174	0	40	0	0
1	50	1	83311	0	13	0	4	2	0	1	0	0	13	0	0
2	38	0	215646	3	9	1	6	3	0	1	0	0	40	0	0
3	53	0	234721	2	7	0	6	2	4	1	0	0	40	0	0
4	28	0	338409	0	13	0	5	0	4	0	0	0	40	12	0

اما این روش کدگذاری داده‌های اسمی یک مشکل بزرگ دارد. برای مثال به کدگذاری متغیر relationship دقت کنید:

```
'relationship': {'Wife':0, 'Own-child':1, 'Husband':2, 'Not-in-family':3,
                 'Other-relative':4, 'Unmarried':5},
```

در این حالت به یک فرد که ازدواج نکرده است عدد ۵ نسبت داده می‌شود. به یک فرد که نقش شوهر دارد عدد ۲ و به فردی که نقش زن دارد، عدد ۰، در حالی که به وضوح در دیتاست این مساله، هیچ تناسبی بین این افراد وجود ندارد. یعنی رابطه‌ای از این جهت که فرد ازدواج نکرده فاصله عددش با زن یا شوهر چقدر باید باشد، وجود ندارد.

بنابراین از ادامه دادن مساله با این روش منصرف شده و در ادامه به سراغ روش One hot encoding می‌رویم.

قبل از کدگذاری با استفاده از این روش، باید راه‌حلی برای داده‌های گم‌شده پیدا کنیم. در این مساله، ما برای داده‌های عددی، میانگین هر ستون را جایگزین مقدار گم‌شده کرده و برای داده‌های اسمی، مقداری که بیشترین تکرار را دارد جایگزین مقدار گم‌شده می‌کنیم. این کار با استفاده از کلاس DataFrameImputer انجام شده‌است.

می‌دانیم که one hot encoding به این صورت است که به ازای هر مقدار یک متغیر، یک ستون جدید ایجاد می‌شود و ردیف‌هایی که آن مقدار را دارند، در آن ستون ۱ و در دیگر ستون‌ها مقدار ۰ خواهند گرفت. ستون‌هایی از دیتافریم را که نیاز به تبدیل شدن به one hot دارند به عنوان ورودی به تابع get_dummies از کتابخانه pandas می‌دهیم. ورودی‌ها را انتخاب کرده و برای خروجی نیز صرفا یکی از ستون‌های مربوط به «بیشتر بودن درآمد از ۵۰ هزار دلار» یا «کمتر مساوی بودن درآمد با ۵۰ هزار دلار» را انتخاب می‌کنیم. دلیل این کار این است که هر گاه مقدار ستون اولی برای یک ردیف برابر با یک باشد، مقدار آن ردیف در ستون دوم برابر با صفر است و برعکس.

یکی دیگر از مراحل پیش‌پردازش، نرمال کردن داده‌ها است. نرمال کردن داده‌ها با این هدف انجام می‌شود که همه مقادیر بین ۰ تا ۱ باشند و ناخواسته تاثیر یک ستون (ویژگی) بیشتر از ستون دیگری در نظر گرفته نشود.

۲ بارگذاری داده‌ها و انجام پیش‌پردازش

پس از بارگذاری داده‌ها و انجام پیش‌پردازش‌های ذکرشده در بخش ۱، ستون label_>50K را به عنوان label در نظر گرفته و بقیه ستون‌های سمت چپ را به عنوان ورودی شبکه عصبی در نظر می‌گیریم.

ابعاد ورودی و خروجی نهایی ما به صورت زیر خواهد بود:

```
x_train shape: (32561, 105)
y_train shape: (32561,)
```

مجموعا ۳۲۵۶۱ داده داریم (که در آینده به عنوان داده آموزش و ارزیابی استفاده خواهد شد) و هر داده، ۱۰۵ بعد (یا ویژگی) دارد.

۳ طراحی مدل

برای طراحی مدل، می‌دانیم لایه ورودی، باید به اندازه شکل داده‌های ما نورون داشته باشد، بنابراین از shape متغیر x_train به عنوان شکل ورودی تابع استفاده می‌کنیم. همچنین می‌دانیم که در لایه آخر می‌توانیم صرفا یک نورون داشته باشیم که با یک تابع فعال‌سازی sigmoid مقدار دو کلاس را مشخص می‌کند.

تابع ارزیابی تمامی لایه‌های میانی را نیز برابر relu انتخاب می‌کنیم.

برای انتخاب بهترین مدل و بهترین مقادیر هایپرپارامترها، ۲۰ درصد از داده‌های آموزشی را جدا کرده و به عنوان داده ارزیابی در نظر می‌گیریم. با ساختارهای مختلف شبکه و مقادیر هایپرپارامتر مختلف، شبکه را آموزش داده و بر اساس مقدار loss مربوط به داده ارزیابی بهترین ساختار را انتخاب می‌کنیم.

طبق درخواست بخش ۵ صورت سوال، از بهینه‌ساز Adam و تابع هزینه binary crossentrpy (به دلیل وجود دو کلاس دسته‌بندی) برای بهینه‌سازی مدل خود استفاده می‌کنیم.

انتخاب ساختار شبکه عصبی بر اساس آزمون و خطا است. در تمامی آزمون‌ها از تابع فعال‌سازی relu در لایه‌های میانی و از تابع فعال‌سازی sigmoid در لایه پایانی استفاده می‌کنیم. دلیل استفاده از تابع relu این است که به طور کلی مشاهده شده است که در مسائل مختلف، نتیجه بهتری می‌دهد. دلیل استفاده از تابع فعال‌سازی sigmoid در لایه آخر این است که نیاز داریم که خروجی بین ۰ و ۱ باشد. در رابطه با تعداد لایه‌ها و تعداد نورون‌های هر لایه، آزمایش‌هایی انجام داده‌ایم که به شرح زیر است:

آزمون ۱:

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 128)	13568
dense_13 (Dense)	(None, 1)	129

Total params: 13,697
Trainable params: 13,697
Non-trainable params: 0

loss: 0.2874 - accuracy: 0.8665 - val_loss: 0.3137 - val_accuracy: 0.8575

آزمون ۲:

Layer (type)	Output Shape	Param #
dense_14 (Dense)	(None, 512)	54272

dense_15 (Dense)	(None, 1)	513

Total params: 54,785
Trainable params: 54,785
Non-trainable params: 0

loss: 0.2821 - accuracy: 0.8692 - val_loss: 0.3297 - val_accuracy: 0.8518

آزمون ۳:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1000)	106000
dense_1 (Dense)	(None, 20)	20020
dense_2 (Dense)	(None, 1)	21

Total params: 126,041
Trainable params: 126,041
Non-trainable params: 0

loss: 0.2755 - accuracy: 0.8718 - val_loss: 0.3299 - val_accuracy: 0.8541

آزمون ۴:

مدل آزمون ۱ را که بهترین نتیجه را تا کنون گرفته است، با تابع فعال‌سازی tanh برای لایه میانی، آموزش دادیم و نتایج به صورت زیر بود:

loss: 0.3006 - accuracy: 0.8605 - val_loss: 0.3190 - val_accuracy: 0.8500

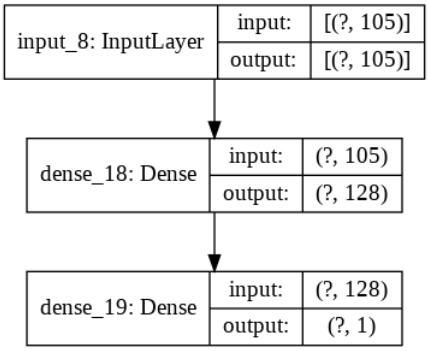
آزمون‌های دیگر:

در آزمون‌های دیگری، ضریب یادگیری را تغییر دادیم و بیش از این تعداد لایه‌ها و نورون‌ها را تغییر دادیم ولی نتیجه بهبود نیافت و از آوردن آن آزمایش‌ها در گزارش صرف نظر شده است.

مشاهده می‌شود که رسیدن به دقت حدود ۸۵ درصد برای مساله چندان مشکل نیست و با افزایش تعداد نورون‌های لایه میانی یا افزایش تعداد لایه‌ها، مساله بهبود پیدا نمی‌کند. بنابر اصل تیغ اوکام، ساده‌ترین مدل ــ که در اینجا بهترین نتیجه بر روی داده‌های ارزیابی را نیز به دست آورده است ــ را انتخاب می‌کنیم. (مدل آزمون اول)

با انتخاب این هایپرپارامترها و آموزش مدل برای ۱۰ epoch ، دقت مدل بر روی داده‌های ارزیابی به ۸۶ درصد و بر روی داده‌های آموزش به ۸۷ درصد رسید.

ساختار مدل:



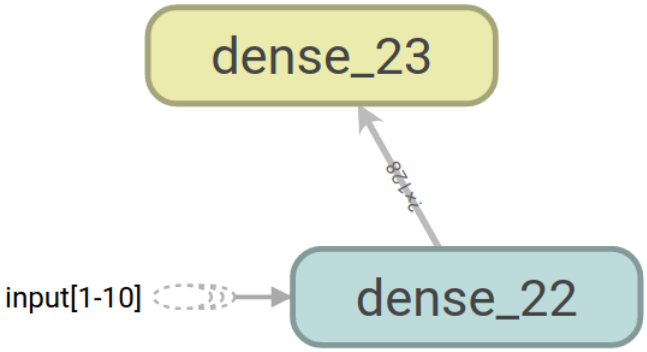
Hyperparameters:
Optimizer: Adam (lr = 0.0005)
loss: binary crossentropy
epochs: 10
shuffle: True

loss: 0.2874
accuracy: 0.8665
val_loss: 0.3137
val_accuracy: 0.8575

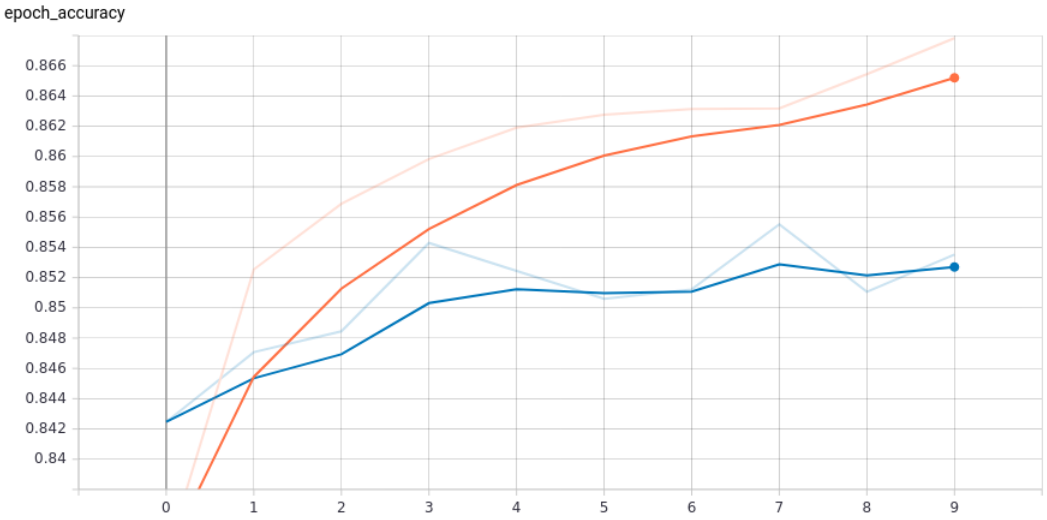
۴ تصویر گراف مدل توسط Tensorboard

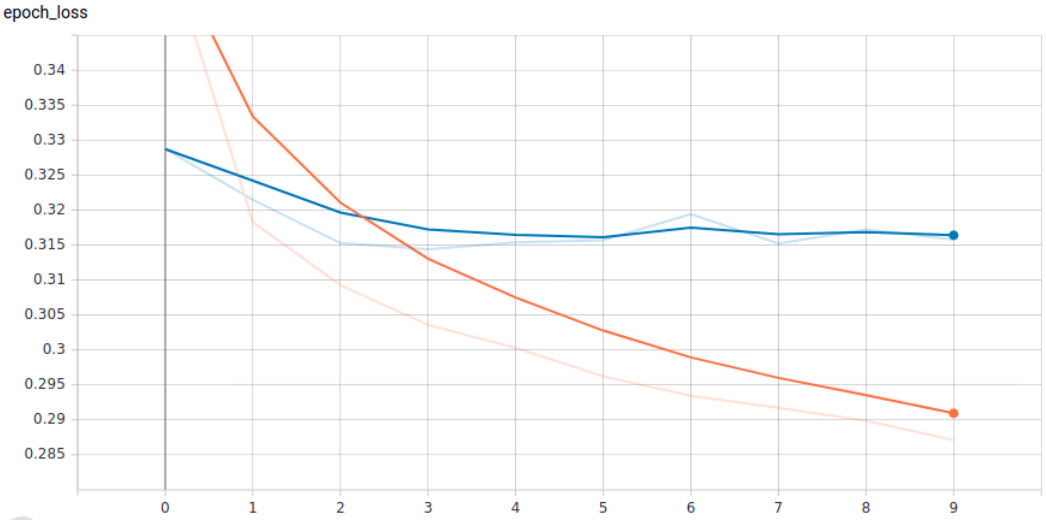
برای استفاده از Tensorboard، یک callback برای آن ساخته و در هنگام اجرای تابع fit، آن را به عنوان ورودی به تابع پاس می‌دهیم.

پس از اجرای Tensorboard، باید دقت شود که تمامی مدل‌هایی که از ابتدای این session نرم‌افزار Google Colab ساخته و آموزش داده شده است، نمایش داده می‌شود. مدل مورد نظر ما، آخرین مدل ساخته شده است. می‌توان با ریست کردن session تنها یک مدل را نمایش داد.



نمودارهای نحوه کاهش دقت و مقدار تابع زیان هم به صورت زیر است. در نمودارهای زیر، محور افقی، شماره epoch و محور عمودی مقدار دقت (در نمودار اول) و مقدار تابع زیان (در نمودار دوم) است. همچنین خطوط نارنجی رنگ مربوط به داده آموزش و خطوط آبی مربوط به داده ارزیابی است.




epoch_loss
0.34
0.335
0.33
0.325
0.32
0.315
0.31
0.305
0.3
0.295
0.29
0.285
0
1
2
3
4
5
6
7
8
9