

پاسخ تمرین سری ۱

محمدرضا عزیزی

۹۸۱۳۱۰۲۲

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

mrazizi@aut.ac.ir

۱ راهکاری برای پیش پردازش داده‌ها

داده‌ها با استفاده از کتابخانه pandas بارگذاری کرده و از آن جایی 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 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 شده است.

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 صرفاً 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,)
```

مجموعاً 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 صرفاً 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_crossentropy (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		

———— Training ————

loss: 0.2734 / accuracy: 0.8729 / precision: 0.765 / recall: 0.6775 /
f1_score = 0.6885

Confusion Matrix – for train:

TP = 4228.0 FP = 1298.0
FN = 2013.0 TN = 18509.0

—— Validation ——

loss: 0.3252 / accuracy: 0.8554 / precision: 0.7232 / recall: 0.6662 /
f1_score = 0.6799

Confusion Matrix – for validation:

TP = 1066.0 FP = 408.0
FN = 534.0 TN = 4505.0

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		

—— Training ——

loss: 0.2835 / accuracy: 0.8684 / precision: 0.7566 / recall: 0.6645 /
f1_score = 0.7076

Confusion Matrix – for train:

TP = 4147.0 FP = 1334.0
FN = 2094.0 TN = 18473.0

—— Validation ——

loss: 0.3238 / accuracy: 0.8537 / precision: 0.7273 / recall: 0.6469 /
f1_score = 0.6847

Confusion Matrix – for validation:

TP = 1035.0 FP = 388.0
FN = 565.0 TN = 4525.0

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		

———— Training ————

loss: 0.2700 / accuracy: 0.8738 / precision: 0.7683 / recall: 0.6779 /
f1_score = 0.7203

Confusion Matrix – for train:

TP = 4231.0 FP = 1276.0
FN = 2010.0 TN = 18531.0

———— Validation ————

loss: 0.3274 / accuracy: 0.8534 / precision: 0.7391 / recall: 0.6231 /
f1_score = 0.6762

Confusion Matrix – for validation:

TP = 997.0 FP = 352.0
FN = 603.0 TN = 4561.0

آزمون ۴:

آزمون ۱ مقدار accuracy بالاتری دارد و آزمون ۲ مقدار امتیاز f1 بالاتری. مقدار بیشتر بودن امتیاز آزمون ۲ در حد ۱ صدم است و بهتر است به خاطر ۱ صدم، پیچیدگی مدل را چندین برابر نکنیم؛ زیرا همان طور که مشاهده می‌شود آزمون ۱ حدود ۱۳ هزار پارامتر و آزمون ۲ حدود ۵۴ هزار پارامتر دارد. پس مدل ۱ را بهترین مدل در نظر می‌گیریم.

مدل آزمون ۱ را با تابع فعال‌سازی tanh برای لایه میانی، آموزش دادیم و نتایج به صورت زیر بود:

———— Training ————

loss: 0.3011 / accuracy: 0.8607 / precision: 0.7466 / recall: 0.6336 /
f1_score = 0.6854

Confusion Matrix – for train:

TP = 3954.0 FP = 1342.0
FN = 2287.0 TN = 18465.0

———— Validation ————

loss: 0.3199 / accuracy: 0.8503 / precision: 0.7181 / recall: 0.6431 /
f1_score = 0.6785

Confusion Matrix – for validation:

TP = 1029.0 FP = 404.0
FN = 571.0 TN = 4509.0

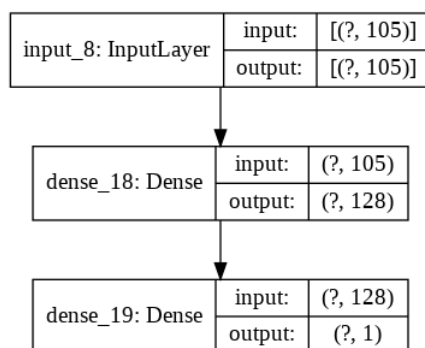
آزمون‌های دیگر:

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 – که در اینجا بهترین نتیجه بر روی داده‌های ارزیابی را نیز به دست آورده است – 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:

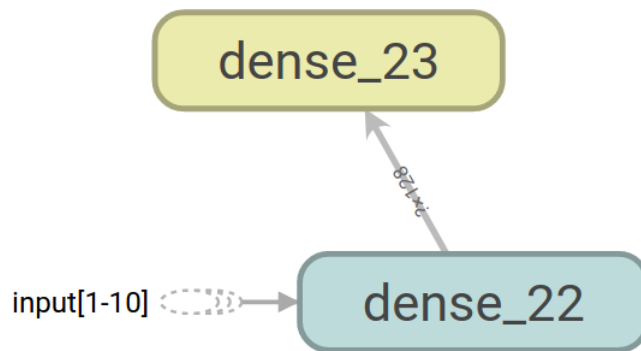
Optimizer: Adam (lr = 0.0005)

loss: binary_crossentropy
epochs: 10
shuffle: True

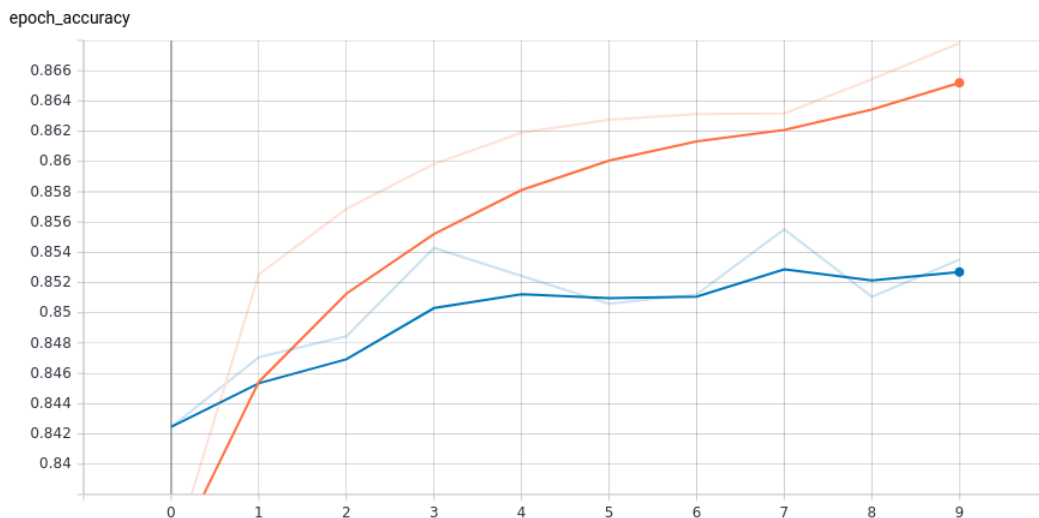
loss: 0.2874
accuracy: 0.8665
val_loss: 0.3137
val_accuracy: 0.8575

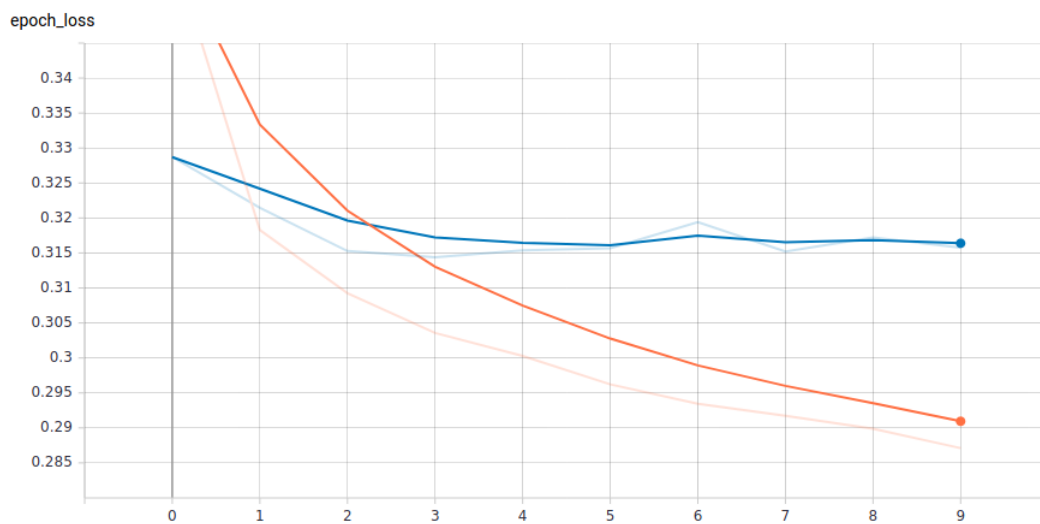
۴ تصویر گراف مدل توسط Tensorboard

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.





۵ بهینه‌ساز مدل و معیارهای ارزیابی

بهینه‌ساز Adam و تابع هزینه Binary CrossEntropy – همان‌طور که در بخش ۳ توضیح داده شد – انتخاب شده است.

تمامی معیارهای ارزیابی مدل، در مجموعه معیارهای تنسورفلو وجود دارد. برای معیار ماتریس درهم‌ریختگی نیز، ۴ معیار، TP، FP و TN و TP را از مدل گرفته و کنار هم قرار می‌دهیم. تمامی این مقادیر به ازای هر آزمایش، در بخش ۳ داکيومنت نوشته شده است.

۶ تقسیم داده‌ها به سه دسته

تا به اینجا، فایل adult.data را بارگذاری کرده بودیم و ۸۰ درصد این فایل را به عنوان داده آموزش و ۲۰ درصد را به عنوان داده ارزیابی در نظر گرفته بودیم.

برای داده تست، فایل adult.test را بارگذاری می‌کنیم و تمامی پیش‌پردازش‌هایی که بر روی داده قبلی انجام داده بودیم، بر روی این مجموعه داده نیز انجام می‌دهیم.

دو نکته در بارگذاری این دیتاست تست وجود دارد. قبل از بارگذاری، خط اول دیتاست که اطلاعاتی است که مربوط به دادگان نیست را به صورت دستی پاک کرده‌ایم. همچنین در دیتاست تست، بعد از مقادیر label، نقطه وجود دارد که این نقاط را نیز با استفاده از تابع replace کتابخانه pandas پاک کرده‌ایم.

وقتی با استفاده از تابع get_dummies از کتابخانه pandas عمل تبدیل دیتافریم تست به کد one hot را انجام می‌دهیم، می‌بینیم که تعداد ستون‌های ایجاد شده برابر با ۱۰۶ ستون است، در حالی که تعداد ستون‌های ایجاد شده از داده‌های آموزشی برابر با ۱۰۷ ستون است. دلیل این اتفاق این است که داده‌های تست، یکی از مقادیر یکی از ستون‌های اسمی را ندارند. در این حالت نمی‌توان ورودی را به شبکه عصبی داد.

باید ابتدا به دنبال ستونی که موجود نیست بگردیم. برای این کار، لیست ستون‌های دیتافریم آموزشی و دیتافریم تست را با هم مقایسه می‌کنیم. مشاهده می‌کنیم که ستونی با نام native-country_Holand-Netherlands در دیتافریم تست در مکان شماره ۷۸ وجود ندارد. این به این معنی است که هیچ داده‌ای با این مقدار وجود ندارد. یک ستون با این نام و در همین مکان به دیتافریم تست اضافه می‌کنیم و در نهایت بررسی می‌کنیم که شکل دو دیتافریم تست و آموزش مشابه هم باشد:

```
test df: (16281, 107)
```

```
train df: (32561, 107)
```

در بخش در رابطه با آموزش شبکه با استفاده از داده‌های آموزش و بررسی کیفیت عملکرد شبکه در هر epoch توسط داده‌های ارزیابی صحبت شد. تعداد epoch آموزش برابر ۱۰ قرار گرفته است که با توجه به نمودارهایی که در بخش ۴ آورده شده است، مشاهده می‌شود که تغییرات دقت و مقدار تابع زیان برای داده‌های ارزیابی تقریباً به ۰ رسیده است. یعنی شبکه به اندازه کافی آموزش دیده است.

evaluate ✓

٨ تصویر گراف مدل توسط Tensorboard

