

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

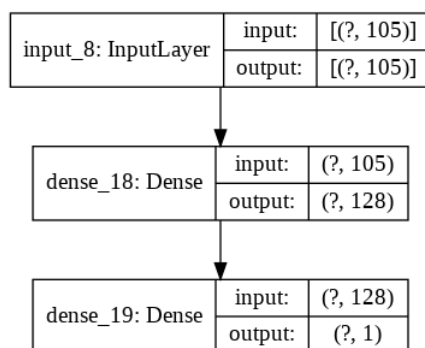
آزمون‌های دیگر:

در آزمون‌های دیگری، ضریب یادگیری را تغییر دادیم و بیش از این تعداد لایه‌ها و نوروها 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 نوروهای 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:

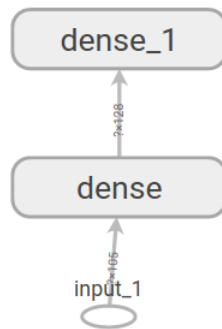
Optimizer: Adam (lr = 0.0005)

loss: binary_crossentropy
epochs: 10
shuffle: True

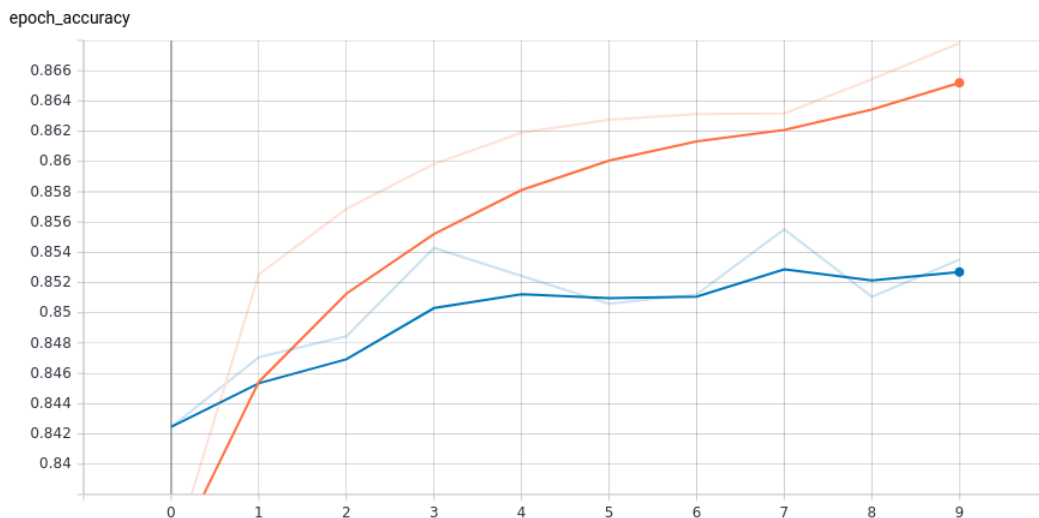
loss: 0.2874
accuracy: 0.8665
val_loss: 0.3137
val_accuracy: 0.8575

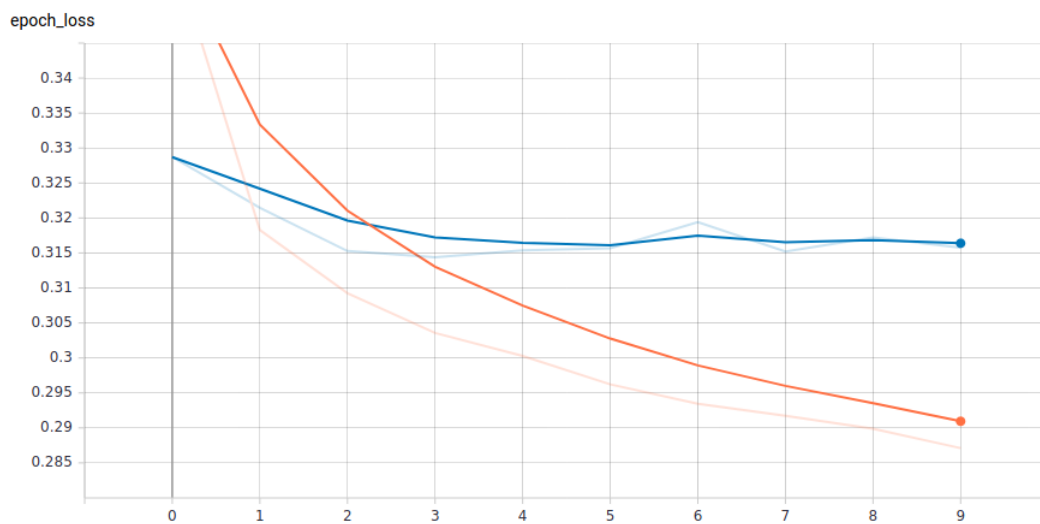
۴ تصویر گراف مدل توسط Tensorboard

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.

در نهایت داده‌های آزمون نیز scale می‌شوند.

نتیجه صدا زدن تابع evaluate بر روی داده‌های تست به صورت زیر است:

———— Test ————

loss: 4110.4199 / accuracy: 0.7914 / precision: 0.8516 / recall: 0.1417 /
f1_score = 0.6785

Confusion Matrix – for validation:

TP = 545.0 FP = 95.0
FN = 3301.0 TN = 12340.0

مشاهده می‌شود که دقت مدل ۷۹ درصد است که مورد قبول است و تفاوت چندانی با دقت داده‌های ارزیابی ندارد. مقدار precision نیز قابل قبول و مانند داده‌های ارزیابی است. تنها مورد منفی، مقدار بسیار پایین recall است. همان‌طور که در ماتریس پیرایشی نیز دیده می‌شود، تعداد FN ها بر روی داده‌های تست بسیار بالا است. یعنی مدل داده‌های زیادی را به اشتباه منفی می‌پندارد. این رفتار ممکن است به دلیل تفاوت اساسی بین داده‌های آزمون و داده‌های آموزش باشد یا به این دلیل باشد که تعداد داده‌های با برچسب منفی در دیتاست آموزشی بیشتر بوده است و مدل به درستی نتوانسته است رابطه موجود بین ورودی و خروجی را تشخیص دهد.

۵ نمونه از داده‌هایی که مدل پیش‌بینی می‌کند کمتر از ۵۰ هزار دلار درآمد خواهند داشت:

#	0	1	4	5	6
age	25	38	18	34	29
workclass	Private	Private	?	Private	?
fnlwgt	226802	89814	103497	198693	227026
education	11th	HS-grad	Some-college	10th	HS-grad
education-num	7	9	10	6	9
marital-status	Never-married	Married-civ-spouse	Never-married	Never-married	Never-married
occupation	Machine-op-inspct	Farming-fishing	?	Other-service	?
relationship	Own-child	Husband	Own-child	Not-in-family	Unmarried
race	Black	White	White	White	Black
sex	Male	Male	Female	Male	Male
capital-gain	0	0	0	0	0
capital-loss	0	0	0	0	0
hours-per-week	40	50	30	30	40
native-country	United-States	United-States	United-States	United-States	United-States
prediction	<=50K	<=50K	<=50K	<=50K	<=50K
true label	<=50K	<=50K	<=50K	<=50K	<=50K

۵ نمونه از داده‌هایی که مدل پیش‌بینی می‌کند که بالای ۵۰ هزار دلار درآمد خواهند داشت:

#	3	7	10	14	15
age	44	63	65	48	43
workclass	Private	Self-emp-not-inc	Private	Private	Private
fnlwgt	160323	104626	184454	279724	346189
education	Some-college	Prof-school	HS-grad	HS-grad	Masters
education-num	10	15	9	9	14
marital-status	M-c-p	M-c-p	M-c-p	M-c-p	M-c-p
occupation	Machine-op-inspct	Prof-specialty	Machine-op-inspct	Machine-op-inspct	Exec-managerial
relationship	Husband	Husband	Husband	Husband	Husband
race	Black	White	White	White	White
sex	Male	Male	Male	Male	Male
capital-gain	7688	3103	6418	3103	0
capital-loss	0	0	0	0	0
hours-per-week	40	32	40	48	50
native-country	United-States	United-States	United-States	United-States	United-States
prediction	>50K	>50K	>50K	>50K	>50K
true label	>50K	>50K	>50K	>50K	>50K

در جدول زیر، به جای Married-civ-spouse ، M-c-p نوشته شده است.

۸ بررسی تاثیر افزایش تعداد لایه‌ها بر عملکرد شبکه

آزمایش ۱: شبکه فعلی را در نظر می‌گیریم:

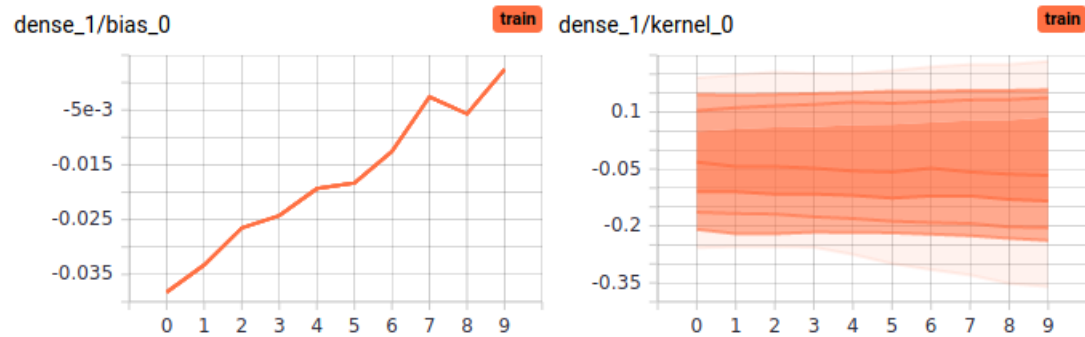
Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	13568
dense_1 (Dense)	(None, 1)	129

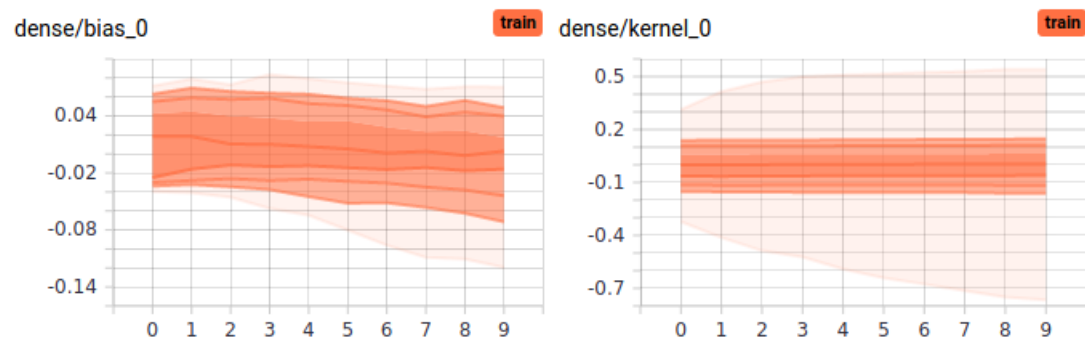
Total params: 13,697
Trainable params: 13,697
Non-trainable params: 0

تغییر وزن‌ها در لایه‌های مختلف با استفاده از Tensorboard به صورت زیر است:

dense_1



dense



مشاهده می‌شود که با گذر زمان (نمودار افقی)، وزن‌ها به اندازه کافی تغییر می‌کنند.

اکنون شبکه‌ای طراحی می‌کنیم که تعداد لایه‌های میانی زیادی دارد. در شبکه زیر، تابع فعال‌سازی همه لایه‌ها به جز لایه آخر، \tanh است. تابع فعال‌سازی لایه آخر sigmoid است. دلیل استفاده از تابع فعال‌سازی \tanh به جای relu این است که تابع فعال‌سازی relu در مقابل $\text{vanishing gradient}$ نسبتاً مقاوم است.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	13568
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 128)	16512
dense_3 (Dense)	(None, 128)	16512
dense_4 (Dense)	(None, 128)	16512
dense_5 (Dense)	(None, 128)	16512
dense_6 (Dense)	(None, 128)	16512
dense_7 (Dense)	(None, 128)	16512
dense_8 (Dense)	(None, 128)	16512
dense_9 (Dense)	(None, 128)	16512
dense_10 (Dense)	(None, 128)	16512
dense_11 (Dense)	(None, 128)	16512
dense_12 (Dense)	(None, 128)	16512
dense_13 (Dense)	(None, 1)	129

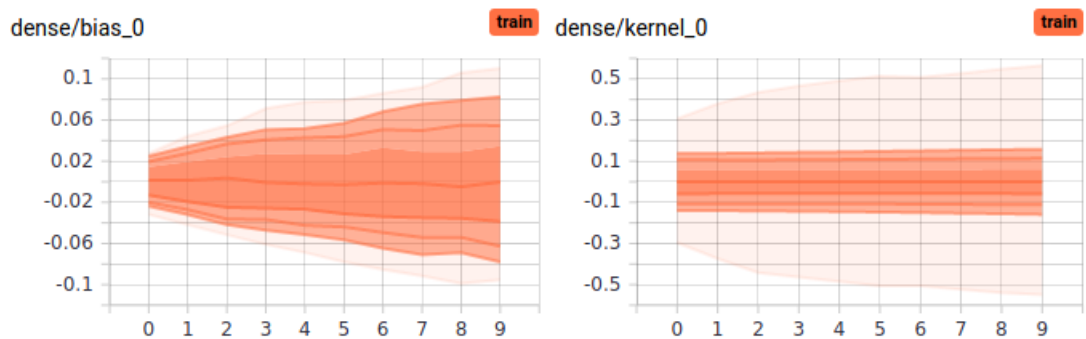
Total params: 211,841

Trainable params: 211,841

Non-trainable params: 0

پس از ۱۰ epoch، دقت این شبکه بر روی داده‌های ارزیابی به ۸۴ درصد رسیده است و این به این معنی است که شبکه آموزش دیده است. تغییرات وزن‌های شبکه را در اولین لایه شبکه بررسی می‌کنیم.

dense



مشاهده می‌شود که تغییرات وزن `kernel_0` نسبتاً کمتر شده است. این به این دلیل است که مقدار گرادیانی که از خروجی گرفته می‌شود، هر چه به لایه‌های ابتدایی‌تر می‌رسد، تاثیرش کمتر می‌شود. به این اتفاق `vanishing gradients` می‌گویند.

۹ بررسی بیش‌برازش مدل

نمودارهای میزان دقت و مقدار تابع زیان مدل، در طول epoch در بخش ۴ آمده است. با توجه به این نمودارها، واضح است که هم میزان تابع زیان برای داده‌های آموزشی پایین آمده است و هم میزان تابع زیان برای داده‌های ارزیابی. در نتیجه، مدل دچار بیش‌برازش نشده است.

برای این که یک مدل را دچار بیش‌برازش کنیم، می‌توانیم مدل را به قدری بزرگ کنیم که مدل به جای یادگیری ارتباط واقعی بین ورودی و خروجی، برچسب هر داده ورودی را حفظ کند. برای این کار هم تعداد لایه‌ها را افزایش داده و هم تعداد نورون‌های لایه میانی را.

مدل‌های مختلفی مورد بررسی قرار گرفت. از مدلی که شامل ۲۰ لایه که هر لایه ۱۲۸ نورون دارد تا نورونی شامل ۳ لایه که دو لایه اولیه هر کدام ۱۰۰۰ نورون دارند. اما هیچ یک دچار بیش‌برازش نشد. همچنین، تعداد epoch آموزش هم زیاد شد اما بیش‌برازش ایجاد نشد. اما نکته کلی بیش‌برازش – همان طور که مطرح شد – این است که مدل بیش از اندازه برای داده‌ها بزرگ است.

آخرین مدلی که بررسی شد، مدل زیر است.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 10000)	1060000
dense_5 (Dense)	(None, 10000)	100010000
dense_6 (Dense)	(None, 1)	10001
Total params: 101,080,001		
Trainable params: 101,080,001		
Non-trainable params: 0		

که به دلیل تعداد پارامتر بسیار زیادش، به مدت یک ساعت مورد آموزش قرار گرفت و نتیجه نهایی، ۸۳ درصد دقت در آموزش و ۸۲ درصد دقت در ارزیابی بود و بیش‌برازش نشد.

۱۰ بهترین مدل برای داده‌ها

بهترین مدل، در بخش ۳ معرفی شد. در اینجا نیز مجدداً ساختار مدل و نتایج آن آورده می‌شود:

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

———— Test ————

loss: 4110.4199 / accuracy: 0.7914 / precision: 0.8516 / recall: 0.1417 /
f1_score = 0.6785

Confusion Matrix – for validation:

TP = 545.0 FP = 95.0
FN = 3301.0 TN = 12340.0

وزن‌های مدل نهایی نیز در فایل model.h5 ذخیره شده است که در آینده می‌توان از آن‌ها استفاده کرد.

وزن‌های آزمایش‌های دیگر نیز ذخیره شده‌اند، اما به دلیل این که حجم فایل تحویلی بیش از اندازه زیاد نشود، همراه فایل‌ها تحویل داده نمی‌شود و در صورت نیاز، در تحویل حضوری استفاده خواهند شد.