"بسمه تعالی"

گزارش تمرین عملی دوم یادگیری ماشین – زهرا لطیفی – ۹۹۲۳۰۶۹

سوال اول

ابتدا کتابخانه نامپای ایمپورت شده. سپس دو آرایه ۹*۹ وردی را همانطور که در صورت سوال خواسته شده بود تعریف کردیم و L و H را ساختیم. به عنوان خروجی هم، برداری تعریف کردیم که دو کلاس 0 برای L و 1 برای H دارد. پس از این بخش، تابع فعالسازی سخت تک قطبی را تحت عنوان step(x) تعریف کردیم که به ازای مقادیر منفی خروجی 0 و به ازای سایر مقادیر خروجی 1 را بر می‌گرداند.

سپس وزن‌ها را با توزیع نرمال مقداردهی اولیه می‌کنیم و بایاس را برابر ۱- تعریف می‌کنیم. حال تابع train تعریف شده که در هر مرحله، بایاس، وزن‌ها و خطا را به عنوان ورودی گرفته، با روابط موجود در جزوه برای پرسپترون تک‌لایه، این سه ورودی را اصلاح کرده و به عنوان خروجی بر می‌گرداند.

برای شروع، error = 0 را تعریف کرده و یک بار تابع train را فراخوانی کردیم. سپس تا هنگامی که error = 0 نشده، آموزش بر روی تمام داده‌های ورودی را داخل حلقه while تکرار می‌کنیم تا شبکه آموزش داده شود.

نهایتا از همان ورودی‌های آموزش استفاده کردیم تا شبکه را امتحان کنیم. شاهدیم که با ضریب یادگیری 0.01 توانستیم در ۶۰ ایپاک، خروجی کاملا درستی در هر بار ران بگیریم.

```
Input: [0 1 1 0 1 1 0 0 0]
Predicted label: L
60
Input: [0 1 0 0 0 0 0 1 0]
Predicted label: H
60
```

یک بار هم داده‌ها را کمی نویزی کردیم:

```
test = np.array([[0, 1, 0,
                  0, 1, 1,
                  0, 0, 0], # noisy L
                 [0, 1, 0,
                  1, 0, 1,
                  0, 1, 0]]) # noisy H
```

این بار هم خروجی به درستی دسته بندی را انجام داد اما این بار در برخی ران‌ها پاسخ اشتباه می‌گیریم که علت آن بسیار محدود بودن دادگان آموزش است.

```
Input: [0 1 0 0 1 1 0 0 0]
Predicted label: L
Input: [0 1 0 1 0 1 0 1 0]
Predicted label: H
```

سوال دوم

در این فاز کتابخانه‌های پانداس، نامپای، matplotlib، seaborn، sklearn، tensorflow و keras ایمپورت شده‌اند. داده‌ها بارگذاری شده و اطلاعات این مجموعه داده با دستور datase.info و dataset.describe() بدست آمدند. پس از آن برای اینکه دید بهتری نسبت به داده‌ها داشته باشیم، correlation بین فیچرهای مختلف را رسم کرده‌ایم:

می‌دانیم که داده‌های ما، برچسب‌هایی از نوع string دارند؛ پس با استفاده از OneHotEncoder() برچسب‌ها را برای تمام داده‌ها encode می‌کنیم تا داشته باشیم:

```
Original labels:  ['Dropout' 'Enrolled' 'Graduate']
One-hot encoded labels:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

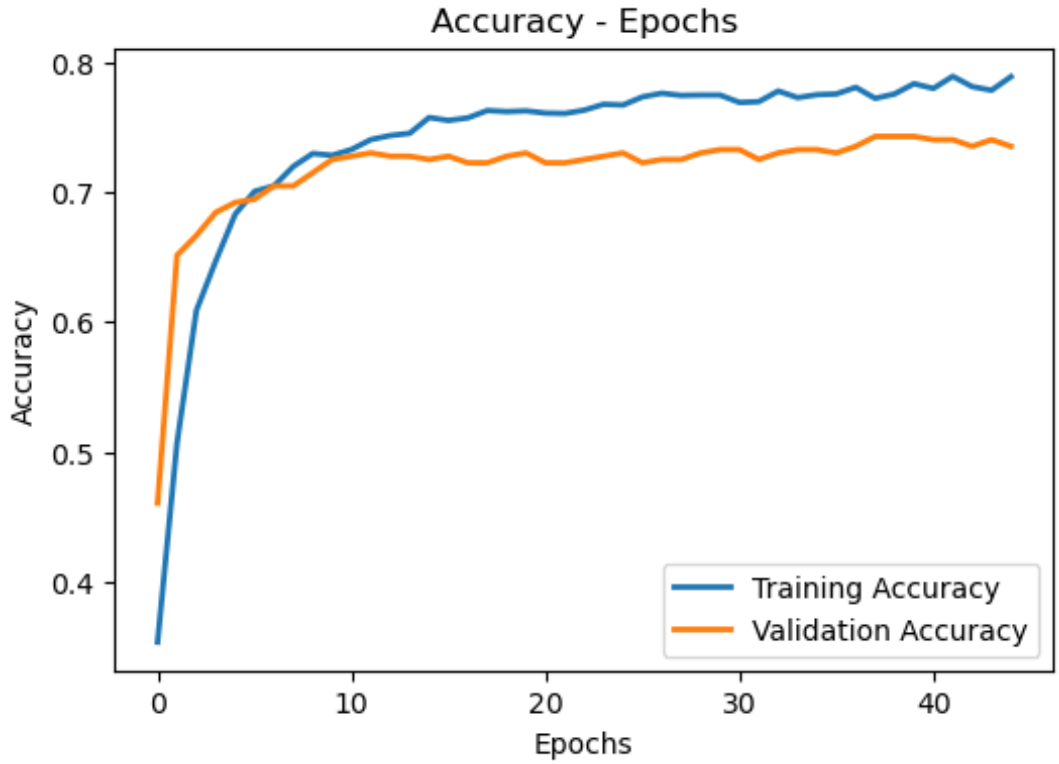
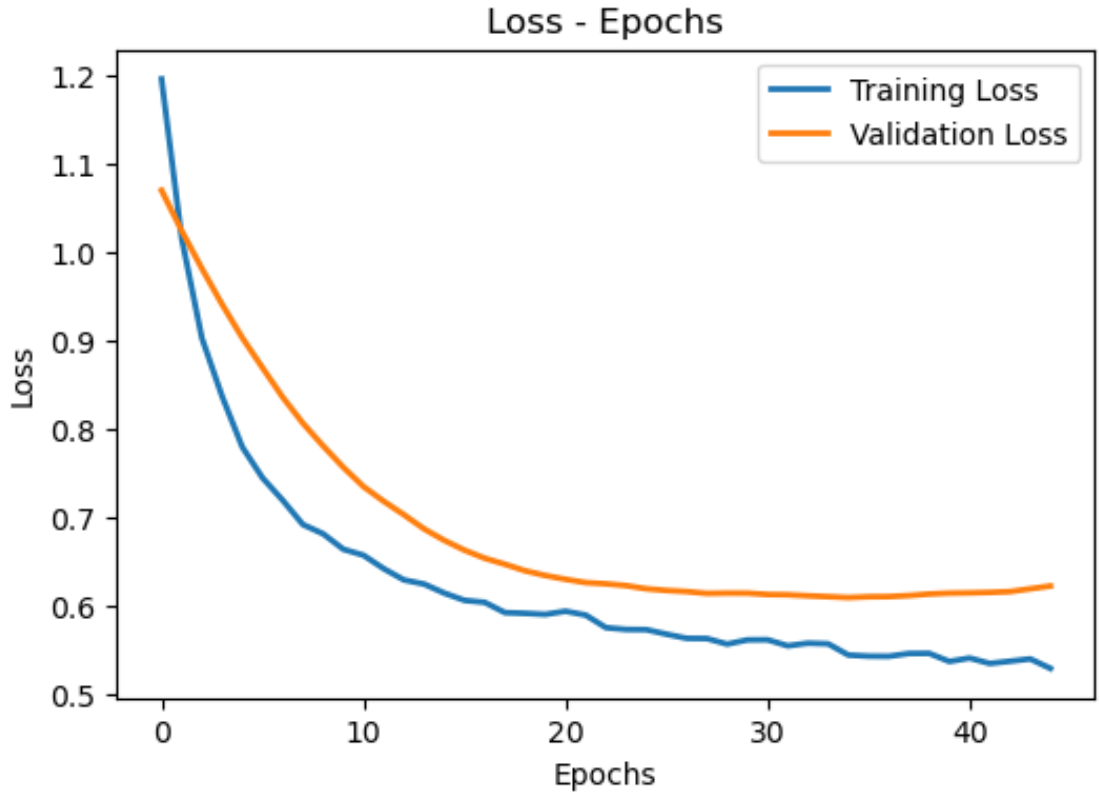
از آنجایی که مقادیر ویژگی‌های مختلف بازه‌های بسیار متفاوتی داشتند، باید *normalization* روی آن‌ها انجام می‌گرفت پس با کد زیر این کار را با روش *Mean-Std* یعنی کم کردن میانگین هر یتون از هر داده و تقسیم کردن بر انحراف معیار داده‌های آن ستون انجام دادیم:

```
 #apply normalization techniques (mean/std)

for column in x_train :

    x_train[column] = (x_train[column] - x_train[column].mean()) /
x_train[column].std()
```

به تعریف مدل می‌رسیم. لایه ورودی ابعادی برابر با ابعاد ورودی (۳۶ فیچر) دارد، لایه بعدی یک لایه پنهان با ۶۴ نورون است که تابع فعالسازی *RelU* دارد. این تابع مشتقی برابر یک در بخش مثبت دارد که کار را راحت‌تر می‌کند و همینطور *sparse* بیشتری مثلا نسبت به *tanh* دارد که به این معناست که از *Overfitting* جلوگیری می‌کند. پس از این لایه از یک لایه *DropOut* استفاده کردیم تا باز هم از *Overfitting* جلوگیری شود. سپس یک لایه پنهان با ۳۲ نورون قرار داده و برای افزایش سرعت آموزش، یک لایه *BatchNormalization* اضافه کردیم. نهایتا یک لایه با ۱۶ نورون دیگر اضافه کردیم تا مدل عمیق‌تر شود. با توجه به اینکه ۳ کلاس در خروجی داریم، لایه خروجی *softmax* با اندازه ۳ خواهد داشت. خلاصه مدل به شکل زیر گزارش شده است:

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_75 (Dense)            (None, 64)                2368

 dropout_18 (Dropout)        (None, 64)                0

 dense_76 (Dense)            (None, 32)                2080

 batch_normalization_20 (Bat  (None, 32)               128
 chNormalization)

 dense_77 (Dense)            (None, 16)                528

 dense_78 (Dense)            (None, 3)                 51

=================================================================
Total params: 5,155
Trainable params: 5,091
Non-trainable params: 64
_________________________________________________________________
```

مدل را *compile* کردیم. در این مرحله از *optimizer* آدام استفاده کردیم اما در روند آموزش ترجیح دادیم ضریب یادگیری را به صورت *manual* به *0.0005* کاهش دهیم. مدل را به داده‌های *train*، در ۱۰۰ ایپاک فیت کردیم. در این مرحله هر بار ۱۰ درصد داده‌ها برای *validation* کنار گذاشته می‌شود و از *earlyStopping* استفاده کردیم تا دچار *Overfitting* نشویم. در ۵۸ ایپاک به دقت ۷۹ درصد بر روی داده‌های آموزش و ۷۴ درصد بر روی داده‌های *valid* رسیدیم.

```
Epoch 58/100
14/14 [==============================] - 0s 5ms/step - loss: 0.5141 - accuracy: 0.7941 -
val_loss: 0.6133 - val_accuracy: 0.7430
Epoch 58: early stopping
```

نمودارهای دقت و *loss* هم رسم شده اند که به شرح زیر هستند: