به نام خدا

گزارش تمرین پنجم درس الگوریتم‌های یادگیری ماشین

استاد راهنما: دکتر کمندی

سیده محیا معتمدی

۸۱۰۸۹۷۰۵۳

مینو احمدی

۸۱۰۸۹۷۰۳۲

دانشکده علوم مهندسی

پردیس دانشکده‌های فنی دانشگاه تهران

فهرست مطالب

سوال ۱

در این سوال می خواهیم الگوریتم backpropagationرا به همراهی تمام روابط و فرمولها برای تابع فعالساز سیگموید شرح دهیم.

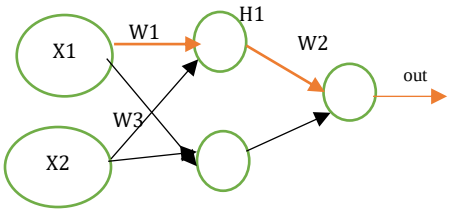
$$sigmoid\ function = \ \sigma = \frac{1}{1+e^{-x}}$$

$$\rightarrow \acute{\sigma} = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(1-\sigma)$$

تابع loss را به صورت زیر تعریف می‌کنیم

$$error = \frac{1}{2}(predicted - actual)^2$$

فرض میکنیم یک شبکه عصبی با یک لایه مخفب به شکل زیر داریم و میخواهیم برای مسیر قرمز رنگ عملیات را انجام دهیم.



$$q1 = w1 * x1$$

$$q2 = w2 * h1$$

$$q3 = w3 * x2$$

$$h1 = \sigma(q1 + q3) = \frac{1}{1+e^{-(w1*x1+w3*x2)}}$$

$$\frac{\partial error}{\partial\, w2} = \frac{\partial error}{\partial out}\frac{\partial out}{\partial q2}\frac{\partial q2}{\partial w2} = (out - actual)\big((1-out)out\big)h1$$

$$= (out - actual)\big((1-out)out\big)\frac{1}{1+e^{-(w1*x1+w3*x2)}}$$

سوال ۲

در این تمرین قصد داریم یک شبکه برای classification بر روی دیتاست breast_cancer آموزش دهیم.

(a, b

در ابتدا دیتاست را به کمک کتابخانه sklearn لود کرده و به کمک روش train_test_split آنها را به دسته‌های تست و ترین تقسیم بندی میکنیم. ۸۰ درصد داه‌ها را به داده های آموزش و ۲۰ درصد داده ها را به داده‌های تست اختصاص می‌دهیم

```
X, y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=0)
```

Figure ۱- breast_cancer Dataset

همچنین توسط روش standardScaler از کتابخانه sklearn داده‌ها را نرمالایز می‌کنیم.

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure ۲- نرمال کردن داده‌ها

(c

در این بخش می‌خواهیم یه کمک کتابخانه tensorflow شبکه را آموزش دهیم.

معماری شبکه

```
model1 = Sequential()
model1.add(Dense(units=15,kernel_initializer='he_uniform',activation='relu',input_dim=input_size))
model1.add(Dense(units=15,kernel_initializer='he_uniform',activation='relu'))
model1.add(Dense(units=1,kernel_initializer='glorot_uniform',activation='sigmoid'))
```

Figure ۳- معماری شبکه

همانطور که در تصویر بالا مشخص است شبکه از ۳ لایه fully connected(Dense) تشکیل شده است که به ترتیب ۱۵و ۱۵و ۱ نورون دارند.

هایپر پارامترها

```
print(np.unique(y_train))
```

```
[0 1]
```

Figure ۴- targetهای شبکه

همانطور که در شکل۳ مشخص است target ها به صورت binary هستند درنتیجه از تابع خطای binary_crossentropy در شبکه استفاده میکنیم. همچنین در لایه آخر از تابع فالساز sigmoid استفاده می‌کنیم. همچنین در ابتدا از بهینه ساز adam استفاده شده‌است.

تعداد ایپاک‌ها در این شبکه ۱۰۰ قرار داده شده است. البته تعداد ایپاک ۳۰ و ۵۰ نیز امتحان شده‌است و از آنجایی که پیشرفت دقت شبکه ادامه داشته است تعدا ایپاک‌ها به ۱۰۰ افزایش دادیم.

```
model1.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
start = time.time()
history1 = model1.fit(X_train, y_train, validation_split=0.1, batch_size= 100, epochs= 100, callbacks=[callback1, callback2])
end = time.time()
print('training time:' , end-start)
```

Figure ۵- استفاده از هایپر پارامترها در توابع compil, fit

CallBackها

در این شبکه از دو callback استفاده شده‌است که شامل Earlystopping و LearningRateScheduler می‌باشند.

```
def scheduler(epoch, lr):
  if epoch < 10:
    return lr
  else:
    return lr * tf.math.exp(-0.5)
```

```
callback1 = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
callback2 = keras.callbacks.LearningRateScheduler(scheduler)
```

Figure ۶-callbacks

نتایج شبکه

نتایج ایپاک آخر و زمان آموزش شبکه به شرح زیر است.

```
Epoch 39/100
5/5 [==============================] - 0s 8ms/step - loss: 0.1972 - accuracy: 0.9389 - val_loss: 0.2170 - val_accuracy: 0.9565 - lr: 5.0435e-10
training time: 2.5479917526245117
```

Figure ۷- نتایج

همانطو رکه در تصویر بالا مشخص است شبکه در ایپاک ۳۹ متوقف شده است

نمودار خطا و دقت و ماتریس آشفتگی و دقت حاصل از داده‌های تست به شرح زیر است.

```
evaluate(X_test, y_test, model1)
plot_loss(history1)
```

```
[[43  4]
 [ 6 61]]
accuracy = : 0.9122807017543859
```

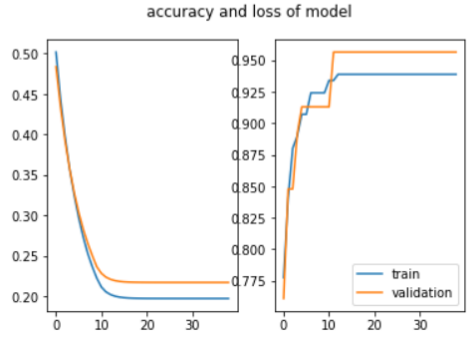


Figure ۸-نمودار خطا و دقت

حذف learning rate scheduler

با حذف callback دوم(Learning rate scheduler) نتایج بهتری به دست می‌اوریم ولی از آنجایی که تعداد ایپاک ها به 74 افزایش می‌یابد زمان آموزش مقداری افزایش می‌یابد.

```
Epoch 74/100
5/5 [==============================] - 0s 10ms/step - loss: 0.0638 - accuracy: 0.9829 - val_loss: 0.0689 - val_accuracy: 1.0000
training time: 4.9283528327941895
```

Figure ۹- نتایج آموزش بدون callback

```
evaluate(X_test, y_test, model2)
plot_loss(history2)
```

```
[[43  4]
 [ 4 63]]
accuracy = : 0.9298245614035088
```

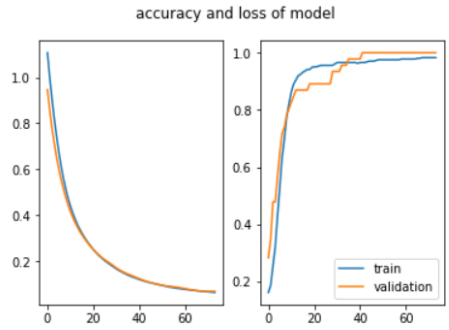


Figure ۱۰- نمودار خطا و دقت

تغییر batch size

در این بخش تعداد batch ها را به ۱۰ تغییر داده‌ایم که نتیج به شرح زیر هستند.

```
Epoch 43/100
41/41 [==============================] - 0s 4ms/step - loss: 0.0216 - accuracy: 0.9927 - val_loss: 0.0771 - val_accuracy: 0.9565
training time: 6.401289701461792
```

Figure ۱۱- نتایج batchsize = 10

از آنجایی که در بعد از هر iteration وزن‌ها به روز رسانی می‌شوند تعداد ایپاک‌ها به ۴۳ کاهش یافته است ولی زمان آموزش افزایش چشم گیری داشته‌است.

نمودار خطا و دقت و ماتریس آشفتگی و دقت حاصل از داده‌های تست به شرح زیر است.

```
evaluate(X_test, y_test, model3)
plot_loss(history3)
```

```
[[46  1]
 [ 2 65]]
accuracy = : 0.9736842105263158
```

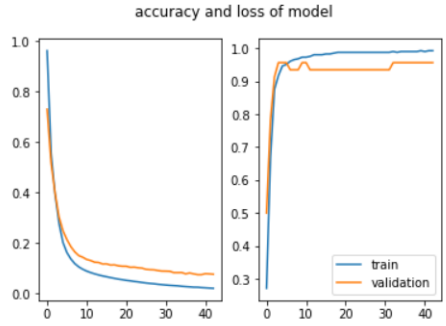


Figure ۱۲-نمودار خطا و دقت

تغییر بهینه ساز

در این بخش از دو بهینه ساز SGD , RMSprop استفاده شده است.

RMSProp

این بهینه ساز یکی از پرکاربرد ترین بهینه سازها در امر classification است در نتیجه همانطور که در تصویر پایین مشخص است با تعداد ایپاک کم‌تر به نتیجه بهتری دست یافته است.

```
Epoch 45/100
5/5 [==============================] - 0s 8ms/step - loss: 0.0565 - accuracy: 0.9829 - val_loss: 0.0635 - val_accuracy: 0.9783
training time: 3.719090461730957
```

Figure ۱۳- نتایج با بهینه ساز RMSProp

نمودار خطا و دقت و ماتریس آشفتگی و دقت حاصل از داده‌های تست به شرح زیر است.

```
evaluate(X_test, y_test, model4)
plot_loss(history4)
```

```
[[46  1]
 [ 4 63]]
accuracy = : 0.956140350877193
```

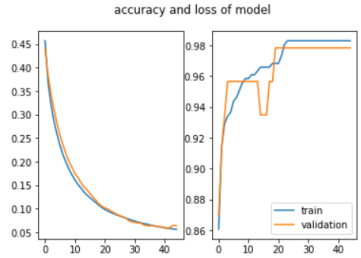


Figure ۱۴نمودار خطا و دقت

SGD

از آنجایی که این بهینه‌ساز بیشتر مناسب مسائل regression است انتظار داریم که نتیجه خوبی نگیری که همانطور که در تصویر پایین مشخص است تعداد ۱۰۰ ایپاک طی شده و زمان آموزش بسیار افزایش پیدا کرده است(تقریبا ۴ برابر) و دقت نیز کاهش یافته است.

```
Epoch 100/100
5/5 [==============================] - 0s 8ms/step - loss: 0.1363 - accuracy: 0.9633 - val_loss: 0.1540 - val_accuracy: 0.9565
training time: 11.006525754928589
```

Figure ۱۵- نتایج بهینه ساز SGD

نمودار خطا و دقت و ماتریس آشفتگی و دقت حاصل از داده‌های تست به شرح زیر است.

```
[[42  5]
 [ 4 63]]
accuracy = : 0.9210526315789473
```

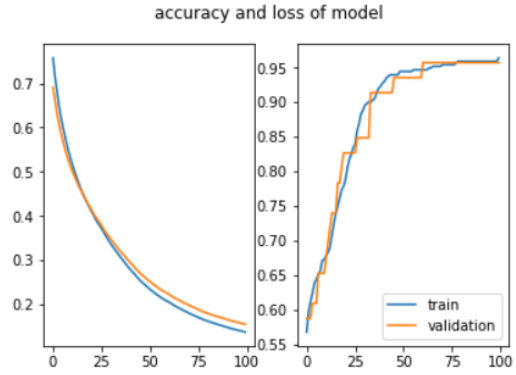


Figure ۱۶نمودار خطا و دقت

سوال۳

در این سوال قصد داریم قصد داریم با دیتاست IMDBیک شبکه عصبی بازگشتی را آموزش دهیم.

a)

برای این کار ابتدا دیتاست را از کتابخانه tensorflow لود کرده و آنها را به داده‌های تست و ترین تقسیم بندی میکنیم.

```
num_words=10000
(train_data,train_labels),(test_data,test_labels) = tf.keras.datasets. imdb.load_data(num_words=num_words)
```

Figure ۱۷- دیتاست IMDB

از آنجایی که داده‌ها از طول های متفاوتی برخوردارند توسط دستور pad sequence طول داده‌ها رایکسان می‌کنیم.

```
maxlen=100
from keras.preprocessing.sequence import pad_sequences
X_train = pad_sequences(train_data, maxlen=maxlen)
X_test = pad_sequences(test_data, maxlen=maxlen)
```

Figure ۱۸- یکسان سازی طول داده‌ها

(b

Callbackها

در این بخش از callback های early stopping, LearningRateScheduler, ModelCheckPoint استفاده کرده‌ایم که کدهای آنها به شرح زیر است.

```
def scheduler(epoch, lr):
  if epoch < 2:
    return lr
  else:
    return lr * tf.math.exp(-0.5)
checkpoint_filepath = '/tmp/checkpoint'
callback1 = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
callback2 = tf.keras.callbacks.LearningRateScheduler(scheduler)
callback3 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

Figure ۱۹- callBack ها

معماری شبکه

همانطور که در تصویر زیر مشخص است یک شبکه را بایک لایه embedding که dimension آن برابر با num_words است که دیتاست را نیز با همین مقدار لود کرده‌ایم. و خروجی آن نیز ۱۲۸ قرار داده شده است(در ادامه برای کاهش زمان اجرا از ۳۲ نورون استفاده شده‌است.) و ورودی آن برابر با max_len که تعداد کاراکترها در هر داده است قرار داده شده است.

در ادامه یک لایه simple RNN قرار داده شده‌است که تعداد خروجی‌های آن ۶۴ است و سپس یک لایه Dense با یک نورون خروجی قرار داده شده است. در انتها نیز یک activation، sigmoid قرار داده شده است

کد های مربوط به این بخش به شرح زیر است.

```
model1 = Sequential()
model1.add(Embedding(num_words,128,input_length =len(X_train[0])))
model1.add(SimpleRNN(64,input_shape = (num_words,maxlen), return_sequences=False,activation="relu"))
model1.add(Dense(1))
model1.add(Activation("sigmoid"))

print(model1.summary())
model1.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
```

Figure ۲۰- معماری شبکه RNN

آموزش شبکه

شبکه را به ۲۰ ایپاک و تابع زیان binary_crossentropy و بهینه‌سازAdam آموزش داده ایم. همچنین ۱۰ درصد از داده‌های آموزش را به validation اختصاص داده‌ایم.

```
history1 = model1.fit(X_train,train_labels, validation_split = 0.1 ,epochs =20 ,batch_size=128, callbacks=[callback1, callback2, callback3])
```

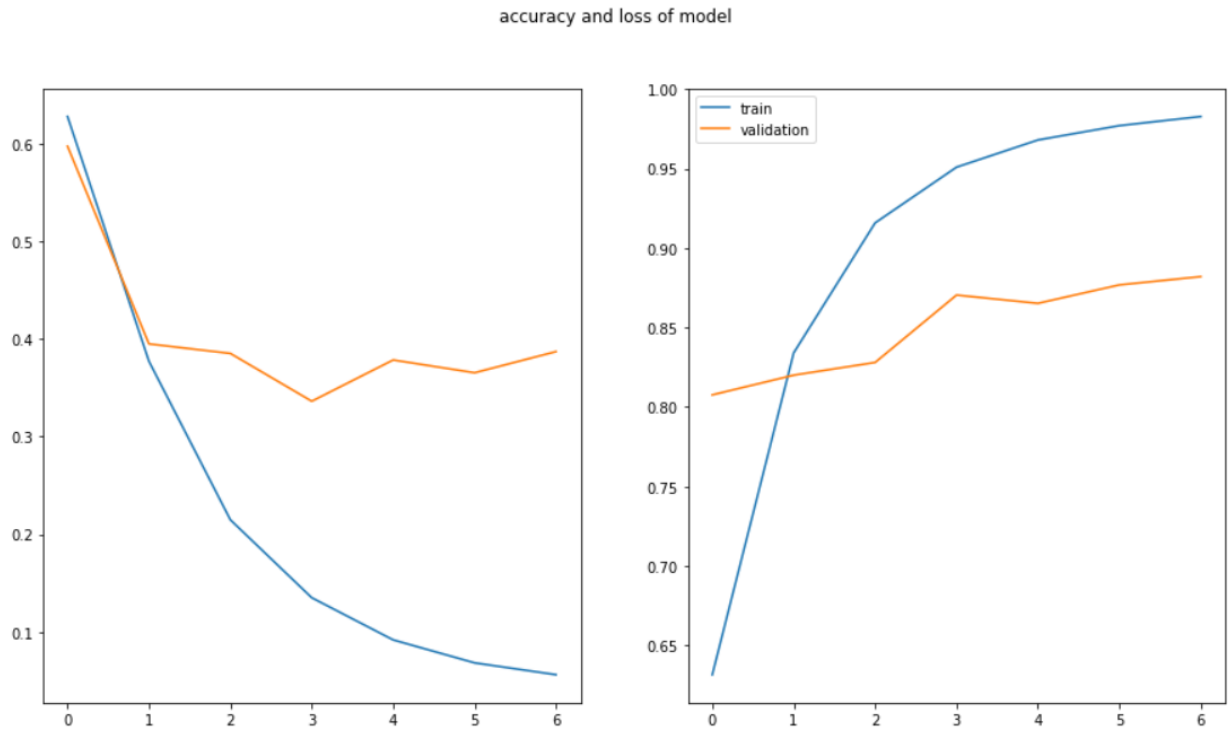
Figure ۲۱- تابع fit

نتایج ایپاک‌های آخر به شرح زیر است.

```
Epoch 5/20
176/176 [==============================] - 32s 182ms/step - loss: 0.0918 - accuracy: 0.9680 - val_loss: 0.3783 - val_accuracy: 0.8652 - lr: 2.2313e-04
Epoch 6/20
176/176 [==============================] - 31s 176ms/step - loss: 0.0683 - accuracy: 0.9770 - val_loss: 0.3652 - val_accuracy: 0.8768 - lr: 1.3534e-04
Epoch 7/20
176/176 [==============================] - 31s 176ms/step - loss: 0.0562 - accuracy: 0.9828 - val_loss: 0.3869 - val_accuracy: 0.8820 - lr: 8.2085e-05
```

Figure ۲۲- نتایج آموزش

نمودار خطا و دقت به شرح زیر است.



ارزیابی داده‌های تست

همانطور که در تصویر پایین مشخص است این شبکه دقت 86.24 درصد را در داده‌های تست کسب کرده‌است.

```
score = model1.evaluate(X_test,test_labels)
```

```
782/782 [==============================] - 13s 16ms/step - loss: 0.4221 - accuracy: 0.8624
```

Figure ۲۳- نتیجه ارزیابی داده‌های تست

سوال۴

معماری شبکه

در این بخش تمامی معماری کاملا شبیه سوال قبل است یعنی در ابتدا یک لایه embedding با خروجی ۳۲و سپس یک لایه GRUقرار داده شده‌است که تعداد خروجی‌های آن ۱۶ است و سپس یک لایه Dense با یک نورون خروجی قرار داده شده است. در انتها نیز یک activation، sigmoid قرار داده شده است

کد های مربوط به این بخش به شرح زیر است.

```
model3 = Sequential()
model3.add(Embedding(num_words,32,input_length =len(X_train[0])))
model3.add(GRU(16,input_shape = (num_words,maxlen), return_sequences=False,activation="relu"))
model3.add(Dense(1))
model3.add(Activation("sigmoid"))

print(model3.summary())
model3.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
```

Figure ۲۴- معماری شبکه با GRU

آموزش شبکه

شبکه را به ۲۰ ایپاک و تابع زیان binary_crossentropy و بهینه‌سازAdam آموزش داده ایم. همچنین ۱۰ درصد از داده‌های آموزش را به validation اختصاص داده‌ایم.

نتایج ایپاک‌های آخر به شرح زیر است.

```
Epoch 6/20
176/176 [==============================] - 70s 396ms/step - loss: 0.1036 - accuracy: 0.9670 - val_loss: 0.3497 - val_accuracy: 0.8712 - lr: 1.3534e-04
Epoch 7/20
176/176 [==============================] - 70s 395ms/step - loss: 0.0939 - accuracy: 0.9712 - val_loss: 0.3738 - val_accuracy: 0.8752 - lr: 8.2085e-05
```

Figure ۲۵-نتایج شبکه با GRU

ارزیابی داده‌های تست

همانطور که در تصویر پایین مشخص است این شبکه دقت 86.24 درصد را در داده‌های تست کسب کرده‌است.

```
score = model3.evaluate(X_test,test_labels)
```

```
782/782 [==============================] - 26s 34ms/step - loss: 0.3857 - accuracy: 0.8635
```

Figure ۲۶نتایج داده‌های تست \GRU

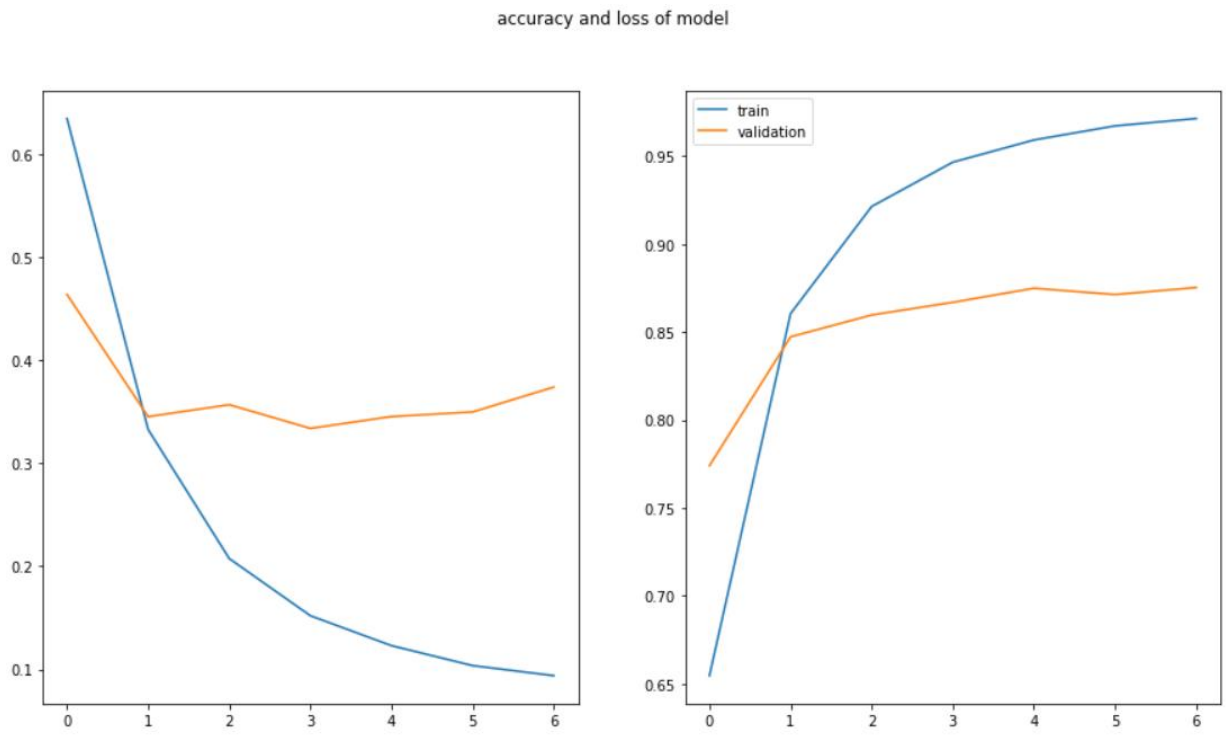


Figure ۲۷نمودار خطا و دقت GRU

سوال۵

معماری شبکه

در این بخش تمامی معماری کاملا شبیه سوال 3 است یعنی در ابتدا یک لایه embedding با خروجی ۳۲و سپس یک لایه LSTM قرار داده شده‌است که تعداد خروجی‌های آن ۱۶ است و سپس یک لایه Dense با یک نورون خروجی قرار داده شده است. در انتها نیز یک activation، sigmoid قرار داده شده است

کد های مربوط به این بخش به شرح زیر است.

```
model4 = Sequential()
model4.add(Embedding(num_words,32,input_length =len(X_train[0])))
model4.add(LSTM(16,input_shape = (num_words,maxlen), return_sequences=False,activation="relu"))
model4.add(Dense(1))
model4.add(Activation("sigmoid"))
print(model4.summary())
model4.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
```

Figure ۲۸-معماری شبکه با LSTM

آموزش شبکه

شبکه را به ۲۰ ایپاک و تابع زیان binary_crossentropy و بهینه‌سازAdam آموزش داده ایم. همچنین ۱۰ درصد از داده‌های آموزش را به validation اختصاص داده‌ایم.

نتایج ایپاک‌های آخر به شرح زیر است.

```
history4 = model4.fit(X_train,train_labels, validation_split = 0.1 ,epochs =20 ,batch_size=128, callbacks=[callback1, callback2, callback3])
```

```
Epoch 1/20
176/176 [==============================] - 34s 170ms/step - loss: 13.2636 - accuracy: 0.6811 - val_loss: 0.5023 - val_accuracy: 0.7904 - lr: 0.0010
Epoch 2/20
176/176 [==============================] - 27s 156ms/step - loss: 6.6295 - accuracy: 0.7776 - val_loss: 0.5444 - val_accuracy: 0.7712 - lr: 0.0010
Epoch 3/20
176/176 [==============================] - 27s 155ms/step - loss: 0.5249 - accuracy: 0.8161 - val_loss: 0.5314 - val_accuracy: 0.7676 - lr: 6.0653e-04
Epoch 4/20
176/176 [==============================] - 27s 156ms/step - loss: 0.4925 - accuracy: 0.8201 - val_loss: 0.5158 - val_accuracy: 0.7748 - lr: 3.6788e-04
```

Figure ۲۹-نتایج داده‌های تست LSTM

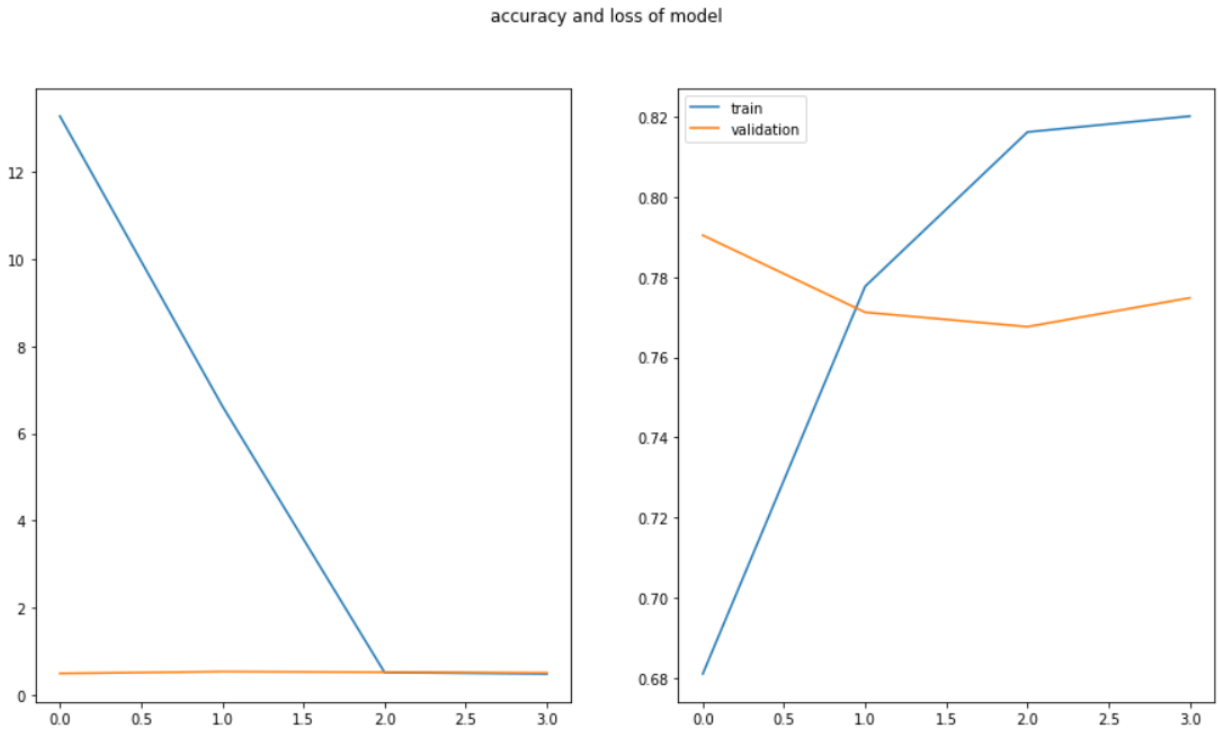


Figure ۳۰-نمودار خطا و دقت LSTM

رزیابی داده‌های تست

همانطور که در تصویر پایین مشخص است این شبکه دقت۷۸.۱۲ درصد را در داده‌های تست کسب کرده‌است.

```
score = model4.evaluate(X_test,test_labels)
```

```
782/782 [==============================] - 11s 14ms/step - loss: 0.5141 - accuracy: 0.7812
```

Figure ۳۱- نتایج داده‌های تست LSTM

سوال۶

معماری شبکه

در این بخش تمامی معماری کاملا شبیه سوال 3 است یعنی در ابتدا یک لایه embedding با خروجی 128و سپس یک لایه bidirectional LSTM قرار داده شده‌است که تعداد خروجی‌های آن 64 است و سپس یک لایه Dense با یک نورون خروجی قرار داده شده است. در انتها نیز یک activation، sigmoid قرار داده شده است

کد های مربوط به این بخش به شرح زیر است.

```
model5 = Sequential()
model5.add(Embedding(num_words,128,input_length =len(X_train[0])))
model5.add(Bidirectional(LSTM(64,input_shape = (num_words,maxlen), return_sequences=False,activation="relu")))
model5.add(Dense(1))
model5.add(Activation("sigmoid"))

print(model5.summary())
model5.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
```

Figure ۳۲- معماری شبکه

آموزش شبکه

شبکه را به ۲۰ ایپاک و تابع زیان binary_crossentropy و بهینه‌سازAdam آموزش داده ایم. همچنین ۱۰ درصد از داده‌های آموزش را به validation اختصاص داده‌ایم.

نتایج ایپاک‌های آخر به شرح زیر است.

```
Epoch 16/20
176/176 [==============================] - 115s 653ms/step - loss: 0.5910 - accuracy: 0.8368 - val_loss: 0.6267 - val_accuracy: 0.7360 - lr: 9.1188e-07
Epoch 17/20
176/176 [==============================] - 113s 645ms/step - loss: 0.5910 - accuracy: 0.8369 - val_loss: 0.6267 - val_accuracy: 0.7356 - lr: 5.5308e-07
Epoch 18/20
176/176 [==============================] - 115s 652ms/step - loss: 0.5909 - accuracy: 0.8369 - val_loss: 0.6267 - val_accuracy: 0.7356 - lr: 3.3546e-07
Epoch 19/20
176/176 [==============================] - 114s 650ms/step - loss: 0.5909 - accuracy: 0.8369 - val_loss: 0.6267 - val_accuracy: 0.7356 - lr: 2.0347e-07
Epoch 20/20
176/176 [==============================] - 116s 660ms/step - loss: 0.5909 - accuracy: 0.8369 - val_loss: 0.6267 - val_accuracy: 0.7356 - lr: 1.2341e-07
```

Figure ۳۳-نتایج داده‌های تست Bidirectional LSTM

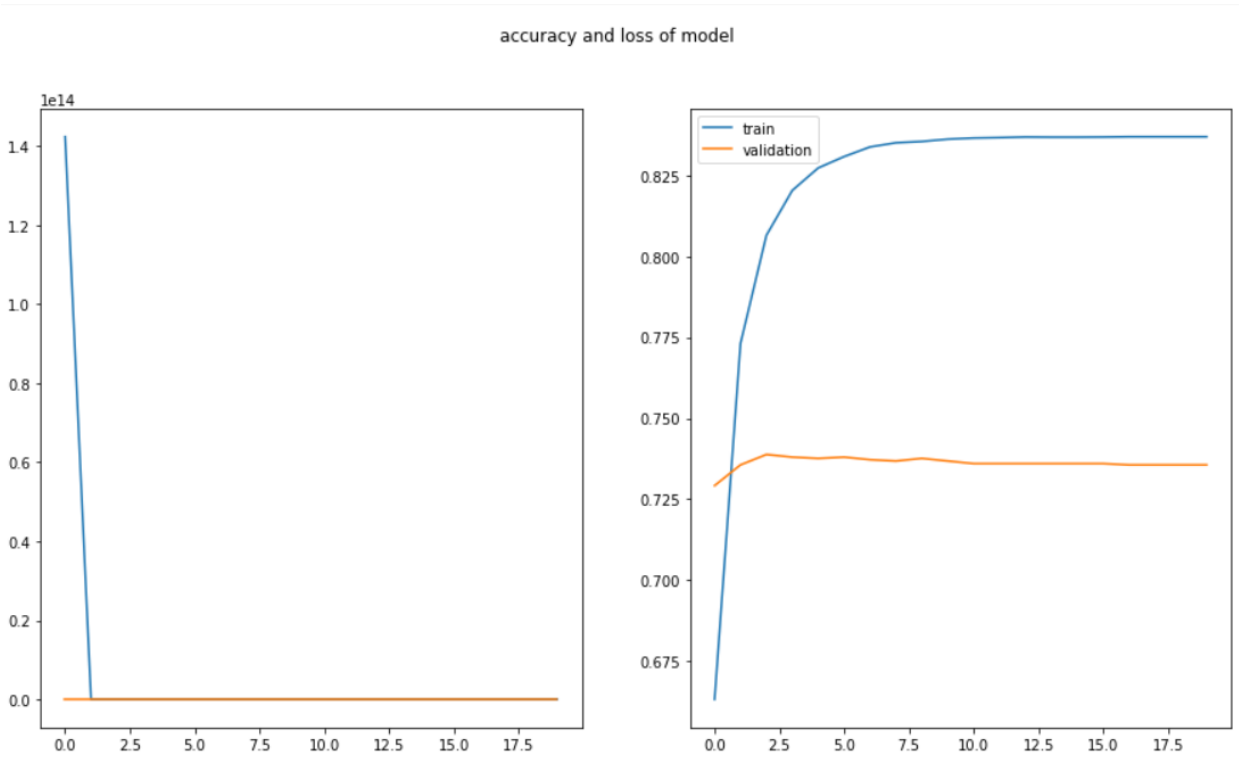


Figure ۳۴-نمودار خطا و دقت

ارزیابی داده‌های تست

همانطور که در تصویر پایین مشخص است این شبکه دقت ۷۱.۸۱ درصد را در داده‌های تست کسب کرده‌است.

```
score = model5.evaluate(X_test,test_labels)
```

```
782/782 [==============================] - 39s 49ms/step - loss: 0.6336 - accuracy: 0.7181
```

Figure ۳۵نتایج داده‌های تست

سوال۷

معماری شبکه

در این بخش تمامی معماری کاملا شبیه سوال 6است یعنی در ابتدا یک لایه embedding با خروجی 128و سپس یک لایه bidirectional LSTM قرار داده شده‌است که تعداد خروجی‌های آن 64 است و سپس یک لایه Dense با یک نورون خروجی قرار داده شده است. در انتها نیز یک activation، sigmoid قرار داده شده است. تنها تفاوت صورت گرفته آن است که این معماری به صورت functional پیاده سازی شده است یعنی به جای اینکه از ساختار sequential استفاده کنیم ورودی ها را گرفته و خروجی هر لایه را محاسبه می‌کنیم.

کد های مربوط به این بخش به شرح زیر است.

```
inputs = keras.Input(shape=(None,), dtype="int32")
x = Embedding(num_words, 128)(inputs)
x = Bidirectional(LSTM(64, return_sequences=False))(x)
#x = Bidirectional(LSTM(64))(x)
outputs = Dense(1, activation="sigmoid")(x)
model6 = keras.Model(inputs, outputs)
model6.summary()
model6.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])
history6 = model6.fit(X_train,train_labels, validation_split = 0.1 ,epochs =20 ,batch_size=128, callbacks=[callback1, callback2, callback3])
```

Figure ۳۶- معماری شبکه

آموزش شبکه

شبکه را به ۲۰ ایپاک و تابع زیان binary_crossentropy و بهینه‌سازAdam آموزش داده ایم. همچنین ۱۰ درصد از داده‌های آموزش را به validation اختصاص داده‌ایم.

نتایج ایپاک‌های آخر به شرح زیر است. همانطور که میبینید شبکه در ایپاک چهارم متوقف شده و به نتیجه بسیار خوبی دست یافته است.

```
Epoch 1/20
176/176 [==============================] - 13s 38ms/step - loss: 0.4387 - accuracy: 0.7810 - val_loss: 0.3529 - val_accuracy: 0.8440 - lr: 0.0010
Epoch 2/20
176/176 [==============================] - 3s 15ms/step - loss: 0.2533 - accuracy: 0.8974 - val_loss: 0.3560 - val_accuracy: 0.8456 - lr: 0.0010
Epoch 3/20
176/176 [==============================] - 3s 15ms/step - loss: 0.1664 - accuracy: 0.9392 - val_loss: 0.4206 - val_accuracy: 0.8336 - lr: 6.0653e-04
Epoch 4/20
176/176 [==============================] - 3s 14ms/step - loss: 0.1078 - accuracy: 0.9667 - val_loss: 0.5002 - val_accuracy: 0.8312 - lr: 3.6788e-04
```

Figure ۳۷- نتایج آموزش به صورت functional

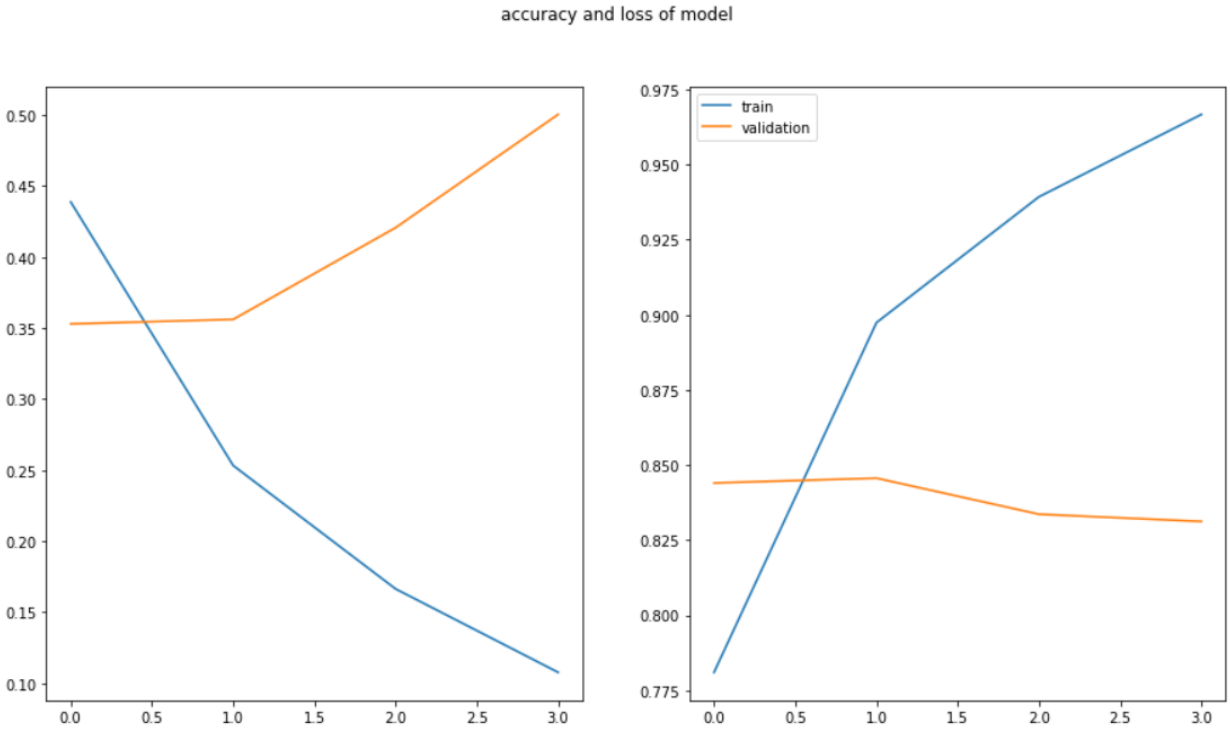


Figure ۳۸نمودار خطا و دقت شبکه functional

ارزیابی داده‌های تست

همانطور که در تصویر پایین مشخص است این شبکه دقت 83.86 درصد را در داده‌های تست کسب کرده‌است.

```
score = model6.evaluate(X_test,test_labels)
```

```
782/782 [==============================] - 4s 6ms/step - loss: 0.4738 - accuracy: 0.8386
```