به نام خدا

تمرین پنجم یادگیری عمیق

غزل زمانی‌نژاد

۹۷۵۲۲۱۶۶

1. **الف) در landmark detection، باید مختصات نقاط مهم را پیدا کنیم. یعنی باید برای هر نقطه (x, y) را بیابیم. با توجه به اینکه 5 نقطه مهم داریم، پس در لایه آخر به 10 = 2 * 5 نورون نیاز داریم (فرض میکنیم در تمامی تصاویر هر 5 نقطه موجود هستند. در غیر این صورت به 5 نورون دیگر نیاز داریم تا پیش بینی کند هریک از نقاط در تصویر موجود است یا خیر). چون مختصات نقاط دارای مقادیر پیوسته هستند پس در اینجا یک تسک regression داریم. از تابع فعالسازی خطی استفاده میکنیم (در صورت استفاده از سایر توابع، خروجی به بازه خاصی محدود میشود در صورتی که مختصات لزوما در بازه مشخص نیست). با فرض توزیع نرمال خطا، از تابع ضرر mean squared error استفاده میکنیم. این تابع فاصله میان نقاط با خط رگرسیون را محاسبه میکند و میتواند به خوبی میزان نزدیکی خط پیش بینی شده با نقاط اصلی را بیابد.**
ب) تابع ضرر: از میانگین فاصله اقلیدسی نقاط واقعی با نقاط پیش بینی شده در هر mini-batch استفاده شده است (هر نقطه ای که توسط انسان قابل تعیین نبوده، در محاسبات مربوط به loss قرار نگرفته است). این تابع عملکردی مشابه MSE دارد.
تابع فعال سازی: در لایه آخر از یک تابع سیگموید استفاده شده است. بدین صورت که طول و عرض کل تصویر را به بازه (0,1) مپ میکنیم. بدین شکل مختصات تمامی نقاط صفحه بین 0 و 1 قرار میگیرند. خروجی تابع سیگموید نیز بین 0 تا 1 است. برای پیش بینی 76 نقطه landmark، 152 نورون خروجی داریم.

2. **در لایه آخر از یک لایه Dense با 4 نورون خروجی استفاده میکنیم زیرا قیمت موبایل مقداری از 1 تا 3 است. سپس از تابع فعالسازی softmax استفاده میکنیم. زیرا این تابع میتواند در یک مسئله چندکلاسه**

احتمال هر یک از نورون های خروجی را به خوبی محاسبه کند و بیشترین احتمال را به عنوان تخمین شبکه برگرداند.

برای تابع ضرر از categorical crossentropy **استفاده میکنیم. این تابع تعمیم** binary crossentropy **است و برای مسائل چند کلاسه که هر داده تنها متعلق به یکی از کلاس های ورودی است عملکرد مناسبی دارد. همچنین در صورت استفاده از رویکرد** maximum likelihood estimation**، برای این نوع مسئله به تابع** crossentropy **میرسیم (و نه** mse**).**

برای بهینه ساز از SGD **استفاده میکنیم. زیرا نسبت به** Adam **بهتر میتواند برای داده های جدیدی که بر روی آنها آموزش ندیده تعمیم دهد. نرخ یادگیری را مقدار کوچکی قرار میدهیم تا بهتر آموزش ببیند.**

سپس مدل را کامپایل کرده و آن را آموزش میدهیم.

```
model = Sequential()
model.add(Dense(300, input_dim=x_train.shape[1], activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(200, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(50, activation='relu'))
### YOU HAVE TO MAKE YOUR CHANGES HERE !!!
model.add(Dense(4, activation='softmax')) # softmax activation for last layer

opt = SGD(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    epochs=300,
                    batch_size=32,
                    verbose=1,
                    validation_data=(x_test,y_test))

# evaluate the keras model
_, accuracy = model.evaluate(x_train, y_train)
print('total train Accuracy: %.2f' % (accuracy*100))

_, accuracy2 = model.evaluate(x_test, y_test)
print('total validation Accuracy: %.2f' % (accuracy2*100))
```

در تصویر زیر نتیجه چند epoch **پایانی به همراه دقت کل مشاهده میشود:**

```
Epoch 296/300
50/50 [==============================] - 0s 3ms/step - loss: 0.9767 - accuracy: 0.5312 - val_loss: 0.9268 - val_accuracy: 0.5525
Epoch 297/300
50/50 [==============================] - 0s 4ms/step - loss: 0.9854 - accuracy: 0.5350 - val_loss: 0.9275 - val_accuracy: 0.5750
Epoch 298/300
50/50 [==============================] - 0s 4ms/step - loss: 0.9716 - accuracy: 0.5281 - val_loss: 0.9344 - val_accuracy: 0.5450
Epoch 299/300
50/50 [==============================] - 0s 3ms/step - loss: 1.0052 - accuracy: 0.5156 - val_loss: 0.9272 - val_accuracy: 0.5675
Epoch 300/300
50/50 [==============================] - 0s 3ms/step - loss: 0.9701 - accuracy: 0.5337 - val_loss: 0.9272 - val_accuracy: 0.5525
50/50 [==============================] - 0s 2ms/step - loss: 0.9022 - accuracy: 0.5625
total train Accuracy: 56.25
13/13 [==============================] - 0s 2ms/step - loss: 0.9272 - accuracy: 0.5525
total validation Accuracy: 55.25
```

دقت داده های آموزشی تقریبا 56 درصد است که مقدار مناسبی نیست. شبکه هنوز به خوبی آموزش ندیده و در واقع underfit شده است. در اینجا با مسئله بایاس مواجه شده ایم. میتوانیم برای رفع کردن آن شبکه را بزرگتر کنیم و یا در epochهای بیشتری به آن آموزش دهیم. اما دقت داده اعتبارسنجی در مقایسه با داده آموزشی مناسب است و شبکه توانسته بر روی داده هایی که از قبل آنها را ندیده، تعمیم دهد.

3. ابتدا دیتاست رویترز را با 1000 کلمه پرتکرار آن load میکنیم.

```
from keras.datasets import reuters

max_words = 1000
(train_data, train_labels), (test_data, test_labels) = reuters.load_data(num_words=max_words)
```

سپس از تابع encode_sequence استفاده میکنیم تا ورودی ها را به شکل خواسته شده دربیاوریم. ابتدا یک بردار صفر با تعداد سطر sequenceها و تعداد ستون 1000 میسازیم. سپس روی تمامی داده ها iterate میکنیم. اگر هر یک از 1000 کلمه پرتکرار در آن نمونه از داده موجود بود خانه متناظر با آن ایندکس را 1 میکنیم. و برای داده آزمون از متد to_categorical استفاده میکنیم.

```
import numpy as np
from tensorflow import keras

def encode_sequence(sequences, d=max_words):
  res = np.zeros((len(sequences), d))
  for i, seq in enumerate(sequences):
    res[i, seq] = 1.
  return res

x_train = encode_sequence(train_data)
y_train = encode_sequence(train_labels)
y_train = keras.utils.to_categorical(train_labels)
y_test = keras.utils.to_categorical(test_labels)

print("train input shape", x_train.shape)
print("train labels shape", y_train.shape)
```

```
train input shape (8982, 1000)
train labels shape (8982, 46)
```

در سلول سوم از callback تنسوربورد استفاده میکنیم تا بتوانیم نمودار تابع دقت و ضرر مدل ها را به خوبی نمایش دهیم.

```
%load_ext tensorboard

import tensorflow as tf
import datetime, os

logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
```

در 5 سلول بعدی، شبکه خواسته شده به همراه تابع فعالسازی و تابع ضرر را پیاده سازی میکنیم:

مدل اول:

```
model1 = keras.models.Sequential()
model1.add(keras.layers.Dense(64, activation='relu', input_shape=(max_words,)))
model1.add(keras.layers.Dense(64, activation='relu'))
model1.add(keras.layers.Dense(46, activation='sigmoid'))


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

h1= model1.fit(x_train, y_train,
               validation_data=(x_test, y_test),
               epochs=20,
               batch_size=512,
               callbacks=[tensorboard_callback])
```

مدل دوم:

```
model2 = keras.models.Sequential()
model2.add(keras.layers.Dense(64, activation='relu', input_shape=(max_words,)))
model2.add(keras.layers.Dense(64, activation='relu'))
model2.add(keras.layers.Dense(46, activation='softmax'))


model2.compile(optimizer='adam',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy'])

h2 = model2.fit(x_train, y_train,
               validation_data=(x_test, y_test),
               epochs=20,
               batch_size=512,
               callbacks=[tensorboard_callback])
```

مدل سوم:

```
model3 = keras.models.Sequential()
model3.add(keras.layers.Dense(64, activation='relu', input_shape=(max_words,)))
model3.add(keras.layers.Dense(64, activation='relu'))
model3.add(keras.layers.Dense(46, activation='sigmoid'))


model3.compile(optimizer='adam',
               loss = 'mse',
               metrics = ['accuracy'])

h3 = model3.fit(x_train, y_train,
               validation_data=(x_test, y_test),
               epochs=20,
               batch_size=512,
               callbacks=[tensorboard_callback])
```

مدل چهارم:

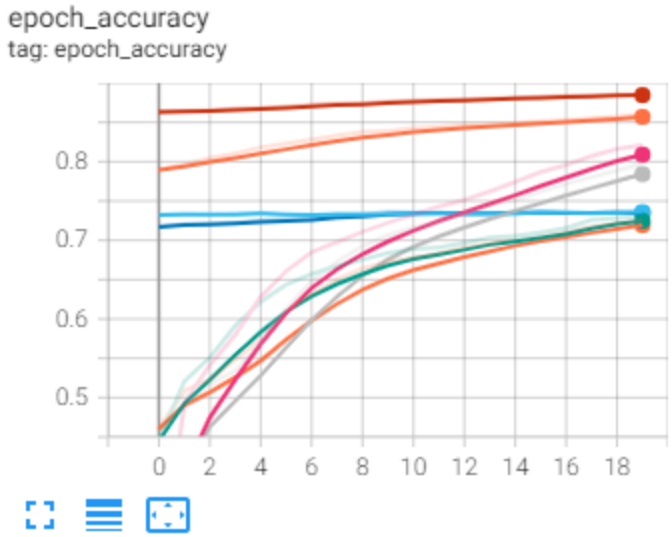
```
model4 = keras.models.Sequential()
model4.add(keras.layers.Dense(64, activation='relu', input_shape=(max_words,)))
model4.add(keras.layers.Dense(64, activation='relu'))
model4.add(keras.layers.Dense(46, activation='softmax'))


model4.compile(optimizer='adam',
               loss = 'mse',
               metrics = ['accuracy'])

h4 = model4.fit(x_train, y_train,
               validation_data=(x_test, y_test),
               epochs=20,
               batch_size=512,
               callbacks=[tensorboard_callback])
```

مدل پنجم:

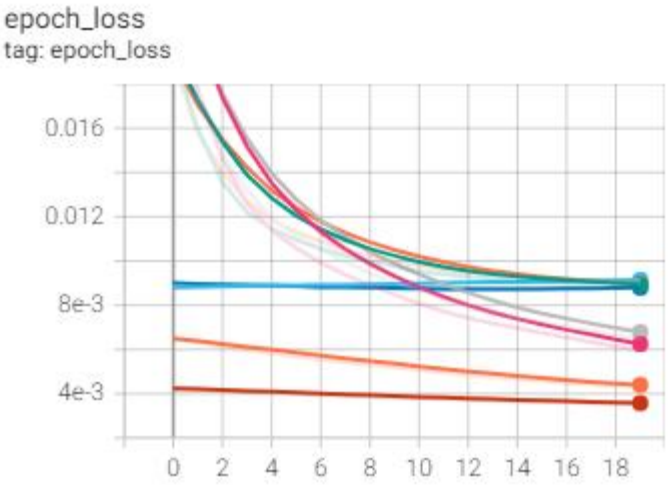
```
model5 = keras.models.Sequential()
model5.add(keras.layers.Dense(64, activation='relu', input_shape=(max_words,)))
model5.add(keras.layers.Dense(64, activation='relu'))
model5.add(keras.layers.Dense(46, activation=None))

model5.compile(optimizer='adam',
               loss = 'mse',
               metrics = ['accuracy'])

h5 = model5.fit(x_train, y_train,
               validation_data=(x_test, y_test),
               epochs=20,
               batch_size=512,
               callbacks=[tensorboard_callback])
```

در سلول انتهایی اطلاعاتی که در تنسوربورد ذخیره کردیم را چاپ میکنیم:

epoch_accuracy



epoch_loss



برای اینکه بتوانیم مدل ها را بهتر مقایسه کنیم، در تمامی آنها از بهینه ساز Adam استفاده کرده ایم.

توضیح کلی:

- **این مسئله دسته بندی یک مسئله طبقه بندی 46 کلاسه است و همانطور که در نمودار دقت مشاهده میشود، هیچ کدام از مدل ها نتوانسته به دقت 100 درصد برسد.**
- **به علت سخت بودن مسئله، حتی تابع ضرر cross entropy هم نتوانسته مقدار ضرر را به 0 برساند.**
- **با استفاده کردن از تابع mse، دقت به خوبی دقت در نمودار cross entropy نشده است.**

برای پیدا کردن بهترین و بدترین عملکرد، میزان دقت شبکه ها در داده اعتبارسنجی را باهم مقایسه میکنیم. در این قسمت بهترین عملکرد مربوط به مدل دوم (softmax + categorical crossentropy) است. تابع ضرر caregorical crossentropy مطابق انتظار توانسته در یک مسئله چندکلاسه با یک لیبل بهترین میزان دقت را بدست بیاورد. زیرا با استفاده از این تابع ضرر، شبکه در مقابل خطای هرچند کوچک در مقایسه با تابع ضرر mse بیشتر جریمه میشود. همچنین تابع فعالسازی سافت مکس، برای هر داده احتمال اینکه متعلق به هر یک از کلاس ها باشد را بهتر پیش بینی کرده است (علت برتری این مدل نسبت به مدل اول: تابع ضرر categorical crossentropy تعمیمی از تابع ضرر binary crossentropy، و تابع فعالسازی softmax تعمیمی از تابع فعالسازی sigmoid است که این دو تابع برای مسئله چندکلاسه بهتر عمل میکنند. دو تابع دیگر در مسائل باینری عملکرد خوبی دارند).

ضعیف ترین عملکرد مربوط به مدل سوم (sigmoid + mse) است. تابع میانگین مربعات خطا نمیتواند تابع ضرر مناسبی برای یک مسئله 46 کلاسه باشد زیرا بیشترین میزان خطای شبکه 1 است و شبکه نمیتواند به خوبی تحت آموزش قرار گیرد (برای آموزش به epochهای بیشتری نسبت به crossentropy نیاز دارد).

منابع استفاده‌شده:

https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/tensorboard_in_notebooks.ipynb#scrollTo=KBHp6M_zgjp4