IMDB REVIEWS

# Import các thư viện cần thiết

from tensorflow.keras import Sequential

from tensorflow.keras import layers

from tensorflow.keras import losses

import tensorflow as tf

import tensorflow\_hub as hub

import tensorflow\_datasets as tfds

# Tải và load tập dữ liệu

Tải tập dữ liệu bằng thư viện tfds và load tập dữ liệu với 3 phần như sau:

|  |  |  |
| --- | --- | --- |
| train | validation | test |
| 60% của train trong dataset | Phần còn lại của train trong dataset | Tập test trong dataset |

train\_data, validation\_data, test\_data = tfds.load(

name="imdb\_reviews",

split=('train[:60%]', 'train[60%:]', 'test'),

as\_supervised=True

)

In ra 10 phần tử trong tập train:

train\_examples\_batch, train\_labels\_batch = next(iter(train\_data.batch(10)))

print(train\_examples\_batch)

print(train\_labels\_batch)

# Vector hoá các bình luận bằng mô hình “gnews-swivel-20dim”

Lấy mô hình từ <https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1> và load bằng thư viện tensorflow\_hub

embedding = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1"

hub\_layer = hub.KerasLayer(

embedding,

input\_shape=[],

dtype=tf.string,

trainable=True

)

Vector hoá và in ra 3 phần tử trong tập train

print(hub\_layer(train\_examples\_batch[:3]))

tf.Tensor(

[[ 1.765786 -3.882232 3.9134233 -1.5557289 -3.3362343 -1.7357955

-1.9954445 1.2989551 5.081598 -1.1041286 -2.0503852 -0.72675157

-0.65675956 0.24436149 -3.7208383 2.0954835 2.2969332 -2.0689783

-2.9489717 -1.1315987 ]

[ 1.8804485 -2.5852382 3.4066997 1.0982676 -4.056685 -4.891284

-2.785554 1.3874227 3.8476458 -0.9256538 -1.896706 1.2113281

0.11474707 0.76209456 -4.8791065 2.906149 4.7087674 -2.3652055

-3.5015898 -1.6390051 ]

[ 0.71152234 -0.6353217 1.7385626 -1.1168286 -0.5451594 -1.1808156

0.09504455 1.4653089 0.66059524 0.79308075 -2.2268345 0.07446612

-1.4075904 -0.70645386 -1.907037 1.4419787 1.9551861 -0.42660055

-2.8022065 0.43727064]], shape=(3, 20), dtype=float32)

# Xây dựng mô hình

Tiến hành xây dựng mô hình gồm 3 tầng:

* Tầng vector hoá đầu vào
* Tầng kết nối đặc (có 16 nút và sử dụng hàm kích hoạt ReLU)
* Tầng dự đoán kết quả

model = Sequential([

hub\_layer,

layers.Dense(16, activation="relu"),

layers.Dense(1)

])

model.summary()

Thông tin về mô hình

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

keras\_layer (KerasLayer) (None, 20) 400020

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 16) 336

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 17

=================================================================

Total params: 400,373

Trainable params: 400,373

Non-trainable params: 0

Tạo mô hình:

* Sử dụng bộ tối ưu Adam
* Sử dụng hàm loss là BinaryCrossentropy
* Phương thức đánh giá là độ chính xác

model.compile(

optimizer="adam",

loss=losses.BinaryCrossentropy(from\_logits=True),

metrics=["accuracy"]

)

# Huấn luyện mô hình

Huấn luyện mô hình 20 lần trên tập train và sử dụng tập validation làm tập kiểm thử

history = model.fit(

train\_data.shuffle(100000).batch(512),

epochs=20,

validation\_data=validation\_data.batch(512)

)

Ta thấy được sau mỗi lần huấn luyện thì trên tập validation giảm chỉ số loss và cải thiện độ chính xác

Epoch 1/20

30/30 [==============================] - 3s 54ms/step - loss: 0.6741 - accuracy: 0.5455 - val\_loss: 0.6496 - val\_accuracy: 0.5809

Epoch 2/20

30/30 [==============================] - 1s 47ms/step - loss: 0.6247 - accuracy: 0.6028 - val\_loss: 0.6110 - val\_accuracy: 0.6262

Epoch 3/20

30/30 [==============================] - 1s 46ms/step - loss: 0.5877 - accuracy: 0.6473 - val\_loss: 0.5776 - val\_accuracy: 0.6648

Epoch 4/20

30/30 [==============================] - 1s 46ms/step - loss: 0.5506 - accuracy: 0.6916 - val\_loss: 0.5423 - val\_accuracy: 0.7088

Epoch 5/20

30/30 [==============================] - 1s 47ms/step - loss: 0.5103 - accuracy: 0.7335 - val\_loss: 0.5050 - val\_accuracy: 0.7403

Epoch 6/20

30/30 [==============================] - 1s 46ms/step - loss: 0.4701 - accuracy: 0.7657 - val\_loss: 0.4719 - val\_accuracy: 0.7664

Epoch 7/20

30/30 [==============================] - 1s 46ms/step - loss: 0.4319 - accuracy: 0.7909 - val\_loss: 0.4430 - val\_accuracy: 0.7808

Epoch 8/20

30/30 [==============================] - 1s 47ms/step - loss: 0.3975 - accuracy: 0.8148 - val\_loss: 0.4178 - val\_accuracy: 0.7995

Epoch 9/20

30/30 [==============================] - 1s 46ms/step - loss: 0.3664 - accuracy: 0.8328 - val\_loss: 0.3963 - val\_accuracy: 0.8137

Epoch 10/20

30/30 [==============================] - 1s 47ms/step - loss: 0.3382 - accuracy: 0.8507 - val\_loss: 0.3780 - val\_accuracy: 0.8227

Epoch 11/20

30/30 [==============================] - 1s 47ms/step - loss: 0.3126 - accuracy: 0.8638 - val\_loss: 0.3620 - val\_accuracy: 0.8332

Epoch 12/20

30/30 [==============================] - 1s 46ms/step - loss: 0.2900 - accuracy: 0.8767 - val\_loss: 0.3493 - val\_accuracy: 0.8437

Epoch 13/20

30/30 [==============================] - 1s 47ms/step - loss: 0.2694 - accuracy: 0.8868 - val\_loss: 0.3399 - val\_accuracy: 0.8497

Epoch 14/20

30/30 [==============================] - 1s 46ms/step - loss: 0.2517 - accuracy: 0.8963 - val\_loss: 0.3323 - val\_accuracy: 0.8474

Epoch 15/20

30/30 [==============================] - 1s 46ms/step - loss: 0.2352 - accuracy: 0.9055 - val\_loss: 0.3252 - val\_accuracy: 0.8512

Epoch 16/20

30/30 [==============================] - 1s 45ms/step - loss: 0.2206 - accuracy: 0.9123 - val\_loss: 0.3263 - val\_accuracy: 0.8636

Epoch 17/20

30/30 [==============================] - 1s 47ms/step - loss: 0.2075 - accuracy: 0.9212 - val\_loss: 0.3168 - val\_accuracy: 0.8614

Epoch 18/20

30/30 [==============================] - 1s 48ms/step - loss: 0.1940 - accuracy: 0.9266 - val\_loss: 0.3146 - val\_accuracy: 0.8586

Epoch 19/20

30/30 [==============================] - 2s 50ms/step - loss: 0.1829 - accuracy: 0.9317 – v al\_loss: 0.3133 - val\_accuracy: 0.8604

Epoch 20/20

30/30 [==============================] - 1s 47ms/step - loss: 0.1716 - accuracy: 0.9375 - val\_loss: 0.3126 - val\_accuracy: 0.8610

# Đánh giá mô hình

Sau khi huấn luyện xong, ta tiến hành đánh giá mô hình trên tập test

results = model.evaluate(test\_data.batch(512))

for name, value in zip(model.metrics\_names, results):

print("%s: %.3f" % (name, value))

Thu được kết quả

loss: 0.328

accuracy: 0.853

# Source code

from tensorflow.keras import Sequential

from tensorflow.keras import layers

from tensorflow.keras import losses

import tensorflow as tf

import tensorflow\_hub as hub

import tensorflow\_datasets as tfds

def main():

train\_data, validation\_data, test\_data = tfds.load(

name="imdb\_reviews",

split=('train[:60%]', 'train[60%:]', 'test'),

as\_supervised=True

)

train\_examples\_batch, train\_labels\_batch = next(iter(train\_data.batch(10)))

print(train\_examples\_batch)

print(train\_labels\_batch)

embedding = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1"

hub\_layer = hub.KerasLayer(

embedding,

input\_shape=[],

dtype=tf.string,

trainable=True

)

print(hub\_layer(train\_examples\_batch[:3]))

model = Sequential([

hub\_layer,

layers.Dense(16, activation="relu"),

layers.Dense(1)

])

model.summary()

model.compile(

optimizer="adam",

loss=losses.BinaryCrossentropy(from\_logits=True),

metrics=["accuracy"]

)

history = model.fit(

train\_data.shuffle(100000).batch(512),

epochs=20,

validation\_data=validation\_data.batch(512)

)

results = model.evaluate(test\_data.batch(512))

for name, value in zip(model.metrics\_names, results):

print("%s: %.3f" % (name, value))

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Hyperparameter tuning

Sau đây là bảng thống kê đánh giá của những lần thay đổi thông số mô hình trên tập test

|  |  |  |
| --- | --- | --- |
| Thông tin sửa đổi | Độ chính xác | Loss |
| 1 layer dense 16 nút với hàm kích hoạt ReLU | 0.853 | 0.328 |
| 2 layer dense 16 nút với hàm kích hoạt ReLU | 0.850 | 0.351 |
| 1 layer dense 64 nút với hàm kích hoạt ReLU | 0.859 | 0.332 |
| 2 layer dense 64 nút với hàm kích hoạt ReLU | 0.851 | 0.426 |
| 1 layer dense 8 nút với hàm kích hoạt ReLU | 0.854 | 0.324 |
| 2 layer dense 8 nút với hàm kích hoạt ReLU | 0.855 | 0.333 |

Kết luận: thay đổi kiến trúc mạng không ảnh hưởng lớn đến độ chính xác và chỉ số loss