2023数据挖掘大作业

张秋雨（1120193285）、刘 倞（1120193289）、许阳霄（1120193405）、杨佳瑶（1120193572）、刘 冉（1120192461）

In [ ]:

```
import tensorflow as tf
import pandas as pd
import numpy as np
import keras
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping,ReduceLROnPlateau
from keras.utils import to_categorical
from tensorflow.keras.layers import Conv1D,InputLayer,Dense,Dropout,Flatten,MaxPooling1D,BatchNo
from tensorflow.keras.models import Sequential,Model
from sklearn.model_selection import KFold
import warnings
warnings.filterwarnings("ignore")
```

In [4]:

```
train = pd.read_csv('./data/train.csv',encoding='utf-8')
test=pd.read_csv('./data/testA.csv',encoding='utf-8')
train.head()
```

Out[4]:

	id	heartbeat_signals	label
0	0	0.9912297987616655,0.9435330436439665,0.764677...	0.0
1	1	0.9714822034884503,0.9289687459588268,0.572932...	0.0
2	2	1.0,0.9591487564065292,0.7013782792997189,0.23...	2.0
3	3	0.9757952826275774,0.9340884687738161,0.659636...	0.0
4	4	0.0,0.055816398940721094,0.26129357194994196,0...	2.0

In [3]:

```
train.tail()
```

Out[3]:

	id	heartbeat_signals	label
119995	119995	1.0,0.8330283177934747,0.6340472606311671,0.63...	0.0
119996	119996	1.0,0.8259705825857048,0.4521053488322387,0.08...	0.0
119997	119997	0.951744840752379,0.9162611283848351,0.6675251...	2.0
119998	119998	0.9276692903808186,0.6771898159607004,0.242906...	0.0
119999	119999	0.6653212231837624,0.527064114047737,0.5166625...	0.0

In [5]:

```
#转换数据类型来减小数据占用内存
def reduce_mem_usage(df):
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
```

In [6]:

```
# 简单预处理
train_list = []

for items in train.values:
    train_list.append([items[0]] + [float(i) for i in items[1].split(',')] + [items[2]])

train = pd.DataFrame(np.array(train_list))
train.columns = ['id'] + ['s_'+str(i) for i in range(len(train_list[0])-2)] + ['label']
train = reduce_mem_usage(train)

test_list=[]
for items in test.values:
    test_list.append([items[0]] + [float(i) for i in items[1].split(',')])

test = pd.DataFrame(np.array(test_list))
test.columns = ['id'] + ['s_'+str(i) for i in range(len(test_list[0])-1)]
test = reduce_mem_usage(test)
```

```
Memory usage of dataframe is 157.93 MB
Memory usage after optimization is: 39.67 MB
Decreased by 74.9%
Memory usage of dataframe is 31.43 MB
Memory usage after optimization is: 7.90 MB
Decreased by 74.9%
```

In [7]:

```
Y = train['label']
X = train.drop(['id','label'],axis=1)
test= test.drop(['id'],axis=1)
print(np.shape(X),np.shape(test))
```

```
(100000, 205) (20000, 205)
```

In [8]:

```
def abs_sum(y_pre,y_tru):
    y_pre=np.array(y_pre)
    y_tru=np.array(y_tru)
    loss=sum(sum(abs(y_pre-y_tru)))
    return loss
```

In [9]:

```
X = np.array(X).reshape(-1,205,1)    #因为模型读入数据要求是三维
test = np.array(test).reshape(-1,205,1)
```

In [10]:

```
Y = to_categorical(Y)
```

In [11]:

```
def buildmodel3():  #线上172，用172当预训练再训练168
    inputs=Input(shape=(205,1))
    x1 = Conv1D(32,kernel_size=32, strides=1, padding='SAME',activation='relu')(inputs)
    x2 = Conv1D(32,kernel_size=16, strides=1, padding='SAME',activation='relu')(inputs)
    x3 = Conv1D(32,kernel_size=48, strides=1, padding='SAME',activation='relu')(inputs)
    x4 = Add()([x1,x2,x3])
    x = BatchNormalization()(x4)

    x = Conv1D(64,kernel_size=16, strides=1, padding='SAME',activation='relu')(x4)
    x = Conv1D(128,kernel_size=8, strides=1, padding='SAME',activation='relu')(x)

    x = MaxPool1D(pool_size=4, strides=2, padding='SAME')(x)
    x = Dropout(rate=0.25)(x)
    x = Flatten()(x)
    x = Dense(512,activation='relu')(x)
    x = Dense(1024,activation='relu')(x)
    output = Dense(4,activation='softmax')(x)
    model = Model(inputs=inputs,outputs=output)

    return model
```

In [12]:

```
model = buildmodel3()
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_1 (InputLayer)         [(None, 205, 1)]          0
_________________________________________________________________
conv1d (Conv1D)              (None, 205, 32)           1056      input_1[0][0]
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 205, 32)           544       input_1[0][0]
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 205, 32)           1568      input_1[0][0]
_________________________________________________________________
add (Add)                    (None, 205, 32)           0         conv1d[0][0]
                                                                 conv1d_1[0][0]
                                                                 conv1d_2[0][0]
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 205, 64)           32832     add[0][0]
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 205, 128)          65664     conv1d_3[0][0]
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 103, 128)          0         conv1d_4[0][0]
_________________________________________________________________
dropout (Dropout)            (None, 103, 128)          0         max_pooling1d[0]
[0]
_________________________________________________________________
flatten (Flatten)            (None, 13184)             0         dropout[0][0]
_________________________________________________________________
dense (Dense)                (None, 512)               6750720   flatten[0][0]
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              525312    dense[0][0]
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4100      dense_1[0][0]
=================================================================
Total params: 7,381,796
Trainable params: 7,381,796
Non-trainable params: 0
_________________________________________________________________
```

In [31]:

```
def cv_model(train_x, train_y, test):
    folds = 10
    seeds = [600]
    tests = []
    cv_scores = []

    for seed in seeds:
        kf = KFold(n_splits=folds, shuffle=True, random_state=seed)
        for i, (train_index, valid_index) in enumerate(kf.split(train_x, train_y)):
            print('************************************ 随机种子{}***第{}折 ************************
            trn_x, trn_y, val_x, val_y = train_x[train_index], train_y[train_index], train_x[valic

            #model = buildmodel3()
            model = tf.keras.models.load_model('/root/model/cnnbestnew_425_1_2000.h5')
            model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0009),
             loss = 'categorical_crossentropy',
            metrics = ['acc']
             )

            best_weights_filepath = '/root/model/cnnbestnew_429_{}_{}.h5'.format(i+1,seed)

            earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_acc',patience=4,verbose=2,m
            reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', patience=3, mode
            saveBestModel = tf.keras.callbacks.ModelCheckpoint(best_weights_filepath, monitor='va

            my_callbacks = [earlystop,reduce_lr,saveBestModel]
            model.fit(trn_x,trn_y,epochs=30,batch_size=256,validation_data=(val_x,val_y),shuffle=

          #注意，这里要重新加载保存的模型，即最优模型；否则他会用最后一次迭代的模型去推理
            model = tf.keras.models.load_model('/root/model/cnnbestnew_429_{}_{}.h5'.format(i+1,se
#             test_pred = model.predict(test)
#             test_temp = pd.DataFrame(np.zeros((20000,4)))
#             for t in range(len(test_pred)):
#                 a = pd.DataFrame(test_pred).iloc[t,:].argmax(0)
#                 test_temp.iloc[t,a] = 1

#             val_pred = pd.DataFrame(val_pred)
#             for col in range(4):
#                 val_pred.iloc[:,col] = val_pred.iloc[:,col].apply(lambda x:0 if x<0.5 else 1)

            val_pred = model.predict(val_x)
            val_temp = pd.DataFrame(np.zeros((10000,4)))
            for t in range(len(val_pred)):
                a = pd.DataFrame(val_pred).iloc[t,:].argmax(0)
                val_temp.iloc[t,a] = 1

            #tests.append(test_temp)
            score=abs_sum(val_y, val_temp)
            cv_scores.append(score)
            print(score)
        print("score_mean:{}".format(np.mean(cv_scores)))

    # return tests
```

In [32]:

```
result = cv_model(X,Y,test)
```

...

In [34]:

```
preds = 0
for i in range(10):
    for seed in [2000]:
        model = tf.keras.models.load_model('。/model/cnnbestnew_429_{}_{}.h5'.format(str(i+1),seed
        pred = model.predict(test)
        preds+=pred
temp = preds/10   #在172的model3上fintuing,
```

In [ ]:

```
tmp = 0
for i in result:
    tmp+=i
tmp = tmp/10
temp = pd.DataFrame(tmp)
```

In [35]:

```
print(np.shape(temp))
```

(20000, 4)

In [36]:

```
temp1 = pd.DataFrame(np.zeros((20000,4)))
temp = pd.DataFrame(temp)
for t in range(len(temp)):
    a = temp.iloc[t,:].argmax(0)

    temp1.iloc[t,a] = 1
temp1.head()

#此种方式相比用阈值分割处理，效果更佳
```

Out[36]:

	0	1	2	3
0	1.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	1.0
3	1.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0

In [37]:

```
results=pd.read_csv('./submit/sample_submit.csv')
results['label_0']=temp1[0]
results['label_1']=temp1[1]
results['label_2']=temp1[2]
results['label_3']=temp1[3]
```

In [38]:

```
results.to_csv('./submit/cnnbest_4_29_1.csv',index=False)
```

In [ ]:

In [ ]: