ANN

May 17, 2020

[1]:
```
import os
import joblib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import regularizers
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.models import Sequential
```

[2]:
```
X = []
y = []
for year in range(1919, 2020):
    df = pd.read_csv('../core/tensors/games/{}.csv'.format(year), header=None)
    for index in range(0, df.shape[0]):
        game = list(df.iloc[index])
        y_sample = int(game[-1])
        game = game[:-1]
        X.append(game)
        y.append(y_sample)
X = np.array(X)
y = np.array(y)
y = y[..., np.newaxis]
```

[3]:
```
X.shape
```

[3]:
```
(176687, 540)
```

[4]:
```
y.shape
```

[4]:
```
(176687, 1)
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

[6]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[7]:
```
epochs = 4000
batch_size = 2048
loss_param = 'binary_crossentropy'
optimizer_param = 'adam'
stop_monitor = 'val_loss'
metric = 'accuracy'
stop_patience = 15
```

[8]:
```
model = Sequential()

model.add(Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001),
                input_shape=X_train.shape[1:]))
model.add(Dropout(0.5))

model.add(Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.
↪0001)))
model.add(Dropout(0.5))

model.add(Dense(units=1, activation='sigmoid'))

model.compile(loss=loss_param, optimizer=optimizer_param)
```

[9]:
```
early_stop = EarlyStopping(monitor=stop_monitor, patience=stop_patience)
```

[11]:
```
results = model.fit(x=X_train, y=y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(X_test, y_test),
                    callbacks=[early_stop]
                    )
```

```
Train on 141349 samples, validate on 35338 samples
Epoch 1/4000
141349/141349 [==============================] - 2s 16us/sample - loss: 0.7246 -
val_loss: 0.7154
Epoch 2/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.7119 -
val_loss: 0.7068
```

```
Epoch 3/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.7048 -
val_loss: 0.7016
Epoch 4/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.7004 -
val_loss: 0.6975
Epoch 5/4000
141349/141349 [==============================] - 1s 8us/sample - loss: 0.6967 -
val_loss: 0.6958
Epoch 6/4000
141349/141349 [==============================] - 1s 8us/sample - loss: 0.6949 -
val_loss: 0.6939
Epoch 7/4000
141349/141349 [==============================] - 1s 8us/sample - loss: 0.6927 -
val_loss: 0.6912
Epoch 8/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6909 -
val_loss: 0.6900
Epoch 9/4000
141349/141349 [==============================] - 1s 8us/sample - loss: 0.6893 -
val_loss: 0.6904
Epoch 10/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6882 -
val_loss: 0.6879
Epoch 11/4000
141349/141349 [==============================] - 2s 12us/sample - loss: 0.6875 -
val_loss: 0.6886
Epoch 12/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6870 -
val_loss: 0.6868
Epoch 13/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6861 -
val_loss: 0.6882
Epoch 14/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6860 -
val_loss: 0.6869
Epoch 15/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6850 -
val_loss: 0.6865
Epoch 16/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6848 -
val_loss: 0.6882
Epoch 17/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6857 -
val_loss: 0.6868
Epoch 18/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6846 -
val_loss: 0.6860
```

```
Epoch 19/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6847 -
val_loss: 0.6848
Epoch 20/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6844 -
val_loss: 0.6867
Epoch 21/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6837 -
val_loss: 0.6853
Epoch 22/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6841 -
val_loss: 0.6864
Epoch 23/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6834 -
val_loss: 0.6858
Epoch 24/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6833 -
val_loss: 0.6866
Epoch 25/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6830 -
val_loss: 0.6851
Epoch 26/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6839 -
val_loss: 0.6856
Epoch 27/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6838 -
val_loss: 0.6867
Epoch 28/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6829 -
val_loss: 0.6838
Epoch 29/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6829 -
val_loss: 0.6850
Epoch 30/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6831 -
val_loss: 0.6839
Epoch 31/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6828 -
val_loss: 0.6857
Epoch 32/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6833 -
val_loss: 0.6852
Epoch 33/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6827 -
val_loss: 0.6862
Epoch 34/4000
141349/141349 [==============================] - 2s 13us/sample - loss: 0.6837 -
val_loss: 0.6845
```

```
Epoch 35/4000
141349/141349 [==============================] - 2s 11us/sample - loss: 0.6834 -
val_loss: 0.6870
Epoch 36/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6830 -
val_loss: 0.6841
Epoch 37/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6831 -
val_loss: 0.6840
Epoch 38/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6826 -
val_loss: 0.6860
Epoch 39/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6823 -
val_loss: 0.6848
Epoch 40/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6830 -
val_loss: 0.6845
Epoch 41/4000
141349/141349 [==============================] - 1s 9us/sample - loss: 0.6828 -
val_loss: 0.6863
Epoch 42/4000
141349/141349 [==============================] - 1s 10us/sample - loss: 0.6832 -
val_loss: 0.6860
Epoch 43/4000
141349/141349 [==============================] - 1s 11us/sample - loss: 0.6823 -
val_loss: 0.6864
```

[12]:
```
losses = model.history.history
losses['loss'] = np.asarray(losses['loss'])
losses['val_loss'] = np.asarray(losses['val_loss'])
final_number_of_epochs = len(losses['loss'])
min_loss = losses['loss'].min()
mean_loss = losses['loss'].mean()
final_loss = losses['loss'][-1]
min_val_loss = losses['val_loss'].min()
mean_val_loss = losses['val_loss'].mean()
final_val_loss = losses['val_loss'][-1]


def get_model_summary():
    output = []
    model.summary(print_fn=lambda line: output.append(line))
    return str(output).strip('[]')


summary = get_model_summary()
```

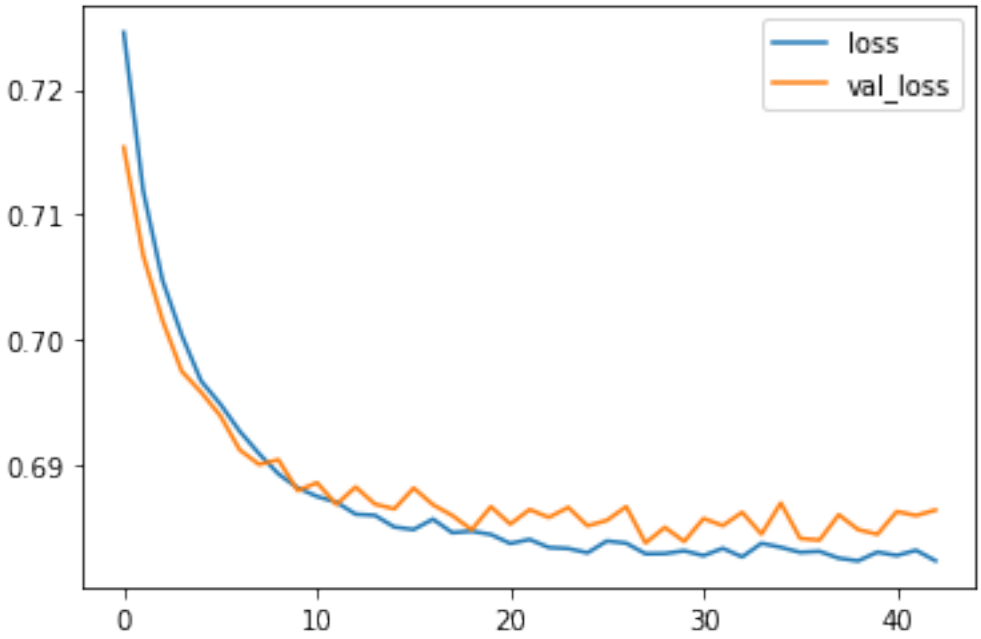
```
record = {
    'Epochs': final_number_of_epochs,
    'Batch_Size': batch_size,
    'Loss_Func': loss_param,
    'Optimizer': optimizer_param,
    'Early_Stop_Monitor': stop_monitor,
    'Early_Stop_Patience': stop_patience,
    'Min_Loss': min_loss,
    'Mean_Loss': mean_loss,
    'Final_Loss': final_loss,
    'Min_Val_Loss': min_val_loss,
    'Mean_Val_Loss': mean_val_loss,
    'Final_Val_Loss': final_val_loss,
    'Model': summary
}
```

[13]:
```
new_data = pd.DataFrame(record, index=[0])
if os.path.exists('../core/records/games_ann.csv'):
    df_records = pd.read_csv('../core/records/games_ann.csv')
    df_records = df_records.append(new_data)
else:
    df_records = pd.DataFrame(new_data)

df_records.to_csv('../core/records/games_ann.csv',
                  index=False, float_format='%g')

model.save('../core/models/games_ann.h5')
```

[14]:
```
losses = pd.DataFrame(model.history.history)
```

[15]:
```
losses.plot()
```

[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x139561a50>
```


loss
val_loss
0.72
0.71
0.70
0.69
0
10
20
30
40

[ ]:

[ ]: