ArtificialMinds

Higgs Boson Classification Model Documentation

18 4 2023

BY Ayla Hmadi, Joelle El Homsi, Bucker Yahfoufi

Code

You can find the source code alongside the python notebook and report in our github repository
https://github.com/JJoellee/Artificial-Minds.git

Importing libraries

These are the necessary libraries for data processing, model development, and model evaluation:

1. Pandas: for data manipulation

2. Numpy: for numerical computations

3. SKlearn : for preprocessing and model selection

4. TensorFlow: for developing deep learning models

5. Keras: for building deep learning models

```
import pandas as pd
import numpy as np
import sklearn
import tensorflow
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, KFold, cross_val_score
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, LeakyReLU, Dense, Dropout, BatchNormalization, Input
from tensorflow.keras.optimizers import Adam, Nadam
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.regularizers import l2


#for data preprocessing
from google.colab import drive

#for loading the model
from tensorflow.keras.models import load_model
```

Data preprocessing

The data is loaded into a pandas dataframe from a CSV file located in Google Drive. The columns in the dataframe are then renamed to have more meaningful names, and missing values are removed. The features are then scaled using StandardScaler from SKlearn.

```
drive.mount('/content/drive')

path = '/content/drive/MyDrive/Artifical Minds - CMPS261/HIGGS_train.csv'
data = pd.read_csv(path)

column_names = ['class_label', 'lepton_pt', 'lepton_eta', 'lepton_phi', 'missing_energy_magnitude', 'missing_energy_phi',
                'jet_1_pt', 'jet_1_eta', 'jet_1_phi', 'jet_1_b-tag', 'jet_2_pt', 'jet_2_eta', 'jet_2_phi', 'jet_2_b-tag',
                'jet_3_pt', 'jet_3_eta', 'jet_3_phi', 'jet_3_b-tag', 'jet_4_pt', 'jet_4_eta', 'jet_4_phi', 'jet_4_b-tag',
                'm_jj', 'm_jjj', 'm_lv', 'm_jlv', 'm_bb', 'm_wbb', 'm_wwbb']
data.columns = column_names

print(data.isna().sum())
data['jet_1_phi'] = pd.to_numeric(data['jet_1_phi'], errors='coerce')
data['jet_4_b-tag'] = pd.to_numeric(data['jet_4_b-tag'], errors='coerce')
data['class_label'] = data['class_label'].astype(int)

data = data.dropna();
data.drop_duplicates(inplace=True)
```

Data scaling

The features are then standardized using StandardScaler from Sklearn. This is done to ensure that all features have the same scale and therefore, the model doesn't give preference to one feature over another.

1. Split the data into features and labels

```
[ ]  X = data.iloc[:, 1:]
     y = data.iloc[:, 0]
     print(y.values)
```

```
[1 1 0 ... 1 1 0]
```

2. Feature scaling

```
[ ]  scaler = StandardScaler()
     X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

Model architecture

The model is defined using the Keras API. It consists of an input layer, four hidden layers, and an output layer. The input layer takes the shape of the number of features. Each hidden layer consists of a dense layer with ReLu activation, a batch normalization layer, and a dropout layer. The output layer consists of a dense layer with sigmoid activation. The model is compiled with binary cross-entropy loss and Adam optimizer. The model is then trained using the Keras Classifier API from SKlearn, using a 33% validation split and the EarlyStopping and ReduceLROnPlateau callbacks.

```
def create_improved_model(activation='relu', optimizer='adam'):
    inputs = Input(shape=(X.shape[1],))
    x = Dense(1024, activation=activation)(inputs)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(512, activation=activation)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.4)(x)
    x = Dense(256, activation=activation)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.3)(x)
    x = Dense(128, activation=activation)(x)
    x = BatchNormalization()(x)
    x = Dropout(0.2)(x)
    output = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=inputs, outputs=output)
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

# Create the model
improved_model = KerasClassifier(build_fn=create_improved_model, epochs=50, batch_size=256 , verbose=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Add EarlyStopping and ReduceLROnPlateau callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
lr_reduction = ReduceLROnPlateau(monitor='val_loss', patience=2, factor=0.5, min_lr=0.00001)

# Train and evaluate the model
history = improved_model.fit(X_train, y_train, validation_split=0.33, callbacks=[early_stopping, lr_reduction], verbose=1)
test_accuracy = improved_model.score(X_test, y_test)
print("Test accuracy:", test_accuracy)
```

The output was the following:

```
469/469 [==============================] - 4s 9ms/step - loss: 0.4858 - accuracy: 0.7595
Test accuracy: 0.7595190405845642
```

Saving the model

The trained model is saved in a .h5 file. The saved model is then located and its summary is printed.

```
# Save the model
improved_model.model.save('my_model.h5')


# Load the saved model
loaded_model = load_model('my_model.h5')

# Print the summary of the loaded model
loaded_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 28)]              0

 dense_5 (Dense)             (None, 1024)              29696

 batch_normalization_4 (Batc  (None, 1024)             4096
 hNormalization)

 dropout_4 (Dropout)         (None, 1024)              0

 dense_6 (Dense)             (None, 512)               524800

 batch_normalization_5 (Batc  (None, 512)              2048
 hNormalization)

 dropout_5 (Dropout)         (None, 512)               0

 dense_7 (Dense)             (None, 256)               131328

 batch_normalization_6 (Batc  (None, 256)              1024
 hNormalization)

 dropout_6 (Dropout)         (None, 256)               0

 dense_8 (Dense)             (None, 128)               32896

 batch_normalization_7 (Batc  (None, 128)              512
 hNormalization)

 dropout_7 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 1)                 129

=================================================================
Total params: 726,529
Trainable params: 722,689
Non-trainable params: 3,840
_________________________________________________________________
```

Overall

This code takes the Higgs boson dataset provided by Dr. Rida, preprocesses it for training, creates a deep learning model using Keras, trains it on the data, and saves the trained model for future use, which is in our case the competition.