Exoplanet Hunting

KEPLER LABELED TIME SERIES DATA ANALYSIS

RENEE ADONTENG

Table of Contents

Introduction

Since NASA's Kepler Space Telescope data is still publicly available, it's possible to make predictions about which exoplanets might be susceptible to life. The goal of this project is to develop neural networks to successfully model data for the detection of exoplanets. Using Keras as the deep learning framework, these models will be able to predict the existence of an exoplanet utilizing the flux (light intensity) readings.

Previously, others have researched the same topic and uploaded kernels of their work on Kaggle. We decided to use neural networks such as multilayer perceptron (MLP), convolutional (CNN) and recurrent (RNN) as an aid to build our own models. In order to improve our models' performance to our best capability, we tried different preprocessing methods and focused on addressing overfitting problems, which allowed us to evaluate our models from several perspectives. We all worked on the data preprocessing, but mainly Zhilin. Since most of data Each of my group members focused on creating their own neural network. I was assigned MLP, Zixuan was assigned RNN and Zhilin decided to create a CNN model.

Individual Work in Detail

I used a base MLP model with 4 neuron hidden layers, a rectified linear activation function on hidden neurons, Dropout of 0.25 between each layer, and sigmoid logistic activation function on output neurons. A batch size of 3 with epochs of 15 and a batch size of 32 with epochs of 50 was used, with the training data. Normally, the training dataset is shuffled after each batch or each epoch, which can aid in fitting the training dataset on classification and regression problems. Shuffling was turned off, as it seemed to result in better performances. The model was compiled using the efficient ADAM or SGD optimization algorithm and binary cross entropy loss function. To judge the performance of the network, a classification report was produced to show the metrics, which included accuracy, precision and F1 score of the validation set.

Binary Cross-Entrophy (BCE)

The dataset was trained as a binary classifier with a target of either exoplanet or non-exoplanet. Therefore, I used a binary cross-entropy as our loss function. The concept behind binary cross-entropy is to compute the cross-entropy on top of the probabilities associated with the true class of each point. Then I can compute the average of all observations in both classes, positive and negative:

$$y_i = 1 \Rightarrow log(p(y_i))$$
$$y_i = 0 \Rightarrow log(1 - p(y_i))$$

After, you can take any observations, either from the positive or negative class, under the same formula. Binary cross-entropy measures how far away the true value (which is either 0 or 1) from the prediction and then averages these class-wise errors to obtain the final loss.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

F1 Score

Since this is a classification problem, I need to look at the results of classification reports to assess the performance of my models. F1 score is an applicable indicator, especially for an imbalanced dataset.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Table I Classification Report Structure

Precision measures about how precise/accurate the model is out of those predicted positive and how many of them are actually positive. Precision is a decent measure to determine when the costs of false positive is high.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall calculates how many of the actual positives the model captures through labeling it as true positive.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

F1 is a function of Precision and Recall combined, the formula is as follows.

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

To seek a balance between Precision and Recall, F1 score was selected as a measure.

Model Structure

Depending on the batch size, each experimental scenario ran between 5-7 minutes and the val_loss and accuracy score on the test set was recorded from the end of each run. To visualize the accuracy of the MLP model, a loss plot was conducted. I investigated the training of a MLP with 4 hidden layers and 64, 32, 8, and 1 neuron(s) in its respective hidden layer.

```
model = Sequential() # initialize network
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))_# adding an input layer and the first hidden layer
model.add(Dropout(0.25))
model.add(Dense(32, activation='relu'))_# adding the second hidden layer
model.add(Dropout(0.25))
model.add(Dense(8, activation='relu'))_# adding the third hidden layer
model.add(Dropout(0.25))
model.add(Dense(1, activation='sigmoid'))_# adding the fourth hidden layer
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])_#compiling the MLP
# model.compile(loss='binary_crossentropy', optimizer='SGD', metrics=['accuracy'])

# Simple Early Stopping to get the best model
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=200)
mc = ModelCheckpoint('MLP.h5', monitor='val_loss', mode='max', verbose=1, save_best_only=True)

# Fitting the MLP model to the training set
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), batch_size=3, epochs=15, class_weight='auto',
          callbacks=[es, mc])
```

Figure 1. Code extracted from MLP.py that shows the multilayer perceptron model structure. There are 4 hidden layers.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                204672
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 264
_________________________________________________________________
dropout_3 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 9
=================================================================
Total params: 207,025
Trainable params: 207,025
Non-trainable params: 0
```

Figure 2 Model structure which investigated the training of a MLP with 4 hidden layers and 64, 32, 8, and 1 neuron(s) in its respective hidden layer, which produced 207,025 trainable parameters. There were no non-trainable parameters.

Results

I imported early stopping feature to avoid overfitting and to ensure the best model was displayed. No matter what the number batch size or epochs I used, class 0 (exoplanets) F1 score and accuracy would not change significantly. It ranged from 0.98 to 1.00 each time I ran the code. I noticed as I decreased the batch size and epochs training, the model performance improved for class 1 (non-exoplanets). Model 1 fit consisted of x_train and y_train sampling, batch size 3, and epochs 15. That model was compiled using the efficient ADAM optimization algorithm and binary cross entropy loss function. Model 2 fit consisted of x_train and y_train sampling, batch size 32 and epochs 50. That model was compiled using the efficient SGD optimization algorithm and binary cross entropy loss function. Both MLP models did not perform well as the F1 score was very low and wouldn't perform above 0.33. There were not many MLP models to compare these models too. It's also important to note, the precision value 0.29 and 0.13 were fairly low as well. Maybe this chosen algorithm was not the best option for this problem. Perhaps CNN and RNN is a better fit to produce efficient models.

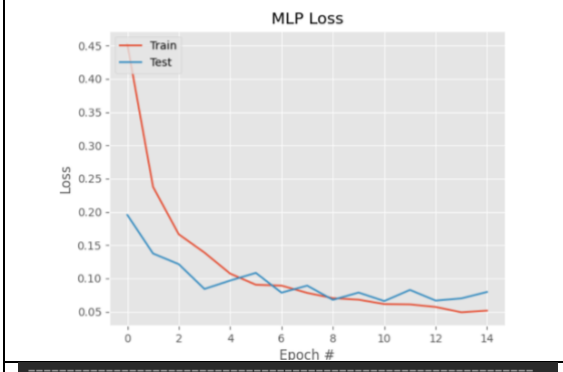
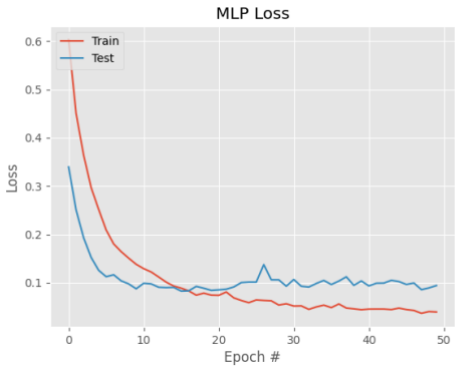
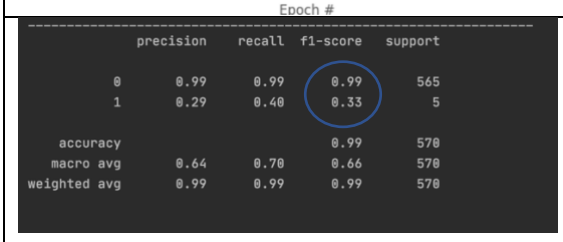
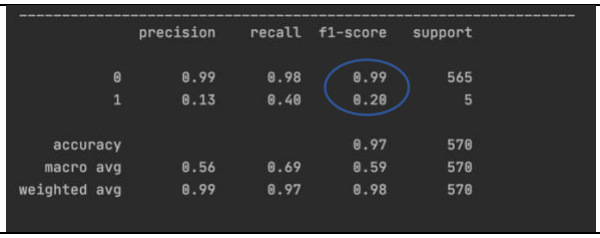
Model 1 Loss Plot + Classification Report	Model 2 Loss Plot + Classification Report
	
	
Optimizer = 'Adam', Batch Size = 3, Epochs = 15	Optimizer = 'SGD', Batch Size = 32, Epochs = 50

Figure 3 Both MLP models did not perform well as the F1 score was very low and wouldn't perform above 0.33. The precision value 0.29 and 0.13 were fairly low as well.

Summary and Conclusions

The MLP model did not perform as the F1 score was very low and wouldn't perform above 0.33. There were not many MLP models to compare my model too. Maybe this chosen algorithm was not the best option for this problem. Evaluating some tree methods such as random forest and gradient boosting could have excelled the model more. Improving the performance with algorithm tuning such adding a learning rate and changing the network topology could have also increased the F1 score for class 1 (non-exoplanets). Although it's easy to make changes to your network structure, it's also difficult to decide which one will pay off. Deciding how many layers and how many neurons you need may throw the entire model off. If

time permitted, I would have tried more combinations of hidden layers with a lot of neurons and some with a few neurons per layer. Lastly, I could have experimented with very large and very small learning rates. Trying a learning rate that decreased over epochs or even adding a momentum term could have made some differences. It's also important to note that the dataset was very imbalanced and already split into test and train datasets. The dataset probably needed to be manipulated differently than it was for CNN or RNN.

Calculate the Code Percentage – 22%

References

"Exoplanet Hunting in Deep Space". Kaggle. https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data. May 2020

Brownlee, Jason. "*How to Develop Multilayer Perceptron Models for Time Series*". Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-multilayer-perceptron-models-for-time-series-forecasting/. May 2020

Brownlee, Jason. *"How to Stop Training Deep Neural Networks".* https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/. May 2020

"What is an Exoplanet" NASA Science, Space Place. https://spaceplace.nasa.gov/all-about-exoplanets/en/. May 2020