Machine Learning Engineer Nanodegree

Capstone Proposal

Daniel Bank
November 3rd, 2018

Proposal

Domain Background

The Large Synoptic Survey Telescope (LSST), currently under construction in the Atacama desert of Chile, is the most awesome telescope that mankind will have built thus far in history. It features an 8.4m mirror, 3.2 gigapixel camera, and 6.5m effective aperture [1]. Every night, it will gather between 20 and 40 TB of image data of the southern sky. While sifting through that much data to identify cosmological objects would be impractical for humans, it is a perfect use case for machine learning.

In preparation for the massive amounts of observations that will come from the LSST, an open data competition has been launched on Kaggle called the Photometric LSST Astronomical Time Series Classification Challenge (PLAsTiCC). The goal of the challenge is to correctly classify time-varying cosmological objects in simulated astronomical time-series data similar to the data that the LSST will generate.

I first became aware of this challenge while browsing the Kaggle competitions. It was fortuitous that it's timeline coincided with that of my capstone project as I am already familiar with the LSST and thought this would be the perfect culmination for my Machine Learning Nanogree. My personal interest in astronomy stems partly from my background as a Physics major but more so from son, Arthur, who loves everything about the night sky. He has taught me to really appreciate the beauty of the cosmos.

Problem Statement

PLAsTiCC includes a 7 GB training dataset of labeled data. That is to say, the cosmological objects in the training data have known classifications that our model can be trained on. Our trained model must then for every object i in the test dataset predict the probabilities of it belonging to class j. For any object, these probabilities should sum to 1.

$$0 \le P_{ij} \le 1 \qquad \forall i, j$$
$$\sum_j P_{ij} = 1 \qquad \forall i$$

The largest `Pij` for object i is what our model will finally classify it as. These predicted classifications can then be compared to the actual classifications for the objects in the test dataset to obtain an accuracy for our model.

One interesting aspect of this competition is that the training dataset is a small subset of the test dataset and is also a poor representation of it. This design decision was made to imitate the real-world challenges that astronomers face when trying to classify cosmological objects.

Datasets and Inputs

The PLAsTiCC website provides both training and test datasets. Each dataset consists of two files: a header file, that summarizes astronomical information about the objects, and a time-series data file, that contains light-curve data as a function of time and passband for each object in the header file.

The columns of the header file are defined as follows in the PLAsTiCC Data Note [2]:

- `object_id`: The Object ID, unique identifier (given as int32 numbers).
- `ra` : Right ascension, sky coordinate: longitude, units are degrees (given as float32 numbers).
- `decl` : Declination, sky coordinate: latitude, units are degrees (given as float32 numbers).
- `gal_l` : Galactic longitude, units are degrees (given as float32 numbers).
- `gal_b` : Galactic latitude, unit are degrees (given as float32 numbers).
- `ddf` : A Boolean flag to identify the object as coming from the DDF (Deep Drilling Field) survey area (with value `ddf` = 1 for the DDF). Note that while the DDF fields are contained within the full WFD (Wide-Fast-Deep) survey area, the DDF fields have significantly smaller uncertainties, given that the data are provided as additions of all observations in a given night.
- `hostgal_specz` : The spectroscopic redshift of the source. This is an extremely accurate measure of redshift, provided for the training set and a small fraction of the test set (given as float32 numbers).
- `hostgal_photoz` : The photometric redshift of the host galaxy of the astronomical source. While this is meant to be a proxy for `hostgal_specz` , there can be large differences

between the two and `hostgal_photoz` should be regarded as a far less accurate version of `hostgal_specz`. The `hostgal_photoz` are given as float32 numbers.
- `hostgal_photoz_err`: The uncertainty on the `hostgal_photoz` based on LSST survey projections, given as float32 numbers.
- `distmod`: The distance (modulus) calculated from the `hostgal_photoz` since this redshift is given for all objects (given as float32 numbers). Computing the distance modulus requires knowledge of General Relativity, and assumed values of the dark energy and dark matter content of the Universe.
- `MWEBV` = MW E(B-V): this 'extinction' of light is a propert of the Milky Way (MW) dust along the line of sight to the astronomical source, and is thus a function of the sky coordinates of the source `ra`, `decl`. This is used to determine a `passband` dependent dimming and reddening of light from astronomical sources and is given as float32 numbers.
- `target`: The class of the astronomical source. This is provided in the training data. Correctly determining the target (correctly assigning classification probabilities to the objects) is the goal of the classification challenge for the test data. The `target` is given as int8 numbers.

The columns for the light-curve data file are defined as follows in the PLAsTiCC Data Note:

- `object_id`: Same key as in the header data table, given as int32 numbers.
- `mjd`: The time in Modified Julian Date (MJD) of the observation. The MJD is a unit of time introduced by the Smithsonian Astrophysical Observatory in 1957 to record the orbit of Sputnik. The MJD is defined to have a starting point of midnight on November 17, 1858. The 25th of September 2018 has an MJD of 58386. The MJD can be converted to Unix epoch time with the formula `unix_time` = (MJD40587)86400. The units are days, and the numbers are given as float64 numbers.
- `passband`: The specific LSST `passband` integer, such that `u`, `g`, `r`, `i`, `z`, `y` = 0, 1, 2, 3, 4, 5 in which it was viewed. These are given as int8 numbers.
- `flux`: The measured flux (brightness) in the `passband` of observation as listed in the `passband` column. The `flux` is correlated for `MWEBV`, but for large dust extinctions the uncertainty will be much larger in spite of the correction. The dust is given as a float32 number (note that the units for both `flux` and `flux_err` are arbitrary).
- `flux_err`: The uncertainty on the measurement of the `flux` listed above, given as float32 number.
- `detected`: If `detected` = 1, the object's brightness is significantly different at the 3σ level relative to the reference template. This is given as a Boolean flag.

Solution Statement

A satisfactory solution to this problem will use the time-dependent light-curve data to solve a multiclass classification problem. There are numerous methods we can employ to achieve this.

One might be to use a Convolutional Neural Network (CNN) where the final layer is a softmax function that yields the `Pij`. Another could be to use a Random Forest Classifier, as was done in the Benchmark Solution. Regardless of the approach chosen, we will evaluate the accuracy of the model by plotting at its confusion matrix for the test dataset and comparing it to that of the naive classifier.

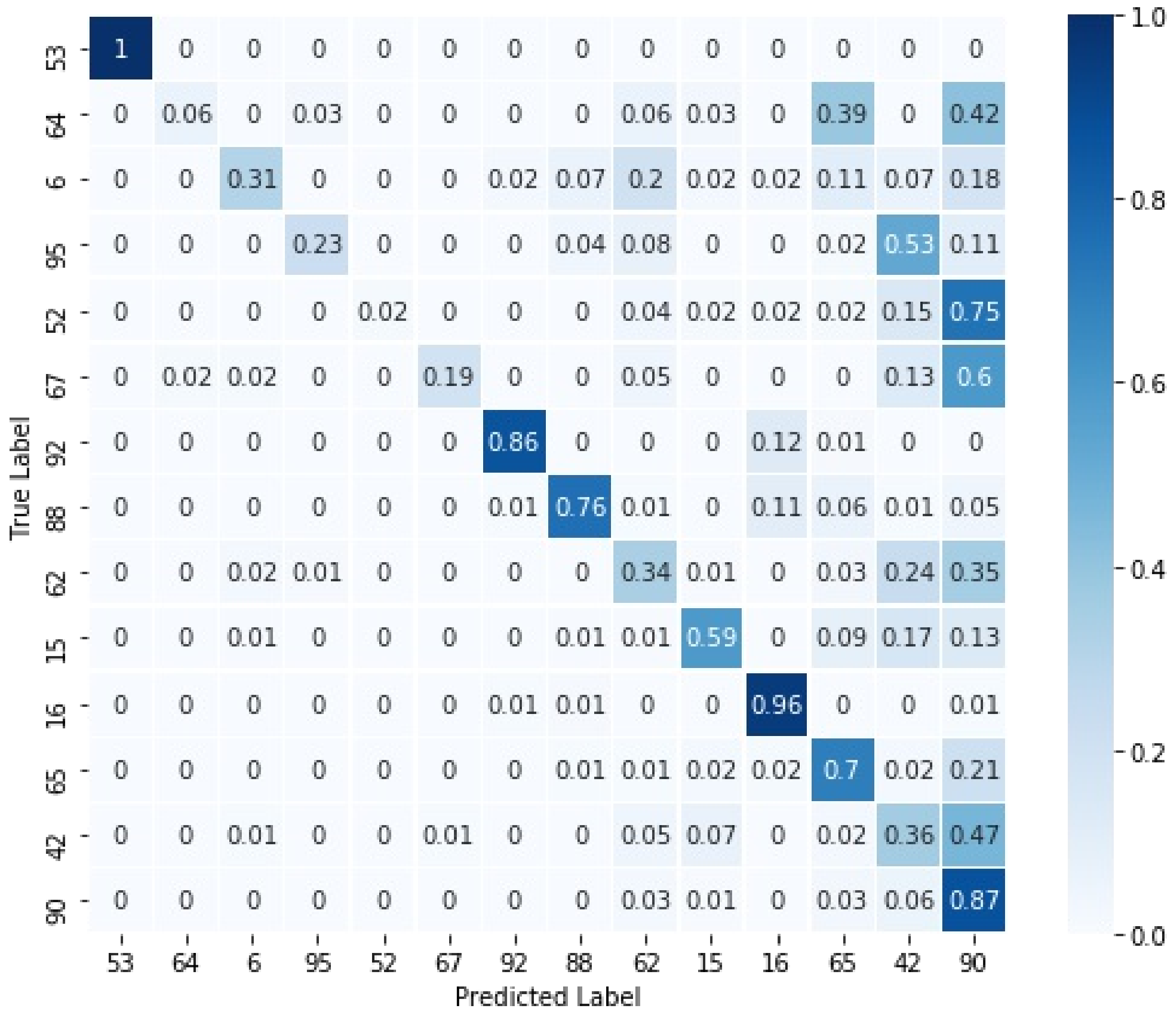
Benchmark Model

The PLAsTiCC team has provided a naive classifier for the challenge. The naive classifier is designed as follows:

1. The raw tabular data is converted into a form suitable for analysis. Using the `cesium` package for Python, `Timeseries` objects are constructed for each light curve.
2. The data is split into a training and test set.
3. The dimensionality of the training data is reduced using Principle Component Analysis (PCA). The result of this step are features which are linear combinations of the `passband` data. With just eight (8) features, we can account for almost all of the explained variance.
4. The reduced dataset is fit with a `RandomForestClassifier`.

Evaluation Metrics

The performance of the naive classifier on the test data can be visualized in the following confusion matrix:



In this matrix, we can see the model's predicted labels versus the true labels for the test set. Ideally, this confusion matrix would be the Identity matrix, with all values along the diagonal being 1 and everything else being 0. That would mean that the model predicted the correct label 100% of the time.

When designing my classifier, I will generate a similar confusion matrix so that my results can be objectively compared to the naive classifier. My objective is to have an average prediction accuracy per label that is higher than the naive classifier's accuracy of 0.518 (see calculation below):

```
(1.00 + 0.06 + 0.31 + 0.23 + 0.02 +
 0.19 + 0.86 + 0.76 + 0.34 + 0.59 +
 0.96 + 0.70 + 0.36 + 0.87) / 14
= 0.518
```

Project Design

We propose the following design for a PLAsTiCC classification pipeline:

1. Separate Classifiers

As recommended by the PLAsTiCC Classification Demo, we will split the data along certain dimensions (yet to be decided) and create separate classifiers for the two sets [3]. One possibility is to split the data according to whether it was captured with DDF or WFD field:

```
# Separating data by DDF / WDF fields
ddf_data = metadata['ddf'] == True
wdf_data = metadata['ddf'] == False
```

Another possibility is to separate the data according to intergalactic and extragalactic sources on the basis of redshift (intergalactic sources have a redshift of 0). We might even divide the data further along the lines of redshift and create multiple redshift bins (more than two for intergalactic and extragalactic).

```
# Separating data by spectroscopic redshift
# (Two bins for extragalactic and intragalactic)
intragal = metadata['hostgal_specz'] == 0.
extragal = metadata['hostgal_specz'] != 0.
```

2. Data Augmentation

Per the PLAsTiCC Classification Demo, there are definitely biases in the data. We will try to derive bias corrections based on the spectroscopic and photometric redshift and use this to apply non-linear transformations to the data prior to processing. We may try other methods for identifying and removing outlier data from our training dataset as well.

3. Principle Component Analysis (PCA)

As the dimensionality of the data is large, it is appropriate to reduce it using PCA so we can have a simpler model that is faster to train and easier to reason about.

```
pca = PCA(n_components=8, whiten=True, svd_solver="full", random_state=42)
Xtrain_pca = pca.fit_transform(Xtrain)
Xtest_pca = pca.transform(Xtest)
```

4. Fit a Classifier

We will evaluate several classifiers and select whichever one achieves the best accuracy. Below are a couple directions which look promising:

- Long Short Term Memory (LSTM) Networks: LSTM networks are Recurrent Neural Networks (RNN) that are composed of LSTM units [4]. These networks specialize in classifying time-series data, and are a perfect candidate for the PLAsTiCC dataset.

- Multi Layer Perceptron (MLP): A simple neural network may perform better than an RNN for classifying the PLAsTiCC dataset. This would especially be true if the time dependence of the light curve data played a smaller role than the relationships of the different passbands themselves in the classification of the object. As it is a simpler model to understand, MLP may be a good starting place.

References

[1] - PLAsTiCC Astronomy Starter Kit

[2] - PLAsTiCC Data Note

[3] - PLAsTiCC Classification Demo

[4] - Long short term memory