Classification of Astronomical Objects on the PLAsTiCC dataset

CSC 461 Final Project Report

Richard Burke, Neeraj Adhikari

December 10, 2019

1 Introduction to the dataset

The Photometric LSST Astronomical Time-Series Classification Challenge[1] (PLAsTiCC) was posted on Kaggle[2] which presents the challenge of finding the best method for classifying celestial objects based on recordings from their respective light curves. This dataset was generated based on previous data to simulate the behavior of many different types of objects observed from Earth using highly advanced telescopes.

The Large Synoptic Survey Telescope (LSST)[3] is a new facility that is expected to be completed in 2020. The LSST will house a new telescope that, according to the web page on Kaggle, will: revolutionize the field, discovering 10 to 100 times more astronomical sources that vary in the night sky than we've ever known. Some of these sources will be completely unprecedented! Astronomers plan on having a 10-year long marathon of recording observations with the LSST. They estimate this amount of data will fill up to several petabytes of data once complete.

2 Why we picked the dataset

Our group decided to take up the challenge because we hold a mutual interest in the field in astronomy. We also like the idea of applying our knowledge towards a real-life situation.

3 Challenges with the dataset involved

Naturally, there were many problems we faced when working on the dataset for this challenge. Although we have a very basic foundation in understanding topics in astronomy we are by no means experts in the field. To elaborate we

were able to understand certain elements of the data such as the relevance of light curve in identifying objects in the Universe but we dont have the in-depth knowledge to know how to use the specific values of light curve for classification. This means there could exist important features that we did not extract from the data.

Another issue we faced was the size of the training data vs. the size of the test data. To specify there were more than 7,800 objects in the training data to represent the 14 classes; however this is dwarfed by the more than 3.45 million objects in the test set. It is safe to assume there will be many instances that could be hard to distinguish between two or maybe even more classes.

A third problem we encountered was the distribution in the training set: there was a severe lack of representation for some classes. For example, among the 7,800 objects from the training set there were some classes that had more than 1,000 instances while there were several classes that contained less than 50 instances. This leaves a higher potential for outliers and noise to influence our data for those classes.

4 Features Extracted

Because the dataset we picked is a time-series dataset where the goal is to classify a time-series of light curves in 6 passbands, we couldnt apply the classification methods we learned in class directly. To overcome this problem and also to reduce the virtually unlimited dimensionality of the raw data, we performed some feature extraction. There are a large number of features that can be computed from any given time-series and what features are used for training and classification can make a huge difference. We used the Cesium library [4] both for handling (storage and retrieval) the time-series and for the task of feature computation. We made the decision to use the Cesium library because unlike other data handling and machine learning libraries, Cesium has good in-built support for the needs of time-series data. That includes support for missing data, non-uniform time scales, error measures for the observations, etc. For each of the 6 passbands, 11 features were computed. Here is a list of the features along with a short description:

1. **Amplitude:** the difference between the maximum value and the minimum value
2. **Percent beyond one standard deviation:** The percentage of observed values that are more than one standard deviation away from the weighted average.
3. **Maximum:** maximum observed value.
4. **Maximum slope:** largest value of the rate of change of observations.

5. **Median:** Median element of the observed values.
6. **Median Absolute Deviation:** Median of the absolute differences from the median of the observed values.
7. **Percent close to median:** the percentage of values within a computed distance from the median.
8. **Minimum:** Minimum observed value.
9. **Skew:** The skewness of observed values, which is a measure of how unsymmetrical the distribution is.
10. **Standard Deviation:** the standard deviation of observed values
11. **Weighted Average:** the arithmetic mean of observed values, weighted by the observation errors

5 Overview of classification techniques used

We used the scikit-learn library for all of the classification techniques described below. This is because we were familiar with scikit-learn and were able to use it with more ease than any other library. We also found scikit-learn simpler and more suitable to rapid experimentation and comparative evaluation of models than other well-known libraries like PyTorch and Tensorflow. Discussed below are the classification techniques we used:

5.1 Random Forest

The first classification approach applied to the data was a Random Forest. We used scikit-learns Random Forest application with a tree count of 200 which seemed like a good compromise between model complexity and ability to capture the intricacies of data. The splitting criterion used was the GINI coefficient. There was no set maximum depth for the Random Forest, the tree was deepened until all nodes were either pure or had a sufficiently small number of supporting examples.

5.2 Support Vector Machines

Two Support Vector Machine models were used, with only a slight difference in results. Both of the models used non-linear kernels, as it made no sense to use a linear kernel when we were already using logistic regression. The kernels we used were the Radial Basis Function (RBF) given by the equation and the sigmoid kernel given by the equation. For unknown reasons, the support vector machines did not perform as we expected them to. In fact, they were the most underperforming of all the models we tried.

5.3 Logistic Regression

We also used a Logistic Regression classifier. The primary purpose of including this model in our evaluation was to test whether the data was — to an acceptable degree — linearly separable. L2 regularization was applied on top of the mean squared error loss and the model internally did the training by solving the matrix system directly for the optimum weights. As expected, Logistic Regression gave us a middling accuracy but it was still higher than the SVMs due to their unexpected underperformance.

5.4 Multi-layer Perceptrons

We trained a basic feed-forward network called the Multi-Layer Perceptron. One hidden layer with 100 neurons (the default values for the scikit-learn libraries) was used. Trying different numbers of hidden layers and different neuron counts didnt provide any better results than with a single layer of 100 neurons. The activation function used was a Rectified Linear Unit (ReLU) and the network was trained using backpropagation along with stochastic gradient descent.

5.5 K Nearest Neighbors

Having read about a different approach to the traditional kNN model, called the kNN model-based approach[5], we felt it would be interesting to include. Before this, we considered using the traditional kNN approach but this would have taken an unreasonable amount of computation and time when it came to finding the k-nearest neighbors for all ∼3.45 million objects in the test set. This model-based approach answered that exact issue by building a model for each classification based on all respective examples in the training set (∼7,800 objects). After these 14 models were built, we compared them to each object in the test set to find the corresponding nearest neighbor (k=1) amongst the models thus giving us our predictions.

6 Results of the different classification techniques

The 4-fold cross validation accuracies obtained on the training set with each of the models we tried were:

Random Forest	72.26%
Logistic Regression	47.67%
SVM (RBF)	27.62%
SVM (sigmoid)	29.47%
MLP (1 hidden layer)	58.00%
KNN (model based)	64.30%

7 Limitations and possible future improvements

Obviously, the approaches we tried are only a few among a vast number of techniques that can be used to make sense of and classify the data. Due to the relatively simple and straight-forward nature of the techniques we used, there were a lot of limitations and room for future improvement. Some of them are listed here:

7.1 Fine-tuning of hyper-parameters

Machine learning libraries like scikit-learn provide a lot of hyper-parameters that can be set by experts to fine-tune the performance and efficiency of the machine learning techniques. However, setting the hyper-parameters to good values requires either very good knowledge of the dataset and the concerned algorithm, or long-enough time and computing power so that a range of hyper-parameter values can be evaluated with cross-validation. We had neither, so we could not set optimum values for the hyperparameters for the methods we used.

7.2 Research on comparative utility of features

The total number of features calculated for each item in the dataset was 66. Each of the 66 features encoded and summarized some information about the original time-series. However, there are a lot more possibilities than using just this many features. Even with just the features we did use, we do not have a clear understanding of which features are influencing the classification decisions the most and which features are irrelevant to the classification. Also, it is highly probable that we did not include in our set some very important features that would have influenced the classification by a large degree hence giving better results. One group of candidates for such features is the frequency-domain decomposition features (like DTFT, DCT, Periodograms, etc.) which decompose the time-series light waves into their frequency components. Especially considering that there are a lot of periodically varying objects in our datasets, such features could have been useful for resolving the high confusion on some of the classes.

8 Elimination of redundancy in the features

We computed the same set of 11 features on all 6 passbands of the light curve and aggregated the total 66 features as a single feature vector. Since for many objects the energy levels in multiple passbands are at least related if not very similar, there was inevitably a lot of redundancy introduced to the feature vector. This may have at least caused the training and prediction to be slower than optimal, if not more inaccurate. So a possible improvement to the pipeline is the addition of a statistical redundancy-removal step, such as Principal Component Analysis.

9 References

1. https://arxiv.org/abs/1810.00001
2. https://www.kaggle.com/c/PLAsTiCC-2018
3. https://www.lsst.org/
4. http://cesium-ml.org/docs/
5. https://link.springer.com/chapter/10.1007/978-3-540-39964-3_62