Project - PulsarDetection

Daniele

Hossein

S289615@studenti.polito.it

Politecnico di Torino

Outline

* Introduction
* Pulsar Features
* Building a classifier for the Avila task
* Experimental validation
* Conclusions

Introduction

The Pulsar dataset is taken from the UCI repository (Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science)

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [1].

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter (see [2] for more uses).

As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.

Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation (see [2] for an introduction to pulsar astrophysics to find out why). Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find.

We keep the original train / evaluation split

The training set contains 17,898 total examples with 639 positive examples and 16,259 negative examples.

We need the data to Train and Test the methods. Regarding to the two datasets offered, we will use all of the Train Dataset for Training and all of the Test Dataset for Testing.

----------------------------------------------------------------------------------------------------------------------------------------------

Source:

Dr. Robert Lyon, University of Manchester, School of Physics and Astronomy, Alan Turing Building, Manchester M13 9PL, United Kingdom, robert.lyon '@' manchester.ac.uk

[1] M. J. Keith et al., 'The High Time Resolution Universe Pulsar Survey - I. System Configuration and Initial Discoveries',2010, Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627. DOI: 10.1111/j.1365-2966.2010.17325.x

[2] D. R. Lorimer and M. Kramer, 'Handbook of Pulsar Astronomy', Cambridge University Press, 2005.

Pulsar Features

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded profile).

This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency (see [3] for more details). The remaining four variables are similarly obtained from the DM-SNR curve (again see [3] for more details). These are summarised below:

1. Mean of the integrated profile.

2. Standard deviation of the integrated profile.

3. Excess kurtosis of the integrated profile.

4. Skewness of the integrated profile.

5. Mean of the DM-SNR curve.

6. Standard deviation of the DM-SNR curve.

7. Excess kurtosis of the DM-SNR curve.

8. Skewness of the DM-SNR curve.

9. Class

Histogram of the Pulsar Detection dataset features without preprocessing. (training set). Features are sorted by their order, from left to right, top to bottom.

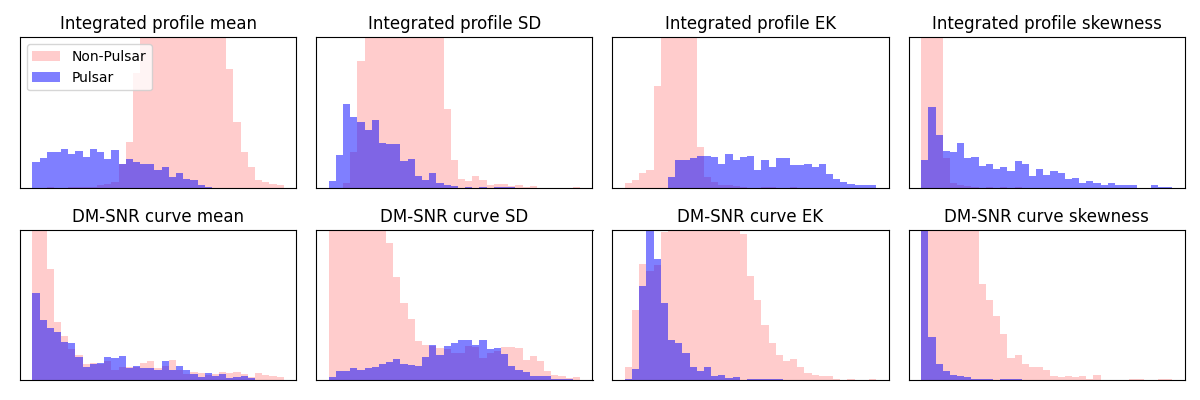


Figure 1: Histogram of the Pulsar Detection dataset features without preprocessing

----------------------------------------------------------------------------------------------------------------------------------------------

[3] R. J. Lyon, 'Why Are Pulsars Hard To Find?', PhD Thesis, University of Manchester, 2016

Pulsar Features

We have to decide which machine learning method would be best. also, we will use Gaussian Model, Logistic Regression and Support Vector Machines models.

We need to do two things with the datasets:

1. Estimate the parameters for the machine learning methods.

* In other words, to use Logistic Regression, we have to use some of the data to estimate the shape of the curve. Here, estimating parameters is called Training the algorithm.

1. Evaluate how well the machine learning methods work.

* In other words, we need to find out if the curve will do a good job categorizing new data. Here, evaluating methods is called Testing the algorithm.

Then, we can compare methods by seeing how well each one categorized the test data. We assume that this data for training and testing are the best way to divide up the data.

In many cases an analysis of the training data shows that, we have many irregular distributions in the raw features, characterized by a presence of significantly large outliers.

Due to the presence of outliers, we expect that classification approaches may produce sub-optimal results (especially Gaussian-based methods)

We therefore further pre-process data by "Gaussianizing" the features

Gaussianization is a procedure that allows mapping a set of features to values whose empirical cumulative distribution function is well approximated by a Gaussian c.d.f.

Pulsar Features

The processing consists in mapping the features to a uniform distribution and then transforming the mapped features through the inverse of Gaussian cumulative distribution function

Data preprocessing allows for the removal of unwanted data with the use of data cleaning, this allows to have a dataset to contain more valuable information after the preprocessing stage for data manipulation later in the data mining process.

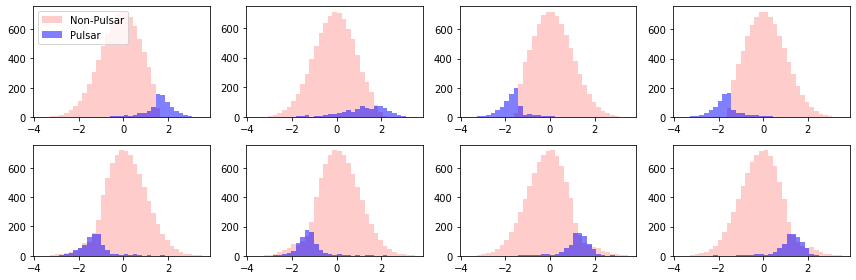


Figure 2: Histogram of the Pulsar Detection dataset features after gaussianization without preprocessing

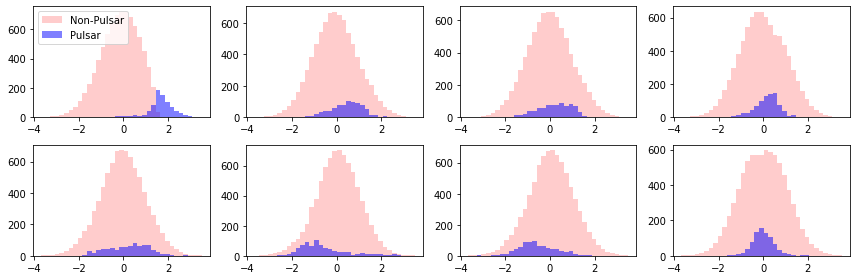


Figure 3: Histogram of the Pulsar Detection dataset features after gaussianization with preprocessing

‘’’

Left green

train\_features, train\_labels = dl.load\_train\_data()

pcc = np.corrcoef(train\_features)

Left red

preprocessed = prep.apply\_all\_preprocess(train\_features)

pcc = np.corrcoef(preprocessed)

Right green

gaussianized = prep.gaussianize(train\_features)

pcc = np.corrcoef(gaussianized)

Right red

gaussianized = prep.gaussianize(preprocessed)

pcc = np.corrcoef(gaussianized)

‘’’

Left green

Correlation coefficient of training features

Left red

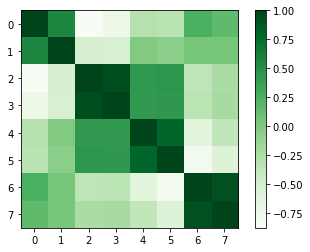
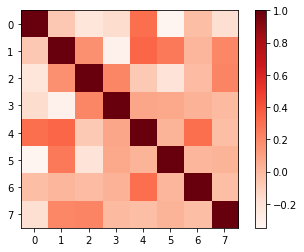
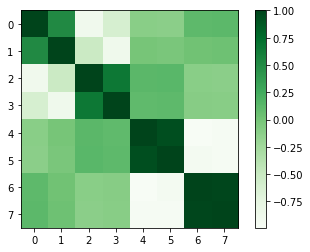
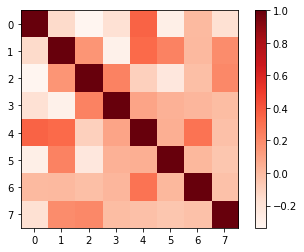
Correlation coefficient of all preprocessed features

Right green

correlation coefficient Gaussianized of training features

Right green

correlation coefficient Gaussianized of preprocessed features

Classifying Pulsar features

We start considering simple Gaussian classifiers

Although we employed Gaussianization over the whole dataset, the histograms for each class show, in some cases, moderate deviations from the Gaussian assumptions.

Since within-class covariance matrices are almost diagonal, we consider both diagonal and full-covariance models

To understand which model is most promising, and to assess the effects of using PCA, we can adopt two methodologies:

* We can split the training dataset into development (for model training) and validation subsets (single-fold in the following)
* We can employ K-Fold cross-validation

Single split:

* The final classifier will be the same that we evaluate on the validation set: model selection and hyper-parameter will be optimal at least for the validation set
* We need to train fewer models, so training is faster
* We have fewer data for validation and model training

K-Fold cross-validation:

* More data available for training and validation
* The final classifier will be obtained by re-training over the whole training set, so it will leverage additional data
* Decisions are made over the validation set for the models trained using folds. They may not be optimal for the model learned from all training data

By the way, Cross-validation uses them all one at a time to which block would be best for testing, and summarizes the results at the end. It keeps track of how well the method did with the test data. Then it uses combination of blocks to train the method.

Cross-validation allows us to compare different machine learning methods and get a sense of how well they will work in practice.

In this method, we divided the Train Dataset into 5 blocks (4 blocks for train, 1 for test). This is called Five-Fold cross-validation.

In the end, every block of data is used for testing and we can compare methods by seeing how well they performed.

In this case, since the Support Vector Machine did the best classifying the test datasets, we will use it.

Beside, we need to summarize how each method performed on the Training data. One way to do this is by creating a **Confusion Matrix** for each method.

The rows in a Confusion Matrix (It is especially called here predicted\_labels) corresponds to what the machine learning algorithm predicted and the columns (It is especially called here true\_labels) corresponds to the know truth.

Since there are only two categories to choose from: “Positives” or “Negatives”, then the bottom right-hand corner contains True Positives. These are the pulsars that had “Positives” that were correctly identified by the algorithm.

The True Negatives are in the top left-hand corner. These are the pulsars that did not have “Negatives” that were correctly identified by the algorithm.

The bottom left-hand corner contains the False Positives. These are pulsars has “Positives”, but the algorithm says they are.

Lastly, the top right-hand corner contains the False Negatives. These are when a pulsar has “Negatives”, but the algorithm said they didn’t.

The numbers along the diagonal (The True Positives and True Negatives) tell us how many times the samples were correctly classified.

The numbers not on the diagonal (the False Positives and False Negatives) are samples the algorithm messed up.

We can apply Logistic Regression to the Testing Dataset and create a Confusion Matrix.

----------------------

--------------------

The first thing we do with the Gaussian Naive Bayes classifier is making an initial guess that they are detected as Pulsar. This guess can be any probability that we want, but a common guess is estimated from the training data. That initial guesses are called Prior Probabilities.

Note: the Likelihood is the y-axis coordinate on the curve that corresponds to the x-axis coordinate. And we multiply that by the Maximum Likelihood.

To talk about a Likelihood, we assume that we have already weighed the Pulsar (or Pulsars, if it is weighed more than one). We logged transforms the individual Likelihood functions.

Step 1) We have moved the log of the first Likelihood function for reference.

Step 2) We have converted the multiplication into addition.

Step 3) We have converted 1 over the square root into the exponent -0.5 and convert the exponent into multiplication.

Step 4) We have converted the -0.5 exponent into multiplication and the log of e = 1

Step 5) The log can convert the multiplication of 2 into addition of and 2

Step 6) We have converted the exponent log(2) into 2 log()

Step 7) Lastly, the 2 divided by 2 term cancels out.

data\_loading

cross\_validation

bayes\_error\_plot

preprocess to clean data

roc det curves

run Gaussian main

خروجی رو ببین و پلات هارو

Tozh bede

Accureccy ha ke chap mishan

Rast bala false negative

Pain chap false positive

At lease 790