Final Project PDF

DBSCAN Clustering

Ruth Feinstein

Partner: Timothy DeLloyd

Math 371

May 16, 2025

Introduction

In our project, we used DBSCAN clustering to detect clusters that help determine whether a data point is likely to be a gamma ray based on its association with other points.

Data Set

What the data represents

The dataset we used is a simulated physics dataset generated using Monte Carlo methods to mimic the behavior of high-energy gamma rays detected by a Cherenkov gamma-ray telescope. The telescope observes Cherenkov radiation, light emitted by particles produced in gamma-ray-induced atmospheric showers, and uses the resulting shower image patterns to classify the origin of the particle: either a gamma ray (signal) or a hadron (background).

Variables included

This data set includes 10 variables.

1. fWidth: represents the minor axis length of the ellipse
2. fSize: the logarithm (base 10) of total light collected in the image
3. fConc: the ratio of the sum of the two brightest pixels to total light (fSize)
4. fConc1: the ratio of the brightest pixel to total light (fSize)
5. fAsym: the distance from the brightest pixel to the image center, projected along the major axis
6. fM3Long: the cube root of the third moment along the major axis (shape-related)
7. fM3Trans: the cube root of the third moment along the minor axis
8. fAlpha: the angle between the major axis and the origin direction
9. fDist: the distance from the origin to the center of the ellipse
10. Class: represents whether the point is a gamma (g), or hadron (h)

We can look at each variable in more detail using the histogram charts in the bottom section of the jupyter notebook. Each chart represents a comparison of the Hadron and Gamma data, scaled so that each has roughly the same number of points. The chart represents the distribution of the data for each variable.

Normalizing the data

We normalized the data so that the different scales of each variable don't affect the results. To normalize it, we set each point x = (x-min)/(max-min), so that each point lies between 0 and 1.

DBSCAN clustering

We used DBSCAN clustering to examine the data. DBSCAN clustering is a machine learning algorithm that finds clusters of points within data to find patterns. It is an unsupervised learning model.

DBSCAN uses 2 hyperparameters: radius, the distance around each point used to locate other points; and minimum sample size, the number of points within this radius that are considered. DBSCAN first examines each point in the data set, seeing if the minimum sample size of points are within the radius of that point. If this is the case, then this point is considered a core point.

When each point has been classified as either a core point or a non-core point, the algorithm will begin forming clusters. It begins with a random core point. It will examine each point within the radius of that point. If a point within the chosen point's radius is a core point, that point is added to the cluster. This process is repeated for each point within the cluster, until every core point within the radius of a core point within the cluster has been added to that cluster. When that is finished, every non-core point within the radius of a core point within the cluster will be added to that cluster as well. At this point, the first cluster is formed.

The process will continue for every remaining core point not within a cluster, not adding points already in a cluster to another cluster. When every core point is in a cluster and each cluster is fully formed, any non-core point not within a cluster is considered an outlier.

Our Model

Our model uses the loss function defined below to determine whether a given radius and minimum sample size do a good job at clustering the data into the categories we want.

Loss function

Let O_P be the number of outlier gammas over total gammas

Let C_r be the cluster ratio (gammas in cluster over cluster size)

Let C_S be the cluster size

Let C_C be the cluster count (total numbers of clusters)

Then we define the loss function as, for n points:

$$\mathcal{L}_S(X, y) = \lambda_1 O_P + \lambda_2 C_C + \lambda_3 \frac{(C_r(1 - C_r) \times 4)C_S}{n}$$

Explanation

$\lambda_1 O_P$: We ultimately want to find clusters of gammas, so if the outliers are all hadrons, we don't care

$\lambda_2 C_C$: We want to keep the cluster count low

$\lambda_3 \frac{(C_r(1-C_r)\times 4)C_S}{n}$: The equation $C_r(1 - C_r)$ means that for each cluster i, the output

of the equation will be larger the closer the cluster i is to being a ratio

of half gammas half hadrons. We want our clusters to be either all

gammas or all hadrons. The rest of the equation means that

larger clusters will have a larger effect on the output.

Choosing lambda

Since our main priority is to put each gamma into a cluster, λ_1 is the largest. After that is λ_3 since we want the ratio of gamma to hadron points to be as low as possible. Finally, since we care the least about the number of clusters, λ_2 is the smallest scalar. We chose $\lambda_1 = 200$, $\lambda_2 = 0.05$, λ_3 =100.

Estimating Hyperparameters

The elbow plot at the bottom of the Jupyter notebook uses K-means clustering to show how within-cluster variation changes with the number of clusters. We use this plot to estimate a reasonable range for the DBSCAN radius parameter by examining how far points typically deviate from their cluster centers.

We then perform individual model tests using radius values from this range to find which minimum sample sizes are appropriate. Once we identify good ranges for both hyperparameters, we can use optimization techniques to select the combination of radius and minimum sample size that minimizes the loss function for our model.

Results

Tests

We first tested the data on different radiuses and sample sizes to get an idea of what range would be ideal. Then we performed DBSCAN clustering and ran the loss function on every combination of radius and minimum sample sizes to find the combination that gives the lowest value for the loss function. Depending on the radius step size that was used, different results were found. With a radius step size of .05, we found the optimal radius to be 0.125 and minimum samples size to be 17. With a radius step size of .02, we found the optimal radius to be 0.95 and minimum samples size to be 13. So, there are various combinations of hyperparameters that work well on this data.

We ran this again for smaller subsets of the data using a variation of cross validation. Regular cross validation could not be used since this is unsupervised learning. We separated the data into 5 random groups and ran the data 5 times, each time leaving a group out of the testing. The step size used was .05 since the computation would take too long if it was any smaller. We averaged the results for each group. We then scaled the minimum sample size up to account for density when comparing it to the optimal hyperparameters for the larger group. We found similar hyperparameters (radius of 0.125 and minimum sample size of 18) for smaller sample sizes compared to larger sample sizes for the same radius step size when scaled for comparison. Both have similar loss function and accuracy values as well.

Accuracy

To calculate the accuracy of the model, we took (gamma points within a cluster + hadron point outliers – hadron points within a cluster)/total points. We found that the hyperparameters of a radius of 0.125 and a minimum sample size of 18 and 17 both had an accuracy of roughly 77%. We can assume that this is the highest accuracy we can get from this data.

Heatmaps

At the very bottom of the jupyter notebook are heatmaps of the data. Heatmaps compare the loss function at each radius and minimum sample size that was observed. Heatmaps are shown for all the data and for each subset that was tested separately. The heatmaps show in yellow which radius and minimum sample size values minimize the loss function. There are many different combinations of hyperparameters that could be used for this function that could minimize the loss function.

Summary

In this project, we applied the DBSCAN clustering algorithm to a simulated dataset representing gamma ray and hadron events detected by a Cherenkov telescope. By normalizing the data and designing a loss function, we were able to identify optimal hyperparameters that resulted in meaningful clusters. These clusters helped separate gamma ray events from hadron events with approximately 77% accuracy.

A possible problem point of my approach was the limited number of tests performed using different sample sizes and radius step sizes (I believe my partner performed more tests). This may have restricted my ability to fully explore the hyperparameter space and identify more precise trends in the data. While I believe that accuracy significantly higher than 77% is unlikely, further testing could provide a clearer understanding of the data structure.

References

UCI Machine Learning Repository. (n.d.).
https://archive.ics.uci.edu/dataset/159/magic+gamma+telescope