Evaluating Unsupervised Clustering Accuracy

# Introduction

One of the most consistent guidelines to succeed with a data science project is to tap into subject matter expertise. As a former Surface Warfare Officer in the US Navy, I spent over four years operating on ships and am familiar with ship navigation and operations. Additionally, I am always interested in Geospatial Information Systems (GIS) so I choose a project that would combine my love of all things maritime and GIS.

Unsupervised clustering is often explained as a key field in machine learning, but is often an after-thought in a world of easy-to-build, high-powered neural networks. However, I argue there are many viable use cases for straight-forward but effective unsupervised clustering approached, particularly in areas where these is a large amount of data and only a small portion can be accurately labeled.

I argue that one of these use cases is dense, geospatial data where we may only care about activity near certain areas. Often we know where some many of those areas are, but it’s difficult for a human or a machine to find new, similar areas in dense geospatial data. If we can build an unsupervised clustering methodology that identifies the known locations with a high level of accuracy, we can assume that the same methodology with the same hyperparameters can find other, similar areas.

A major limitation with unsupervised clustering is evaluating how well a clustering approach performed in grouping “like with like.” Evaluating decision boundaries and comparing means and variances across groups can give a good idea how precise an unsupervised clustering algorithm separated observations, and have been used in many cases to get impactful, real-world results.

However, when using unsupervised clustering to group observations into groups that exist in the real world, it’s difficult to compare these groups to unlabeled data. Since unsupervised clustering approaches like K-means, hierarchical clustering, and DBSCAN will cluster observations together in groups that are numbered as they are identified, how can you tell if these groups correspond to the real-world groupings you are trying to achieve. For a concrete example with our AIS data, how can we tell if a model’s results identified ships’ positions at a port, or if the cluster is in the middle of the ocean? Individual clusters can be manually reviewed, but how can one execute this at scale, with millions of points and thousands or tens of thousands of clusters?

To accomplish this, we need an effective approach to evaluate an unsupervised clustering model’s accuracy across many different hyperparameters. I propose a new methodology that will set three different accuracy metrics for each unsupervised clustering model and then learn the hyperparameters that minimize the errors across these metrics.

# Data Description

This project analyzes shipboard Automatic Identification System (AIS) activity as collected and shared by the US Coast Guard. AIS is an automatic tracking system that uses transponders on ships and is used by ships and port authorities to avoid collisions and manage ship traffic. Because of the tremendous size of the data, I have limited my analysis to a section of data from January 2017 that includes 500 million individual position reports from 20,731 unique ship identifiers, known as an Maritime Mobile Service Identity (MMSI). The January 2017 dataset is about 25 gigabytes of data.

The first step was to build an ingest pipeline for the data. The details of this pipeline are beyond the scope of this project, but are available on my GitHub with details about scraping the Coast Guard site and loading the data into a local PostGres database.

The large size of the data complicated most analytic approaches and required some novel approaches to parse. First, using PostGres’ GIS extension PostGIS, I reduced each individual ship’s track into one line segment to represent the trip. In the ‘ship\_trips\_analysis’ notebook located on my GitHub, I analyzed the ships’ trips and additional metadata such as length of trip, number of days detected in the data, and number of position reports to select a smaller but representative sample from the data. This also helped to eliminate ships that never moved much, or seemed to have erroneous position reports from incredibly long distances. This smaller sample size includes 200 individual ships and just over 2 million unique position reports and will serve as our dataset for this project.

# Accuracy

To help identify activity near known ports, I downloaded and ingested the World Port Index’s list of ports worldwide. Once this data was in the PostGres database, I used PostGIS’s ST\_Distance function to find the nearest port to each of the ship positions, and labeled each point with that port’s name if that position was within set distances of that position. I created a separate table labeling all such positions 2, 5, and 10 kilometers away for each distance. This data will be used later in one of the metrics to validate the unsupervised clustering.

To actually cluster the positions in an attempt to find ports, I used Density-based spatial clustering of applications with noise, or DBSCAN. Originally I hoped to use Scikit-Learn’s implementation. But unfortunately because of the size of the data, DBSCAN clustering of the 2 million positions consistently crashes my Python kernel in Jupyter Notebook and Spyder. Instead, I used PostGres’ native DBSCAN algorithm which does not need to hold all of the data in memory. I used both my own local PostGres instance for validation and a PostGres cluster stood up in Amazon Web Services (AWS).