Auto-Separation of Magnetospheric Regions via Unsupervised Learning

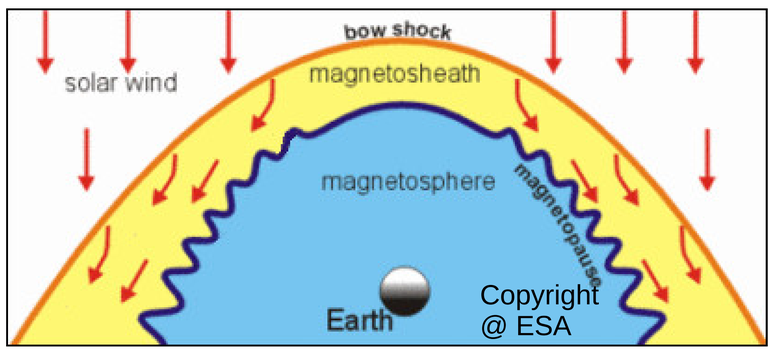
James “Andy” Edmond[1], Joachim Raeder[1], Banafsheh Ferdousi[3], Maria Elena Innocenti[2], Matthew Argall[1]

1: University of New Hampshire, Durham, NH, USA | 2: Ruhr-Universität Bochum, Bochum, NRW, Germany | 3: Air Force Research Lab, Albuquerque, NM, USA

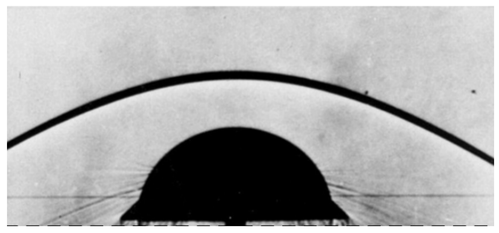
1. Introduction

Earth’s magnetosphere is composed of several distinct plasma regions: the **solar wind** (fast, cold plasma), the **magnetosheath** (heated, slowed, dense, diverted plasma), and the **magnetosphere** (magnetically strong, very hot plasma). Identifying spacecraft crossings between these regions is often easy but time-consuming. We aim to automate the identification of these crossings using a combination of *unsupervised* methods. Specifically,

1) PCA to find embedded and uncorrelated variables,
2) Self-Organizing Maps to find “representative” points of the embeddings
3) Hierarchical Clustering to cluster the representative points



is similar to...



Copyright
@ NASA

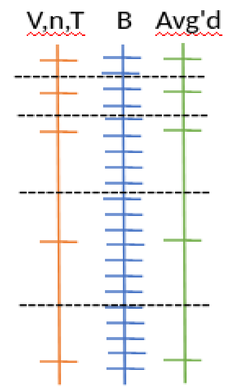
2. Data Source and Preprocessing

We use data from 2 missions: THEMIS (Time History of Events and Macroscale Interactions during Substorms) and MMS (Magnetospheric MultiScale mission)

Input variables:
- O Magnetic field vector **B** (3 comps + 1 magnitude)
- O Ion Velocity **V** (3 comps + 1 magnitude)
- O Log10-Abs Ion Momentum Density n * **V** (3 comps)
- O Log10 Ion density n (1)
- O Log10 Ion temperature T (1)

MMS:
- O Measurements available at high resolution (150 ms or less), so trivial to avg down to 1 min
- O MMS 1,2,3 (2016 – 2021), MMS 4 (2016 – June 2018)
- O 4.09 M pts

THEMIS:
- O Ion measurements have changing time resolution (1.5 or 6.5 mins)
- O Magnetic field measurements available at high resolution (~3 sec)
- O Data are avg’d along changing time resolution (see right figure)
- O THEMIS A,D,E (2007 – 2021), THEMIS B,C (2007 – 2009; before moving to lunar orbit)
- O 8.13 M pts



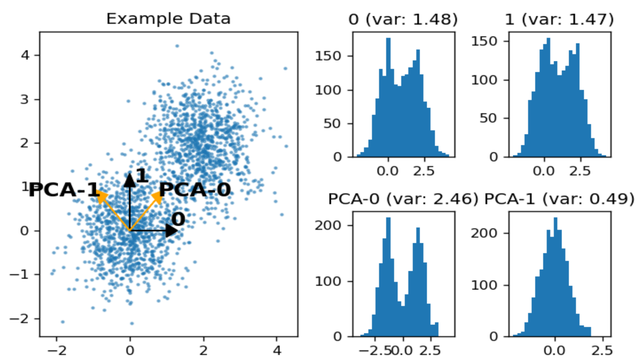
Final Cleaning:
- O Remove close-in / far-out data by requiring 7 R_E <= R <= 35 R_E.
- O 9.6 M pts
- O 5% / 95% Train-Test Split (480K training size)
- O Min-Max rescale according to training data

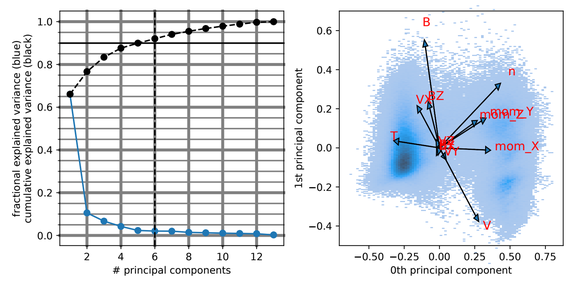
3. PCA

Our dataset has moderate dimensionality (13) and correlated variables. The higher dimensionality can make it harder to correctly cluster outliers and the correlations can introduce bias for certain features. To rectify both of these issues, we will use a popular dimensionality reduction tool, principal component analysis (PCA)

In short, PCA works by…
1) Rotating the original axes along directions of maximal variance
2) Truncating the number of dimensions

Choosing a 90% variance threshold, we select six principal components to extract. The first two represent about 77% of the original variance. Since the components are linear combinations of the original features, we can depict the weights as arrows over the first two components of the training data.





4. Self-Organizing Maps

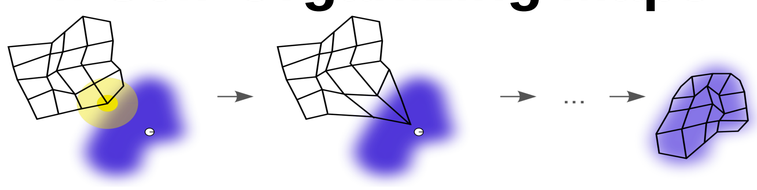


Image credit:
McID – Own Work
CC BY-SA 3.0
https://bit.ly/3BpgABx

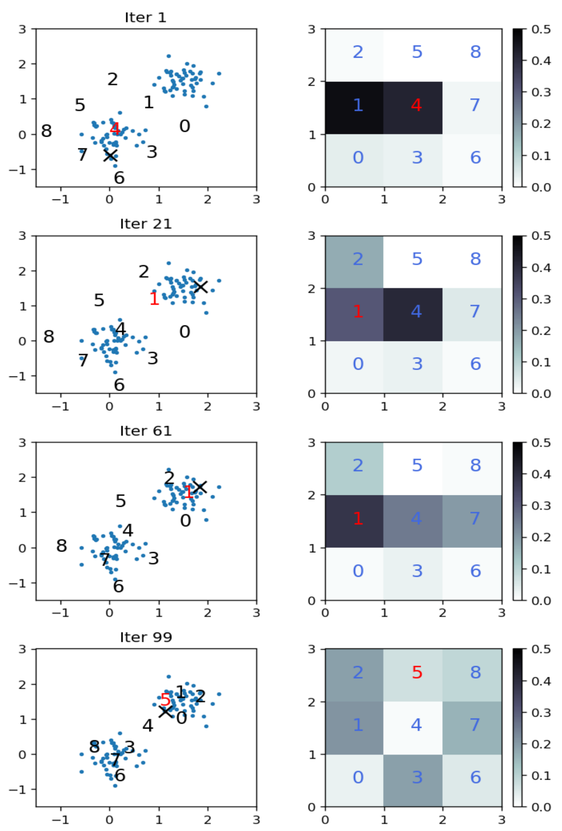
Self-Organizing Maps (SOMs) are an unsupervised neural network that can be used for clustering and dimensionality reduction, enabling 2d visualizations of higher dimensional data. They can be trained to resemble the input data distribution by simultaneously considering the input space of the data and the 2D space of the SOM network. This is done by...

1) Selecting a single point **q** of the input data
2) Calculating the distances of all nodes from **q** (in the input space)
3) Identifying the closest node **BMU** in the map
4) Calculating the distances of all nodes from **BMU** (in the network space)
5) Updating node positions in input-space in proportion to both their distance from **q** and their distance from **BMU**.

$$\mathbf{w}_a(i+1) = \mathbf{w}_a(i) + \alpha(i)h(i,a,BMU)(\mathbf{q} - \mathbf{w}_a(i))$$

To ensure convergence, a maximum iteration number is specified and the learning rate and “neighborhood” size are forced to decay over iterations.

To expedite finding the optimal map, we run K-means with 10k clusters over our MinMax-rescaled PCA-reduced training data and extract the 10k closest points to these centroids. 500 maps with different hyperparameters will be trained on these points and the map’s performance will be evaluated on the remainder of the training set (470k) using the cost function on the right. The hyperparameters of the map with the lowest cost value will be used to train a final map on the remaining 470k points.

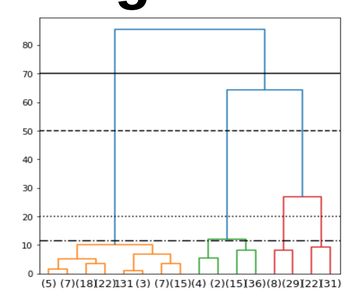


$$L = Q * \left(\frac{n_x n_y}{(n_x)_{max}(n_y)_{max}} + \frac{max\{n_x, n_y\}}{min\{n_x, n_y\}} \right).$$

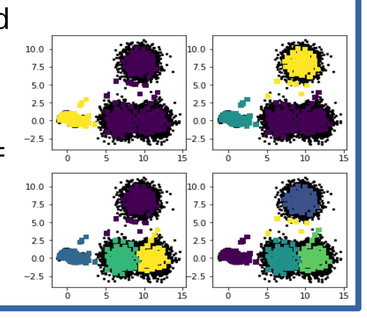
The optimal hyperparameters are:
- O (nx, ny) = (14, 14)
- O σ = 5.518
- O α = 0.843
- O Exponential Decay
- O Ricker Neighborhood

5. Hierarchical Clustering

With a trained map, we can use the node positions as representative points of the data and apply clustering algorithms to them. The classifications assigned to the nodes are then propagated to the data they represent. Since the map is small (~200 nodes) and acts as a sort of “interface” between a clustering algorithm and the data, we can use a wider variety of clustering methods, even including transductive ones!

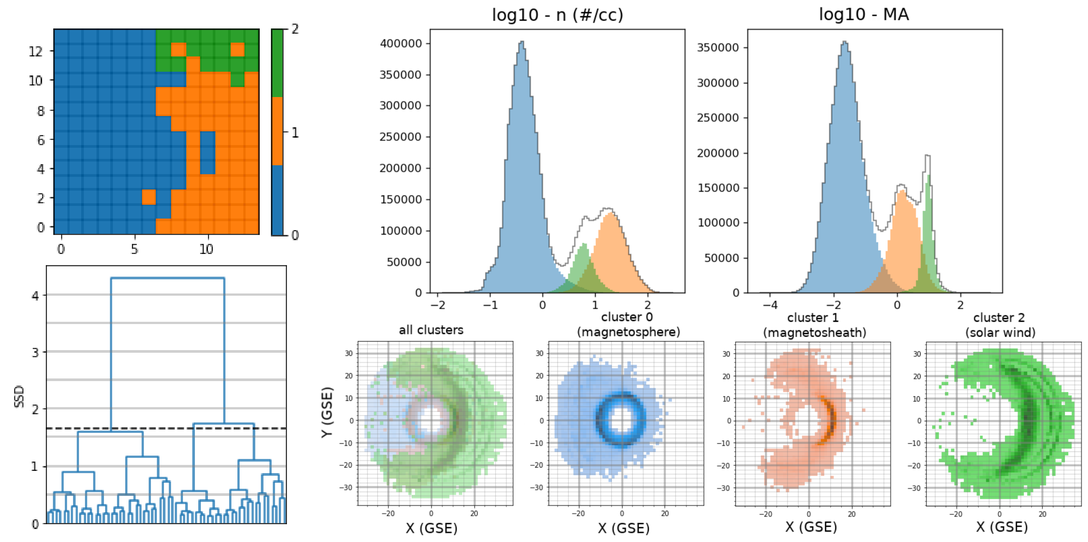


We use agglomerative hierarchical clustering with a ward linkage to find clusters based on minimization of variance. This works by …
1) Initially treating each node as its own cluster
2) Merging the two most similar clusters (“most similar” = minimal distance with linkage)
3) Repeat (2) until all data has been merged into single cluster.

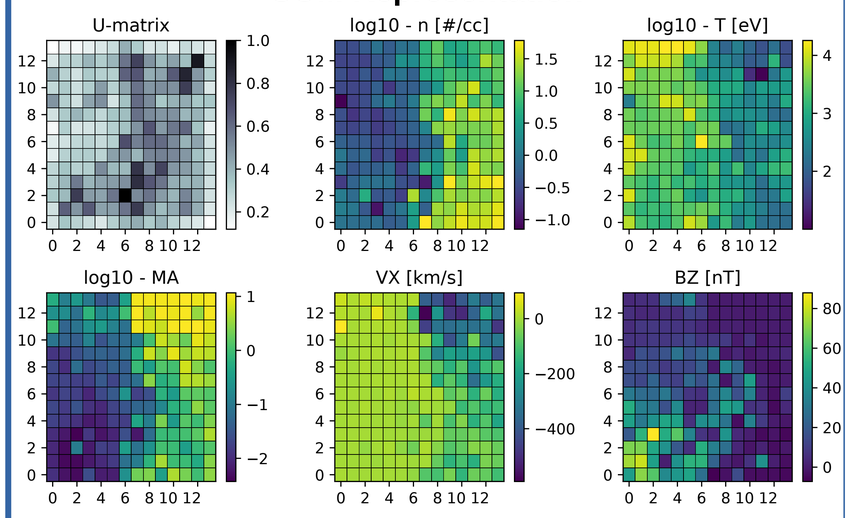


6. Results

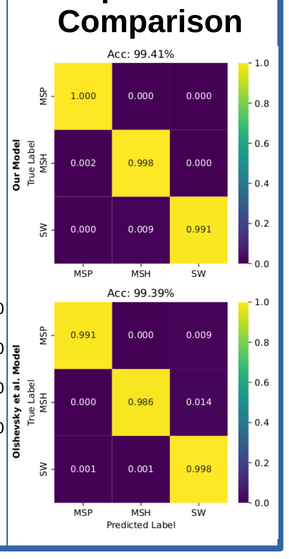
Clustering Visualization



SOM Representation



Supervised Comparison



Want to easily pip-install and use this model? See my repo at: github.com/jae1018/GMClustering

SCAN ME

Get predictions in 3 steps:
Import: from gmclustering import GMClustering
Instantiate: gmc = GMClustering()
Predict: gmc.predict(data)

University of New Hampshire