Auto-Classification of Global Magnetospheric Data using Unsupervised Learning

Paper #: NG13B - 0615

James “Andy” Edmond[1], Joachim Raeder[1], Banafsheh Ferdousi[1], Maria Elena Innocenti[2], Matthew Argall[1]
1: University of New Hampshire, Durham, NH, USA | 2: Ruhr-Universität Bochum, Bochum, NRW, Germany

Contact Email:
james.edmond@unh.edu

I. INTRODUCTION

Earth’s magnetosphere and its surrounding area is composed of several regions, being the solar wind, the magnetosheath, and the magnetosphere itself. The magnetosphere acts as an obstacle to the supersonic solar wind flow where the magnetosheath is the shocked flow diverted around the magnetosphere. Determining which region spacecraft measurements occur in is generally not difficult and we automate this process using a combination of unsupervised methods, including clustering, wherein data are separated into homogeneous groups.

The noise present in observations and the changes in measurements occurring across spacecraft make direct applications of unsupervised methods difficult. Moreover, many clustering methods are prohibitive of large dataset sizes, but the applicability of unsupervised methods can be extended by using a combination of methods. Using data from the THEMIS and MMS missions and Self-Organizing Maps, we have created a model that can predict the corresponding region of measurement for THEMIS and MMS data.

2. DATA SOURCE

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

Input variables:
- O magnetic field (Bx, By, BZ), GSE [nT]
- O Ion velocity (Vx, Vy, Vz), GSE [km/s]
- O Ion density [#/cc]
- O Ion temperature [eV]

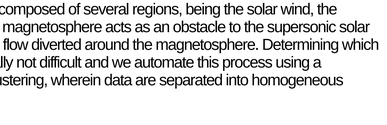
MMS:
- O Ion measurements from Fast Plasma Investigation at 150 ms res
- O Magnetic field measurements from Flux Gate Magnetometer at (10 ms) res
- O Avg’d down to 1-min
- O MMS 1,2,3 (2016 – 2021), MMS 4 (2016 – June 2018)
- O 4.09 M pts

THEMIS:
- O Ion measurements from ElectroStatic Analyzer “full” packets – with changing time resolution between 1.5 and 6.5 minutes.
- O Magnetic field measurements from Flux Gate Magnetometer at spin resolution (~3 sec)
- O Data are avg’d according to floating ESA-Full 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 .

3. PRE-PROCESSING

These original eight features alone do not possess enough variance for many unsupervised methods to sufficiently cluster the data. We will create additional features to express nonlinear relationships in a linear way

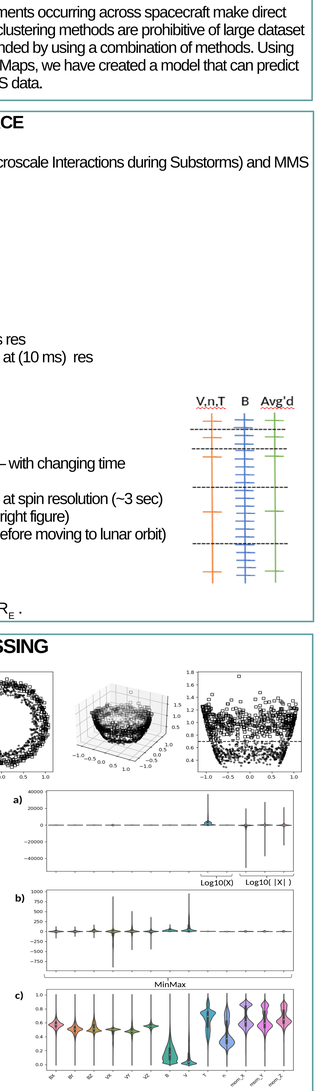


New features:
- O magnetic field and ion velocity magnitudes
- O Ion momentum density vector (n * **V**)

The density and temperature (and the derived ion momentum density) features cover a large range, so we will rescale these using a log10(x) and log10(abs(x)) transforms.

To remove bias against features with large magnitudes, we rescale all data wth a MinMax scaling (i.e. modifying the min and max of each feature to be 0 and 1, respectively).

Finally, we do a train – test split of 5 to 95 %. This gives us a test and train set size of 9.15M and 480K points.

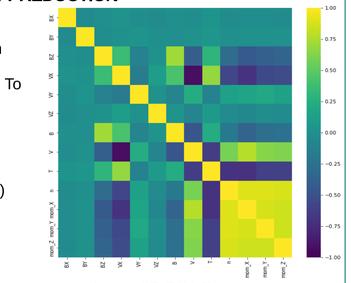


4. DIMENSIONALITY REDUCTION

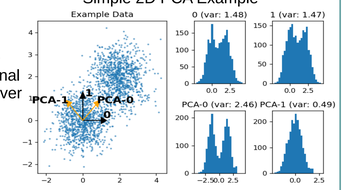
Our dataset has moderate dimensionality (13) and correlated variables. The higher dimensionality can make it harder to correctly cluster outliers and the correlations can introduce bias for certain features. To rectify both of these issues, we will use a popular dimensionality reduction tool, principal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

In short, PCA works by…
1) Computing the eigenvalues (λ), eigenvectors (e) of covariance matrix where λi ∝ amount of variance along eir
2) Pick smaller number of dimensions d < original dimensionality D that represents majority of the variance.

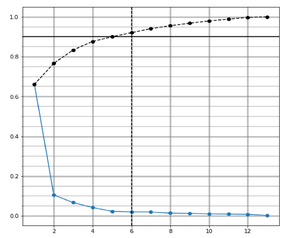
Choosing a 90% variance threshold, we select six principal components to extract. The first two represent about 77% of the original variance. Since the components are linear combinations of the original features, we can depict the the weights as arrows over the first two components of the training data.



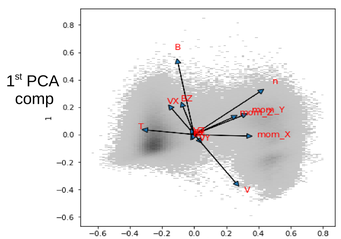
Simple 2D PCA Example



Cumulative explained variance (black)



components (x-axis) /
Explained variance ratio per comp (blue)



5. SELF-ORGANIZING MAPS

Both the dimensionality and multicollinearity have been handled, but now there is the matter of the training size (500K pts). A large number of clustering algorithms exist with their own strengths and weaknesses, and many can struggle with large training sizes or are transductive (i.e. difficult / impossible to make predictions on data outside of the training set). To expand the amount of clustering algorithms available to us, we will use a method of vector quantization to get a greatly-discretized representation of the training data. This method is a Self-Organizing Map

A Self-Organizing Map (SOM) is a unsupervised neural network that transforms higher dimensional data into a 2D grid. A SOM does this by having each neuron, or “node”, simultaneously occupy a position in the data (“feature-space”) and be connected to other nodes in the network (“node-space”). In short, a SOM is trained to represent data by …
1) presenting a data point q to the network
2) Identifying the closest node in feature space (called the best-matching unit, or BMU)
3) Moving the BMU closer to q so that it "better represents" q
4) All other nodes are moved closer to q in proportion to both their feature-space distance to q and their node-space distance to the BMU. Nodes that are "in the neighborhood" of the BMU are moved closer (where “in the neighborhood" is quantified with a neighborhood function - commonly a gaussian)

At each iteration, the position (or “weight”) of each node is update according to

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

Convergence during training is guaranteed by specifying …
- O Max number of iterations
- O Decay Function (e.g. linear, exponential)
- O Starting / Ending values for learning rate (α) / neighborhood size (σ)

For our data, we trained a SOM using the XPySom python package using the hyperparameters:
- O (14 x 14) map size
- O σ = 5.518
- O α = 0.843
- O Exponential Decay and Mexican-Hat Neighborhood Function

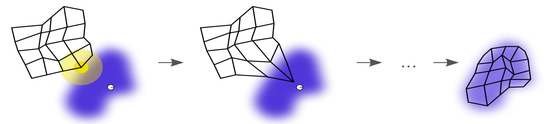
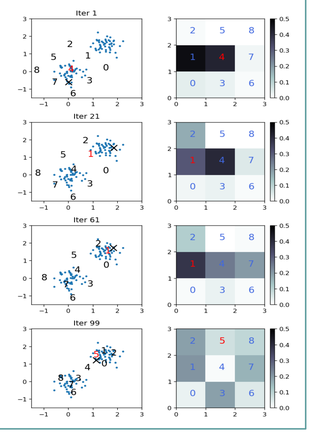


Image credit:
McID – Own Work
CC BY-SA 3.0

https://bit.ly/3BpgABx



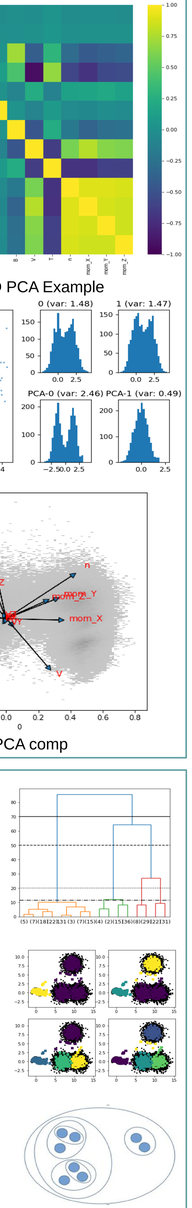
6. CLUSTERING OF SOM NODES

Using multiple methods, we have altered the training data from high data size (500K pts x 13 dims) with correlated features to low (14^2 = 196 nodes x 6 dims) with uncorrelated features. This enormously extends the choice of clustering methods available for use. We will use a hierarchical agglomerative clustering method to cluster the nodes directly.

Hierarchical agglomerative clustering works by …
1) Assuming every node starts as its own cluster
2) Merging the two most similar clusters.
3) Repeat (2) until all data is merged into a single cluster

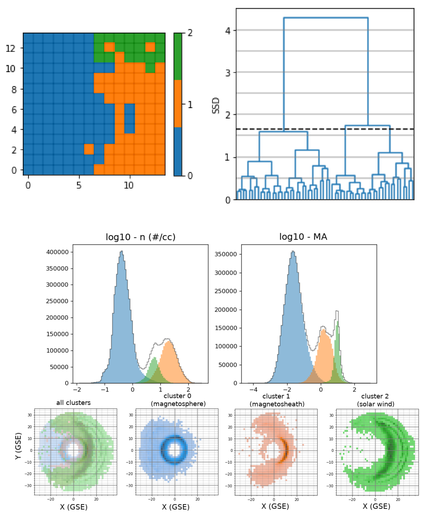
“Similarity” is quantified using a linkage function. We use a ward linkage, which defines the similarity between two clusters A and B as the variance of the merged cluster.

Note that while hierarchical clustering is a transductive method in general, certain linkages might be more amenable to getting prediction on data not seen during training. Regardless, new data can be represented with the same map and the hierarchical clustering of the map can be propagated to the data. By using a SOM to act as an interface between the data and the clustering algorithm, we are able to use a transductive method in an inductive manner.

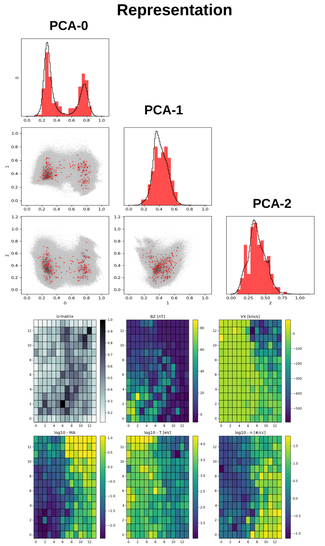


7. RESULTS

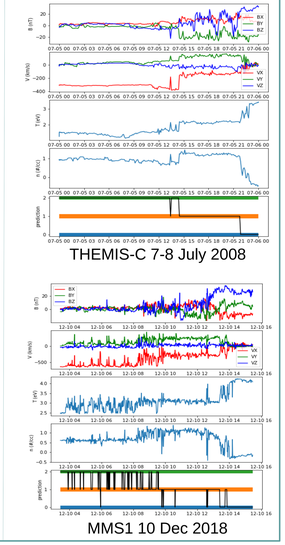
Clustering Visualization



SOM Representation



Time Series Predictions



AGU23
San Francisco, CA & Online Everywhere
11-15 December 2023

11 Dec 2023

This work has been submitted to:
JGR: Machine Learning and Computation

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
RUB