Using Multi-Stage Unsupervised Clustering to Automatically Separate Plasma Regions in the Dayside Magnetosphere

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 dayside magnetosphere is composed of several regions, such being the magnetosphere, the magnetosheath, and the solar wind, and their boundaries, the magnetopause and the bow shock. Individually identifying crossings from data between these regions is often easy but time-consuming, and so it would be beneficial to automate this process. We do so by separating the regions using unsupervised methods.

However, the noise present in even cleaned spacecraft data can make direct application of unsupervised methods somewhat challenging. Results will generally be good… but not great. Improving performance requires a reduced-representation of the data that accounts for variance in measurements. This representation can be created using Self-Organizing Maps (SOMs).

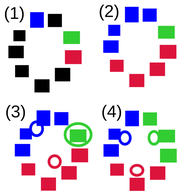
II. SELF-ORGANIZING MAPS

SOMs create a N-D → 2-D representation that maps out the density of the data. It is analogous to binning data in N-D… but if the bins themselves could “compete” for points. An deeper explanation of it requires understanding a simpler clustering algorithm: KMeans

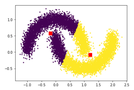
Simple Model: K-Means

K-Means assumes you want to find *k* clusters in data:
In short ...

(1) k data points are selected at random to act as cluster centers.
(2) Points are assigned to the cluster they are closest to.
(3) Cluster centers are re-computed as the average over all data in the cluster
(4) Steps (2) and (3) are repeated until convergence to a **local** optimum



K-Means can work well in many applications… but the nature of a small number of cluster centers to compete for points can lead to artificial clusters



Upgrade: K-Means → SOM

Let’s upgrade K-Means by adding 5 concepts:
(1*) Use **many** clusters (very often > ~ 30)
(2*) Allow clusters to be dependent on adjacent ones – when a cluster center moves to match data, let it pull others nearby along with it
(3*) Keep clusters topologically ordered (e.g. if 1-D clusters A, B, C exist such that A < B < C at the start – then keep it that way at all times)
(4*) Consider one data point at a time instead of all at once.
(5*) How much clusters move should diminish over time (guarantees local convergence)

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

https://bit.ly/3BpgABx

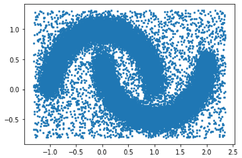
The concepts above are reflected in the choice of hyperparameters for the SOM:
- Choose the number of cluster centers (called ‘nodes’) – by specifying map size to be *m* x *n* (1*, 3*)
- Control how quickly a node moves to a point – by the learning rate alpha and a function representing the decay rate (typically exp or linear decay) (5*)
- Specify the amount of ‘pull’ a cluster can have on its neighbors (2*)

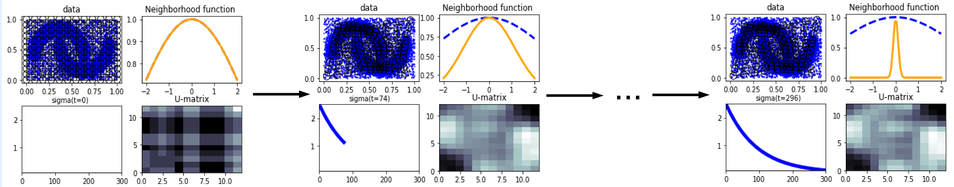
II. SELF-ORGANIZING MAPS (CONT’D)

SOM Visualization

SOMs can be visualized in 2-D by taking advantage of (3*). Computing the average distance between a node and its neighbors and expressing it using a grey-scale colorbar means that regions of high density can be identified in white and sparser regions in black. The matrix of these distances of the map is called the U-matrix.

Here’s an example of the convergence of a 15 x 15 map with sigma = 2.25, learning rate = 0.25, an exponential decay function, and a gaussian neighborhood function.

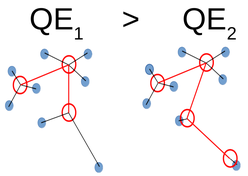




Quantifying SOM’s Representation

A huge combination of hyperparameter choices exist when fitting a SOM to data – but how to choose the best option?

Generally best to choose combo that minimizes the **Quantization Error (QE)…** to within reason. Making an enormous map could really cut down on QE, but will cost you in both memory and run-time.



QE = Σ (black lines)²

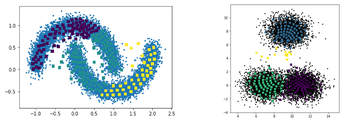
III. CLUSTERING OF SELF-ORGANIZING MAP NODES

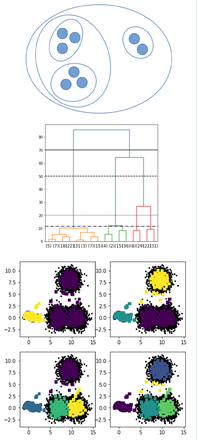
Once the map has converged, other clustering methods can be applied to the node positions. The most popular is K-Means, but the static nature of the converged SOM positions also allows for inductive methods, which is commonly **not** an option for large datasets or models meant to have a test / train split (but sometimes possible for models that support out-of-bag sampling).

We use Agglomerative Hierarchical Clustering with a Ward linkage to cluster the nodes. It is deterministic (not stochastic!), and breaks data into clusters by …
(1) Assuming every node is its own cluster
(2) Merging two clusters that yield the **smallest increase** in variance
(3) Repeating (2) until one cluster encompassing all the nodes remains.

The hierarchical structure allows for investigating sub-structures within clusters!

Caution: SOMs are a **linear** method (but non-linear variants exist)! If you map non-linear data, the distribution of the nodes will (probably) still be non-linear – which means linear clustering methods may not work!





IV. SOURCE DATA

Input variables:
- o Magnetic field **B** vector [nT]
- o Ion velocity **V** vector [km/s]
- o log10 scale - Ion density n [#/cc]
- o log10 scale - Ion temperature T [eV]

Constraints:
- o abs(x-y angle) <= 110 degree (so mostly dayside)
- o 7 <= R <= 40

Data distribution:
- o 2.9M THEMIS points from ESA-Full (2007 - 2020)
- o 2.0M MMS points (2016 – 2021)

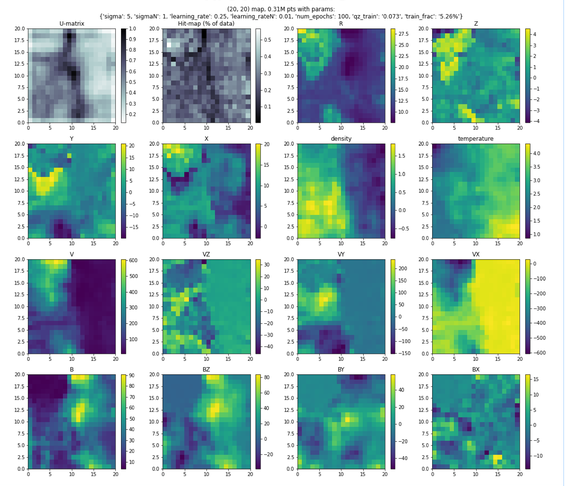
Processing Pipeline:
- o (0,1) feature scale → PCA → (0,1) scale post-PCA

95% / 5% Validation – Train Split

V. SOM PARAMS

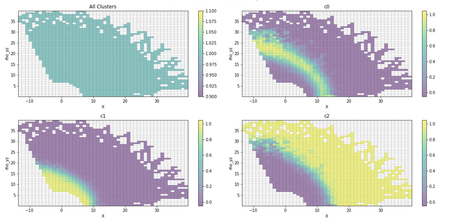
We train our SOM using the following params:
(20 x 20) grid
Start / end σ: 5 – 1 (end > 0.01 to keep map smooth)
Initial / final learning rate: 0.25 - 0.01
Gaussian neighborhood - Linear decay - square topology

VI. SOM RESULTS

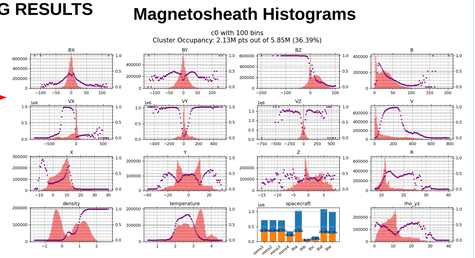


VII. CLUSTERING RESULTS

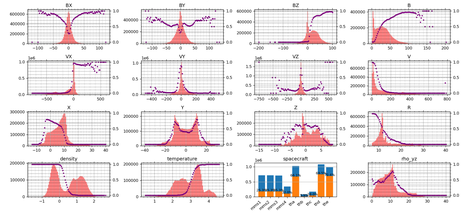
Probability Heatmap



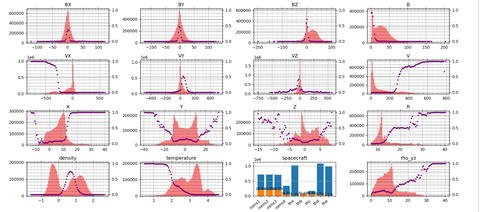
Magnetosheath Histograms



Magnetosphere Histograms



Solar Wind Histograms



Want to learn more about SOMs? See my Jupyter notebook from a talk on Clustering methods at: https://bit.ly/3I9Cqwx

University of New Hampshire