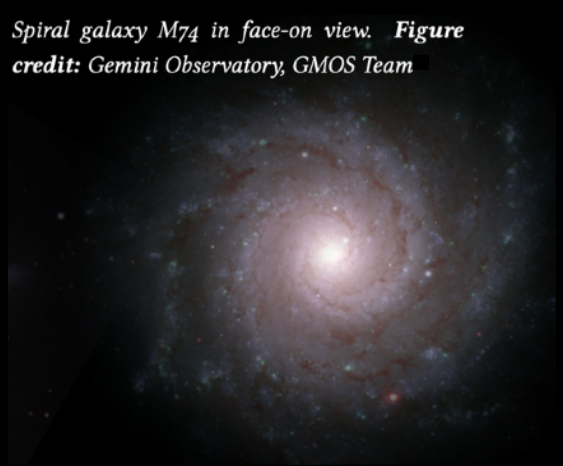
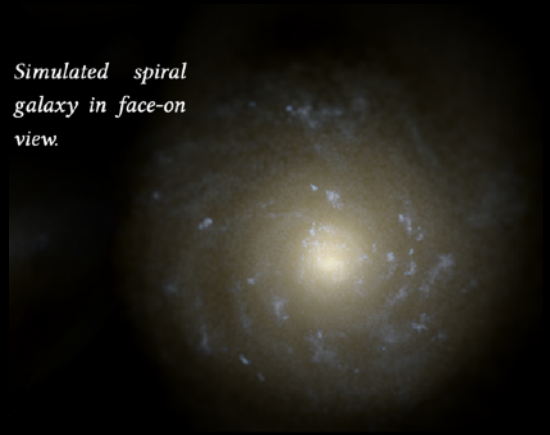
ASTR4004

COMPUTATIONAL ASTRONOMY

Week 8 https://github.com/svenbuder/astr4004_2024_week9



Spiral galaxy M74 in face-on view. ***Figure credit:*** *Gemini Observatory, GMOS Team*



Simulated spiral galaxy in face-on view.

Figure credit: *Tobias Buck*

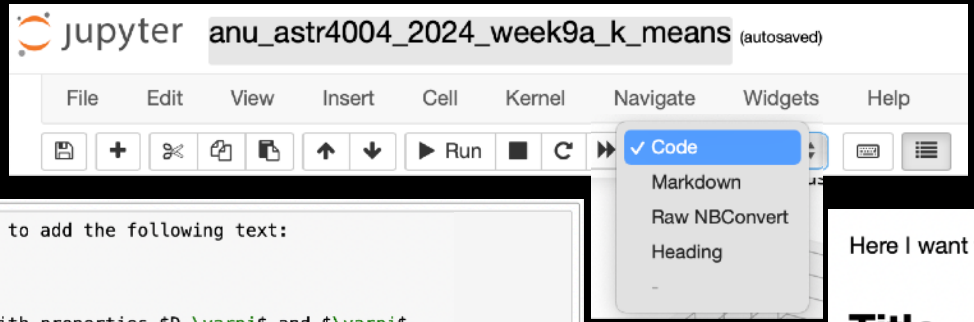
DR SVEN BUDER | RESEARCH SCHOOL OF ASTRONOMY AND ASTROPHYSICS

Australian National University

Adding description in Jupyter notebooks (.ipynb)

Markdown (.md) is an often used hybrid format between text, latex, and html.

It comes in very handy for adding text between code, e.g. for "discussion" tasks of an assignment.



Here I want to add the following text:

Title

Some text with properties D_ϖ and ϖ. I also want to show how to convert between them via:

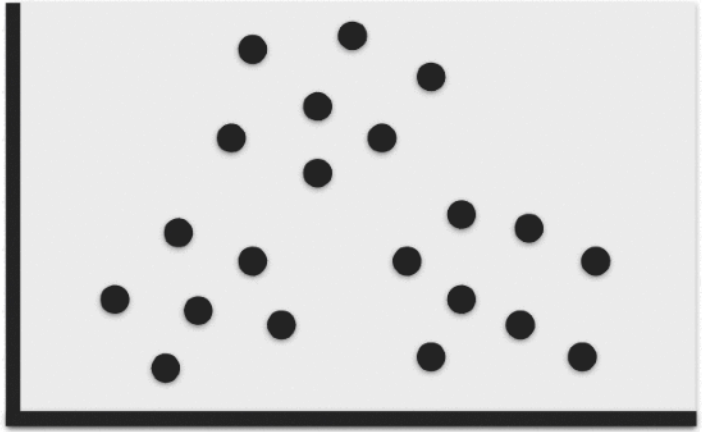
$$D_\varpi = \frac{1}{\varpi}$$

and then continue.

Most GitHub repository and code documentation (e.g. README.md) is written in markdown.

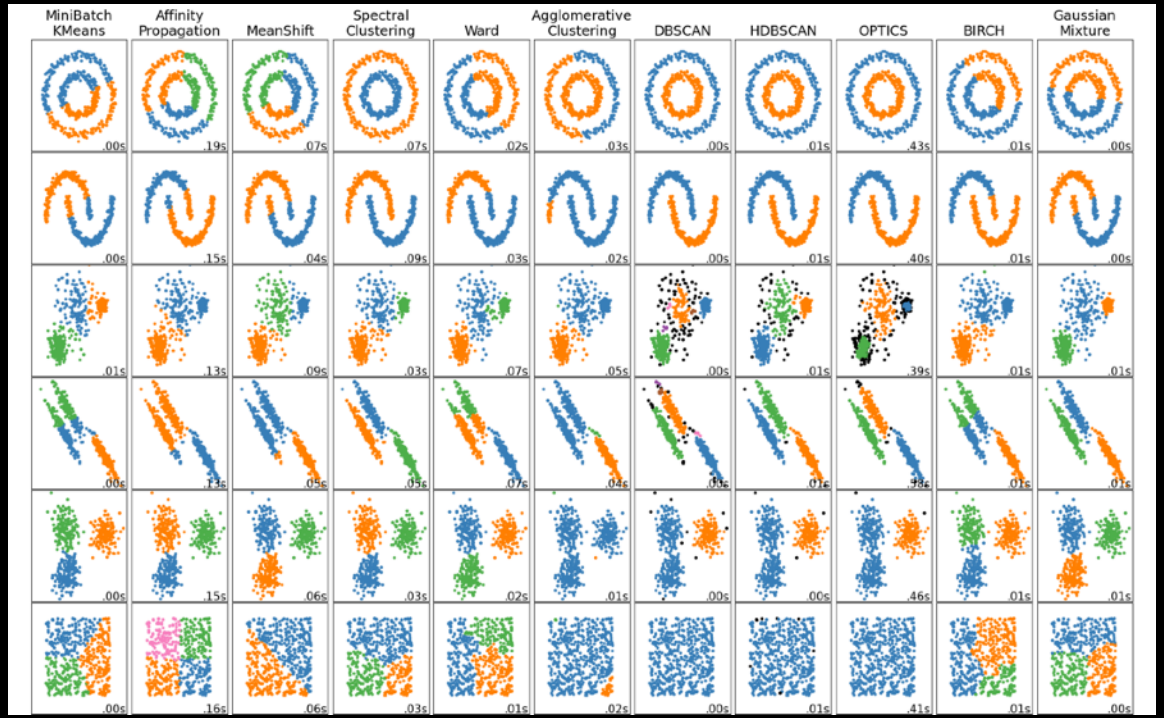
CLUSTERING:

1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



Clustering

You can imagine a lot of examples of easy or difficult clustering problems



Overview from https://scikit-learn.org/stable/modules/clustering.html

Clustering Checklist

- Why am I clustering these data? What do I want to learn or infer from the data?
- Since the clustering algorithm I choose is almost guaranteed not to be representative of the generative model that produced the data, what artefacts in the clustering outputs do I need to be worried about?
- How much can I believe the clustering results? What things can I cross-check to make sure they look sensible?
- Even though you may disbelieve the clustering results, are they **sufficient for my purpose**? That is to say: even if the model is wrong, does it still have utility?

K-means: Unsupervised clustering

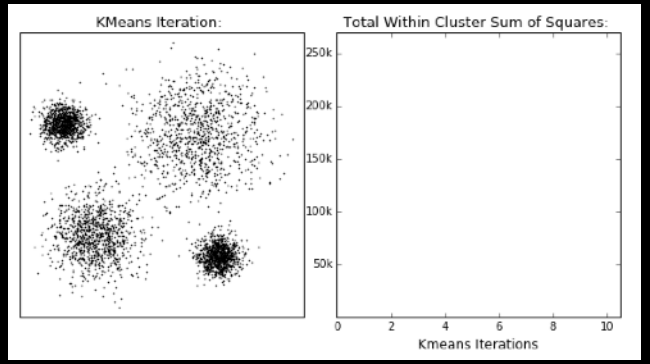
Goal: partition *n* data points into *K* clusters, so that points in the same cluster are “close” to each other and “far” from points in other clusters. We want to minimise the **within-cluster variance** (also called inertia). Formally, given *n* data points $\{x_1, \ldots, x_n\}$ and *K* cluster centres $\{\mu 1, \ldots, \mu_K\}$, we want to minimise:

$$\text{Objective:} \quad J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \, \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_{ik} \, \mathbf{x}_i}{\sum_{i=1}^{n} r_{ik}}$$

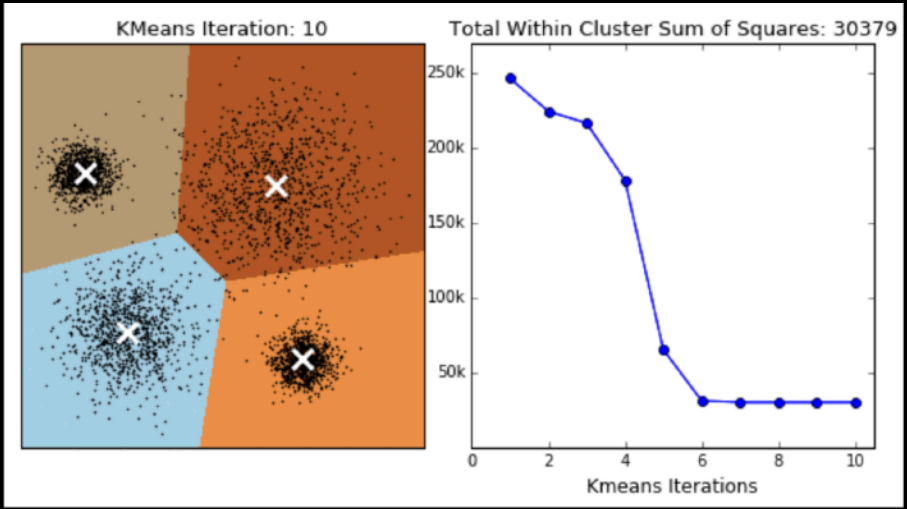
1. Initialise cluster centres μ_j
2. Expectation"E-step": Iteratively assign data points x_i to the closest centroid
3. Maximisation "M-step": Update centroids to the centre of their assigned points

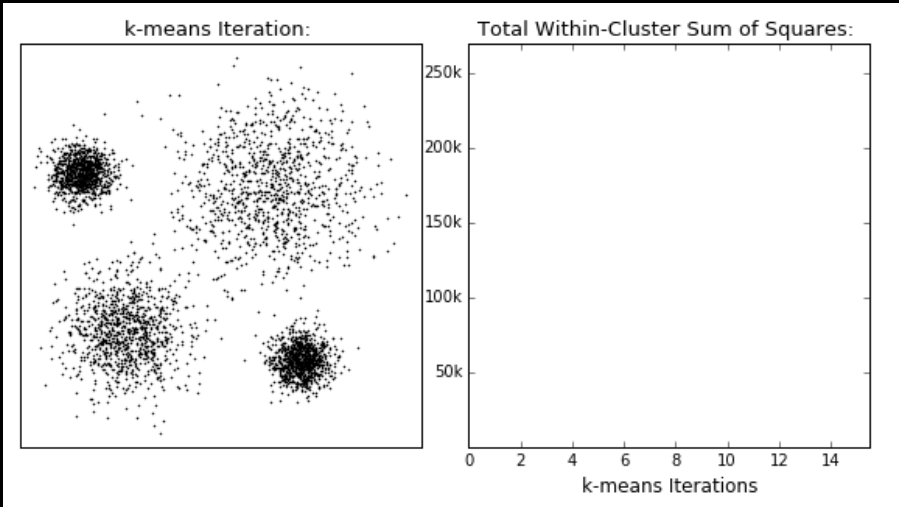


4. Repeat EM: Iterate 2+3 towards more compact & separated clusters until "convergence"

K-means: Unsupervised clustering

k-means algorithms are sensitive to the starting position of the cluster centres, as each method converges to local optima



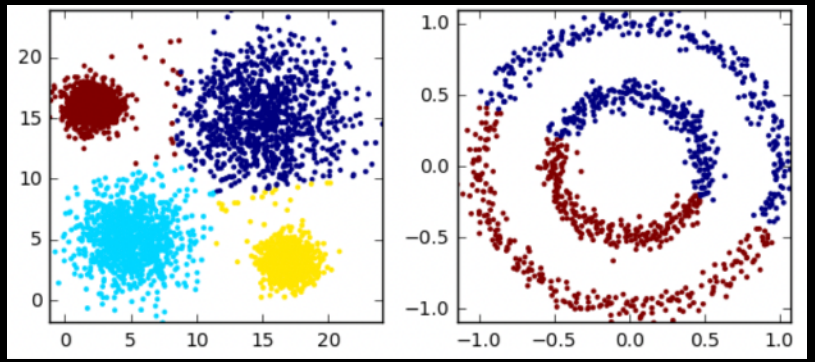


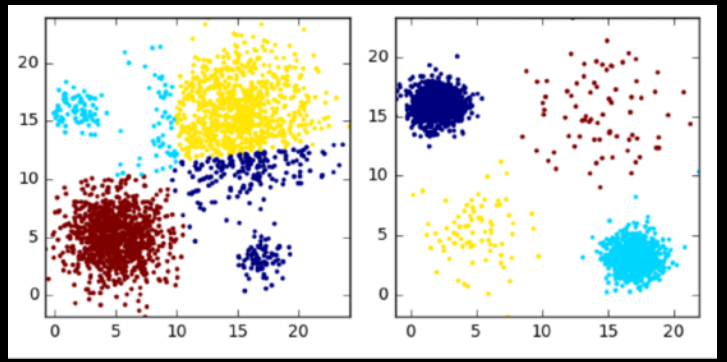
sklearn allows you to initialise "smarter" than random (init='k-means++') and multiple times (**n_init)

K-means: Unsupervised clustering

The algorithm seeks and identifies globular (essentially spherical) clusters.

If this assumption doesn't hold,
the model output may be inadequate (or just really bad).





k-means can also underperform with clusters of different size and density.

But: k-means is one of the least complex and thus fastest algorithms!

GAUSSIAN MIXTURE MODELS

Australian National University

Gaussian Mixture Models

Let's try to describe our data x with *K* d-dimensional Gaussians. Each cluster *k* has:

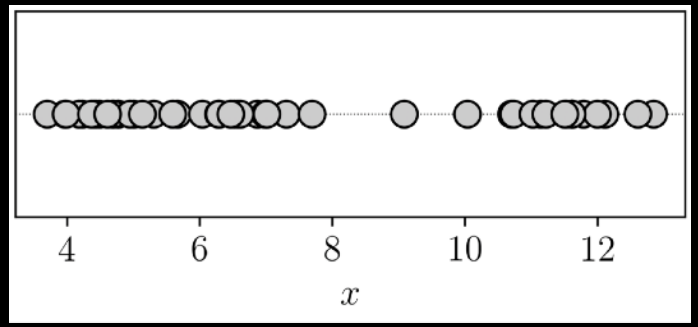
a **mean vector** $\boldsymbol{\mu}_k \in \mathbb{R}^d$

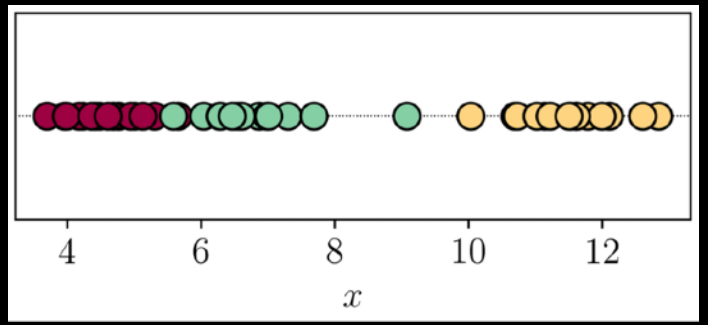
a **covariance matrix** $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d \times d}$

a **mixing weight** π_k (fraction of points in that cluster, $\sum_k \pi_k = 1$)

Probability Density for **x**:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$





Simple case (no covariances) for k=1 (==red)

$$\mu_1 = \frac{1}{N_{\text{red}}} \sum_{i=1}^{N_{\text{red}}} x_i$$

$$\sigma_1^2 = \frac{1}{N_{\text{red}}} \sum_{i=1}^{N_{\text{red}}} (x_i - \mu_1)^2$$

$$\pi_1 = \frac{N_{\text{red}}}{N}$$

Multivariate Gaussian: $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right)$

GMM: Expectation-Maximisation (EM)

1. Initialise cluster centres μ_K, Σ_K, π_K (for example with K-means)

2. Expectation "E-step": For each data point *i* and cluster *k*, compute the responsibility (posterior probability of belonging to cluster *k*):

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i \mid \mu_j, \Sigma_j)}$$

GMMs do a "soft" assignment
(K-means had 0 or 1)

3. Maximisation "M-step": Update parameters using weighted (N_k) averages:

$$N_k = \sum_{i=1}^{n} \gamma_{ik}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{ik} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{ik} \mathbf{x}_i$$

$$\pi_k = \frac{N_k}{n}$$

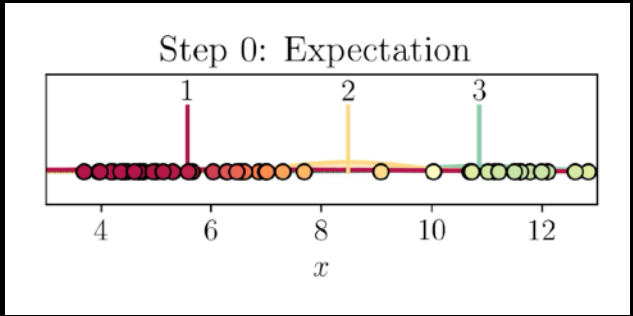
4. Repeat Expectation-Maximisation (EM) until convergence of the (log-)likelihood:

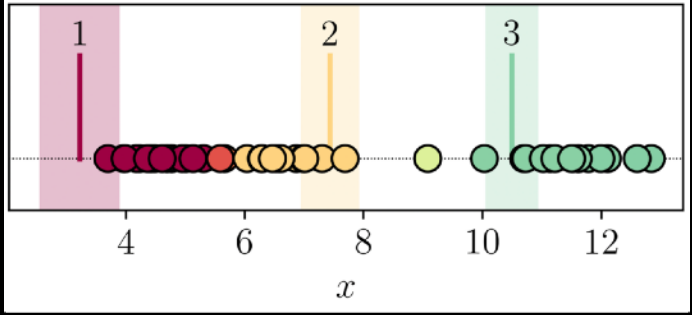
$$\log L = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i \mid \mu_k, \Sigma_k) \right)$$

Gaussian Mixture Models

$$p(\mathcal{M}_k|x_i) = \frac{p(x_i|\mathcal{M}_k)p(\mathcal{M}_k)}{\sum_{j=1}^{K} p(x_i|\mathcal{M}_j)p(\mathcal{M}_j)}$$

posterior probability of membership for belonging to one component, compared to all components





data points in between components have a membership proportion that is split between the components

Expectation-Maximisation (EM):
Alternate steps between expectation which component data point comes from and update of model parameter estimate (maximisation of likelihood)

K-MEANS AND GAUSSIAN MIXTURE MODELS

IN PRACTICE

Australian National University

DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE (DBSCAN)

Australian National University

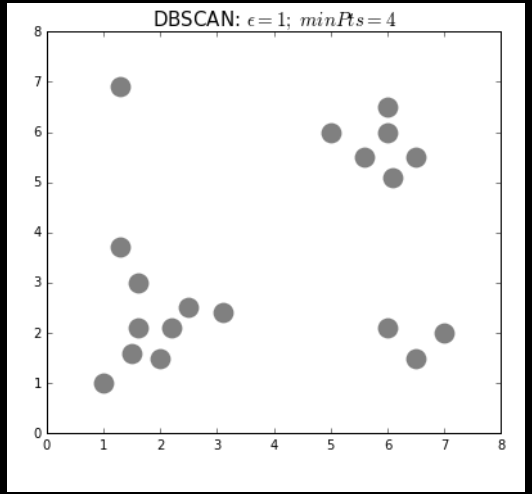
DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE (DBSCAN)

Clusters are considered zones that are sufficiently dense

Points that lack neighbours do not belong to any cluster and are thus classified as noise

Doesn't require the user to specify the number of clusters; it works that out for you

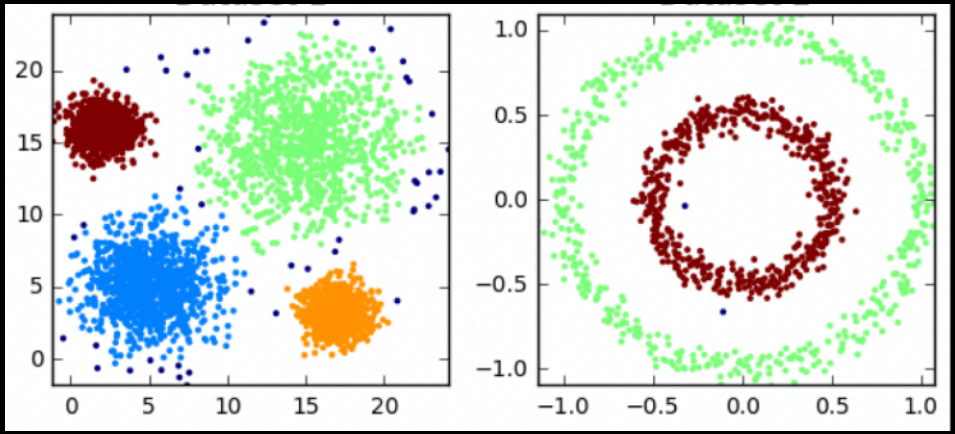
Needs minimum of points that constitutes a cluster (minPts) and the size/radius of the neighbourhoods (epsilon)

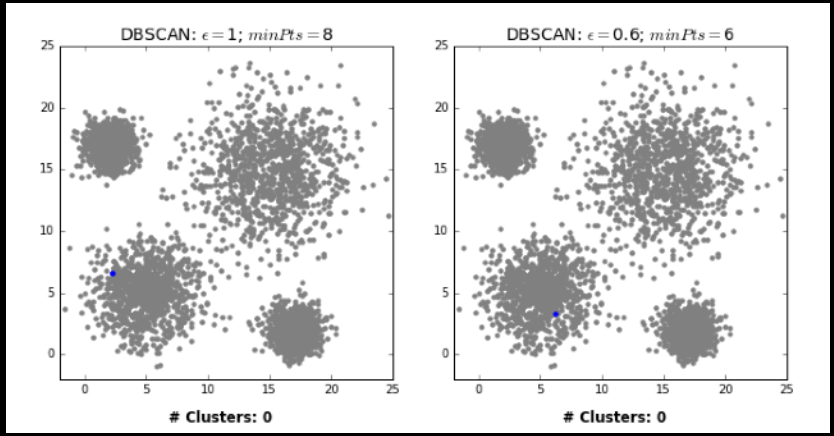


Identifies clusters and then expands clusters by scanning the neighbourhoods

Once all neighbourhoods have been exhausted, the process repeats with a new cluster, until all observations belong to a segment or have been classified as noise

DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE (DBSCAN)





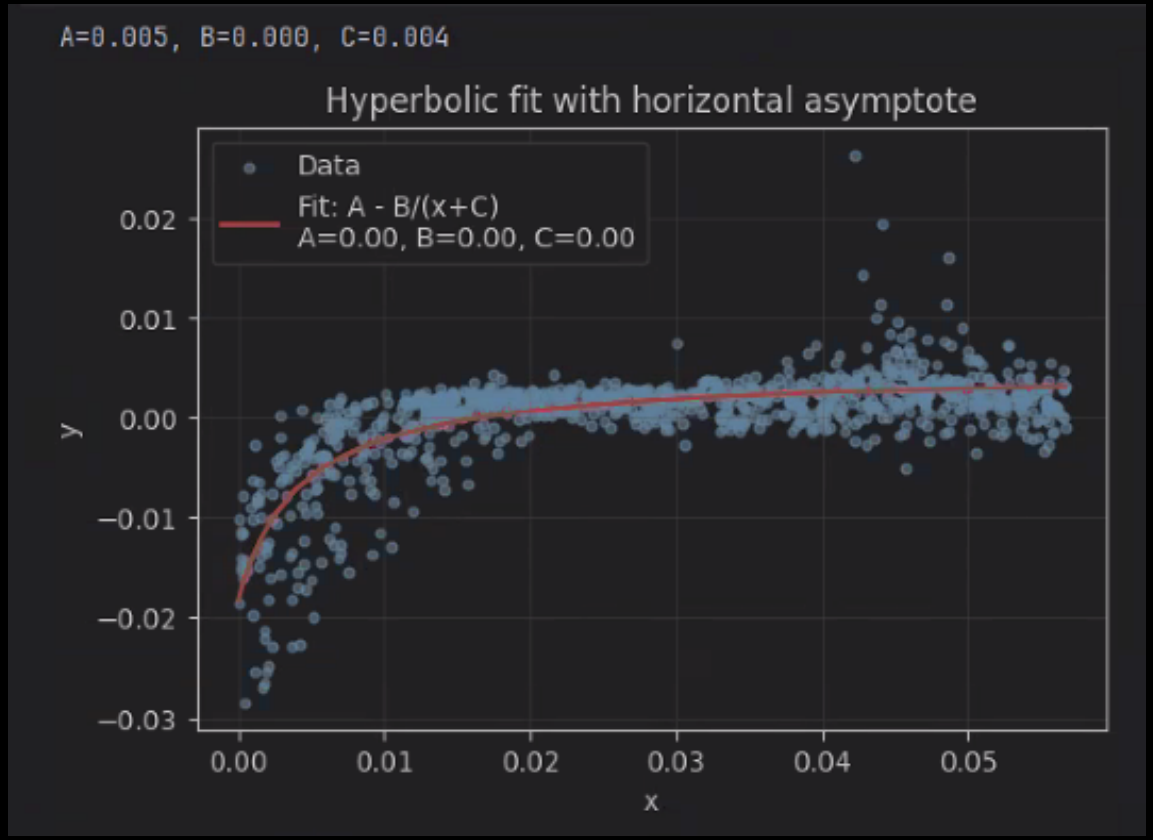
Hierarchical DBSCAN (HDBSCAN), for example, is building a hierarchy of clusters and then condensing it into the most stable ones.

It is continuously varying the density parameter epsilon to detect clusters at multiple density levels (dense points of small clusters -> less dense and larger clusters)

There are millions of other clustering algorithms!

PLOT CLINIC

Australian
National
University



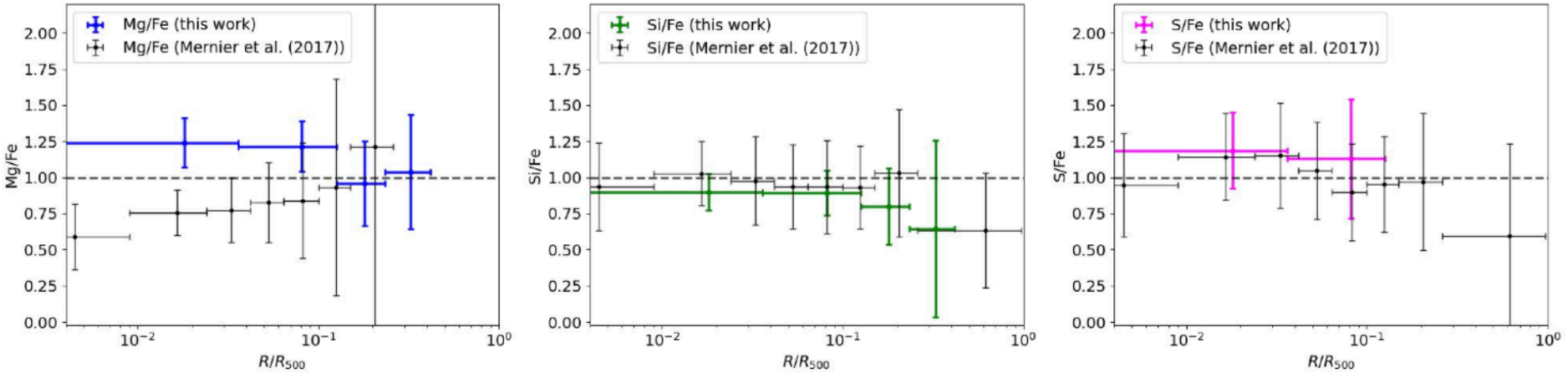


Figure 6. Radial abundance ratio profiles ([Mg/Fe], [Si/Fe] and [S/Fe]) of Mrk1216 and groups average from Mernier et al. (2017).

CAN YOU BEAT THE COMPUTER (PROGRAMMER)?

Australian National University