Physics 361 - Machine Learning in Physics

Lecture 8 – Unsupervised methods

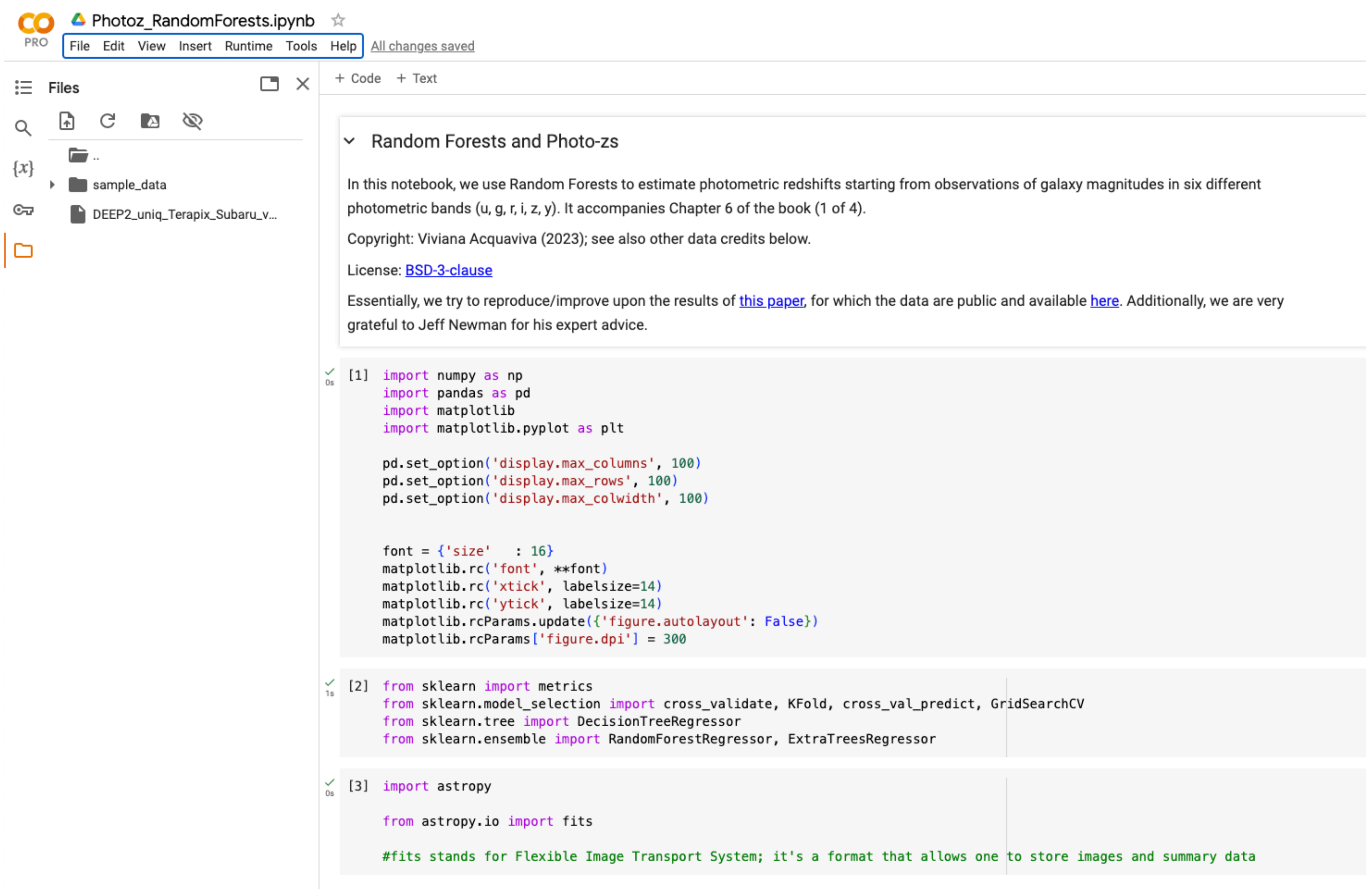
Feb. 13th 2025

AI
∩
Universe
$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi} \not{D} \psi + h.c.$
$+ \bar{\psi}_i y_{ij} \psi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

Moritz Münchmeyer (with slides from Gary Shiu)

Colab notebook

We first want to tie up some lose ends from the last lecture on decision trees.


Photoz_RandomForests.ipynb
File Edit View Insert Runtime Tools Help
All changes saved
Files
sample_data
DEEP2_uniq_Terapix_Subaru_v...
+ Code + Text

Random Forests and Photo-zs

In this notebook, we use Random Forests to estimate photometric redshifts starting from observations of galaxy magnitudes in six different photometric bands (u, g, r, i, z, y). It accompanies Chapter 6 of the book (1 of 4).

Copyright: Viviana Acquaviva (2023); see also other data credits below.

License: BSD-3-clause

Essentially, we try to reproduce/improve upon the results of this paper, for which the data are public and available here. Additionally, we are very grateful to Jeff Newman for his expert advice.

```
[1] import numpy as np
    import pandas as pd
    import matplotlib
    import matplotlib.pyplot as plt

    pd.set_option('display.max_columns', 100)
    pd.set_option('display.max_rows', 100)
    pd.set_option('display.max_colwidth', 100)


    font = {'size'   : 16}
    matplotlib.rc('font', **font)
    matplotlib.rc('xtick', labelsize=14)
    matplotlib.rc('ytick', labelsize=14)
    matplotlib.rcParams.update({'figure.autolayout': False})
    matplotlib.rcParams['figure.dpi'] = 300
```

```
[2] from sklearn import metrics
    from sklearn.model_selection import cross_validate, KFold, cross_val_predict, GridSearchCV
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor
```

```
[3] import astropy

    from astropy.io import fits

    #fits stands for Flexible Image Transport System; it's a format that allows one to store images and summary data
```

Decision tree ensembles

4. Details of the training process

In our decision tree training notebook we will encounter a few concepts which we have not yet discussed in the lecture.

These are not specific to decision trees, but let’s cover them here. In particular we need to know about

- R^2-score
- Cross-Validation
- Hyperparameter optimization

R^2-score

- The tree-regressor in scikit learn does not report the MSE loss, but rather the R^2 score. How does it measure the quality of a regression?

- The R^2 score also known as the **coefficient of determination**, is a statistical measure that indicates how well a regression model explains the variance in the dependent variable. It is defined as.

- We can attempt to measure the fraction of observed variance of the target variable ("outcome") that can be explained by the features ("predictors"):

$$\mathrm{SS}_{\mathrm{tot}} = \sum_{i=1}^{N} (y_i - \bar{y})^2; \qquad \mathrm{SS}_{\mathrm{res}} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2,$$

    - where y_i = true values, i is the index of the training example.
    - the bar denotes the mean, and the hat denotes the prediction.

- $\mathrm{SS}_{\mathrm{tot}}$ = Sum of Total Squares
- $\mathrm{SS}_{\mathrm{res}}$ = Sum of Squares due to Residuals

R^2-score

How “strong” is relationship between predictors & outcome?

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

where:
  SS_{res} = sum of squares due to residuals
  SS_{tot} = total sum of squares

1 indicates perfect correlation, and 0 indicates no relationship (a model that predicts the mean of true values will have $R^2 = 0$).

R^2 can be negative on the test set (e.g., the predictions can be arbitrarily bad, worse than the mean!)

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \in (-\infty, 1]$$

R^2-score

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

- Example: If R^2 has a value of 0.6, this means 60% of the variation in the dependent variable (y) is explained by your regression model. The remaining 40% is unexplained.
- In terms of optimization, **minimizing MSE and maximizing R^2** are equivalent because the model weights only influence the residual sum of squares SS_{res}.

- **Interpretation:**
  - The higher the value of R^2, the better the model fits the data.
  - R^2 is dimensionless.
  - When it is used in a machine learning setting (i.e. we look at R^2 for predictions, on test set), it is a useful tool to compare models.
  - If features and targets are the same, a model with higher R^2 score on the predicted values is better.
  - But we can’t distinguish between a poor R^2 that comes from bad modeling, and a poor R^2 that comes from noisy data.

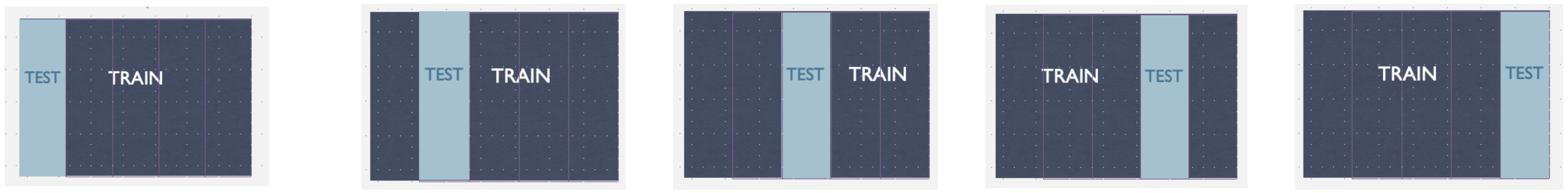
Cross-Validation

- We have not yet explained an important practice of supervised training, of particular importance when the data set is small: Cross-Validation

- The idea is to split the data set in multiple ways into training and test data. Then we train the model K times, and evaluate its average performance.

- This is a good idea because:
  - We want to use all the data for training (and not “lose” the test data)
  - We avoid the risk of under/overestimating performance because of a non-typical performance of a particular training/test split.
  - We get an estimate of how much scores fluctuate because of variance in the data.

- K-fold cross validation looks like this (for K=5)



- Disadvantage: We need to train the model K times, which takes more computation time.

- If we used cross-validation to pick hyper parameters we can then train a **new model** on the **entire dataset** using the same hyperparameters that performed best on average across the folds, using the full data set.

Hyperparameter tuning

- The most common procedure to optimize hyperparameters is a cross-validated **Grid Search of the hyperparameter space**.

- As a reminder, we do this to pick optimal hyperparameters, but the test scores we obtain are still optimistic (there is leakage of information between the optimization and the test scores).

- The correct procedure involves a 3-tiered structure: **train/validation/test set**

- Like with any parameter space search, the grid method can be inefficient, or too time-consuming. Alternatives are varying parameters one at a time (ignores correlations), Random Search (often good enough), Bayesian parameter search.

- There are elaborate hyper parameter optimization algorithms. Libraries specifically designed for this purpose include:
  - RayTune https://docs.ray.io/en/latest/tune/index.html
  - HyperOpt https://github.com/hyperopt/hyperopt
  - These libraries include several hyperparameter tuning algorithms. Rather than just performing a grid search, these algorithms optimize the hyper parameters by approximating the gradient of the optimization target (e.g. MSE) with respect to the hyper parameters.

Hyperparameter tuning with scikit-learn

- It is recommended to search the hyper-parameter space for the best cross validation score.

- **A search consists of:**
  - an estimator (regressor or classifier such as sklearn.tree.DecisionTreeRegressor);
  - a parameter space;
  - a method for searching or sampling candidates;
  - a cross-validation scheme; and
  - a score function (e.g. R2 score).

- Two generic approaches to parameter search are provided in scikit-learn: for given values,
  - **GridSearchCV** exhaustively considers all parameter combinations while
  - **RandomizedSearchCV** can sample a given number of candidates from a parameter space with a specified distribution.

- We use **GridSearchCV in the notebook**. It systematically works through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance based on a specified score.
  - https://scikit-learn.org/stable/modules/grid_search.html#grid-search

Classic Unsupervised ML Methods

Classic Unsupervised ML Methods

Overview

Unsupervised Methods

- After discussing two classic and important supervised methods (Feed Forward Neural Networks and Random Forrests), we now want to discuss some basic unsupervised methods.

- Unsupervised machine learning is a type of machine learning where the **model learns patterns and structures from unlabeled data—meaning there are no predefined outputs or labels.** It is mainly used for
  - **clustering**
  - **dimensionality reduction**
  - **anomaly detection**

- In addition, generative models are usually trained unsupervised, but we will talk about them in a later unit. E.g. (variational) Auto-encoders, Diffusion models, normalizing flows etc.

Classic Unsupervised ML Methods

Dimensionality reduction - PCA

References: 1803.08823, Deep Learning Book

Dimensionality Reduction and Latent Space

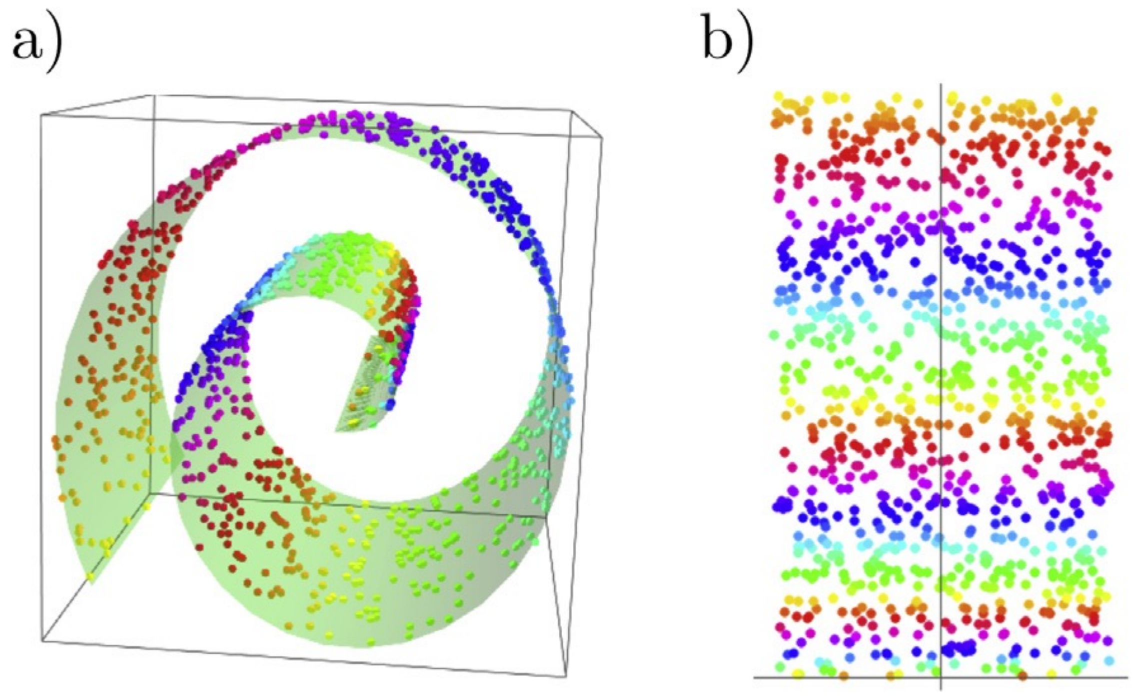
- **Discovering structure in unlabelled data**
- Need to dimensionally reduce data. Raw data is often impractical for data analysis or modeling.
- We call the **dimensionally reduced space latent space**.
- By dimensional reduction **we often loose information. This is not necessarily bad. Some of the information might be noise or irrelevant.** By loosing only irrelevant information, we can find good representations.

Challenges of High-dimensional Data

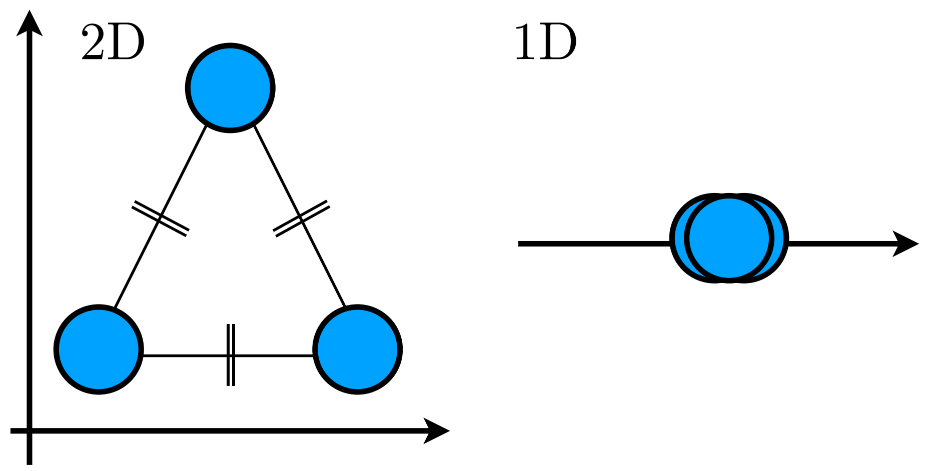
- **Real-world data is usually not random or uniformly distributed** (data lives in a lower-dim. space compared with original space).
- “**Blessing of non-uniformity**”: Data will typically be locally smooth (local variation will not incur a large change in the target variable).
- Objective: preserve relative pairwise distances between data points when going to latent space.

Challenges of High-dimensional Data

- **Intrinsic dimensionality and the crowding problem:**



In this example a two-dimensional space is sufficient to capture almost the entirety of the information in the data. Intrinsic dim = min. # parameters to parametrize the data.



Attempts to represent data in a space with dim < intrinsic dimensionality lead to a “crowding” problem. all mapped data points collapse to the center of the map.

Principal Component Analysis

- A ubiquitous method for dimensional reduction, data visualization and analysis is **Principal Component Analysis (PCA)**. It is based on linear algebra.
- The goal of PCA is to perform an orthogonal transformation of the data in order to find high-variance directions.
- PCA is inspired by the observation that in many cases, the relevant information in a signal is contained in the directions with largest variance. Directions with small variance are ascribed to “noise” and can potentially be removed or ignored.

Principal Component Analysis (PCA)

- **Perform an orthogonal transformation of the data to find the high variance directions ⇔ minimizing the error in projection.**

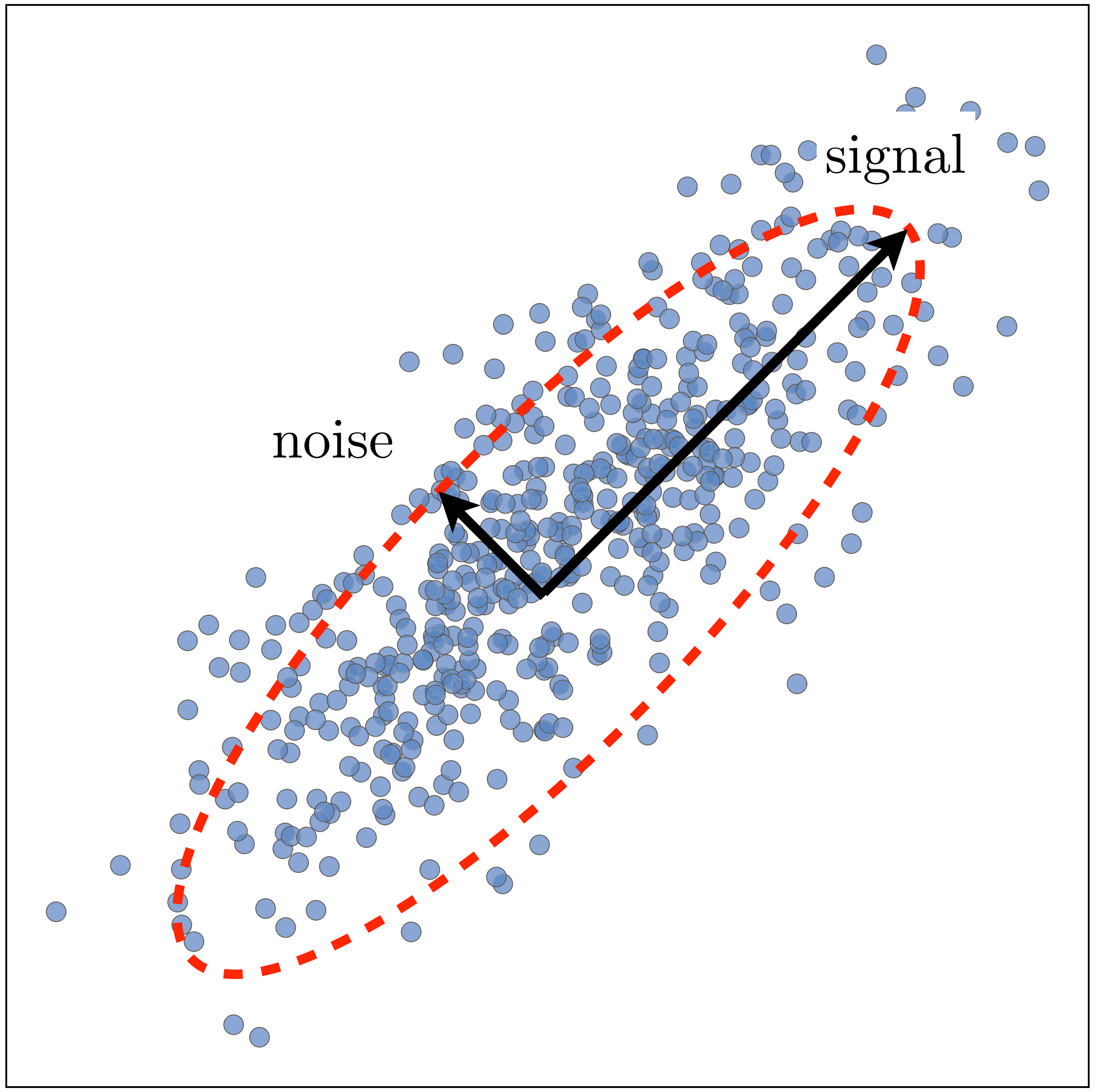


FIG. 50 PCA seeks to find the set of orthogonal directions with largest variance. This can be seen as "fitting" an ellipse to the data with the major axis corresponding to the first principal component (direction of largest variance). PCA assumes that directions with large variance correspond to the true signal in the data while directions with low variance correspond to noise.

PCA — Maximizing Variance

- The covariance matrix of data (design) matrix $\mathbf{X}$ is defined as:

$$\boldsymbol{\Sigma}(\boldsymbol{X}) = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X}$$

- $\Sigma(\mathbf{X})_{jj}$ corresponds to the variance of the j-th feature while $\Sigma(\mathbf{X})_{ij}$ measures the covariance (correlation) between feature i & feature j.

- We want to find a new basis that emphasizes highly variable directions while reducing redundancy between basis vectors. Perform singular value decomposition (SVD):

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T,$$

Recall SVD

Singular Value Decomposition (SVD) is a factorization of a matrix that expresses it as the product of three matrices:

$$A = U\Sigma V^T$$

where:

- A is an $m \times n$ matrix.
- U is an $m \times m$ orthogonal (or unitary) matrix containing the **left singular vectors**.
- Σ is an $m \times n$ diagonal matrix containing the **singular values** (which are non-negative and ordered from largest to smallest).
- V^T is an $n \times n$ orthogonal (or unitary) matrix containing the **right singular vectors**.

Key Properties

1. **Singular values** (σ_i in Σ) represent the importance (variance) of each singular vector.
2. **Left singular vectors** (U) span the column space of A.
3. **Right singular vectors** (V) span the row space of A.
4. **Dimensionality reduction**: The first few singular values capture most of the important structure in A, allowing approximations:

$$A_k = U_k \Sigma_k V_k^T$$

   where k is the rank of the reduced approximation.

If A is **symmetric** ($A^T = A$), then SVD reduces to **eigendecomposition**:

$$A = U\Lambda U^T$$

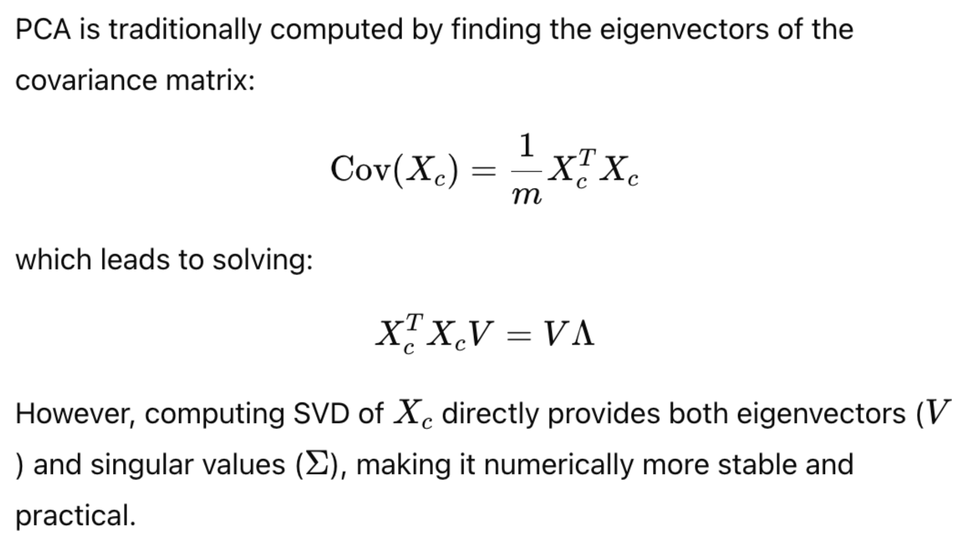
where:

- U contains **orthonormal eigenvectors**.
- Λ is a diagonal matrix of **eigenvalues**.

PCA — Maximizing Variance

- Back to the PCA: Using singular value decomposition (SVD):

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T,$$



PCA is traditionally computed by finding the eigenvectors of the covariance matrix:

$$\mathrm{Cov}(X_c) = \frac{1}{m} X_c^T X_c$$

which leads to solving:

$$X_c^T X_c V = V \Lambda$$

However, computing SVD of X_c directly provides both eigenvectors (V) and singular values (Σ), making it numerically more stable and practical.

We get

$$\begin{aligned} \boldsymbol{\Sigma}(\boldsymbol{X}) &= \frac{1}{N-1} \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \\ &= \boldsymbol{V}\left(\frac{\boldsymbol{S}^2}{N-1}\right)\boldsymbol{V}^T \\ &\equiv \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T. \end{aligned}$$

- The eigenvalues λ_i of Λ are given by $\lambda_i = s_i^2/(N-1)$.

PCA — Maximizing Variance

- To **reduce the dimensionality of data** from n to l, construct the $n \times l$ projection matrix $\mathbf{V}_l$ by selecting the singular components with the l largest singular values. The projection is then

$$\mathbf{Y} = \mathbf{X}\mathbf{V}_l$$

- The singular vector with the largest singular value (largest variance) is the first principal component; the singular vector with the second largest variance is the second principal component, etc.

- Common in data visualization is to project on the first few principal components (as long as a large part of the variance is explained in those components, e.g., Ising Model).

- Low explained variance may imply that the intrinsic dimensionality of the data is high, or it cannot be captured by a linear representation.

Example of PCA in physics: Ising Model

From 1803.08823

The Ising dataset we use throughout the review was generated using the standard Metropolis algorithm to generate a Markov Chain. The full dataset consist of 16×10000 samples of 40×40 spin configurations (i.e. the design matrix has 160000 samples and 1600 features) drawn at temperatures $0.25, 0.5, \cdots 4.0$. The samples are drawn for the Boltzmann distribution of the two-dimensional ferromagnetic Ising model on a 40×40 square lattice with periodic boundary conditions.

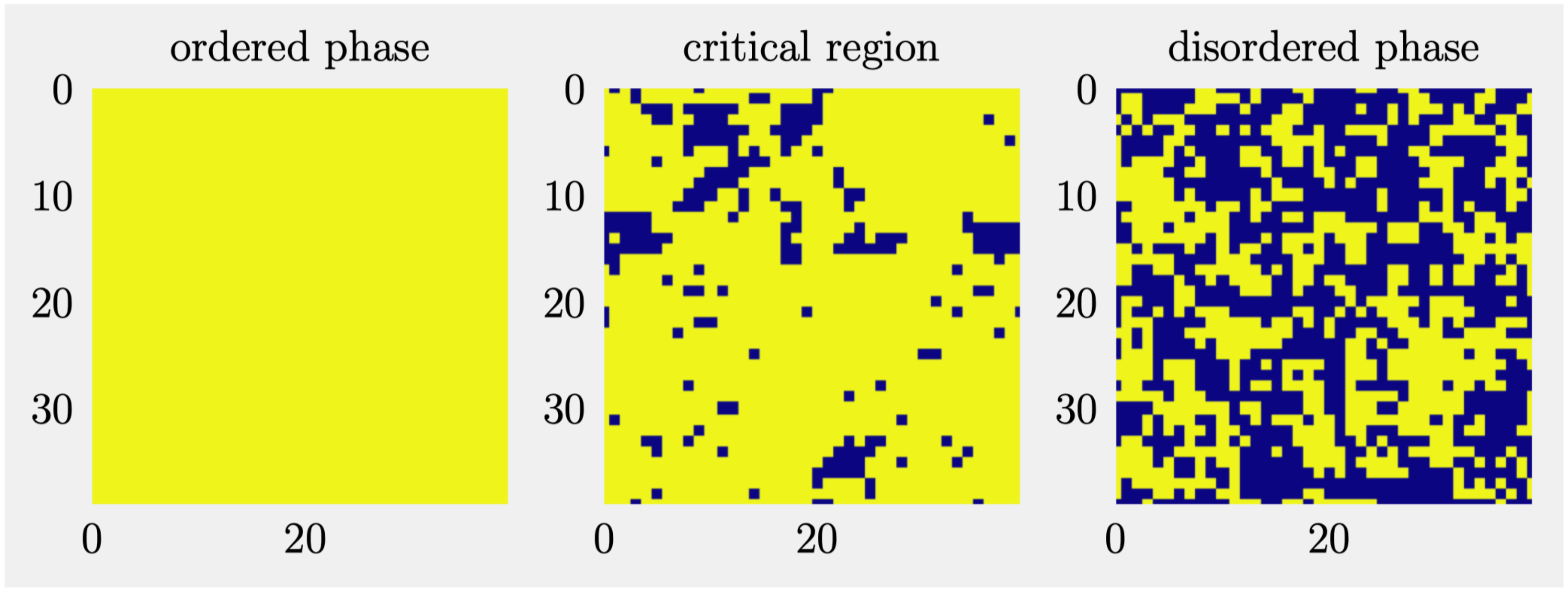


FIG. 20 Examples of typical states of the $2D$ Ising model for three different temperatures in the ordered phase ($T/J = 0.75$, left), the critical region ($T/J = 2.25$, middle) and the disordered phase ($T/J = 4.0$, right). The linear system dimension is $L = 40$ sites.

Example of PCA in physics: Ising model

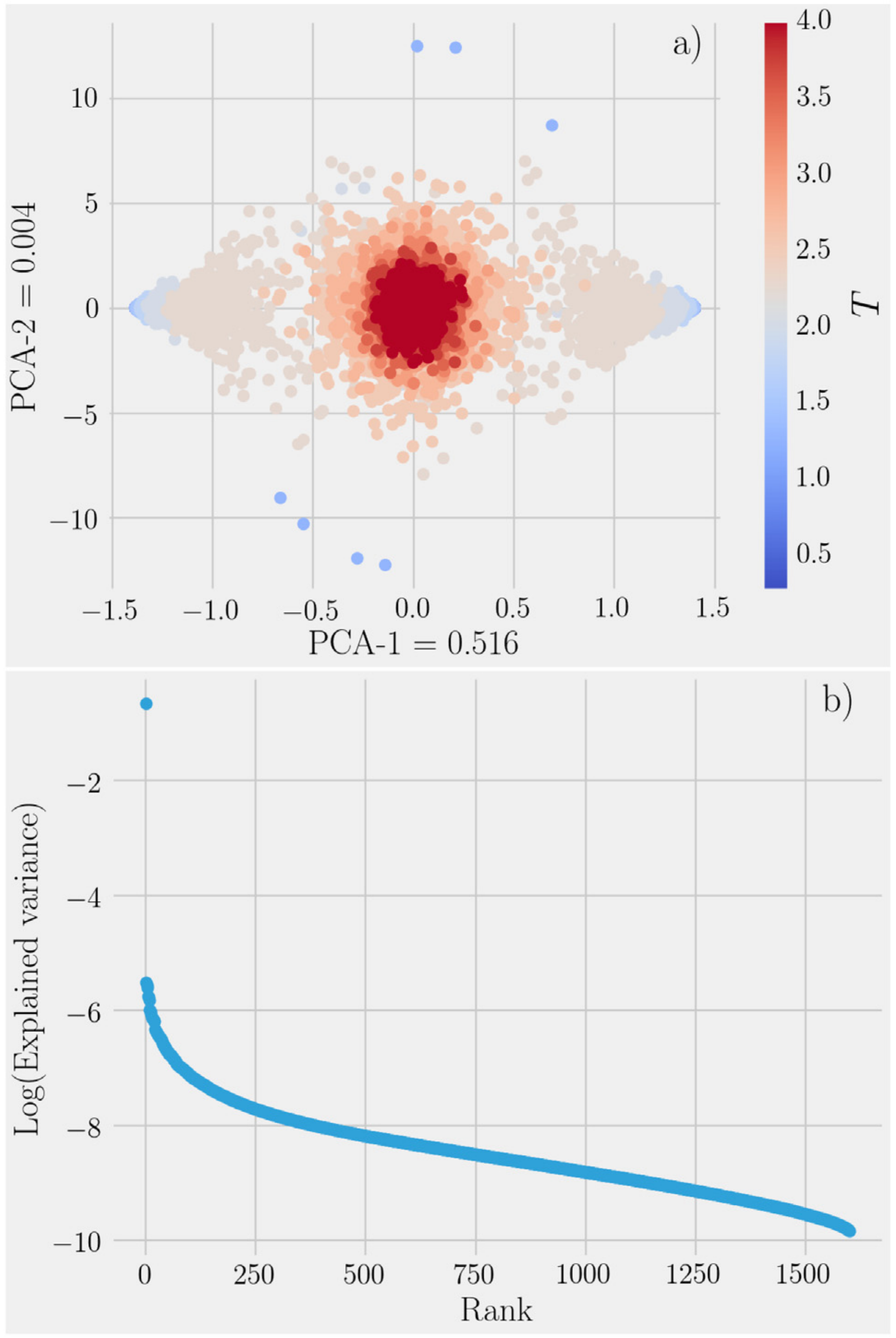
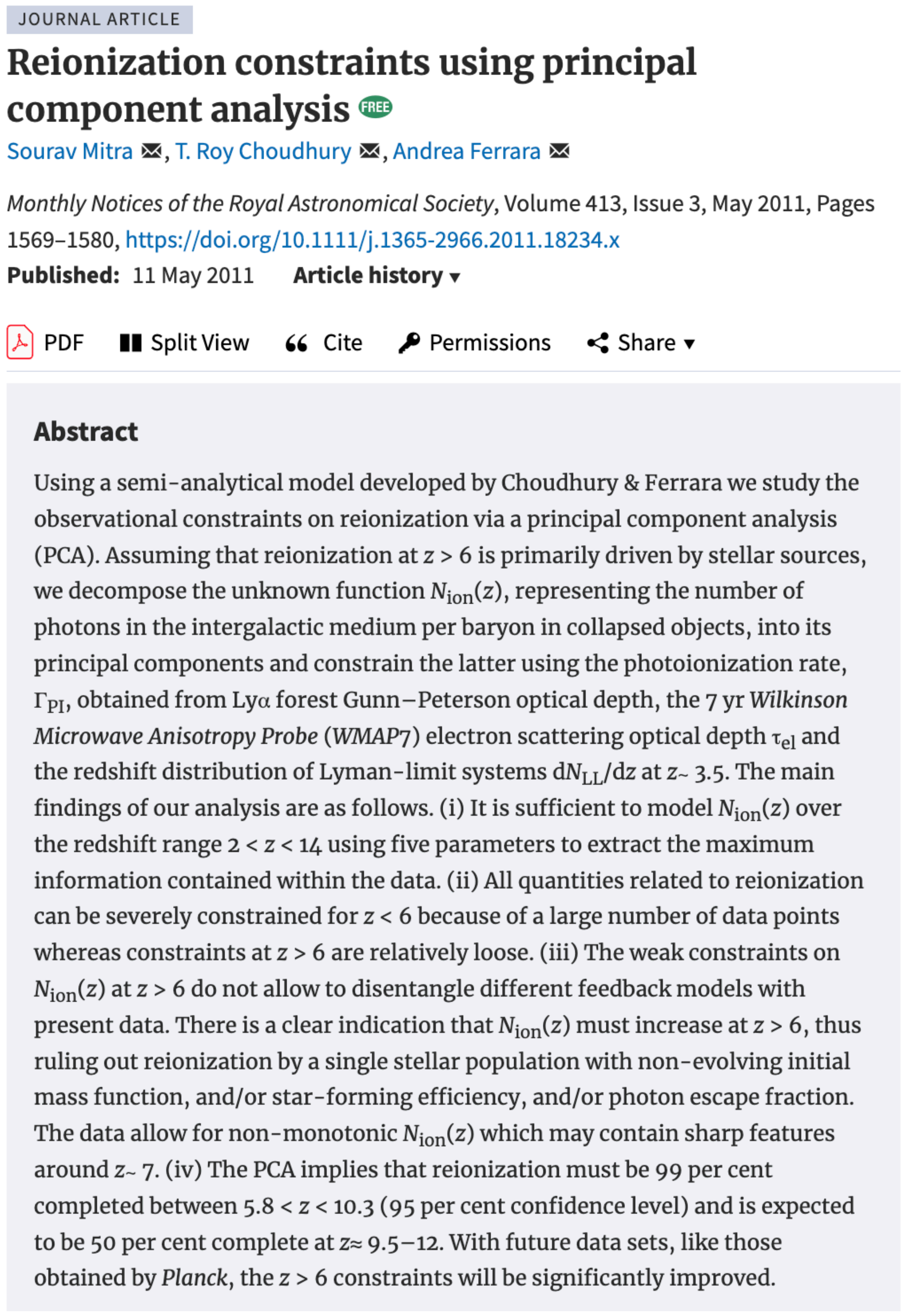


FIG. 51 (a) The first 2 principal component of the Ising dataset with temperature indicated by the coloring. PCA was performed on a joined dataset of 1000 samples taken at each temperatures $T = 0.25, 0.5, \cdots, 4.0$. Almost all the variance is explained in the first component which corresponds to the magnetization order parameter (linear combination of the features with weights all roughly equal). The paramagnetic phase corresponds to the middle cluster and the left and right clusters correspond to the symmetry-related ferromagnetic phases (b) Log of the spectrum of the covariance matrix versus rank ordering. Only one dimension has high-variance.

Example of PCA in cosmology



JOURNAL ARTICLE

Reionization constraints using principal component analysis FREE

Sourav Mitra, T. Roy Choudhury, Andrea Ferrara

Monthly Notices of the Royal Astronomical Society, Volume 413, Issue 3, May 2011, Pages 1569–1580, https://doi.org/10.1111/j.1365-2966.2011.18234.x

Published: 11 May 2011 **Article history** ▾

PDF Split View Cite Permissions Share ▾

Abstract

Using a semi-analytical model developed by Choudhury & Ferrara we study the observational constraints on reionization via a principal component analysis (PCA). Assuming that reionization at $z > 6$ is primarily driven by stellar sources, we decompose the unknown function $N_{\rm ion}(z)$, representing the number of photons in the intergalactic medium per baryon in collapsed objects, into its principal components and constrain the latter using the photoionization rate, $\Gamma_{\rm PI}$, obtained from Lyα forest Gunn–Peterson optical depth, the 7 yr *Wilkinson Microwave Anisotropy Probe* (*WMAP*7) electron scattering optical depth $\tau_{\rm el}$ and the redshift distribution of Lyman-limit systems $dN_{\rm LL}/dz$ at $z\sim 3.5$. The main findings of our analysis are as follows. (i) It is sufficient to model $N_{\rm ion}(z)$ over the redshift range $2 < z < 14$ using five parameters to extract the maximum information contained within the data. (ii) All quantities related to reionization can be severely constrained for $z < 6$ because of a large number of data points whereas constraints at $z > 6$ are relatively loose. (iii) The weak constraints on $N_{\rm ion}(z)$ at $z > 6$ do not allow to disentangle different feedback models with present data. There is a clear indication that $N_{\rm ion}(z)$ must increase at $z > 6$, thus ruling out reionization by a single stellar population with non-evolving initial mass function, and/or star-forming efficiency, and/or photon escape fraction. The data allow for non-monotonic $N_{\rm ion}(z)$ which may contain sharp features around $z\sim 7$. (iv) The PCA implies that reionization must be 99 per cent completed between $5.8 < z < 10.3$ (95 per cent confidence level) and is expected to be 50 per cent complete at $z\approx 9.5$–12. With future data sets, like those obtained by *Planck*, the $z > 6$ constraints will be significantly improved.

Non-linear generalizations of PCA

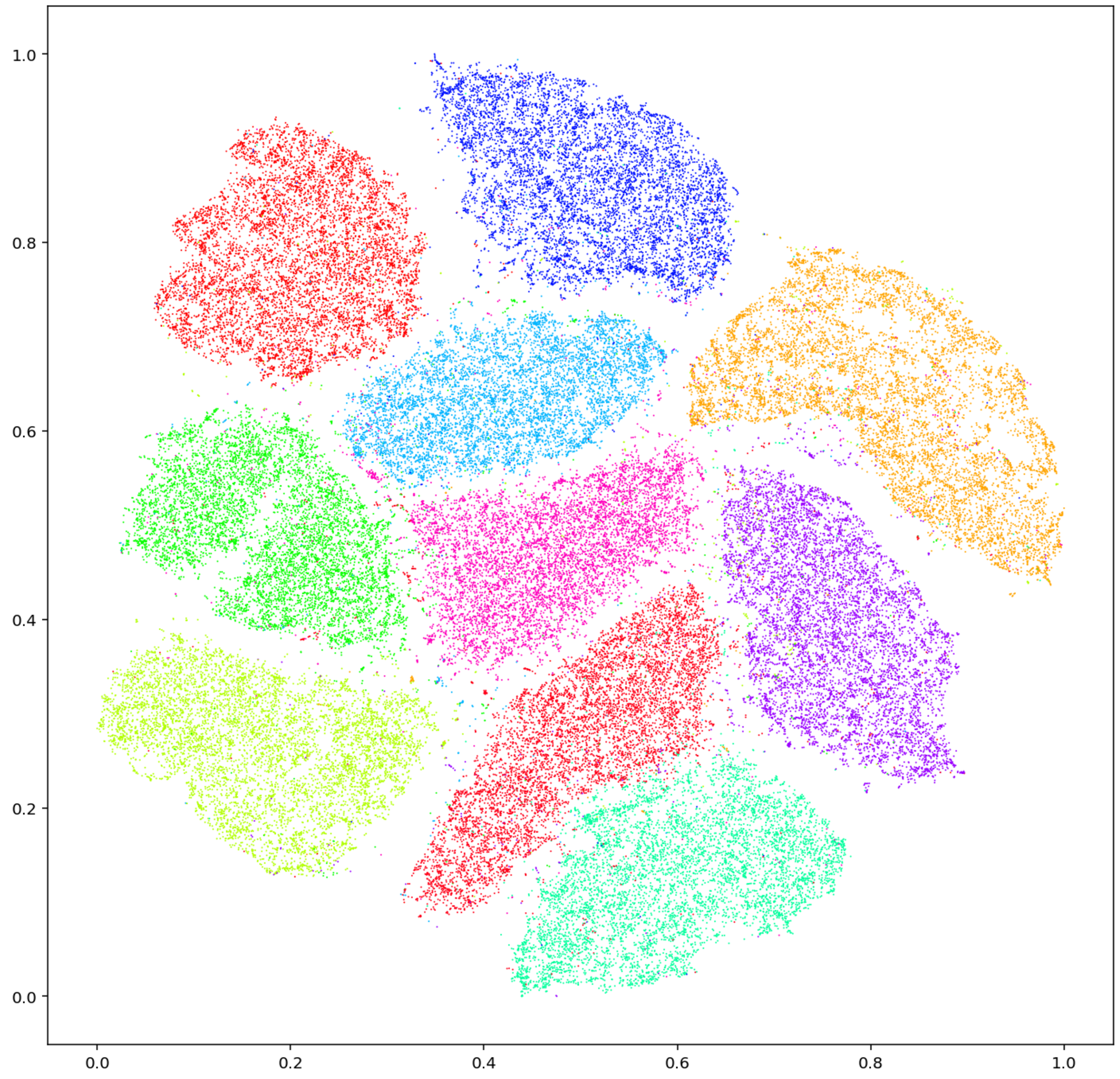
- there are several nonlinear generalizations of PCA designed to handle data that lies on a nonlinear manifold rather than a linear subspace.
- Here are some of the most important ones:
  - Kernel PCA (kPCA)
  - **Autoencoders** (Neural Network-based PCA). See later!
  - **t-SNE** (t-Distributed Stochastic Neighbor Embedding)
  - UMAP (Uniform Manifold Approximation and Projection)

Classic Unsupervised ML Methods

Dimensionality reduction - t-SNE

t-SNE

- t-SNE is a nonlinear dimensionality reduction technique used primarily for **visualizing high-dimensional data** in 2D or 3D while **preserving local structure**.
- It works by converting high-dimensional distances into probabilities and minimizing the divergence between probability distributions in the high- and low-dimensional spaces.
- Similar points in high-dimensional space stay close together in lower dimensions.



Summary of how t-SNE works

Overview of the approach, before looking into the math

- **Measure Similarities in High-Dimensional Space**
  - It calculates how similar each data point is to others based on distances.
  - Instead of raw distances, it converts distances into probabilities.
- **Measure Similarities in Low-Dimensional Space**
  - It maps the data into a lower-dimensional space (e.g., 2D) and tries to preserve the same probability-based relationships.
  - Instead of using Gaussian distributions, it uses a Student’s t-distribution (which prevents crowding).
- **Match the Two Representations**
  - t-SNE minimizes the difference between the high-dimensional and low-dimensional probability distributions.
  - It does this using an optimization method called gradient descent, which fine-tunes the positions of points in lower dimensions.

By converting distances into **probabilities**, t-SNE ensures that the notion of "closeness" is meaningful **regardless of the absolute scale of distances**.

t-SNE math

Step 1: Compute Pairwise Similarities in High-Dimensional Space

For each data point x_i, we define the probability that x_j is its neighbor using a Gaussian kernel:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

where:

- σ_i is a local **perplexity-controlled** bandwidth (adaptive for different densities),
- $p_{j|i}$ represents the probability that x_j is chosen as a neighbor of x_i.

To ensure symmetry, the final pairwise similarity measure is:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

where N is the number of data points.

t-SNE math

Step 2: Compute Pairwise Similarities in Low-Dimensional Space

We now embed points into a lower-dimensional space (e.g., 2D) and define a similar probability distribution.

Instead of Gaussians, t-SNE uses a **Student's t-distribution** (with 1 degree of freedom, i.e., a Cauchy distribution) to compute similarities:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

where:

- y_i and y_j are the low-dimensional representations of x_i and x_j,
- The t-distribution has **fatter tails**, preventing crowding in the lower-dimensional space.

t-SNE math

Step 3: Minimize the KL Divergence Between Distributions

The goal is to make q_{ij} as close as possible to p_{ij}. This is done by minimizing the **Kullback-Leibler (KL) divergence**:

$$C = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- $KL(p||q)$ measures how different the distributions p_{ij} and q_{ij} are.
- **Gradient descent** is used to iteratively update the low-dimensional coordinates y_i.

The **goal of t-SNE** is to find a **set of points** $y_1, y_2, \ldots, y_N$ **in 2D or 3D** that best preserves the local structure of the high-dimensional data.

Performance

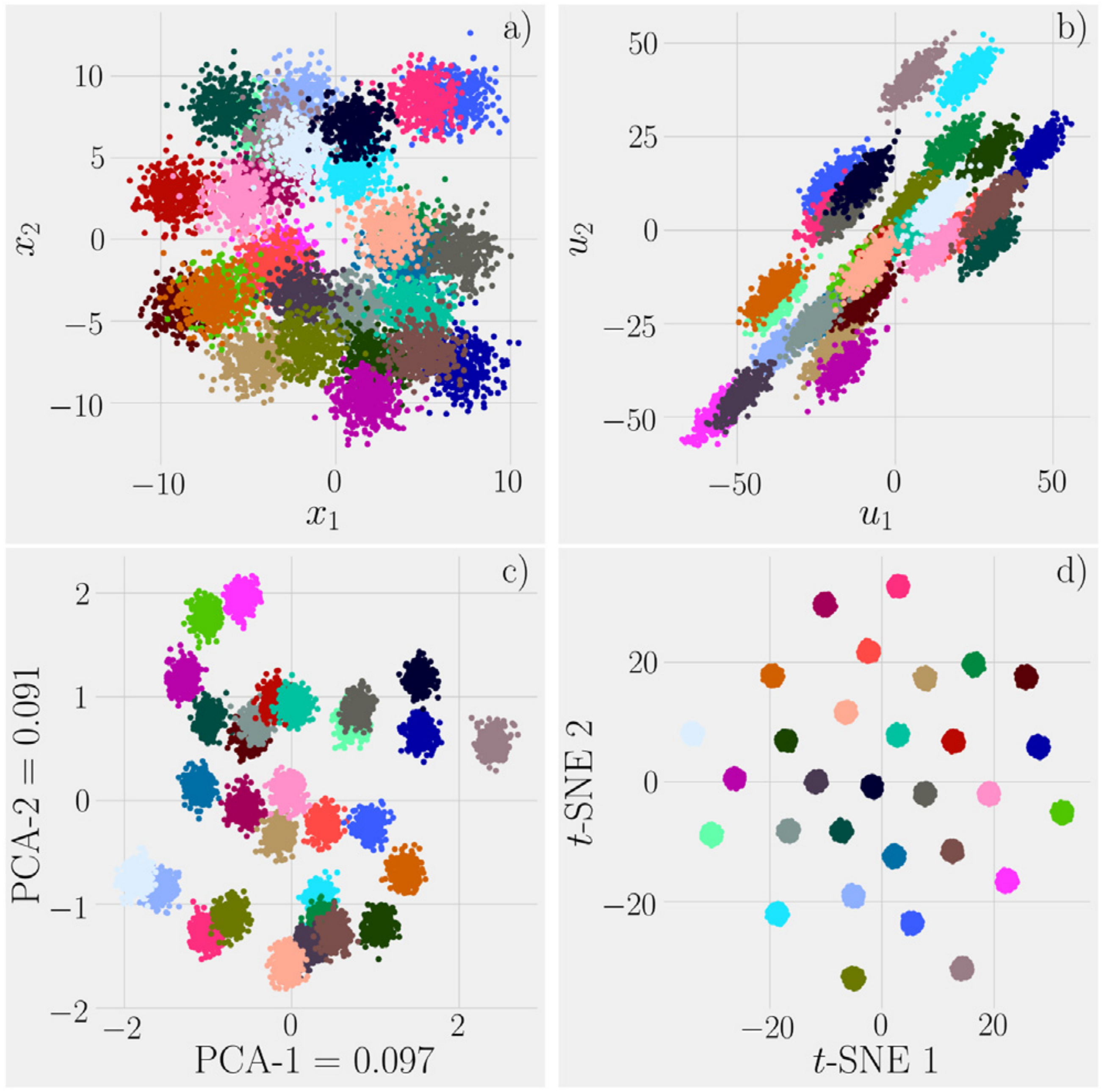


Fig. 53. Different visualizations of a Gaussian mixture formed of $K = 30$ mixtures in a $D = 40$ dimensional space. The Gaussians have the same covariance but have means drawn uniformly at random in the space $[-10, 10]^{40}$. (a) Plot of the first two coordinates. The labels of the different Gaussian are indicated by the different colors. Note that in a realistic setting, label information is of course not available, thus making it very hard to distinguish the different clusters. (b) Random projection of the data onto a 2 dimensional space. (c) Projection onto the first 2 principal components. Only a small fraction of the variance is explained by those components (the ratio is indicated along the axis). (d) t-SNE embedding (perplexity = 60, # iteration = 1000) in a 2 dimensional latent space. t-SNE captures correctly the local structure of the data.

Performance

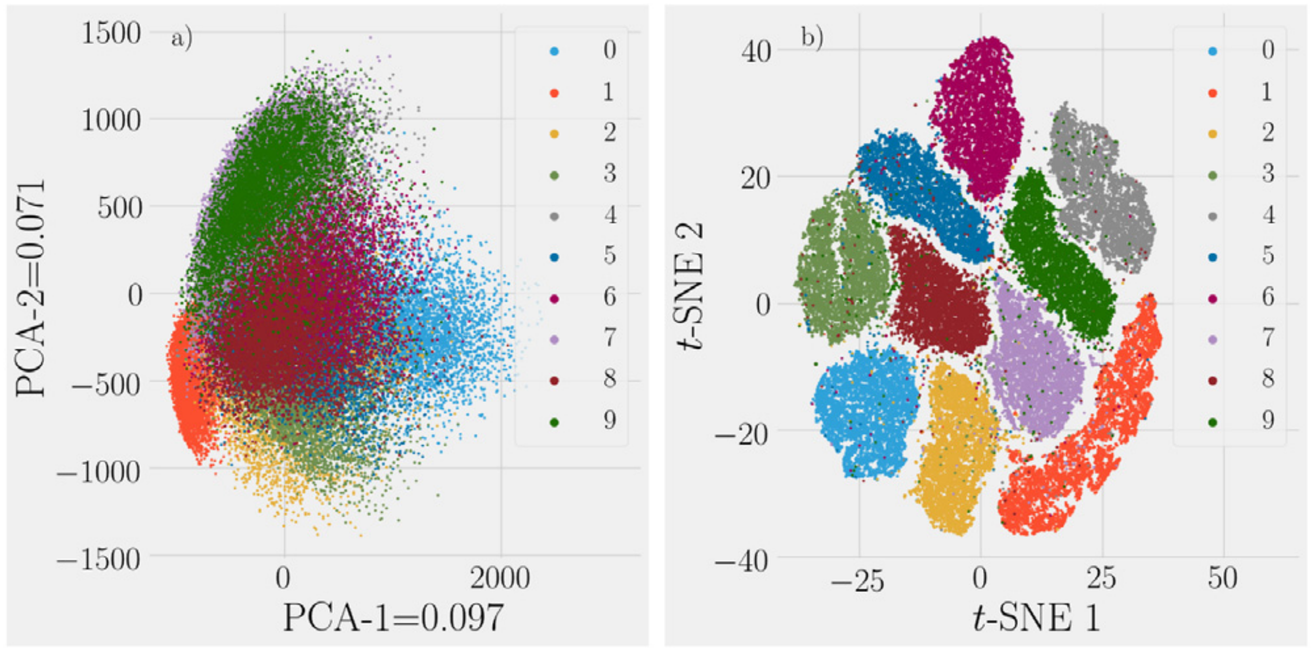
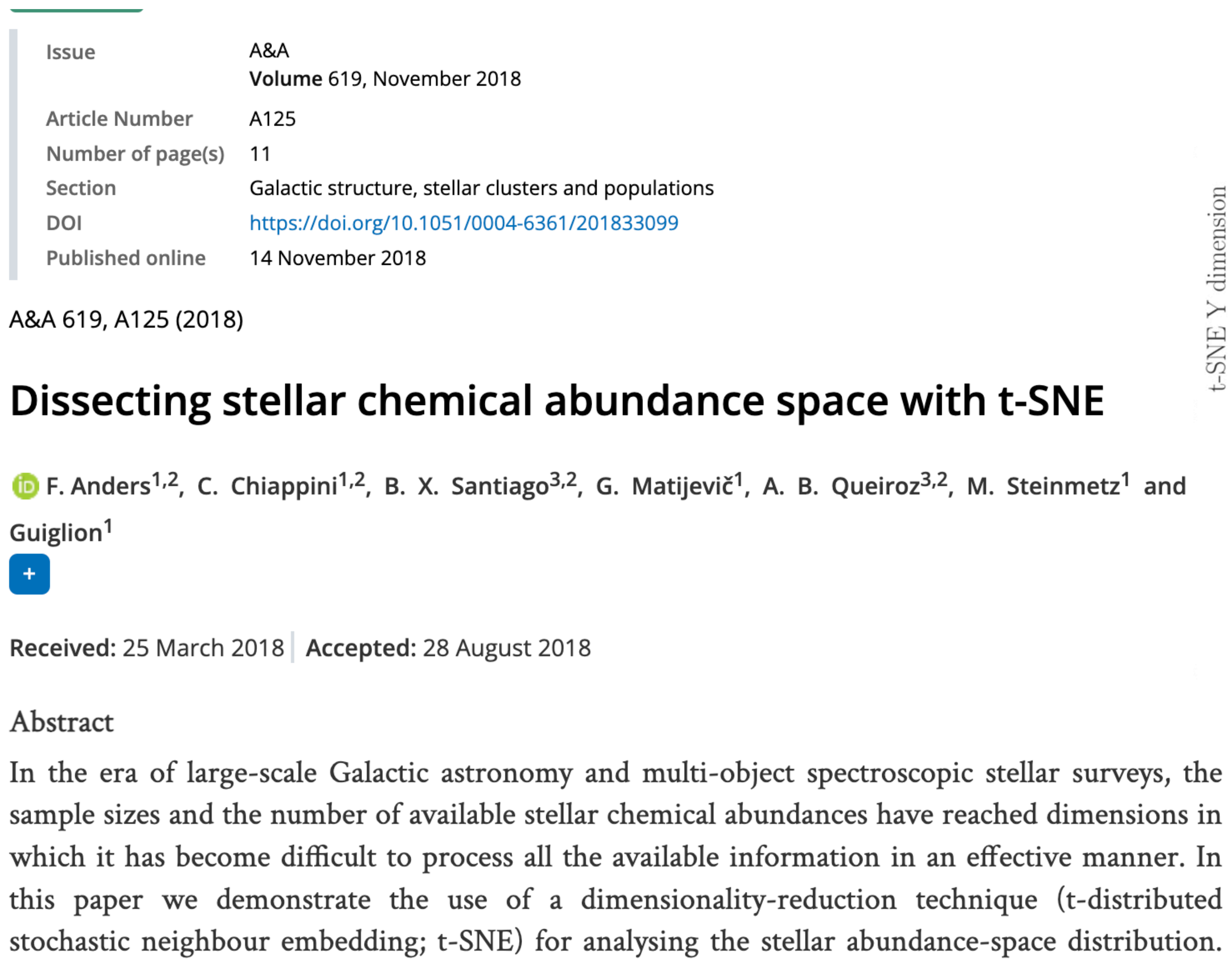


Fig. 54. Visualization of the MNIST handwritten digits training dataset (here $N = 60\ 000$). (a) First two principal components. (b) t-SNE applied with a perplexity of 30, a Barnes–Hut angle of 0.5 and 1000 gradient descent iterations. In order to reduce the noise and speed-up computation, PCA was first applied to the dataset to project it down to 40 dimensions. We used an open-source implementation to produce the results (Linderman et al., 2017), see https://github.com/KlugerLab/FIt-SNE.

Application in physics: Stars



Issue	A&A **Volume** 619, November 2018
Article Number	A125
Number of page(s)	11
Section	Galactic structure, stellar clusters and populations
DOI	https://doi.org/10.1051/0004-6361/201833099
Published online	14 November 2018

A&A 619, A125 (2018)

Dissecting stellar chemical abundance space with t-SNE

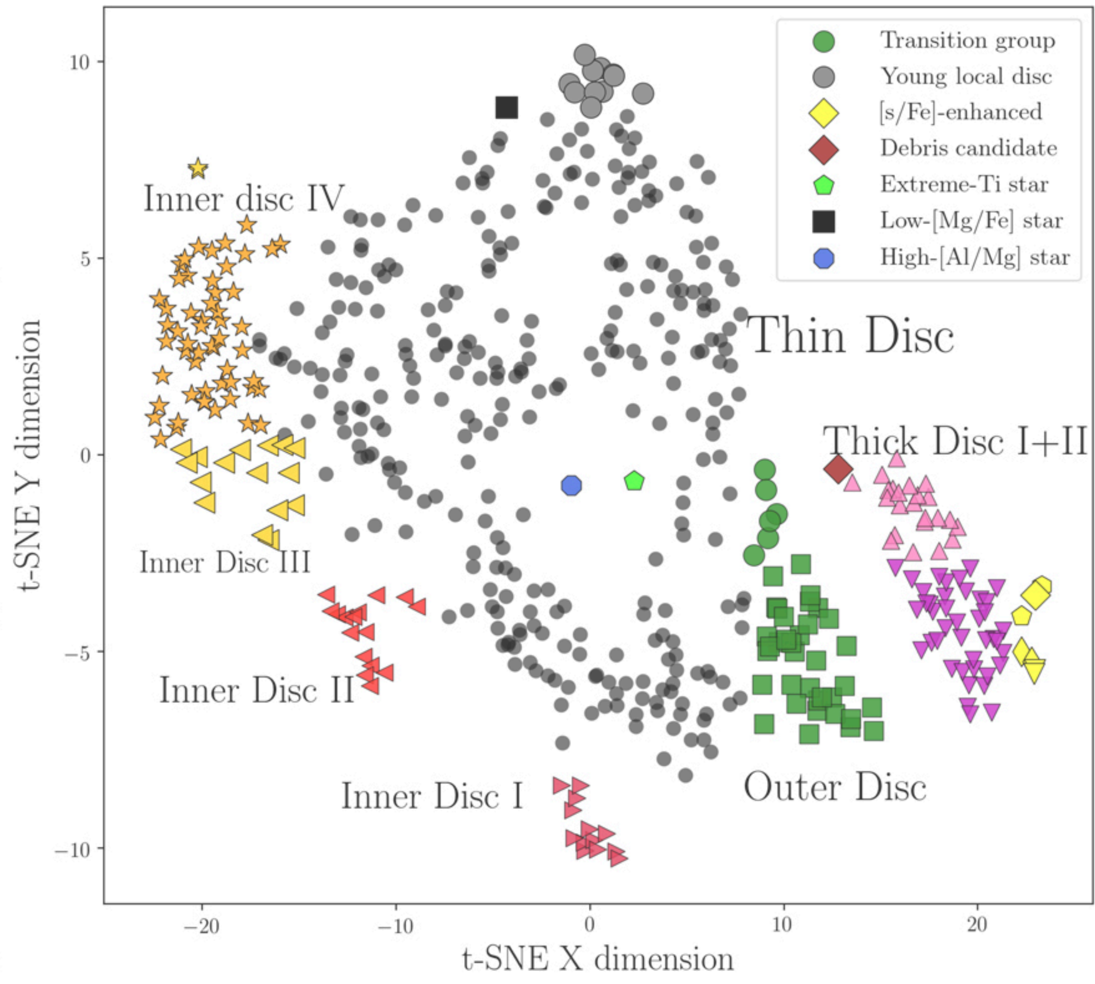
F. Anders[1,2], C. Chiappini[1,2], B. X. Santiago[3,2], G. Matijevič[1], A. B. Queiroz[3,2], M. Steinmetz[1] and Guiglion[1]

+

Received: 25 March 2018 | **Accepted:** 28 August 2018

Abstract

In the era of large-scale Galactic astronomy and multi-object spectroscopic stellar surveys, the sample sizes and the number of available stellar chemical abundances have reached dimensions in which it has become difficult to process all the available information in an effective manner. In this paper we demonstrate the use of a dimensionality-reduction technique (t-distributed stochastic neighbour embedding; t-SNE) for analysing the stellar abundance-space distribution.



Classic Unsupervised ML Methods

Clustering Methods

Clustering

- Clustering is a way to look for hidden structure in high dimensions (coarse features or high-level structures in unlabelled data).
- Points to take into account when choosing a particular method:
  - Distribution of clusters (overlapping/noisy clusters vs. well-separated clusters)
  - Geometry of the data (flat vs. non-flat)
  - Cluster size distribution (multiple vs. uniform sizes)
  - Dimensionality of the data (low-dimensional vs. high-dimensional)
  - Computational efficiency of desired method

Clustering and Latent Variables

- Central to unsupervised learning is **the idea of a latent or hidden variable (not directly observable; yet influence visible structure)**.

- Example: The **cluster identity of each datapoint is a latent variable**. We cannot observe the label directly, but points in the same cluster are "close".

- In this abstract language, **clustering is an algorithm to learn the most probably value of a latent variable** associated with each datapoint.

- Need to make assumption about the structure of data (common to unsupervised learning), e.g., underlying probability distribution from which the data was generated.

K-means Clustering

- Divide data set into K different clusters of data points which are *near* each-other.
- Consider a set of N unlabeled data points $\{\mathbf{x}_n\}_{n=1}^{N}$ where $\mathbf{x}_n \in \mathbb{R}^p$.
- K cluster centers called the cluster means: $\{\mu_k\}_{k=1}^{K}$ with $\mu_k \in \mathbb{R}^p$.
- Minimize the cost: $\mathcal{C}(\{x, \boldsymbol{\mu}\}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)^2,$
- One-hot encoding: $r_{nk} = 1$ if $\mathbf{x}_n \in$ cluster k and 0 otherwise;

$$\sum_k r_{nk} = 1 \ \forall \ n \ \text{and} \ \sum_n r_{nk} \equiv N_k,$$

- Find the best cluster means (center of mass) such that variance (moment of inertia) is minimized.

K-means Algorithm

- **Expectation:** Given $\{r_{nk}\}$, minimize C with respect to μ_k:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_n r_{nk} \mathbf{x}_n.$$

- **Maximization:** Given $\{\mu_k\}$, find $\{r_{nk}\}$ which minimizes C:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_{k'} (\mathbf{x}_n - \boldsymbol{\mu}_{k'})^2 \\ 0 & \text{otherwise} \end{cases}$$

- Alternative between the above two steps until some convergence criterion is met (e.g., change in C is smaller than a threshold).

- **Guaranteed to converge to local minimum**. Complexity $\mathcal{O}(kN)$.

- Hard-assignment limit of the Gaussian mixture model (introduce later), where all cluster variances are assumed to be the same.

- If the true clusters have very different variances (spreads), K-means can lead to spurious results since the **underlying assumption is that the latent model has uniform variances.**

K-means Algorithm

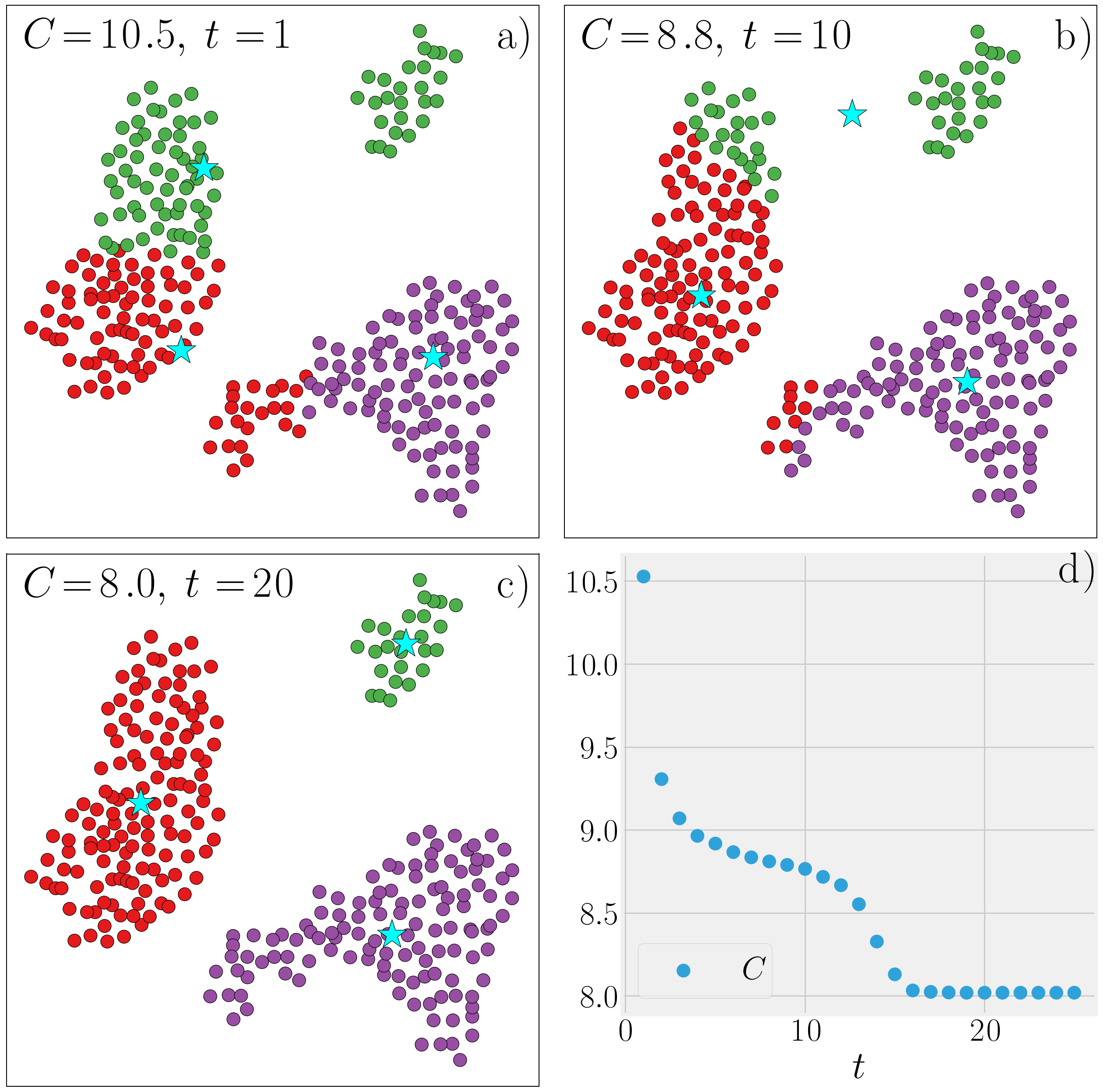
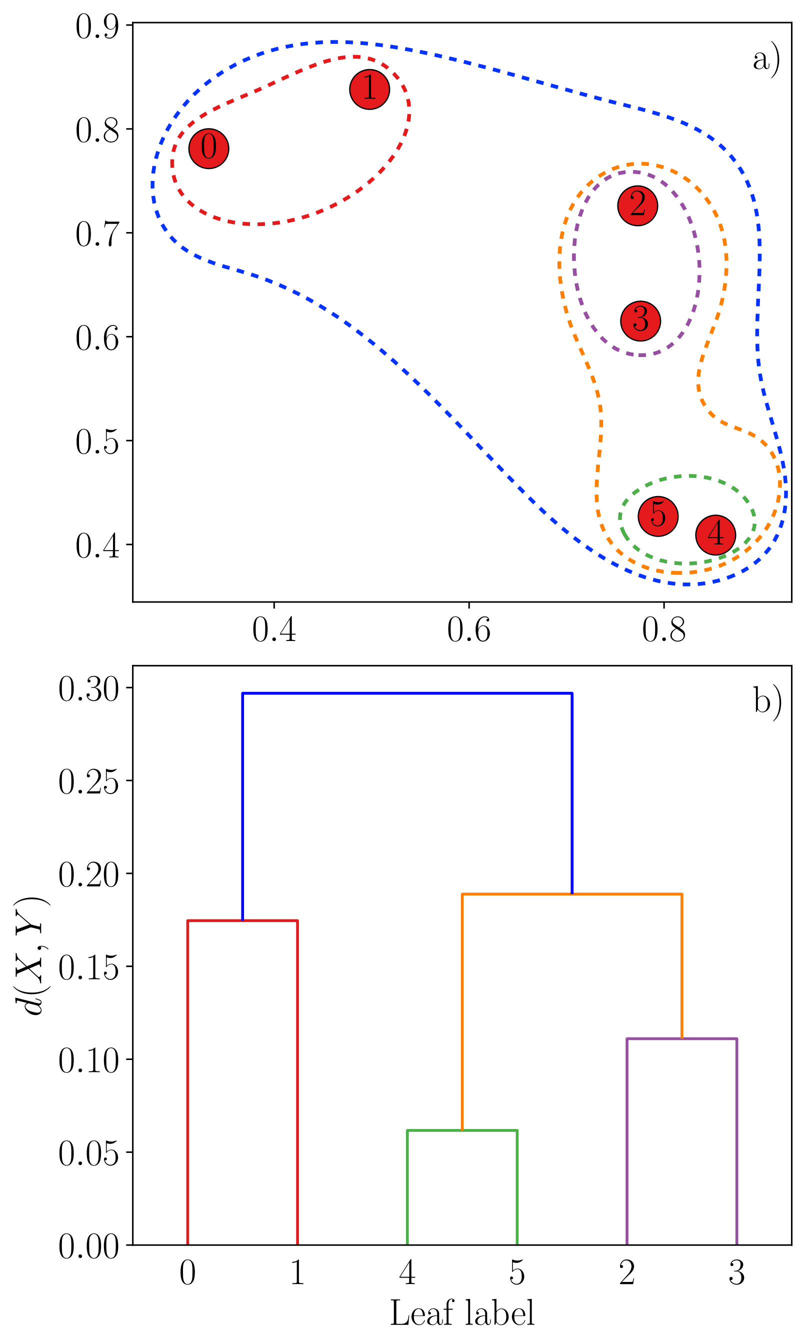


Fig. 55. K-means with $K = 3$ applied to an artificial two-dimensional dataset. The cluster means at each iteration are indicated by cyan star markers. t indicates the iteration number and $\mathcal{C}$ the value of the objective function. (a) The algorithm is initialized by randomly partitioning the space into 3 sectors to generate an initial assignment. (b)–(c) For well separated clusters, the algorithm converges rapidly to the true clusters. (d) The objective function as a function of the iteration. $\mathcal{C}$ converges after $t = 18$ iterations for this choice of random seed (for center initialization).

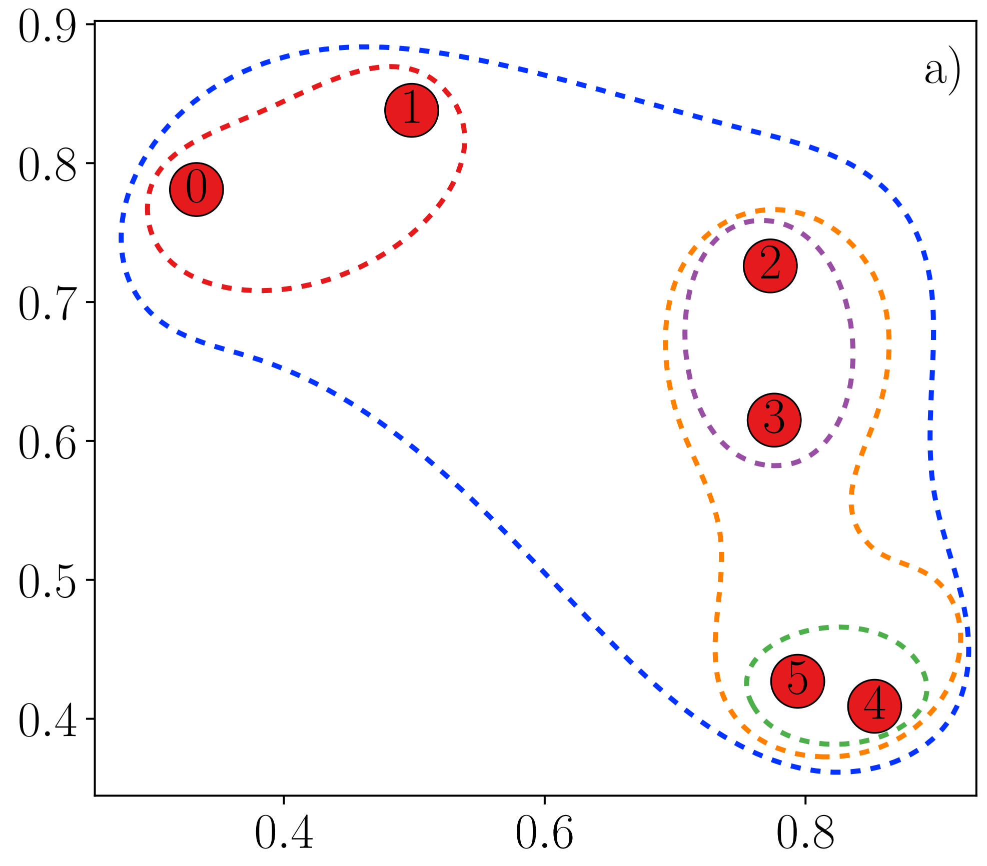
Agglomerative Method

- Start from small initial clusters, then progressively merged to form larger clusters.
- Hierarchy of cluster can be visualized in the form of a **dendrogram**.
- Define a distance measure $d(X, Y)$ between clusters X and Y.
- Two distances that are closest with respect to $d(X, Y)$ are merged until a single cluster is left.



Agglomerative Clustering Algorithm

- Initialize each point to its own cluster.
- Given a set of K clusters $X_1, X_2, \ldots, X_K$, merge clusters until one cluster is left ($K = 1$):
  - Find the closest pair of clusters $(X_i, X_j) : (i, j) = \text{argmin}_{(i',j')} d(X_{i'}, X_{j'})$
  - Merge the pair. Update $K \leftarrow K - 1$.
- Different linkage methods (distances) result in different algorithms.



Density-based (DB) Clustering

- Clusters are defined by regions with high density of data points.
- Noise or outliers are expected to form regions of low density.
- Unlike a distance-based approach, DB clustering considers clusters of multiple shapes and sizes while identifying outliers.
- Assumption: relative local density estimation is possible (becomes difficult for very high-dimensional data due to large sampling noise).
- Widely used algorithm: DBSCAN

DBScan Algorithm

- **Density-based spatial clustering of applications with noise** (Ester et al, 1996).

- Crude estimate of local density is the ϵ-neighborhood of point $\mathbf{x}_n$:

$$N_\varepsilon(\mathbf{x}_n) = \{\mathbf{x} \in X | d(\mathbf{x}, \mathbf{x}_n) < \varepsilon\}$$

- $\mathbf{x}_n$ is a **core-point** if at least **minPts** are in its ϵ-neighborhood. A point $\mathbf{x}_i$ is **density-reachable** if it's in a core-point's ϵ-neighborhood.

- The DBSCAN algorithm uses the following steps:

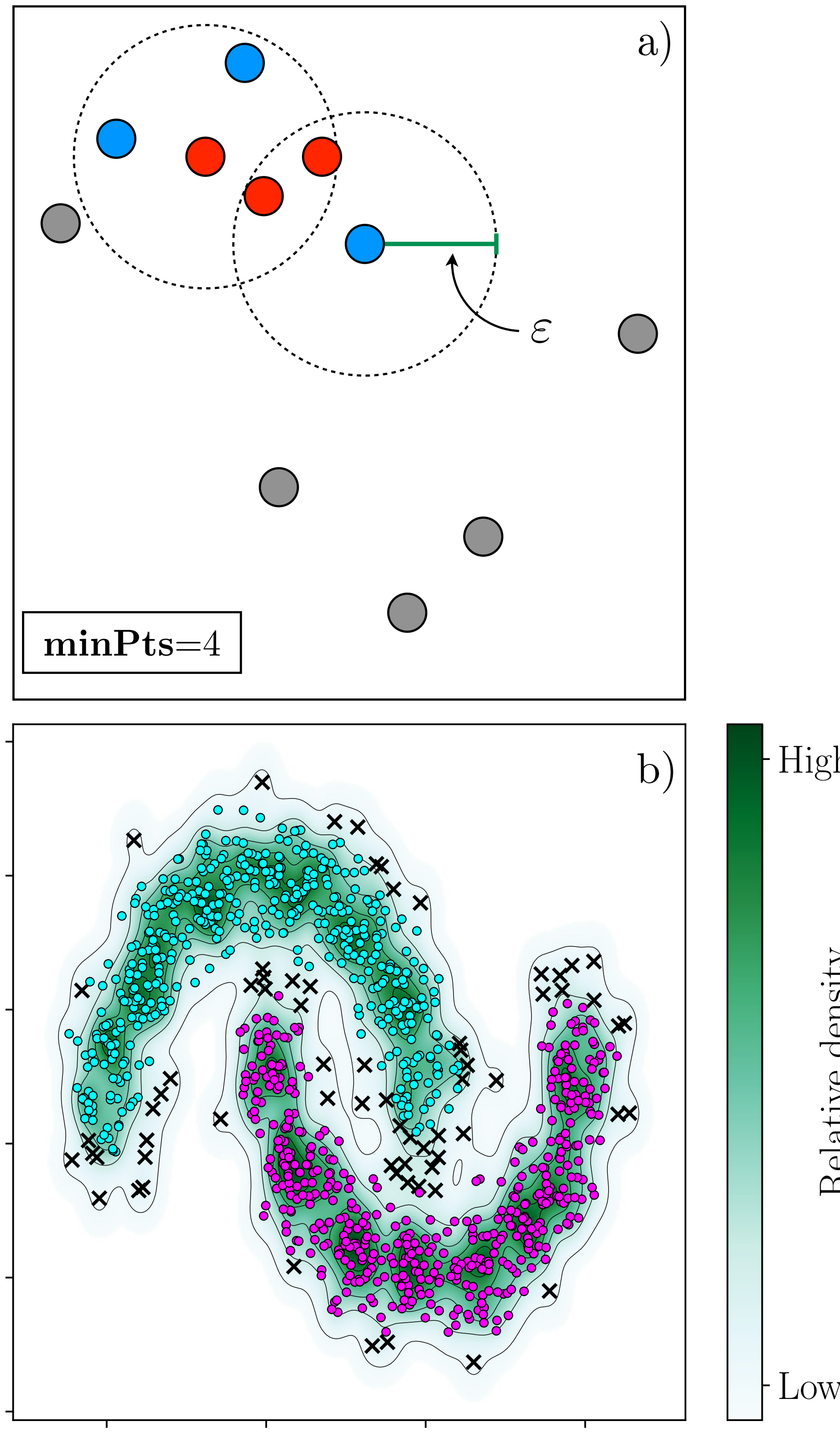
  - Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than minPts neighbors.

  - Find the connected components of core points on the neighbor graph, ignoring all non-core points.

  - Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise.

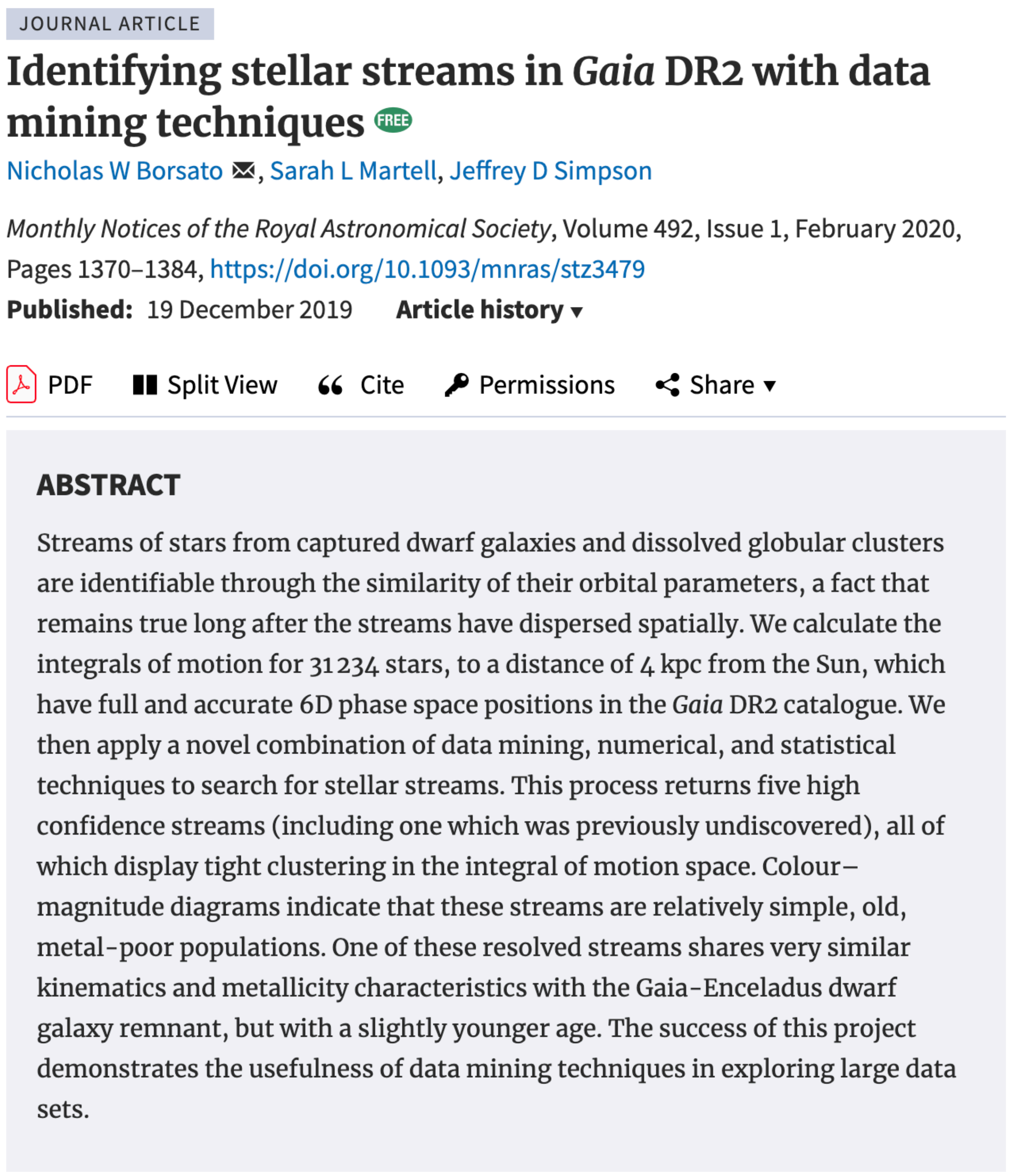
DBScan Algorithm

- Do not need to specify # clusters but only the hyperparameters ϵ and **minPts**.
- Scalable to large datasets as computational cost $\sim \mathcal{O}(N \log N)$.
- Note cluster with different shapes and sizes.
- Crosses are outliers.



nice visualization: https://www.youtube.com/watch?v=RDZUdRSDOok&t=180s&ab_channel=StatQuestwithJoshStarmer

Application in physics: Stellar streams



JOURNAL ARTICLE

Identifying stellar streams in *Gaia* DR2 with data mining techniques

FREE

Nicholas W Borsato, Sarah L Martell, Jeffrey D Simpson

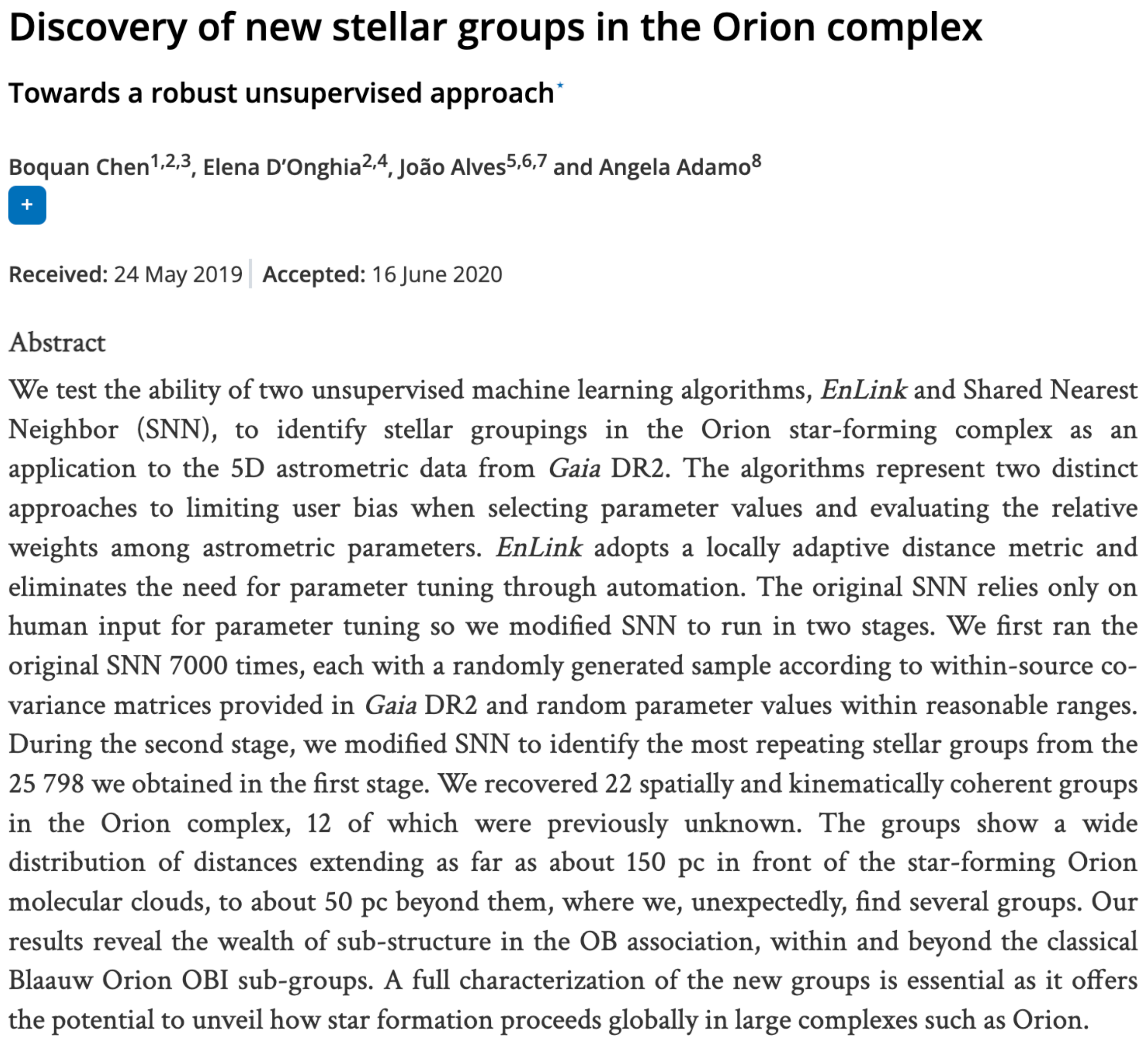
Monthly Notices of the Royal Astronomical Society, Volume 492, Issue 1, February 2020, Pages 1370–1384, https://doi.org/10.1093/mnras/stz3479

Published: 19 December 2019 **Article history** ▾

PDF Split View Cite Permissions Share ▾

ABSTRACT

Streams of stars from captured dwarf galaxies and dissolved globular clusters are identifiable through the similarity of their orbital parameters, a fact that remains true long after the streams have dispersed spatially. We calculate the integrals of motion for 31 234 stars, to a distance of 4 kpc from the Sun, which have full and accurate 6D phase space positions in the *Gaia* DR2 catalogue. We then apply a novel combination of data mining, numerical, and statistical techniques to search for stellar streams. This process returns five high confidence streams (including one which was previously undiscovered), all of which display tight clustering in the integral of motion space. Colour–magnitude diagrams indicate that these streams are relatively simple, old, metal-poor populations. One of these resolved streams shares very similar kinematics and metallicity characteristics with the Gaia-Enceladus dwarf galaxy remnant, but with a slightly younger age. The success of this project demonstrates the usefulness of data mining techniques in exploring large data sets.



Discovery of new stellar groups in the Orion complex

Towards a robust unsupervised approach[*]

Boquan Chen[1,2,3], Elena D'Onghia[2,4], João Alves[5,6,7] and Angela Adamo[8]

+

Received: 24 May 2019 | **Accepted:** 16 June 2020

Abstract

We test the ability of two unsupervised machine learning algorithms, *EnLink* and Shared Nearest Neighbor (SNN), to identify stellar groupings in the Orion star-forming complex as an application to the 5D astrometric data from *Gaia* DR2. The algorithms represent two distinct approaches to limiting user bias when selecting parameter values and evaluating the relative weights among astrometric parameters. *EnLink* adopts a locally adaptive distance metric and eliminates the need for parameter tuning through automation. The original SNN relies only on human input for parameter tuning so we modified SNN to run in two stages. We first ran the original SNN 7000 times, each with a randomly generated sample according to within-source co-variance matrices provided in *Gaia* DR2 and random parameter values within reasonable ranges. During the second stage, we modified SNN to identify the most repeating stellar groups from the 25 798 we obtained in the first stage. We recovered 22 spatially and kinematically coherent groups in the Orion complex, 12 of which were previously unknown. The groups show a wide distribution of distances extending as far as about 150 pc in front of the star-forming Orion molecular clouds, to about 50 pc beyond them, where we, unexpectedly, find several groups. Our results reveal the wealth of sub-structure in the OB association, within and beyond the classical Blaauw Orion OBI sub-groups. A full characterization of the new groups is essential as it offers the potential to unveil how star formation proceeds globally in large complexes such as Orion.

UW Madison Astronomy Dept.

Course logistics

- **Reading for this lecture:**
    - This lecture was mostly based on arxiv:1803.08823