Dimensionality Reduction Algorithms

(and how to interpret their output)

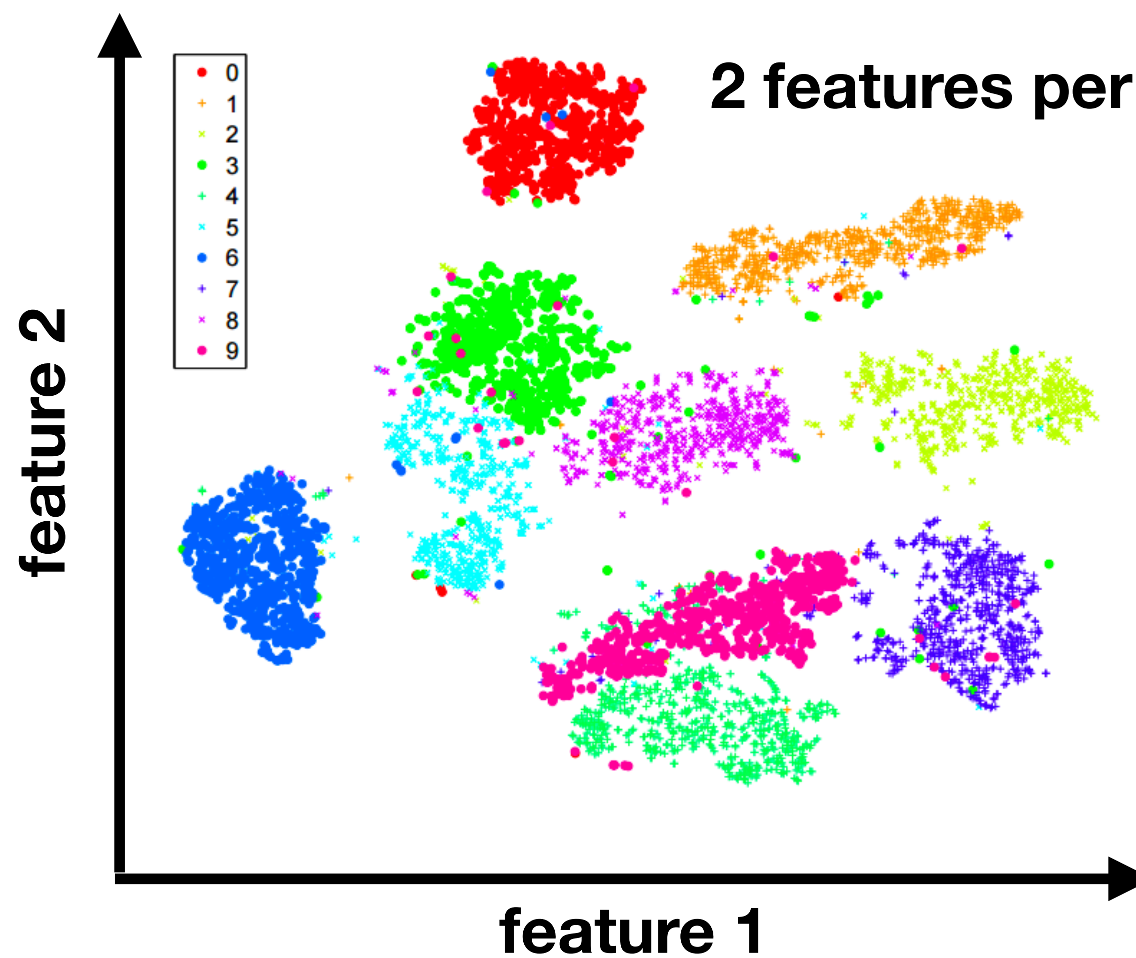
Dalya Baron (Tel Aviv University)

XXX Winter School, November 2018

What is Dimensionality Reduction?

28 x 28 features per object

Dimensionality Reduction algorithm



2 features per object

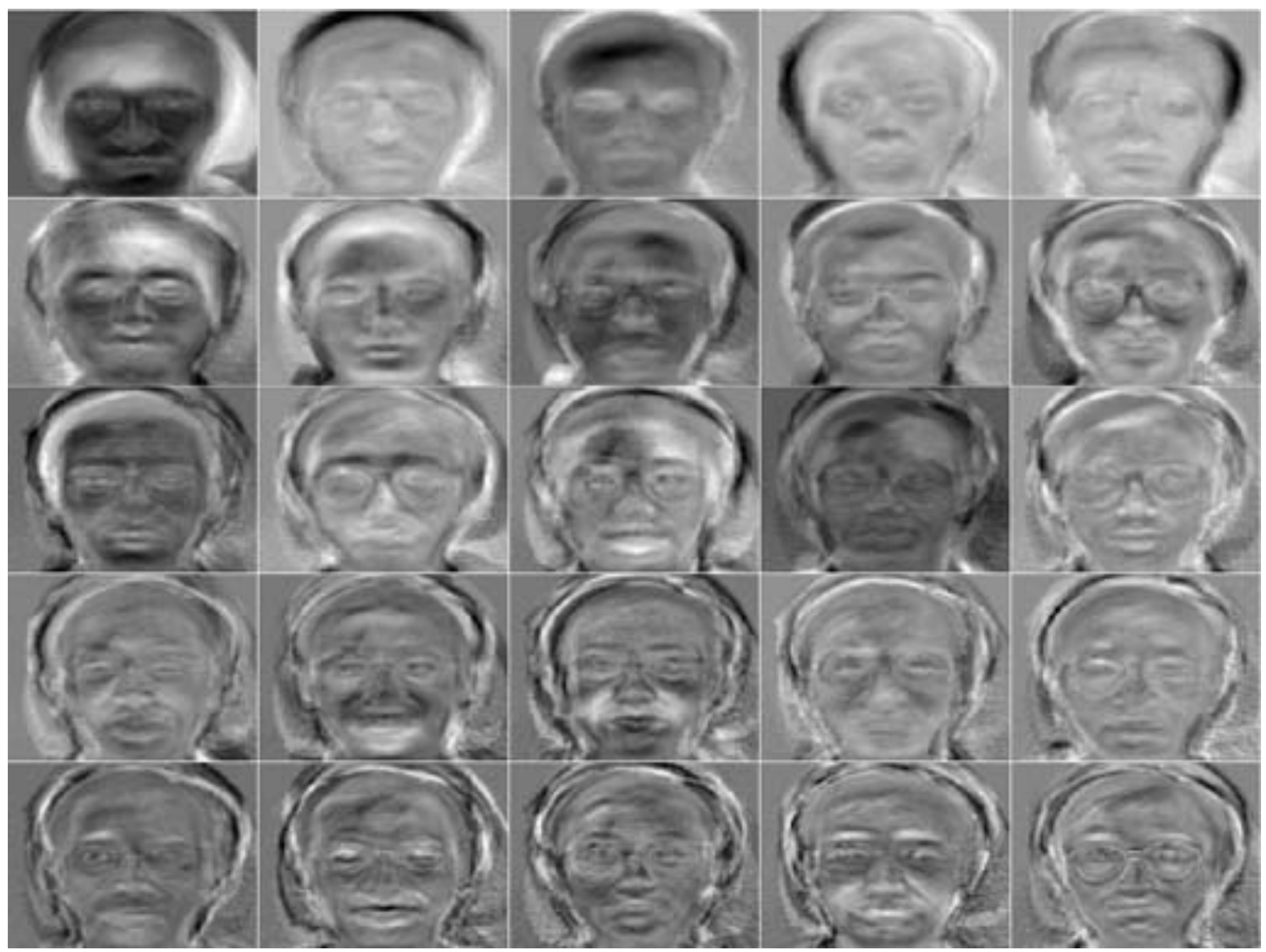
Why do we need dimensionality reduction?

- **“Practical”:**
  - Improve performance of supervised learning algorithms: original features can be correlated and redundant, most algorithms cannot handle thousands of features.
  - Compressing data (e.g., SKA).
- **“Artistic”:**
  - Data visualization and interpretation.
  - Uncover complex trends.
  - Look for “unknown unknowns”.

Two types of dimensionality reduction

1. Decomposition of the objects into “prototypes”. Each object can be represented using the prototypes.

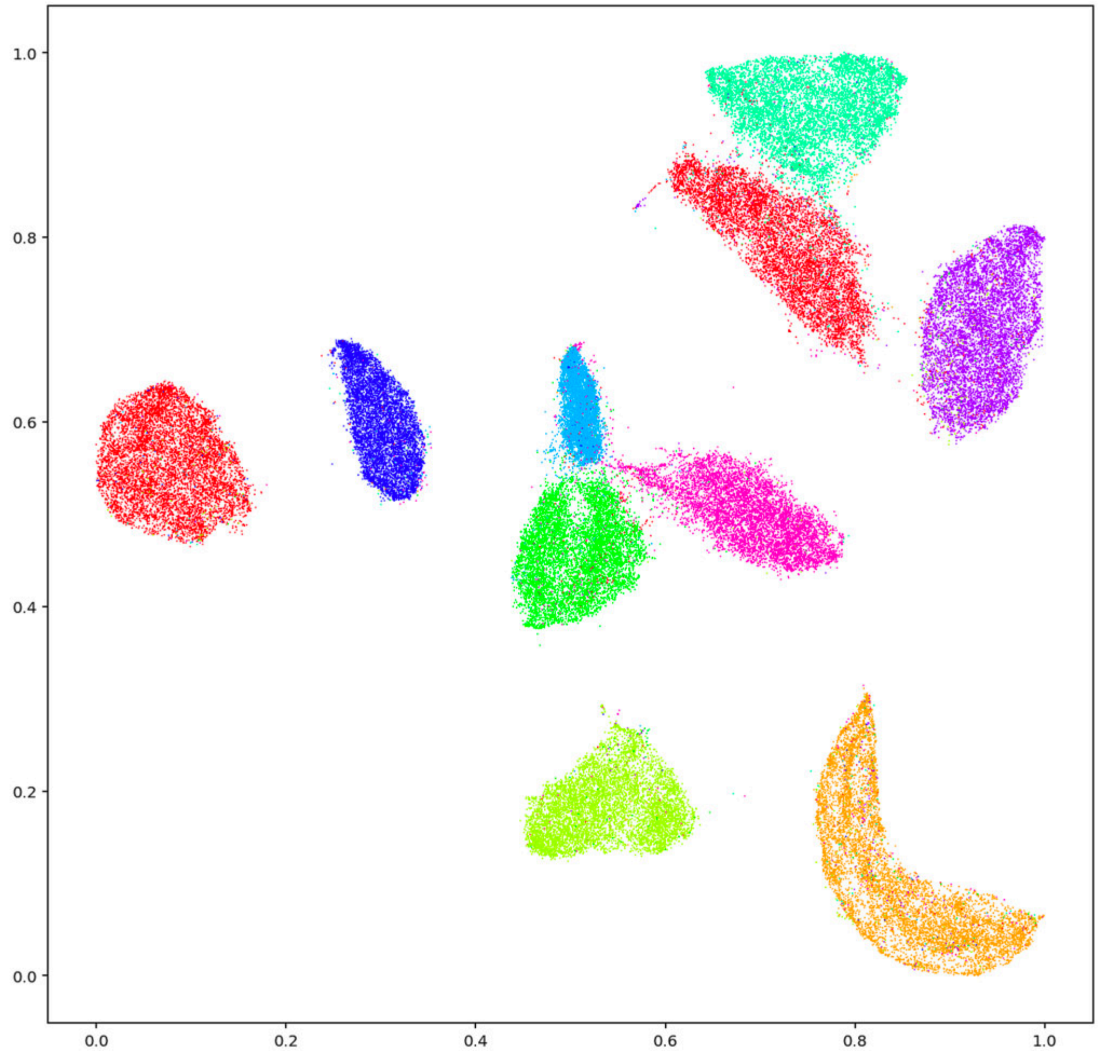
We gain: prototypes that represent the population **and** low-dimensional embedding.



For example: SVD, PCA, ICA, NNMF, SOM and more…

Two types of dimensionality reduction

2. Embedding of a high-dimensional dataset into a lower dimensional dataset.
We gain: low-dimensional embedding.

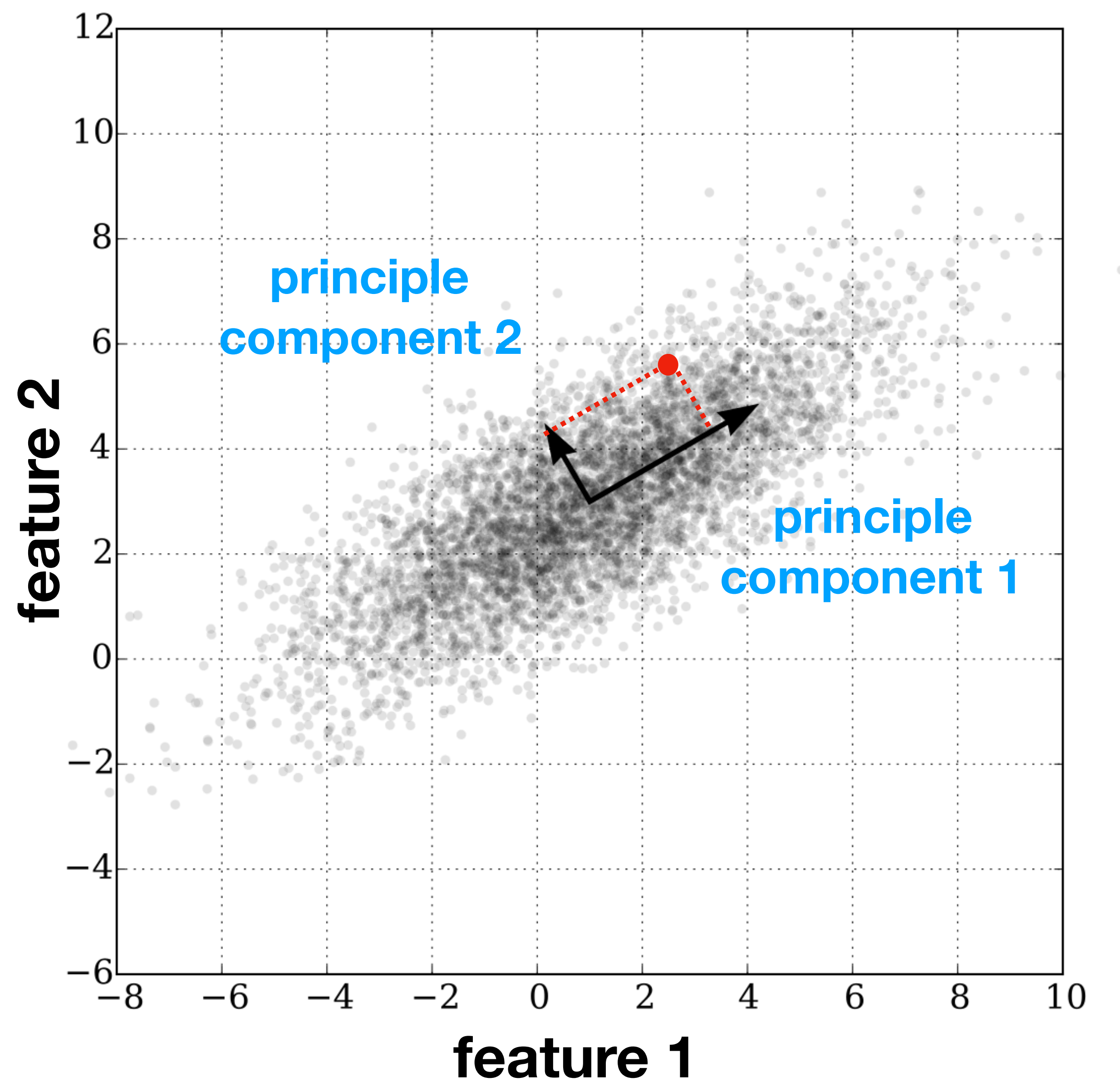


For example: tSNE, autoencoders

Principle Component Analysis (PCA)

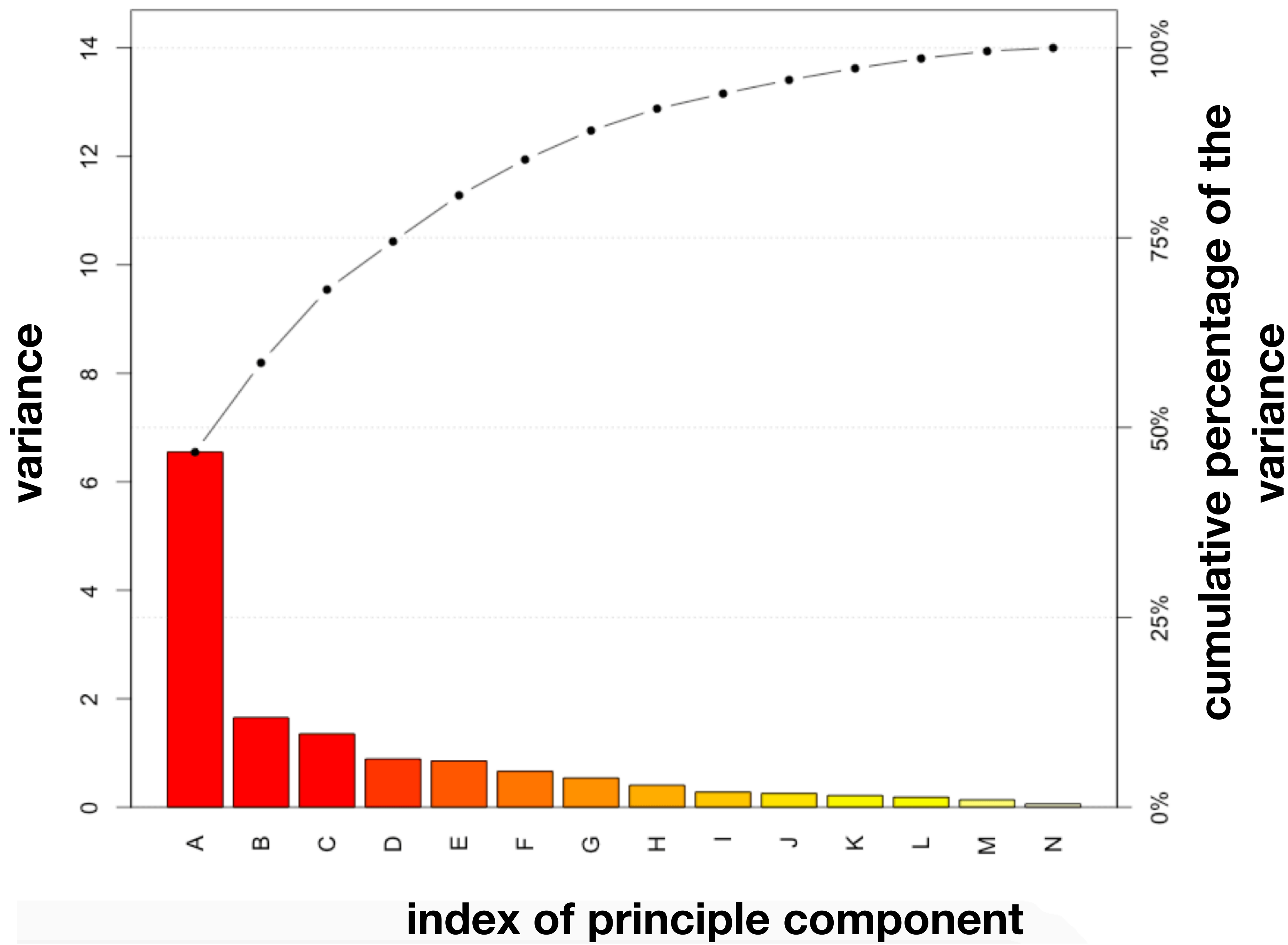
PCA is a transformation that converts a set of observations (possibly from correlated variables) into a set of values of linearly uncorrelated variables, called **principle components.**

- The first principle component has the largest possible **variance.**
- Each succeeding component has the highest possible variance, under the constrain that it is orthogonal to the preceding components.



Principle Component Analysis (PCA)

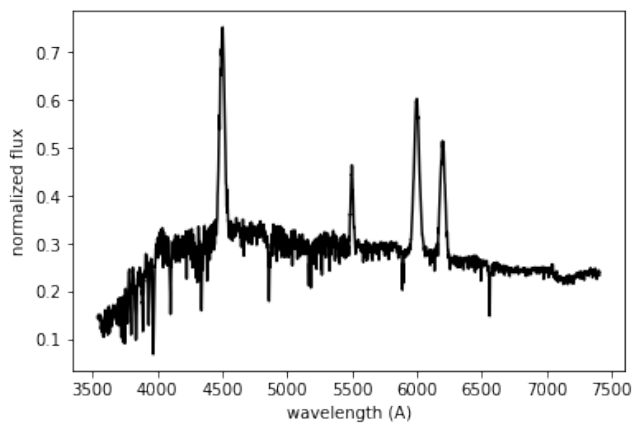
PCA allows us to compress the data, by representing each object as a projection on the first principle components.



Principle Component Analysis (PCA)

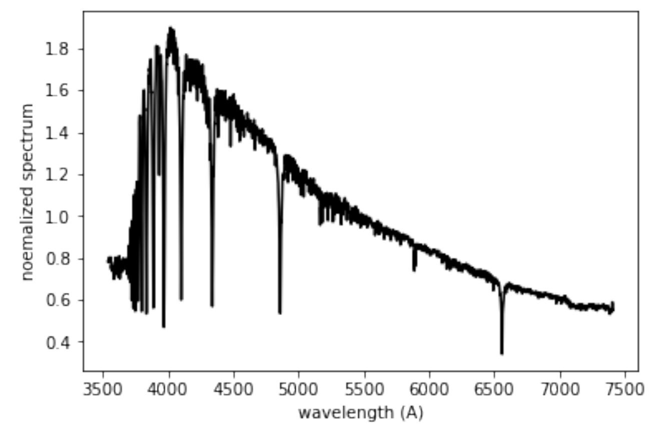
The principle components **may** represent the true **building blocks** of the objects in our dataset.

observed object



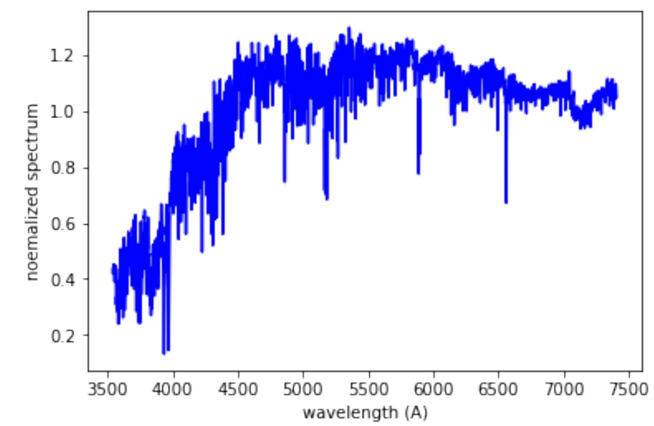
= A *

principle comp. 1



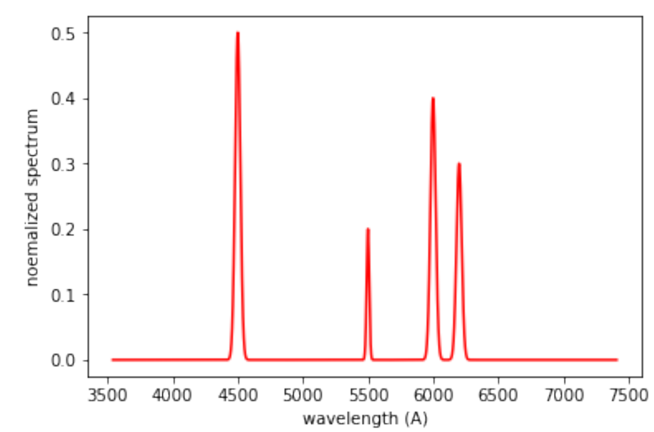
+ B *

principle comp. 2



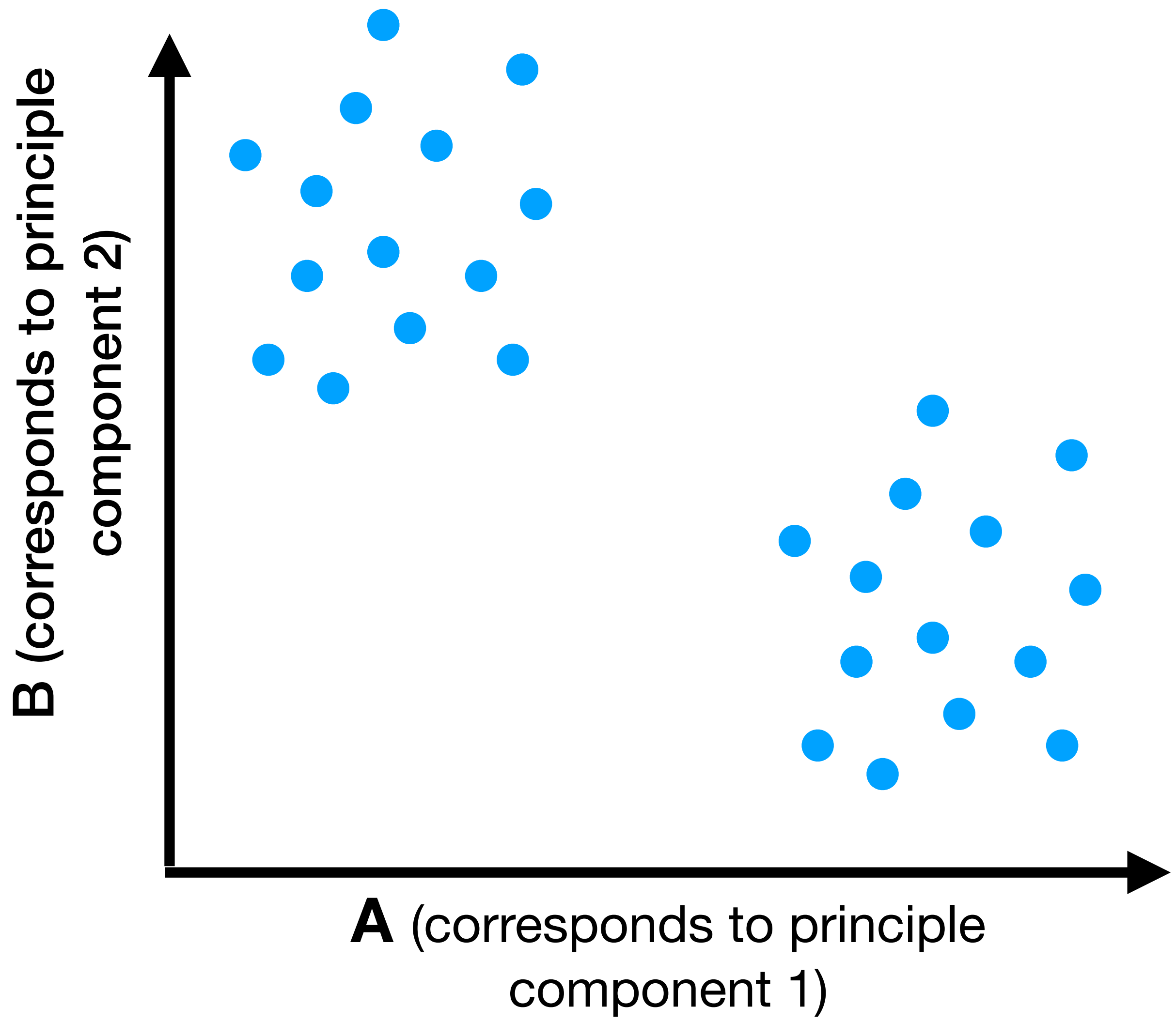
+ C *

principle comp. 3

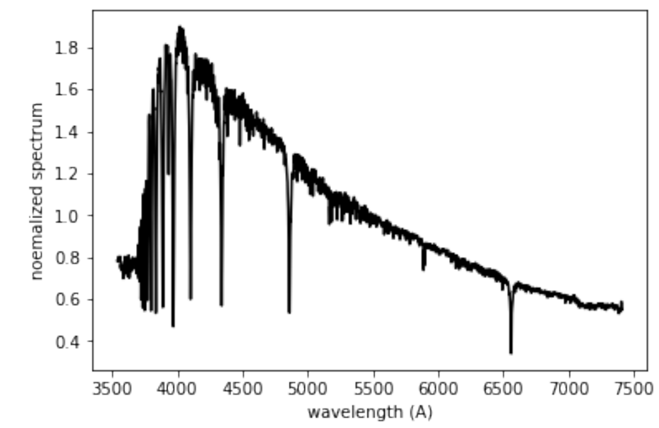


Principle Component Analysis (PCA)

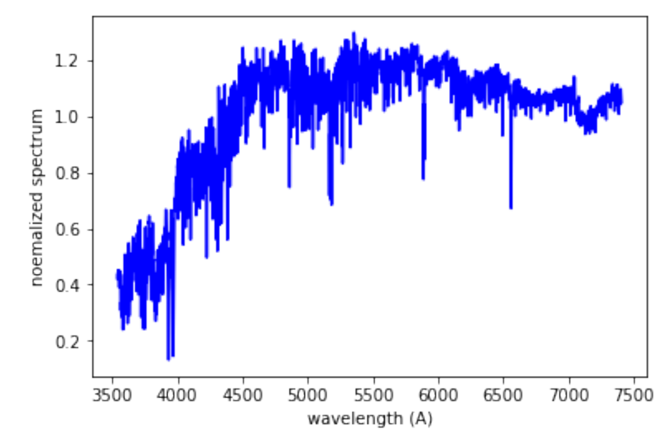
The projection onto the principle components gives a low-dimensional representation of the objects in the sample.



principle comp. 1

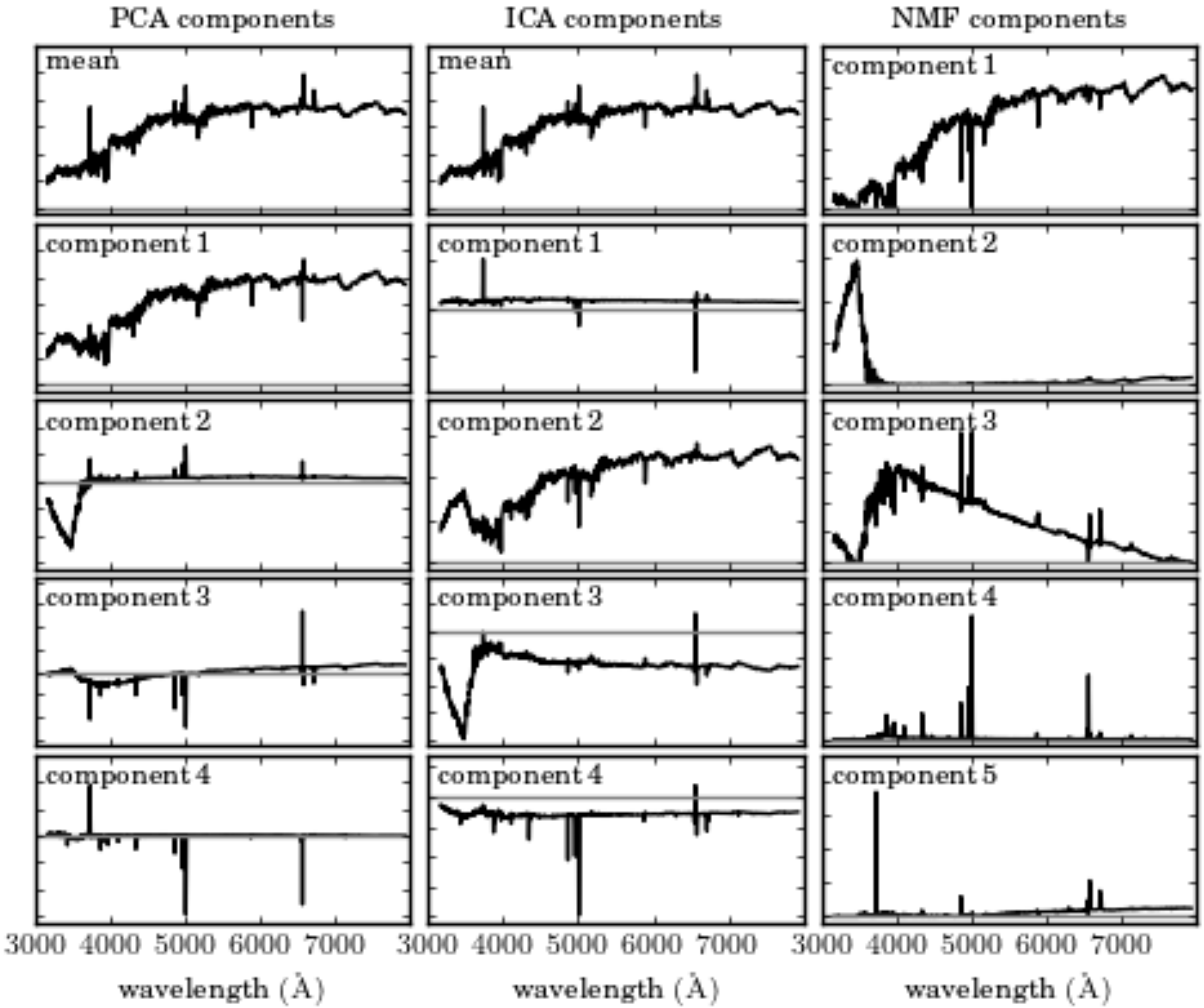


principle comp. 2



PCA: Pros & Cons

- **Advantages:**
  - Very simple and intuitive to use.
  - No free parameters!
  - Optimized to reduce variance.
- **Disadvantages:**
  - Linear decomposition: we will not be able to describe absorption lines, dust extinction, distance, etc..
  - Can produce negative principle components, which is not always physical in astronomy.



From: http://www.astroml.org/book_figures/chapter7/fig_spec_decompositions.html

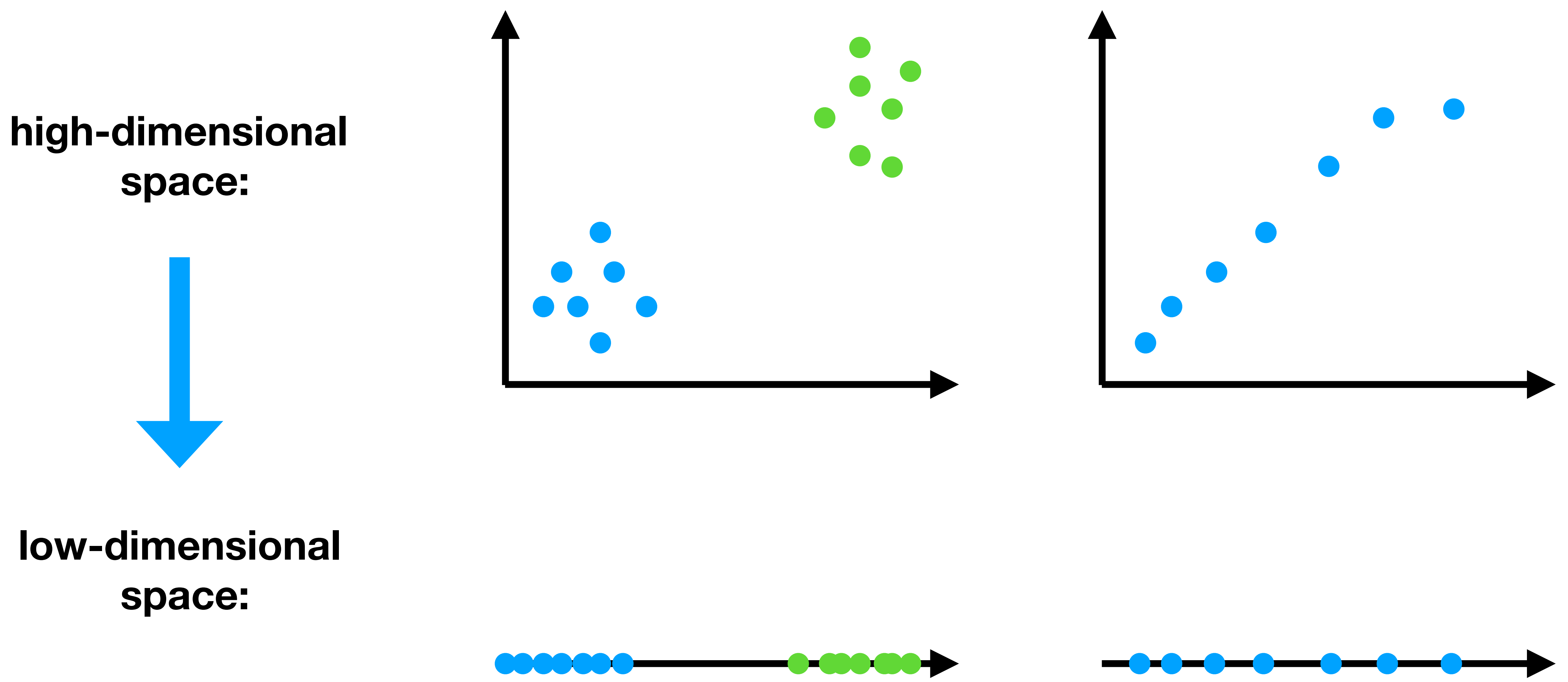
t-distributed stochastic neighbor embedding

(tSNE)

Embedding high-dimensional data in a low dimensional space (2 or 3)

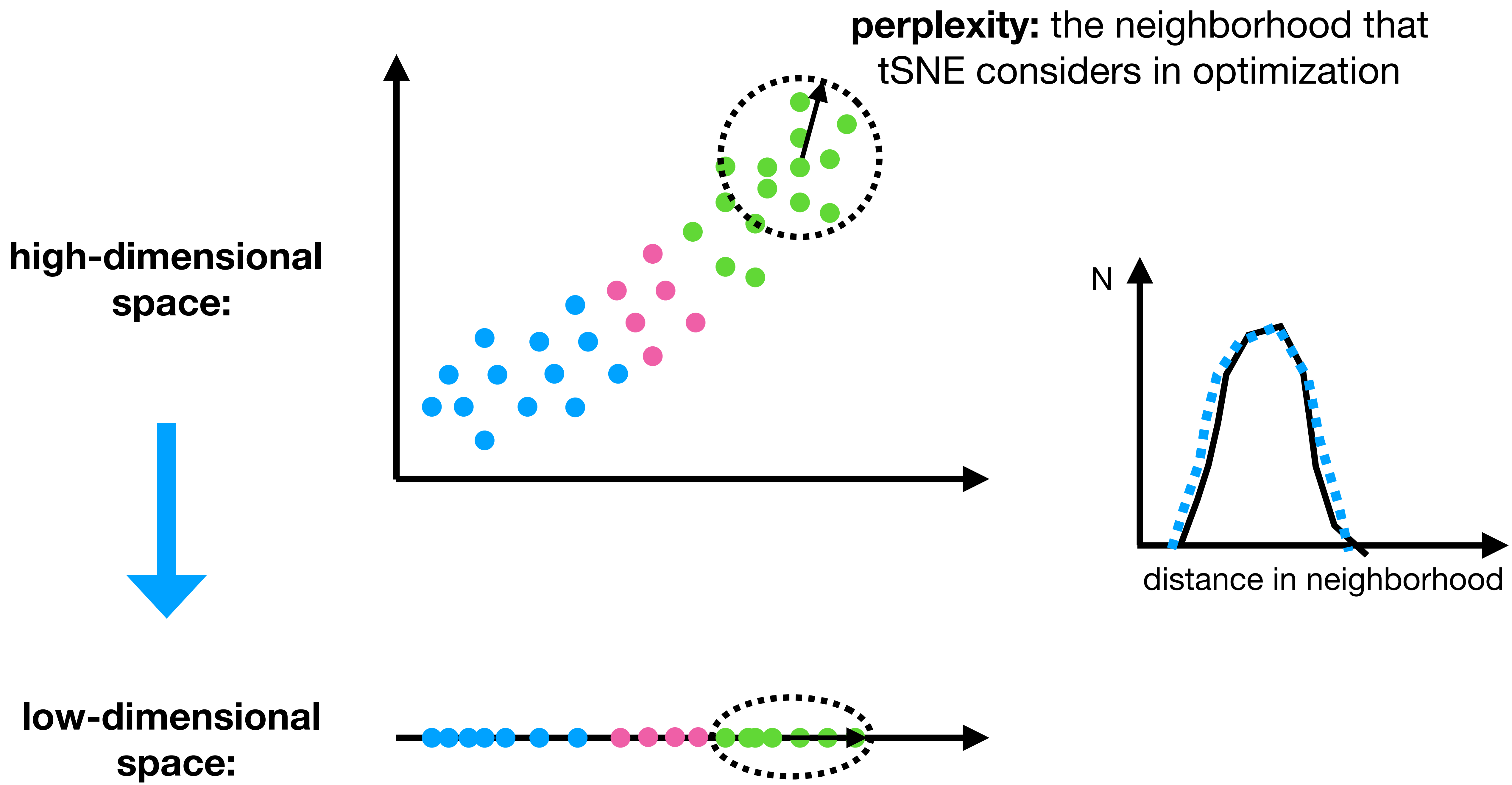
Input: (1) raw data, extracted features, or a distance matrix

(2) hyper-parameters: **perplexity**



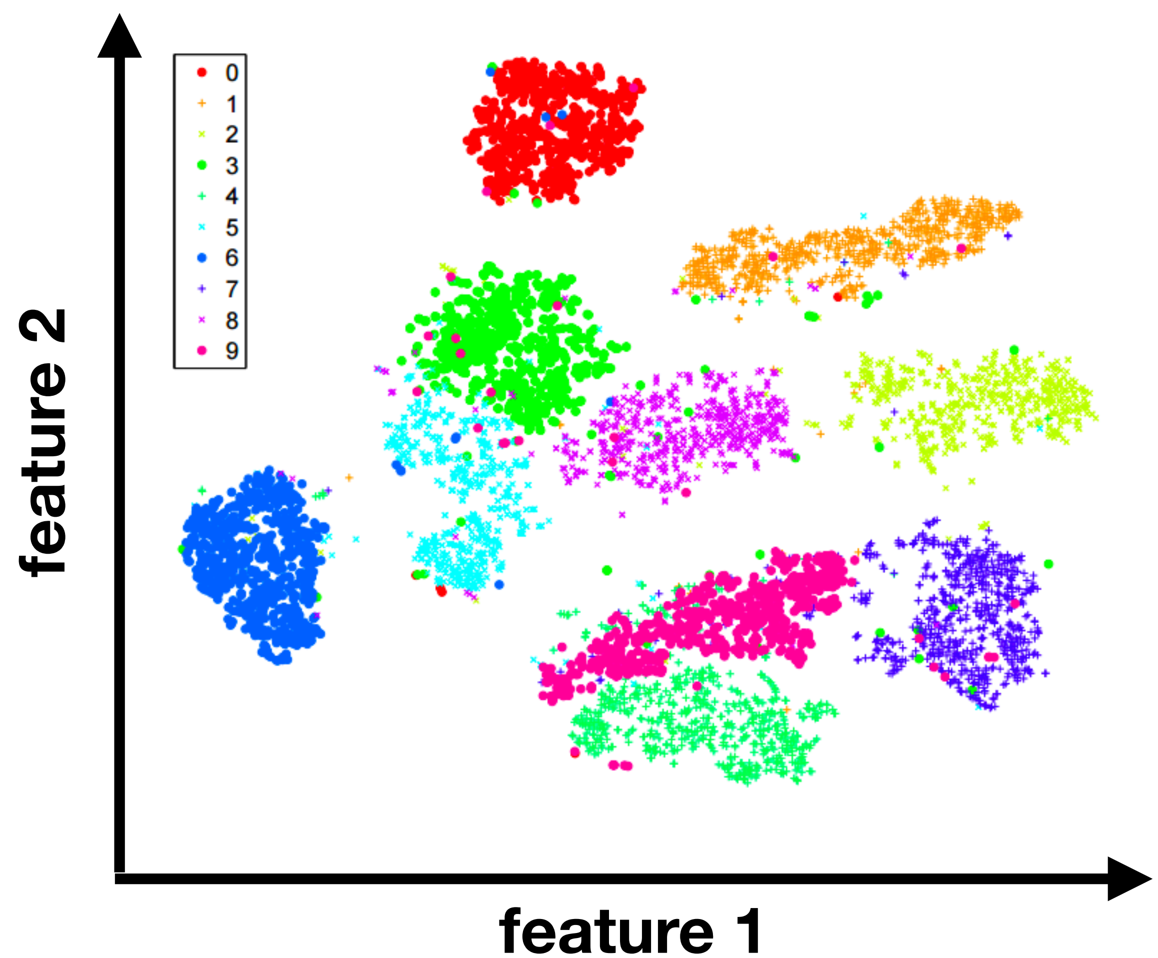
tSNE

Intuition: tSNE tries to find a low-dimensional embedding that preserves, as much as possible, the distribution of distances between different objects.



tSNE - example

28 x 28 features per object



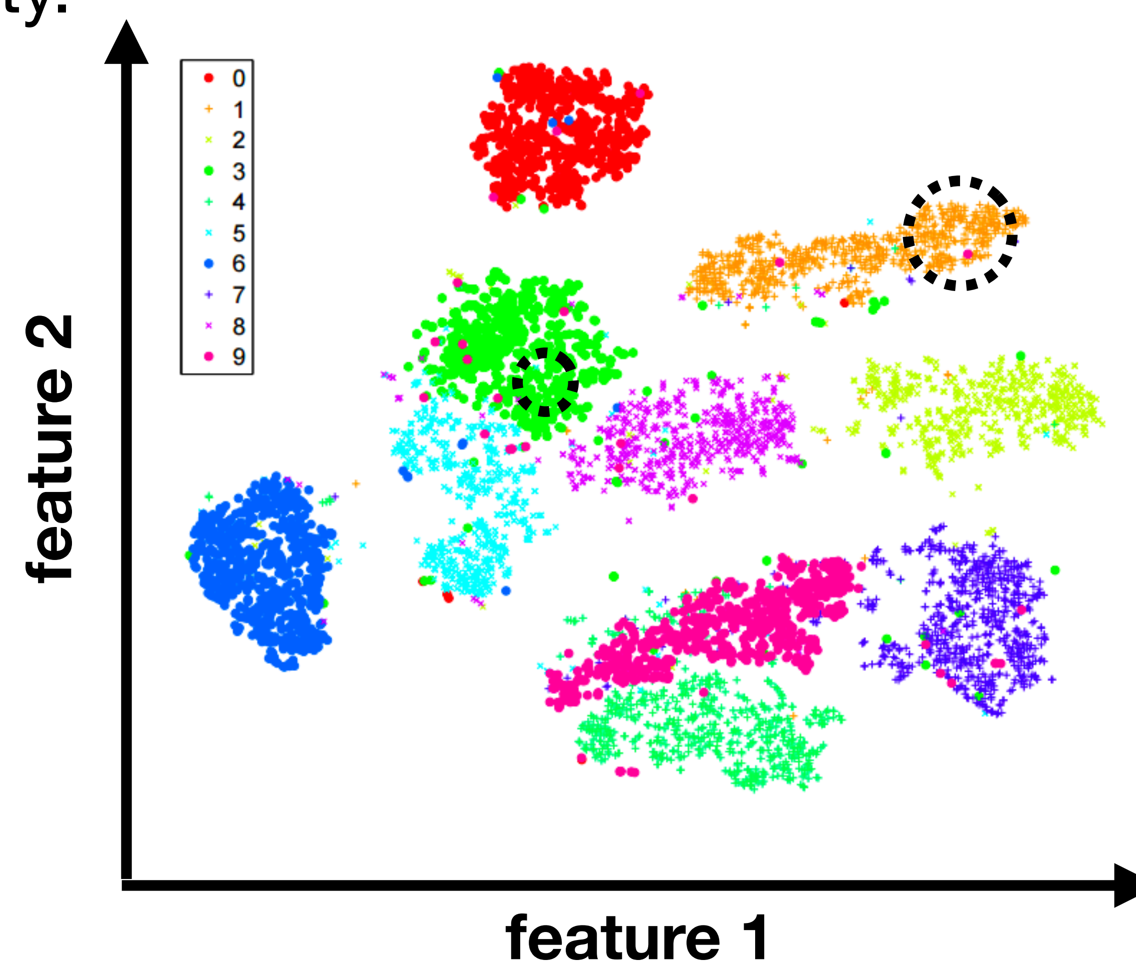
tSNE - example

https://distill.pub/2016/misread-tsne/

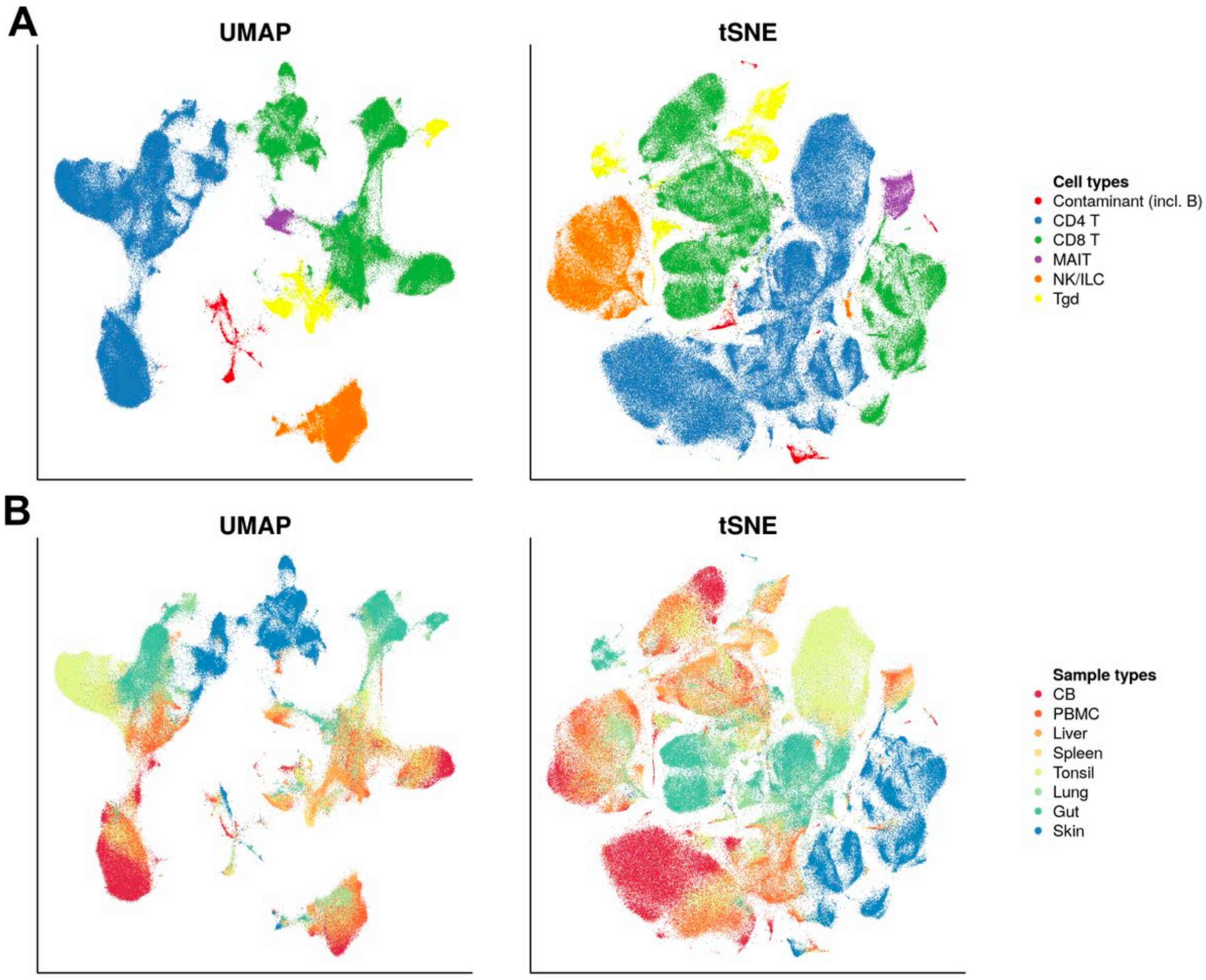
tSNE : Pros & Cons

- **Advantages:**
  - Can take as an input a general distance matrix.
  - Non-linear embedding.
  - Preserves high-dimensional clustering well (depending on the chosen perplexity).
- **Disadvantages:**
  - No prototypes.
  - Sensitive to distance scales < perplexity.
  - Large distances are meaningless.

28 x 28 features per object

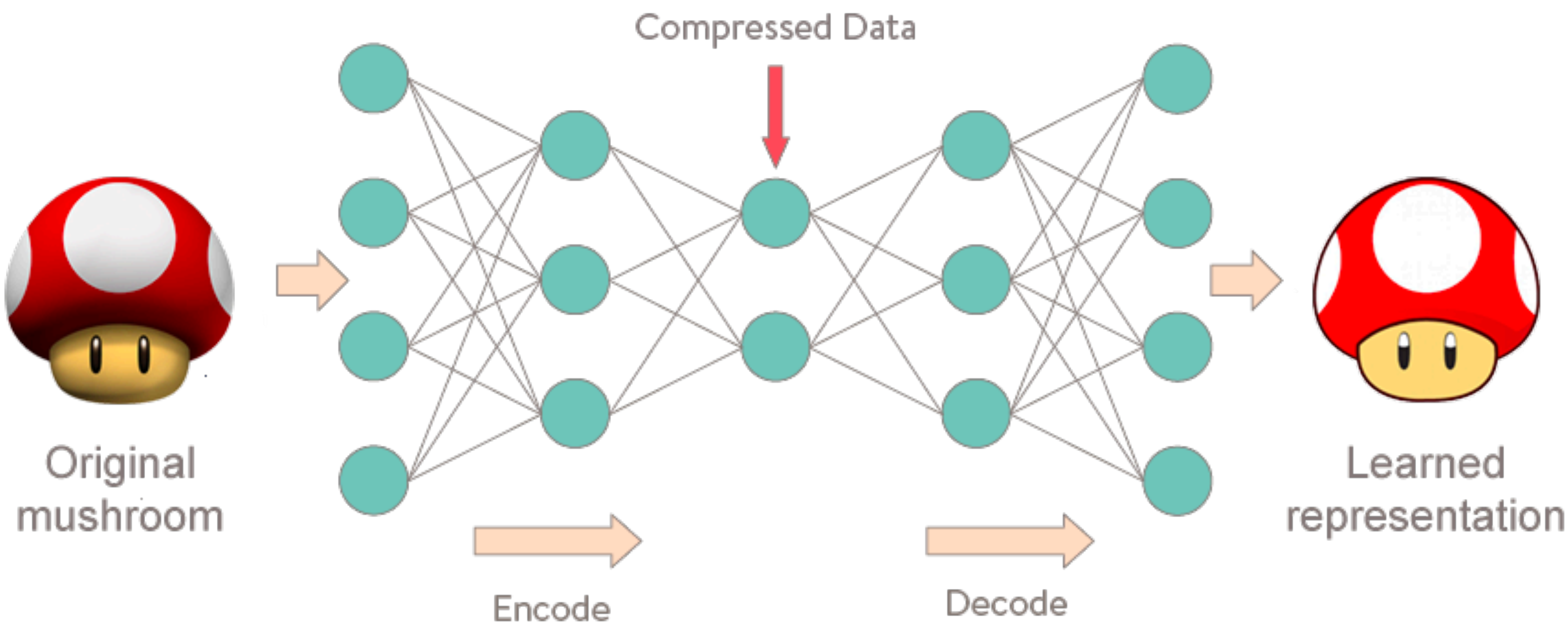


UMAP



See: https://arxiv.org/abs/1802.03426
https://github.com/lmcinnes/umap

Autoencoders



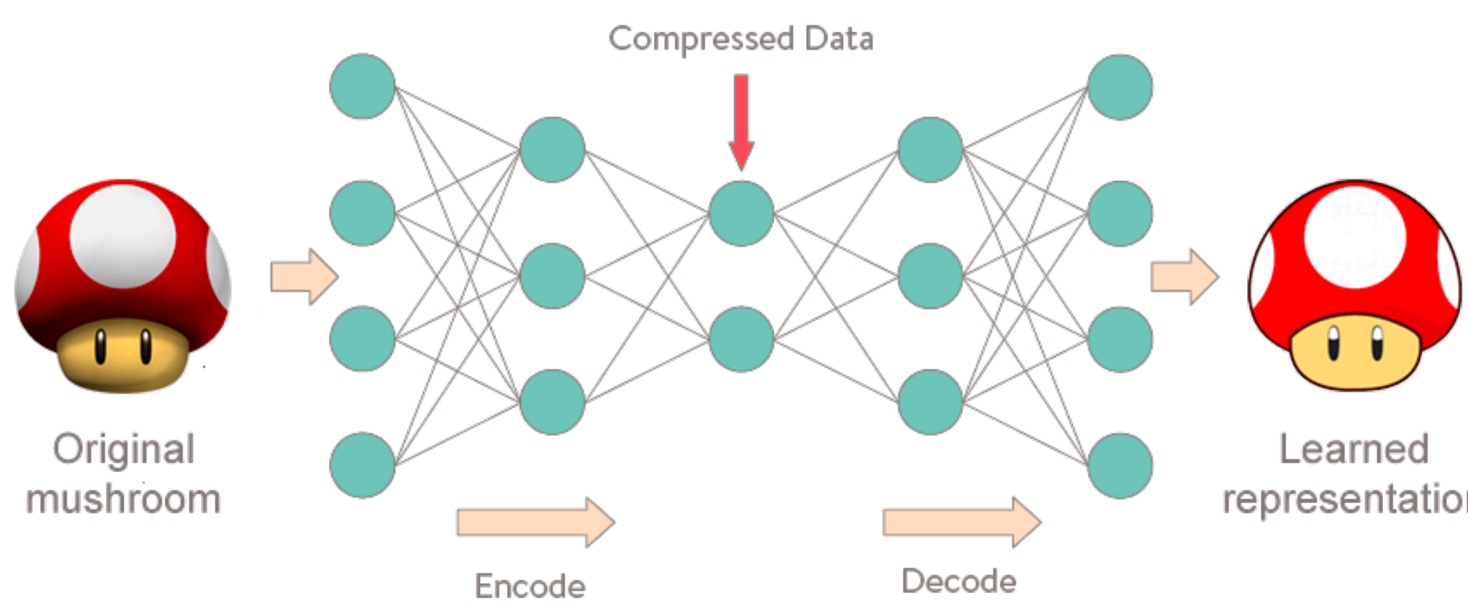
loss function = ( **-**  **)^2**

Original
mushroom

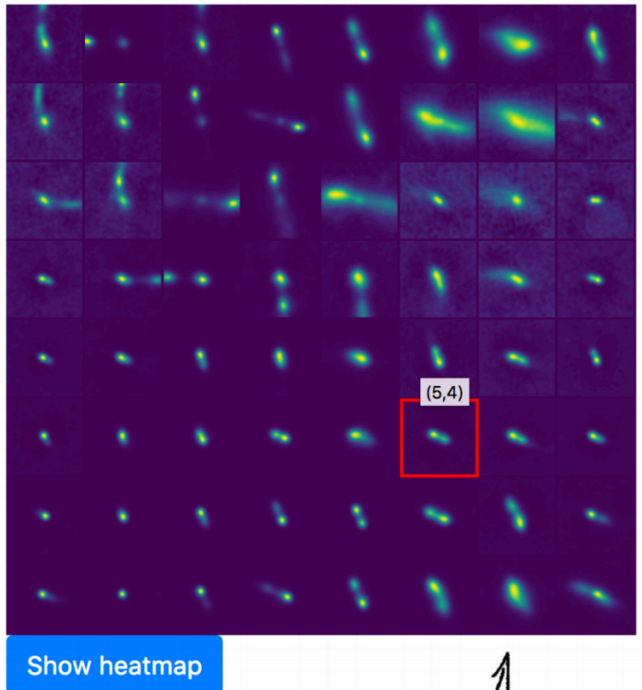
Learned
representation

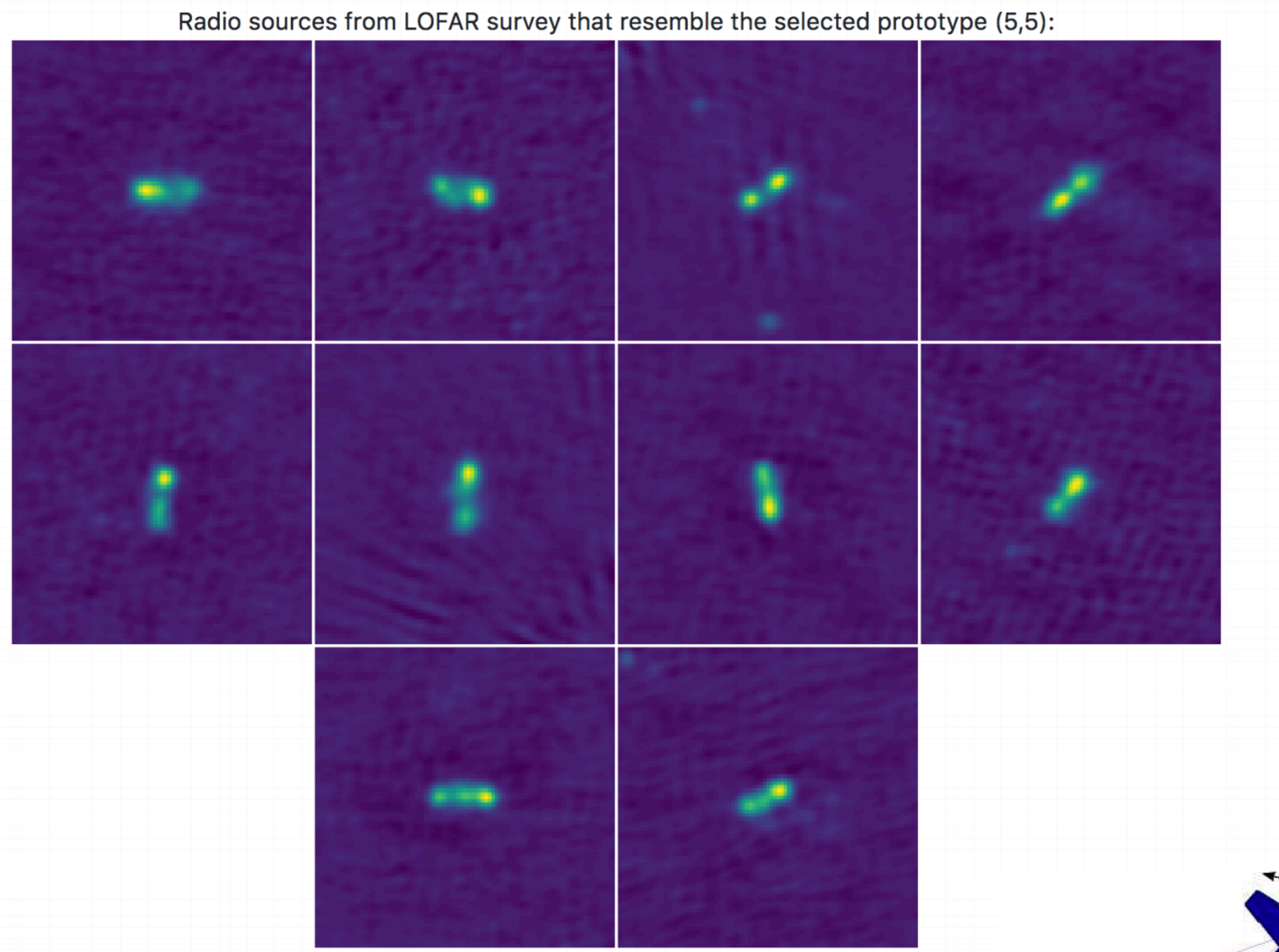
Autoencoders - Pros & Cons

- **Advantages:**
  - Can reduce the dimensions of raw images (CNN) or time-series (RNN)!
  - Can be used to produce an uncertainty on the embedding.
- **Disadvantages:**
  - No prototypes.
  - Complexity and interpretability.

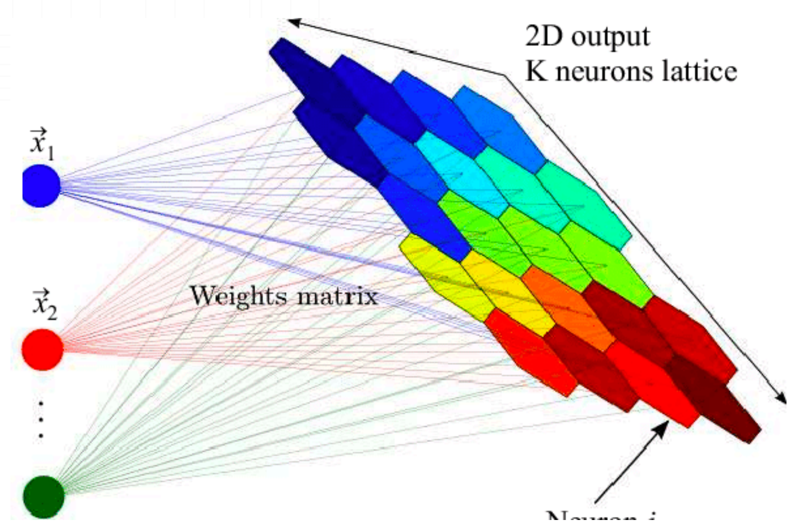


Self Organizing Maps (SOM) and PINK



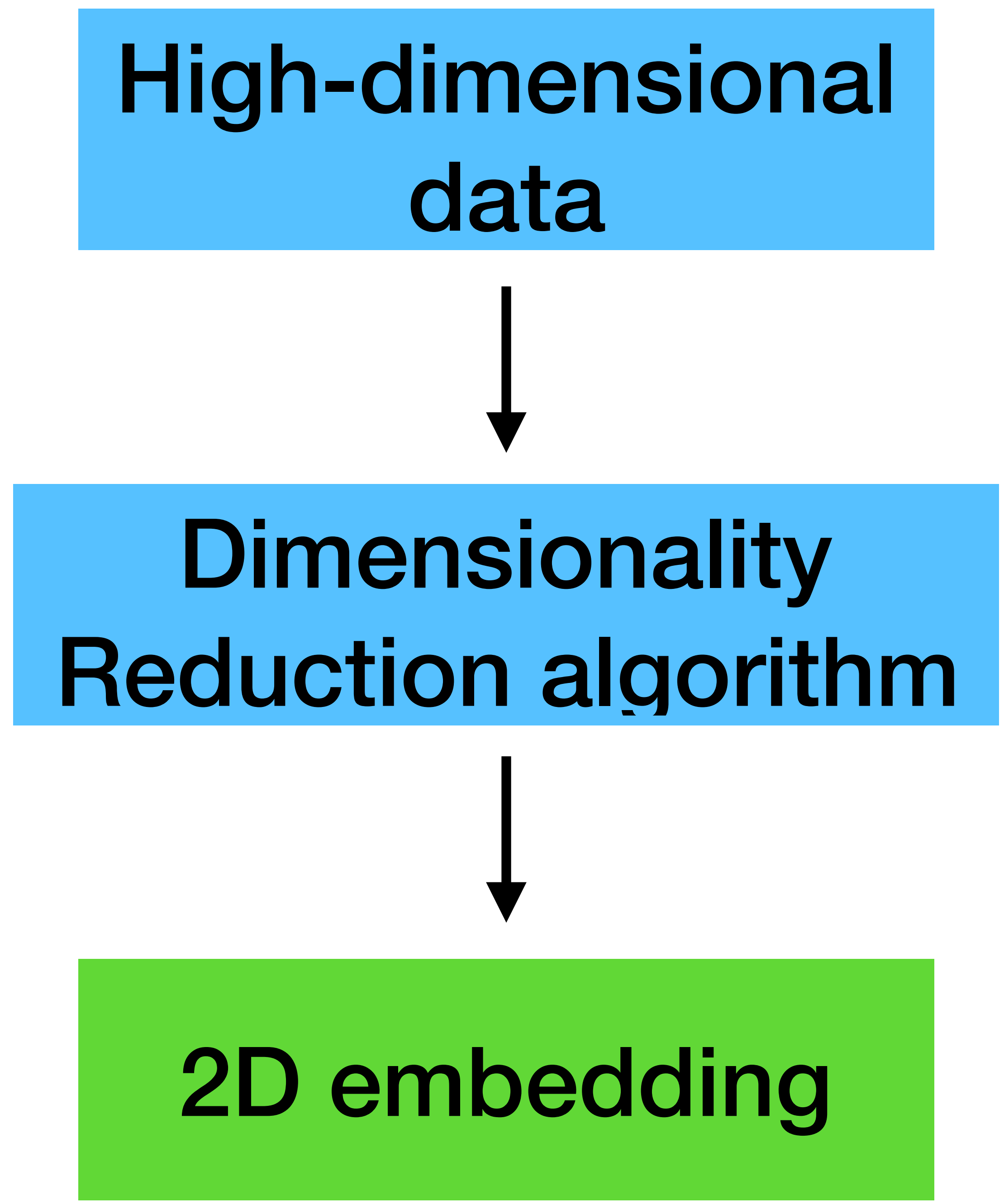


This is a Self-Organizing Map, trained on sources from the LOFAR survey. Click on one of these prototypes.



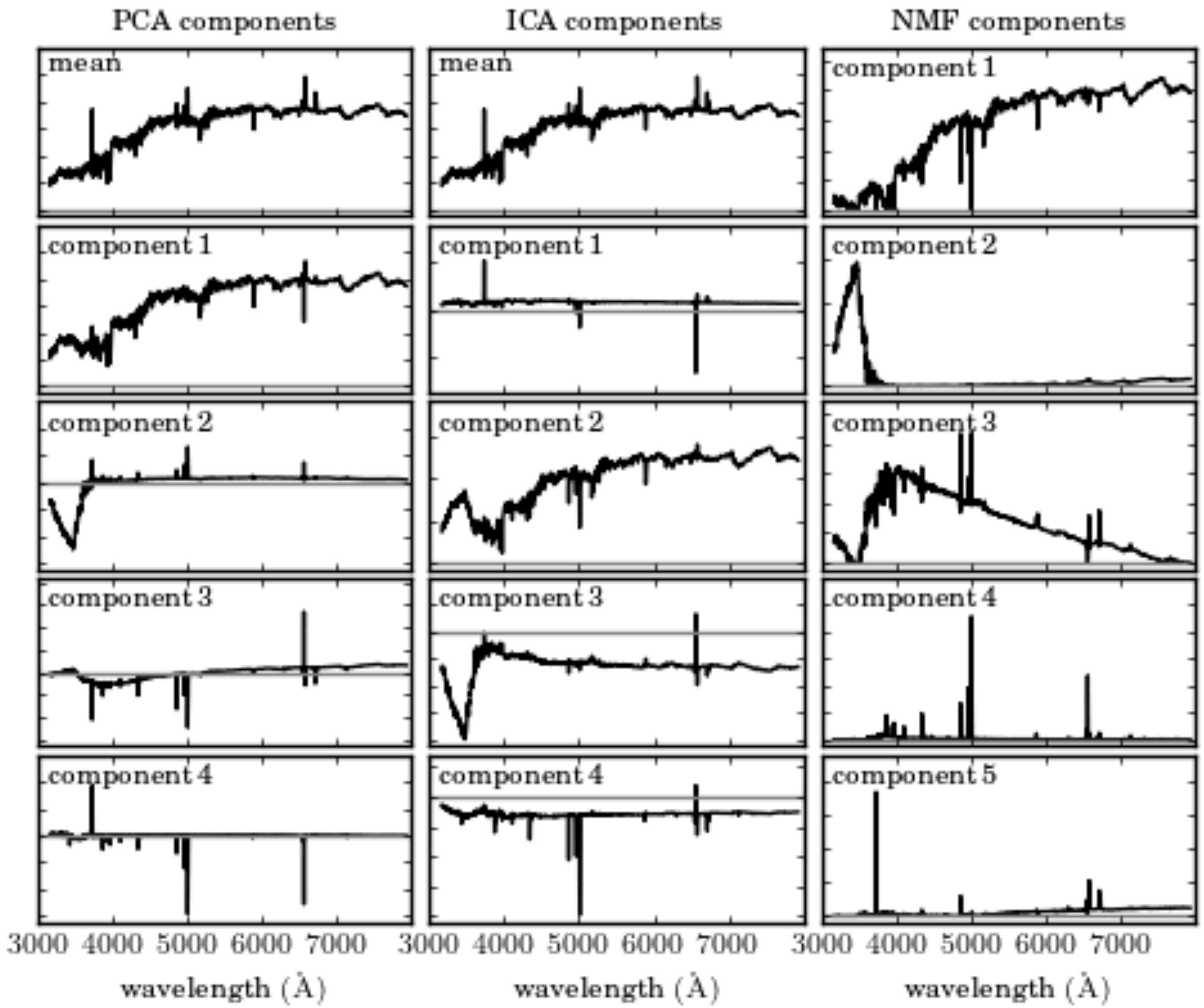
See: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-116.pdf
http://www.astron.nl/LifeCycle2018/Documents/Talks_Session1/Harwood_LifeCycle18.pdf

How to interpret the output of a dimensionality reduction algorithm?

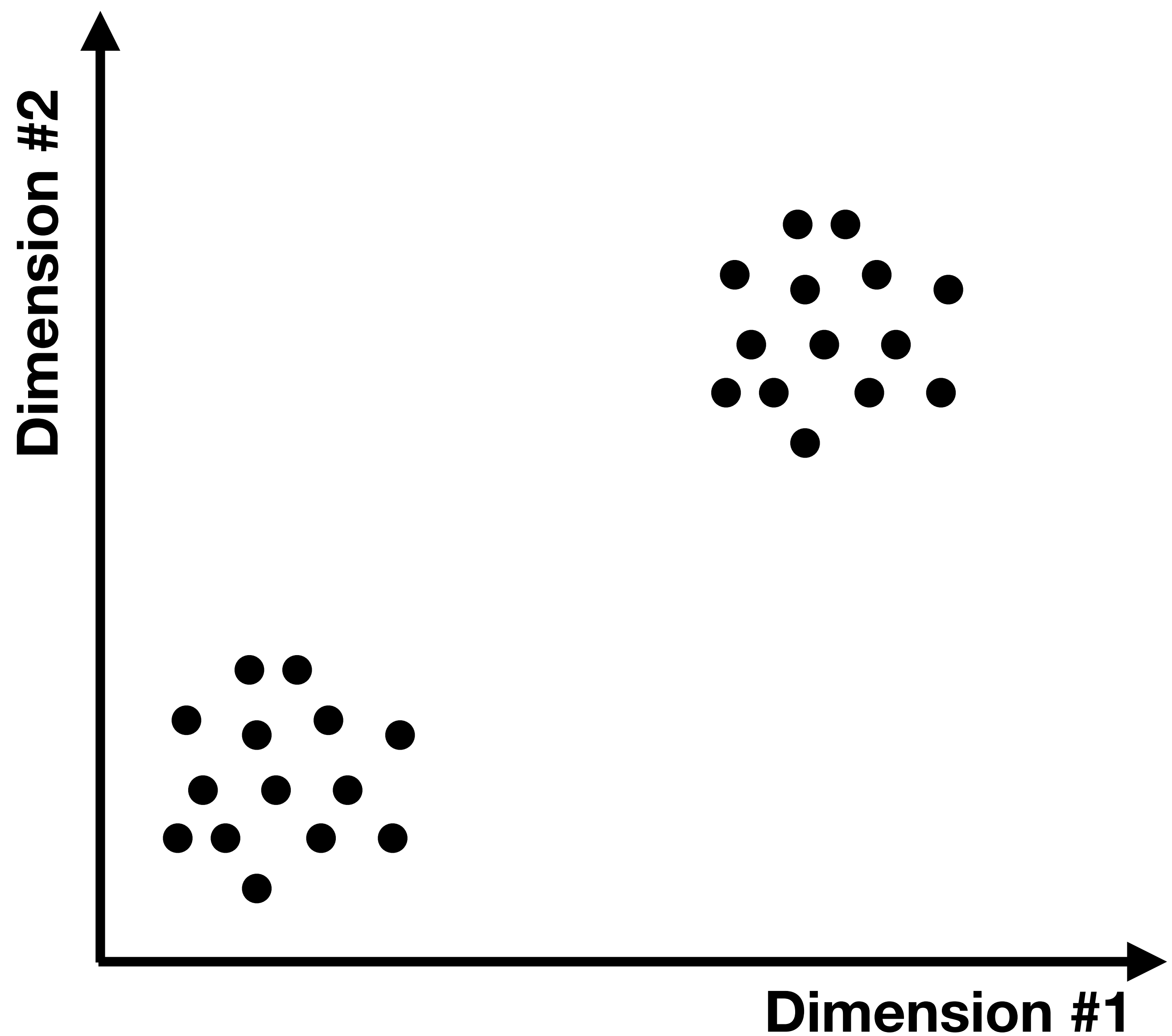


How to interpret the output of a dimensionality reduction algorithm?

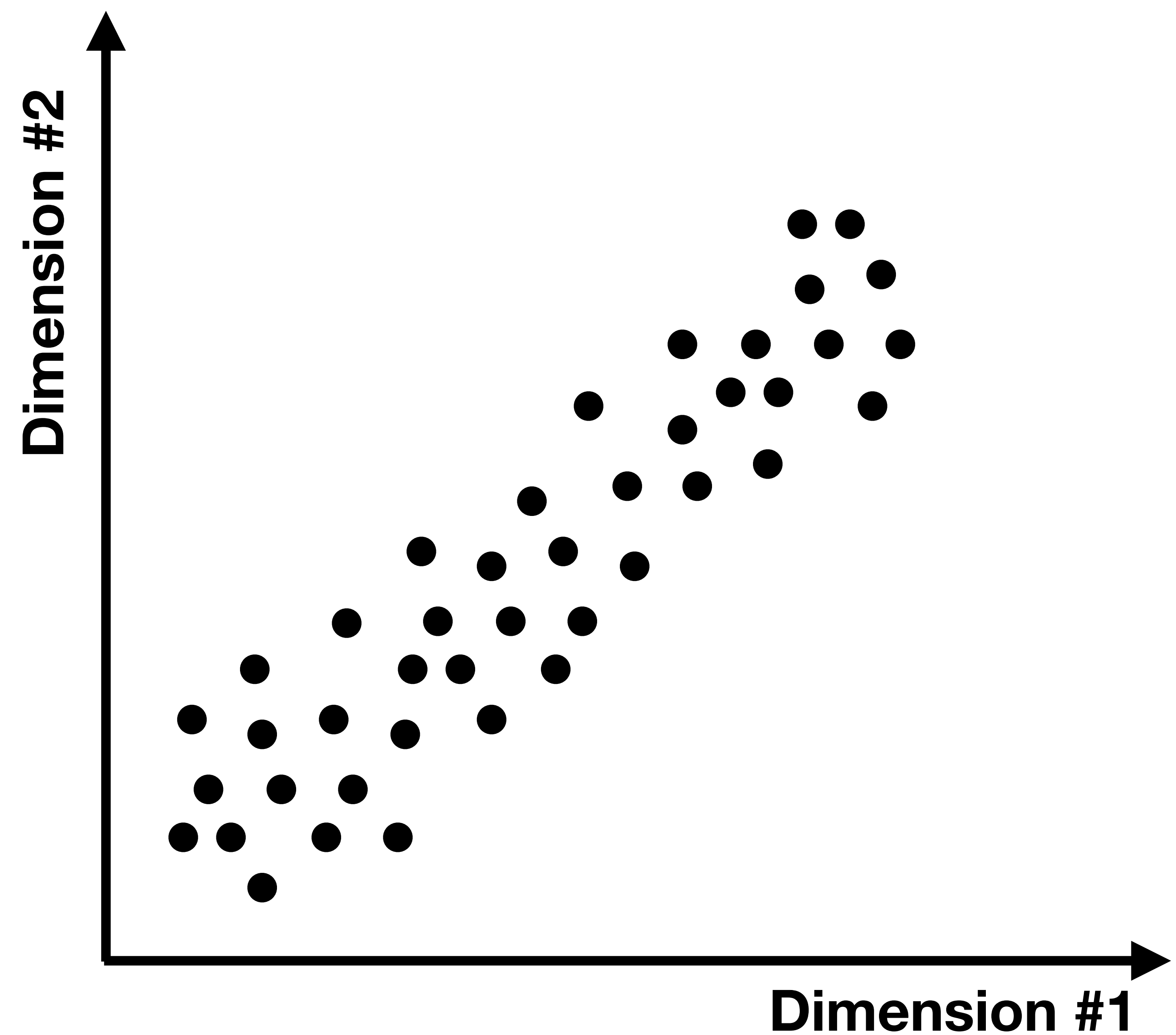
If we have prototypes - try to understand what they mean

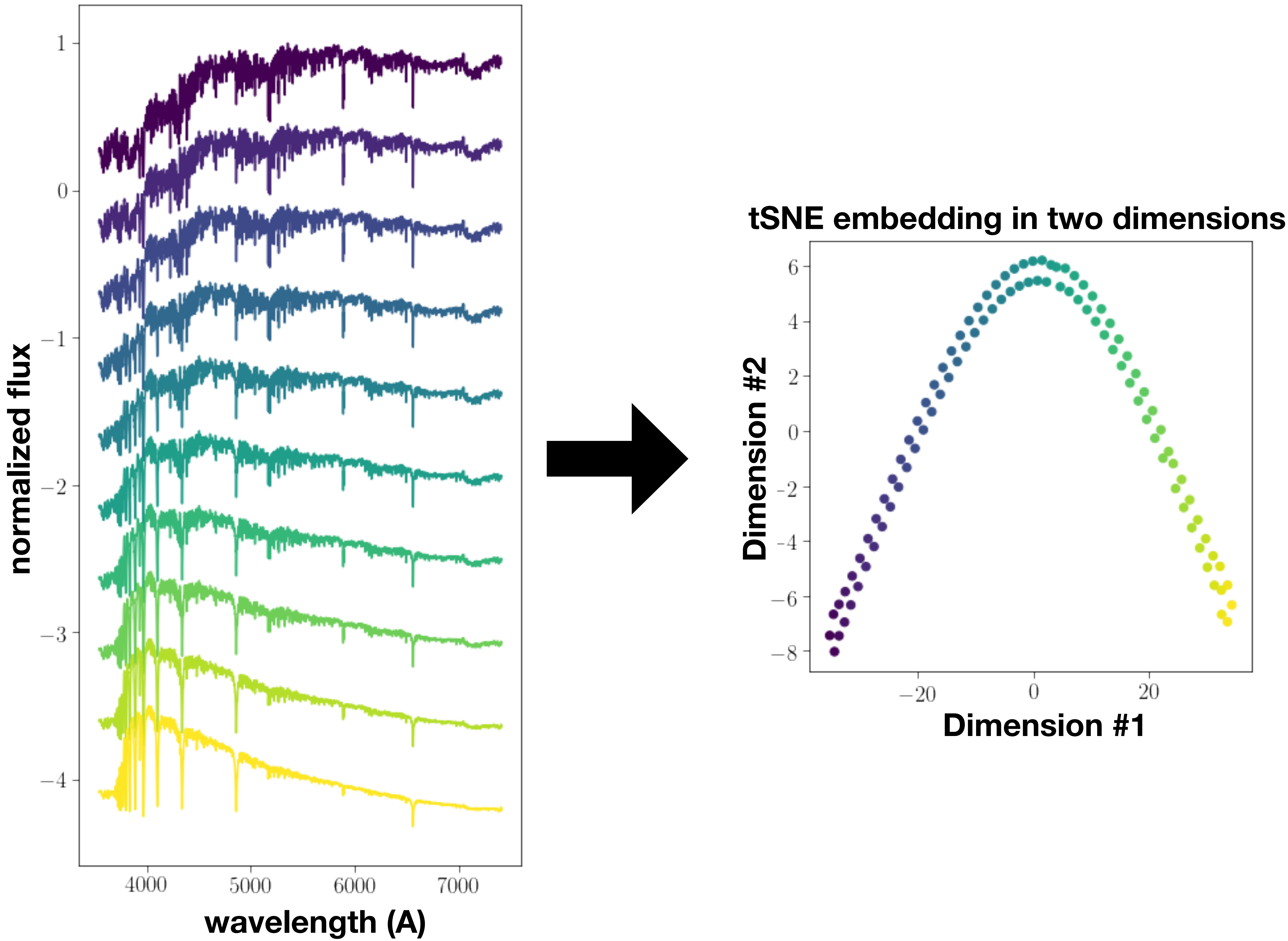


How to interpret the output of a dimensionality reduction algorithm?



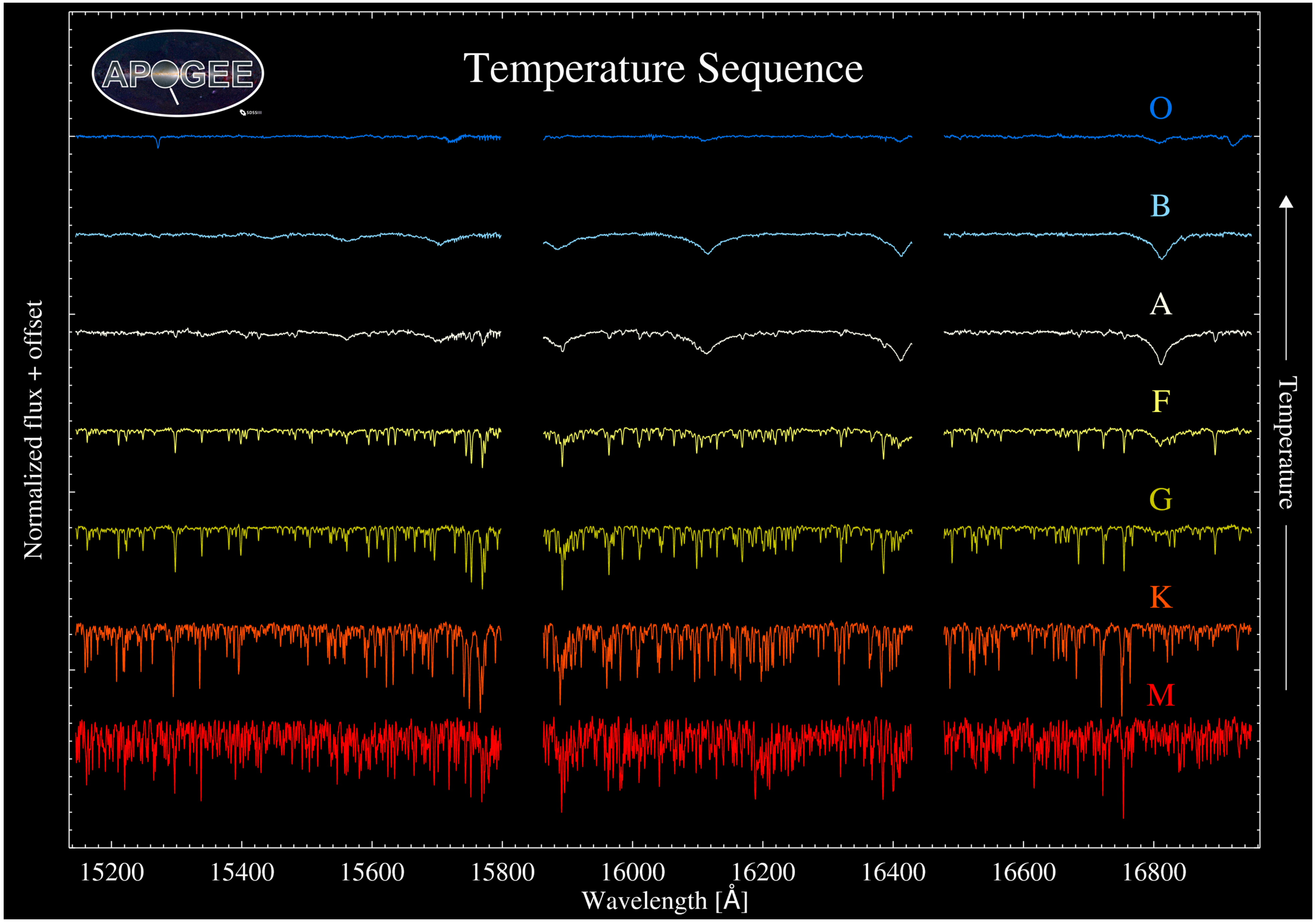
How to interpret the output of a dimensionality reduction algorithm?



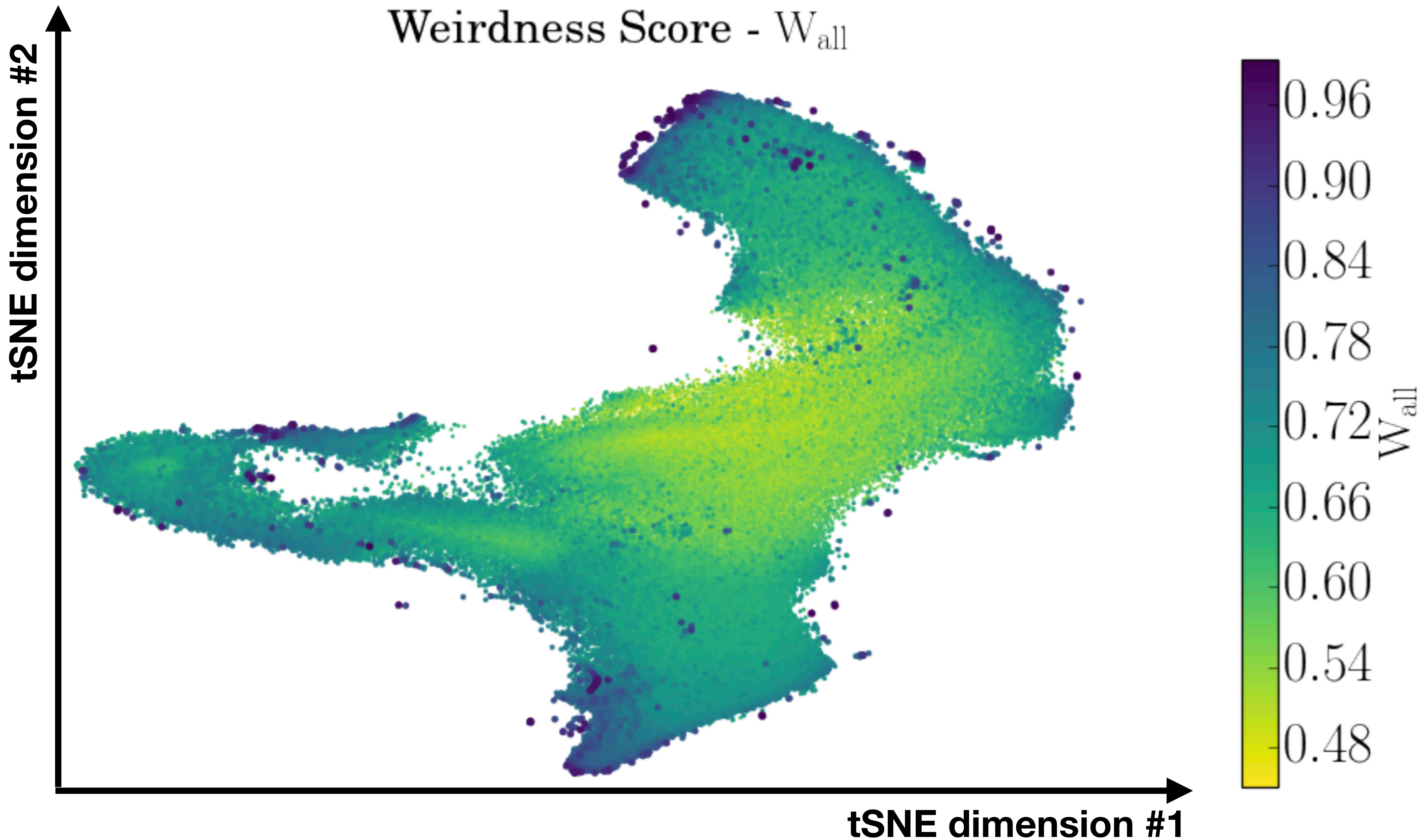

normalized flux
1
0
−1
−2
−3
−4
4000
5000
6000
7000
wavelength (A)
tSNE embedding in two dimensions
Dimension #2
6
4
2
0
-2
-4
−6
−8
−20
0
20
Dimension #1

Example with the APOGEE dataset

- **APOGEE stars:** infrared spectra of ~250K stars.
- Calculate **Random Forest** distance matrix —> Apply tSNE for dimensionality reduction.
- See Reis+17.

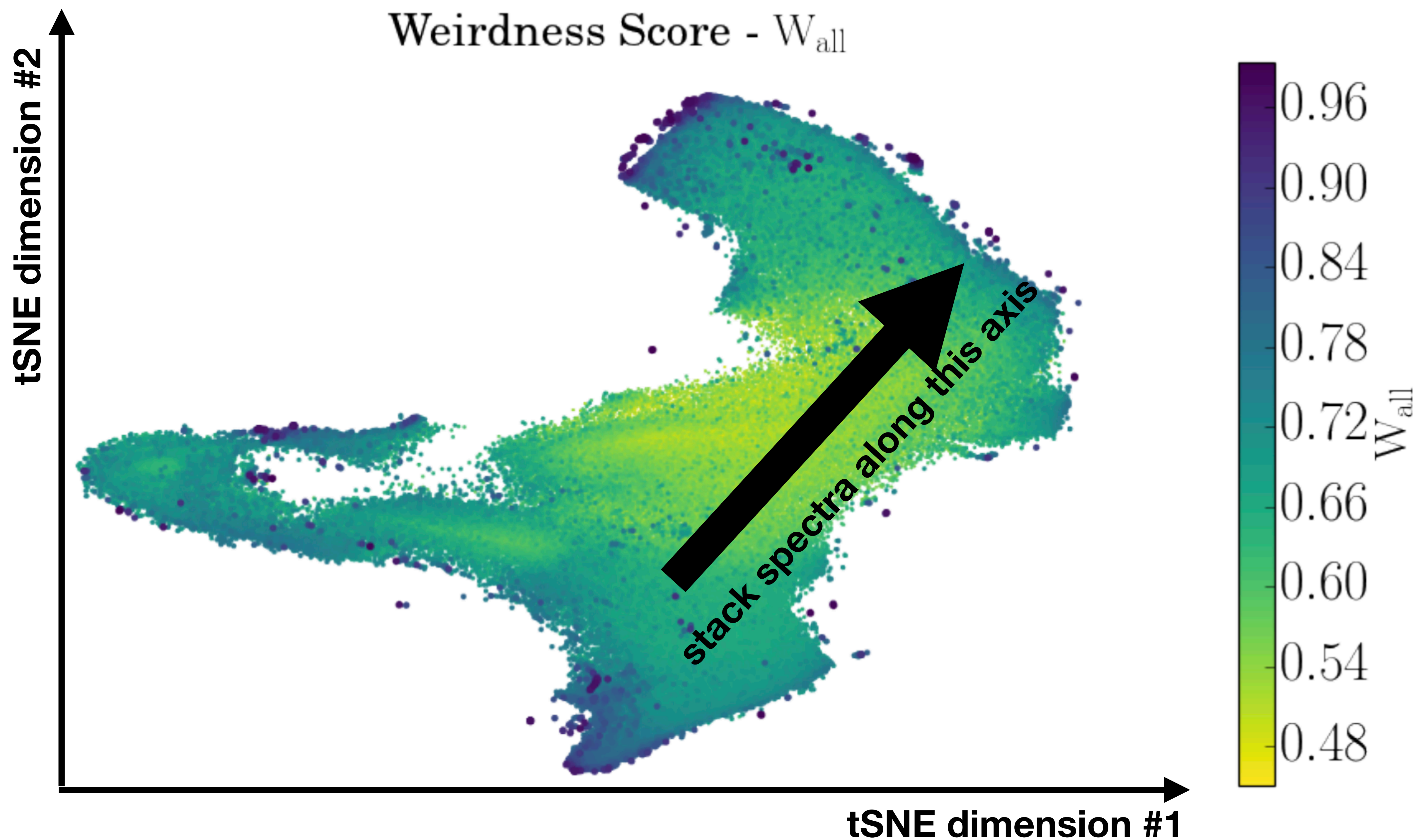


Example with the APOGEE dataset



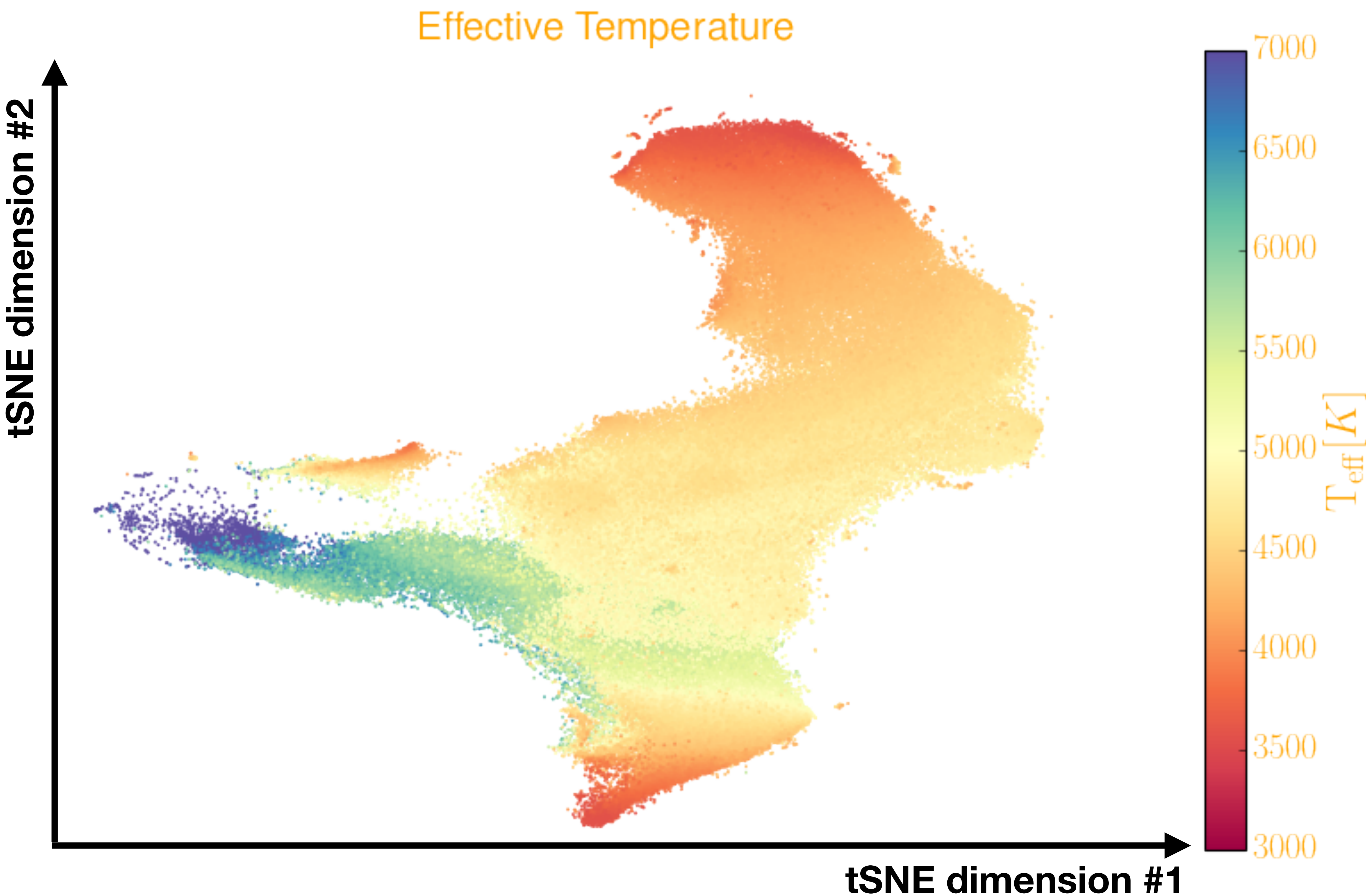
Example with the APOGEE dataset

1. Stack observations along different axises.



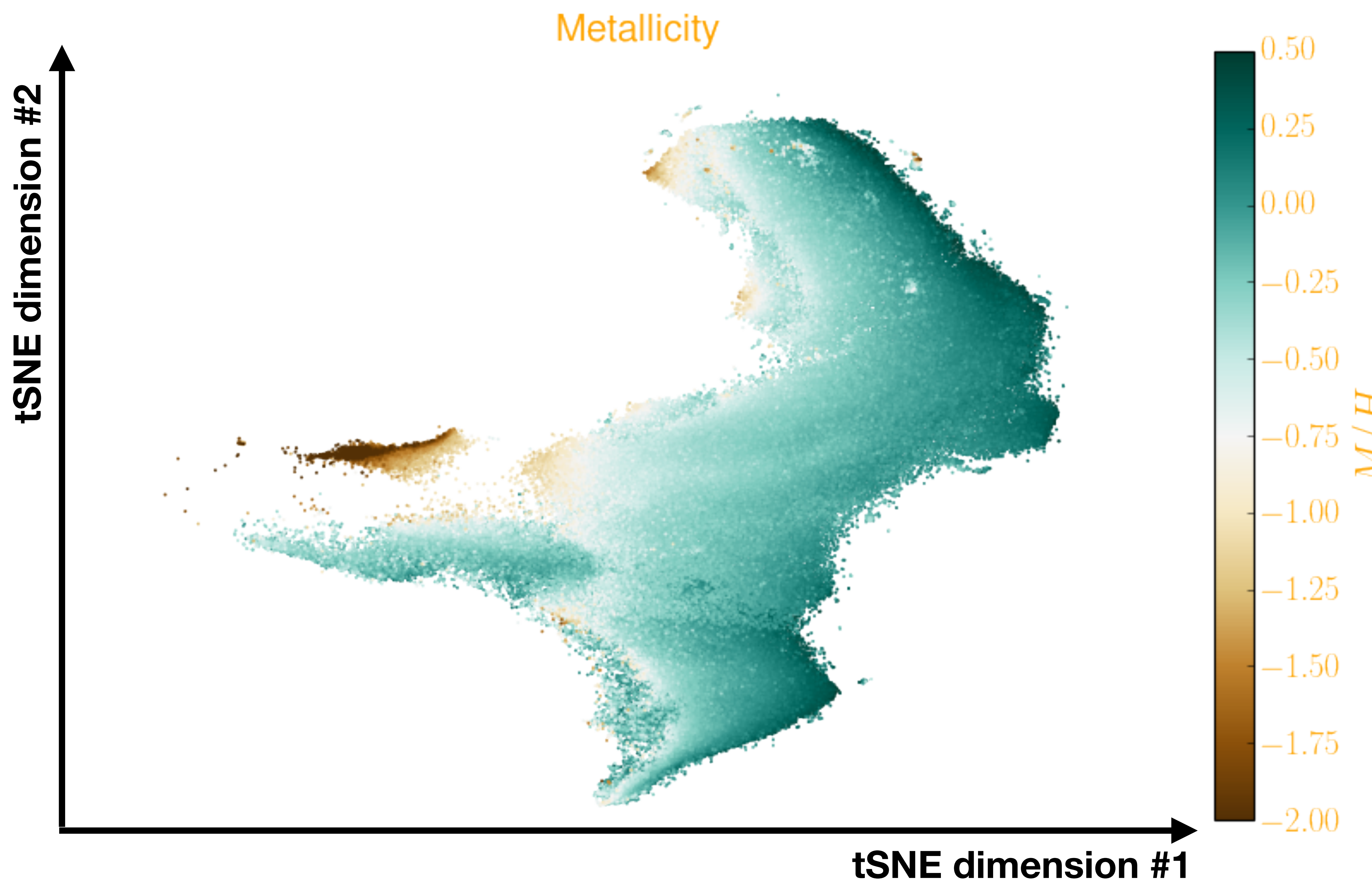
Example with the APOGEE dataset

2. Color points according to tabulated parameters (e.g., from the SDSS)



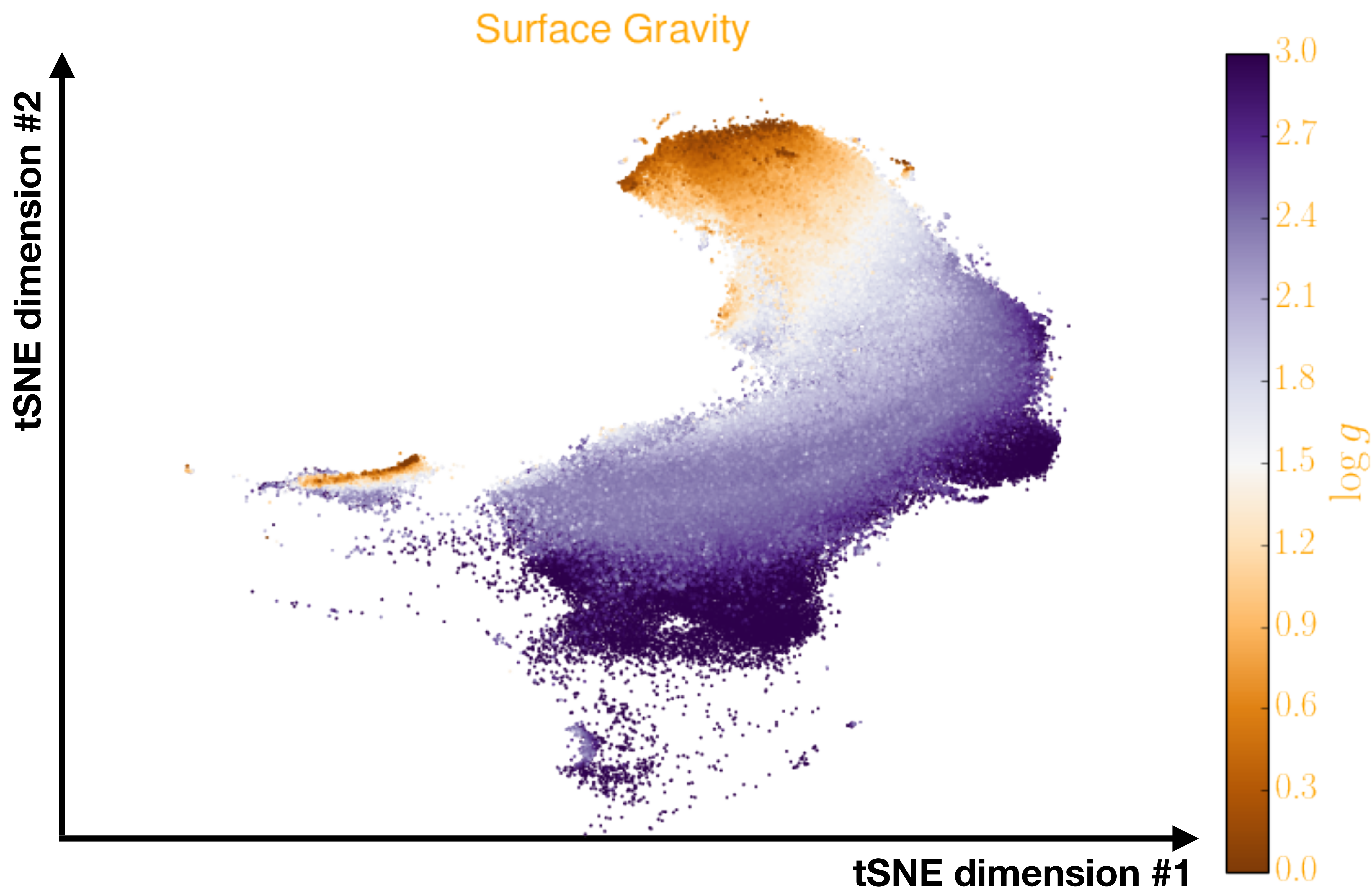
Example with the APOGEE dataset

2. Color points according to tabulated parameters (e.g., from the SDSS)



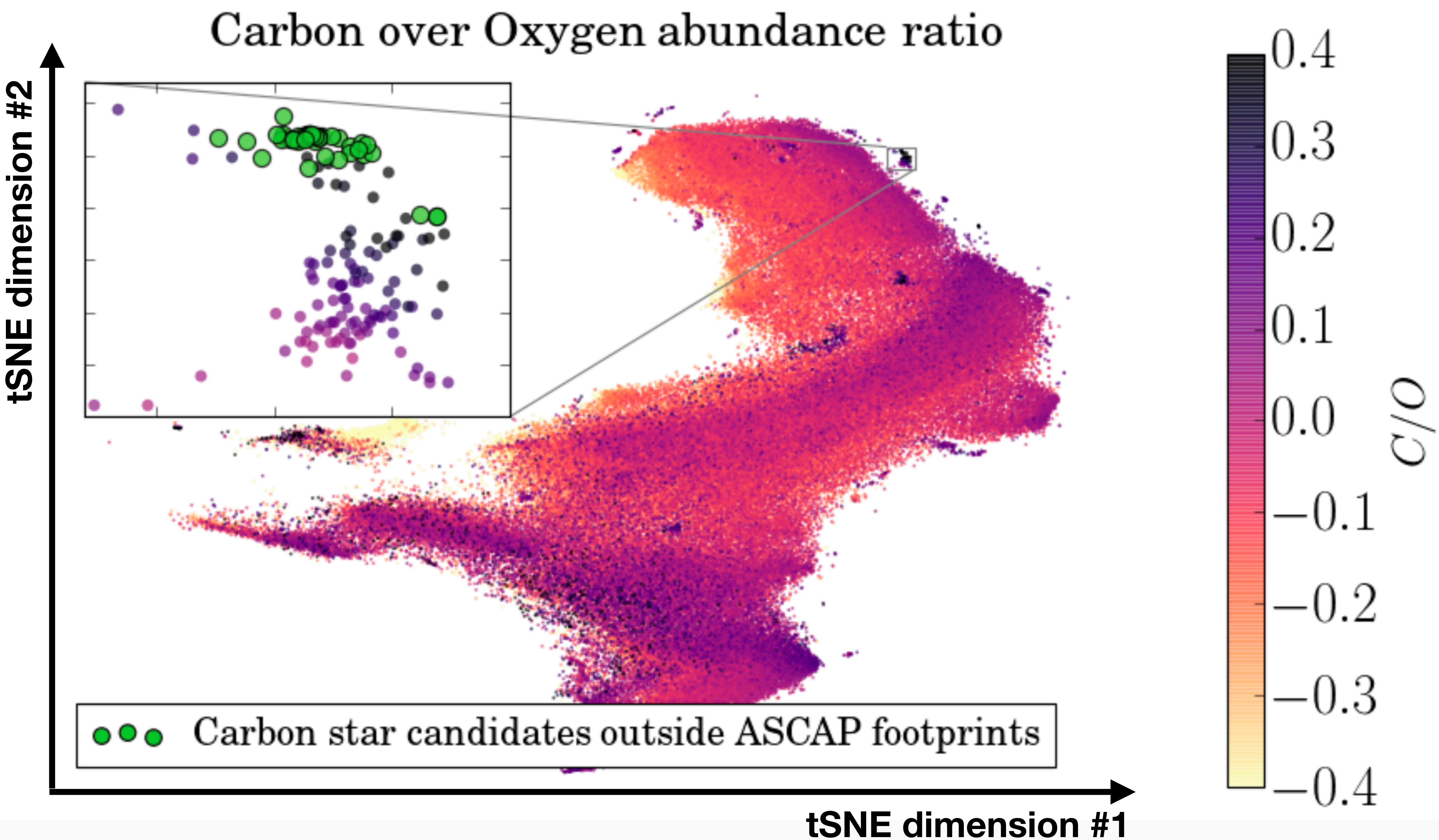
Example with the APOGEE dataset

2. Color points according to tabulated parameters (e.g., from the SDSS)



Example with the APOGEE dataset

2. Color points according to tabulated parameters (e.g., from the SDSS)



Questions?