Introduction to Machine Learning

Dalya Baron (Tel Aviv University)

April 2019

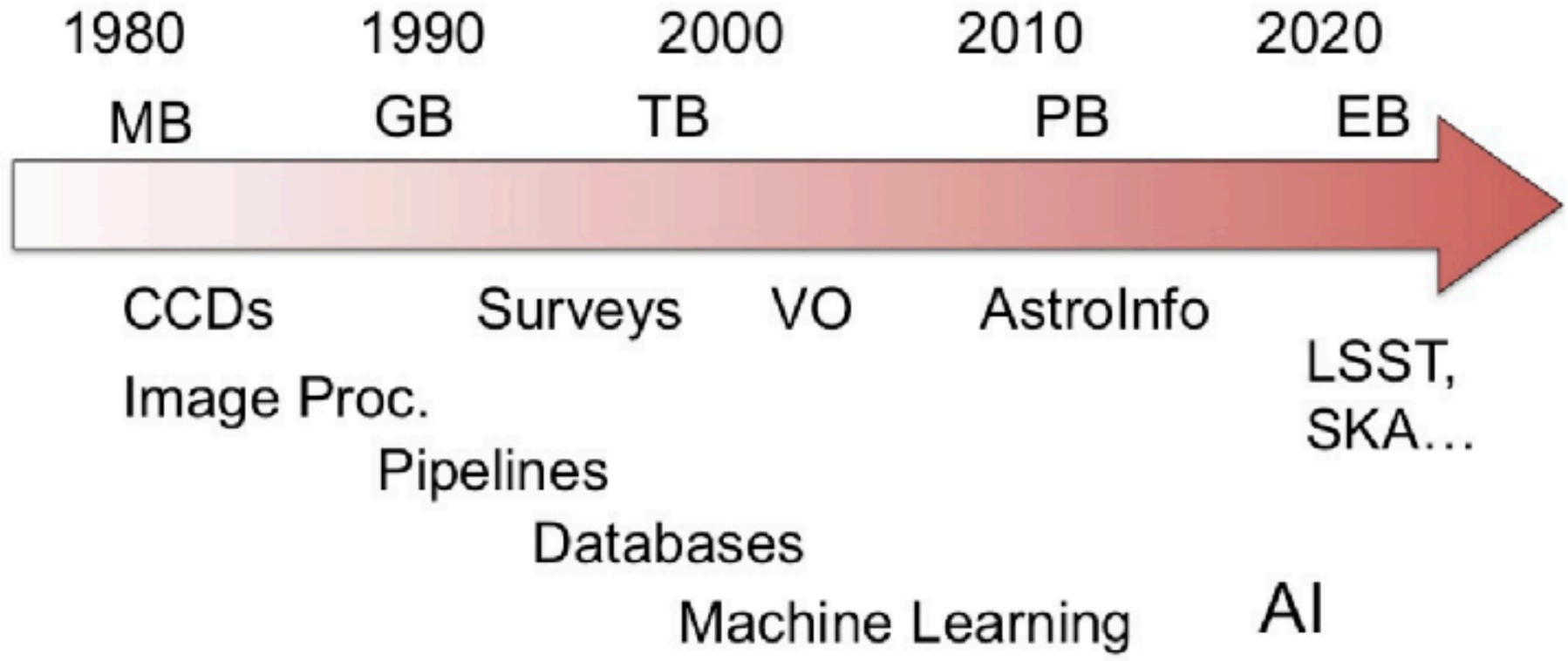
https://arxiv.org/abs/1904.07248

Data Revolution in Astronomy

What is fundamentally different now?

1. **The volume and rate of information grows exponentially**:

- **Sky surveys now generate ~1PB of data + derived data products**.
- **Telescopes are now becoming front-ends to data factories**.
- **Humans can no longer see all the data**.



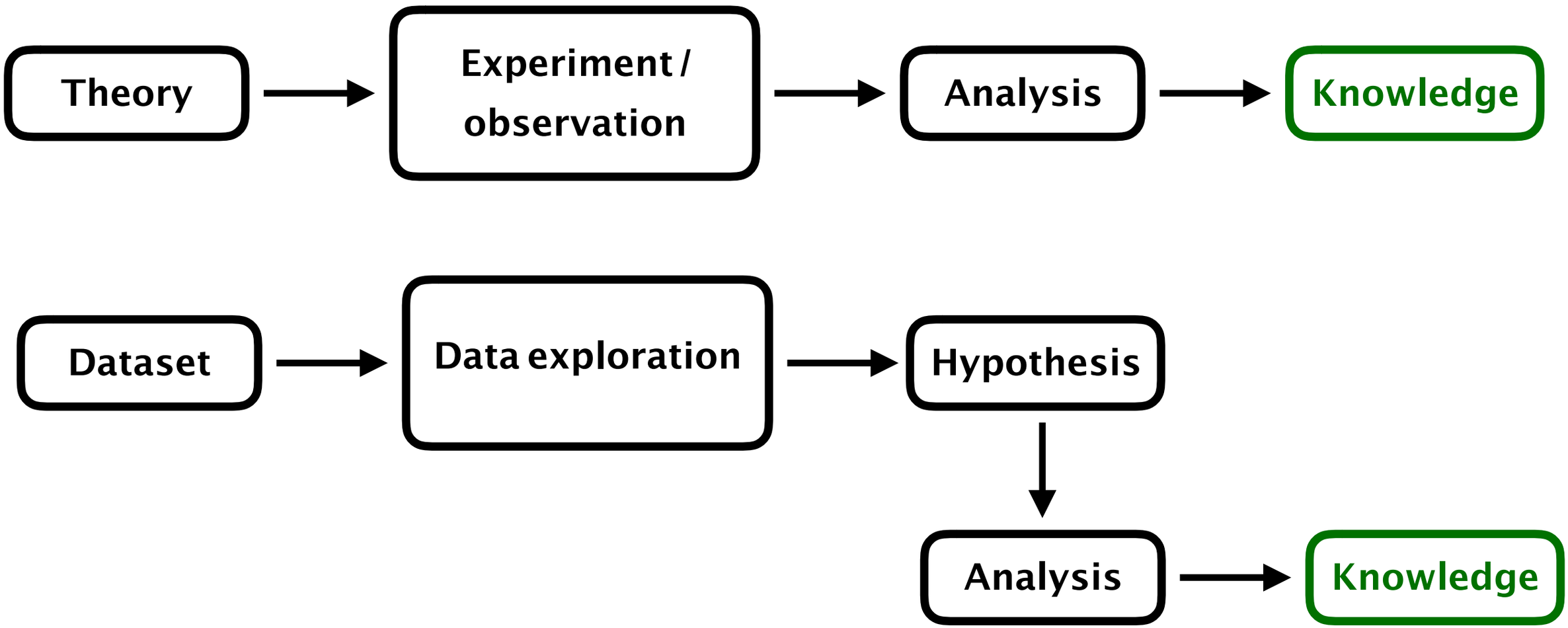
taken from G. Djorgovski (Caltech)

Data Revolution in Astronomy

What is fundamentally different now?

2. **A great increase in data information content**:

- **We combine hypothesis-driven and data-driven science.**
- **This gives rise to a new scientific methodology.**

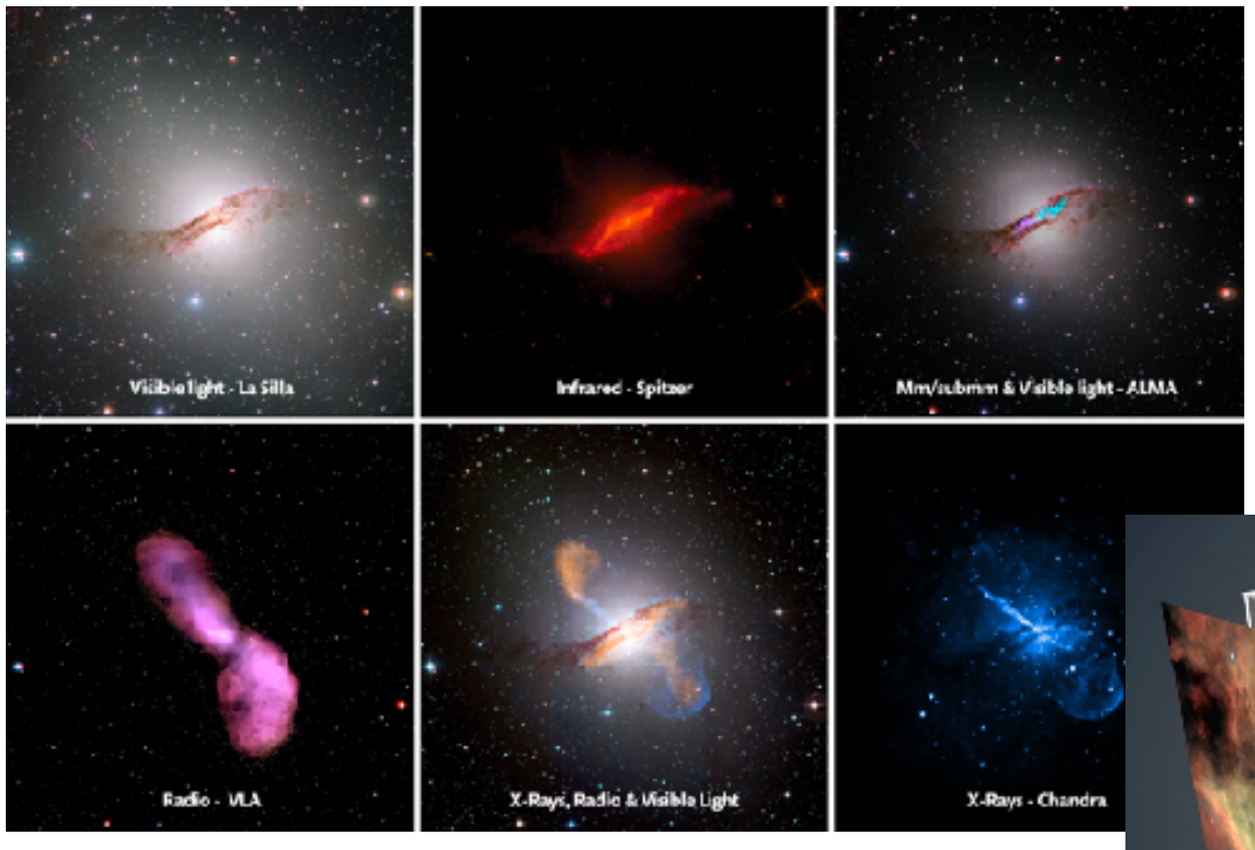


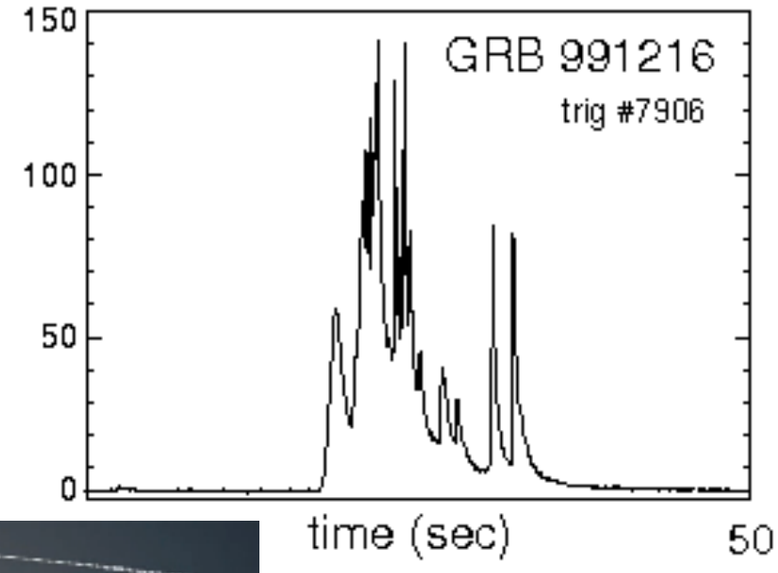
Data Revolution in Astronomy

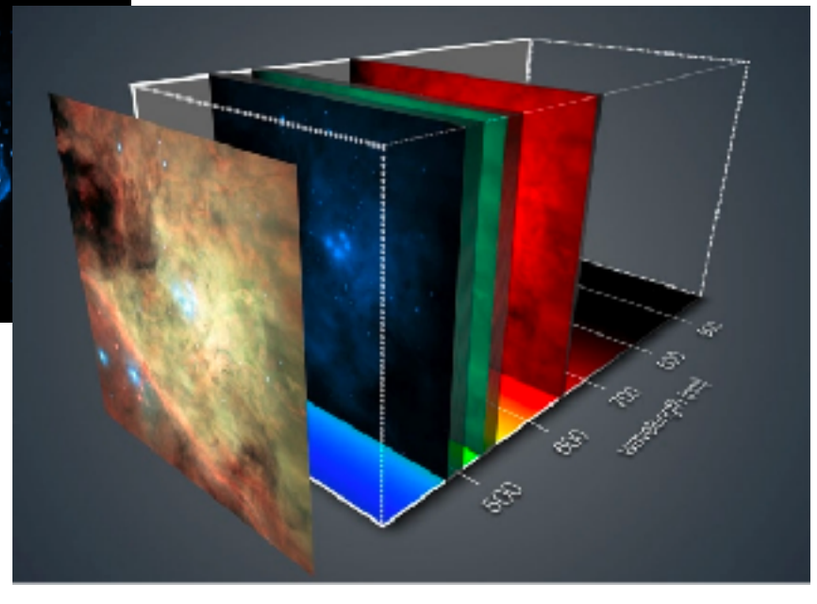
What is fundamentally different now?

3. A great increase in data dimensionality and complexity:

- Data is heterogeneous high-dimensional.
- Patterns and correlations in the data that cannot be visualized in 3D.







Machine Learning as a tool in the astronomers toolkit

Challenges in the big-data era: classification and sorting, clustering, correlation searches, visualization, anomaly detection.

Machine Learning as a tool in the astronomers toolkit

1. **Data gathering and processing**: automated object discovery and classification, automated cross-matching between different observables, and data-quality control.
2. **Data mining and knowledge discovery**: classification and regression, clustering and pattern recognition, outlier detection, dimensionality reduction and visualization.

Two flavors of machine learning

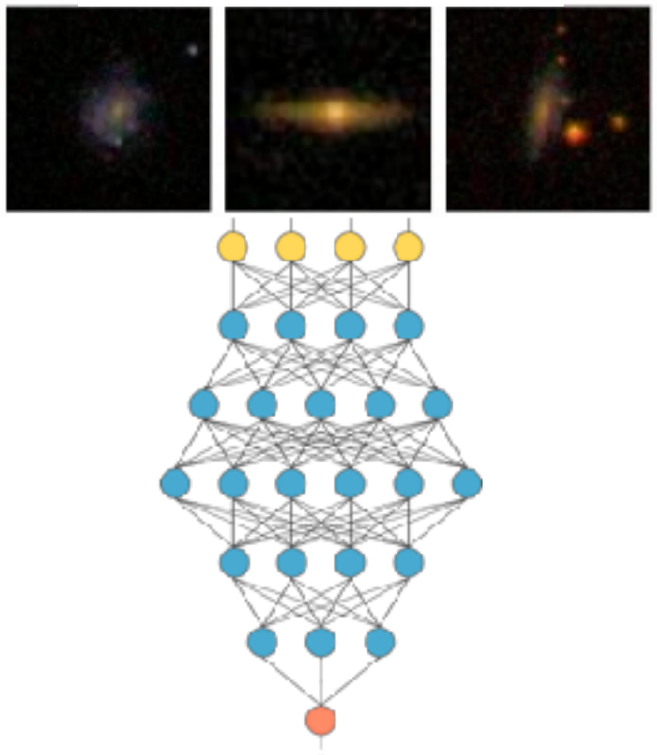
Supervised Learning: given a list of objects with measured properties and a target variable, train an algorithm to predict the target variable of previously-unseen examples.

Common tasks: classification and regression.

In other words: "find me more of this type!"

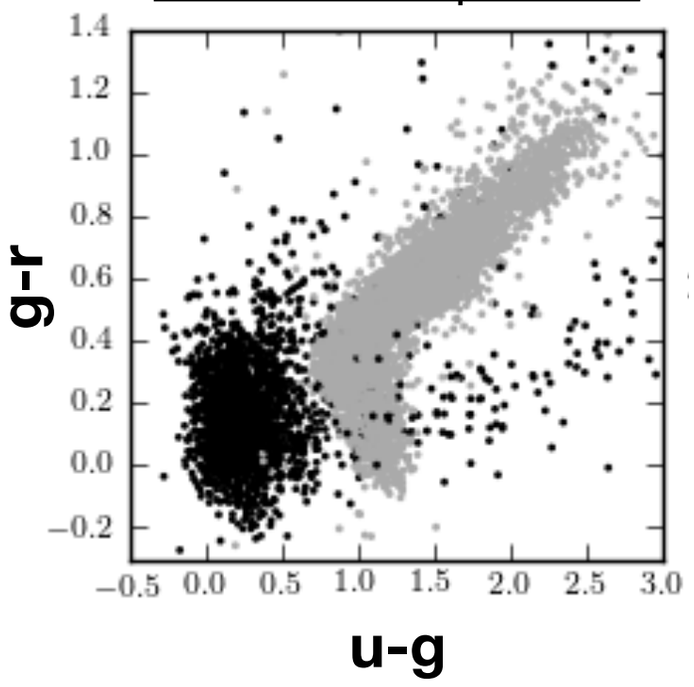
Hard to find previously-unknown objects/phenomena!

input: galaxy images

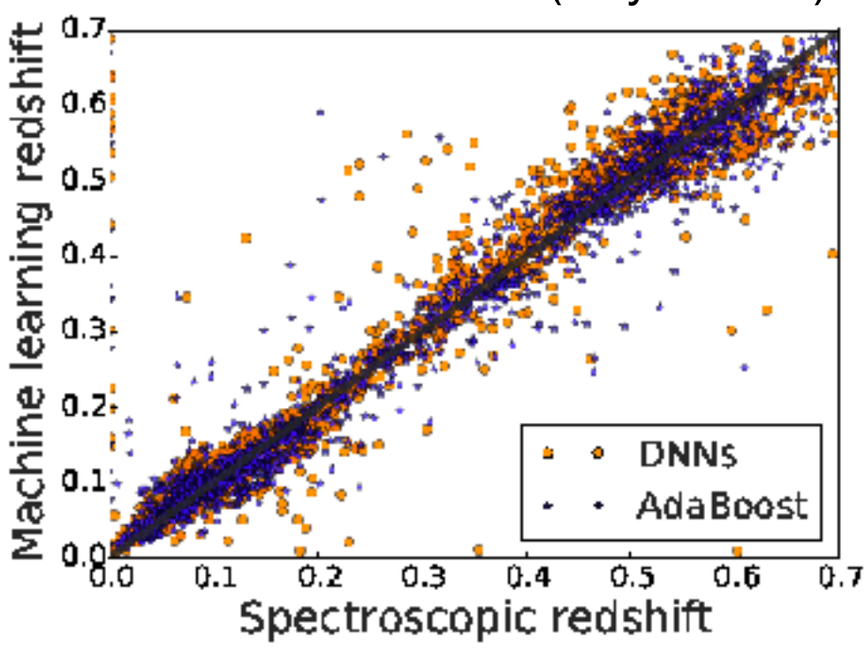


output: galaxy class

QSO/stars separation



Redshift estimation (Hoyle 2016)



Two flavors of machine learning

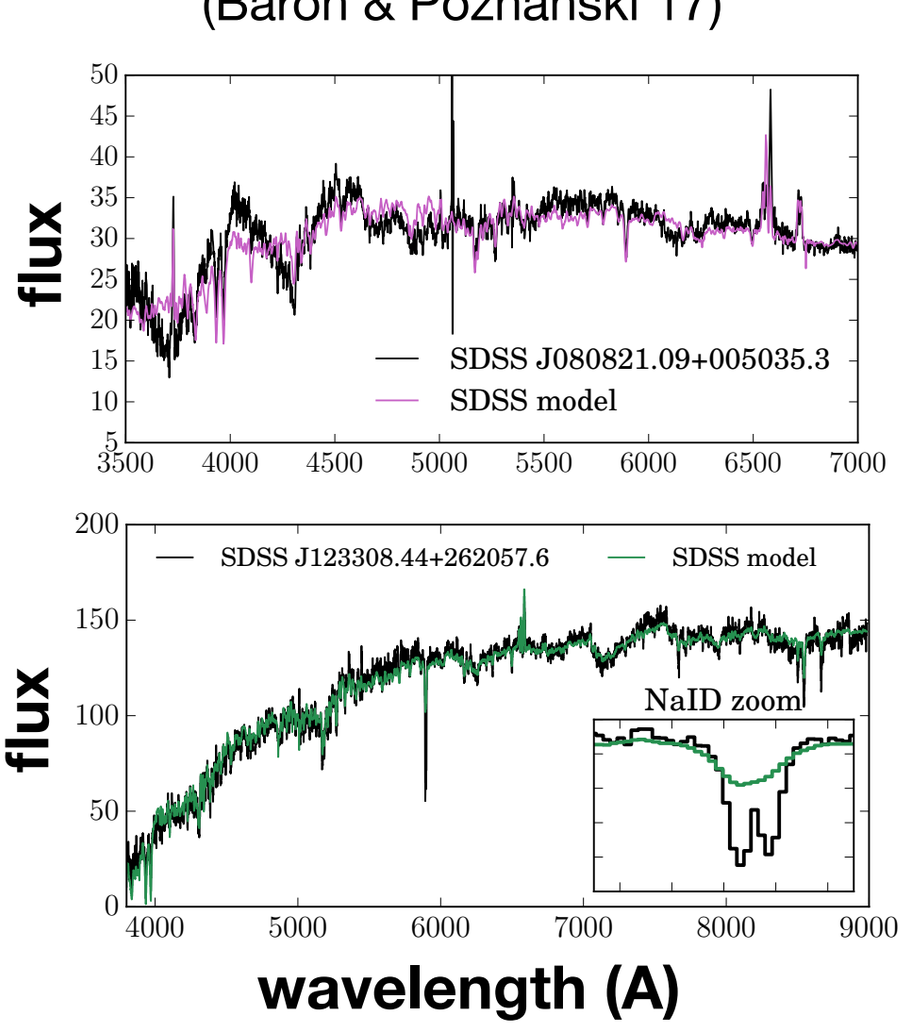
Unsupervised Learning: given a list of objects with measured properties (but no target variable!), find patterns in the dataset.

Common tasks: clustering, dimensionality reduction, outlier detection.

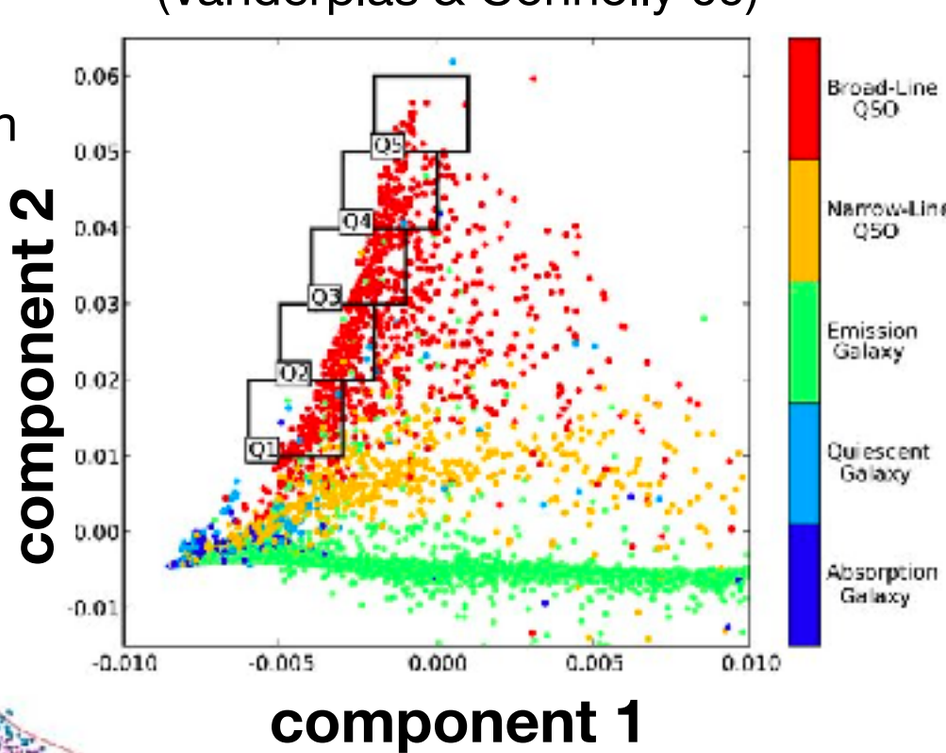
In other words: "find me something new!"

No clear figure of merit!

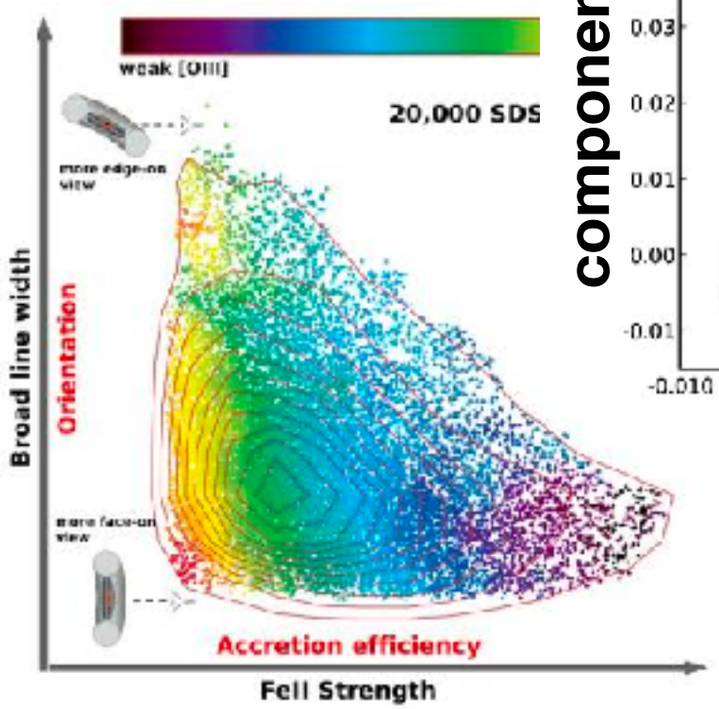
Outlier detection on SDSS spectra
(Baron & Poznanski 17)



Dimensionality reduction
of SDSS spectra
(Vanderplas & Connolly 09)



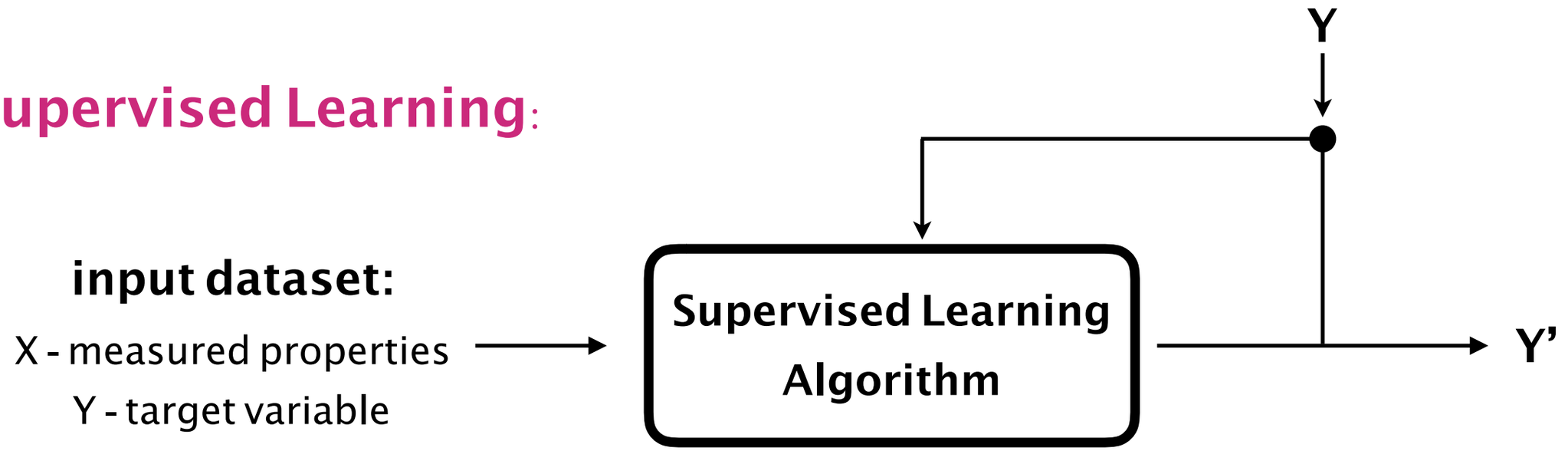
Dimensionality reduction
of QSO features
(Shen & Ho 14)



Two flavors of machine learning

Generally, in Machine Learning the model is built from the data*

Supervised Learning:



Unsupervised Learning:

Supervised Learning

Two types of tasks:

In **classification**, the target variable is discrete, and we call them labels. For example: star/QSO, elliptical/spiral, supernova/variable star.

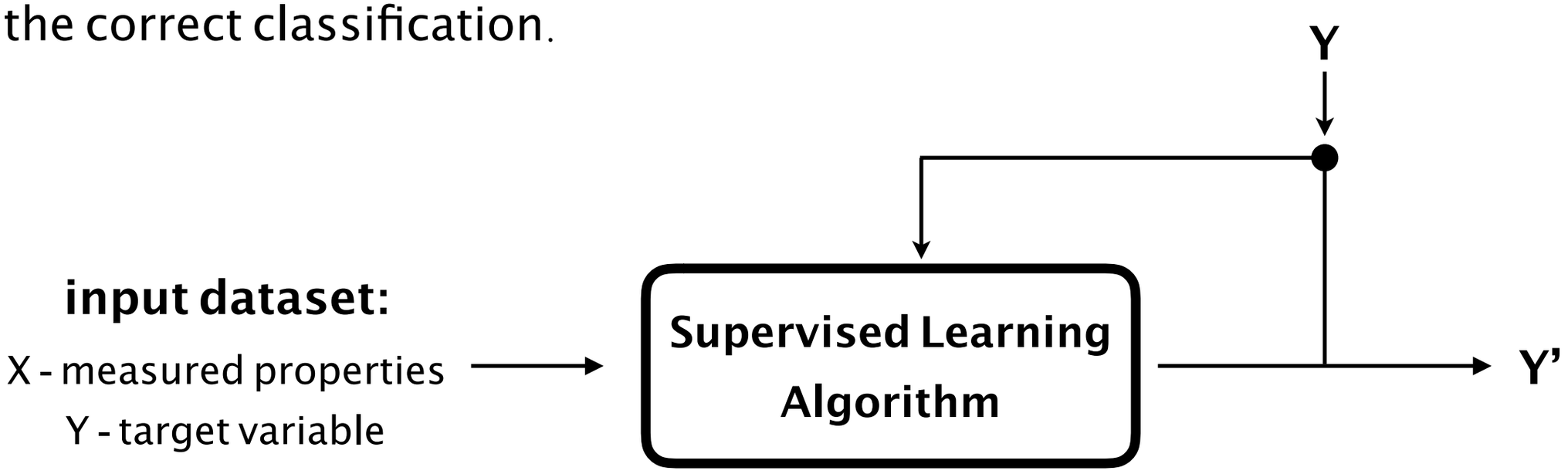
In **regression**, the target variable is continuous.

For example: photometric redshift, stellar effective temperature, bulge mass, metallicity.

Figure of merit:

In **regression** the **MSE** is typically used to find an optimal solution: $MSE = (Y - Y')^2$.

For **classification** we define the **accuracy** to be the number of times the algorithm made the correct classification.



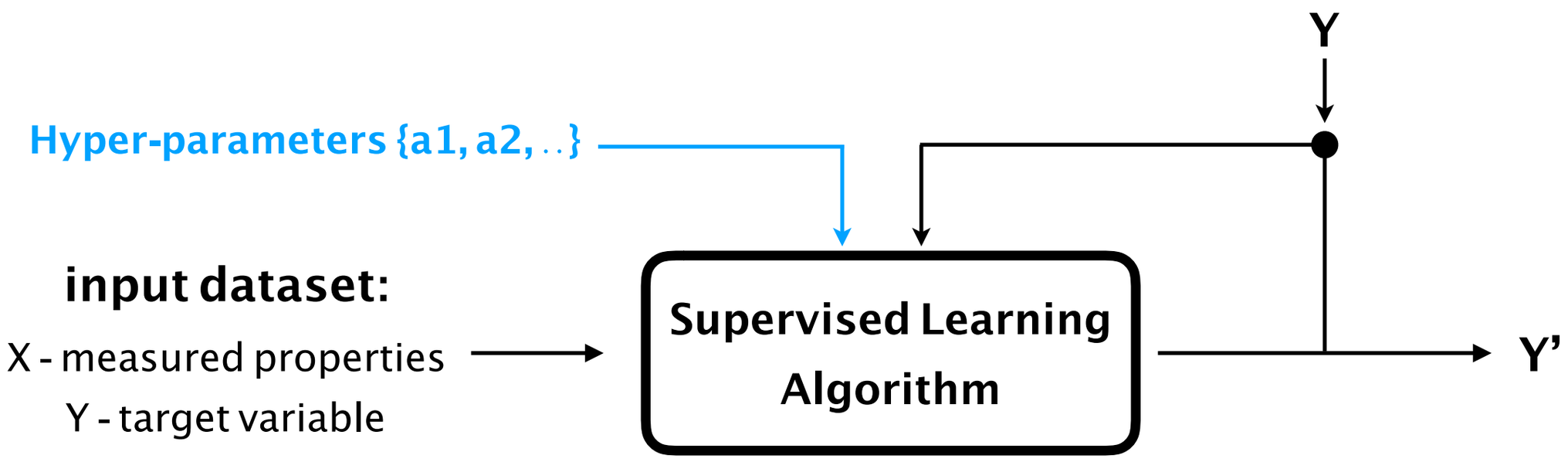
Supervised Learning

Phases in a supervised learning:

Training stage - the algorithm is building a model from the observed dataset (X), given predefined hyper-parameters, while trying to optimize the figure of merit.

Validation stage - the algorithm's hyper-parameters are optimized using the figure of merit.

Prediction stage - the algorithm is trained and the best hyper-parameters were found. The Trained model is used to predict the class on previously unseen datasets.



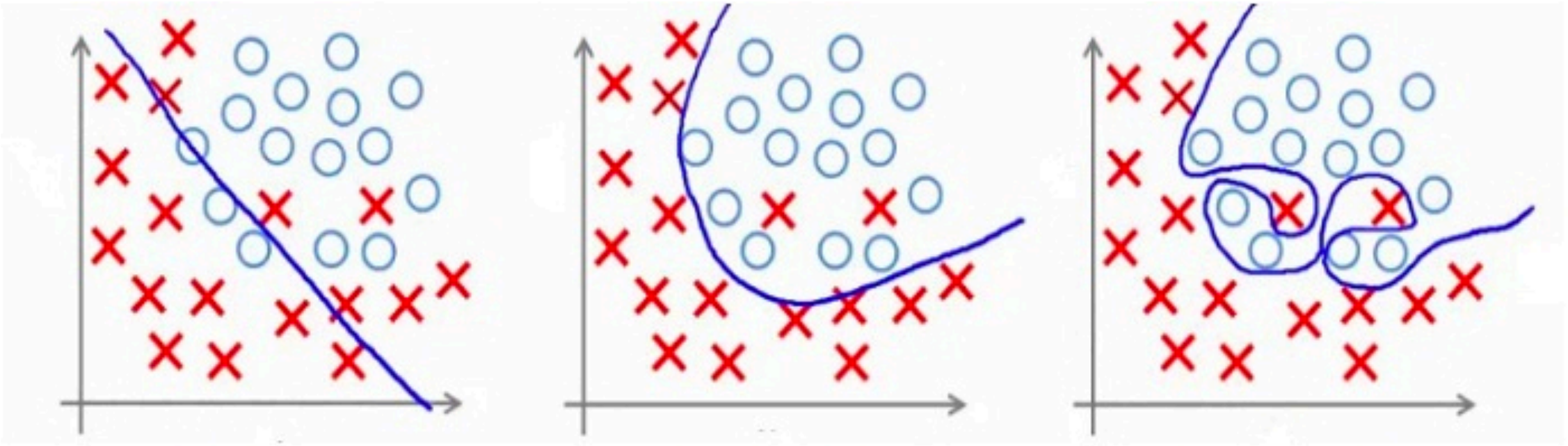
Supervised Learning

We divide our dataset into three sets:

Training set - the algorithm is trained on this dataset.

Validation set - the hyper-parameters are optimized on this dataset.

Test set - the final accuracy/MSE of the trained algorithm is computed on this dataset.



Under-fitting

Appropriate-fitting

Over-fitting

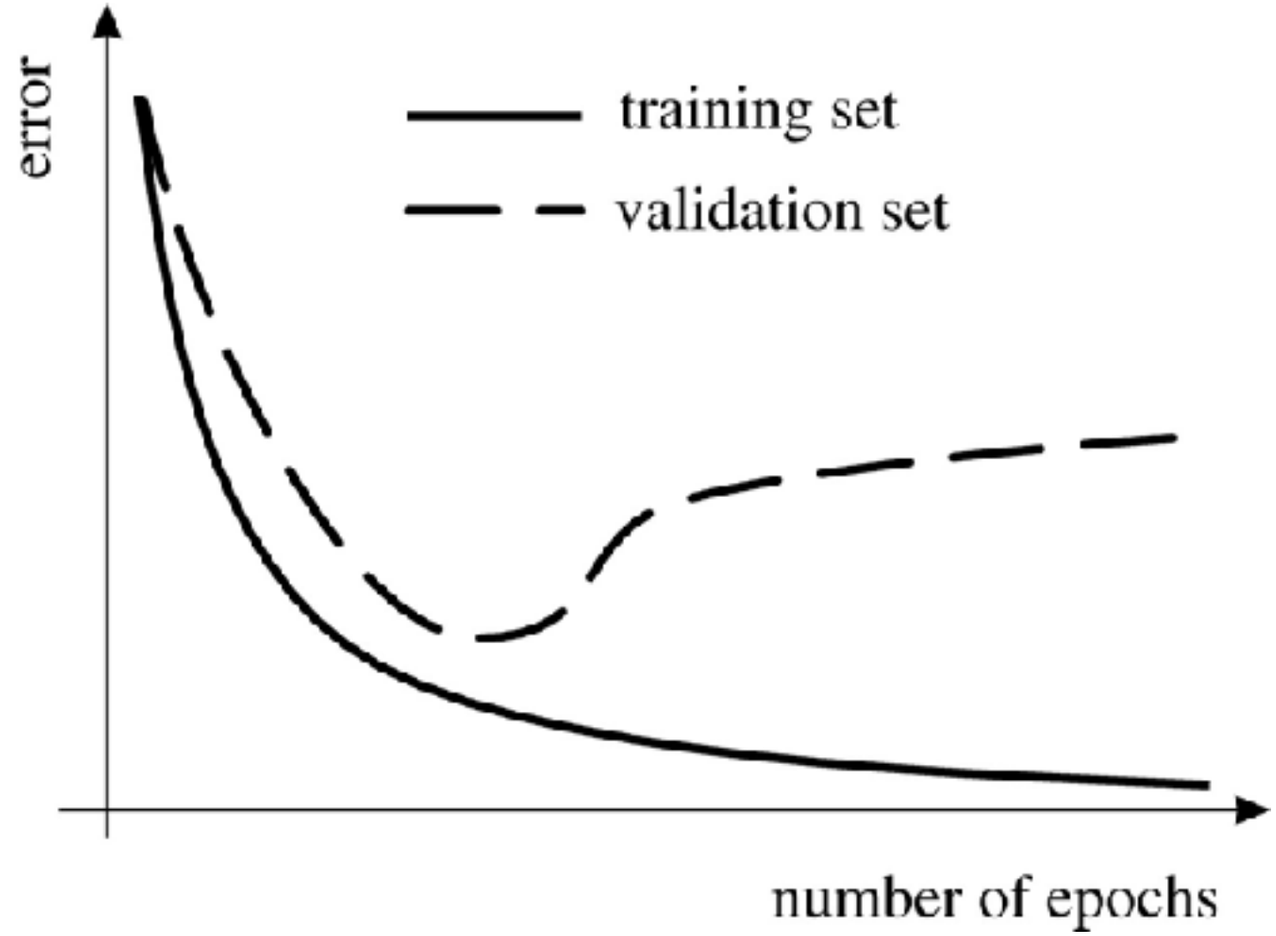
Supervised Learning

We divide our dataset into three sets:

Training set - the algorithm is trained on this dataset.

Validation set - the hyper-parameters are optimized on this dataset.

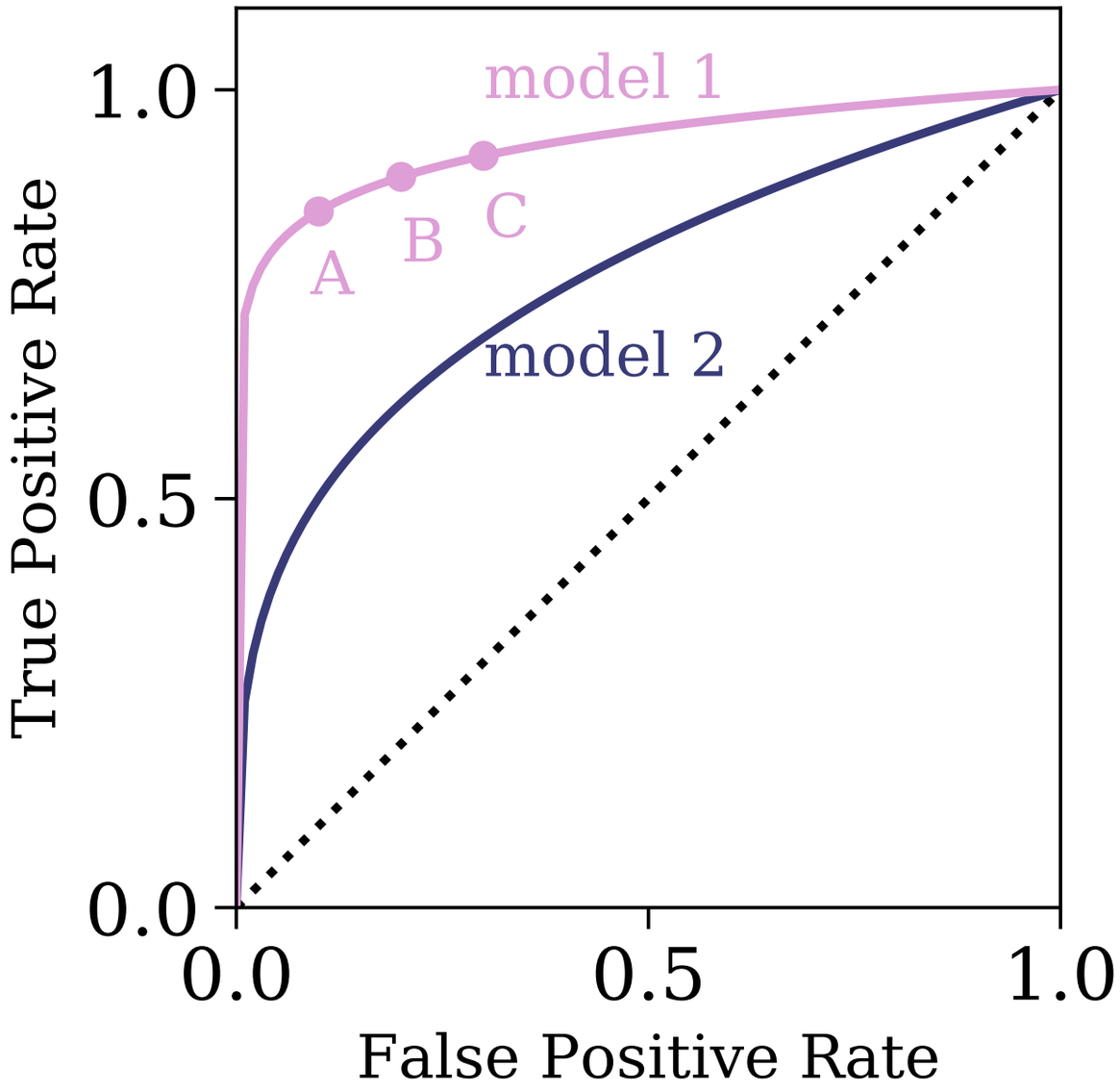
Test set - the final accuracy/MSE of the trained algorithm is computed on this dataset.



Receiver operating characteristic - ROC curve

True positive rate - the number of objects from the “true” class that were correctly classified as “true” objects, divided by the number of “true” objects in the dataset.

False positive rate - the number of “false” objects that were classified as “true”, divided by the number of “false” objects in the dataset.



The diagram is typically populated by varying the model hyper-parameters. The common choice is the classification threshold.

Supervised Learning - Input dataset

Your classifier will be as good as your input dataset: “bad” dataset -> bad classifier!

Input dataset can be:

1. **Raw data**: spectra, images, light-curves. Most algorithms will fail dramatically on these.
2. **Derived features** from the raw data. Most algorithms cannot handle numerous features, and you should do feature selection/engineering.

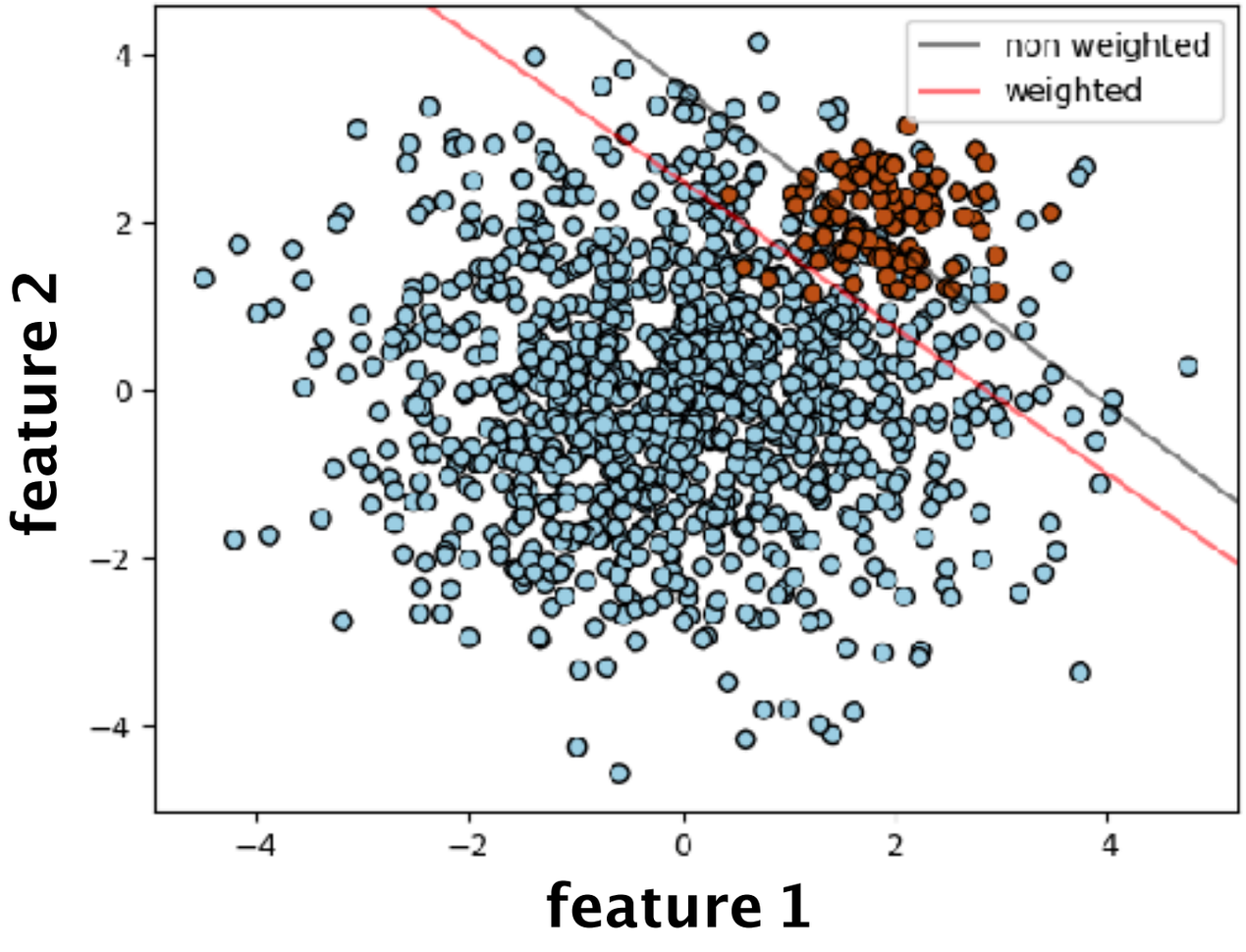
Selecting the “best” features:

1. **Domain knowledge**: we have a reason to believe that some feature is important and others are not.
2. **Dimensionality reduction** (unsupervised learning!) of a large set of features: remove correlated and uninformative features.
3. **Select features** that give the highest accuracy on the validation set.

Features must be normalized!!

Supervised Learning - Imbalanced datasets

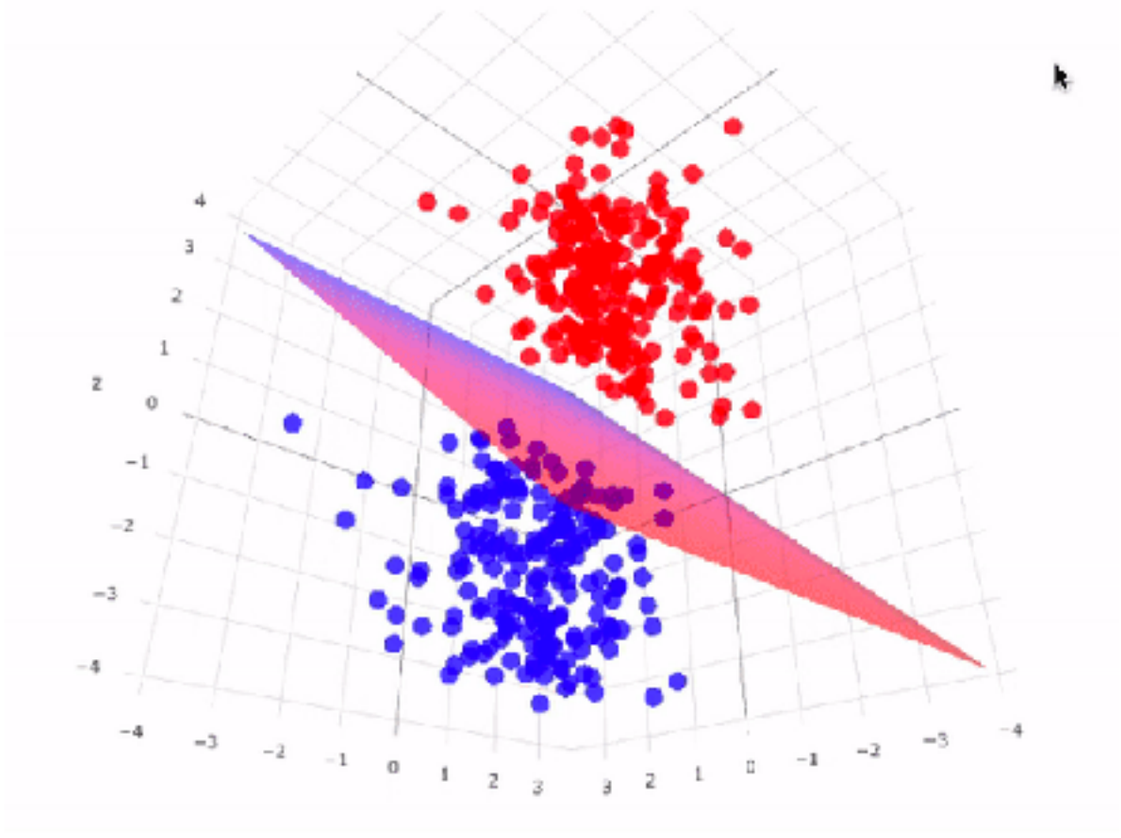
Our datasets are sometimes imbalanced: Supervised Learning algorithms will tend to do well on the most common object in our dataset.
For example, in a star/QSO classification, our training data contains 10,000 stars and 10 QSOs.
Possible solutions: under-sampling the large class, over-sampling the small class, assigning different weights for different classes in our figure of merit.

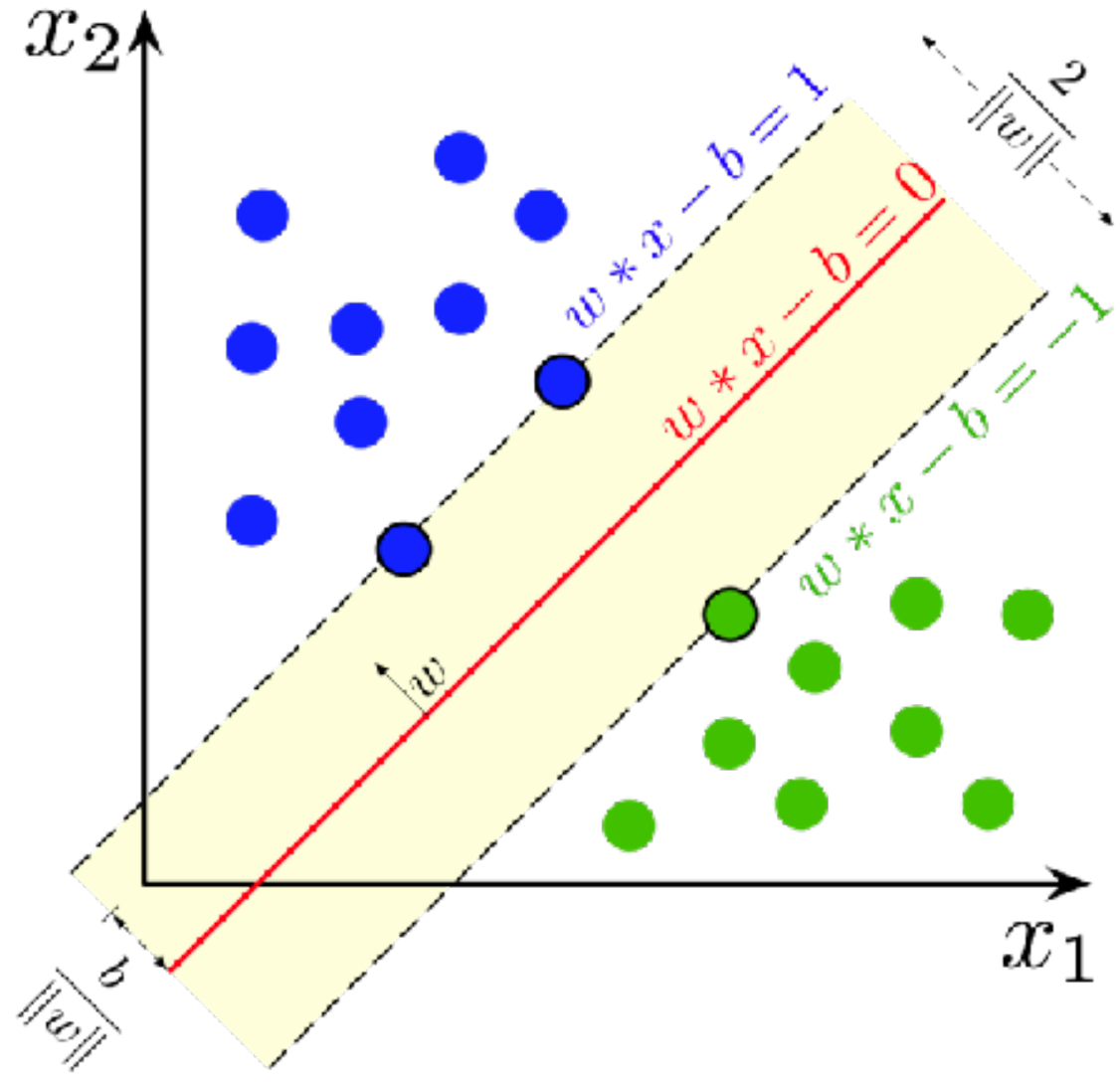


Support Vector Machine (SVM)

Supervised Machine Learning algorithm used for **classification**. The constructed model is a **multidimensional hyper-plane** that best separates the two classes. For example, our dataset has two features and two classes. SVM will find the one-dimensional hyper-plane that provides the best separation of the two classes.

Training stage: given a labeled dataset, find the best multidimensional hyperplane.



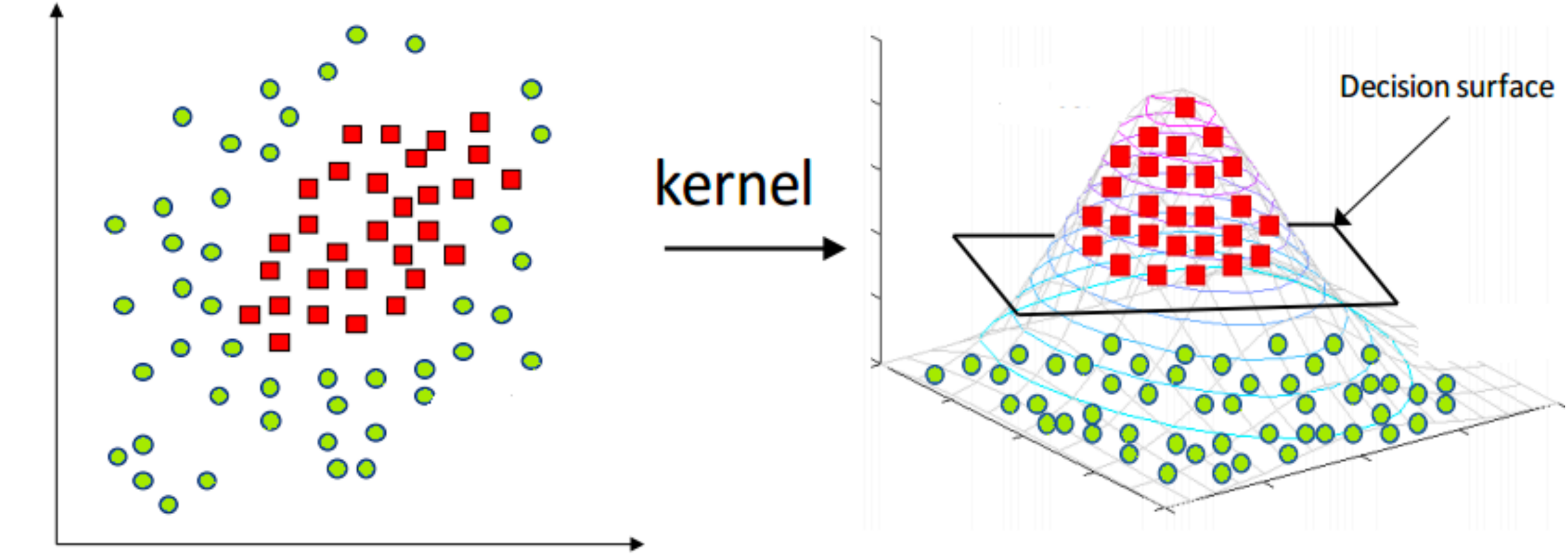


Support Vector Machine (SVM)

For datasets which are not-linearly separable, we apply the **Kernel Trick**.

We map the input dataset onto a higher dimension, using a predefined kernel, where the data is linearly-separable.

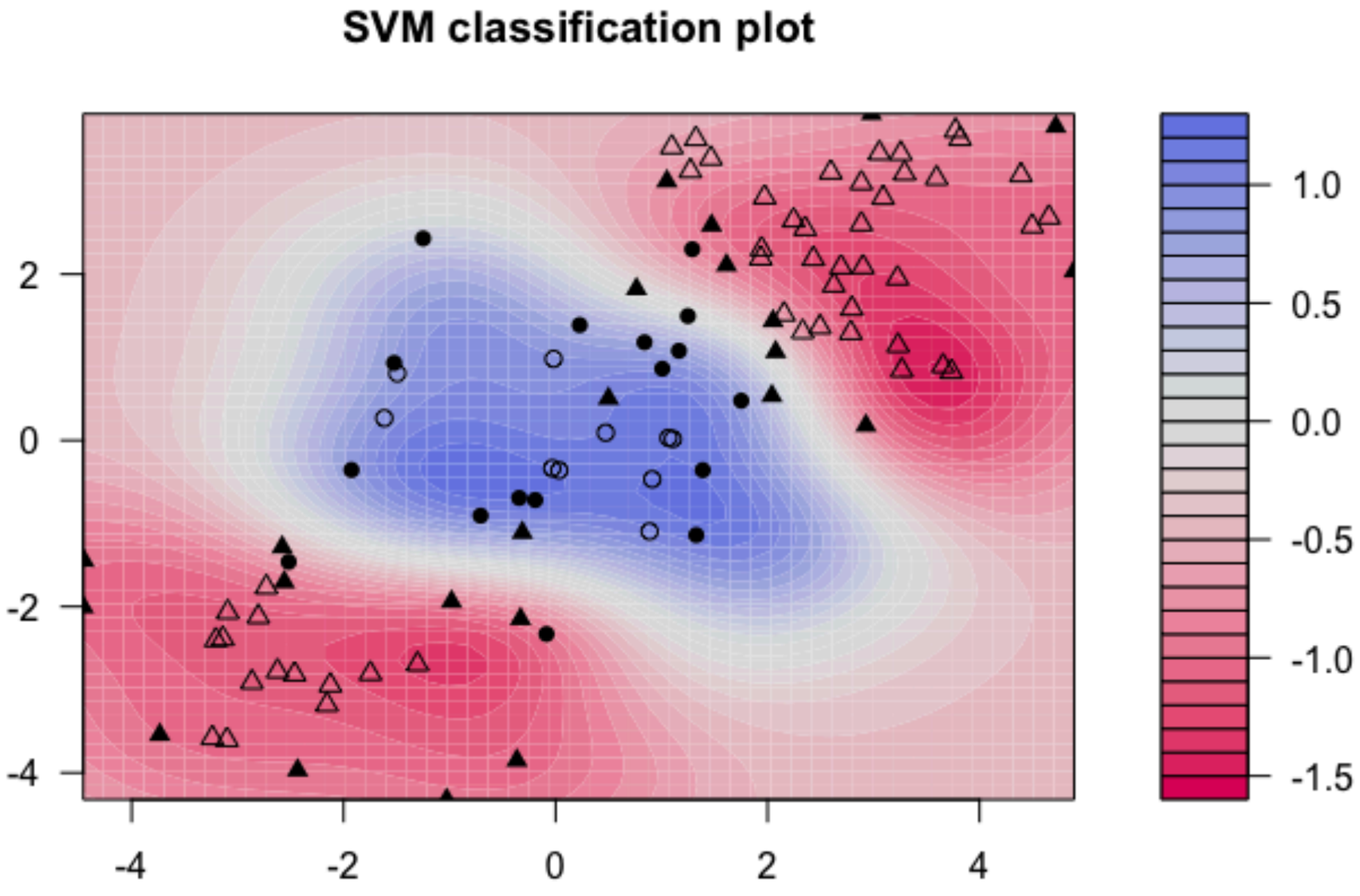
The kernel shape is a **hyper-parameter** of the algorithm!



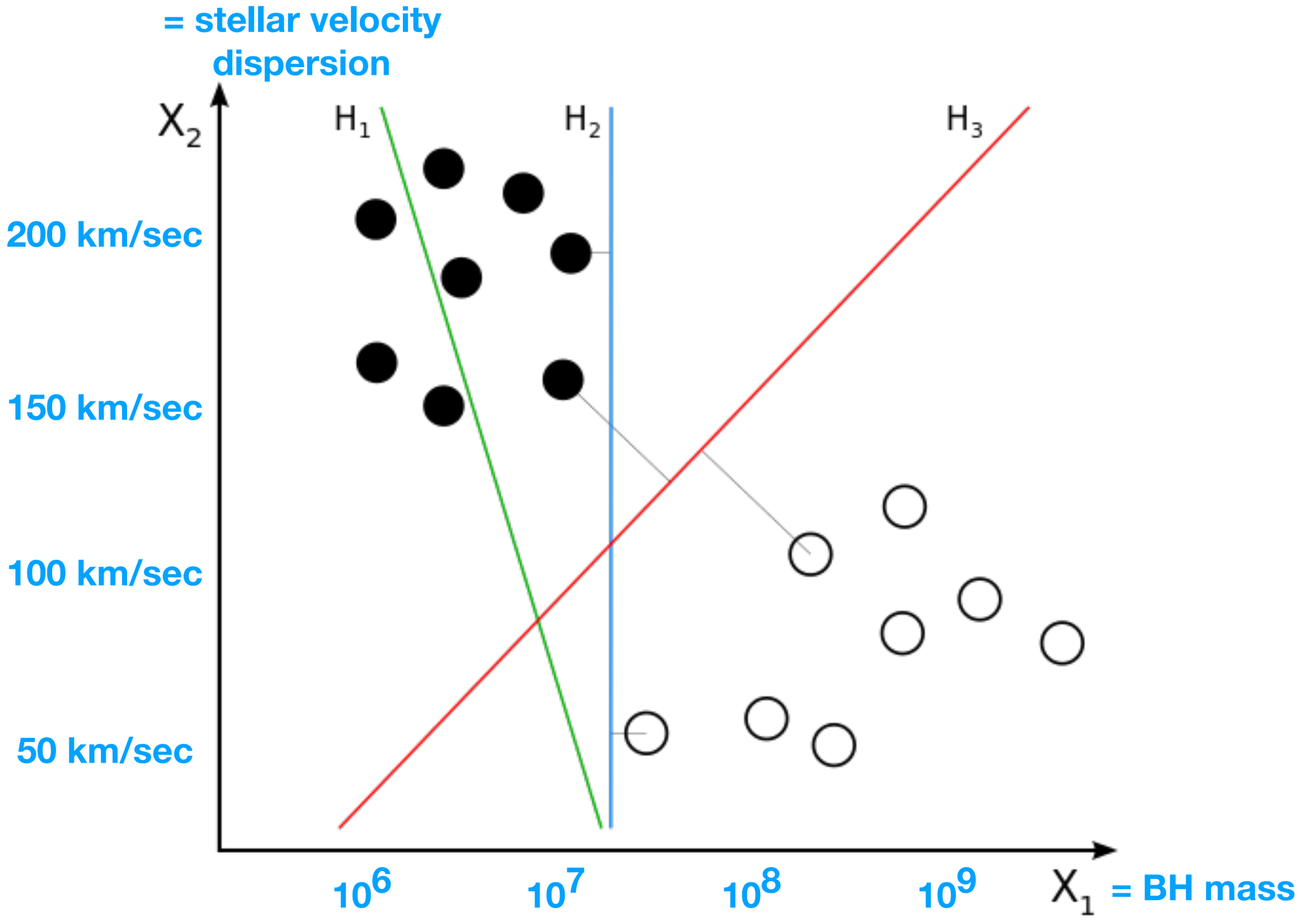
Support Vector Machine (SVM)

Prediction on previously unseen datasets: use the decision boundary to classify the objects according to their features.

The distance of an object from a decision boundary can be used as a classification uncertainty.

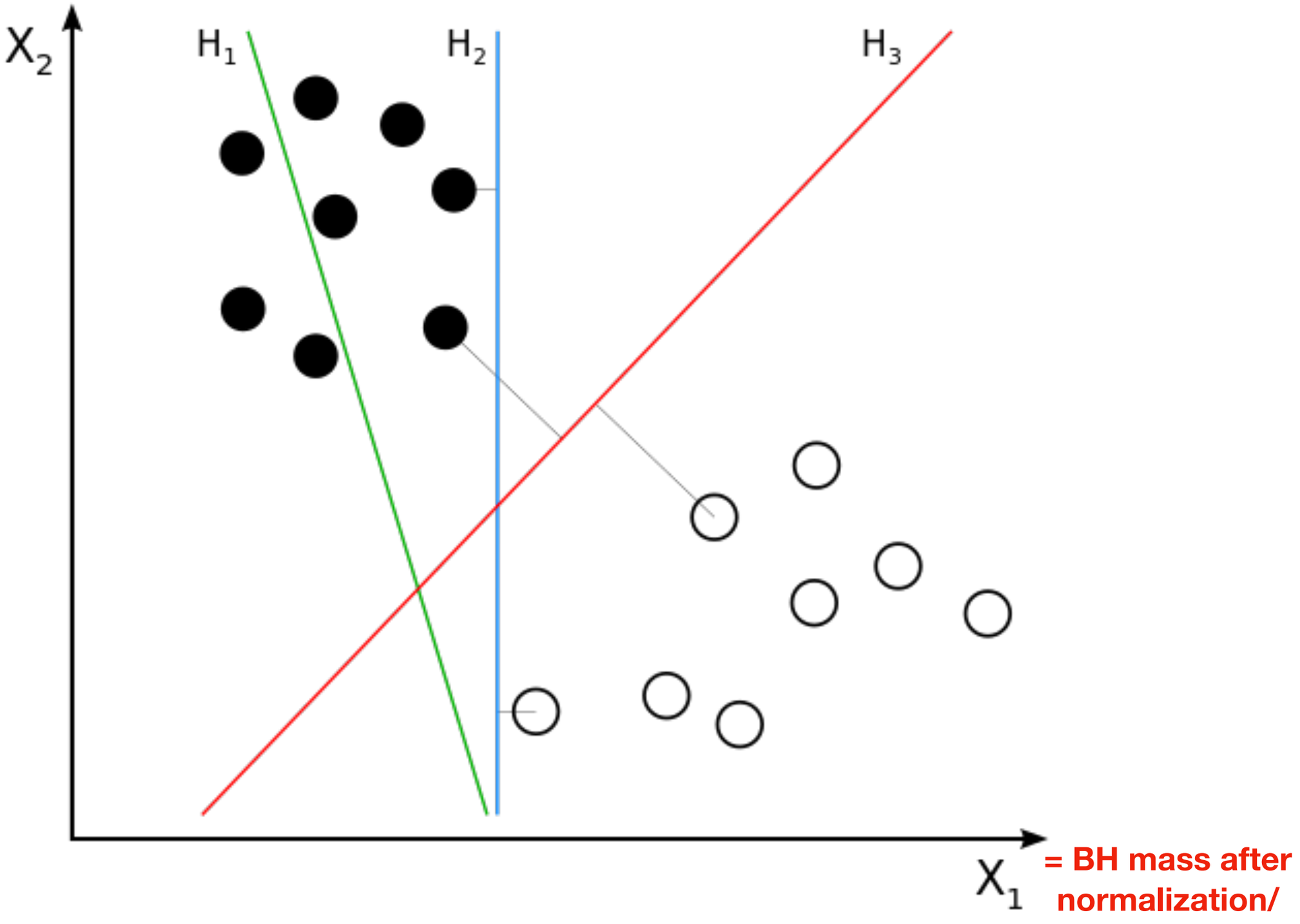


SVM - Importance of feature normalization



SVM - Importance of feature normalization

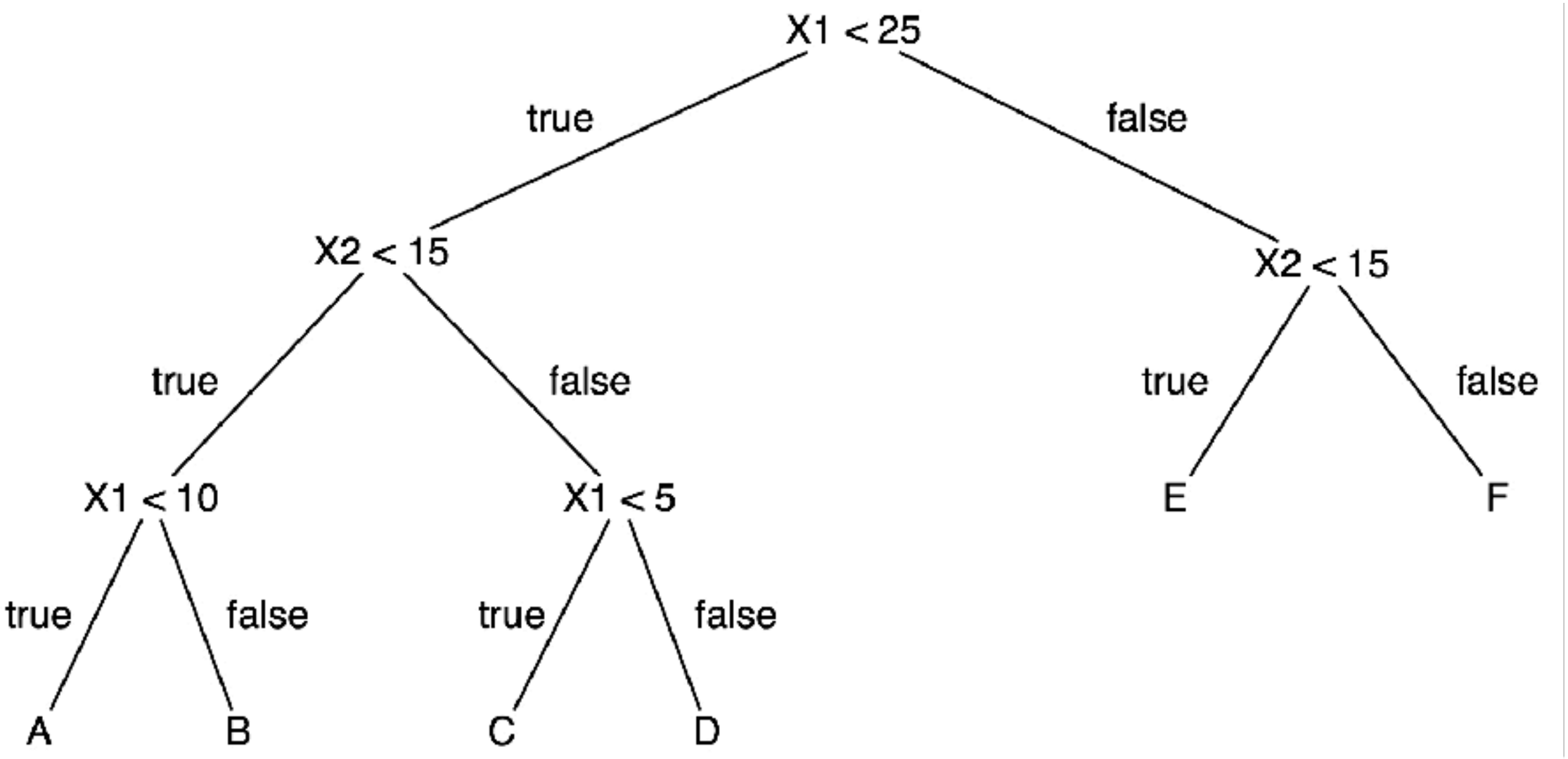
= stellar velocity dispersion after normalization/standardization



= BH mass after normalization/ standardization

SVM - Pros & Cons

Decision Trees



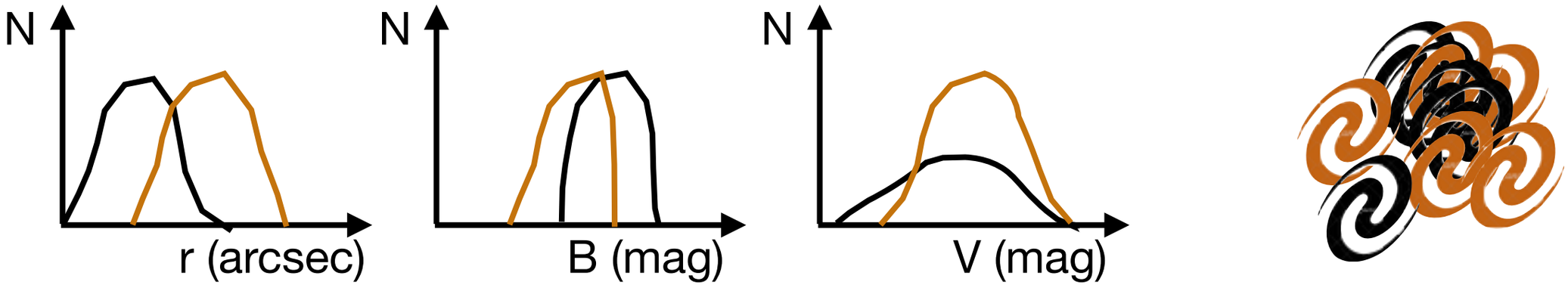
Decision tree: a non-parametric model, constructed during training, which is described by a tree-like graph. It can be used for classification or regression.

Decision Tree Construction

Input training set: a list of objects with measured features and known labels.
Classes: “black” and “brown” galaxies.
Measured features: r (arcsec), B (mag), V(mag).

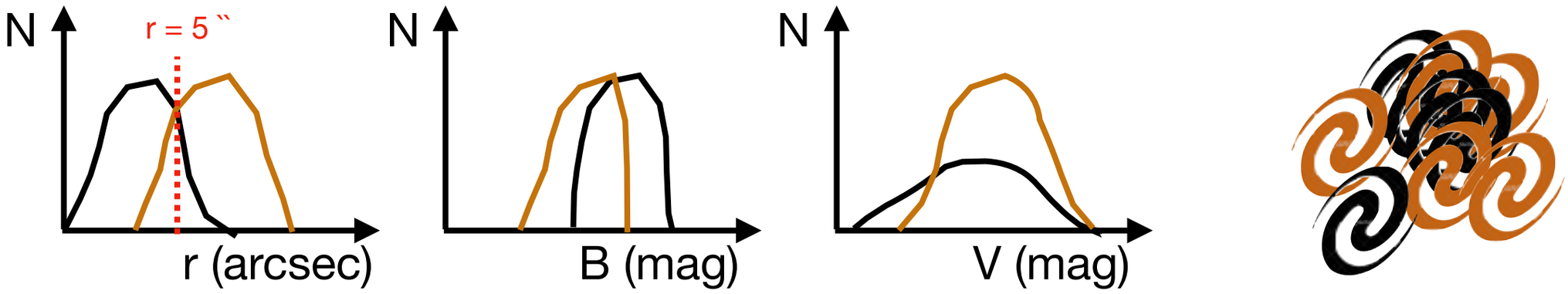
Decision Tree Construction

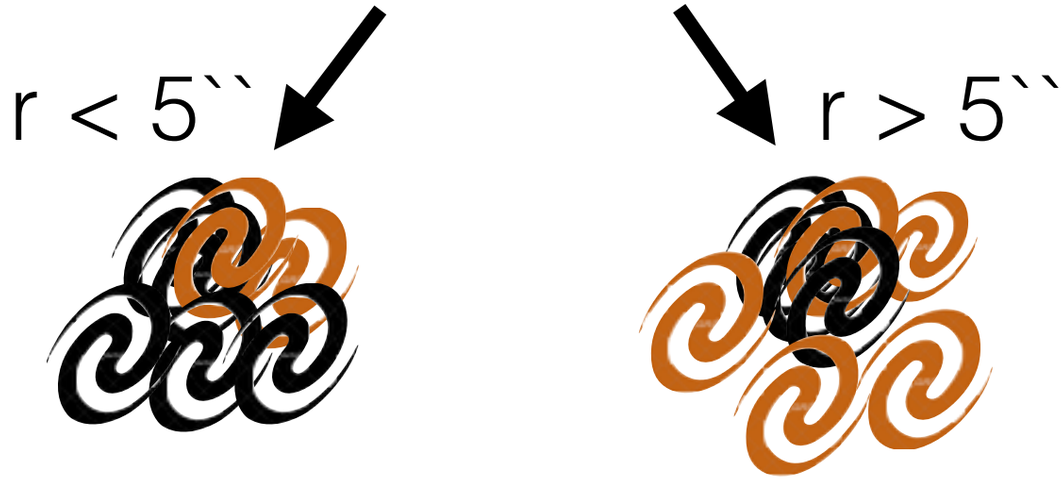
Input training set: a list of objects with measured features and known labels.
Classes: “black” and “brown” galaxies.
Measured features: r (arcsec), B (mag), V(mag).



Decision Tree Construction

Input training set: a list of objects with measured features and known labels.
Classes: “black” and “brown” galaxies.
Measured features: r (arcsec), B (mag), V(mag).

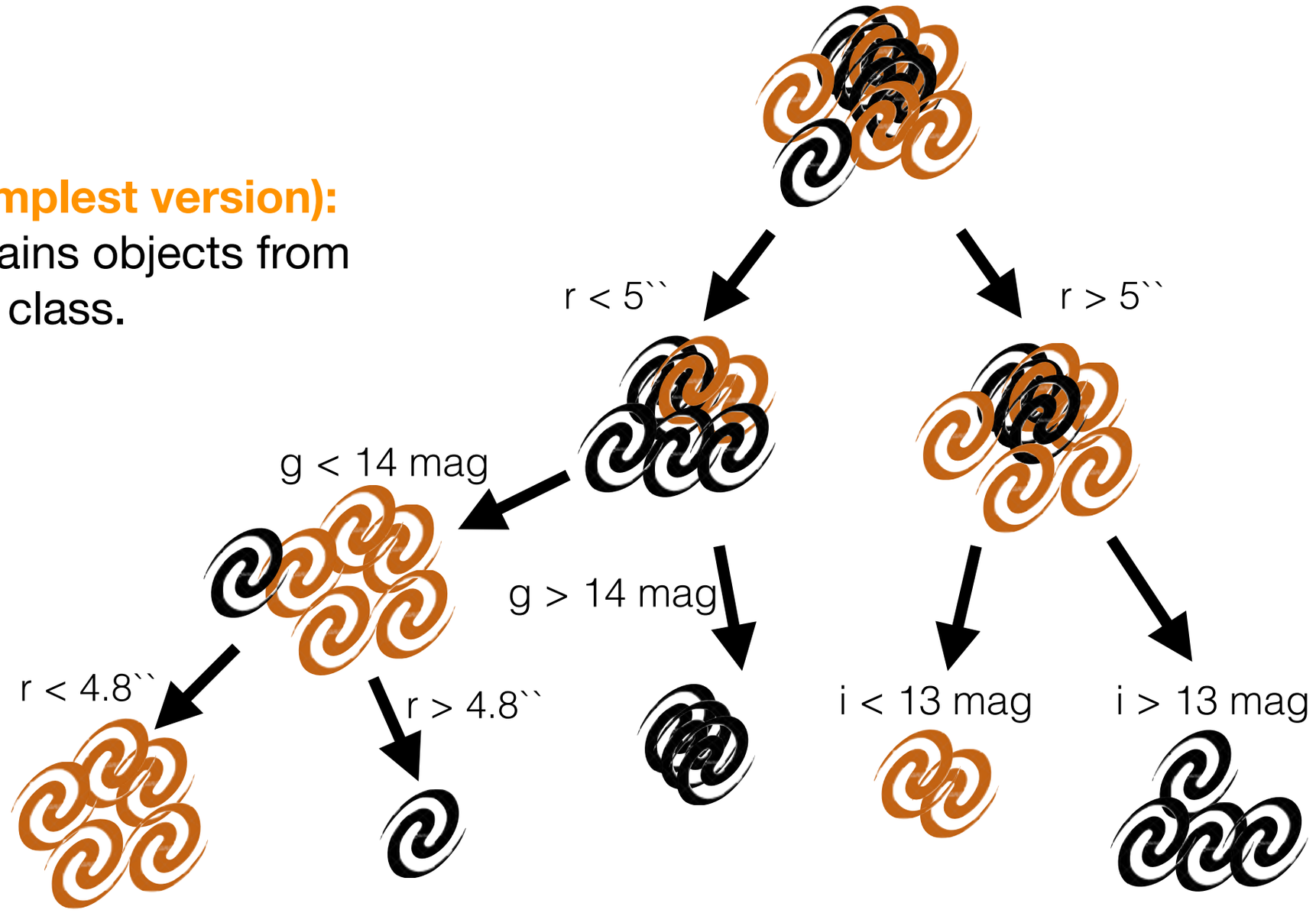




Decision Tree Construction

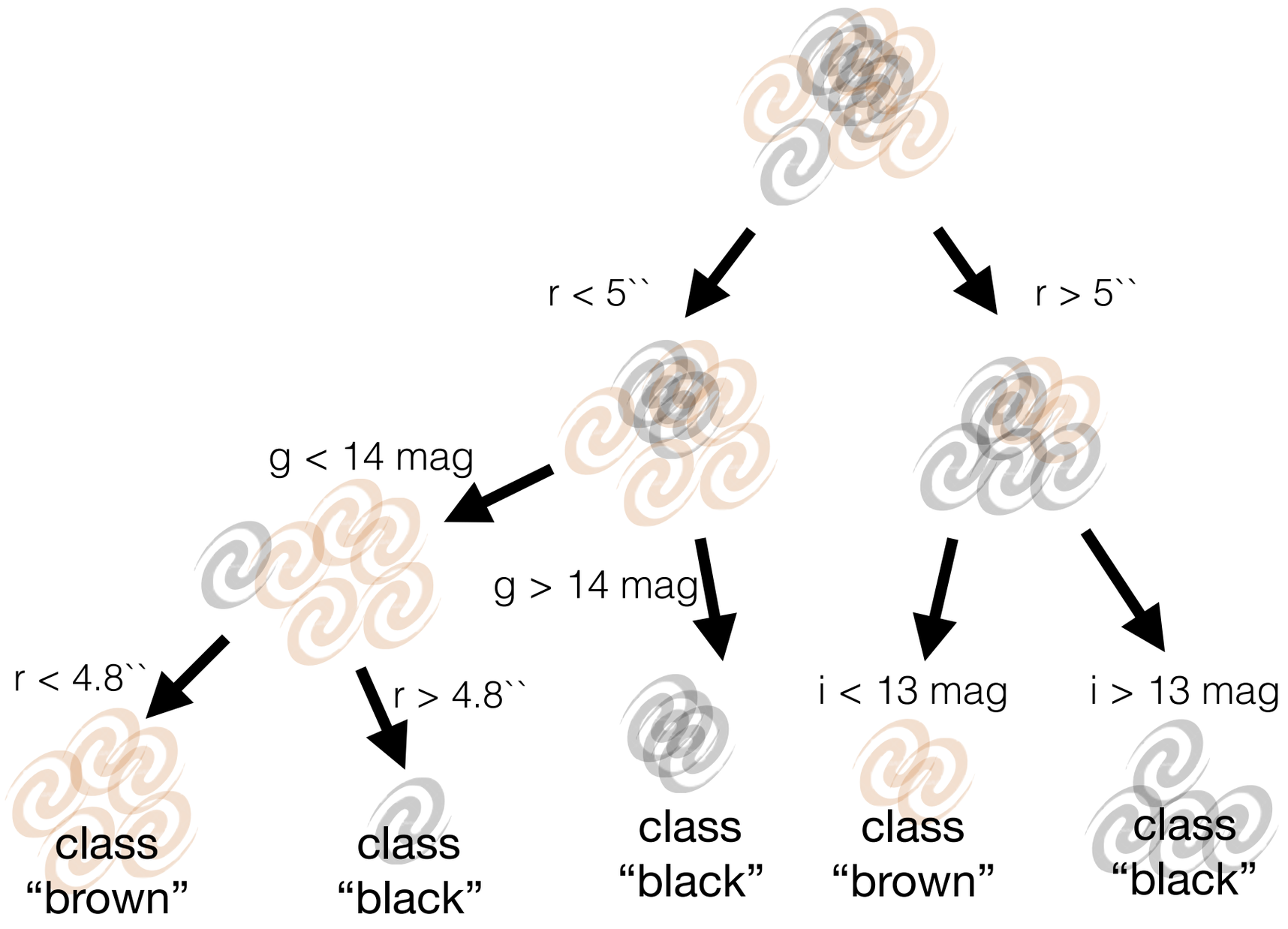
Input training set: a list of objects with measured features and known labels.
Classes: “black” and “brown” galaxies.
Measured features: r (arcsec), g (mag), i(mag).

Stop criterion (simplest version):
each terminal contains objects from a single class.



Decision Tree Prediction

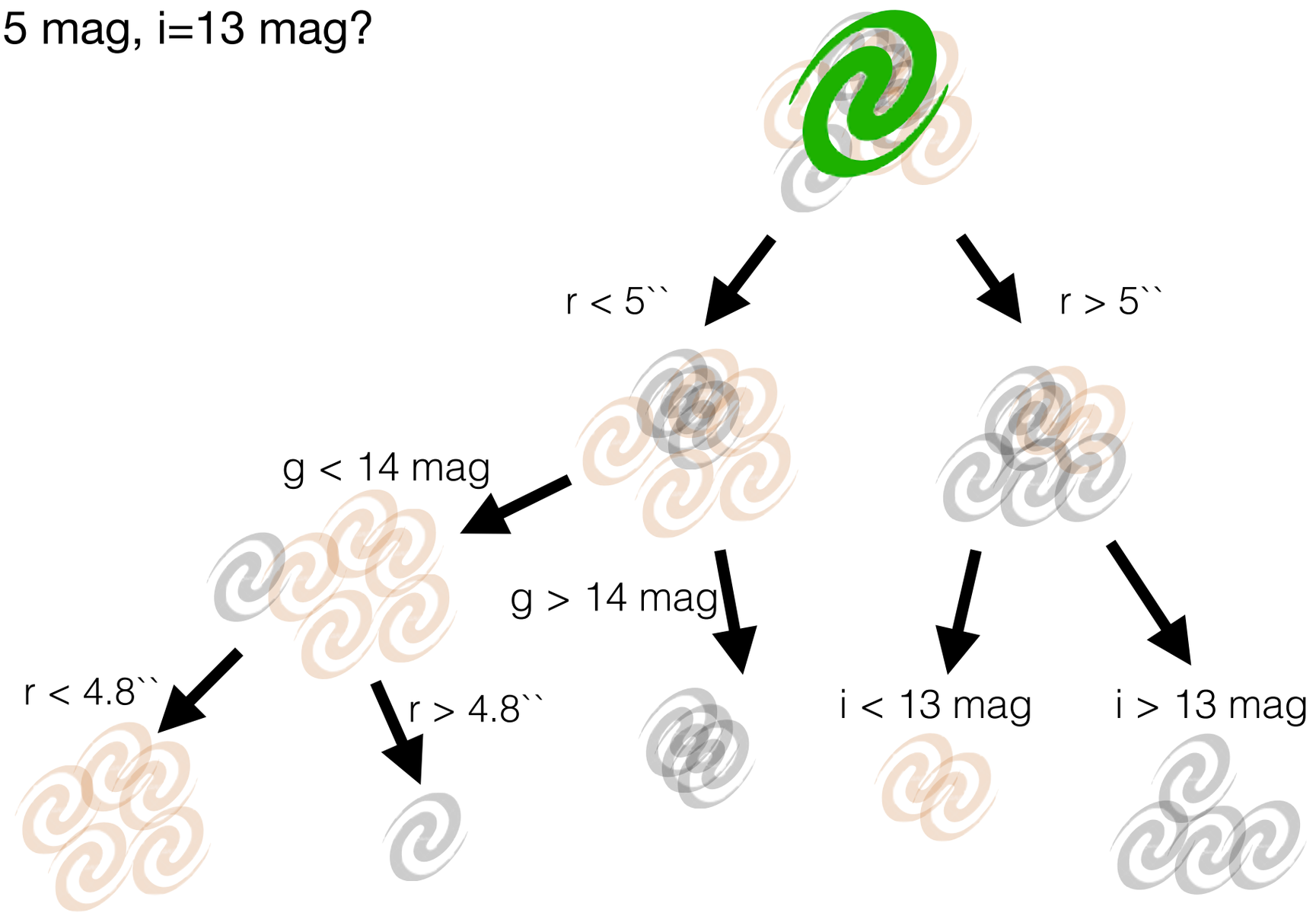
Input set: a list of objects with measured features and unknown labels.
Objects are propagated through the tree according to their measured features.



Decision Tree Prediction

Input set: a list of objects with measured features and unknown labels.
Objects are propagated through the tree according to their measured features.

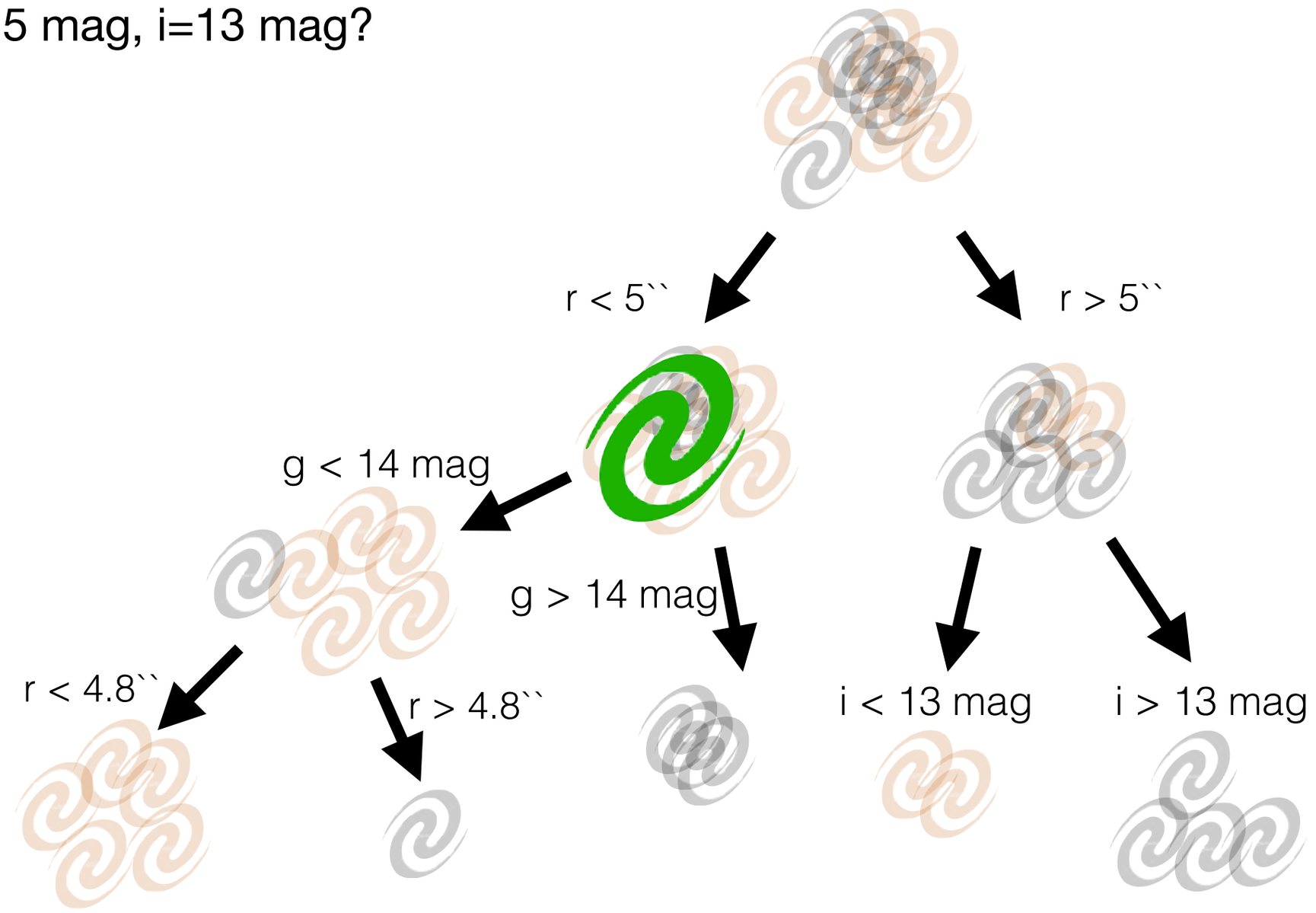
Example: what is the predicted label for a galaxy with the measured features:
r=3``, g=15 mag, i=13 mag?



Decision Tree Prediction

Input set: a list of objects with measured features and unknown labels.
Objects are propagated through the tree according to their measured features.

Example: what is the predicted label for a galaxy with the measured features:
r=3``, g=15 mag, i=13 mag?

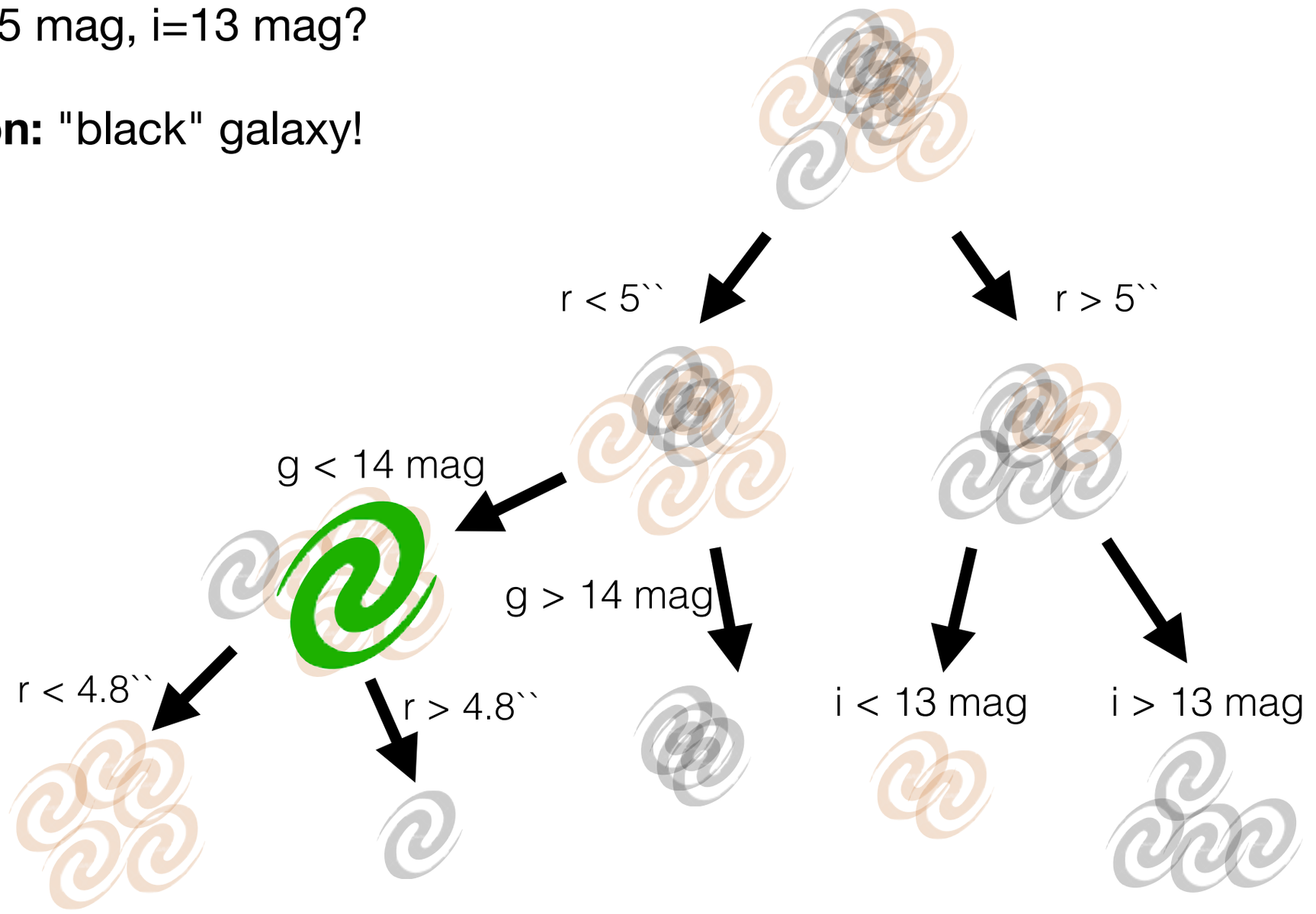


Decision Tree Prediction

Input set: a list of objects with measured features and unknown labels.
Objects are propagated through the tree according to their measured features.

Example: what is the predicted label for a galaxy with the measured features:
r=3``, g=15 mag, i=13 mag?

Prediction: "black" galaxy!

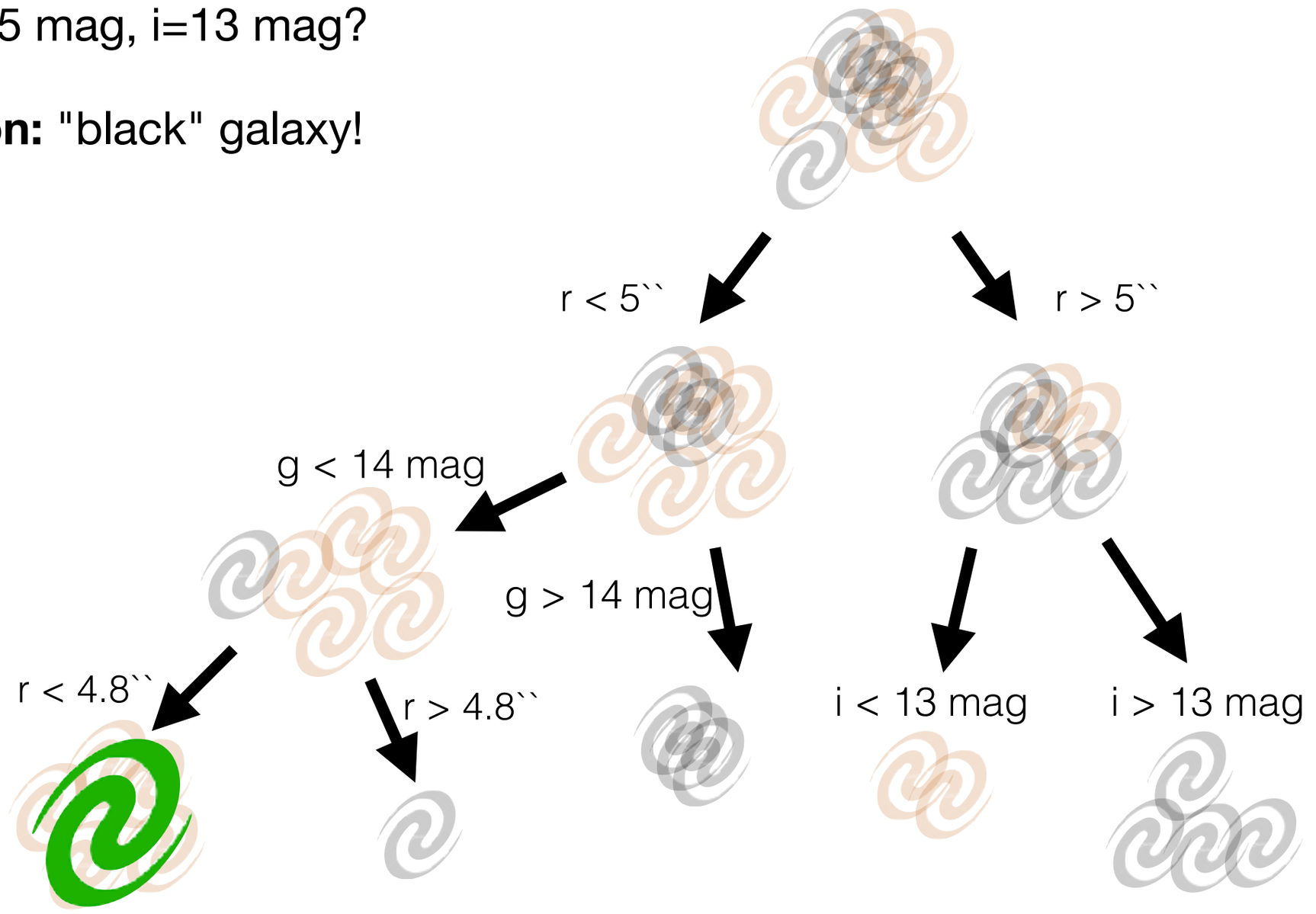


Decision Tree Prediction

Input set: a list of objects with measured features and unknown labels.
Objects are propagated through the tree according to their measured features.

Example: what is the predicted label for a galaxy with the measured features:
r=3``, g=15 mag, i=13 mag?

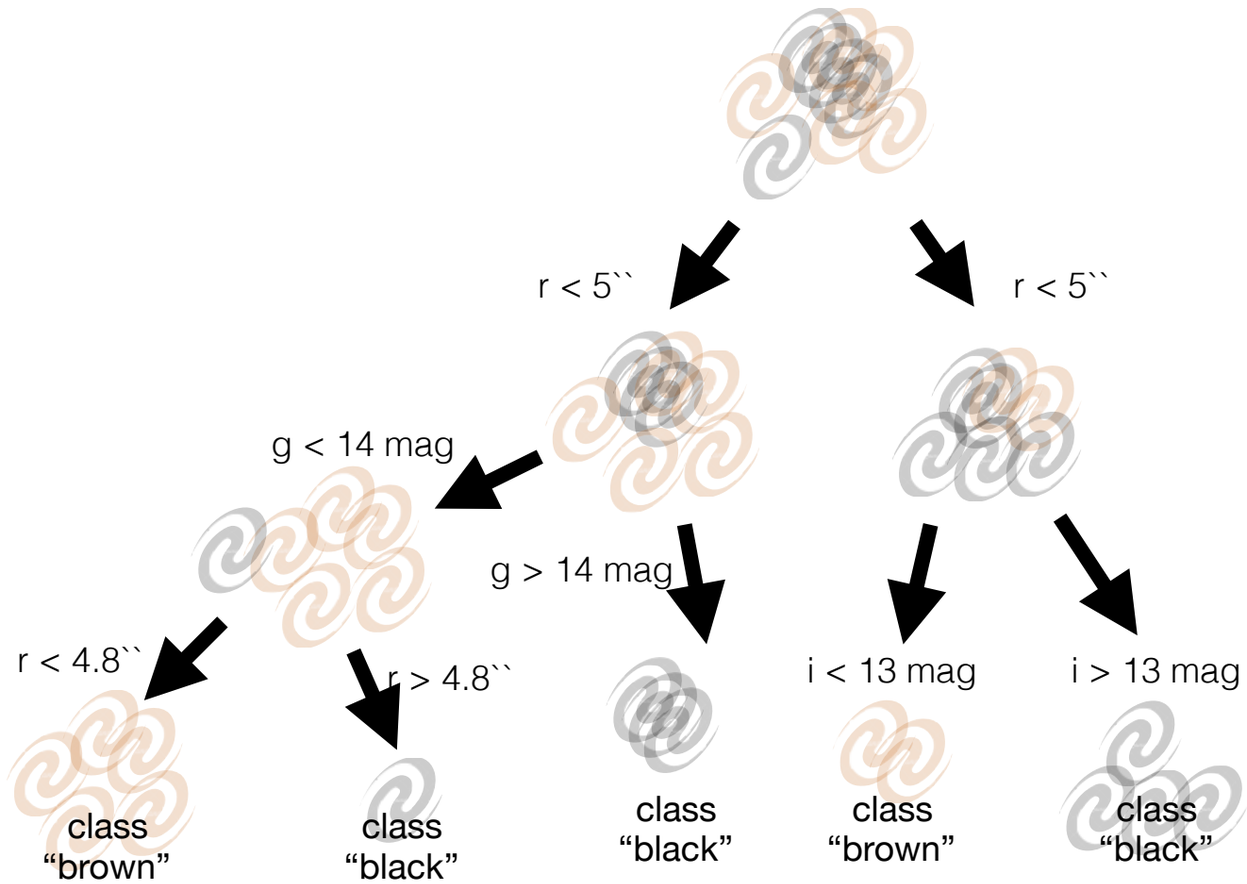
Prediction: "black" galaxy!



Decision Trees: Pros & Cons

Advantages:

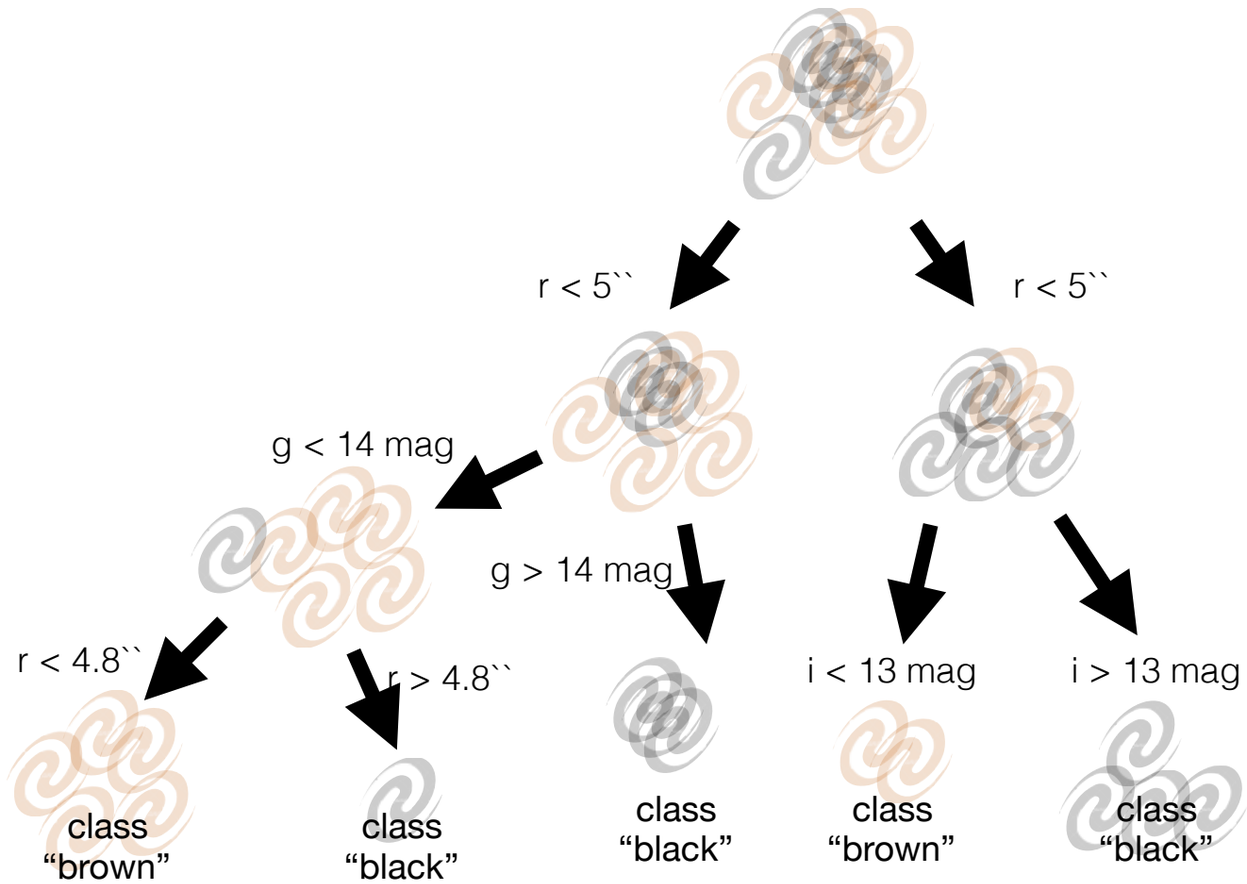
(1) Non-linear model, which is constructed during training.
(2) In its simplest version, very few hyper-parameters.
(3) Handles numerous features and numerous objects.
(4) No need to scale the feature values to the same “units”.
(5) Produces classification probability (in its more complex version).
(6) Produces feature importance.



Decision Trees: Pros & Cons

Advantages:

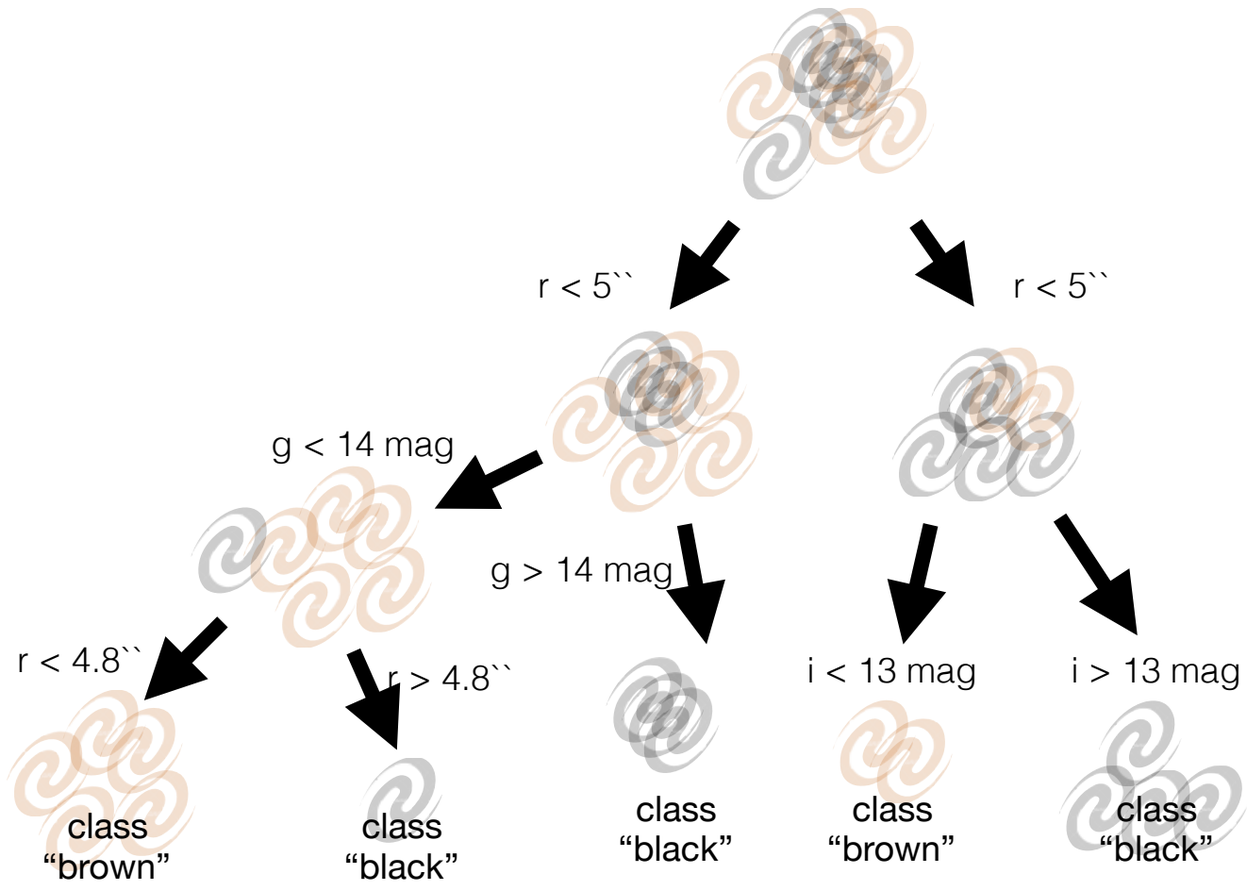
(1) Non-linear model, which is constructed during training.
(2) In its simplest version, very few free parameters.
(3) Handles numerous features and numerous objects.
(4) No need to scale the feature values to the same “units”.
(5) Produces classification probability (in its more complex version).
(6) Produces feature importance.



Feature importance & feature selection

Rule of thumb: the higher a feature is in a decision tree, the more important it is for the classification task. The locations of features within the tree can be used to produce feature importance.
In our example, feature importance: r, i, and then g.

Useful trick: add non-informative features to your dataset (a feature with random values, or a constant feature). If your physical features are ranked less important, remove them!



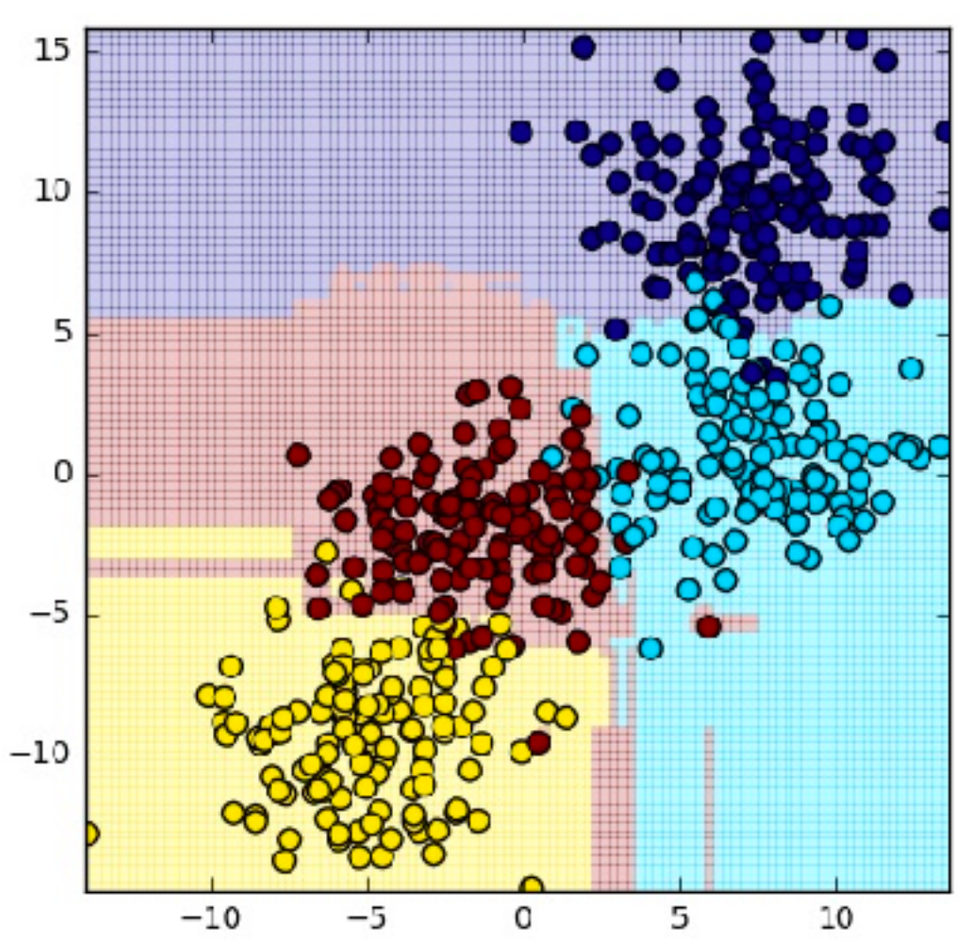
Decision Trees: Pros & Cons

Advantages:

(1) Non-linear model, which is constructed during training.
(2) In its simplest version, very few free parameters.
(3) Handles numerous features and numerous objects.
(4) No need to scale the feature values to the same “units”.
(5) Produces classification probability (in its more complex version).
(6) Produces feature importance.

Disadvantages:

(1) Usually does not generalize well to unseen datasets:
(1) Mediocre performance on test set.
(2) Tends to overfit.

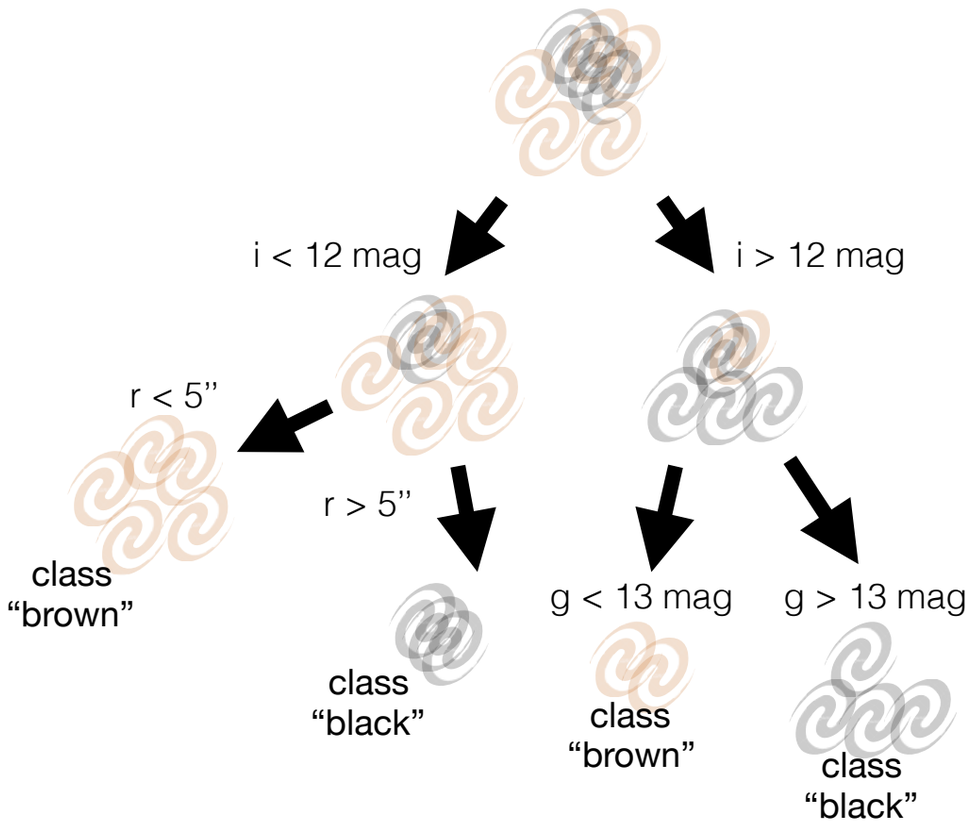


Random Forests

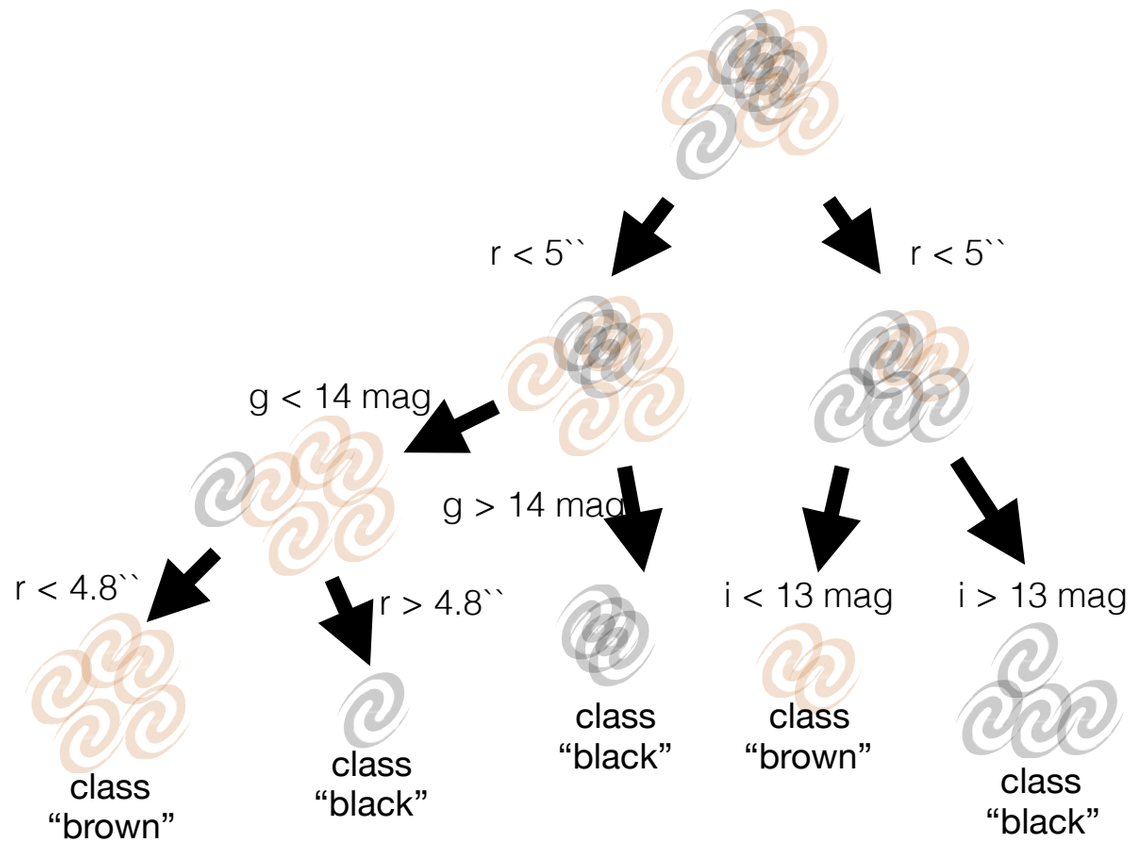
Random Forest is an ensemble of decision trees, where **randomness** is injected into the training process of each individual tree with a **bagging** approach.

Bagging: -The training set is split into randomly-selected subsets, and each decision tree is trained on a subset of the data.
-In each node in the decision tree, only a randomly-selected subset of the feature is considered.

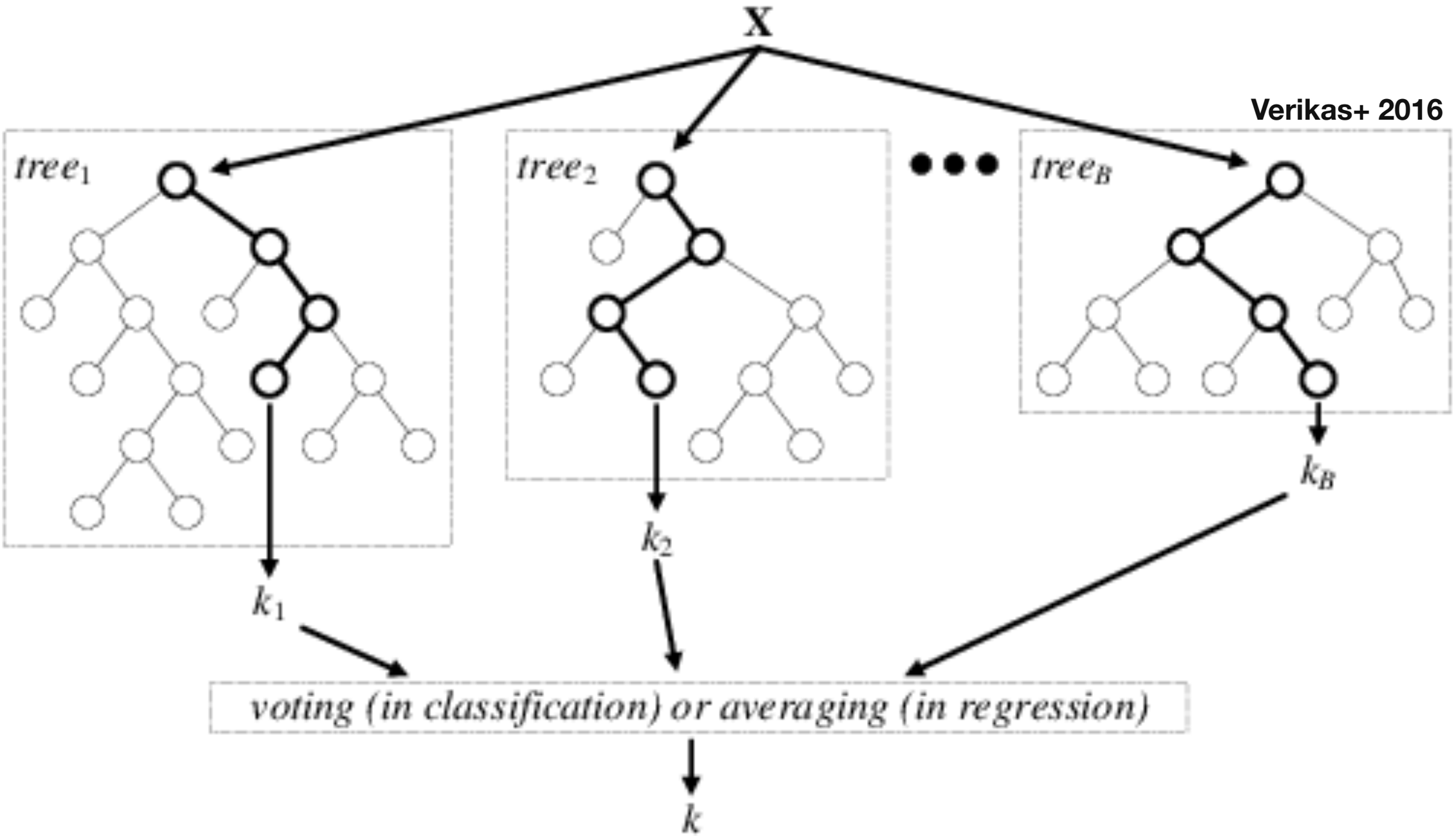
decision tree #2



decision tree #1



Random Forest Prediction



Verikas+ 2016

Random Forest Prediction

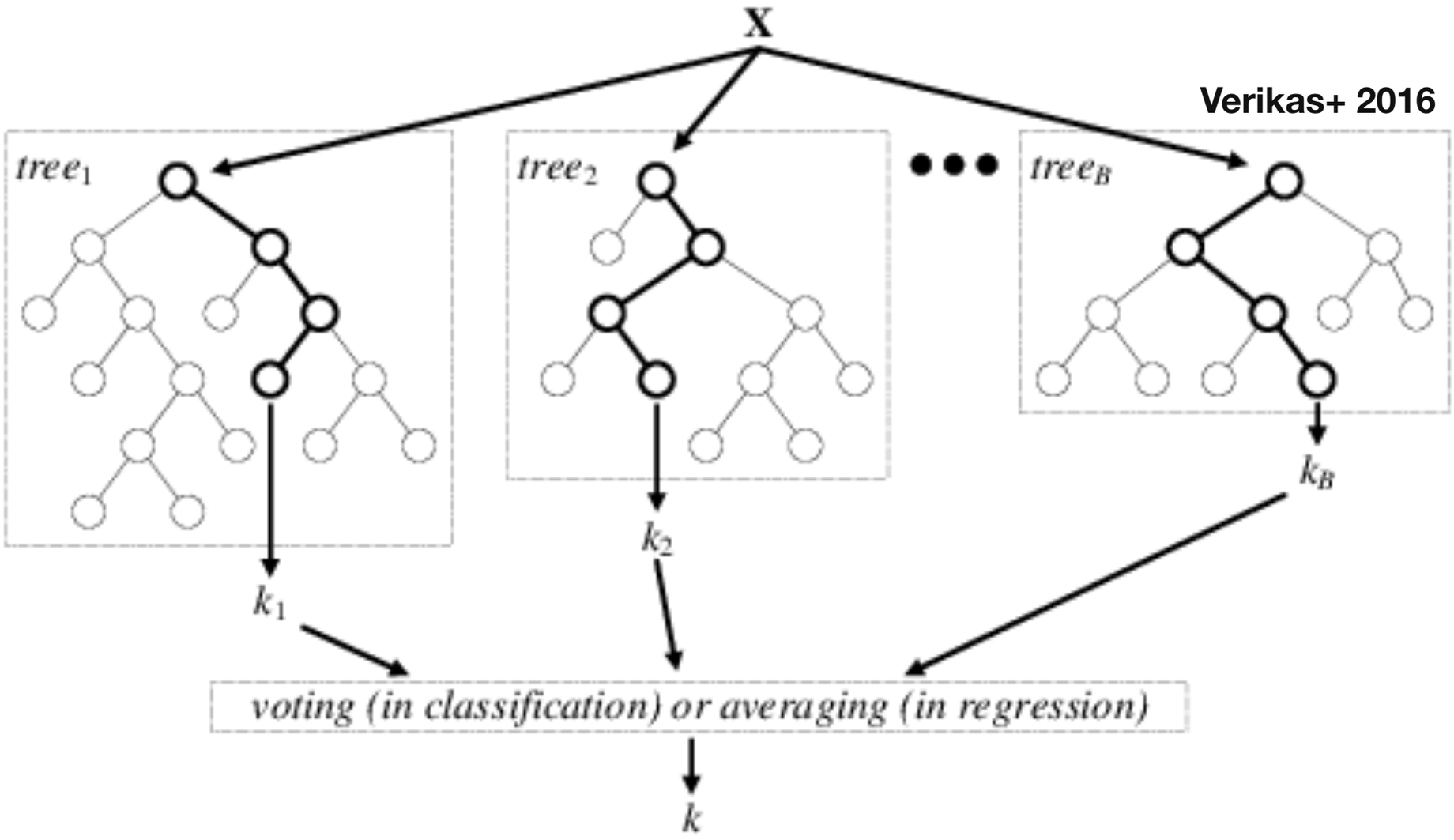
Hyper parameters:

(1) Number of trees in the forest

(2) Number of randomly-selected features to consider in each split.

(3) Splitting criterion (also for Decision Trees).

(4) Class weight.



Verikas+ 2016

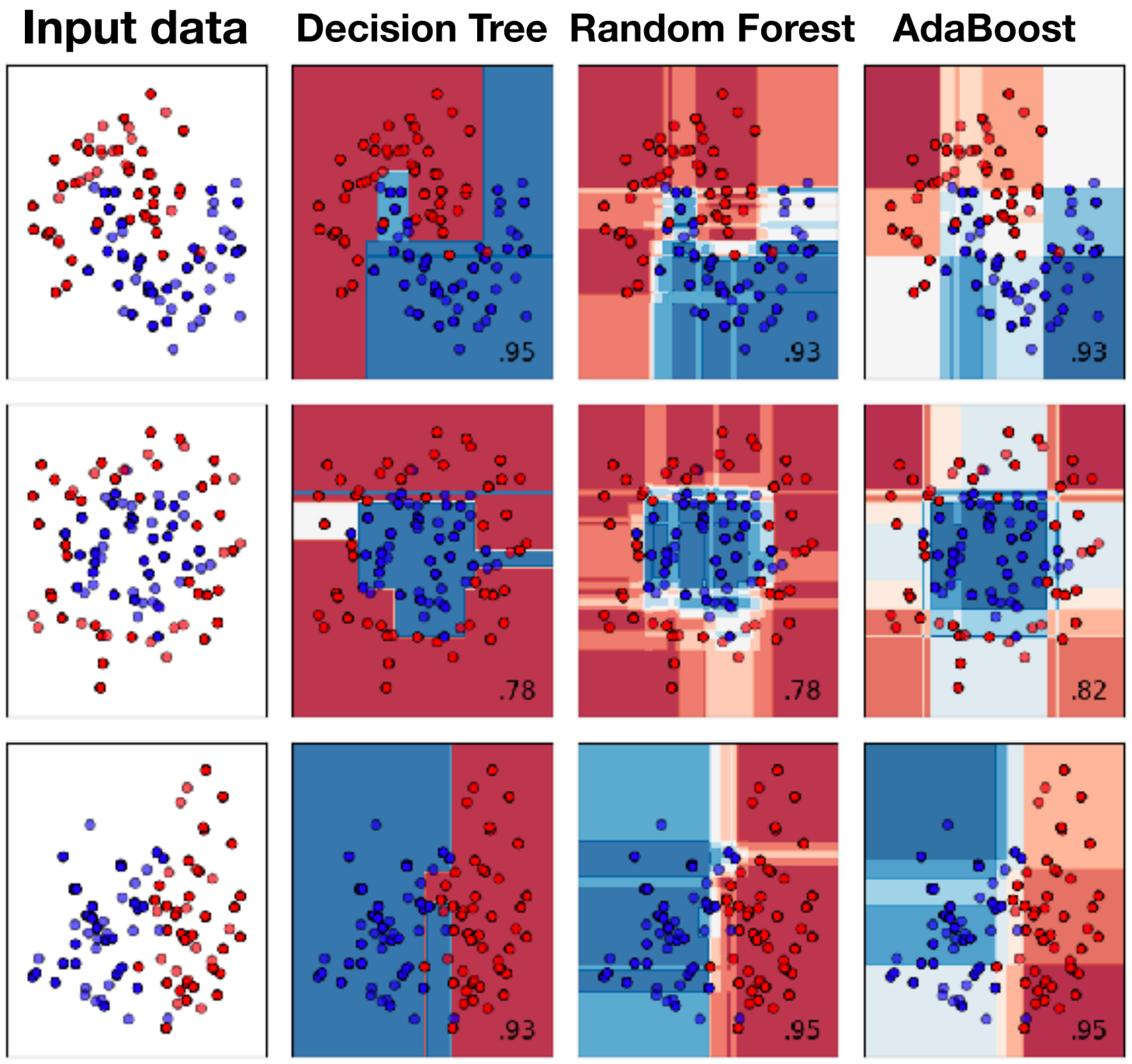
Random Forest: Pros & Cons

Advantages:

(1) Same advantages as in a single Decision Tree.
(2) Specifically, can handle thousands of features!
(3) Generalizes well to unseen datasets.
(4) Easily parallelizable.

Disadvantages:

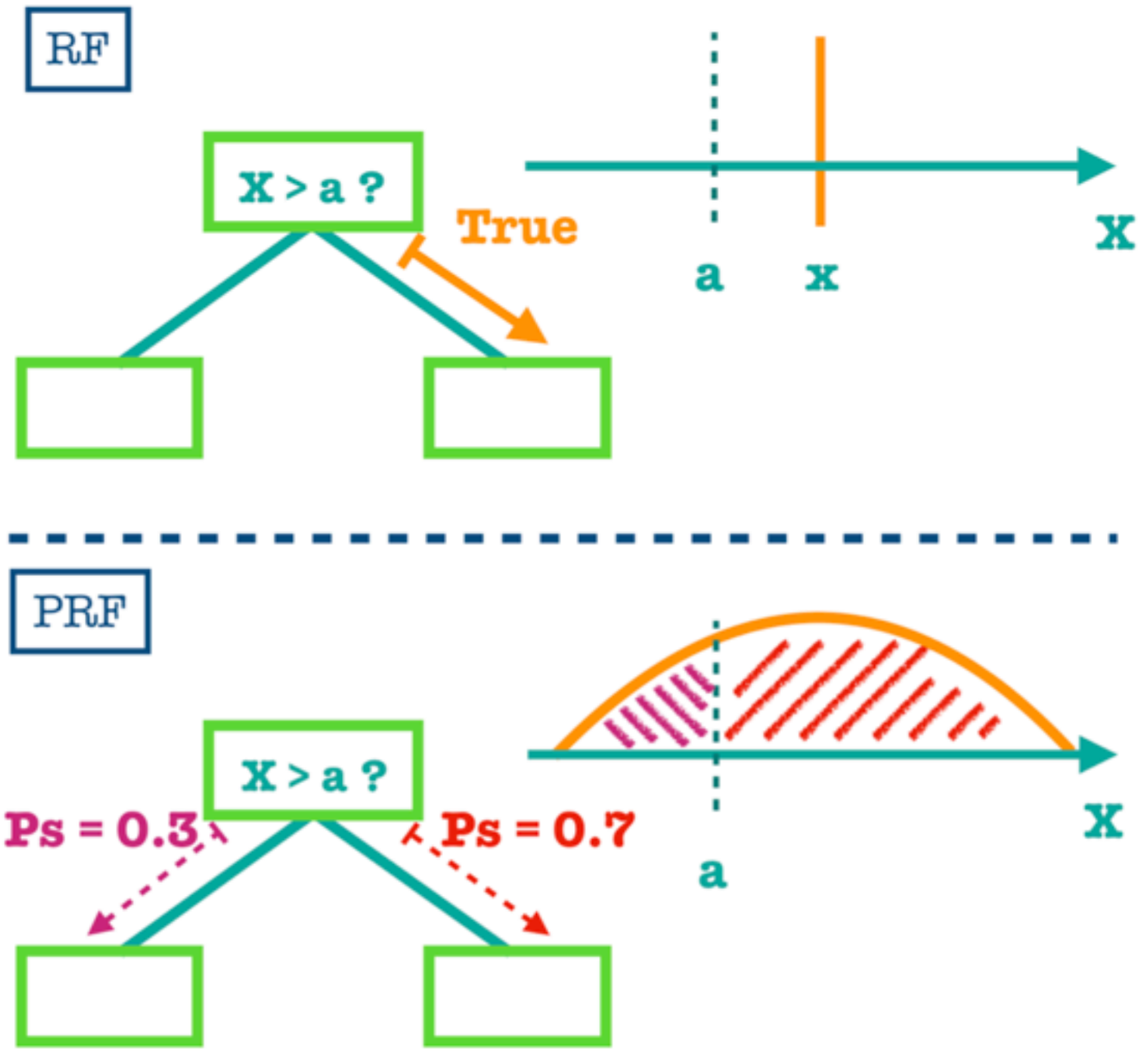
(1) Cannot handle measurement uncertainties (true for most ML algorithms!).



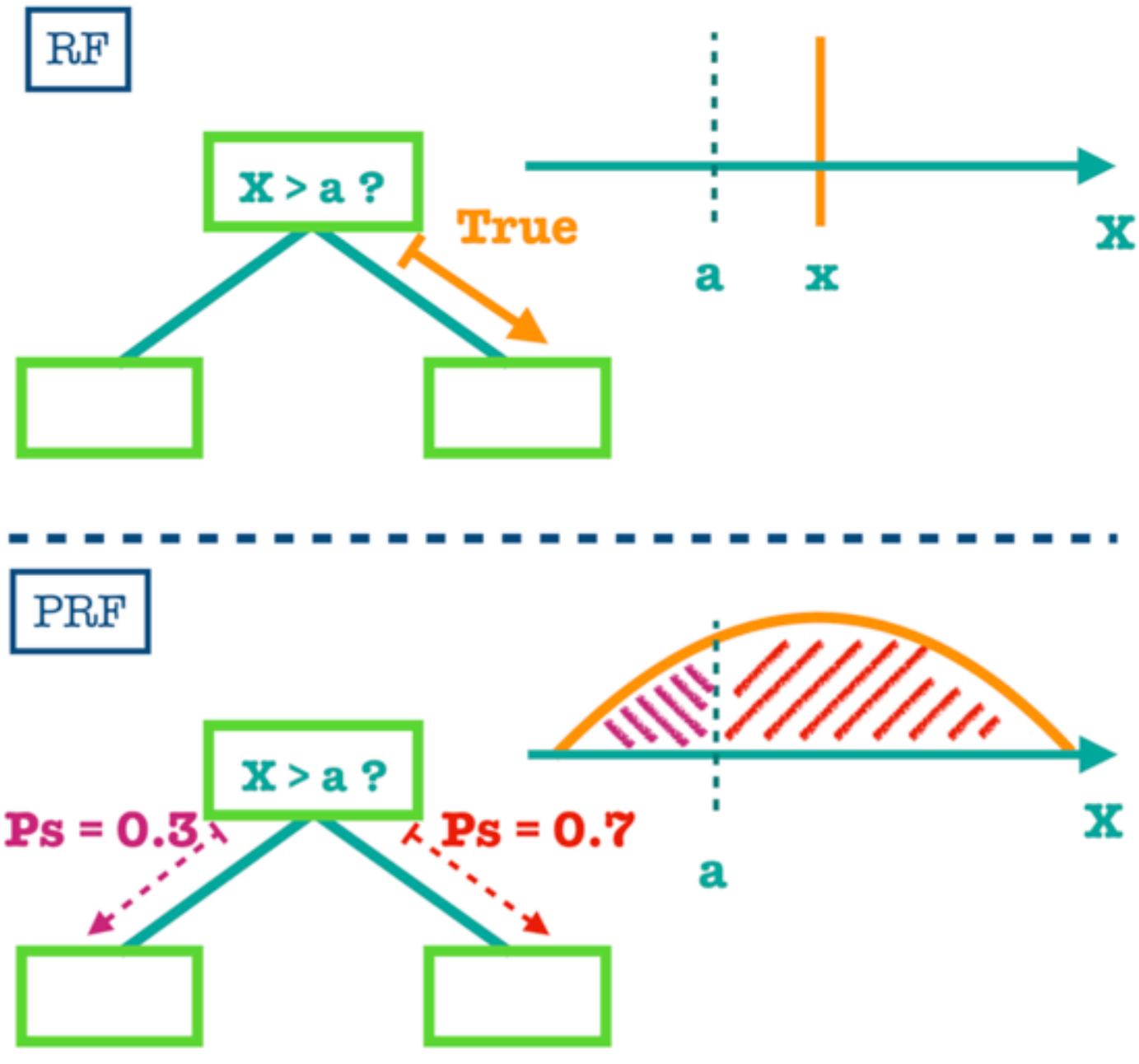
http://scikit-learn.org/stable

Probabilistic Random Forest

A Random Forest that takes into account the uncertainties in both the features and the input labels. The Probabilistic Random Forest treats all measurements as random variables (see Reis, Baron & Shahaf 2019).

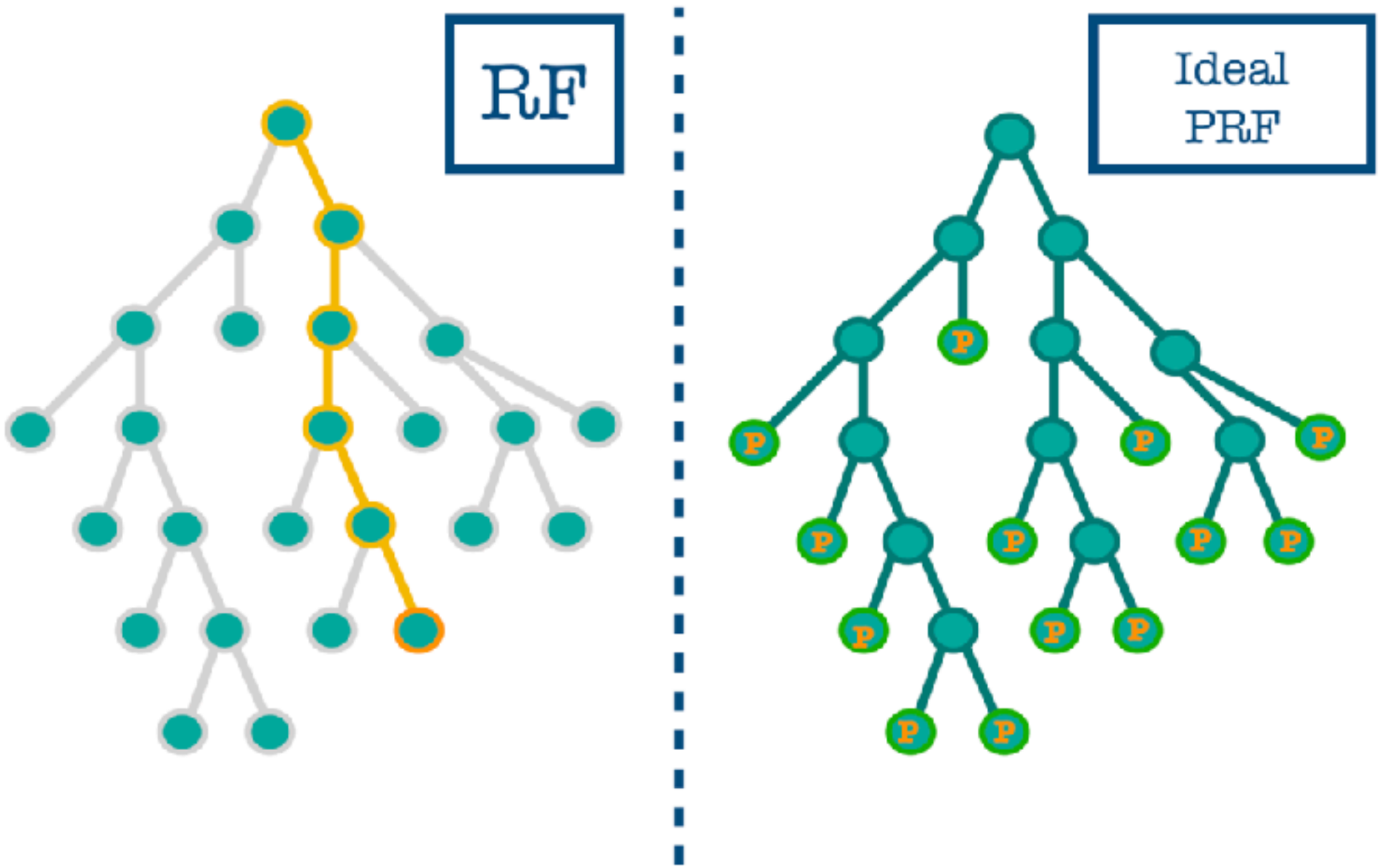


PRF is able to handle a dataset with missing values!

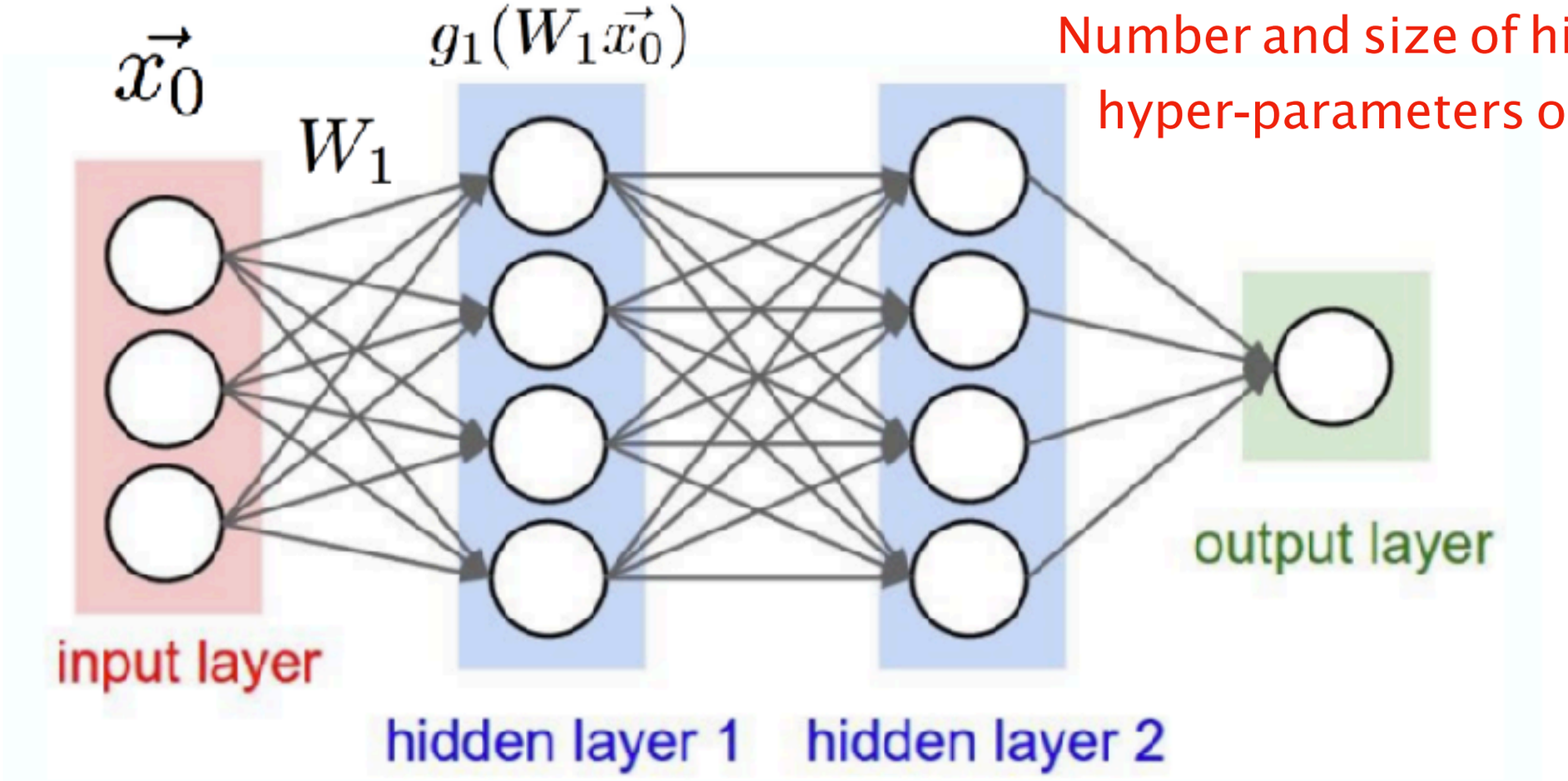


Probabilistic Random Forest

A Random Forest that takes into account the uncertainties in both the features and the input labels. The Probabilistic Random Forest treats all measurements as random variables (Reis, Baron & Shahaf 2019).



Artificial Neural Networks



Number and size of hidden layers: also hyper-parameters of the algorithm!

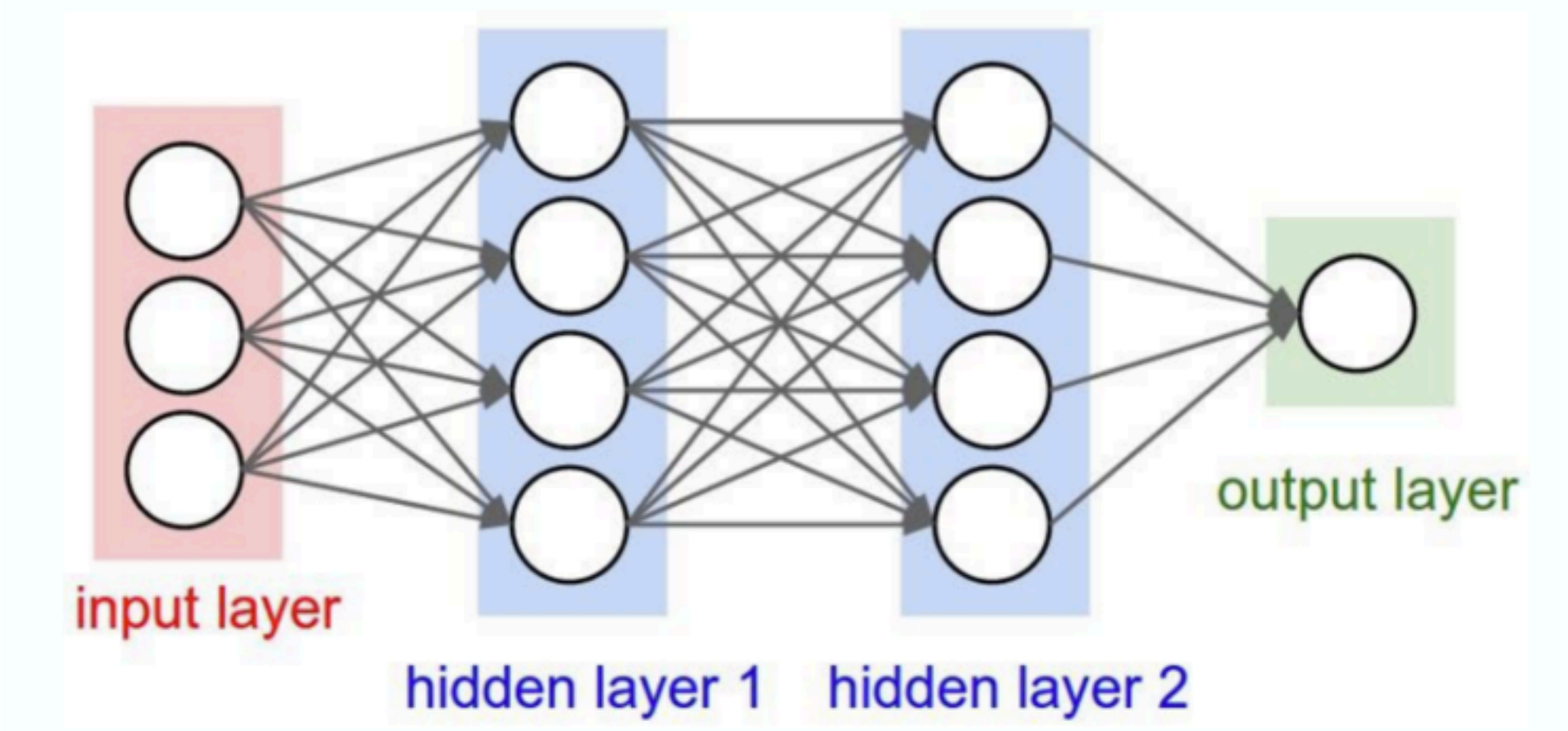
$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$ ← NETWORK FUNCTION

non-linear function

Hyper-parameter of the algorithm!

weights learned during training

Artificial Neural Networks

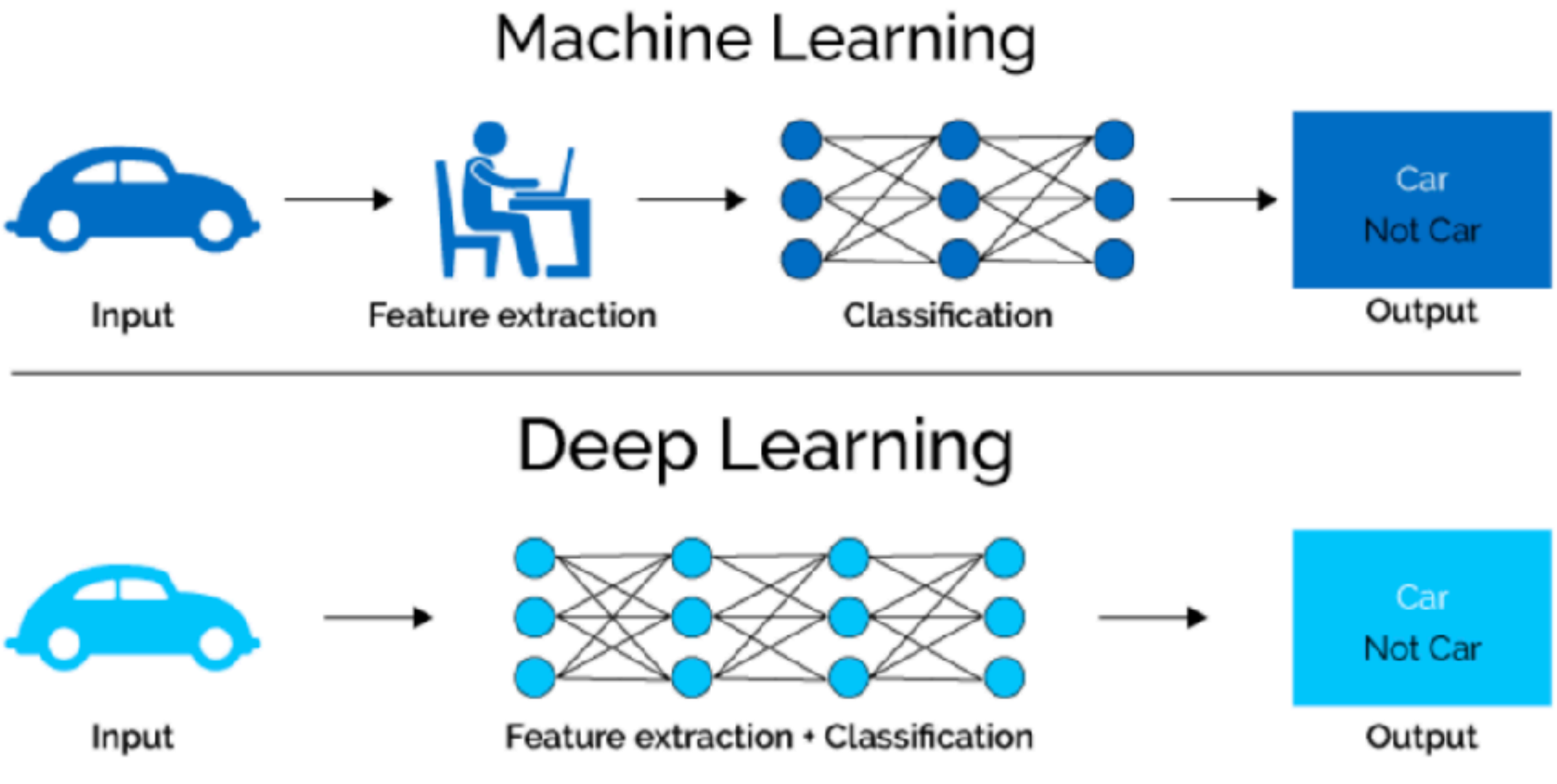


$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

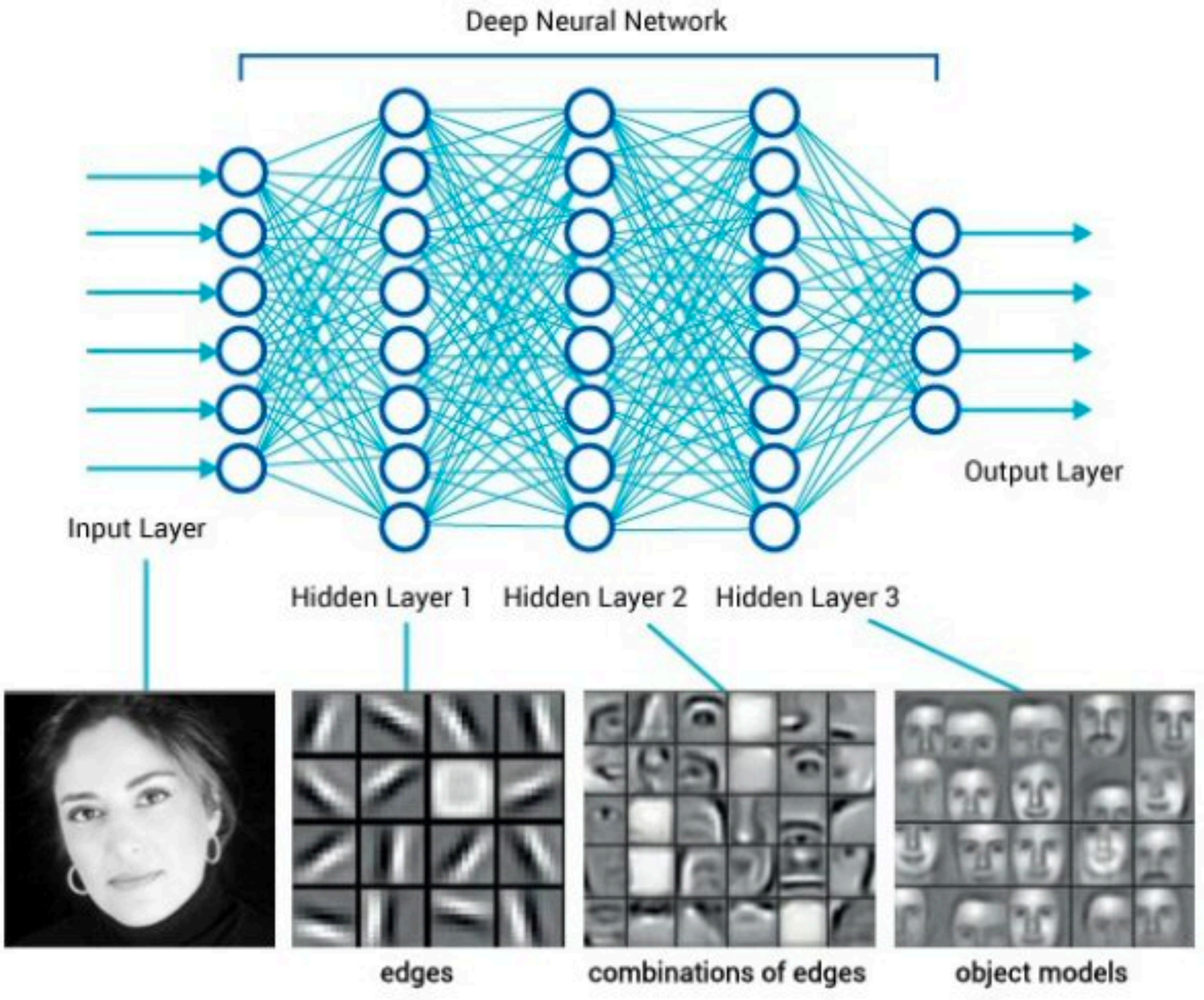
$$\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$ ← LOSS FUNCTION

Convolutional Neural Networks and Deep Learning

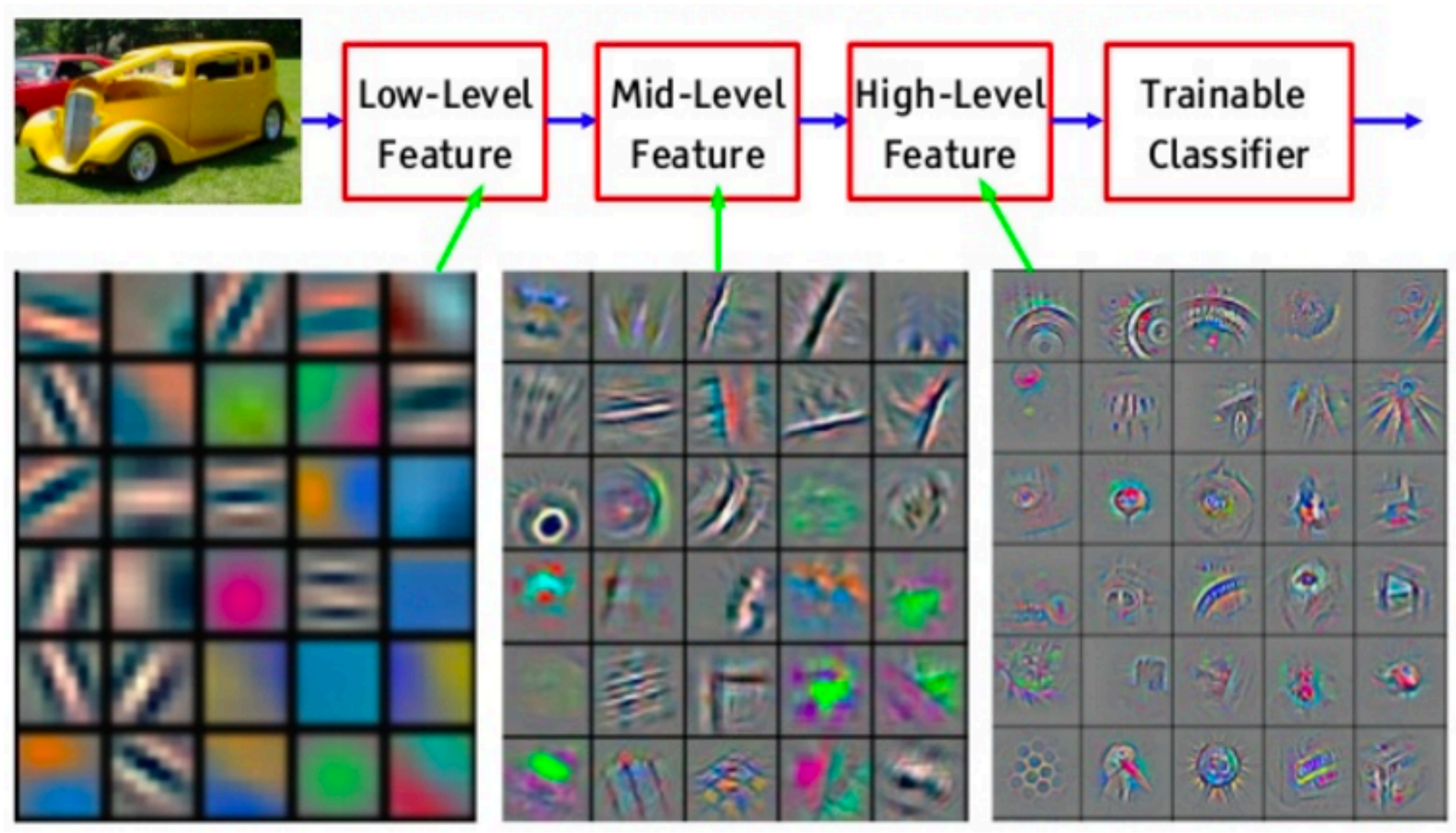
CNNs and deep learning are particularly useful in cases when we want to use raw data, like images, spectra, and light-curves. This is because convolutional layers practically perform their own feature extraction!



Convolutional Neural Networks and Deep Learning



Going deep: increasing dramatically the model complexity, and loosing control

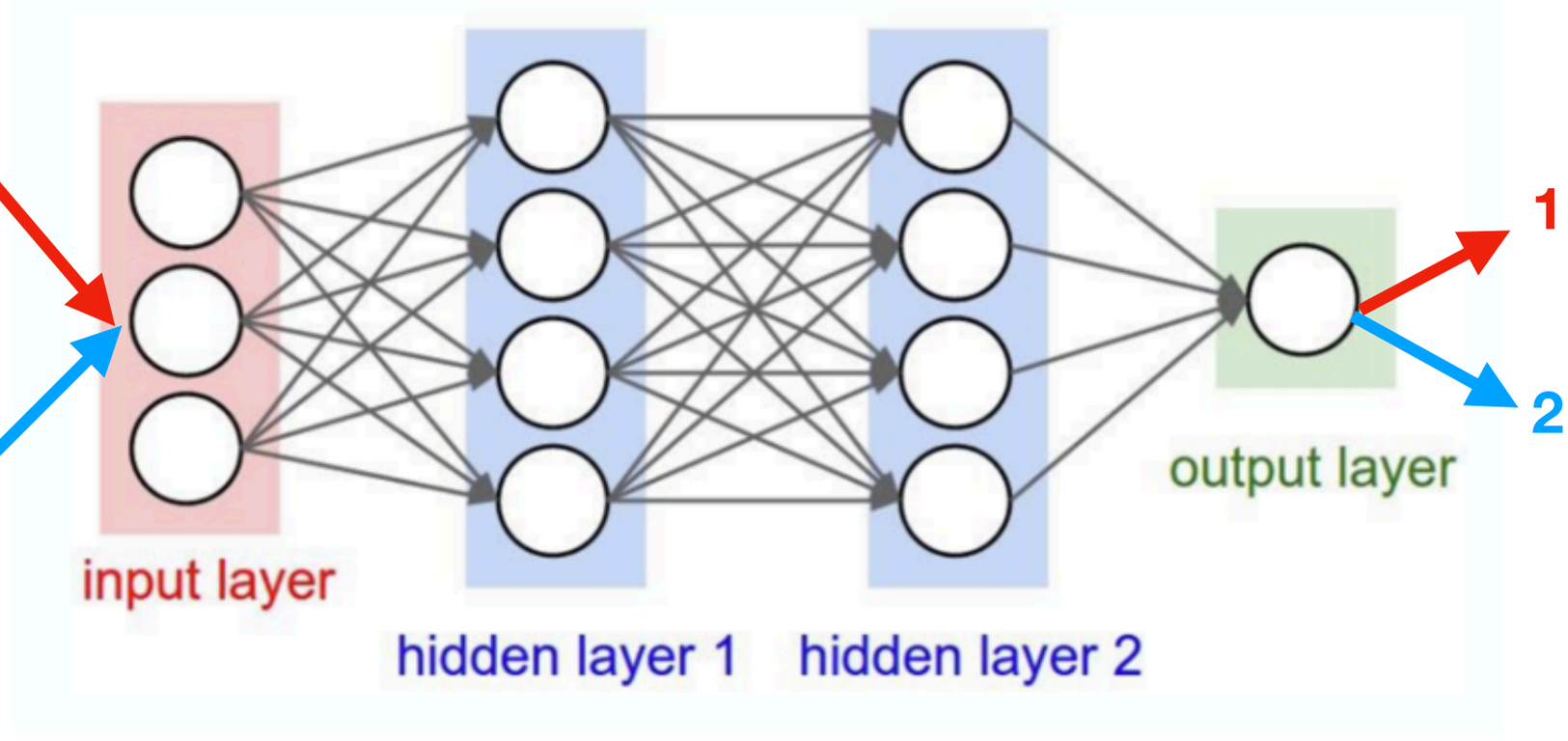


CNNs - the effect of imbalanced dataset

We have a more than a billion of galaxy images. Out of these, a very small fraction are strong gravitational lenses. How do we find them?

1,000 gravitational lenses

10^9 regular galaxies

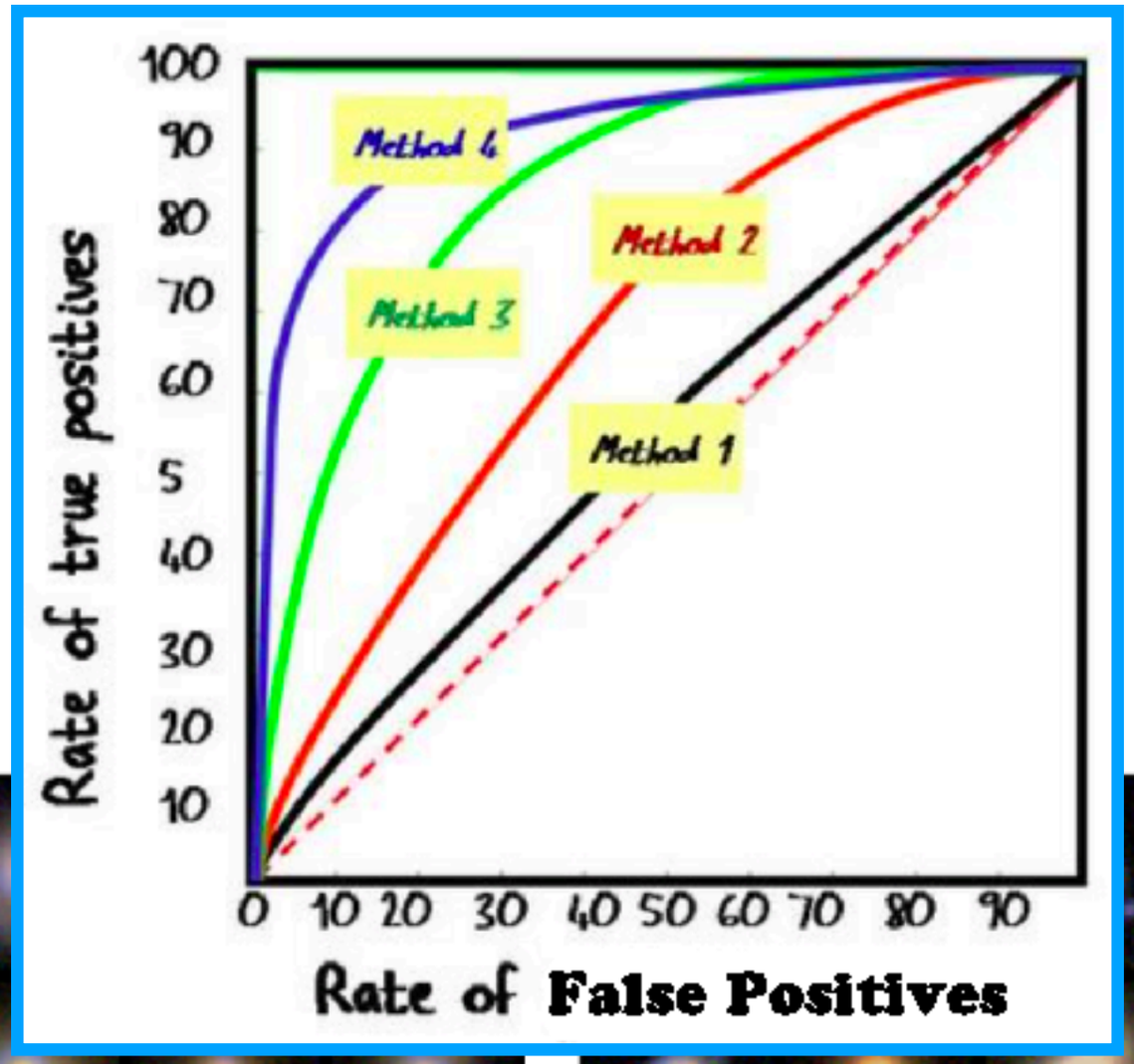


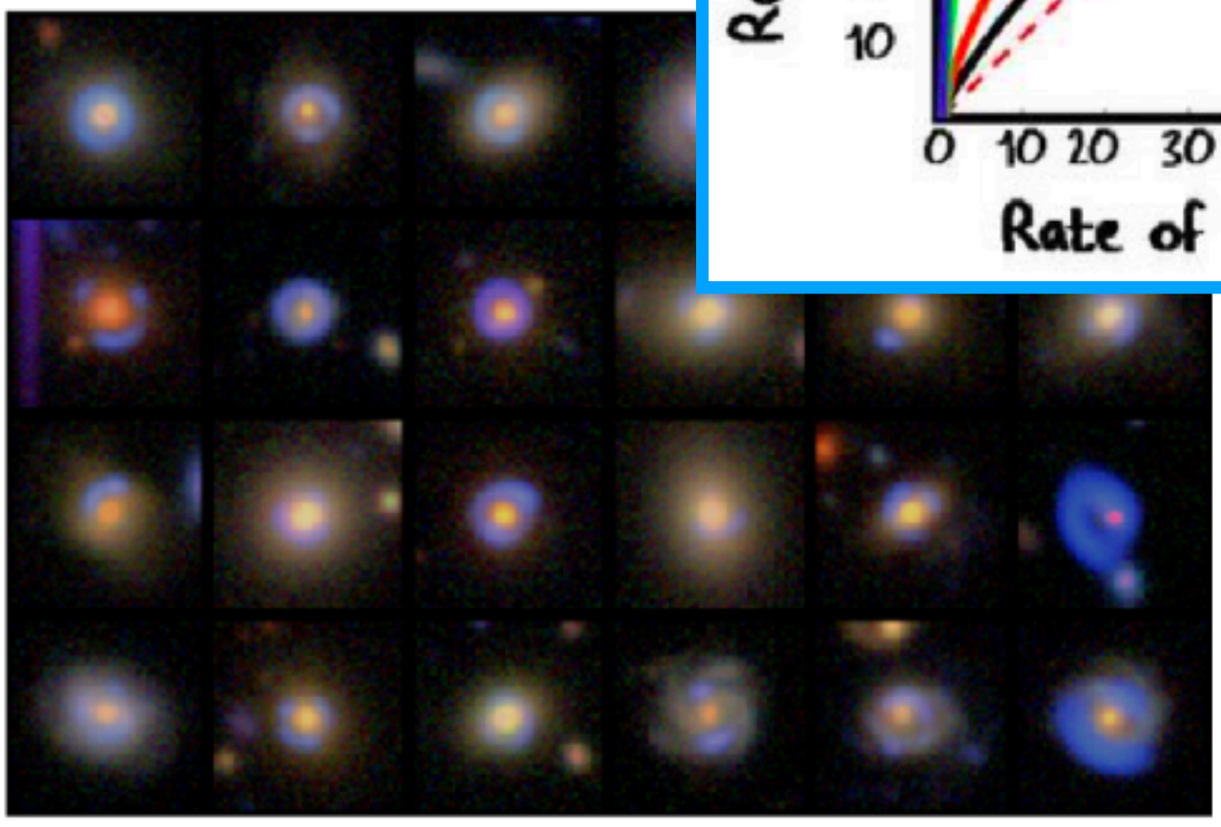
$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$

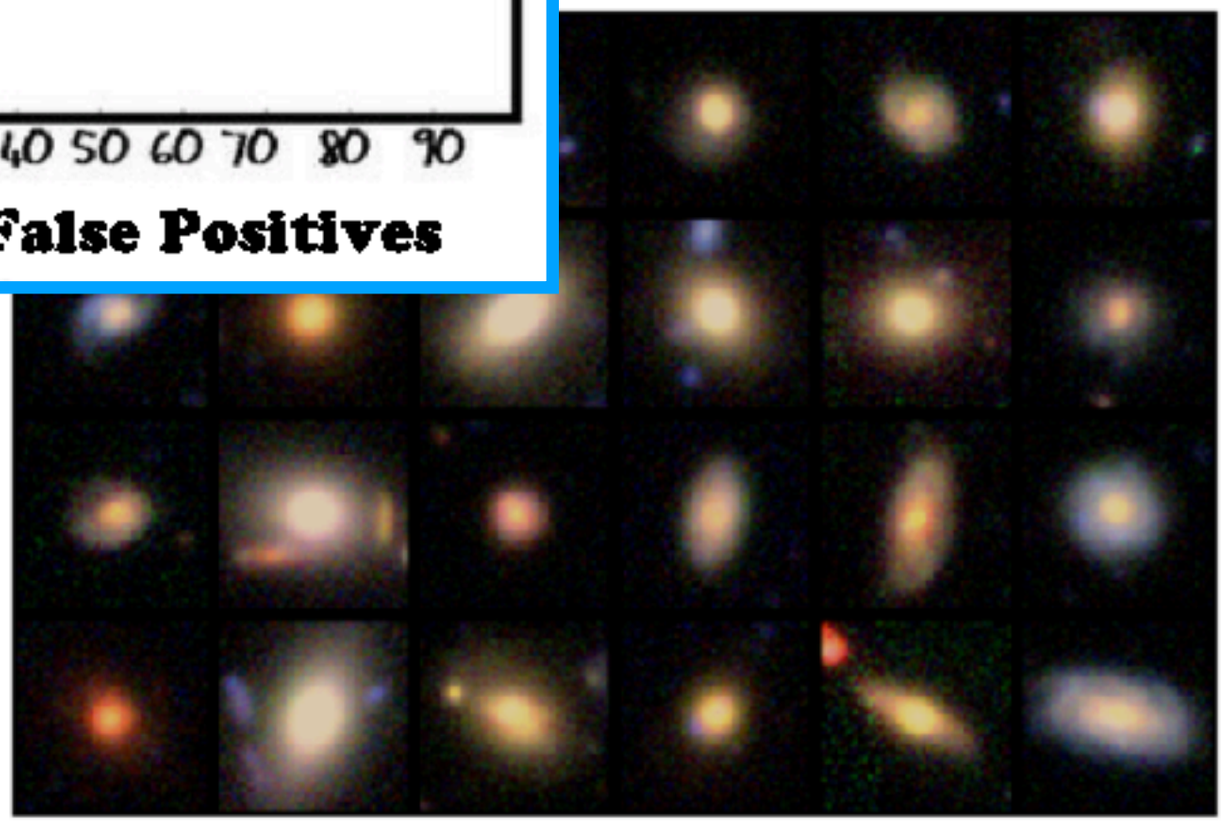
LOSS FUNCTION

Imbalanced datasets and the ROC curve





LENS

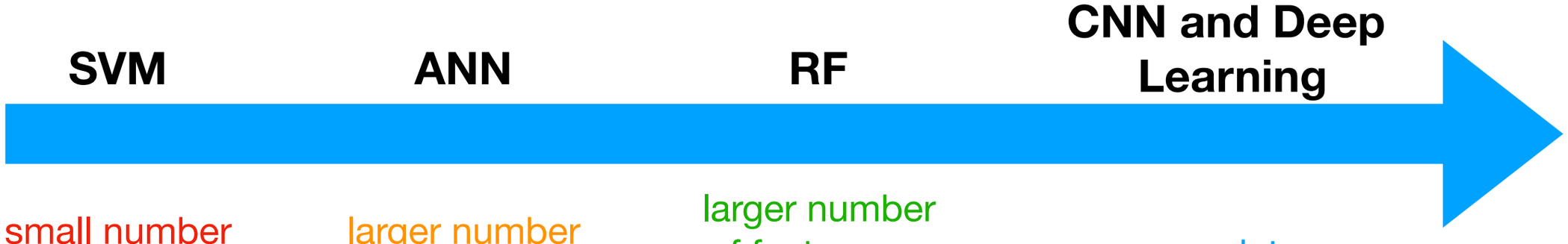


NON-LENS

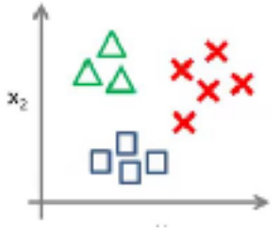
but Dalya, which algorithm should I use?!

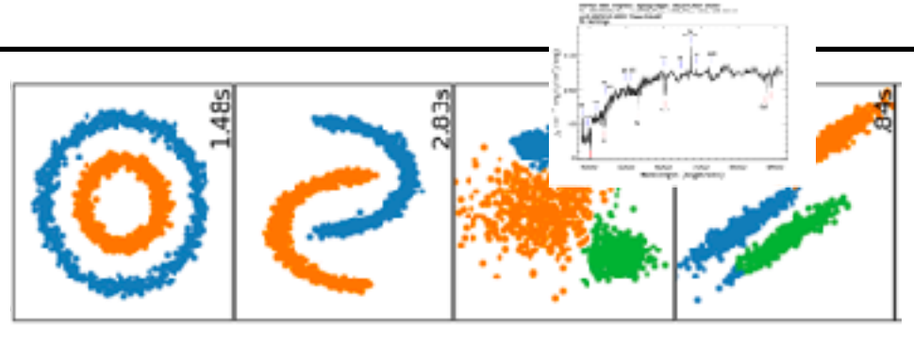
Generally, start with the simplest, and slowly increase the complexity.

Use the algorithm that gives you an acceptable result.

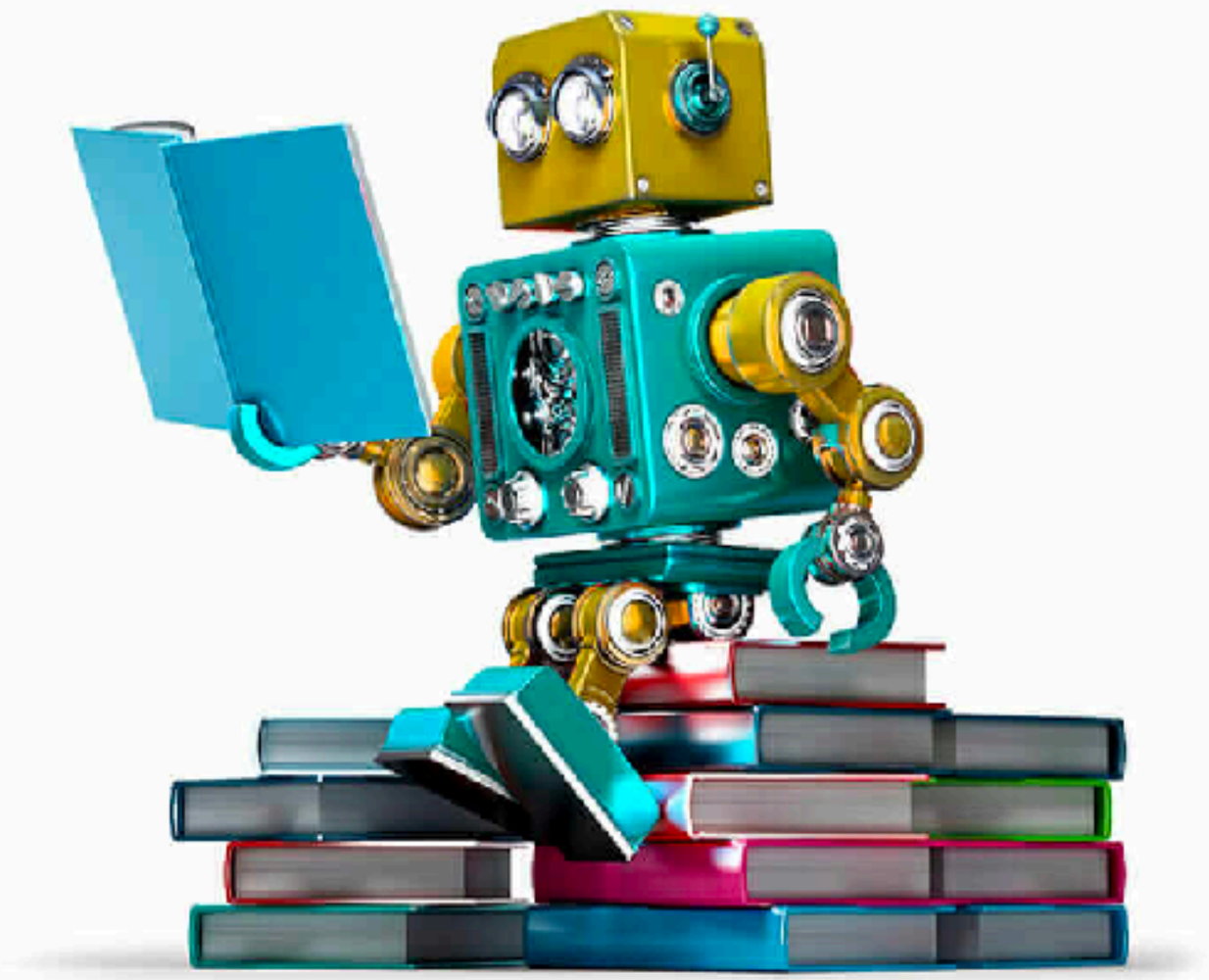


	SVM	ANN	RF	CNN and Deep Learning
input dataset	small number of features	larger number of features	larger number of features or raw data	raw data
model complexity	low	mediocare	mediocare	highly-complex
number of hyper parameters	~4	~10	~4	~10^6
overfitting?	no	maybe	maybe	sometimes
time to converge	seconds	~10s seconds	~10s seconds	could take a PHD time





Questions?



Unsupervised Learning

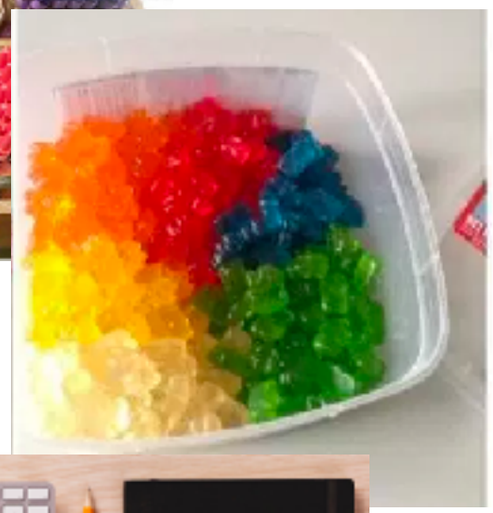
or, human & machine-made discoveries

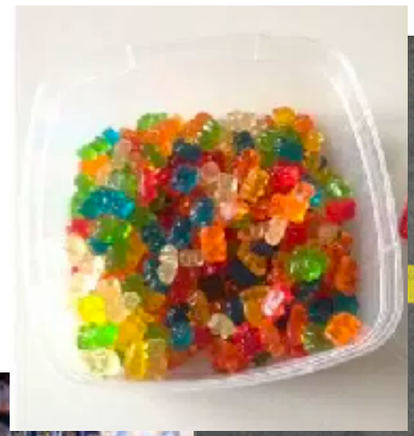
Humans

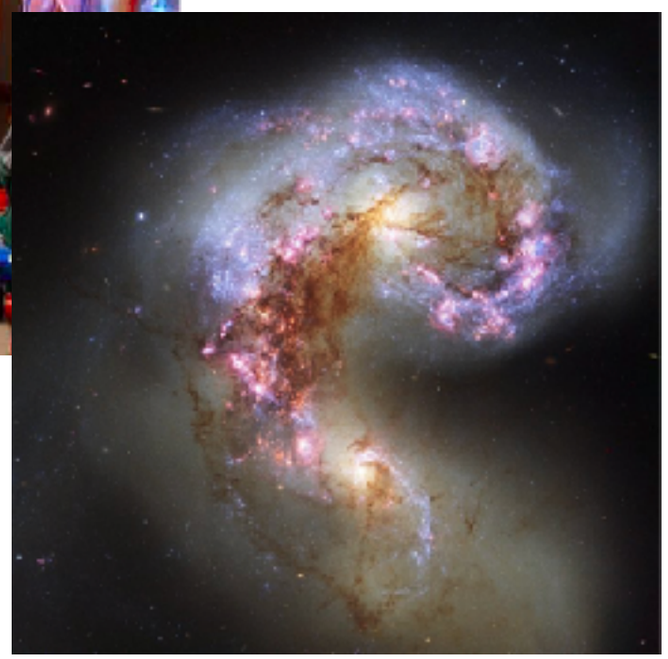
VS.

The second law of thermodynamics

$$S = k \log W$$







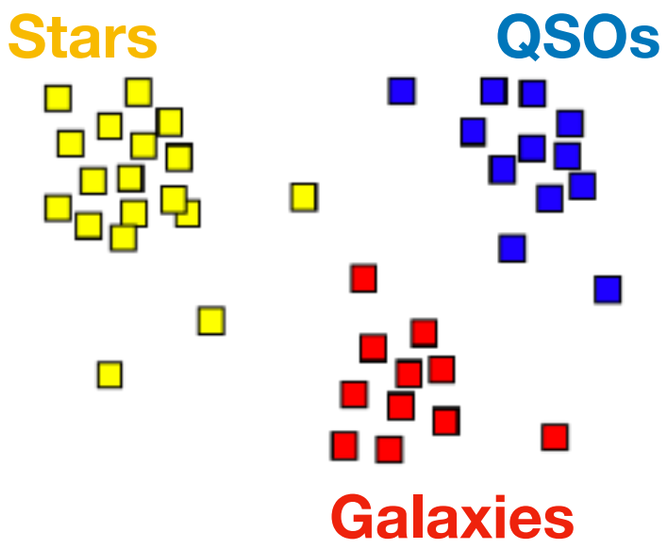
Science is about compression.

Obtaining knowledge = decreasing entropy.

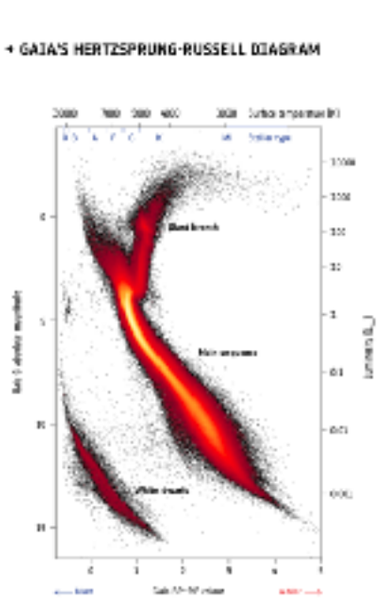
Exercise: ask Natalie about a type Ia supernova.

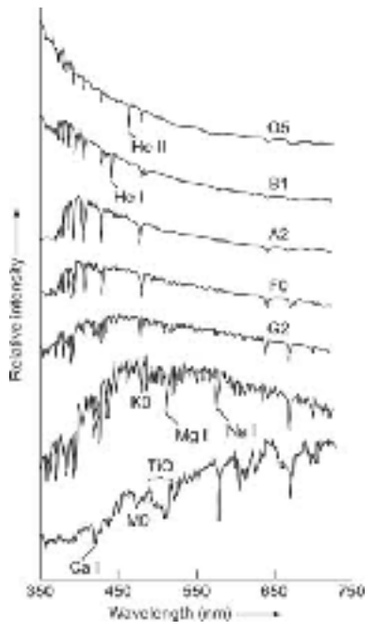
$$G_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

Clustering

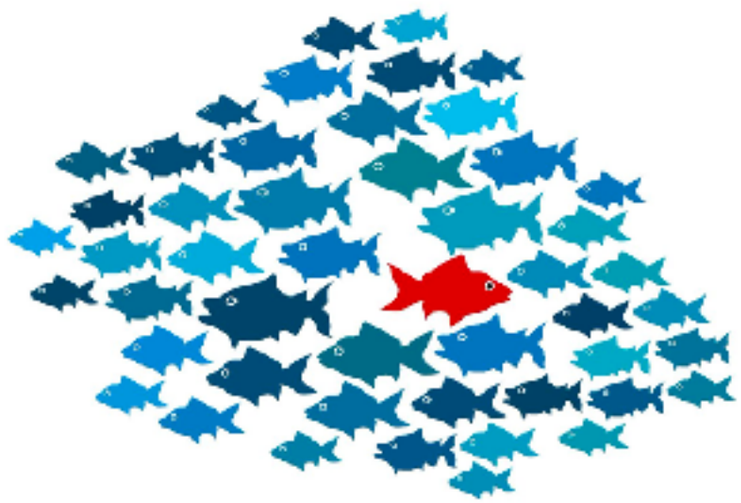


Pattern Recognition & Dimensionality Reduction





Outlier Detection



Anatomy of unsupervised learning algorithms

$$f(\overrightarrow{X}, \{a_1, a_2, ...\}) = \overrightarrow{y}$$

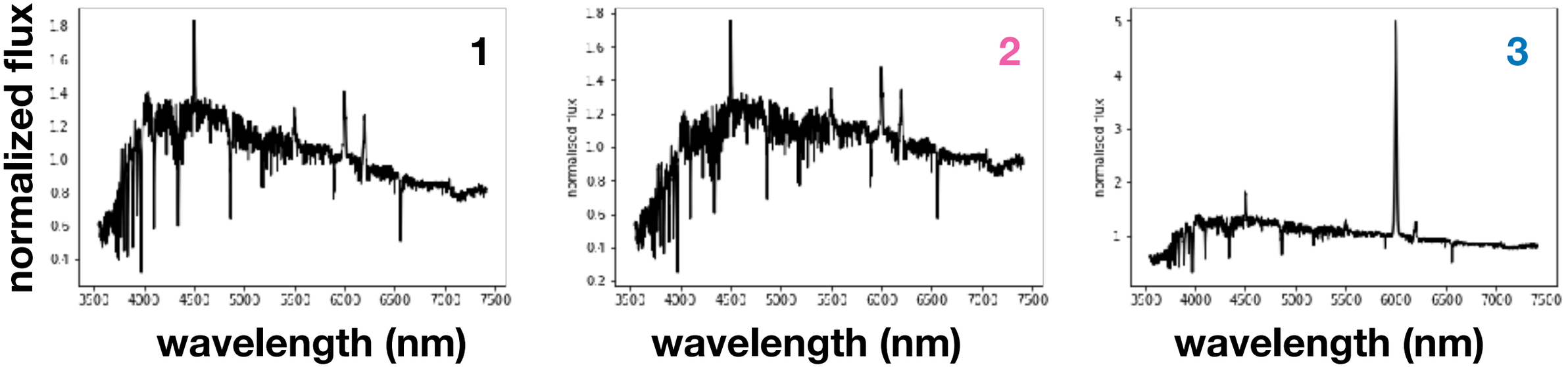
Anatomy of unsupervised learning algorithms

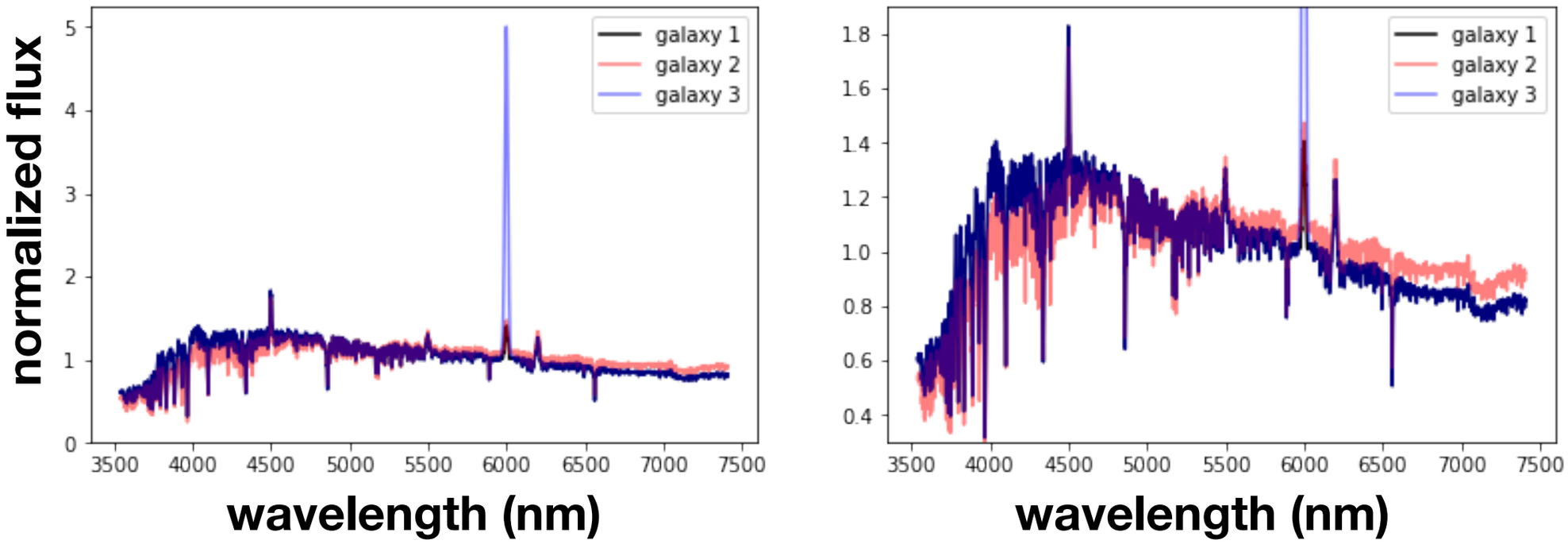
$$f(\overrightarrow{X}, \{a_1, a_2, ...\}) = \overrightarrow{y}$$

Input dataset:

- Raw data (spectra, images, light-curves).
- Extracted features.
- Measured relations between different objects (distances, correlations).

Why should we measure distances between objects?





Anatomy of unsupervised learning algorithms

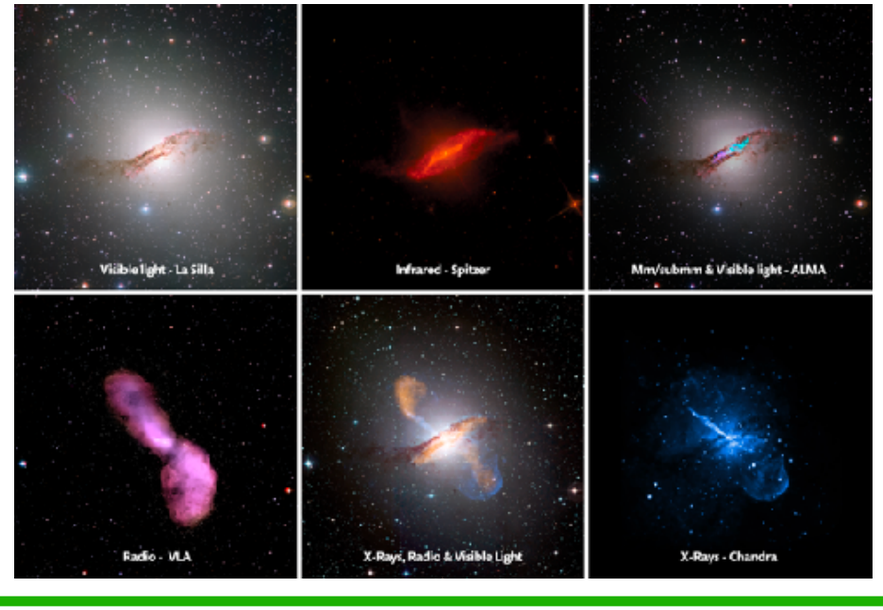
$$f(\vec{X}, \{a_1, a_2, ...\}) = \vec{y}$$

Input dataset:

- Raw data (spectra, images, light-curves).
- Extracted features.
- Measured relations between different objects (distances, correlations).

Hyperparameters

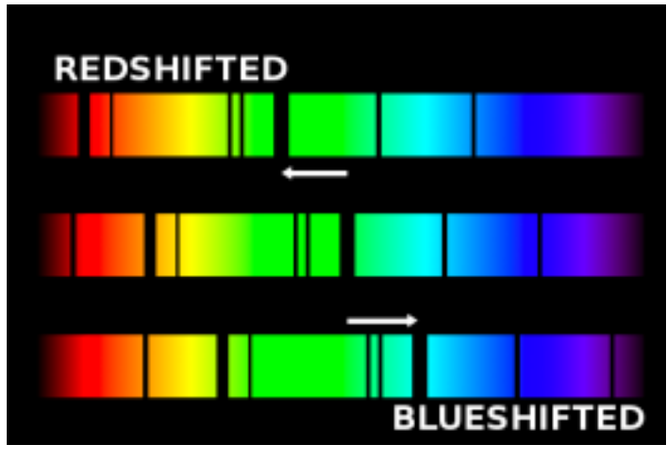
- Tuning parameters of the algorithm.
- Can strongly affect the result.
- Traditionally, cannot be optimized for.



Anatomy of unsupervised learning algorithms

Internal choices and/or internal cost function

- Usually, we cannot control these.
- Strongly affect the result, and define the range of possible outputs.



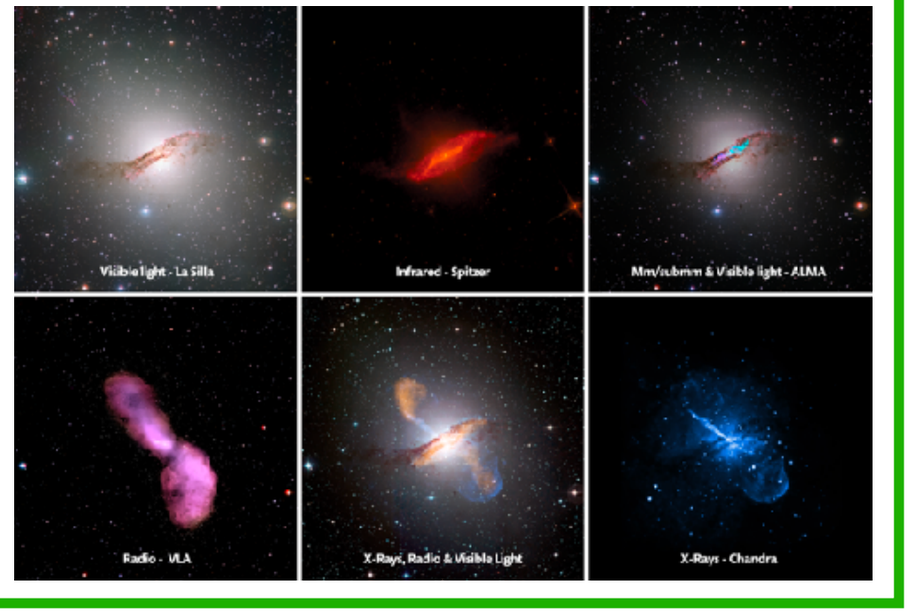
$$f(\vec{X}, \{a_1, a_2, ...\}) = \vec{y}$$

Input dataset:

- Raw data (spectra, images, light-curves).
- Extracted features.
- Measured relations between different objects (distances, correlations).

Hyperparameters

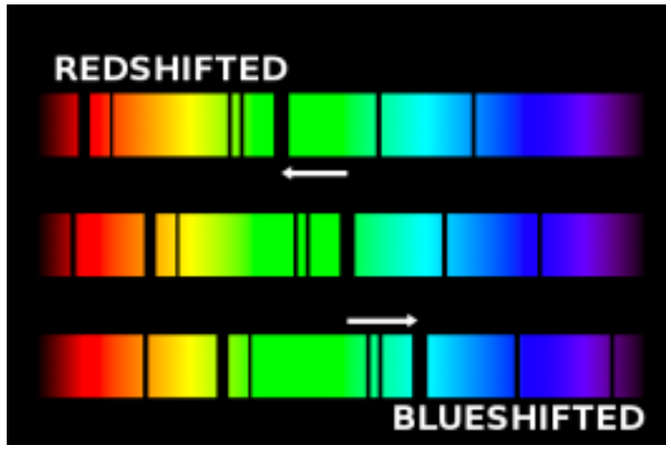
- Tuning parameters of the algorithm.
- Can strongly affect the result.
- Traditionally, cannot be optimized for.



Anatomy of unsupervised learning algorithms

Internal choices and/or internal cost function

- Usually, we cannot control these.
- Strongly affect the result, and define the range of possible outputs.



Algorithm output: clusters, high-dimensional relations, outliers, sparse representation.

Our goal:
exploration or **inference**

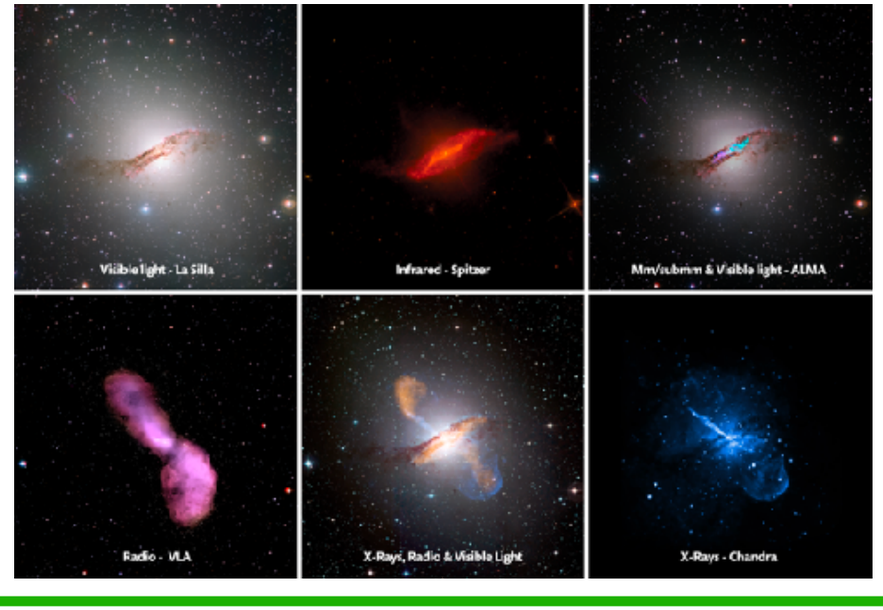
$$f(\vec{X}, \{a_1, a_2, ...\}) = \vec{y}$$

Input dataset:

- Raw data (spectra, images, light-curves).
- Extracted features.
- Measured relations between different objects (distances, correlations).

Hyperparameters

- Tuning parameters of the algorithm.
- Can strongly affect the result.
- Traditionally, cannot be optimized for.



Good Practices

- **Start simple:**
  - Simulate simple low-dimensional dataset, without noise, where the output can be anticipated.
  - Compare the output of the algorithm for different data representations and different choices of hyper-parameters.
- **Gradually complicate the model:**
  - Add more dimensions (some of them should be uninformative).
  - Add noise.
  - Compare the output for different representations and hyper-parameters.
- **Physically-motivated model:**
  - Simulate a physically-motivated dataset.
  - Experiment with different noise properties, different representations, and hyper-parameters.
- **Try to break the algorithm.**

Clustering Algorithms:

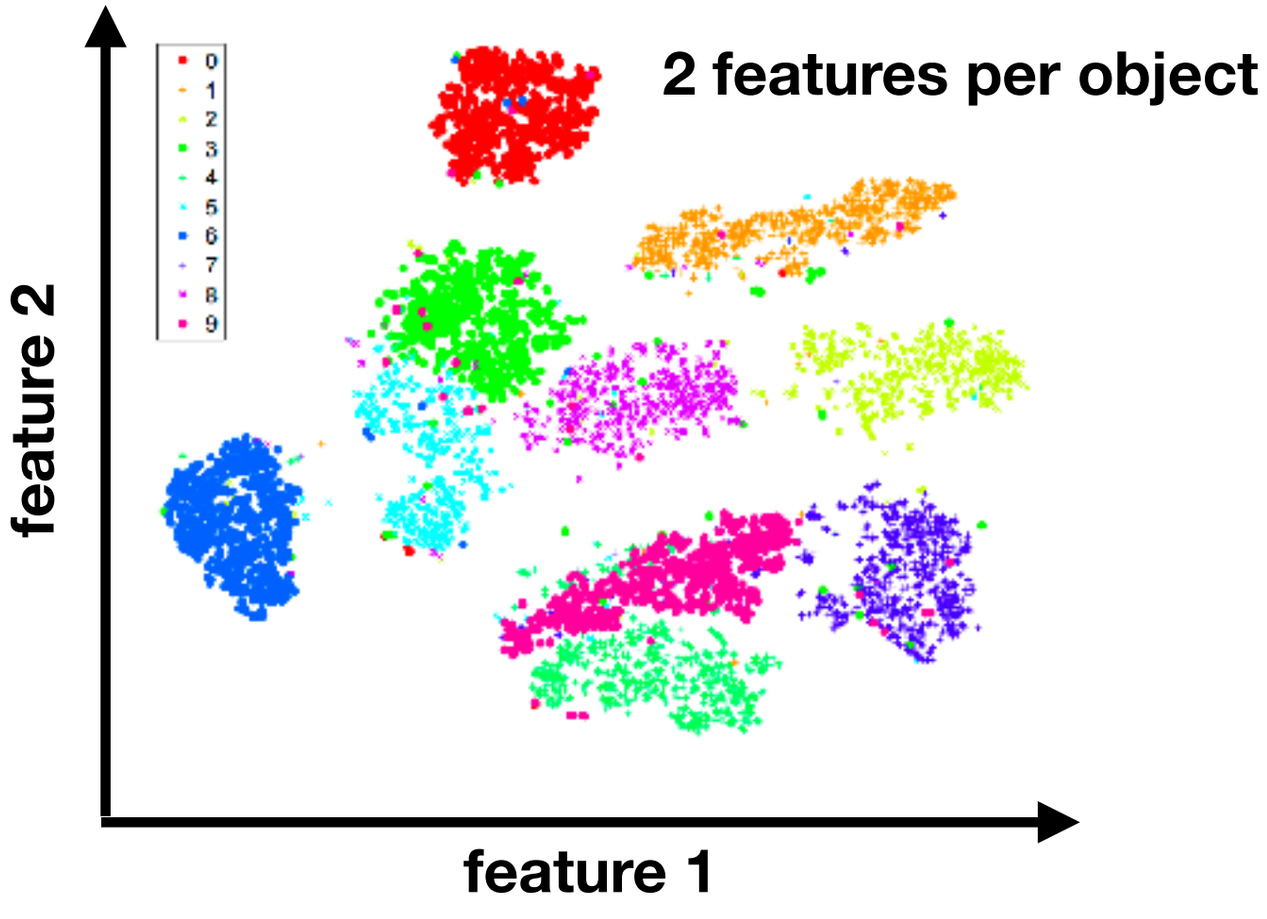
K-means and Hierarchical Clustering

Dimensionality Reduction

What is Dimensionality Reduction?

28 x 28 features per object

Dimensionality Reduction algorithm



2 features per object

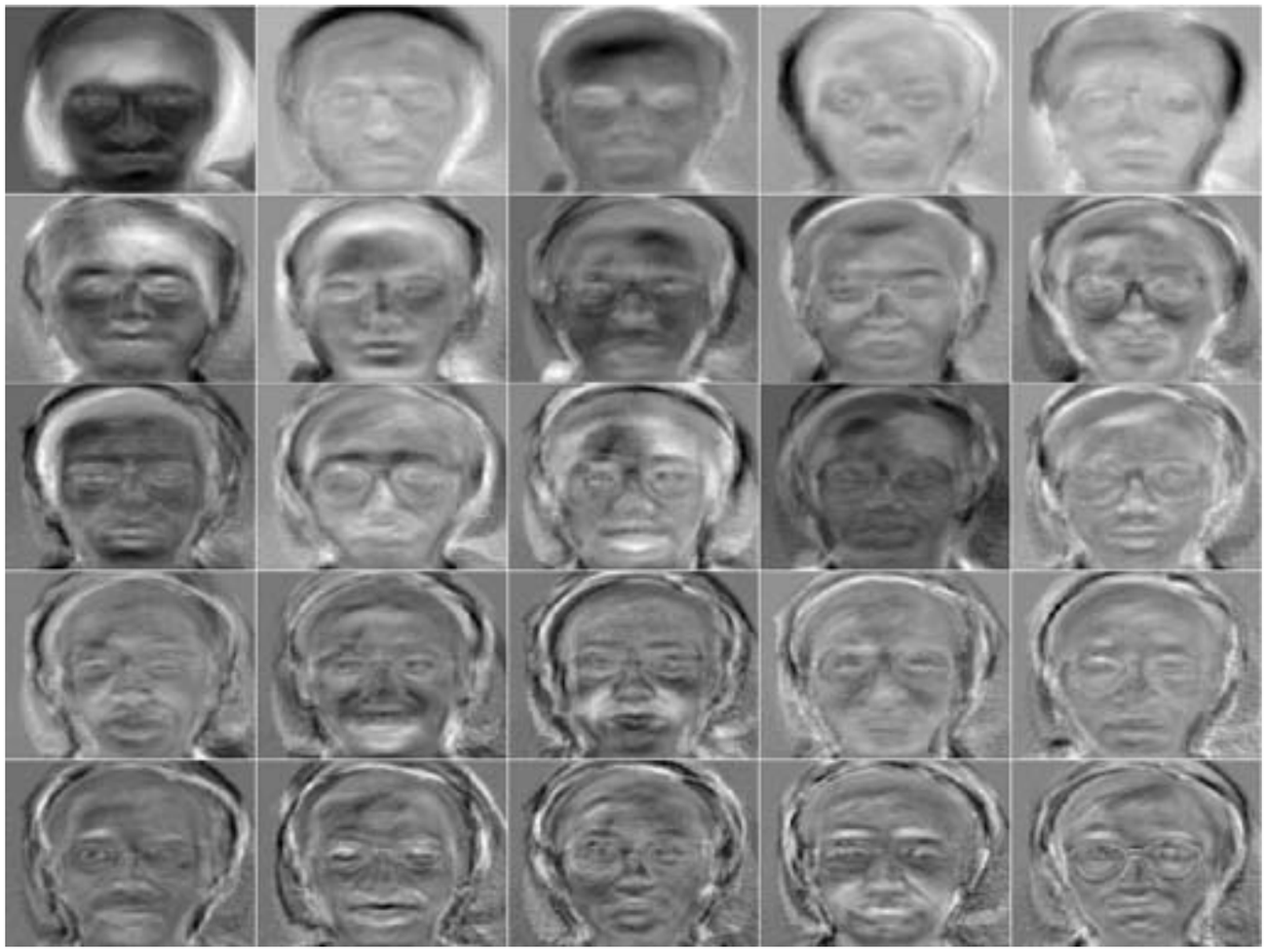
Why do we need dimensionality reduction?

- **“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.

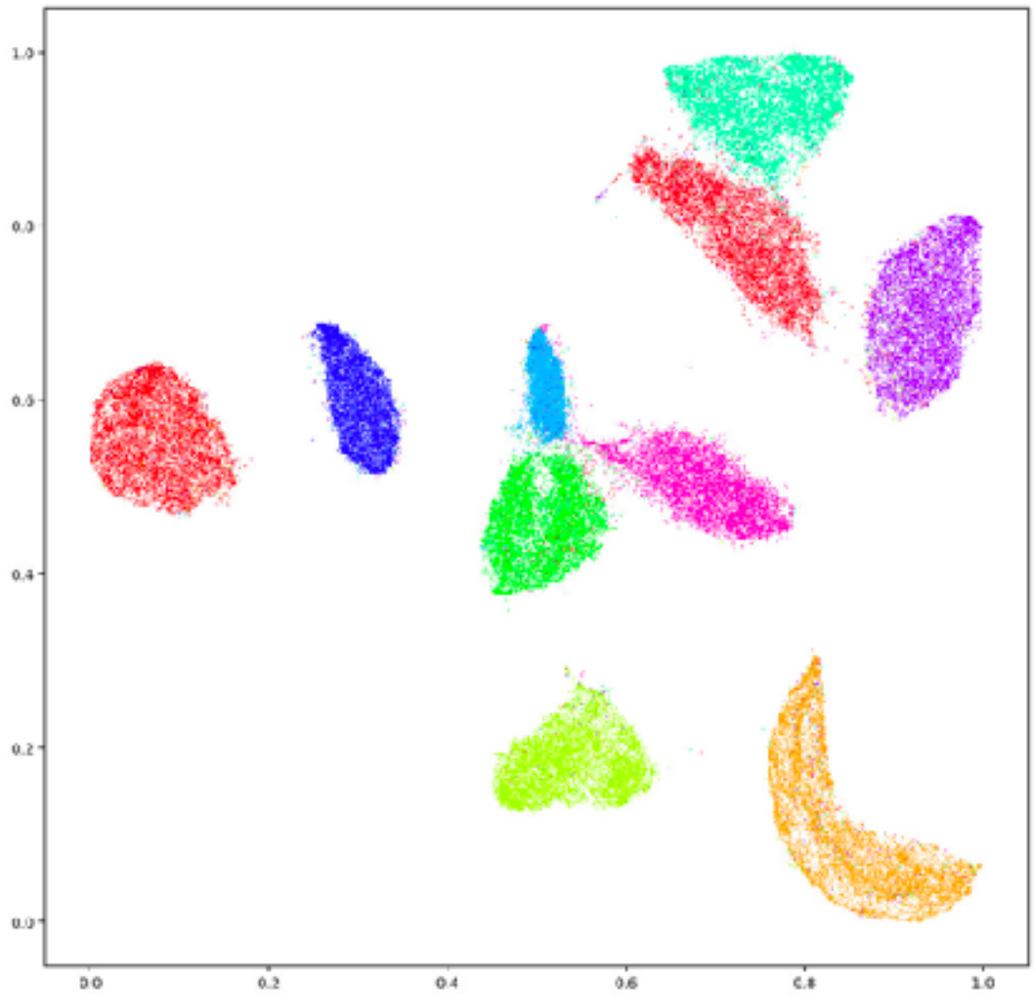
We gain: prototypes that represent the population **and** low-dimensional embedding.



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.

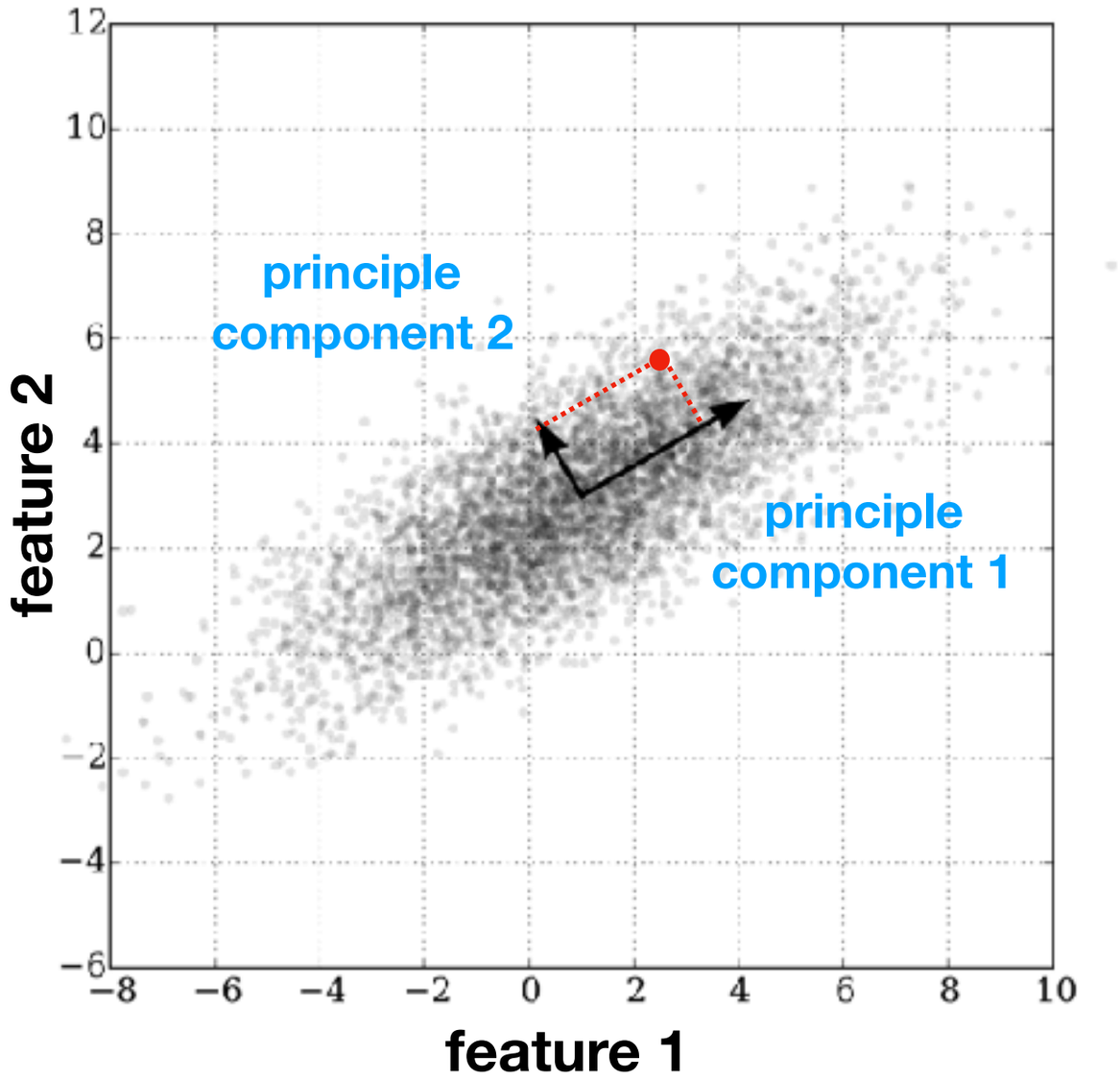


For example: tSNE, UMAP, and autoencoders.

Principle Component Analysis (PCA)

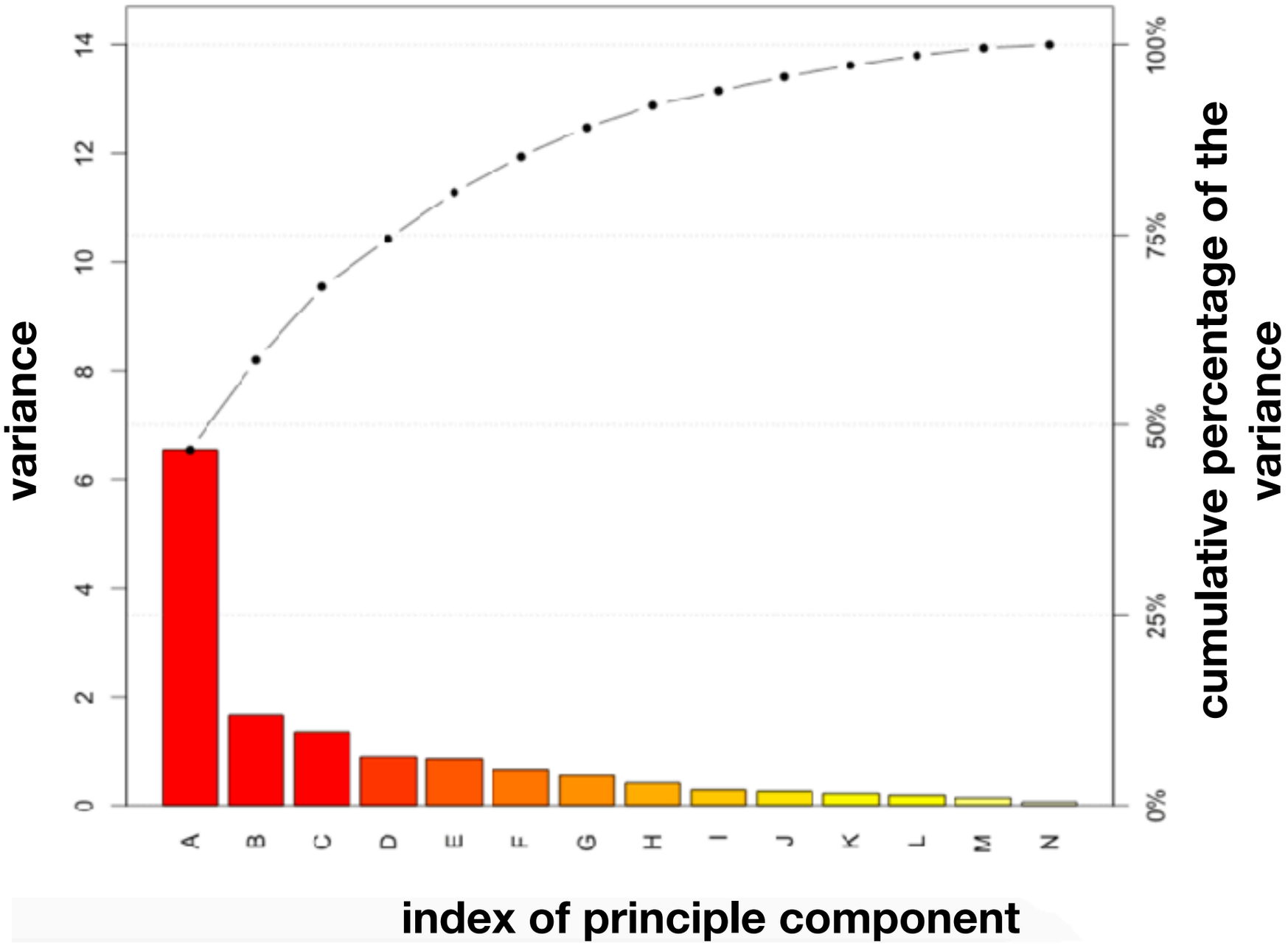
PCA is a transformation that converts a set of observations (possibly from correlated variables) into a set of values of linearly uncorrelated variables, called **principle components.**

- 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)

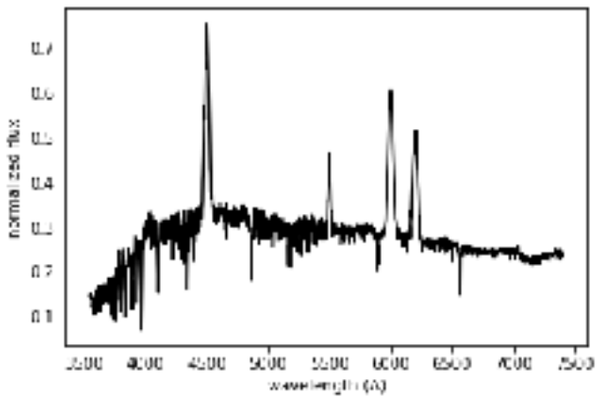
PCA allows us to compress the data, by representing each object as a projection on the first principle components.



Principle Component Analysis (PCA)

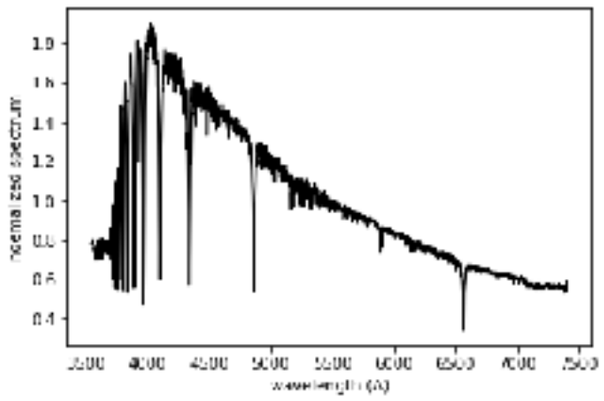
The principle components **may** represent the true **building blocks** of the objects in our dataset.

observed object



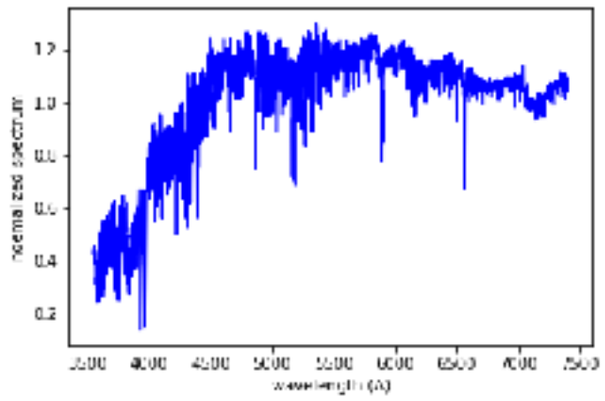
= A *

principle comp. 1



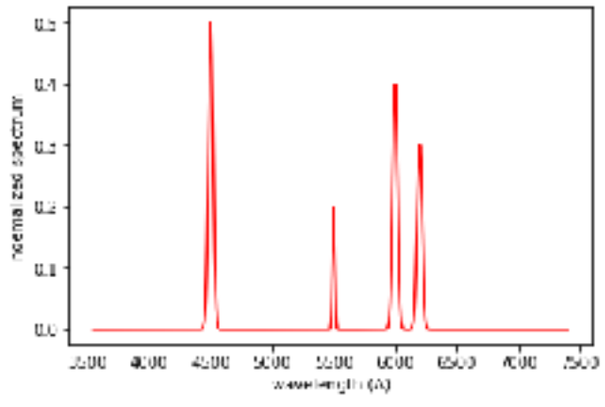
+ B *

principle comp. 2



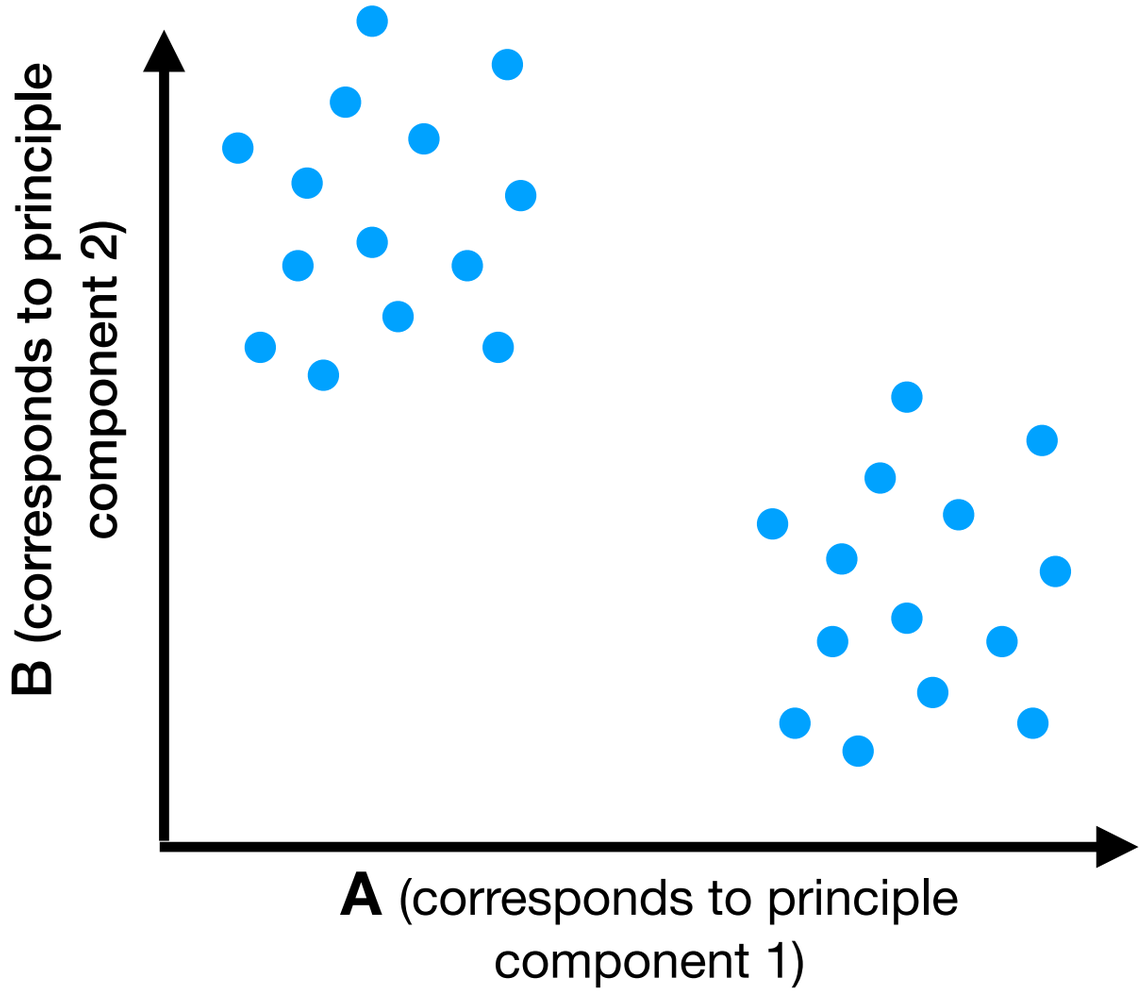
principle comp. 3

+ C *

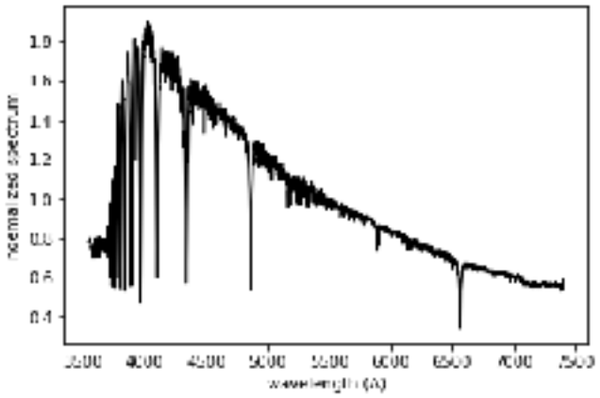


Principle Component Analysis (PCA)

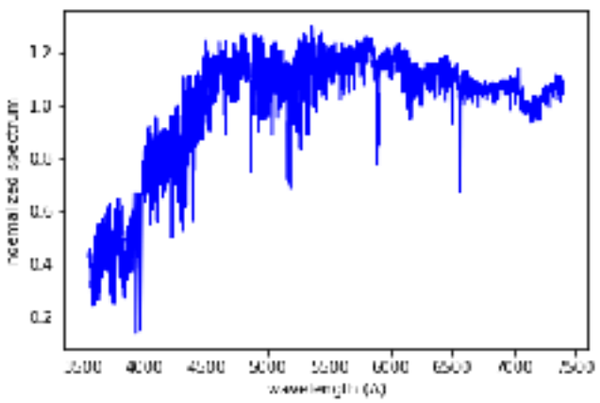
The projection onto the principle components gives a low-dimensional representation of the objects in the sample.

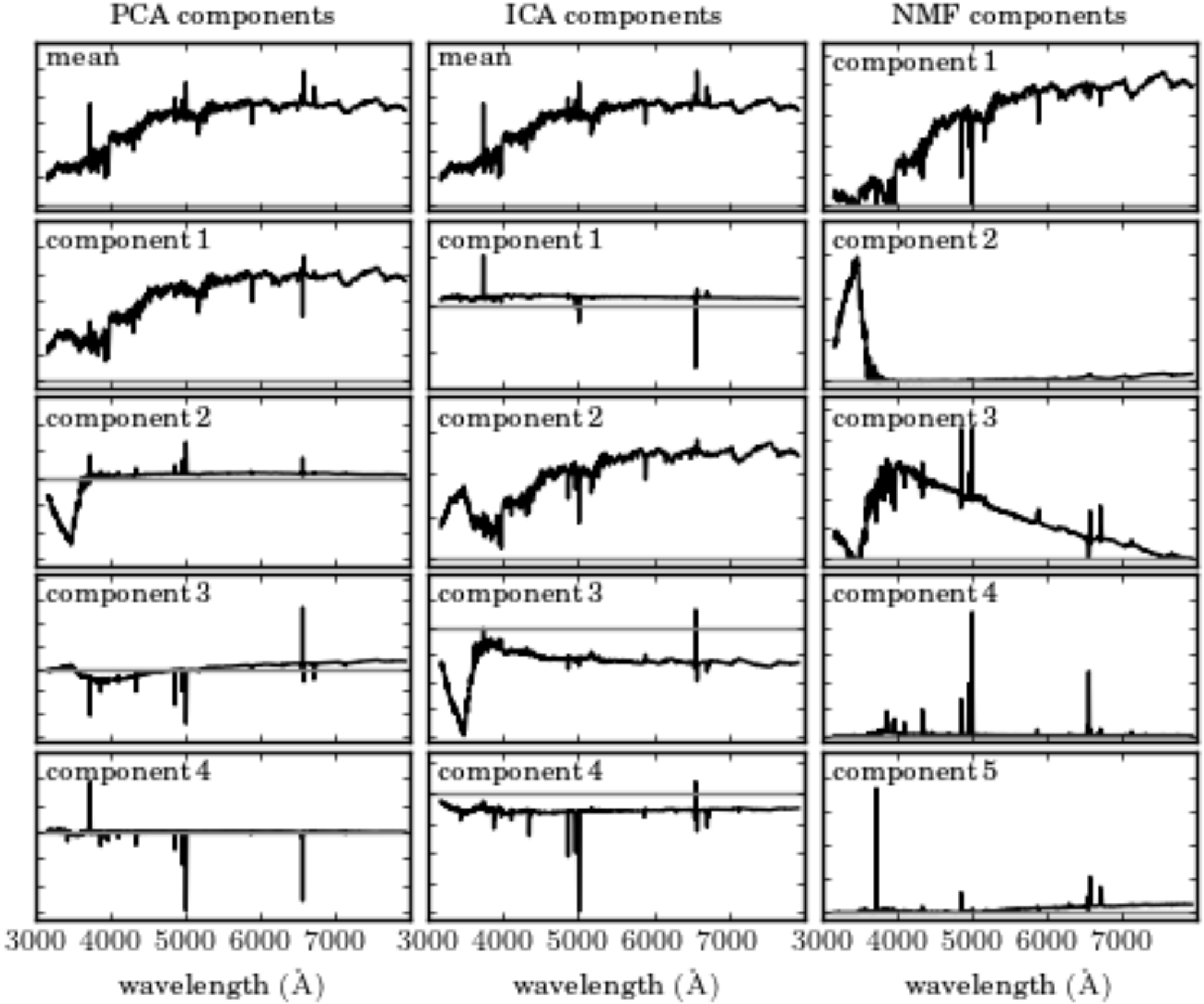


principle comp. 1



principle comp. 2





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

t-distributed stochastic neighbor embedding

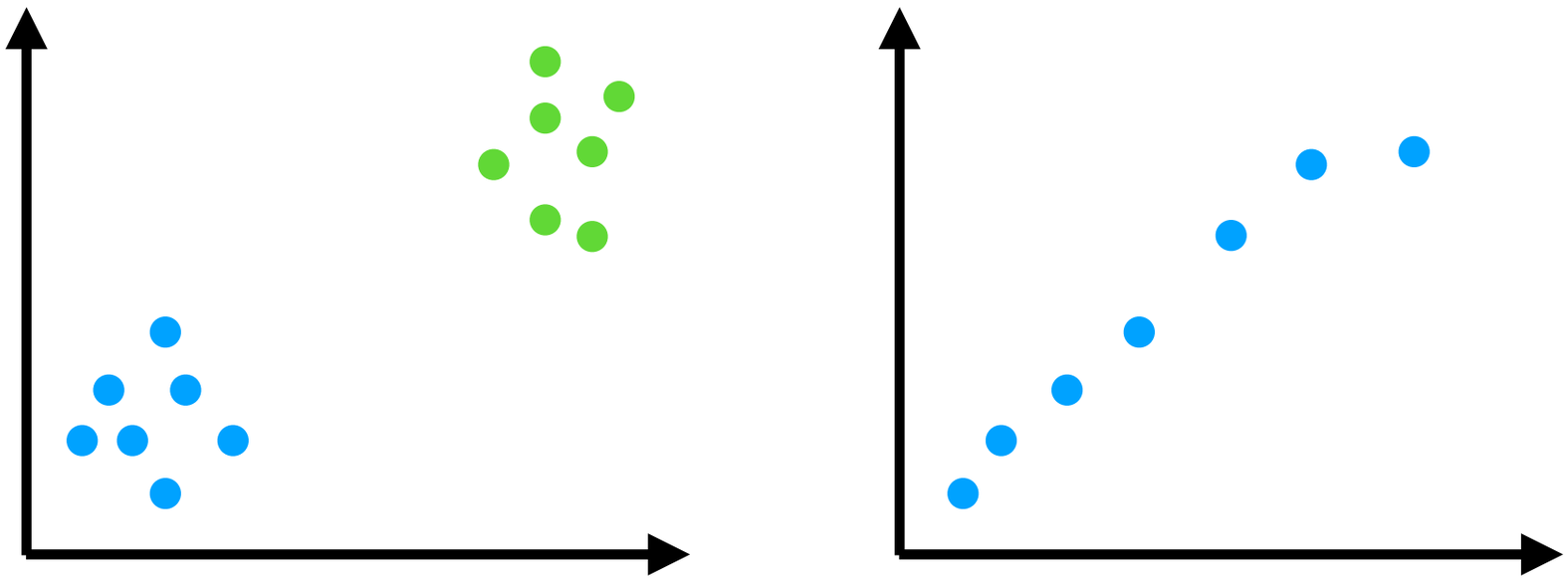
(tSNE)

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**

high-dimensional space:

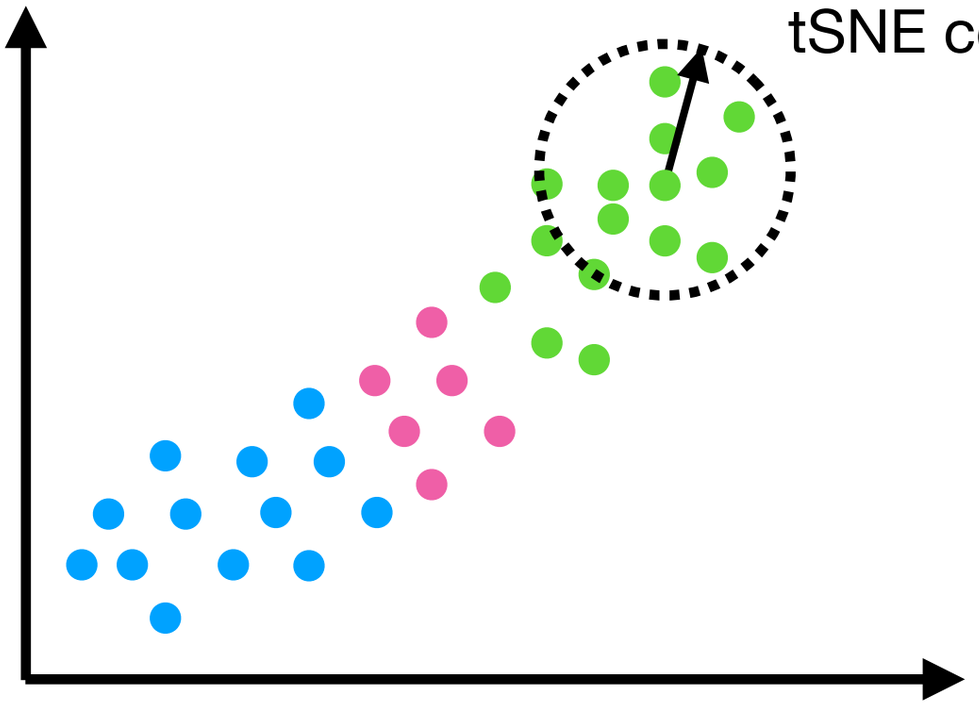


low-dimensional space:

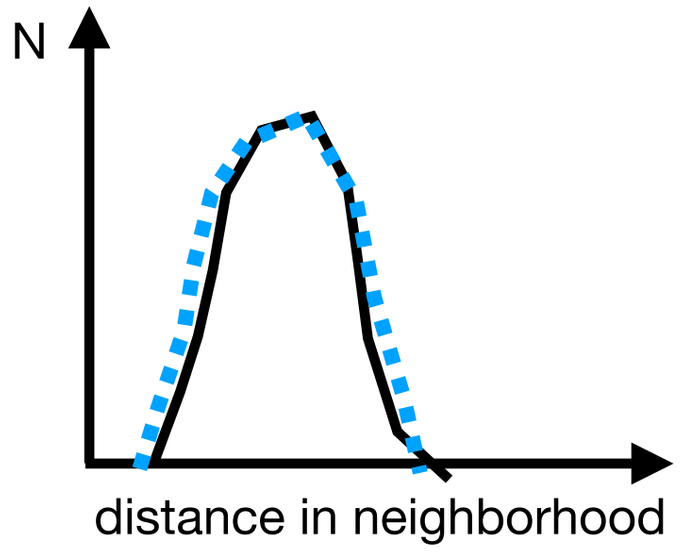
tSNE

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

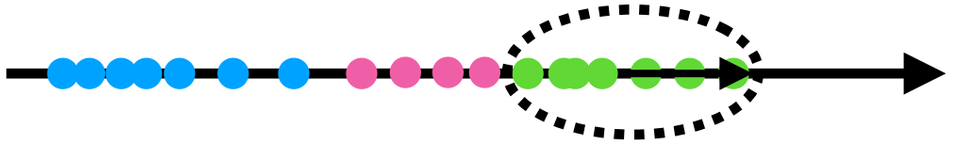
high-dimensional space:



perplexity: the neighborhood that tSNE considers in optimization

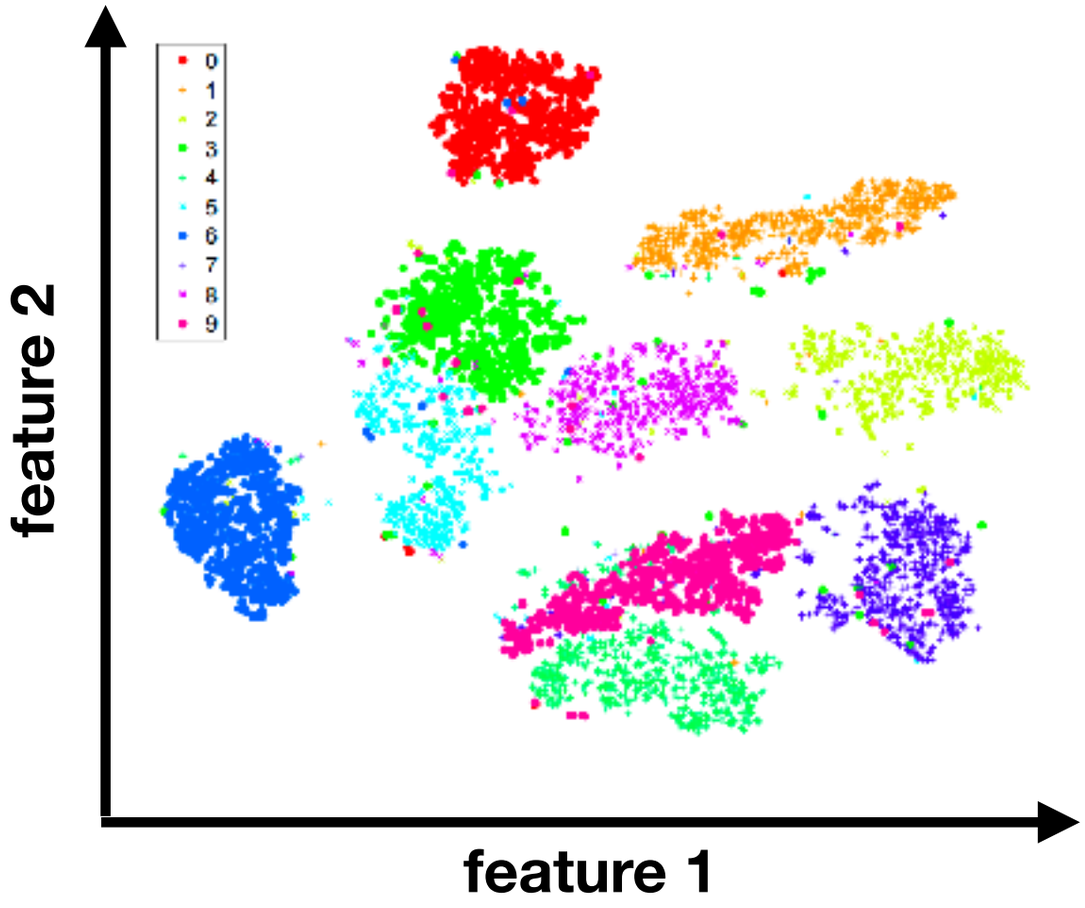


low-dimensional space:



tSNE - example

28 x 28 features per object



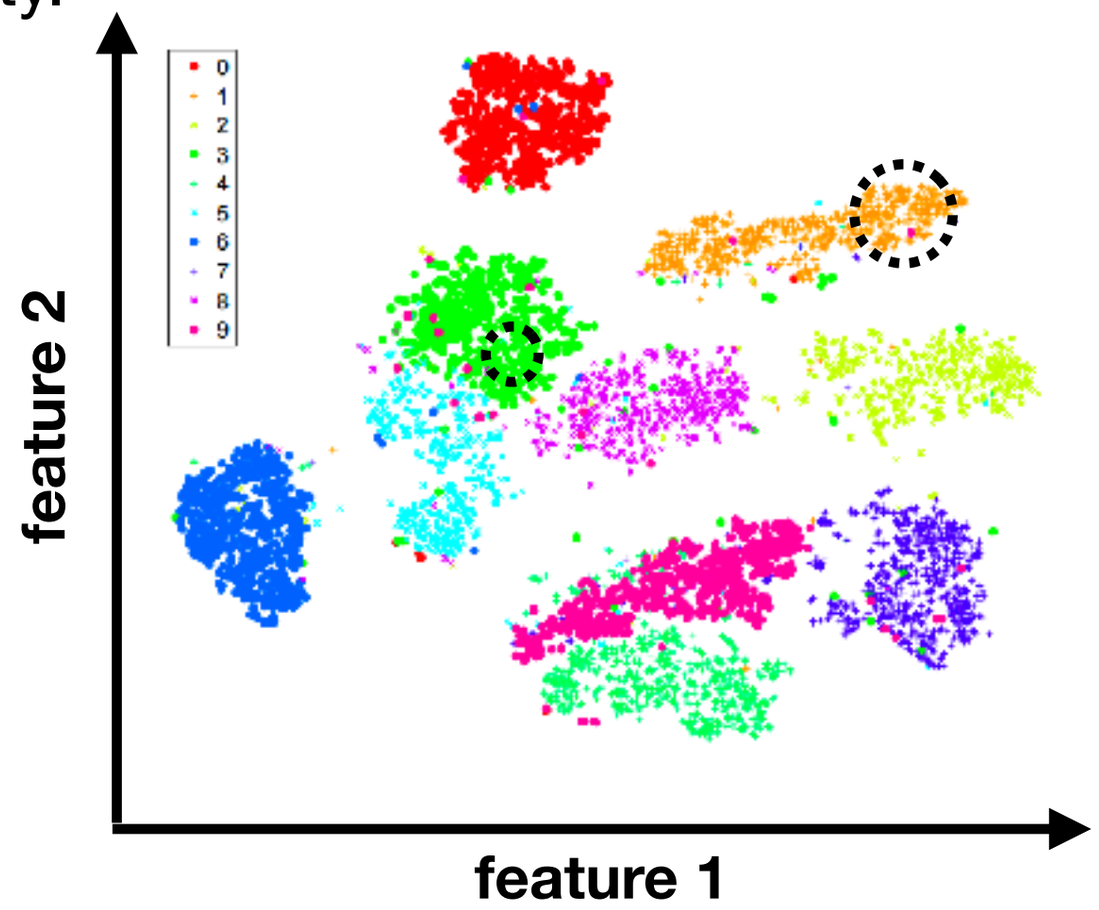
tSNE - example

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

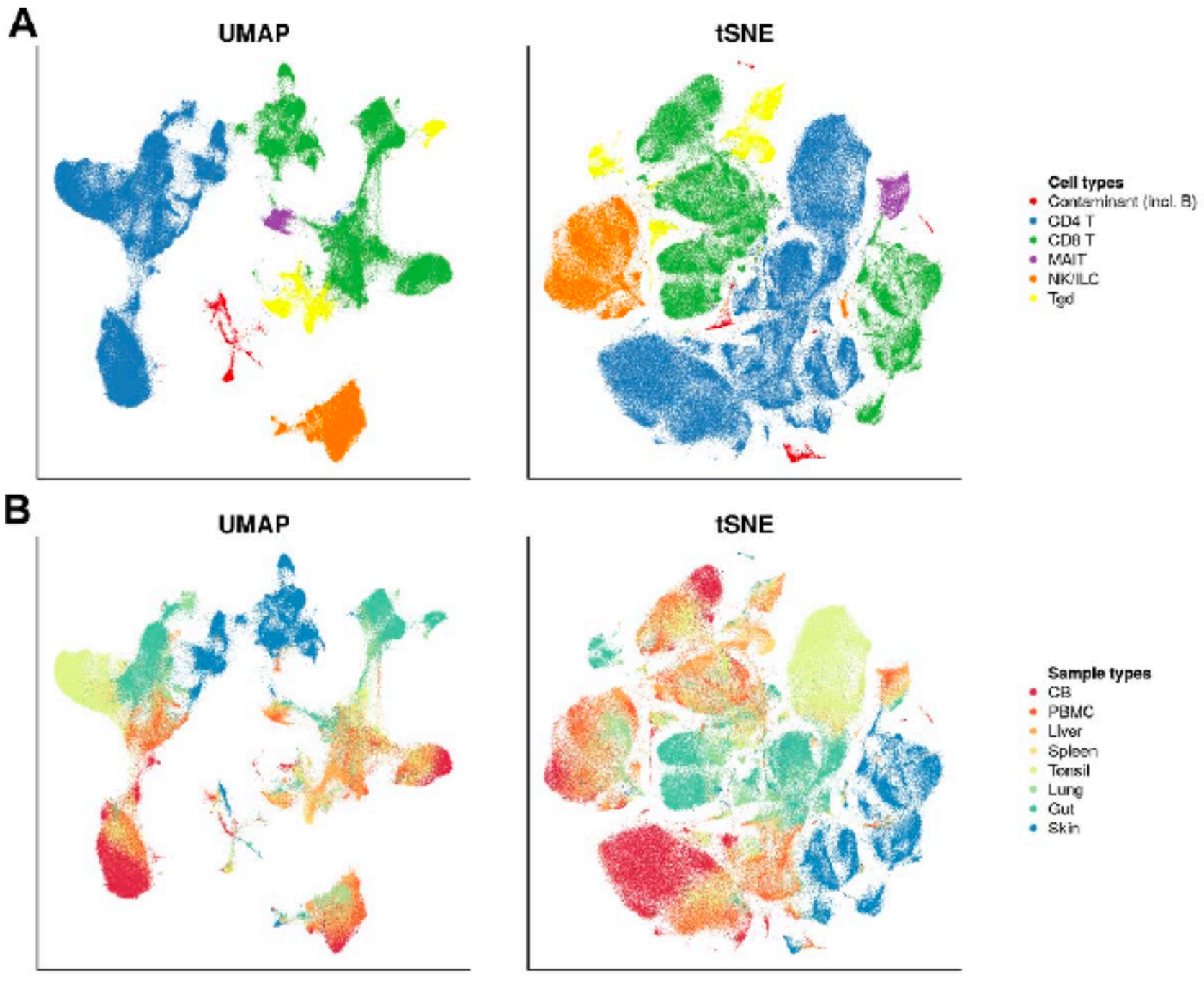
tSNE : Pros & Cons

- **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

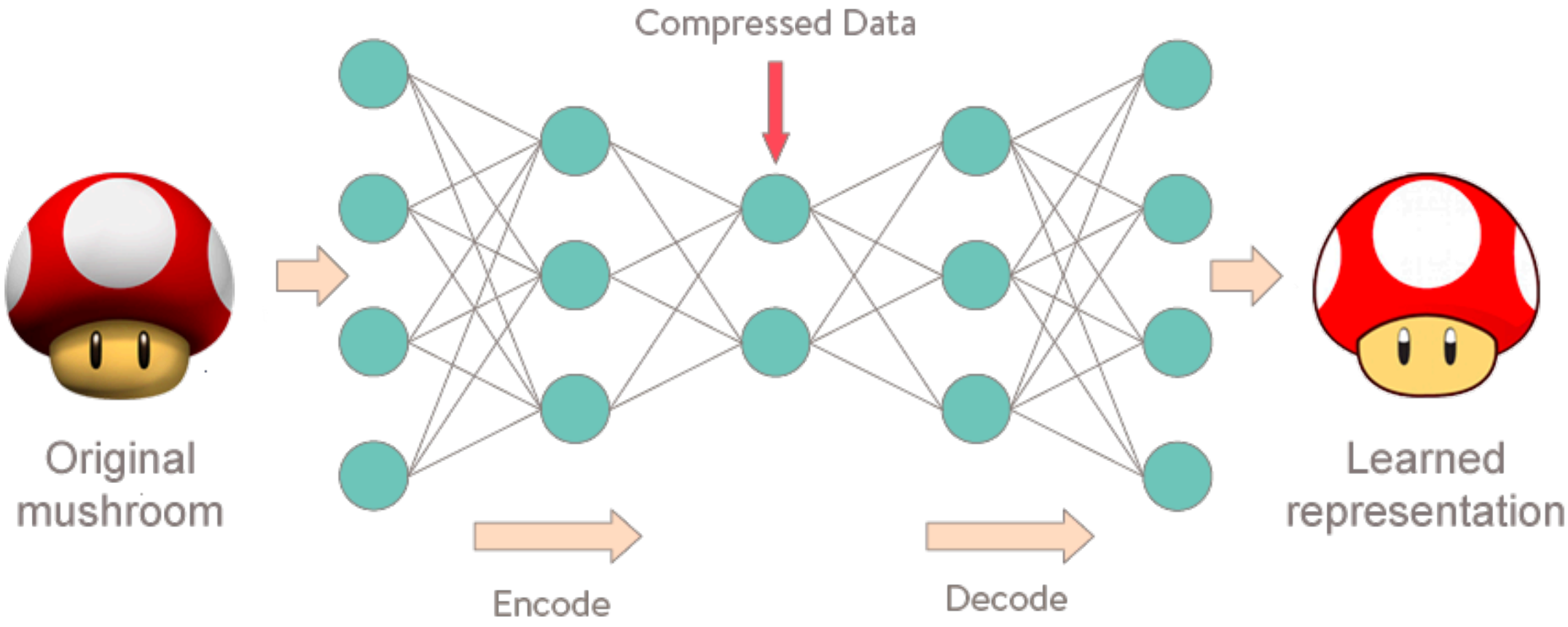


UMAP



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

Autoencoders



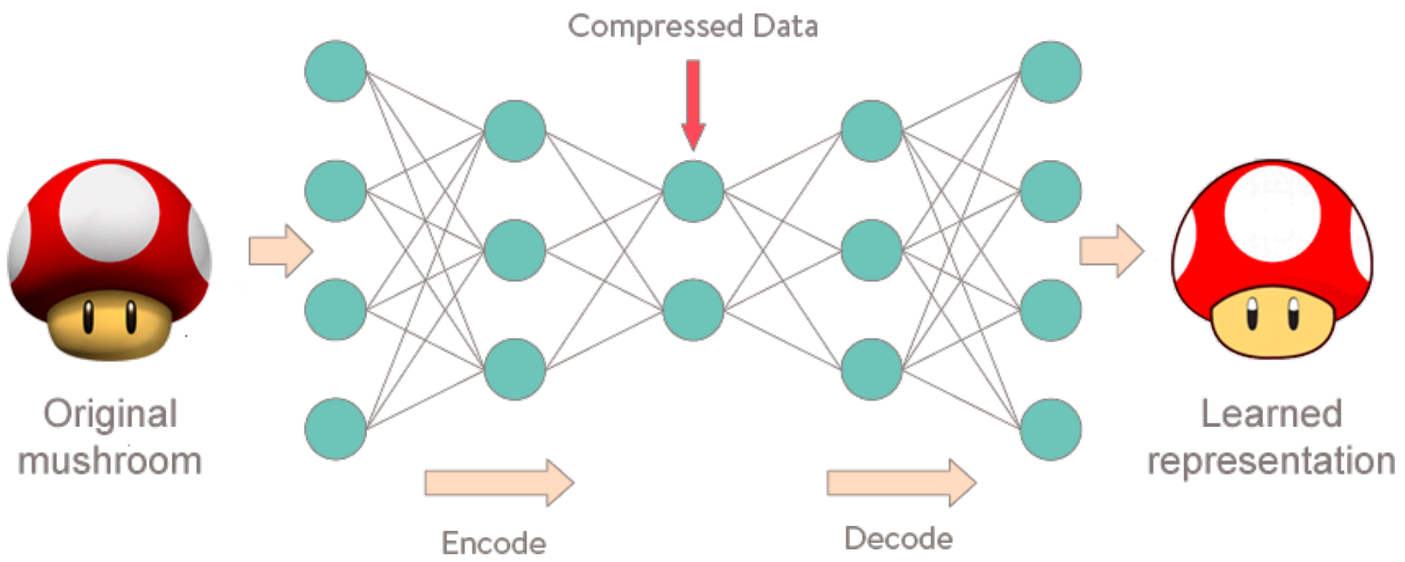
loss function = $($  $-$  $)^2$

Original mushroom

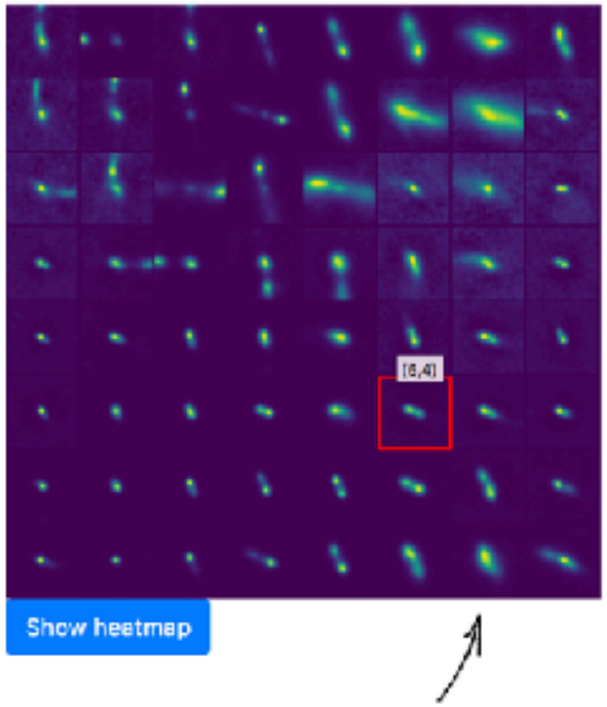
Learned representation

Autoencoders - Pros & Cons

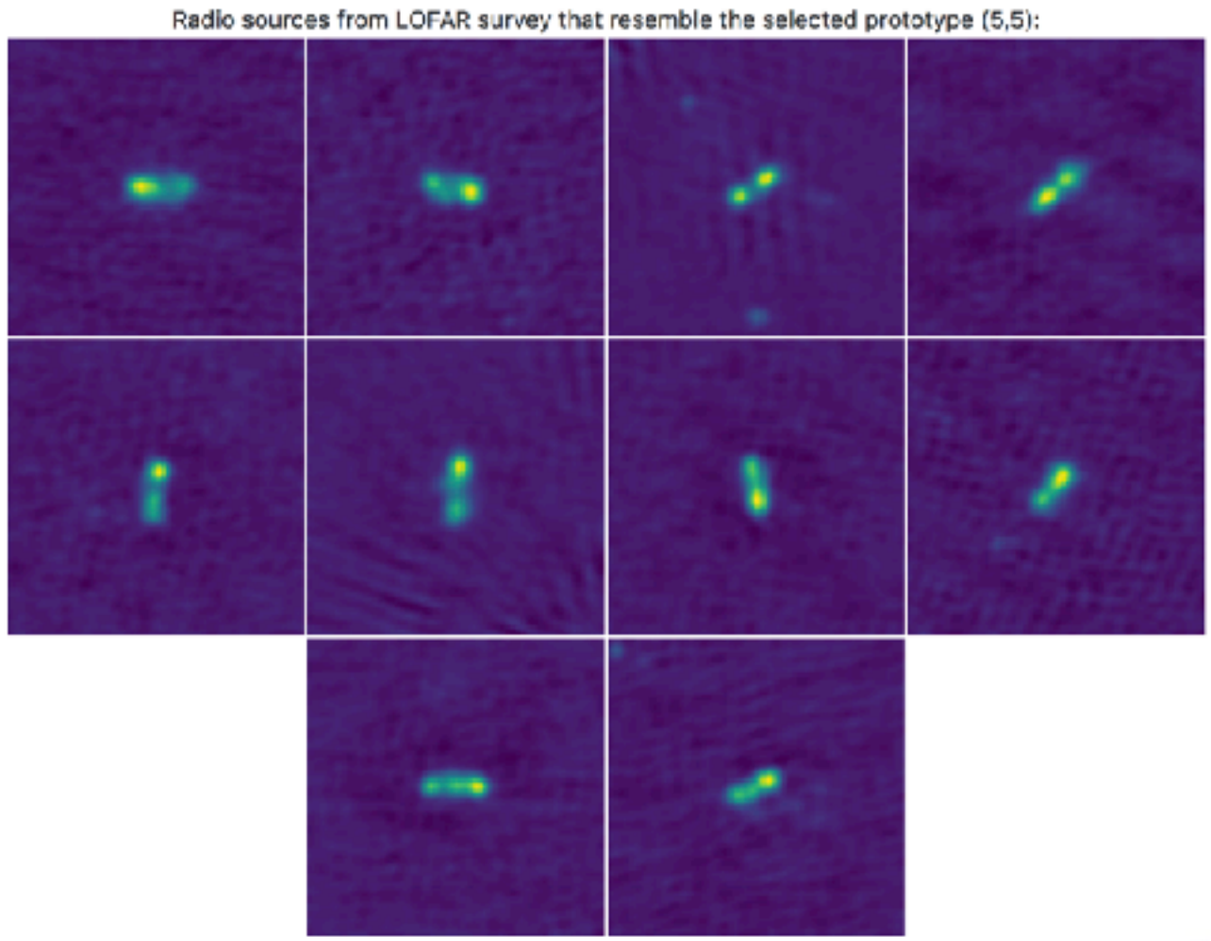
- **Advantages:**
  - Can reduce the dimensions of raw images (CNN) or time-series (RNN)!
  - Can be used to produce an uncertainty on the embedding.
- **Disadvantages:**
  - No prototypes.
  - Complexity and interpretability.

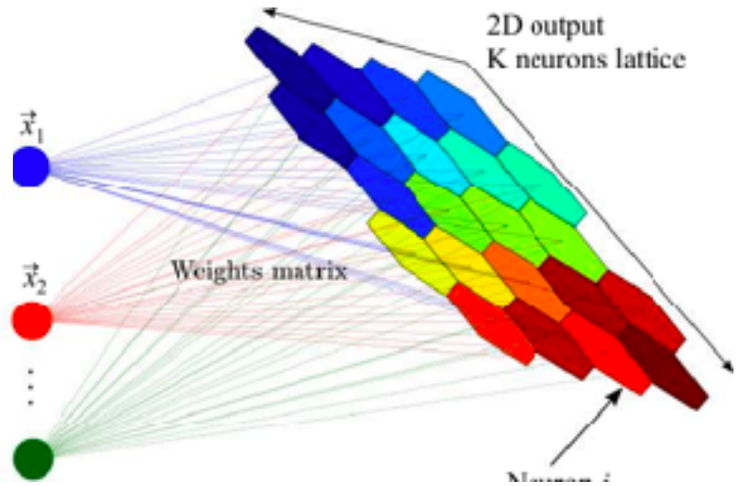


Self Organizing Maps (SOM) and PINK



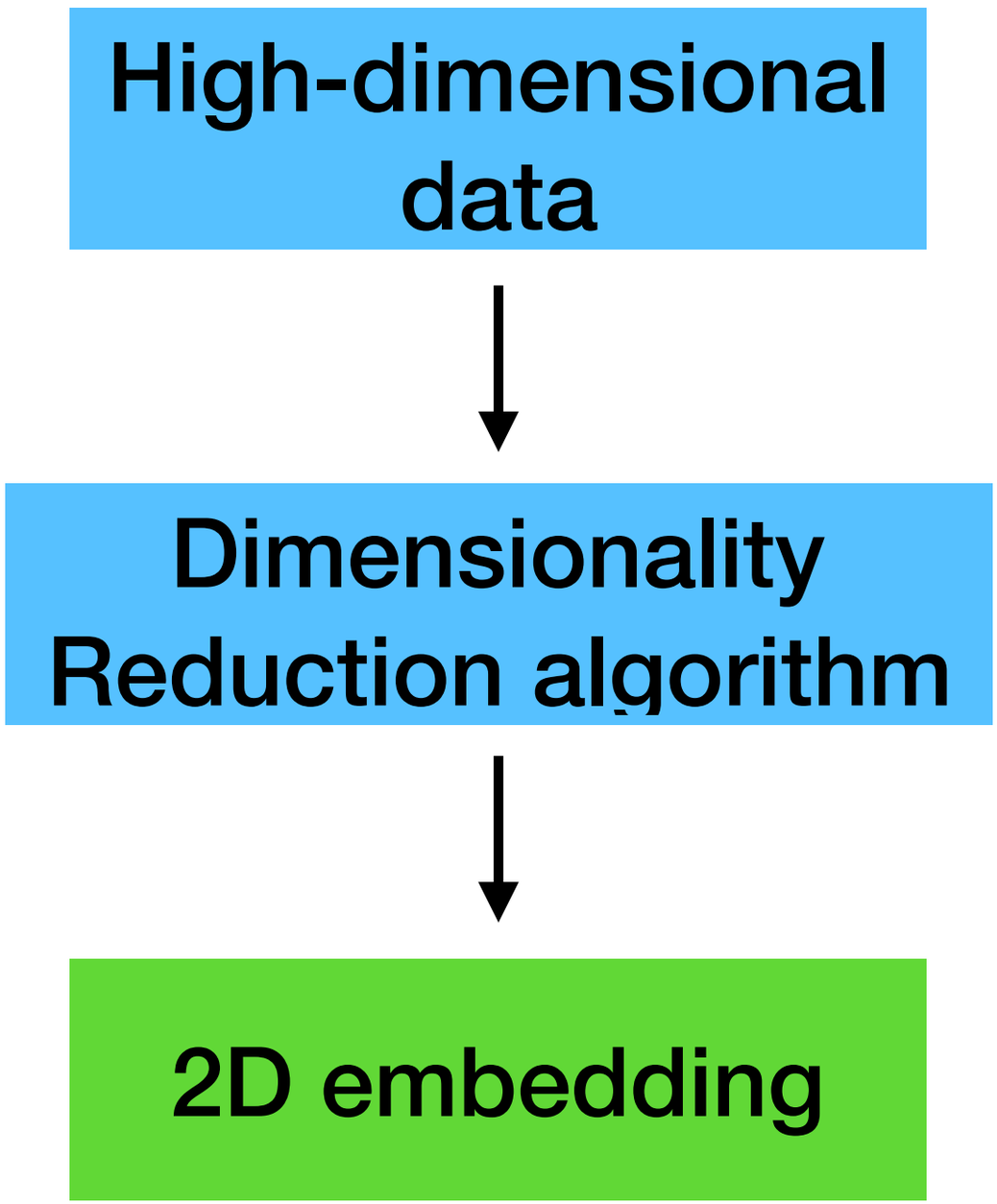
This is a Self-Organizing Map, trained on sources from the LOFAR survey. Click on one of these prototypes.





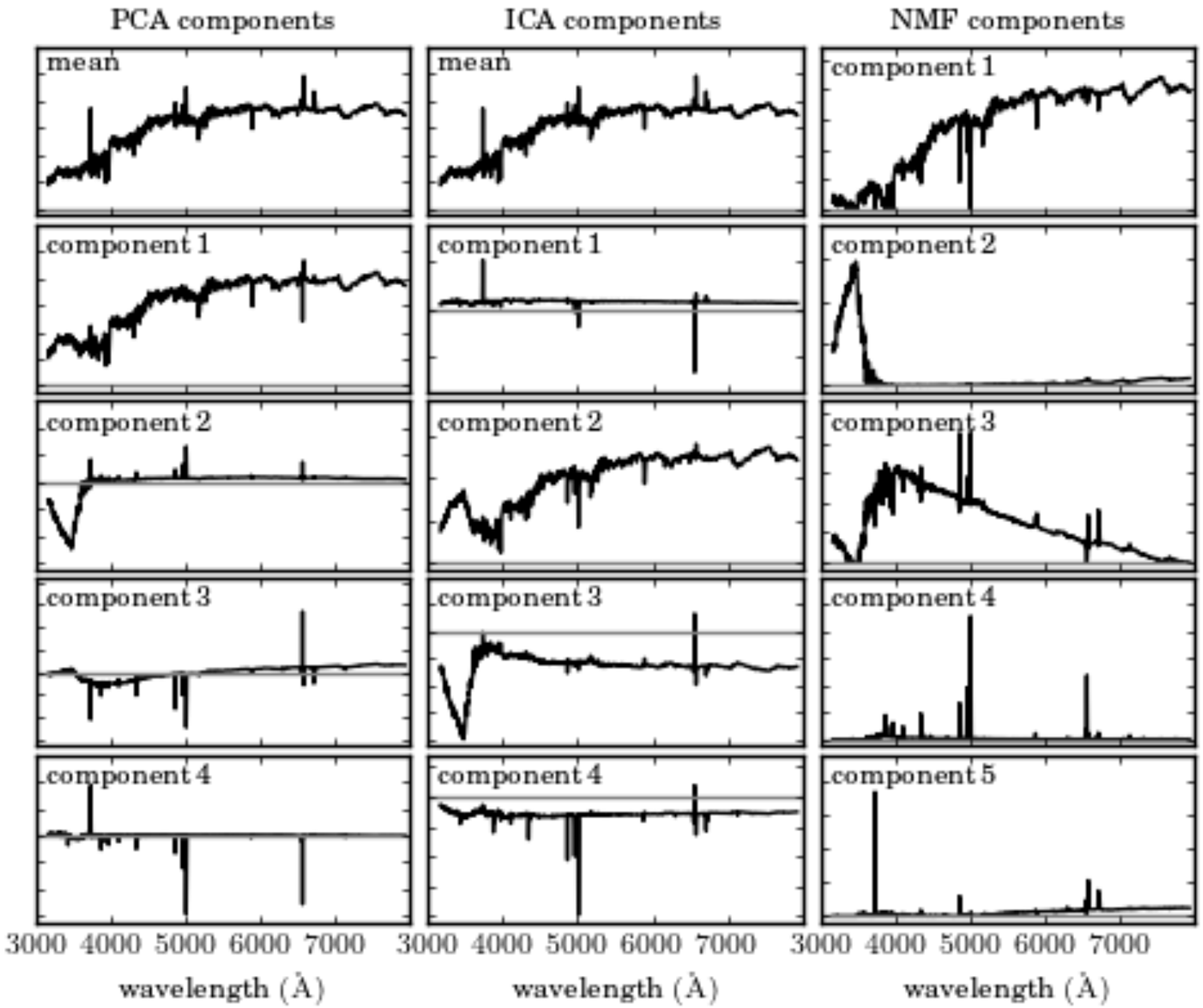
See: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-116.pdf
http://www.astron.nl/LifeCycle2018/Documents/Talks_Session1/Harwood_LifeCycle18.pdf

How to interpret the output of a dimensionality reduction algorithm?

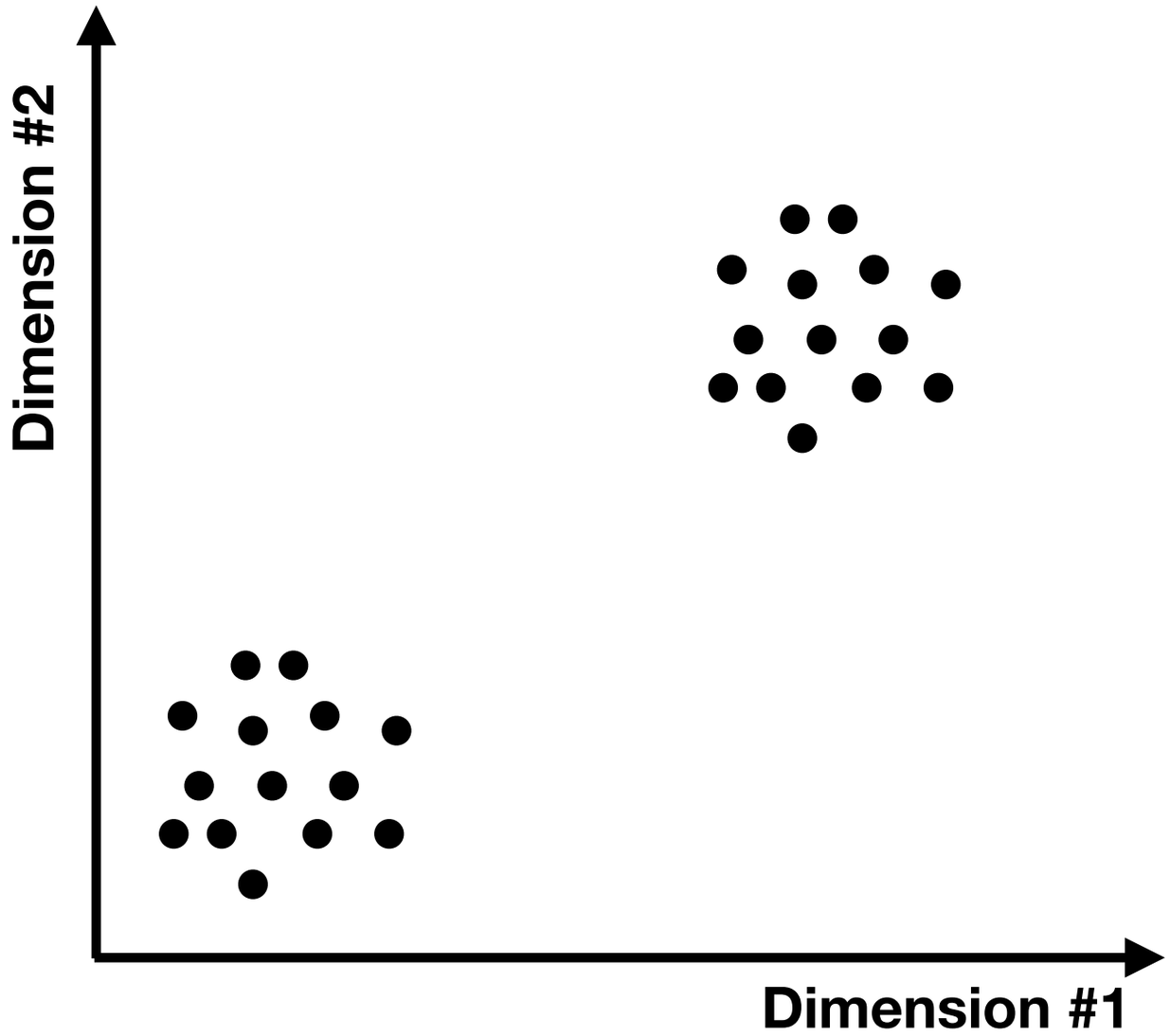


How to interpret the output of a dimensionality reduction algorithm?

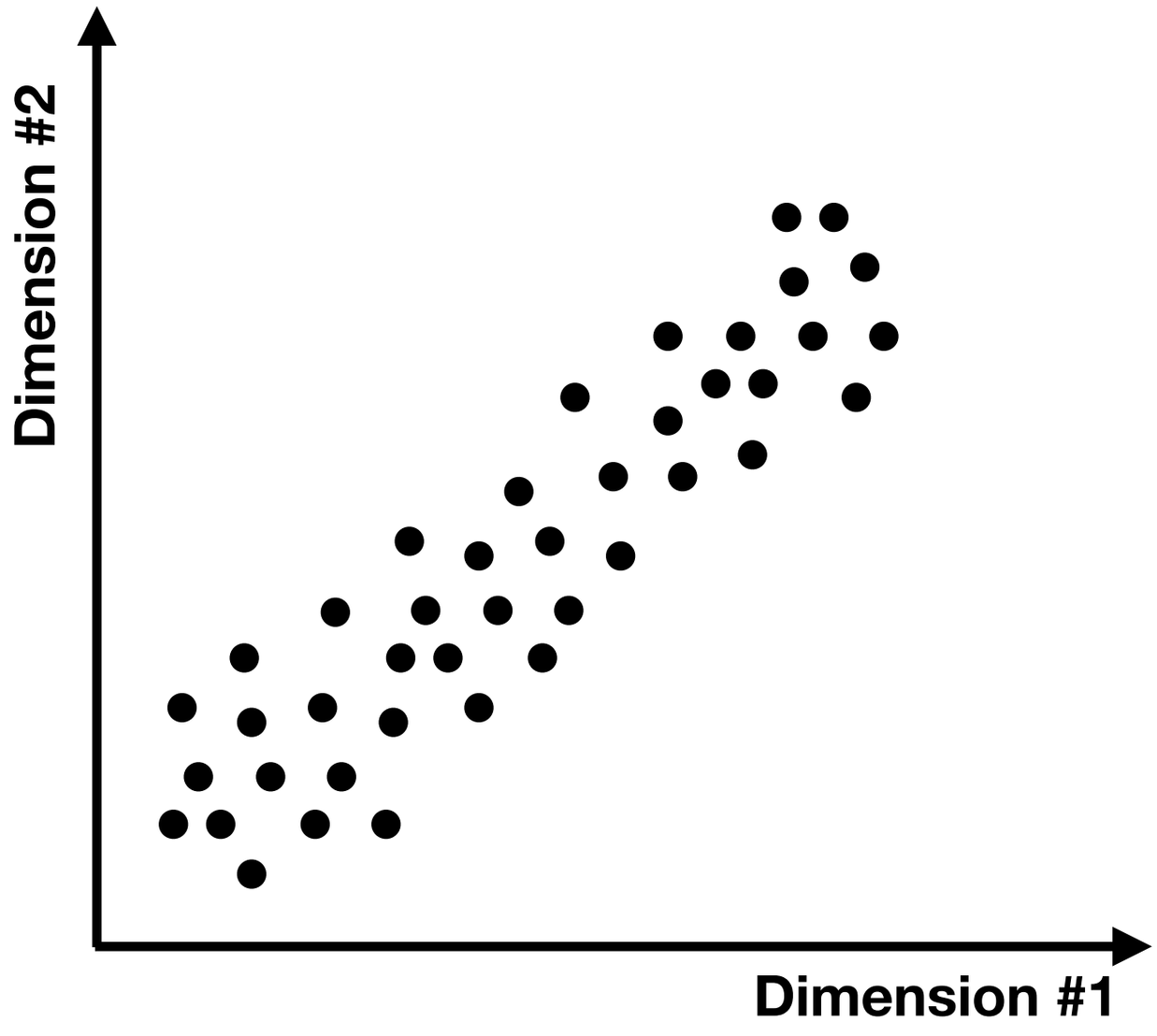
If we have prototypes - try to understand what they mean

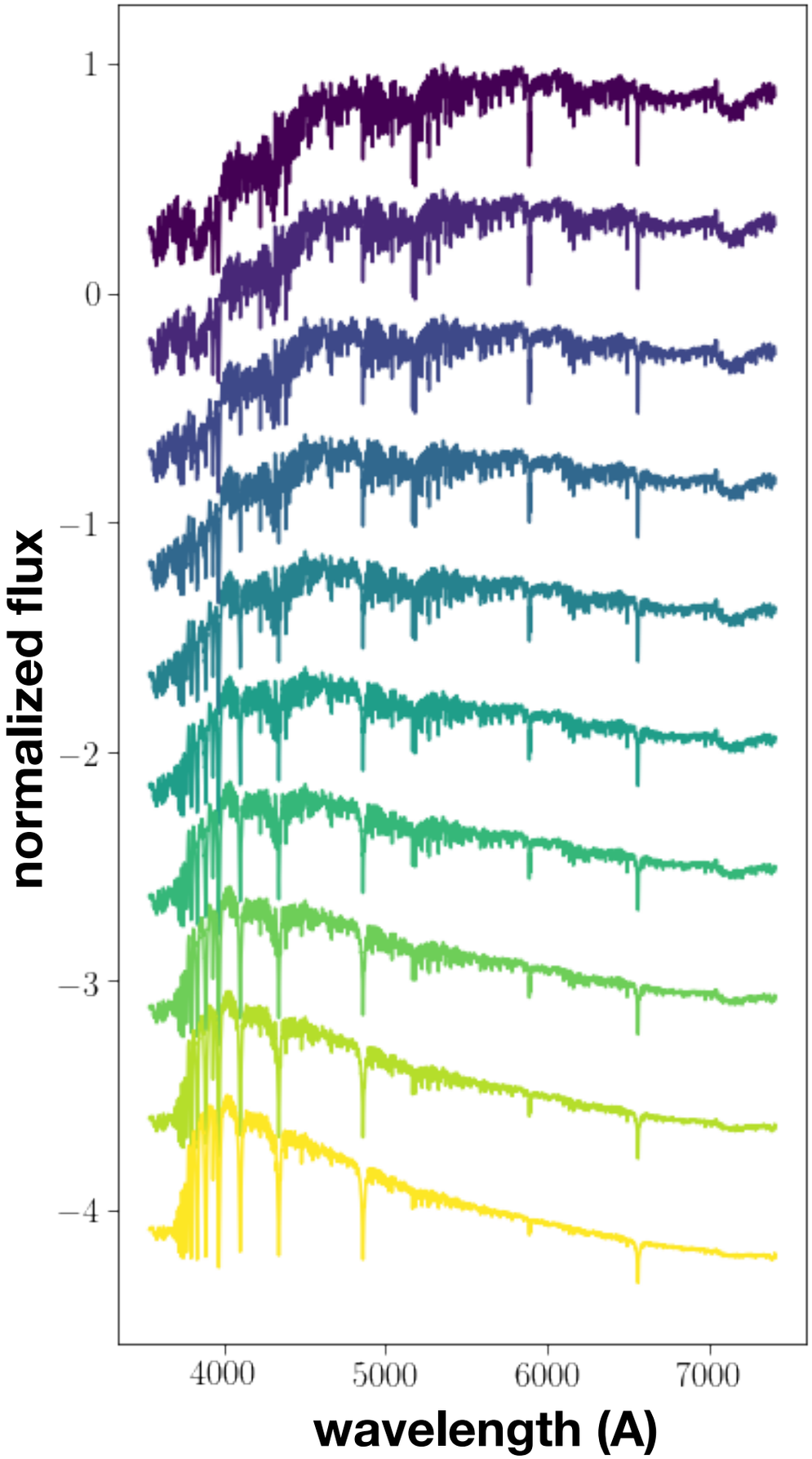


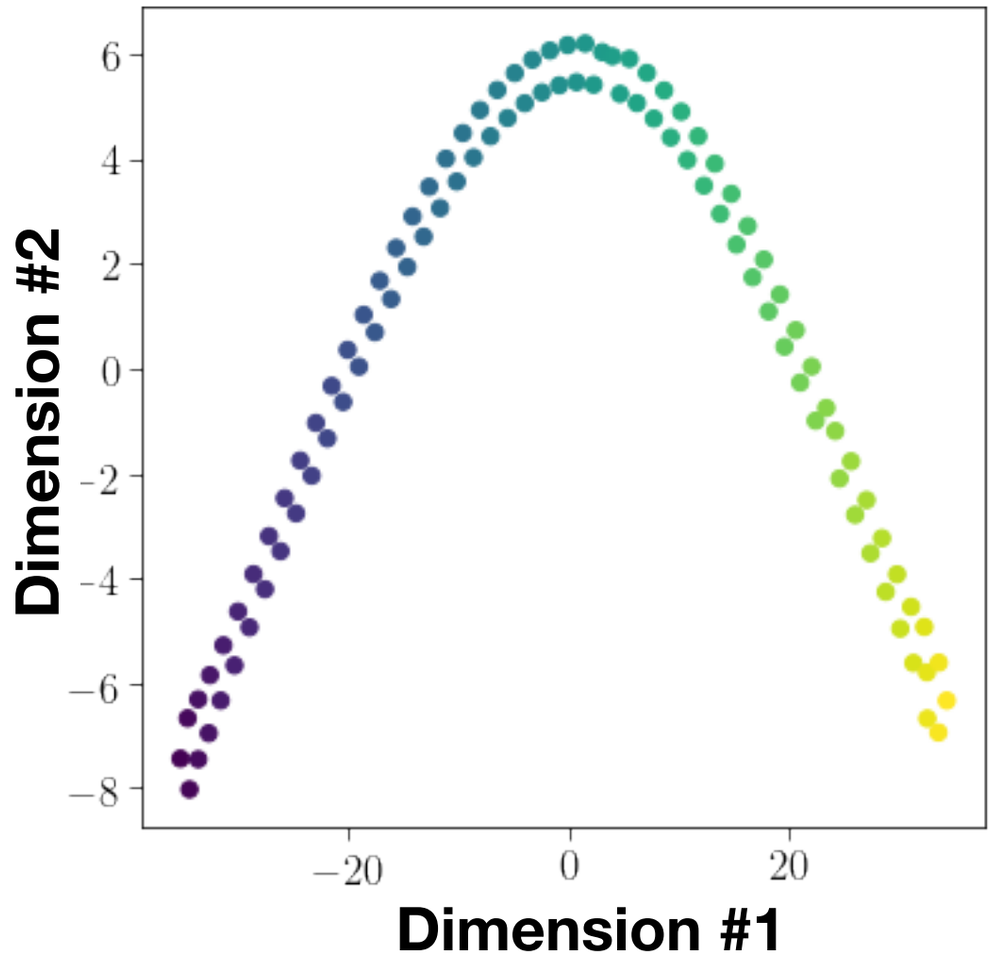
How to interpret the output of a dimensionality reduction algorithm?



How to interpret the output of a dimensionality reduction algorithm?

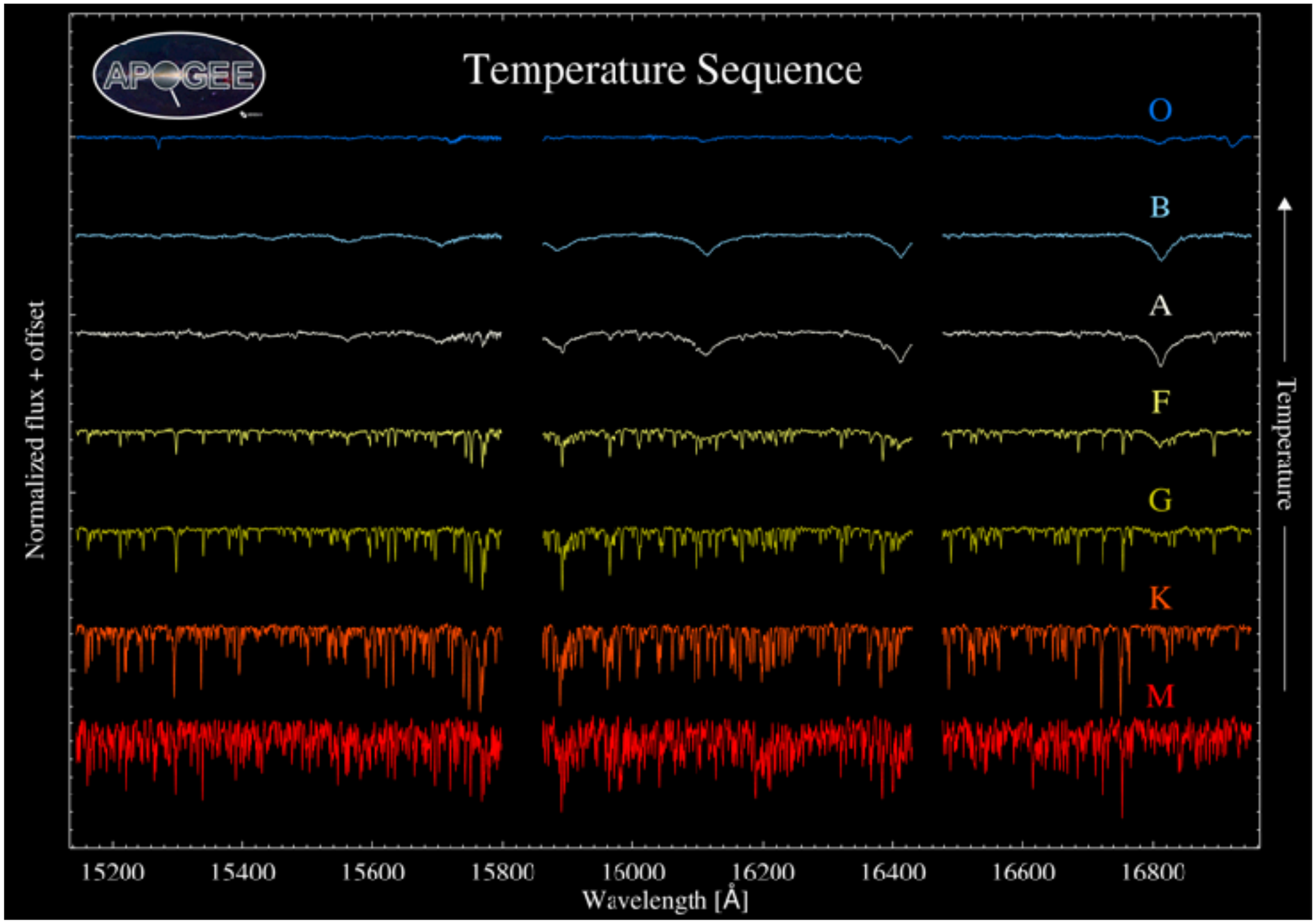



normalized flux
wavelength (A)
1
0
−1
−2
−3
−4
4000
5000
6000
7000

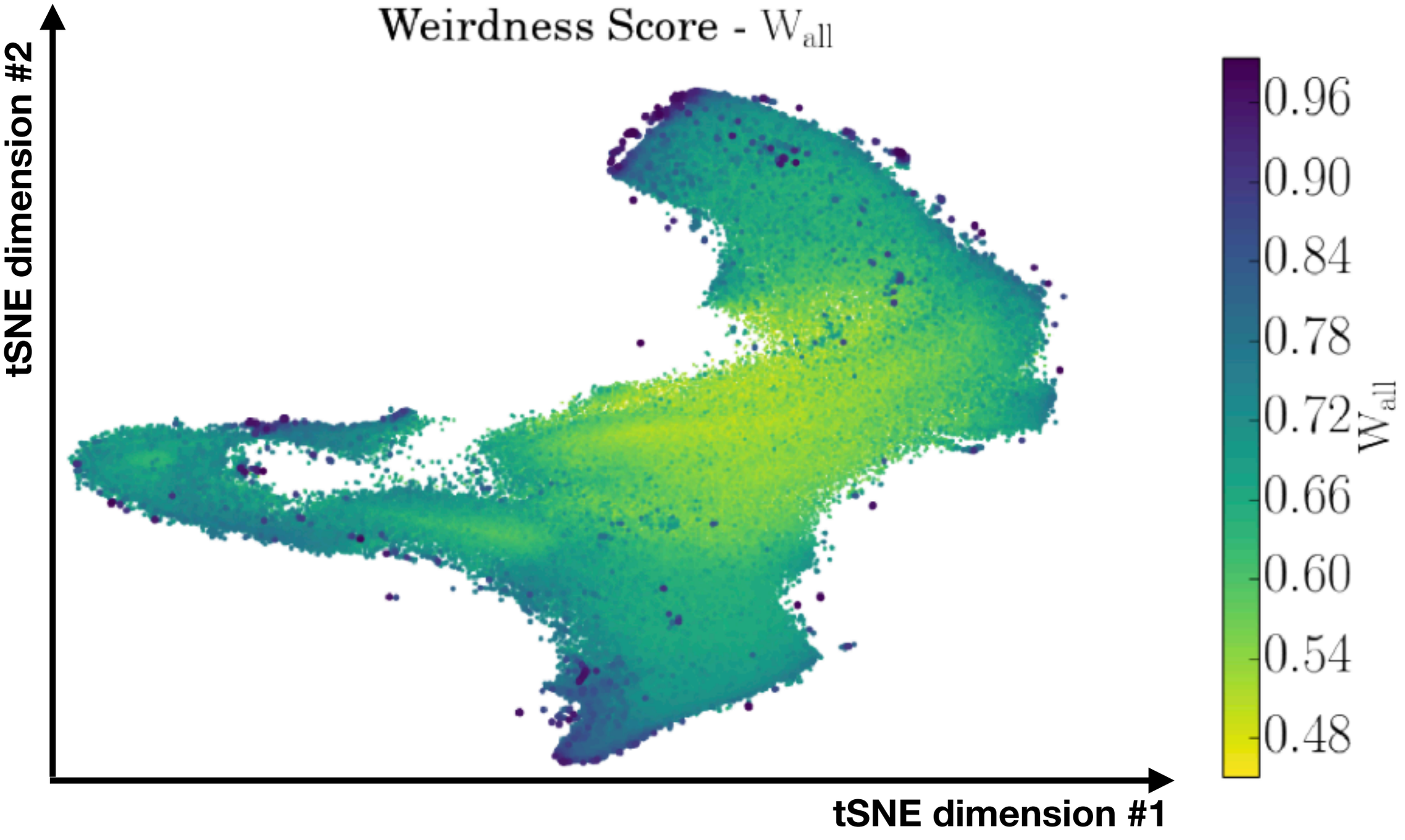

tSNE embedding in two dimensions
Dimension #2
Dimension #1
6
4
2
0
−2
−4
−6
−8
−20
0
20

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.

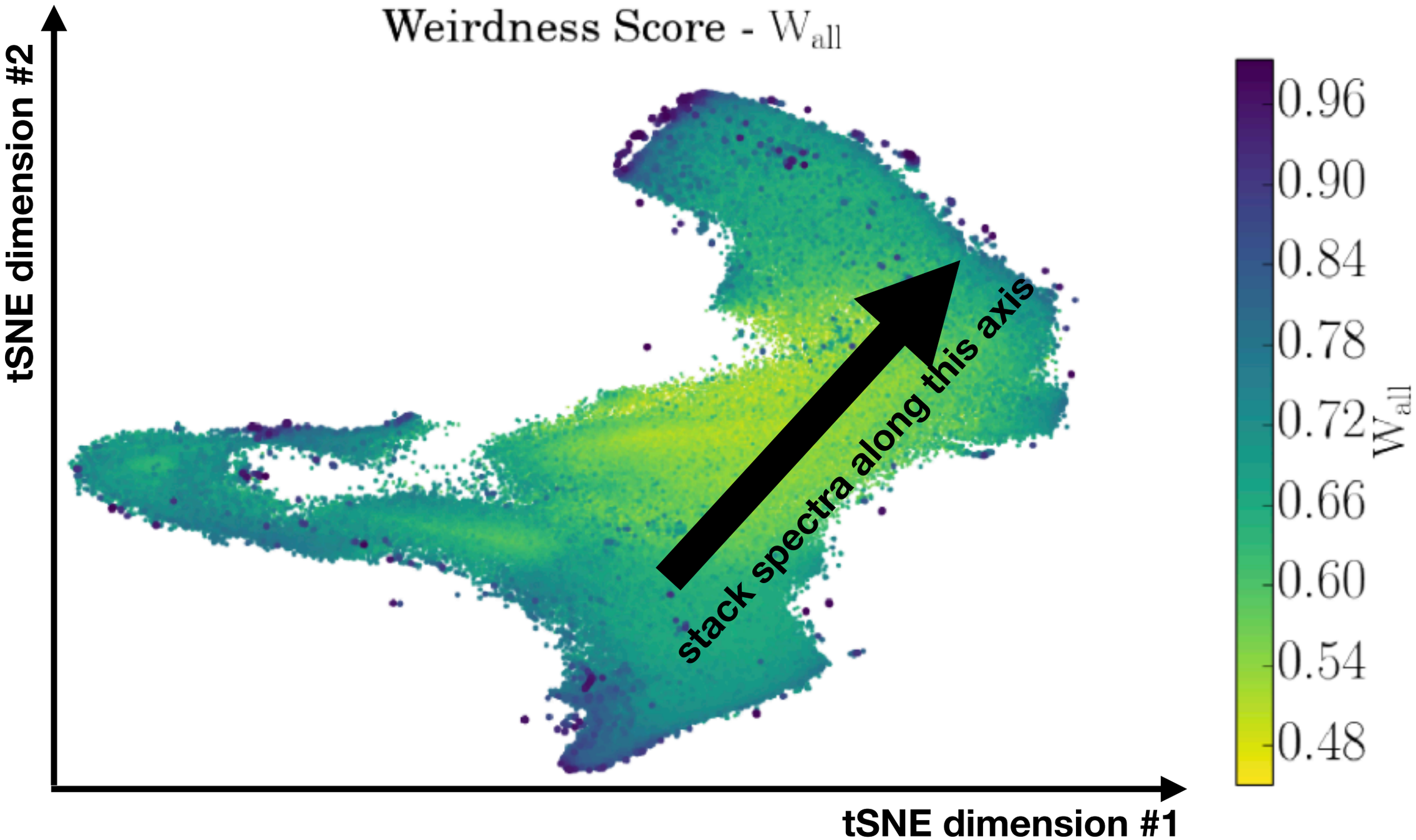


Example with the APOGEE dataset



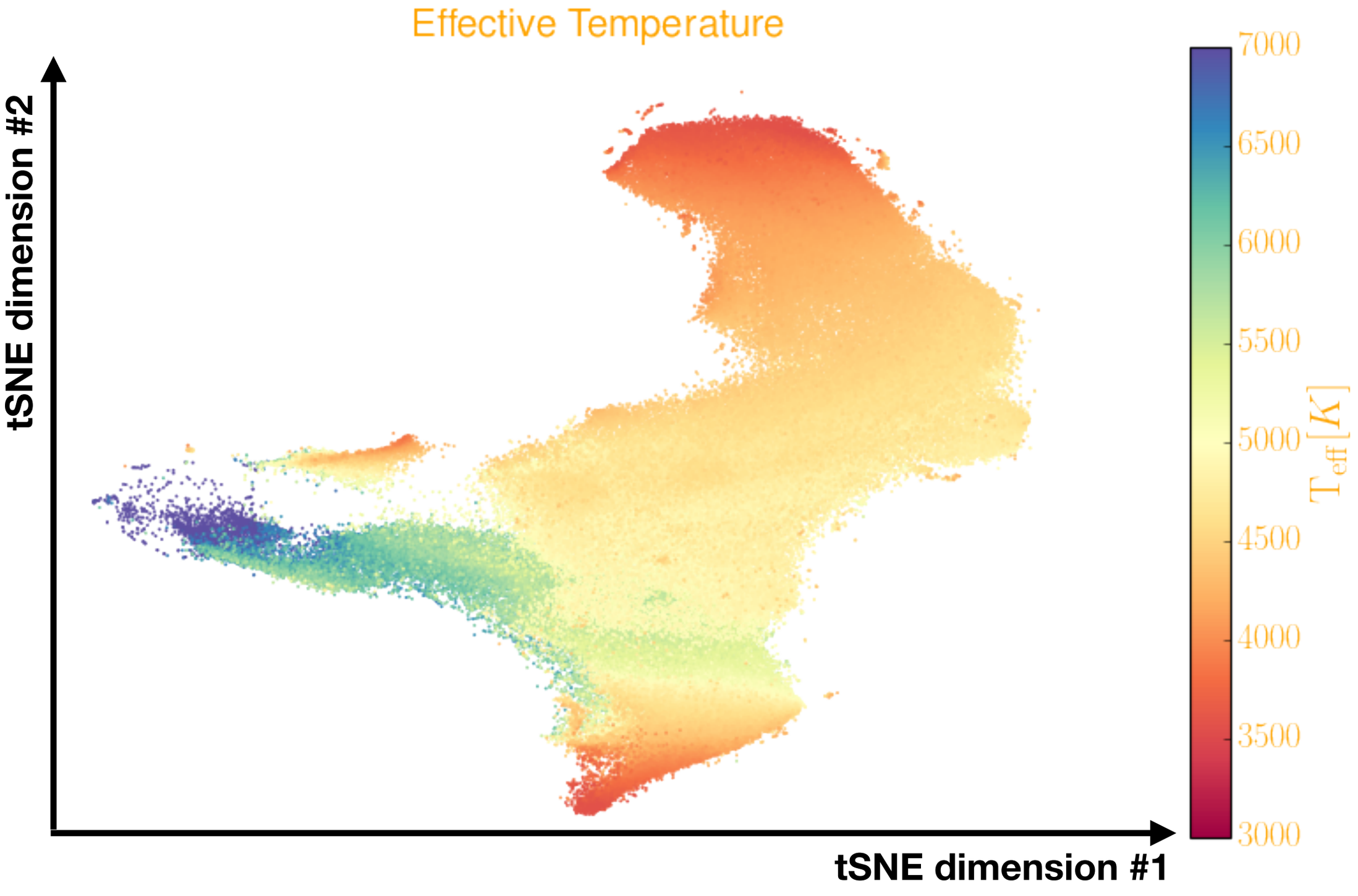
Example with the APOGEE dataset

1. Stack observations along different axises.



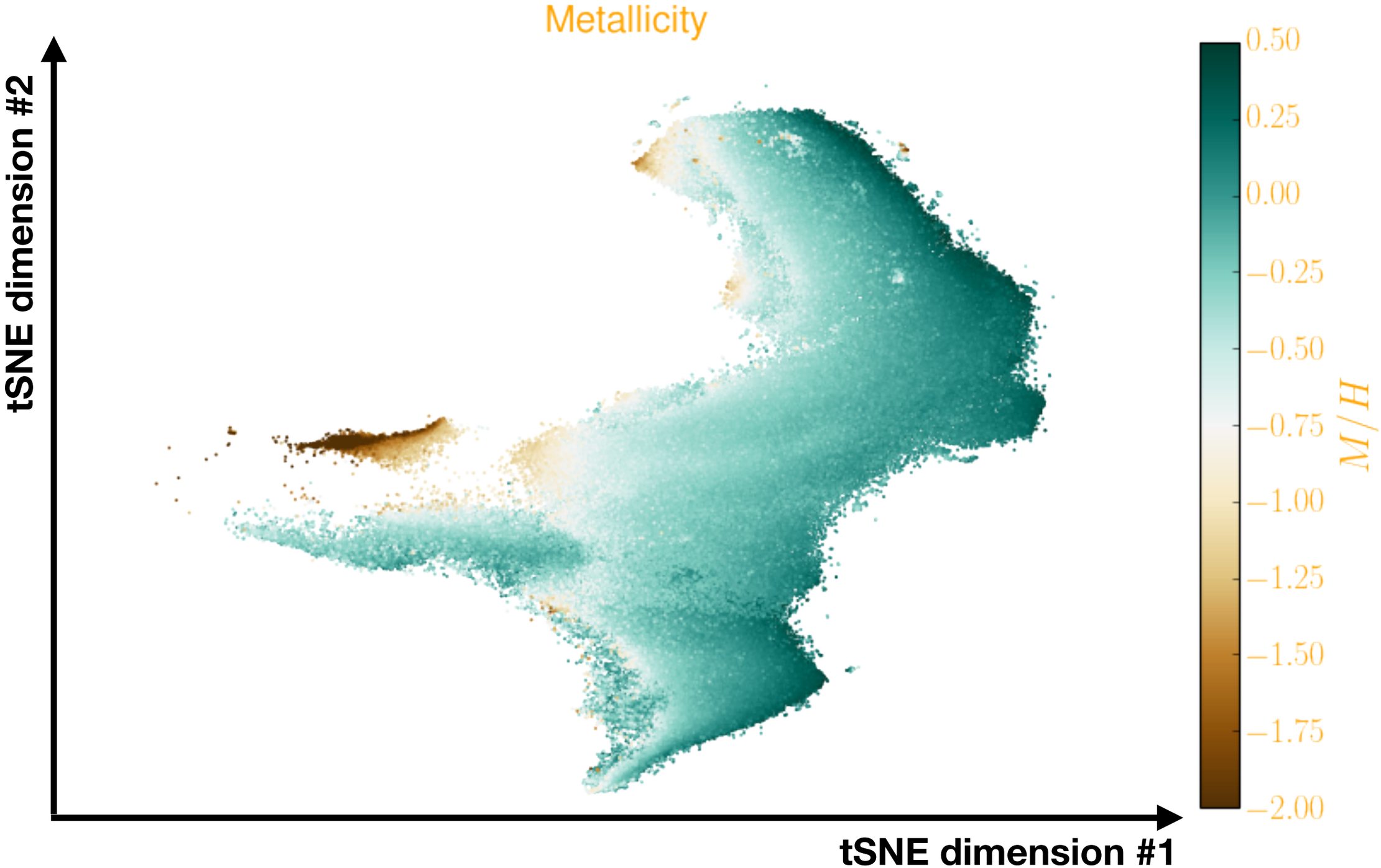
Example with the APOGEE dataset

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



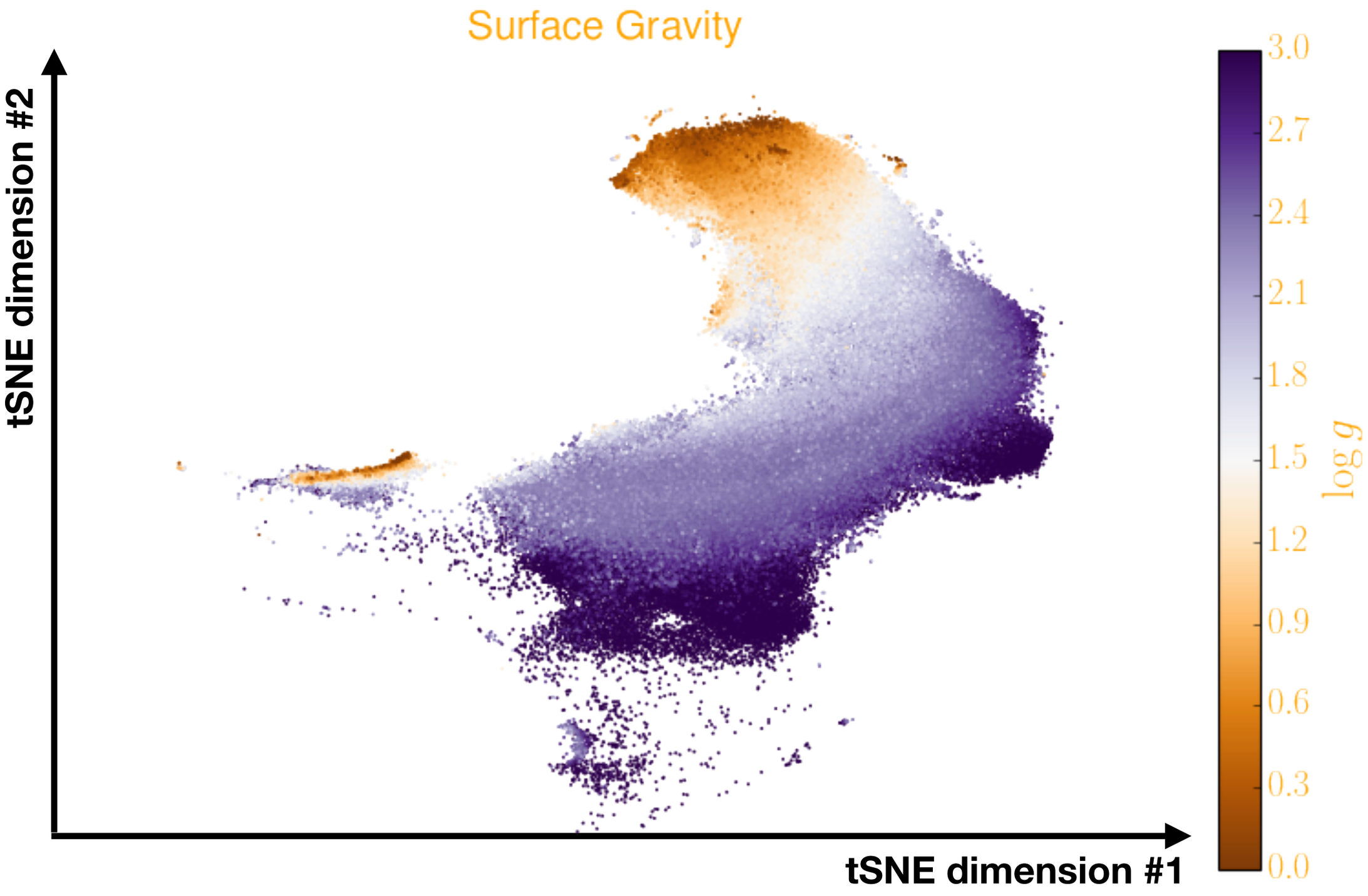
Example with the APOGEE dataset

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



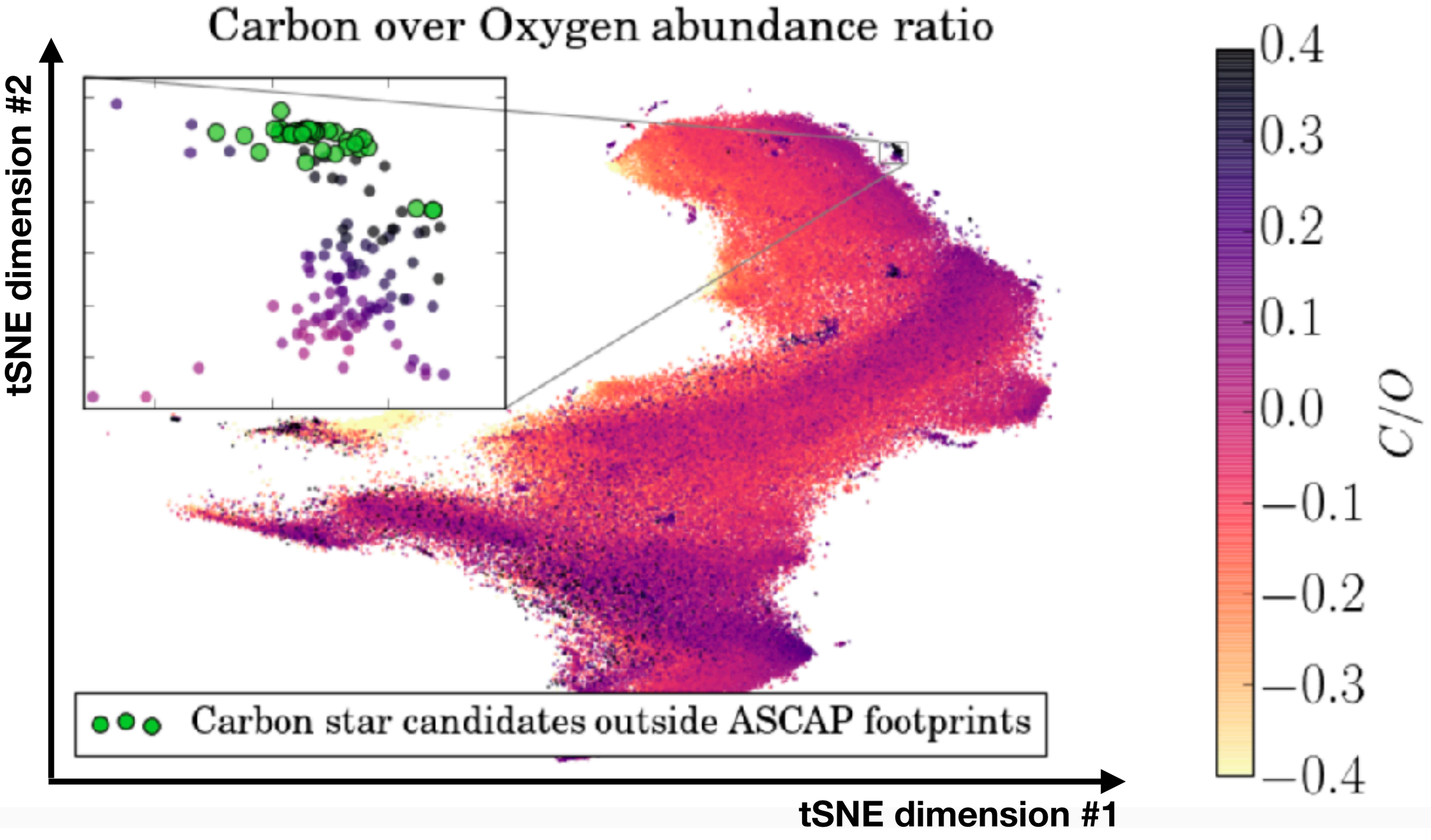
Example with the APOGEE dataset

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)



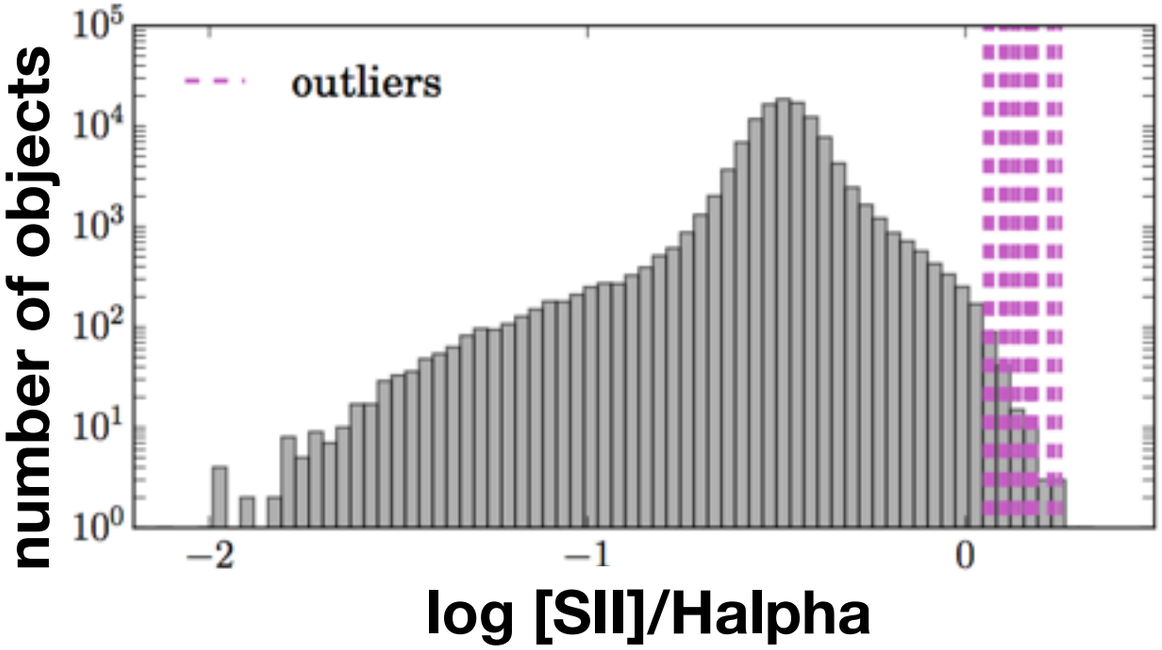
Outlier Detection

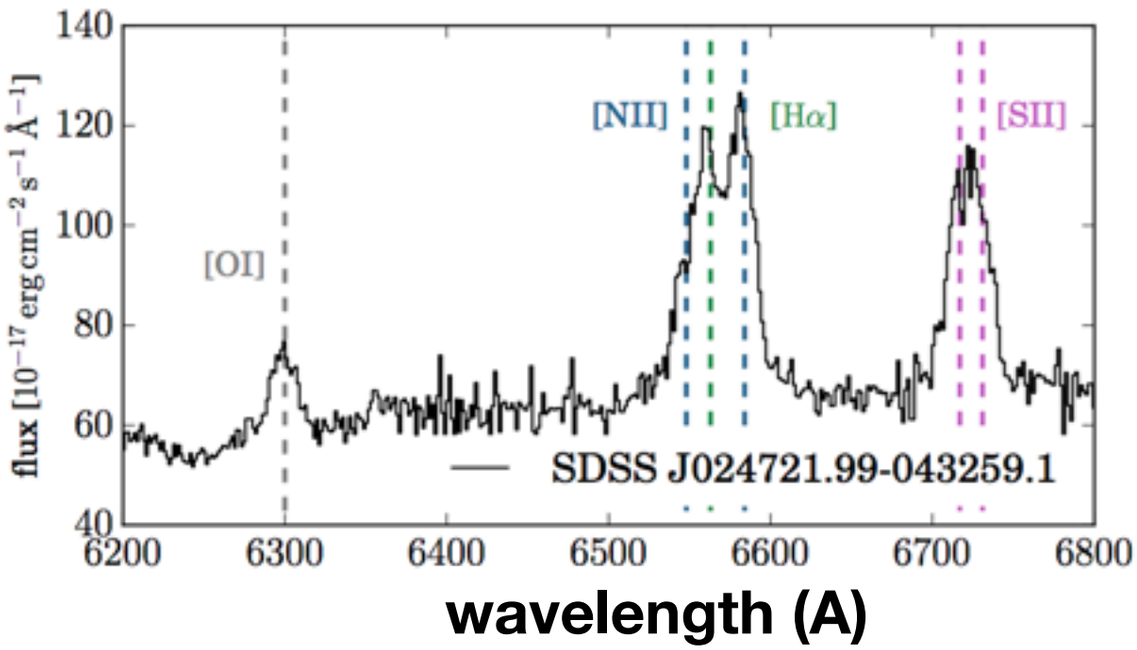
What is an outlier?

- **“Bad” object:** artifacts, cosmic rays, bad reduction.
- **Misclassified object:** star classified as QSO, variable star classified as SN.
- **Tail of a distribution:** most luminous SN, fastest accreting BH.
- **Unknown unknowns:** completely new objects we did not know we should be looking for.
- **In astronomy:** processes which happen on shorter time scales.

How do we find outliers?

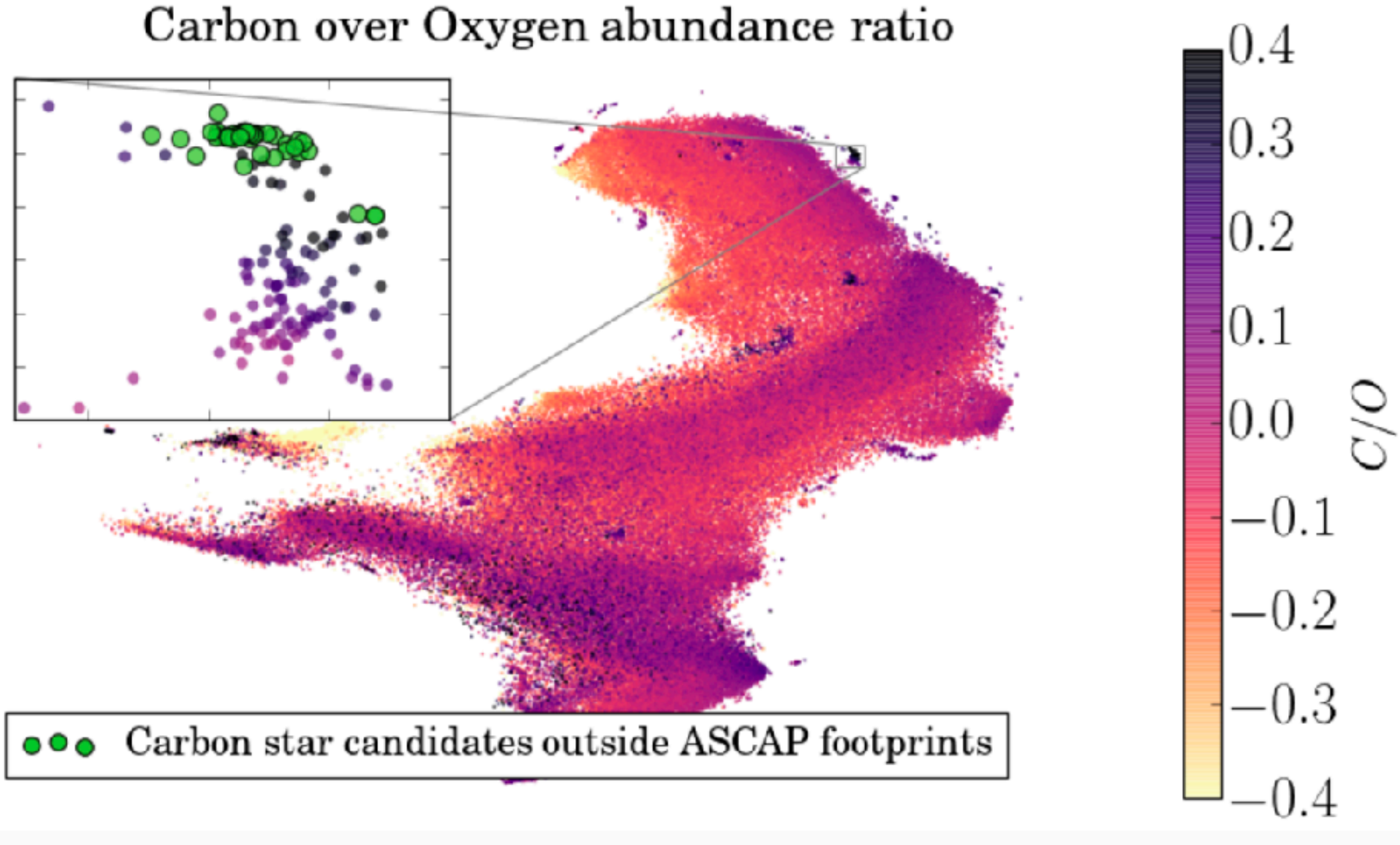
1. Stay as close as possible to the original dataset. Measured features will not always help (see Baron & Poznanski 2017).





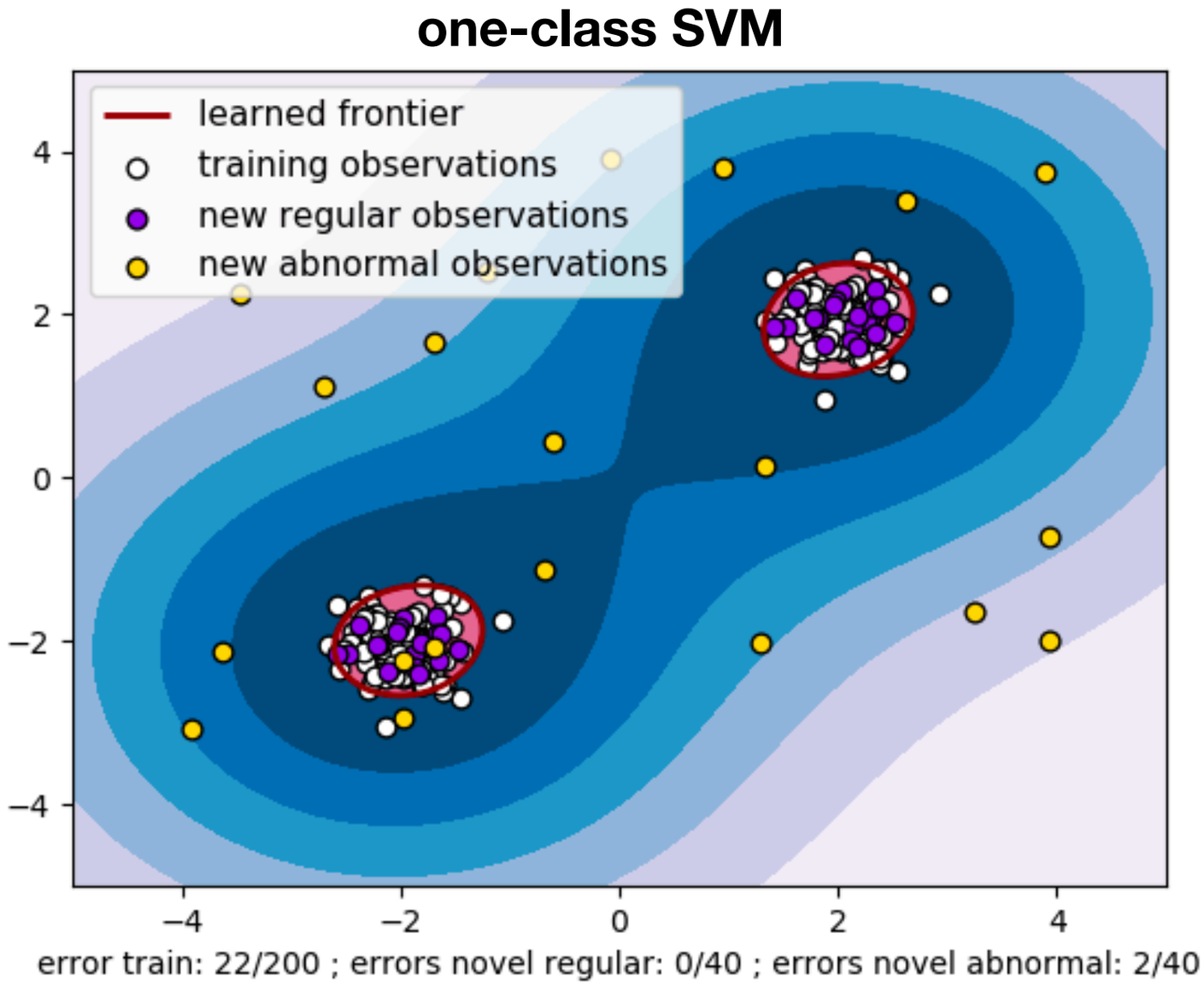
How do we find outliers?

1. Stay as close as possible to the original dataset. Measured features will usually not help (see Baron & Poznanski 2017).



How do we find outliers?

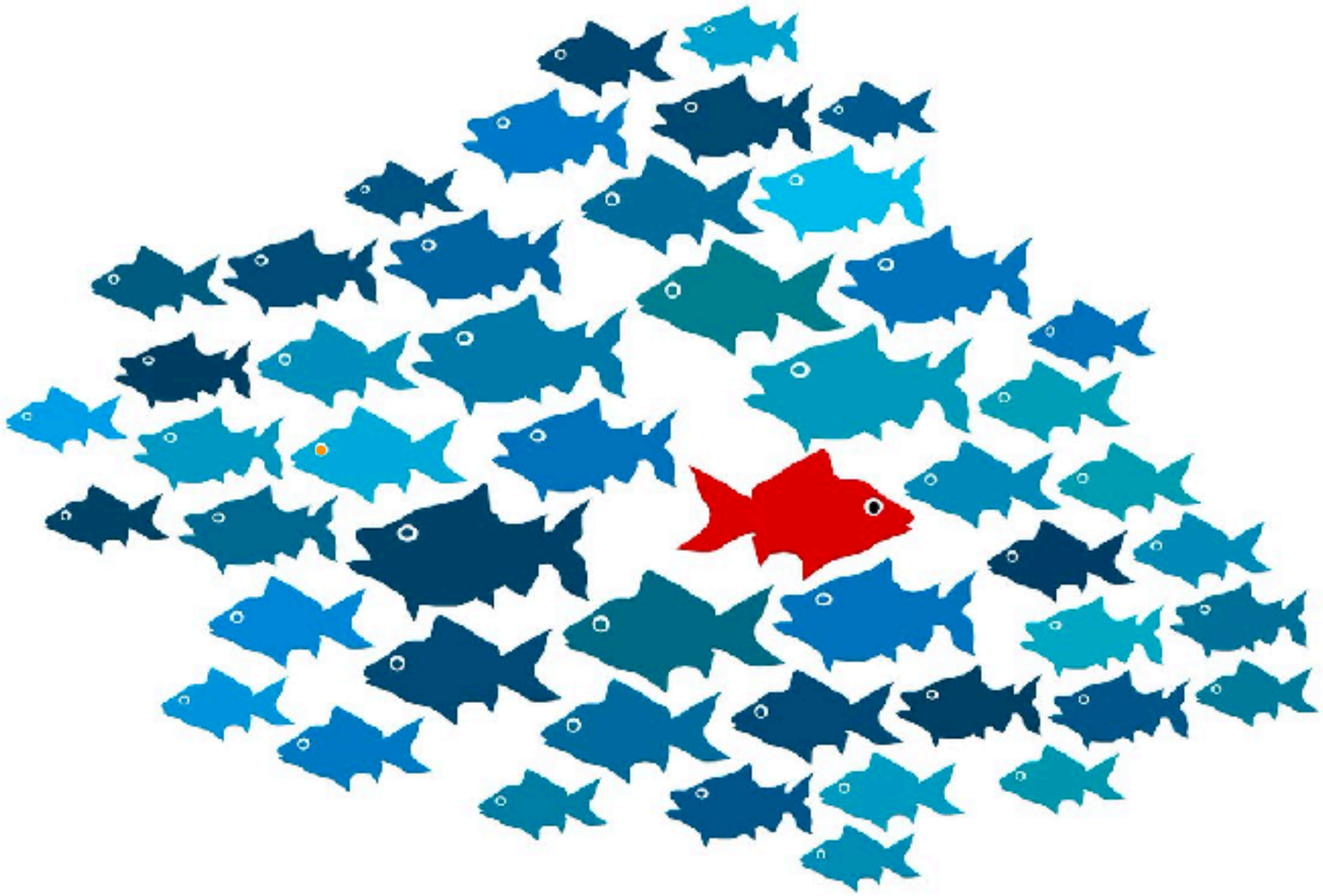
2. Supervised learning-based outlier detection will uncover the outliers that “shout the strongest”.



Additional supervised algorithms that can be used: **RF, ANN, etc..**

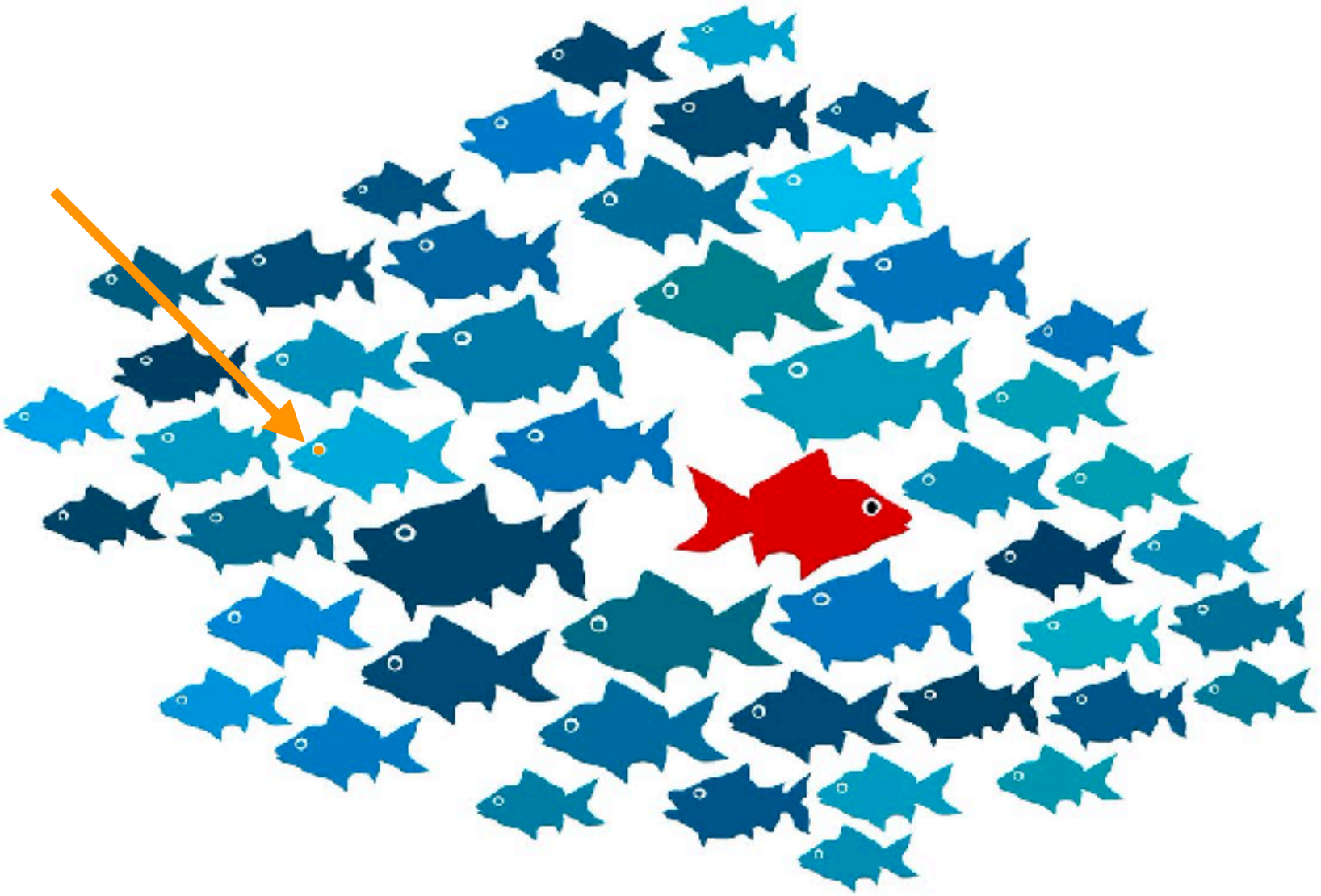
How do we find outliers?

2. Supervised learning-based outlier detection will uncover the outliers that “shout the strongest”.



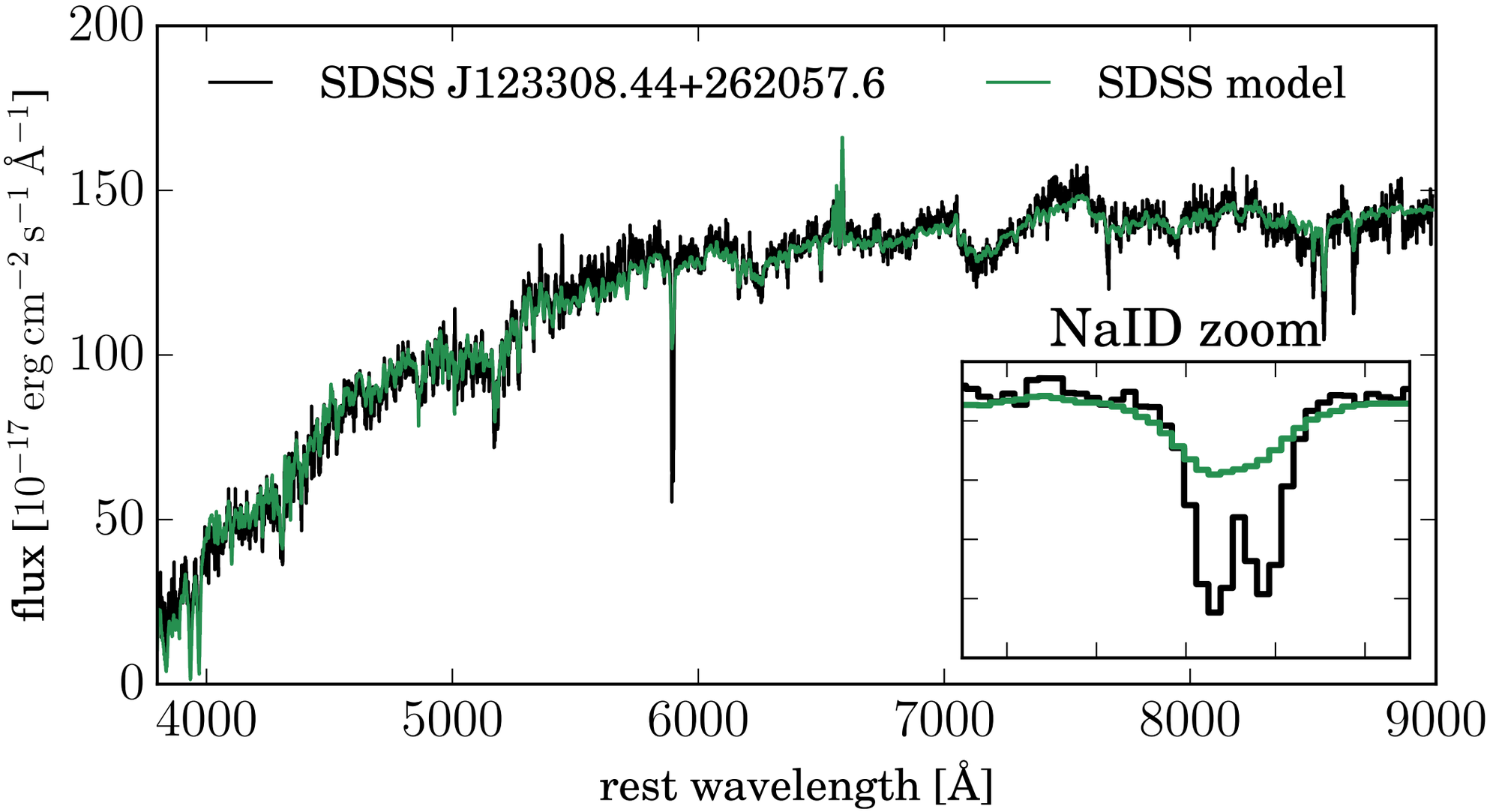
How do we find outliers?

2. Supervised learning-based outlier detection will uncover the outliers that “shout the strongest”.



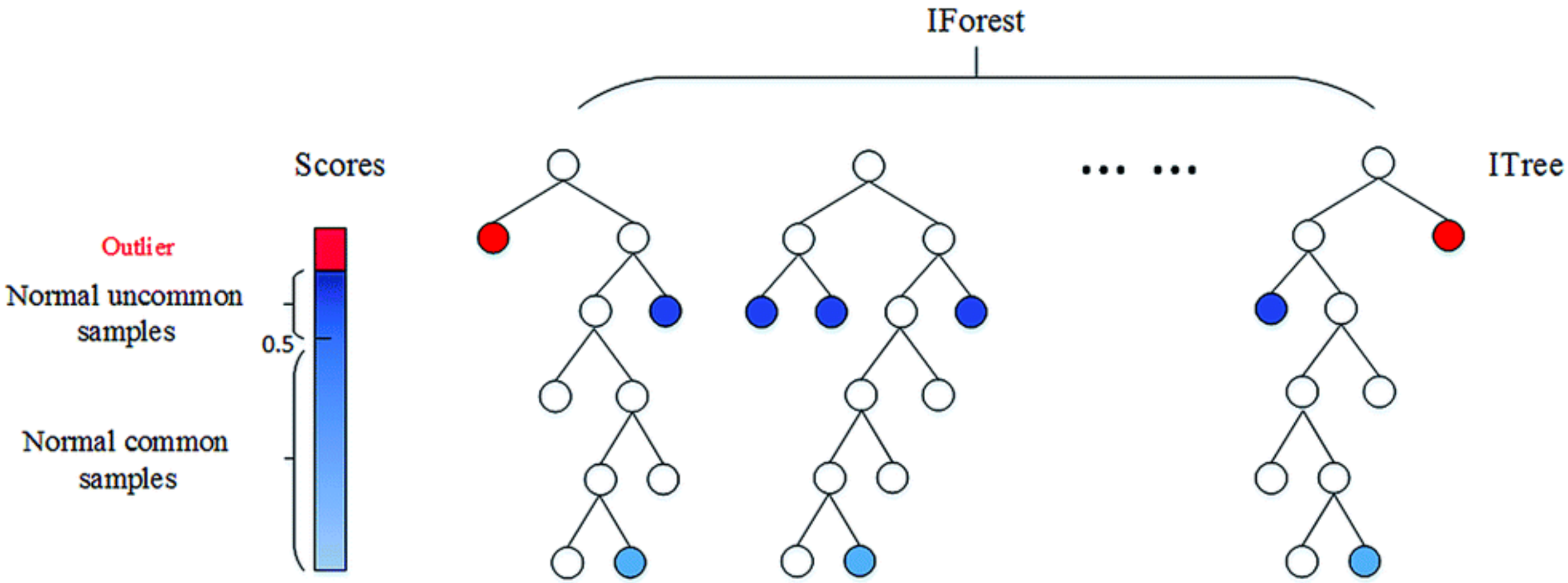
How do we find outliers?

2. Supervised learning-based outlier detection will uncover the outliers that “shout the strongest” (see Baron & Poznanski 2017).



Isolation Forests

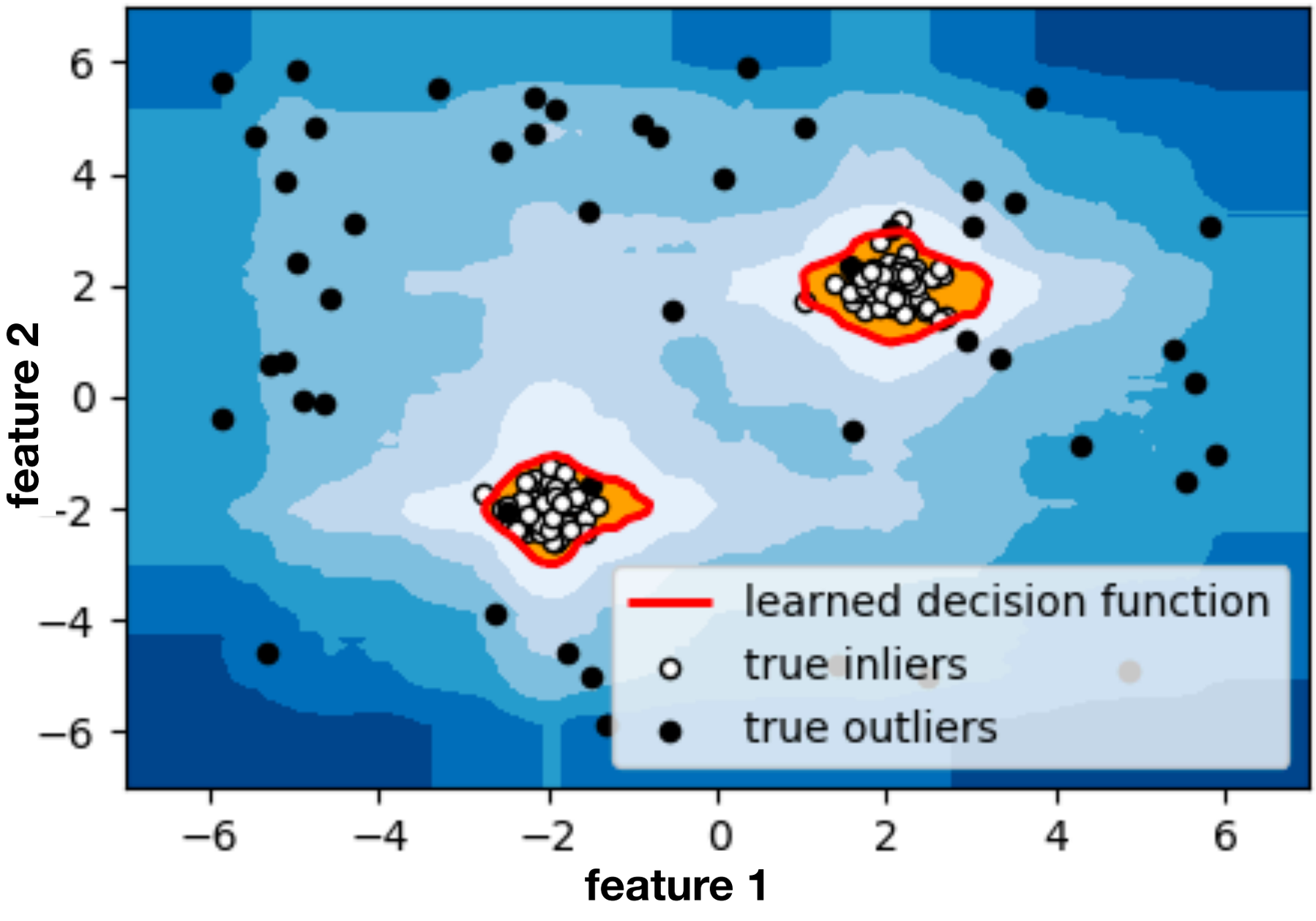
A fast algorithm that does not require distance measurements.
Outliers are objects that are separated from the rest of the dataset higher in the tree.



http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Isolation Forests

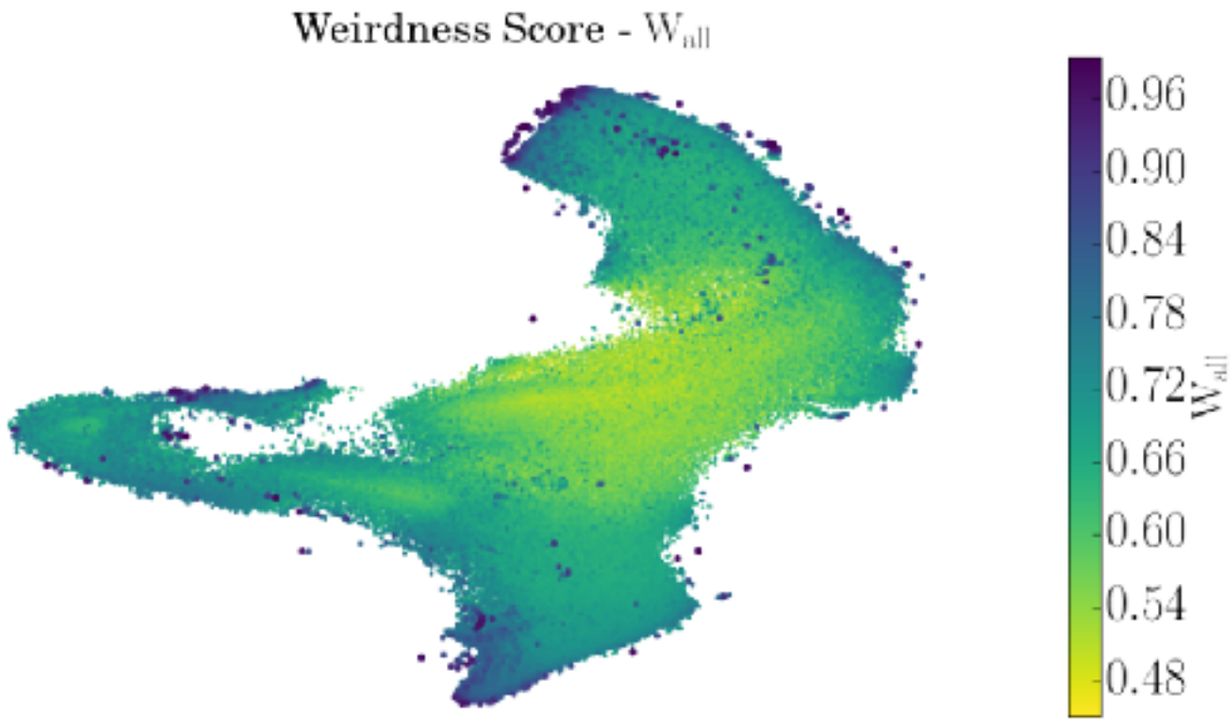
Example:

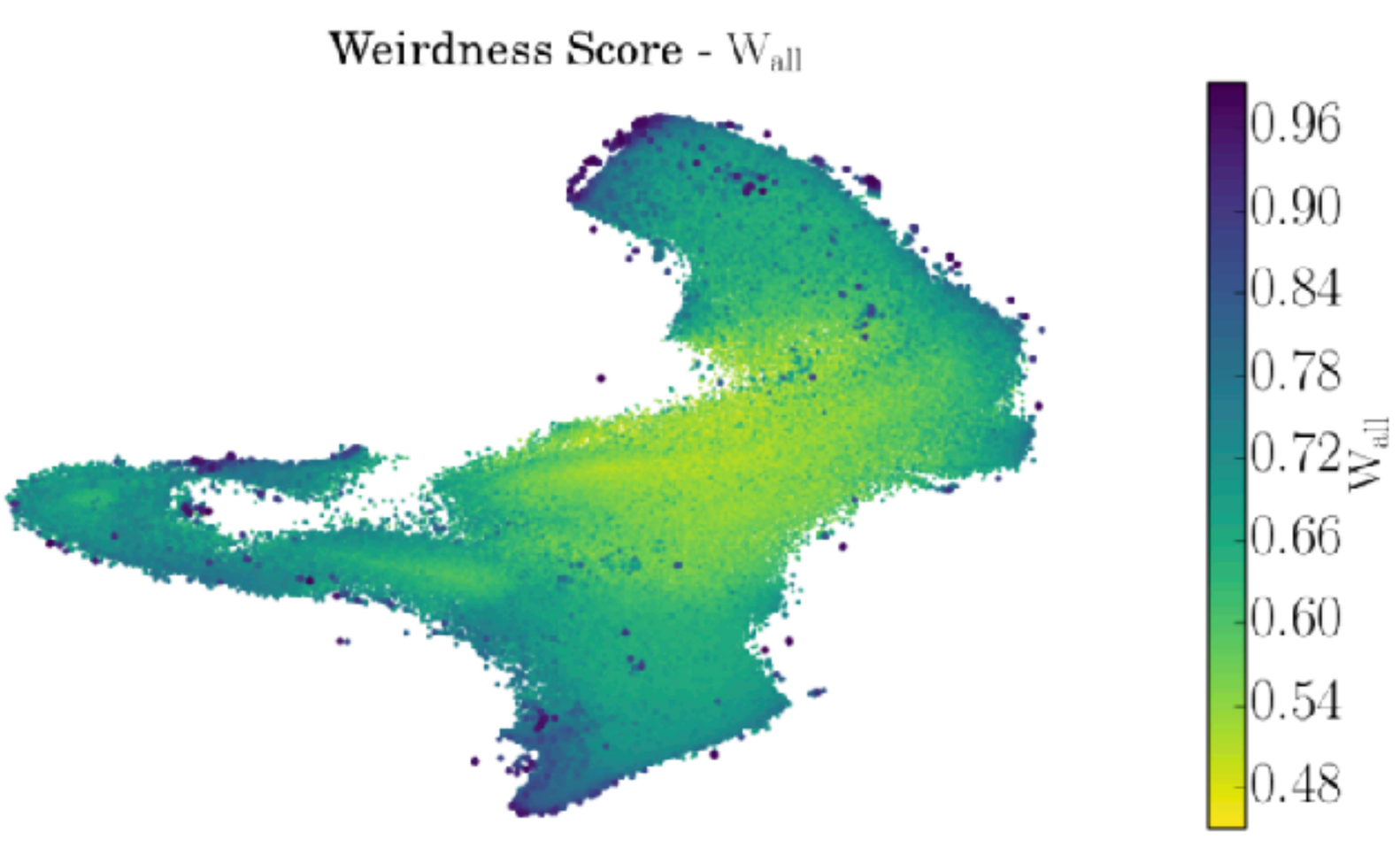


Outlier detection on spectra using unsupervised RF

See: Baron & Poznanski 2017, Reis+18

Use unsupervised Random Forest to define distances between the objects in the sample. Define outliers are objects with a large average distance from all the rest, or from their ~250 nearest neighbors. Use **tSNE** to explore outlier clusters.

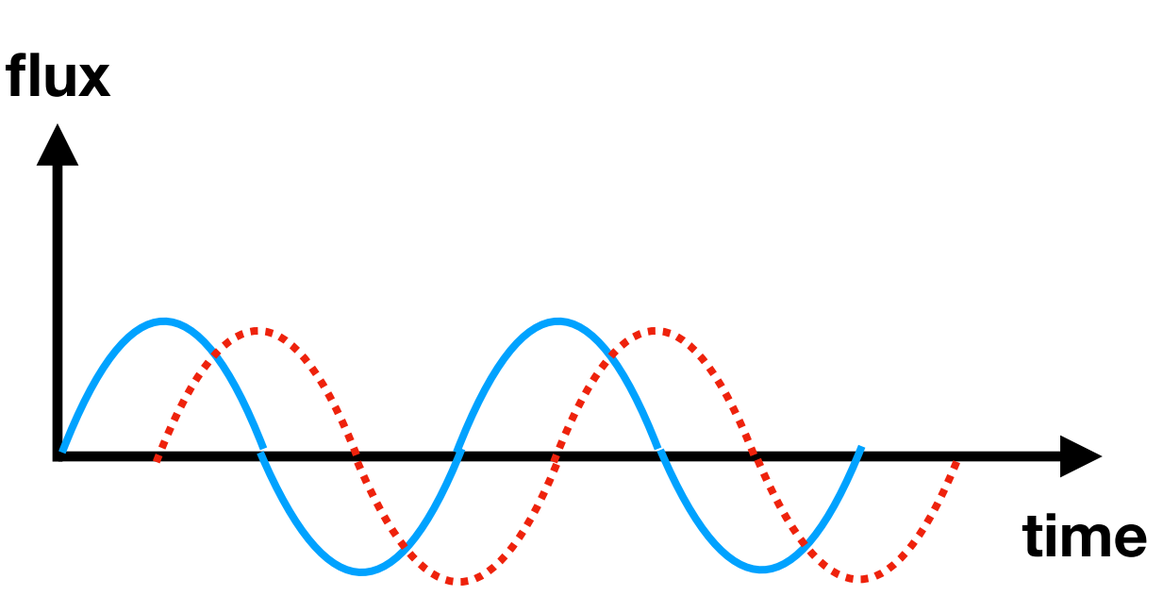


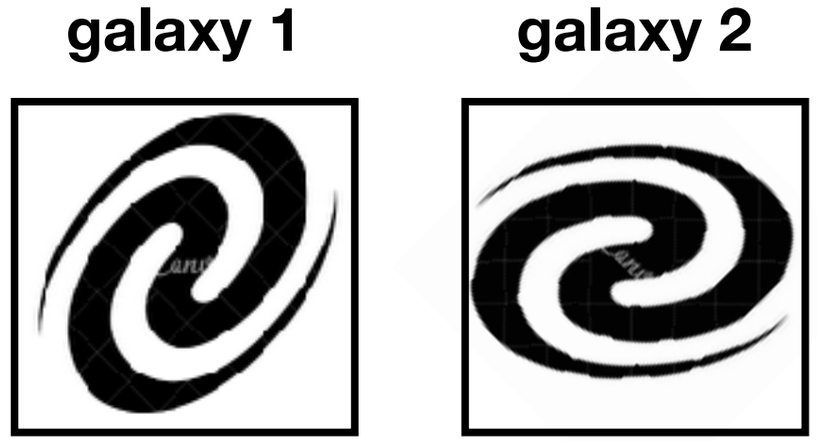

Weirdness Score - W_{all}
0.96
0.90
0.84
0.78
0.72
0.66
0.60
0.54
0.48
W_{all}


Weirdness Score - W_{250}
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
W_{250}

Outlier detection on other datasets using unsupervised RF?

The unsupervised Random Forest assumes a regular grid, and thus will work for spectra or extracted features.
It will not work for images or time series, because it does not have translational and rotational symmetry!

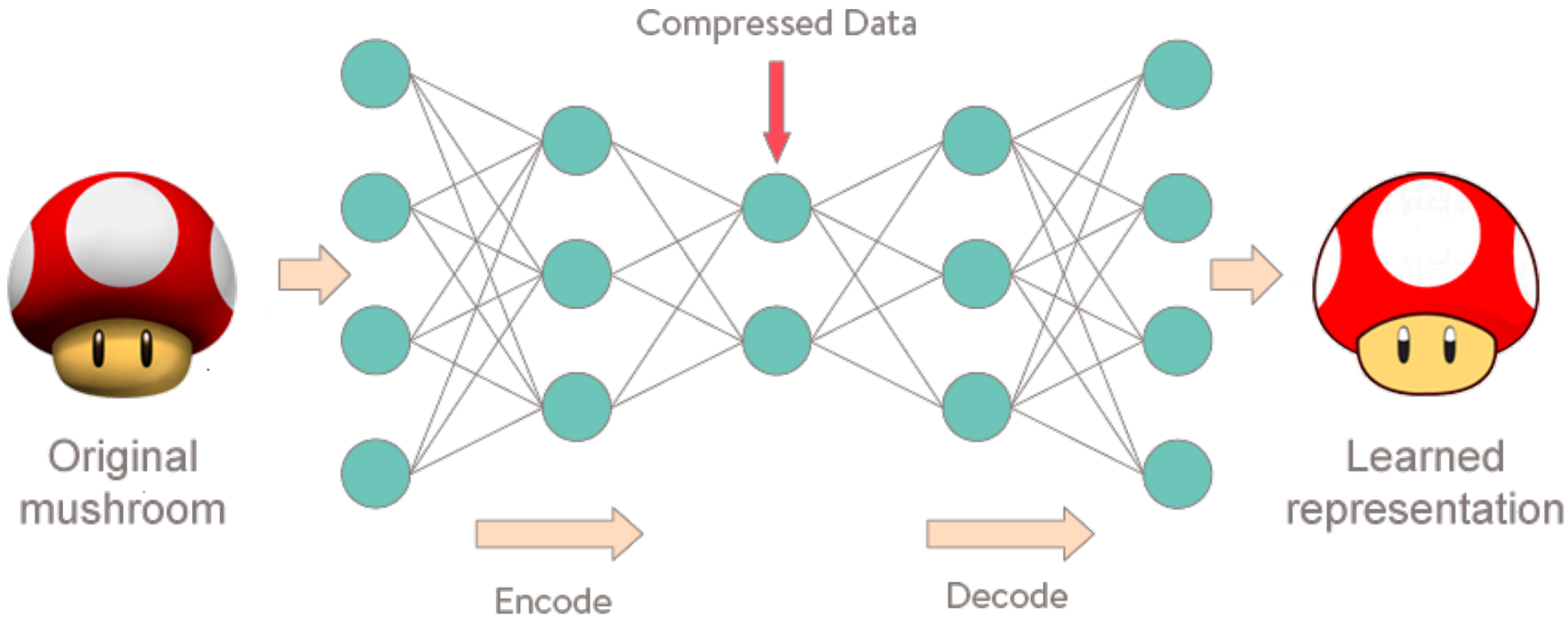




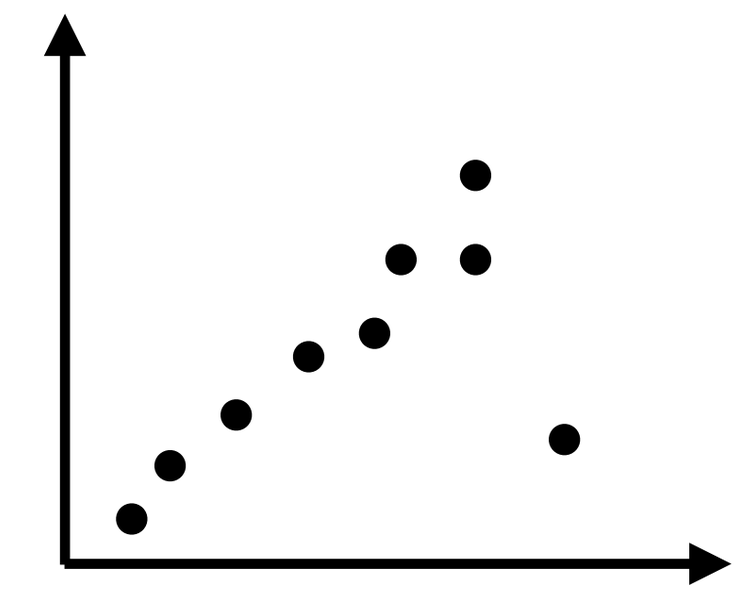
possible solution: find a representation of the signal on a regular grid (e.g., FFT of time series).

Outlier detection using Autoencoders

Autoencoders can represent images (CNN) and time-series (RNN) well.
Use the latent space to look for, and examine outliers.

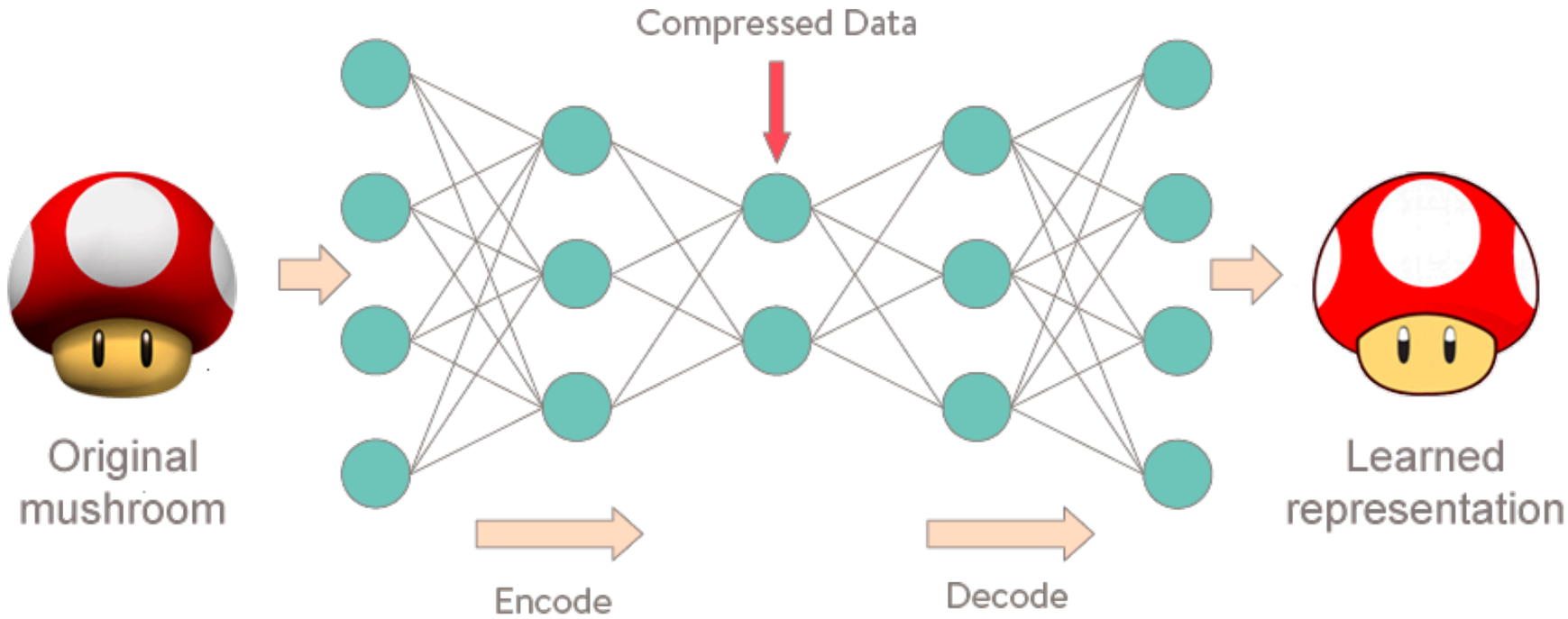


Latent (compressed) space:

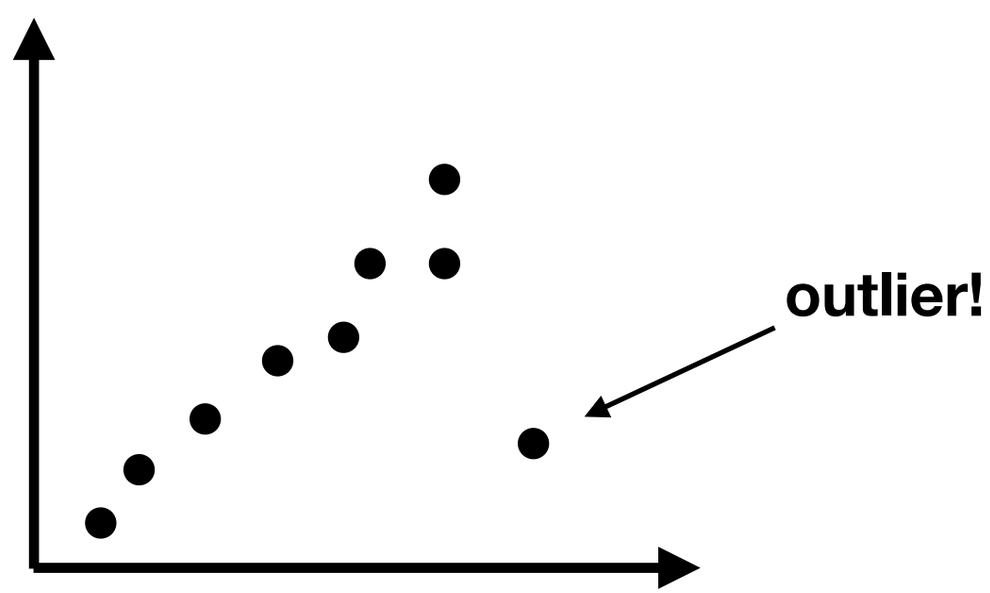


Outlier detection using Autoencoders

Autoencoders can represent images (CNN) and time-series (RNN) well.
Use the latent space to look for, and examine outliers.

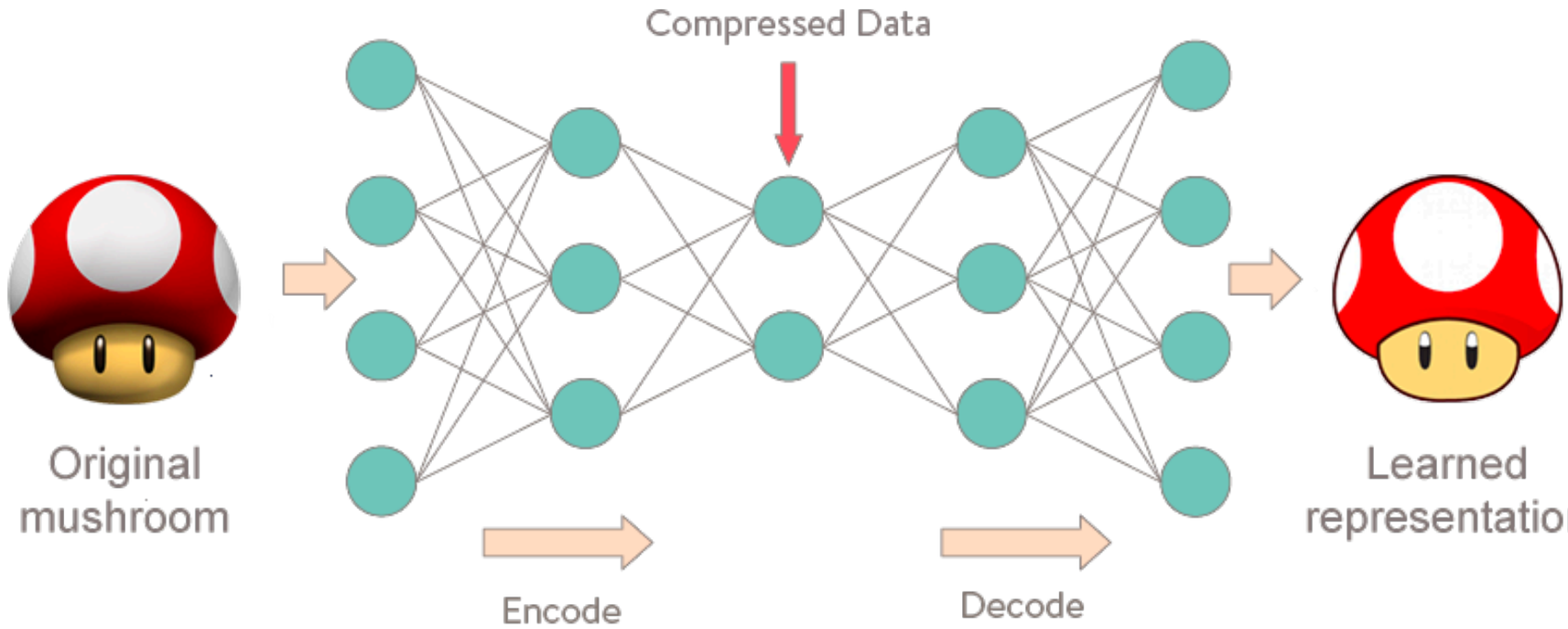


Latent (compressed) space:

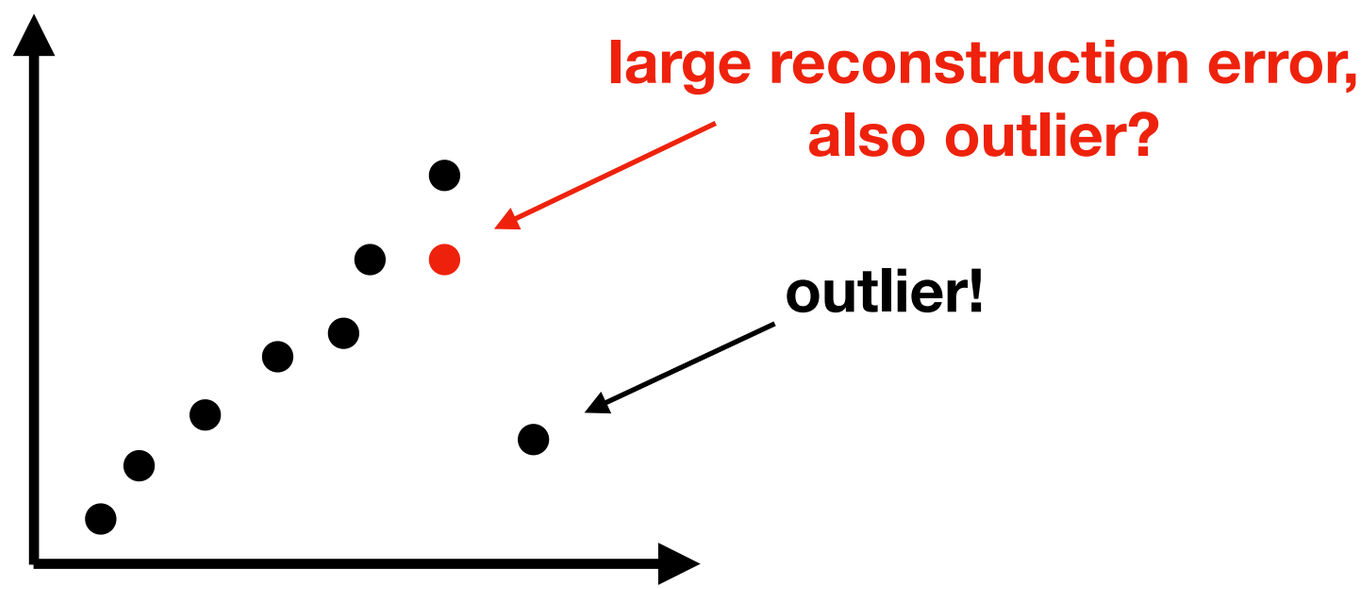


Outlier detection using Autoencoders

Autoencoders can represent images (CNN) and time-series (RNN) well.
Use the latent space to look for, and examine outliers.



Latent (compressed) space:



Want to learn more?

- Basics of Machine Learning: https://www.coursera.org/learn/machine-learning?utm_source=gg&utm_medium=sem&campaignid=685340575&adgroupid=32639001341&device=c&keyword=coursera%20machine%20learning%20course&matchtype=b&network=g&devicemodel=&adpostion=1t1&creativeid=243289762778&hide_mobile_promo&gclid=EAIaIQobChMI_aHhgZ7q3gIVbZPtCh0AYA7iEAAYASAAEgI-0PD_BwE

- Book: Statistics, Data Mining, and Machine Learning in Astronomy, by Ivezic, Connolly, Vanderplas, and Gray (2013).

- Winter school on Big Data is astronomy: http://www.iac.es/winterschool/2018/pages/about-the-school/syllabus.php

Summary

- Machine learning algorithms are just a tool. Not every problem is appropriate for machine learning.
- Unsupervised (and most supervised) algorithms are not black boxes!!
- Don’t trust me.
- Make friends with computer scientists and engineers.
- Don’t be afraid to change off-the-shelf tools.
- **Open questions:**
  - Uncertainties in supervised and unsupervised algorithms.
  - Unsupervised: we need better cost functions.