Dimensionality Reduction Part 1: PCA & NMF

Dalya Baron
Carnegie Observatories

Vatican Observatory Summer School on Big Data and Machine Learning 2023 (VOSS-2023)

Science is about compression

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

How can we extract simplicity from the observed complex world?

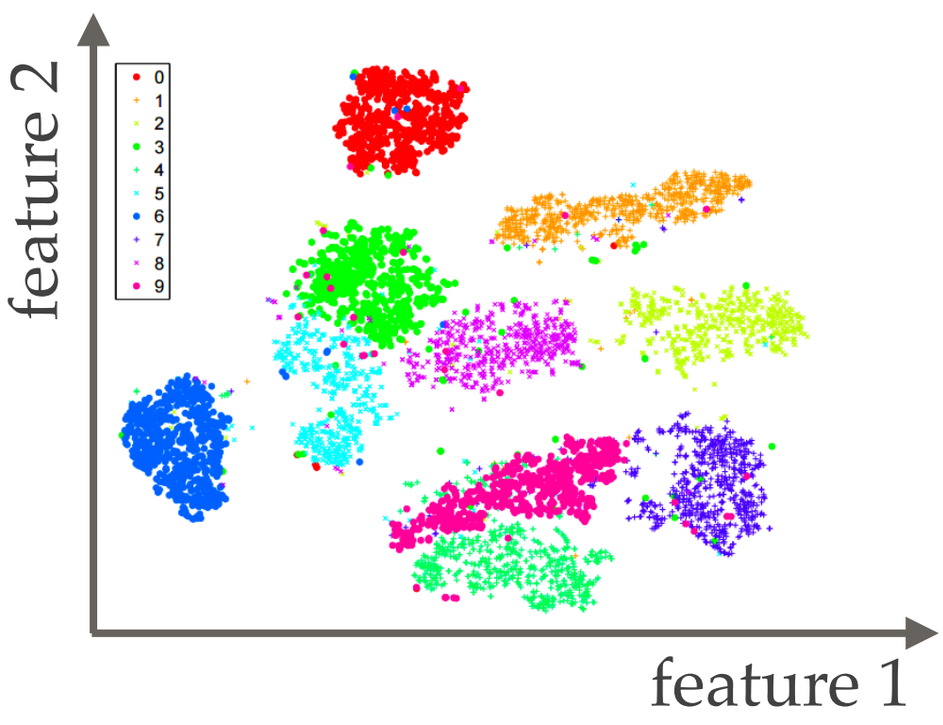
What is dimensionality reduction?

- Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation *retains some meaningful* properties of the original data (taken from wiki).

28 x 28 features per object

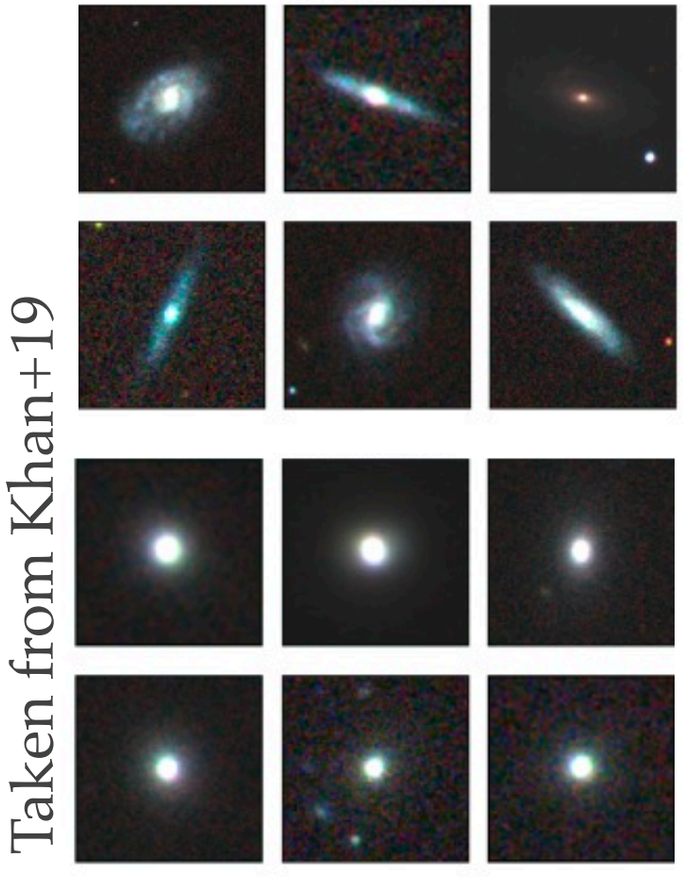
dimensionality reduction algorithm

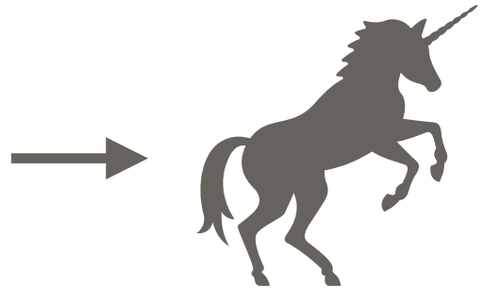
2 features per object



What is dimensionality reduction?

- Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation *retains some meaningful* properties of the original data (taken from wiki).



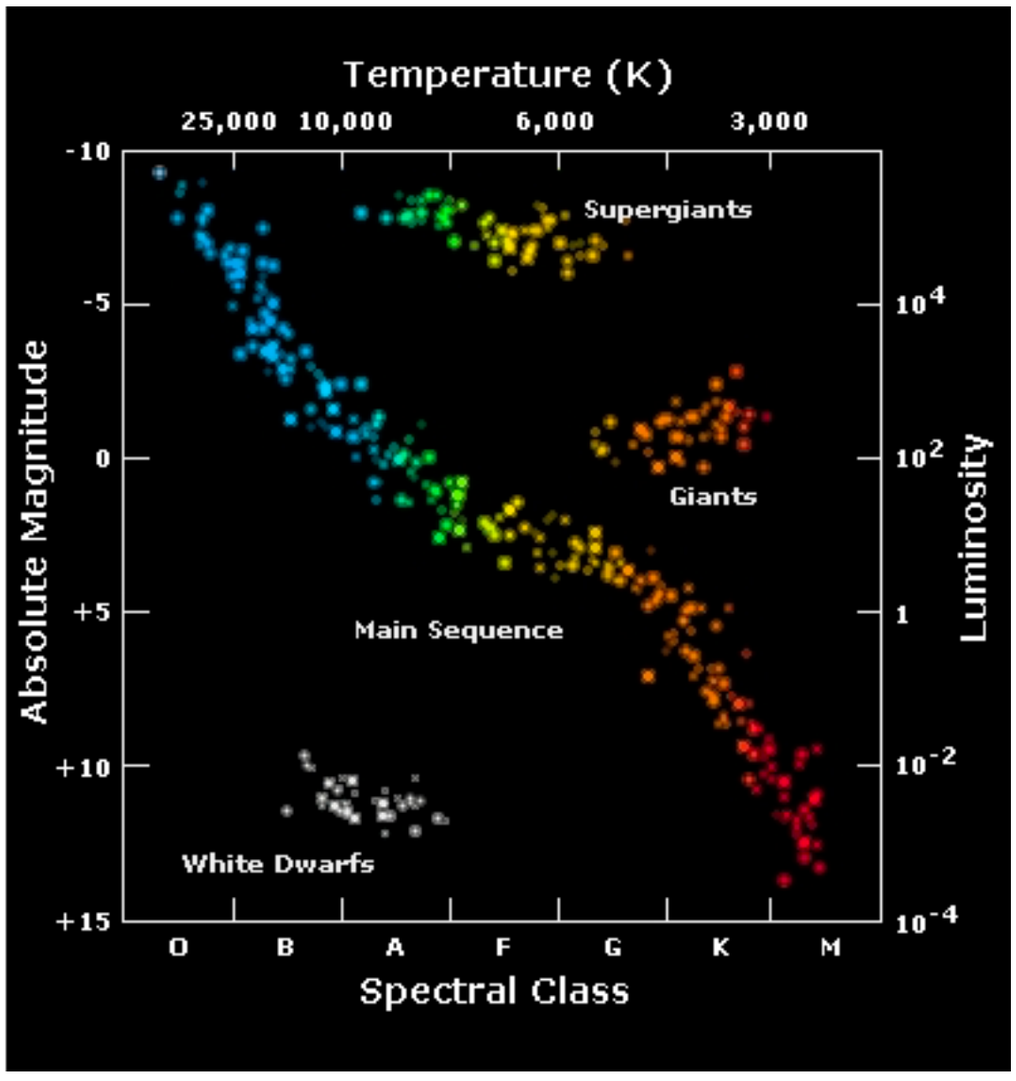


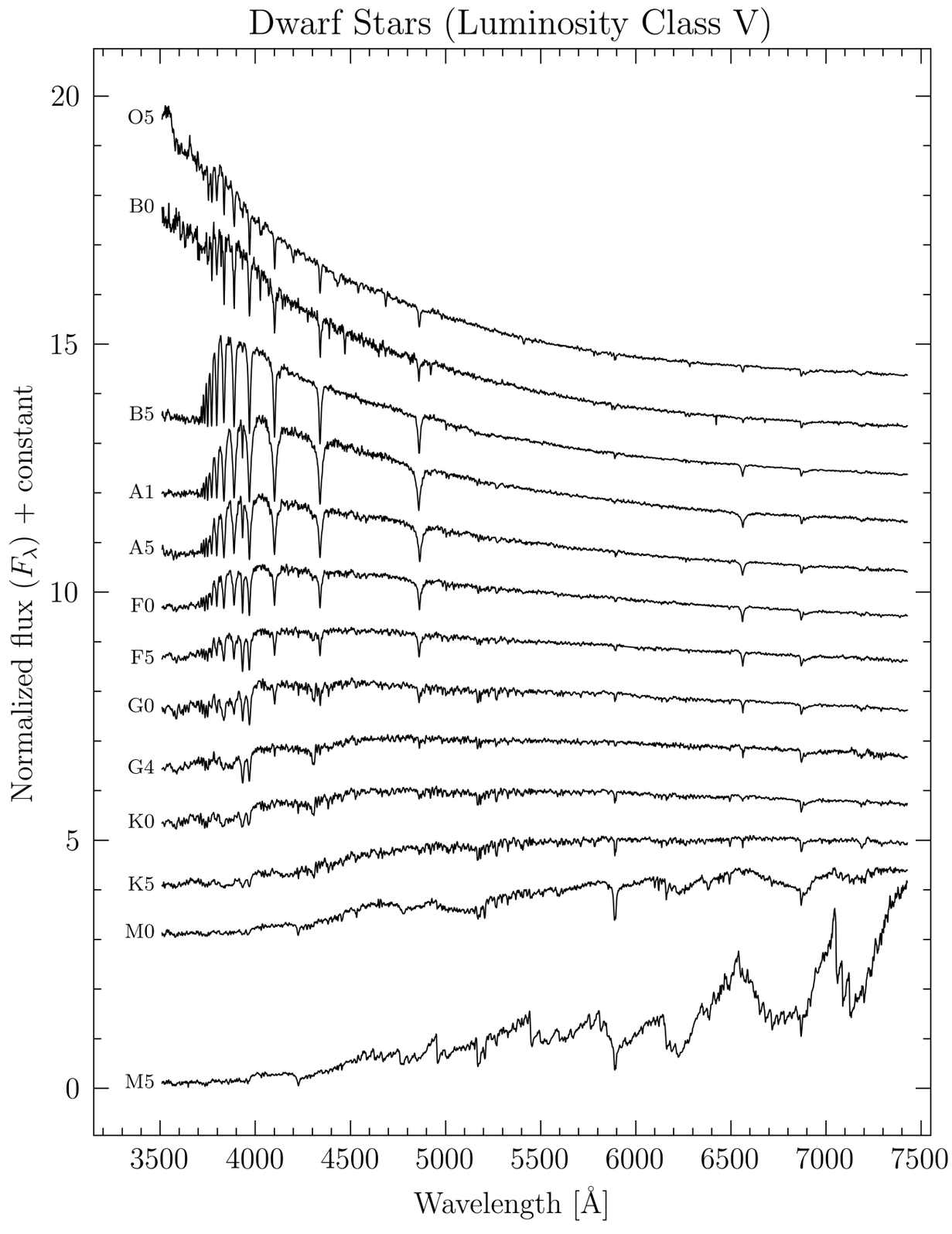
dimensionality reduction algorithm

2 features per object

“Traditional” dimensionality reduction in astronomy

- The stellar sequence:

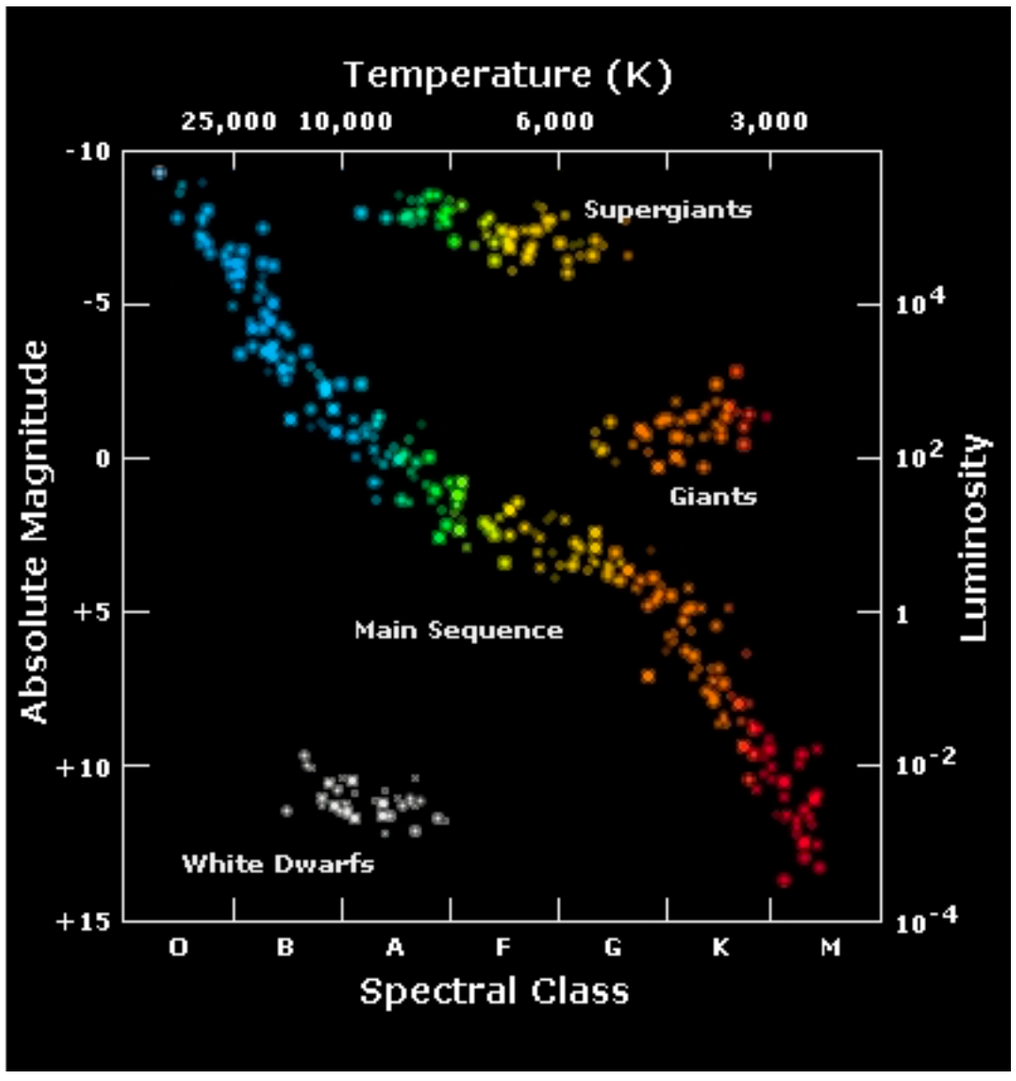


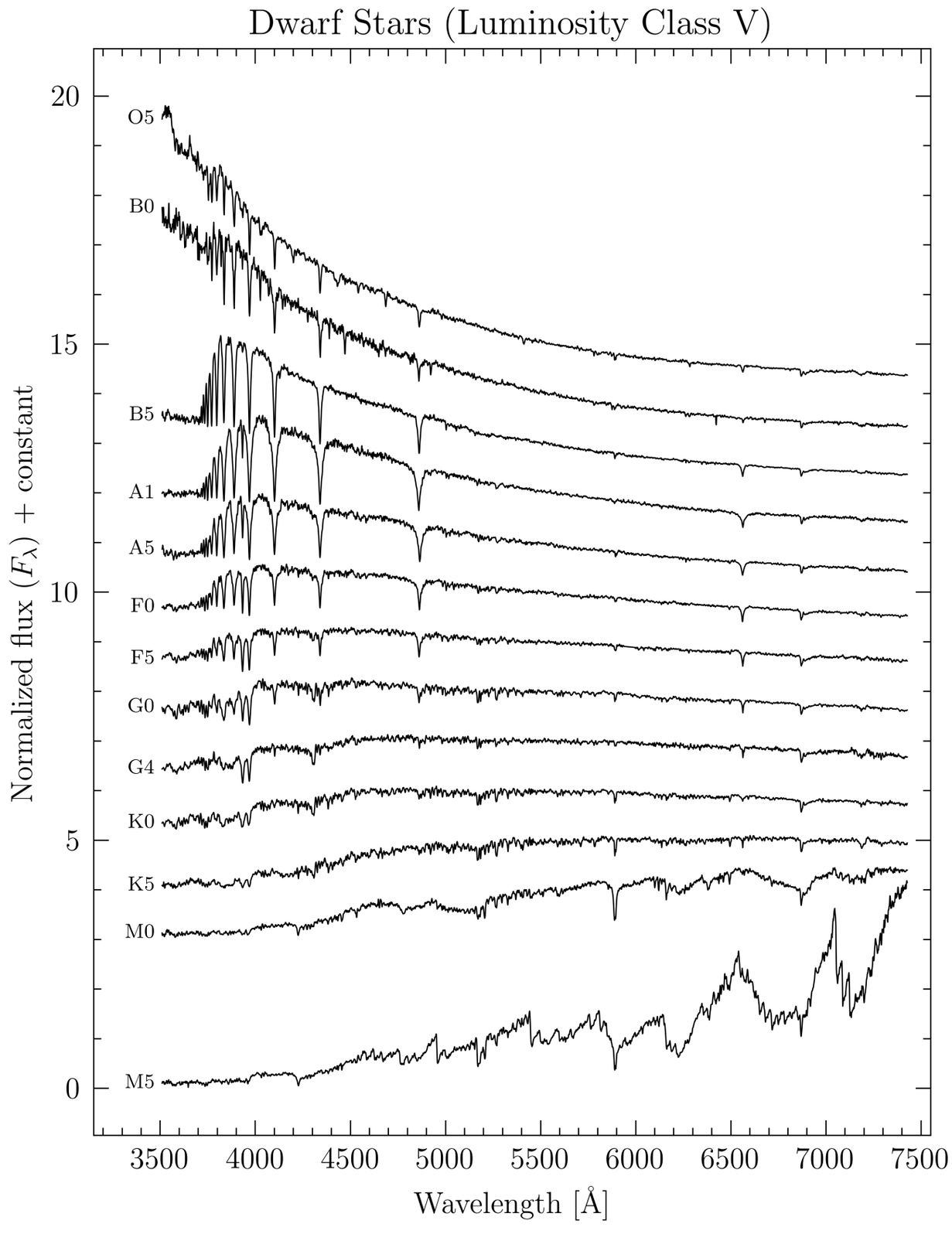


“Traditional” dimensionality reduction in astronomy

- The stellar sequence:

Having a deja-vu?

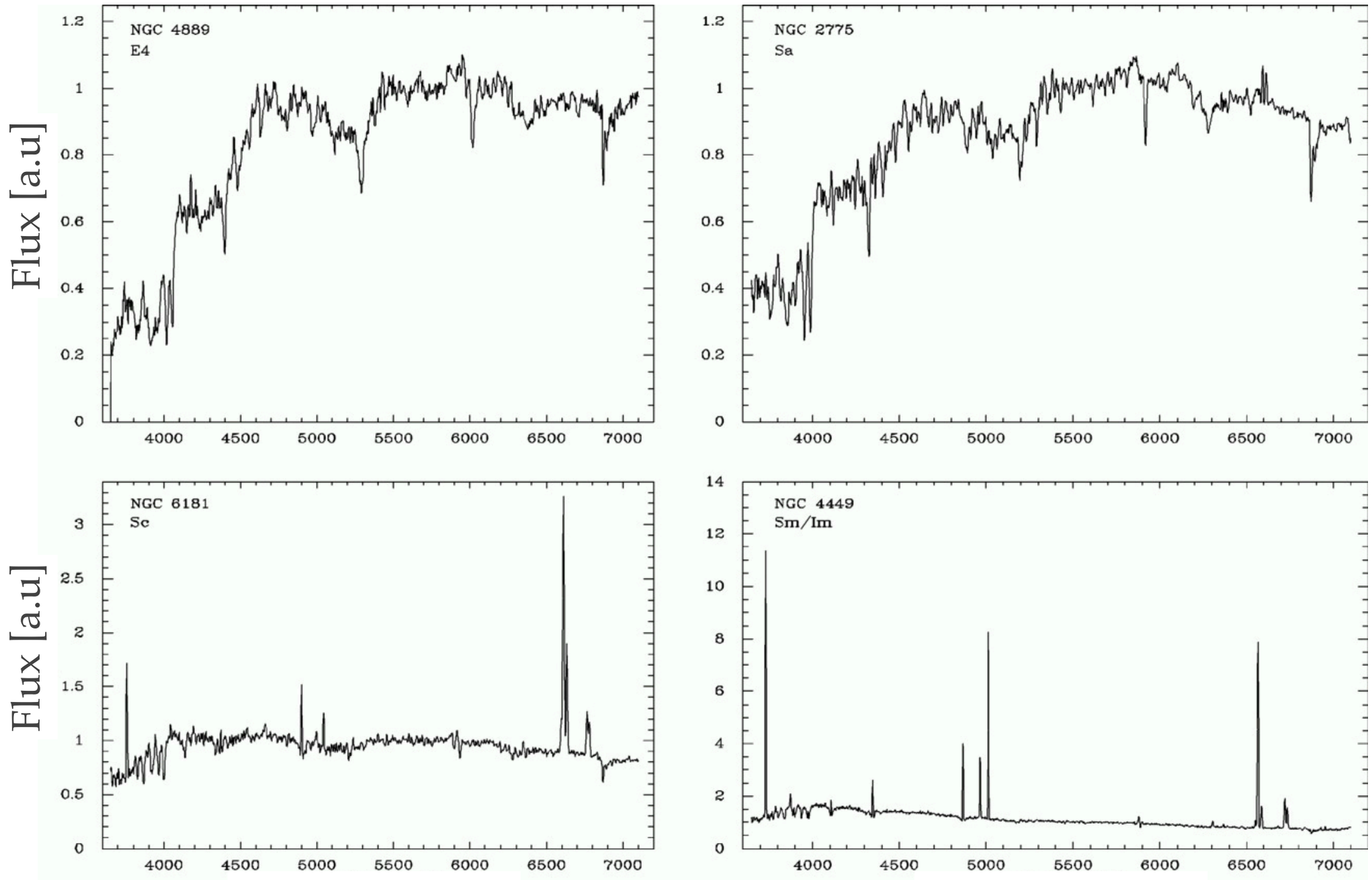




“Traditional” dimensionality reduction in astronomy

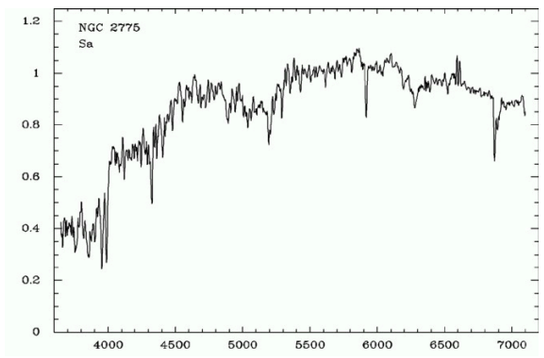
- Galaxy spectra: color-magnitude diagram and the BPT diagram

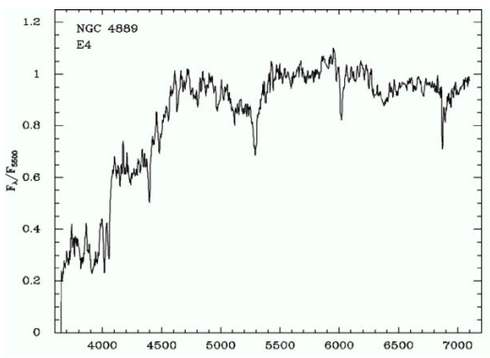
Optical spectra of galaxies (Kennicutt 1998)

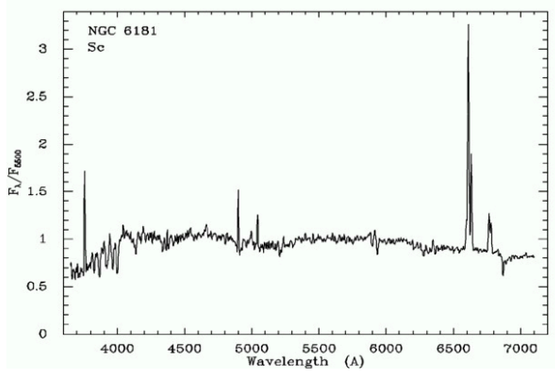


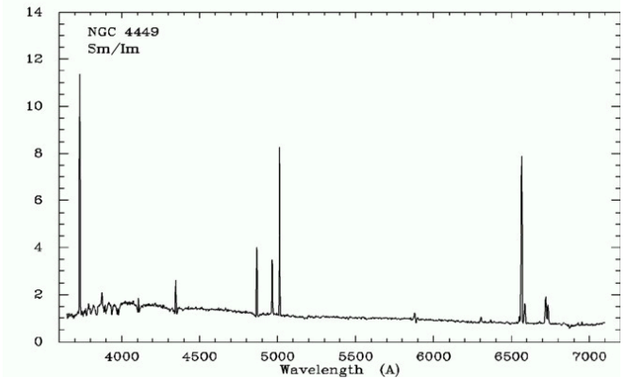
“Traditional” dimensionality reduction in astronomy

- Galaxy spectra: color-magnitude diagram and the BPT diagram

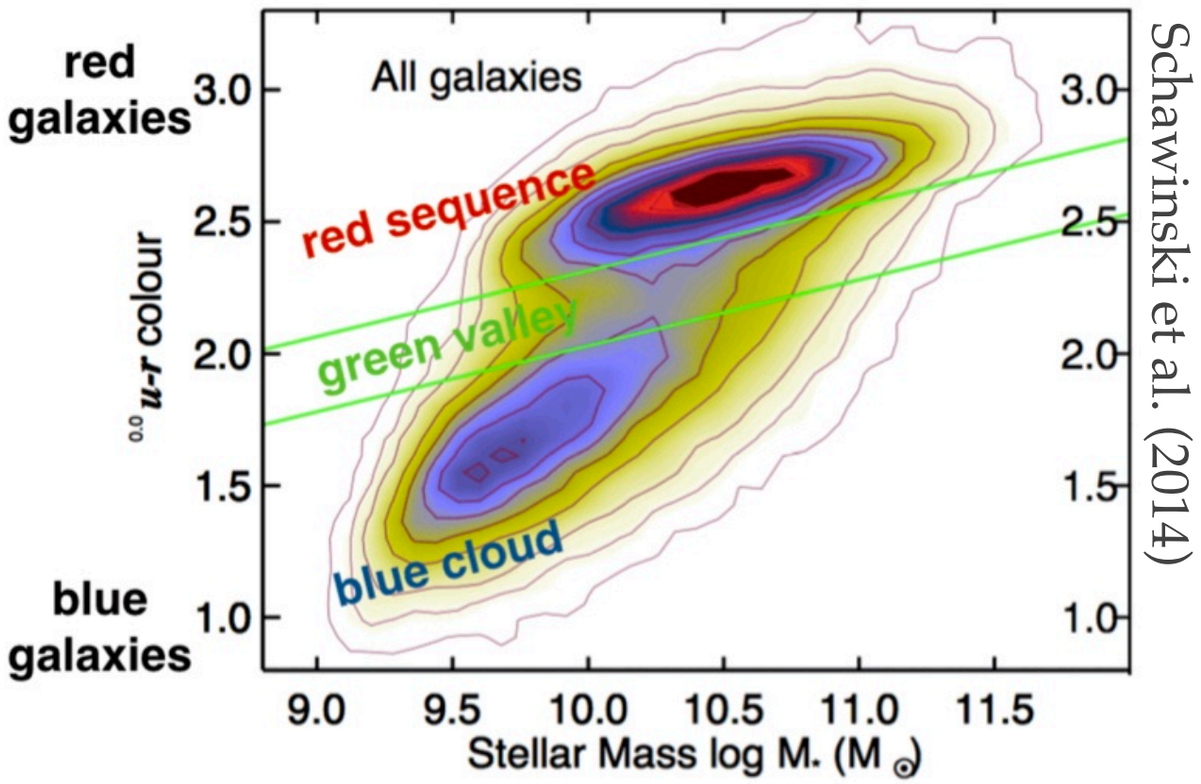




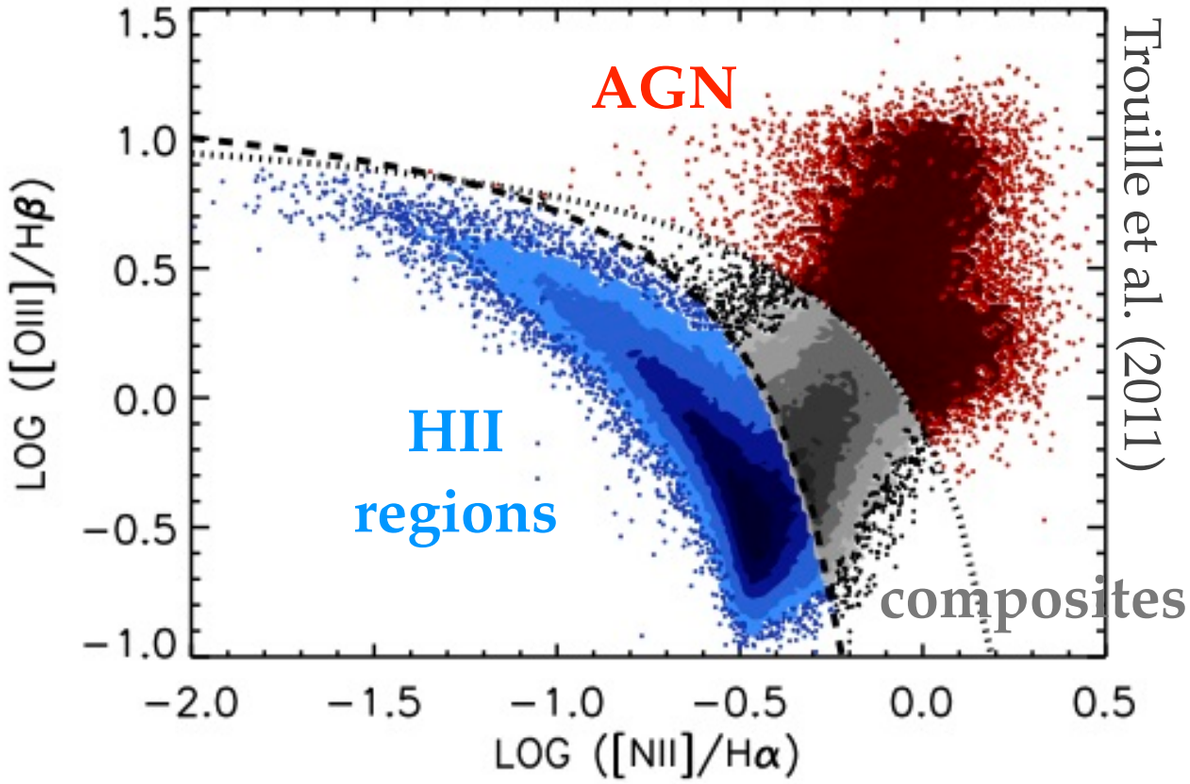




Color-magnitude diagram

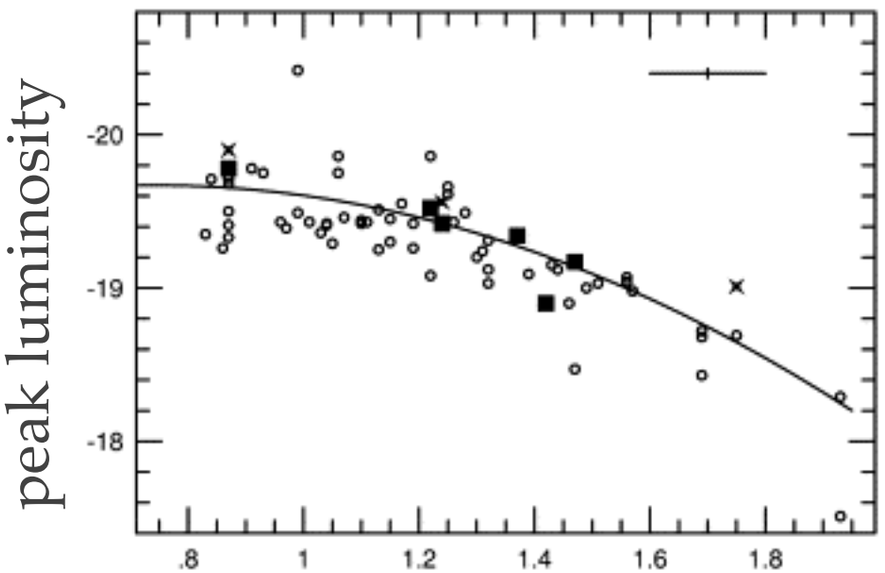


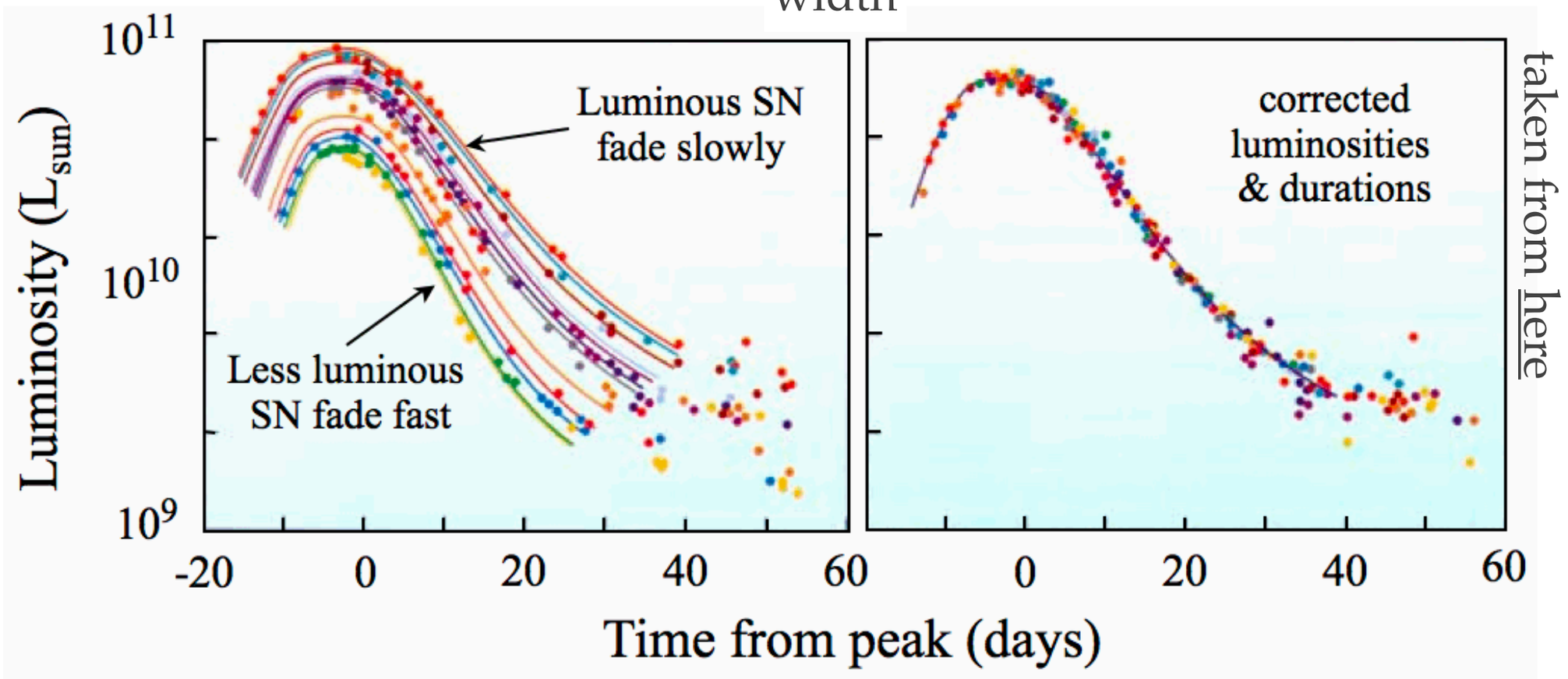
BPT diagram



“Traditional” dimensionality reduction in astronomy

- Type Ia supernovae: Phillips relation





What is fundamentally different now?

1. The volume and rate of information grows exponentially:

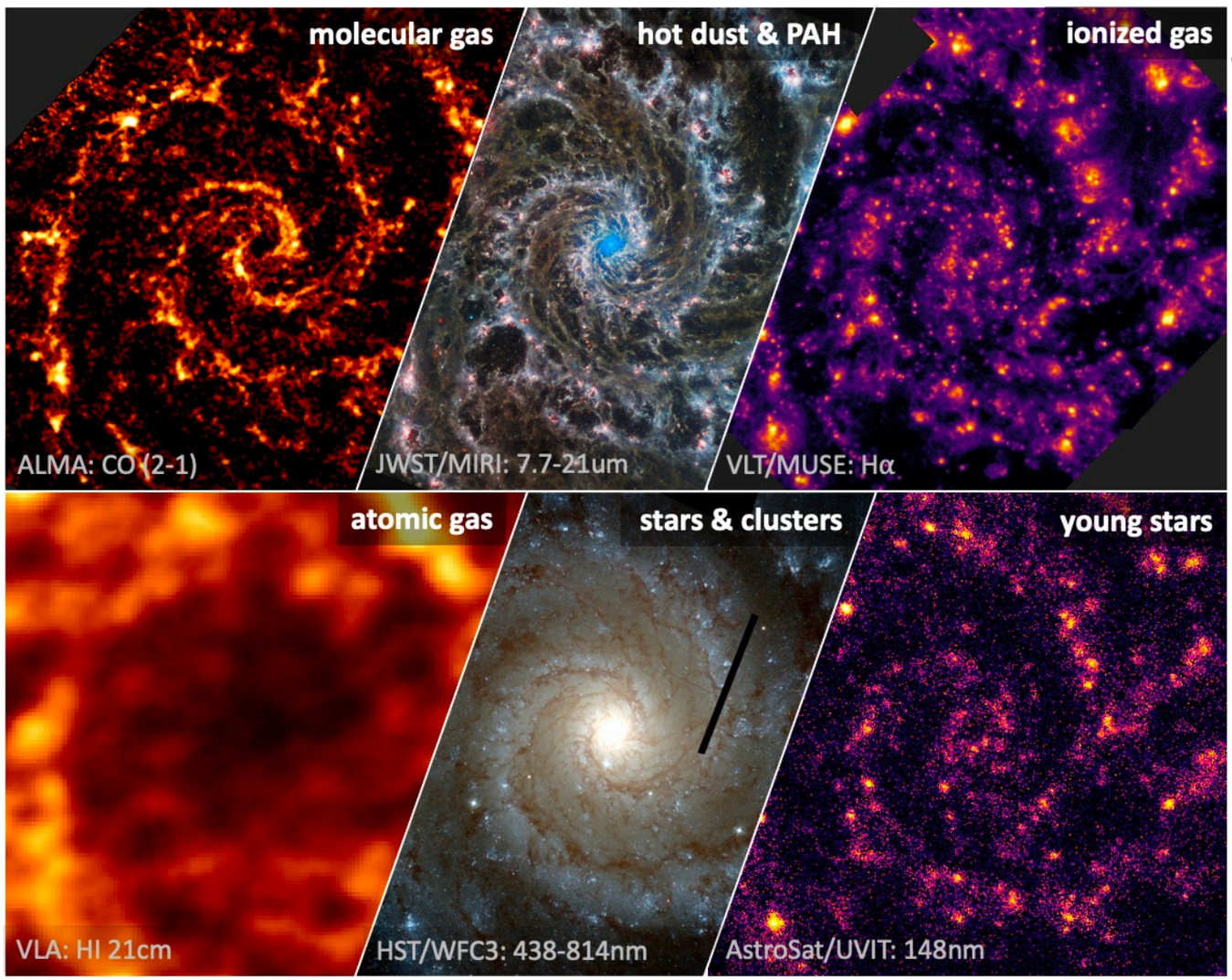
   - Sky survey now generate ~1 PB of data + derived products. Astronomical databases now routinely include $10^6 - 10^9$ objects.

   - We can no longer manually-inspect all the data we collect.

What is fundamentally different now?

2. A great increase in data dimensionality and complexity:

   - Data is heterogeneous and high-dimensional.
   - Patterns and correlations in the data can no longer be visualized in 3D.



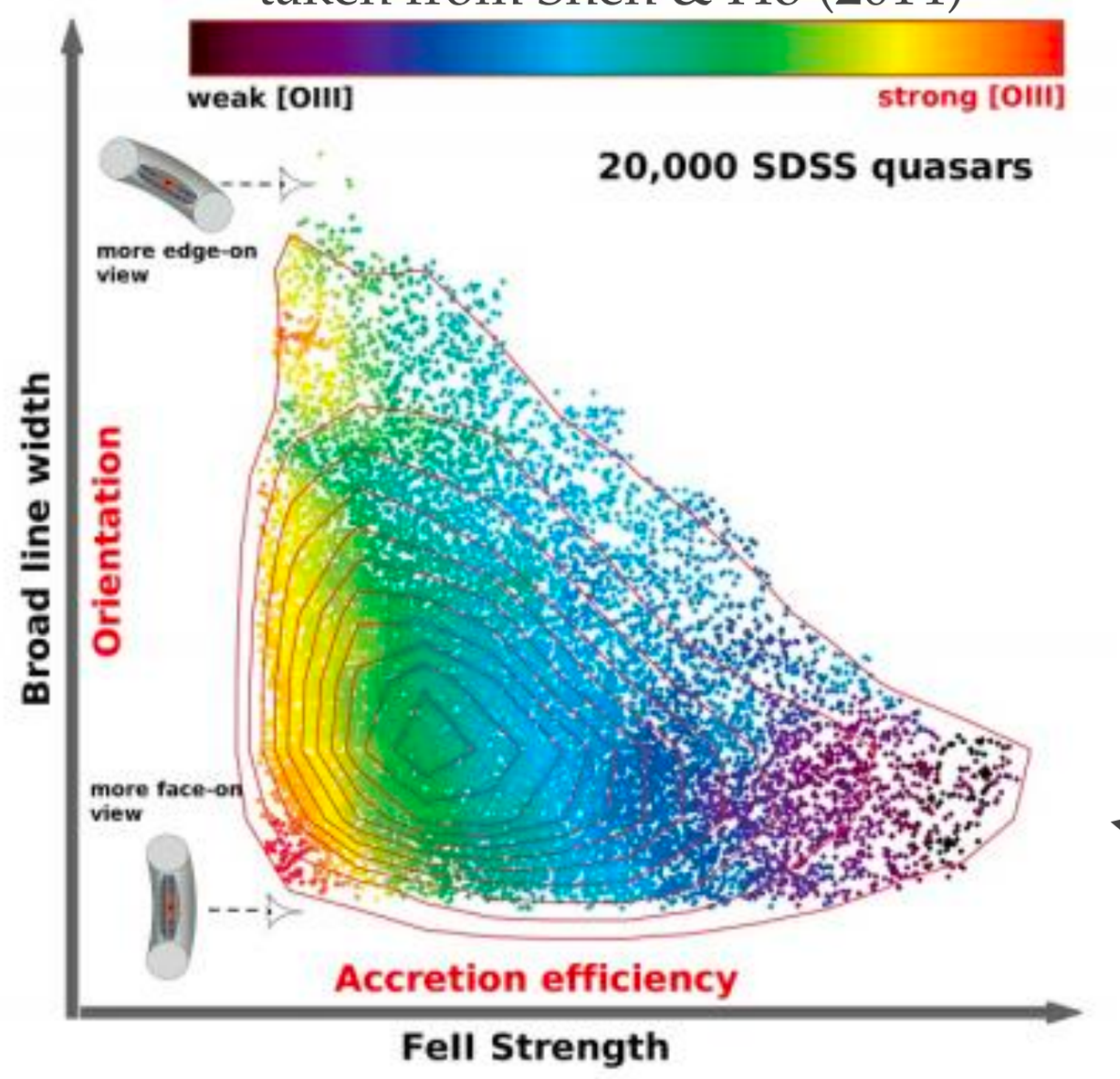
By J. Sun

Image by the PHANGS collaboration

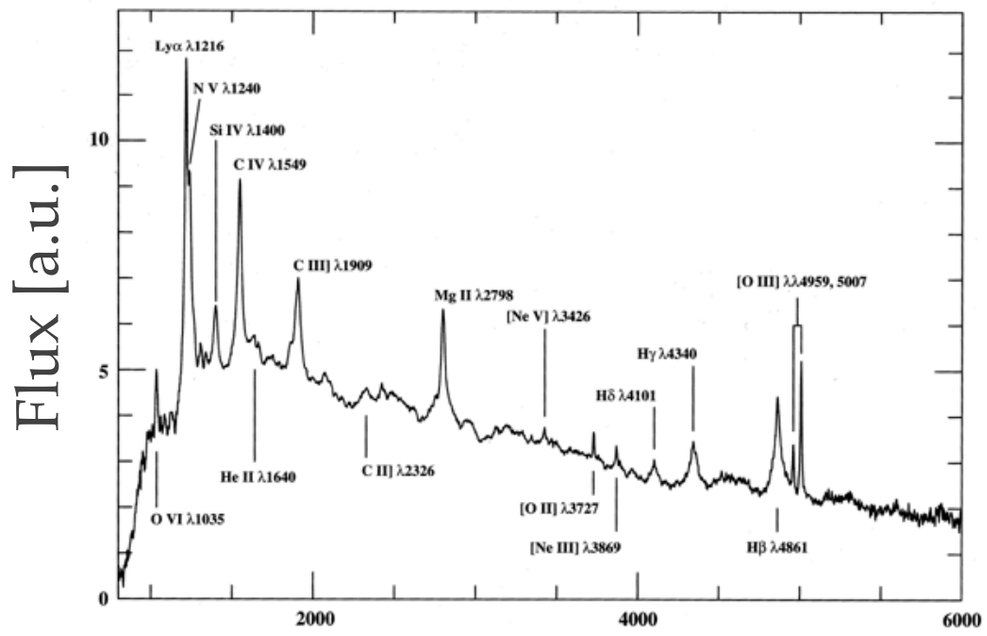
More recent examples of dimensionality reduction

- PCA dimensionality reduction of quasar spectra (Boroson & Green 1992):

taken from Shen & Ho (2014)



A set of quasar spectra



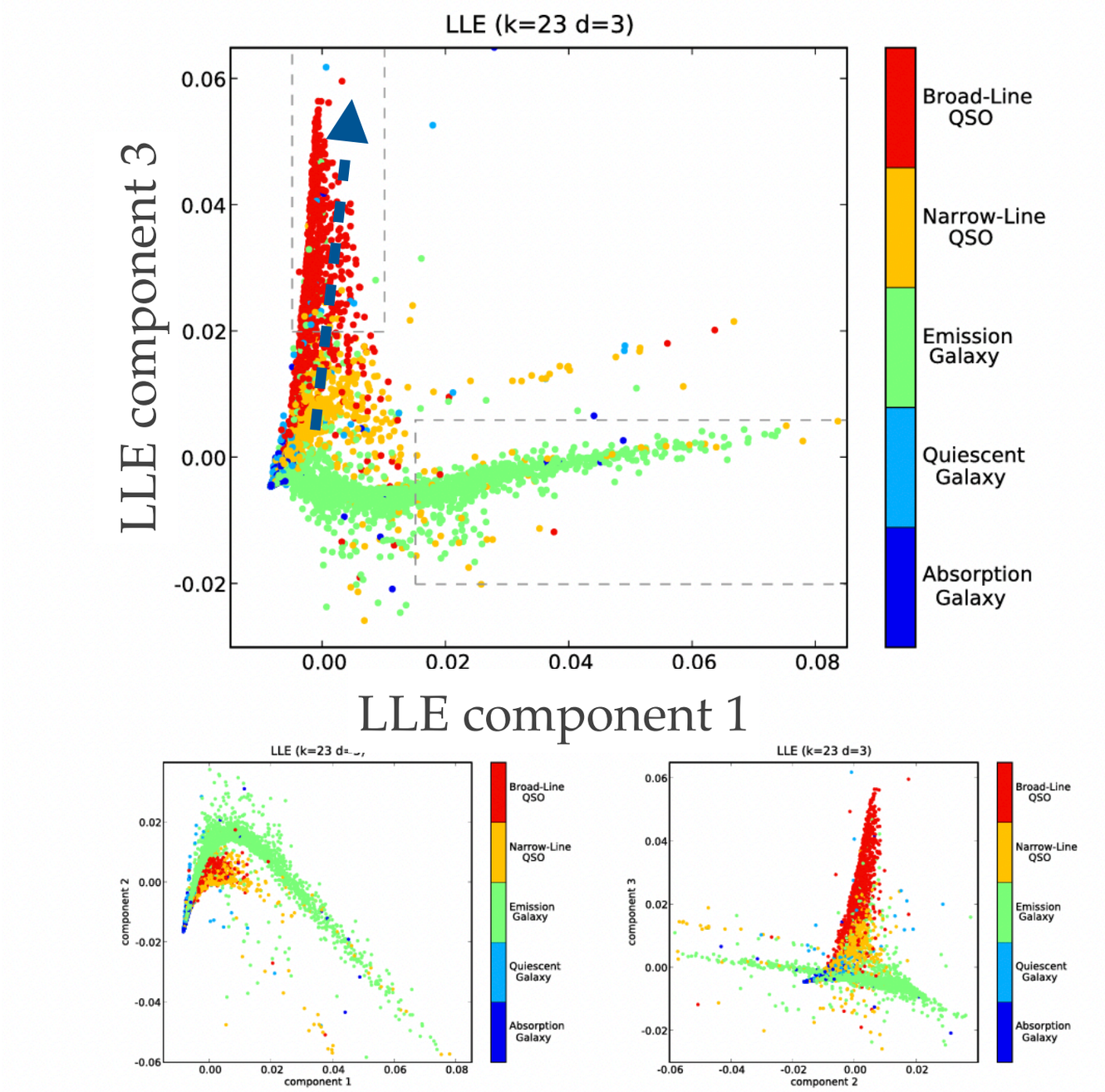
Wavelength

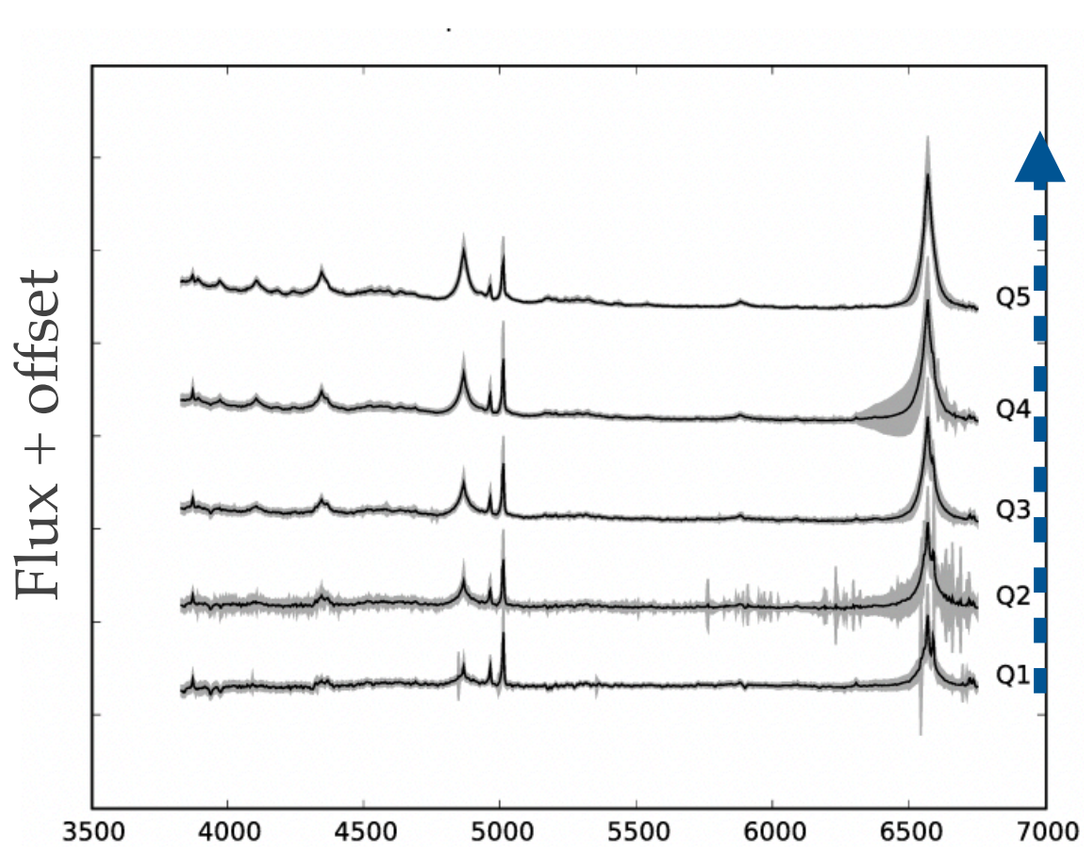
A set of derived features

PCA

More recent examples of dimensionality reduction

- Locally Linear Embedding of optical galaxy spectra (Vanderplas & Connolly 2009):





Different use cases

- Uncover new trends/correlations.
- Data visualization and interpretation.
- Look for outliers or interesting objects.

Different use cases

- Uncover new trends/correlations.
- Data visualization and interpretation.
- Look for outliers or interesting objects.
- Improve performance of supervised machine learning:
  - Original features can be correlated and redundant.
  - Most algorithms cannot handle thousands of features.

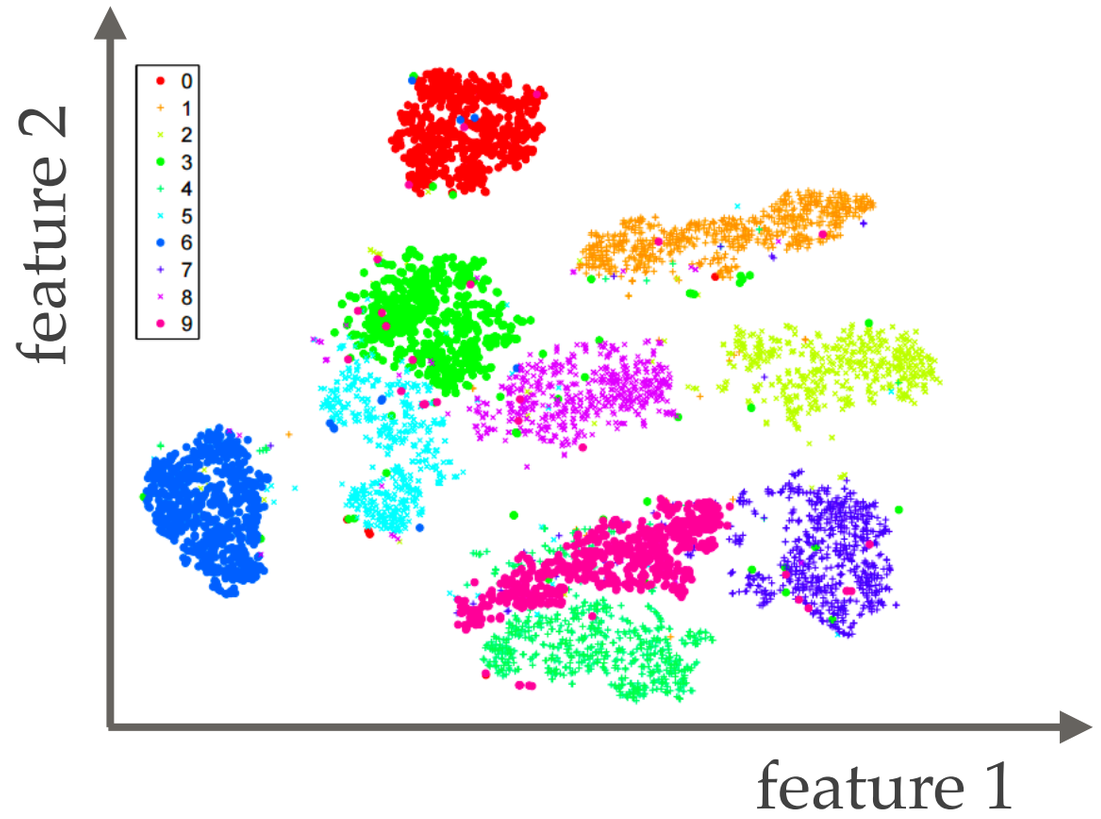
Different use cases

- Uncover new trends/correlations.
- Data visualization and interpretation.
- Look for outliers or interesting objects.
- Improve performance of supervised machine learning:
  - Original features can be correlated and redundant.
  - Most algorithms cannot handle thousands of features.
- Compressing data (e.g., the Square Kilometre Array; SKA).

Two types of outputs

1. Low-dimensional embedding of our dataset.

This is a common output of all dimensionality reduction algorithms: PCA, ICA, NNMF, LLE, SOM, tSNE, UMAP, etc.



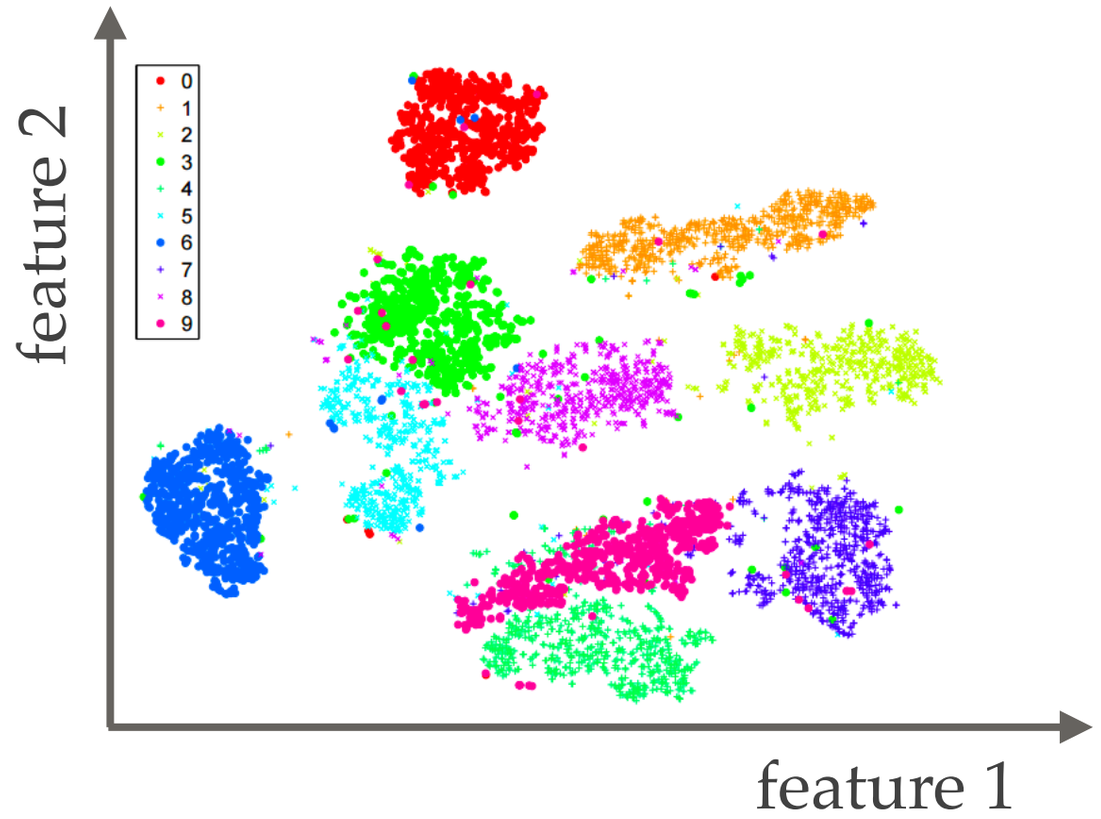
2. Prototypes or Eigen-vectors.

Along with the low-dimensional embedding, this is the output of: PCA, ICA, NNMF, and SOM.

Two types of outputs

1. Low-dimensional embedding of our dataset.

This is a common output of all dimensionality reduction algorithms: PCA, ICA, NNMF, LLE, SOM, tSNE, UMAP, etc.



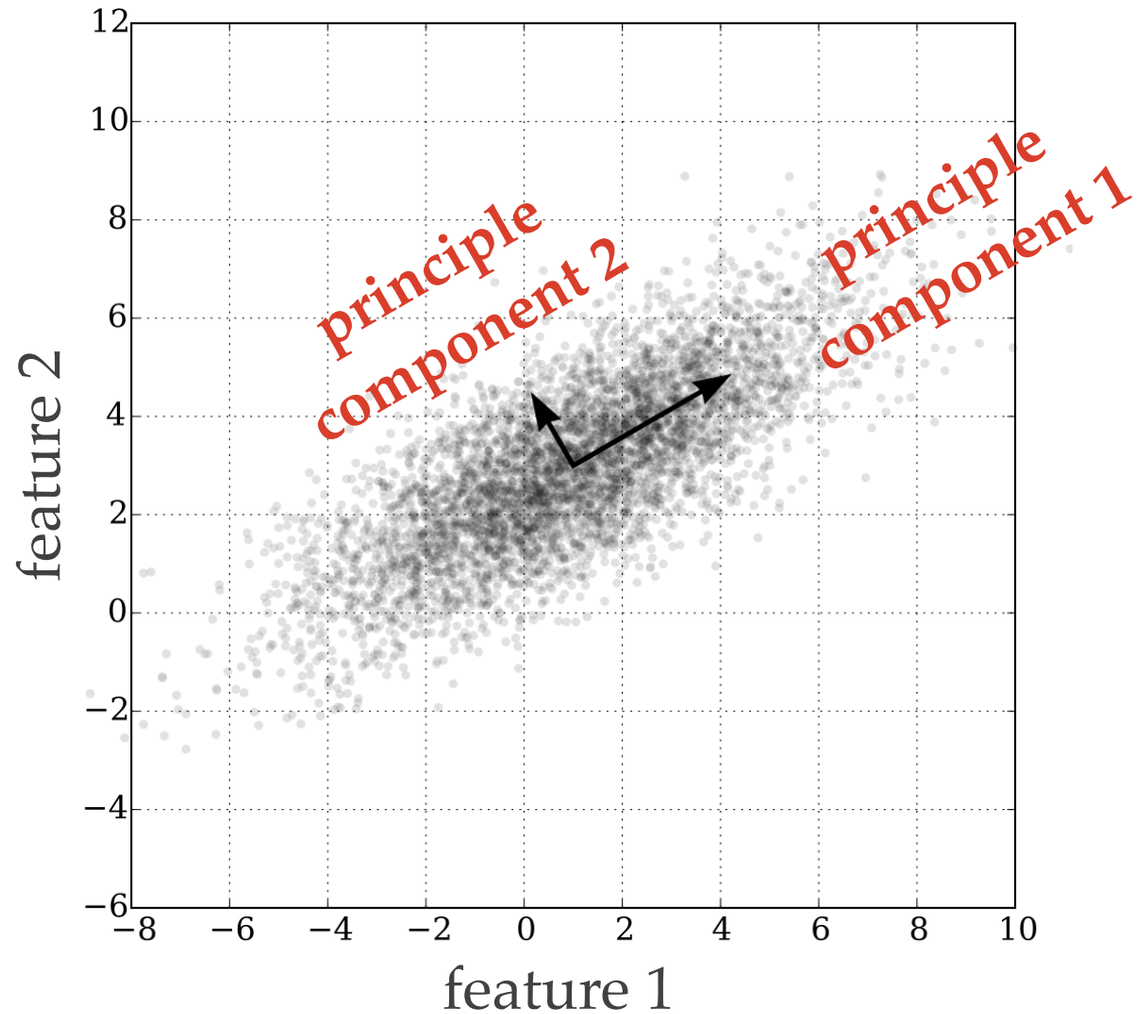
2. Prototypes or Eigen-vectors.

Along with the low-dimensional embedding, this is the output of: PCA, ICA, NNMF, and SOM.

Can be useful when trying to interpret the resulting embedding!

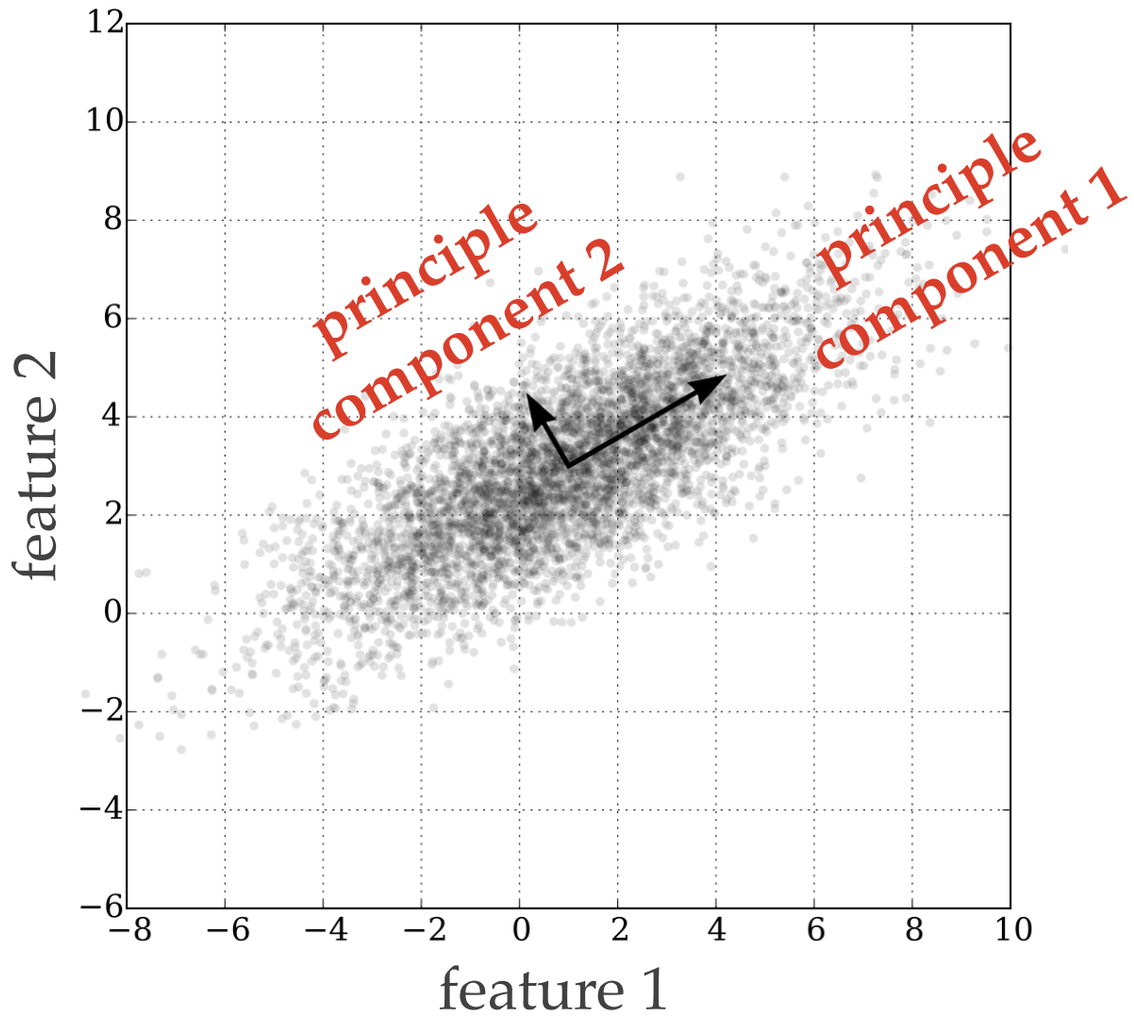
Principle Component Analysis (PCA)

- PCA is one of the most popular dimensionality reduction algorithms in machine learning and data science.
- It is a linear transformation of the input data into a new coordinate system, defined by the *principle components*.



Principle Component Analysis (PCA)

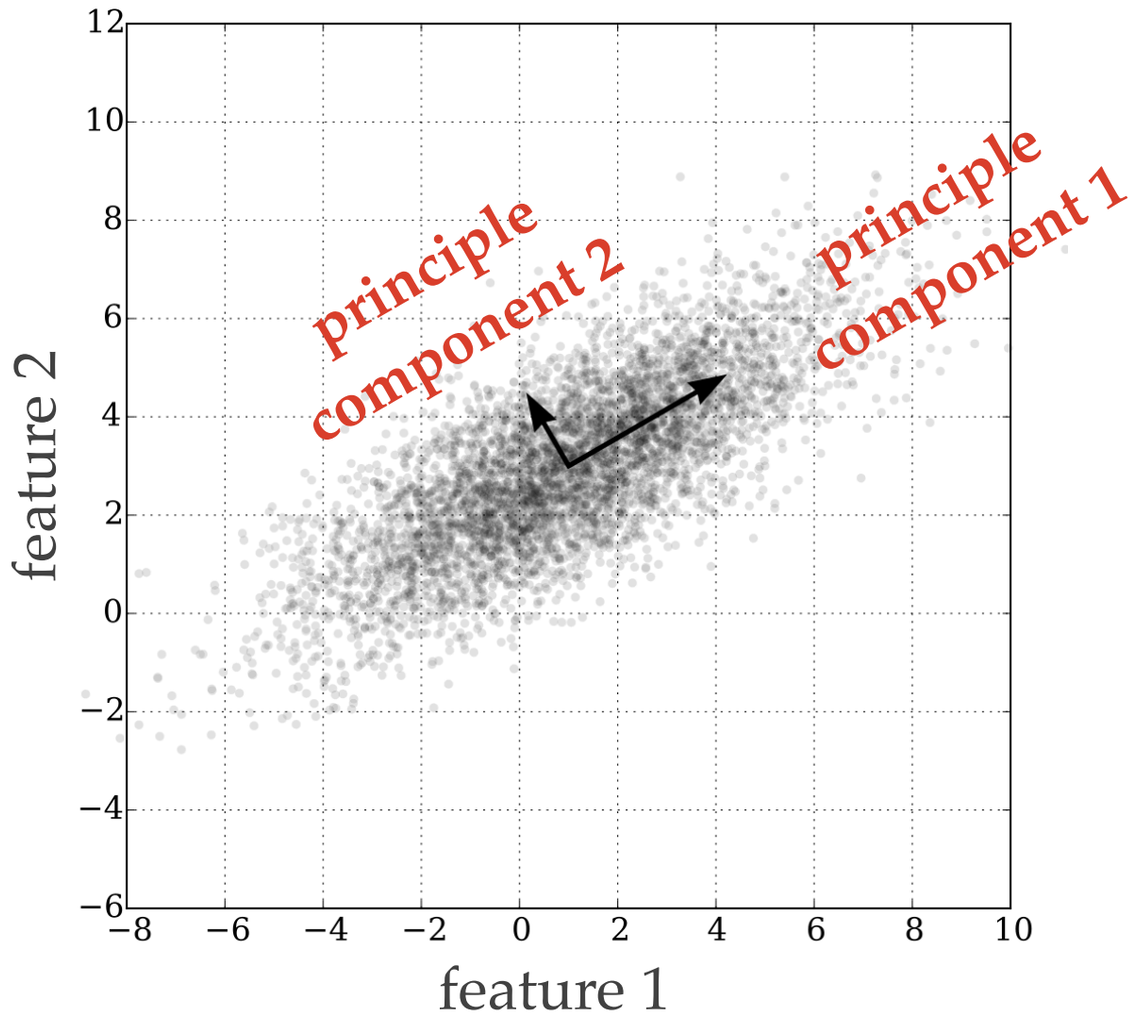
- PCA is one of the most popular dimensionality reduction algorithms in machine learning and data science.
- It is a linear transformation of the input data into a new coordinate system, defined by the *principle components*.



- The first principle component has the largest possible **variance**.

Principle Component Analysis (PCA)

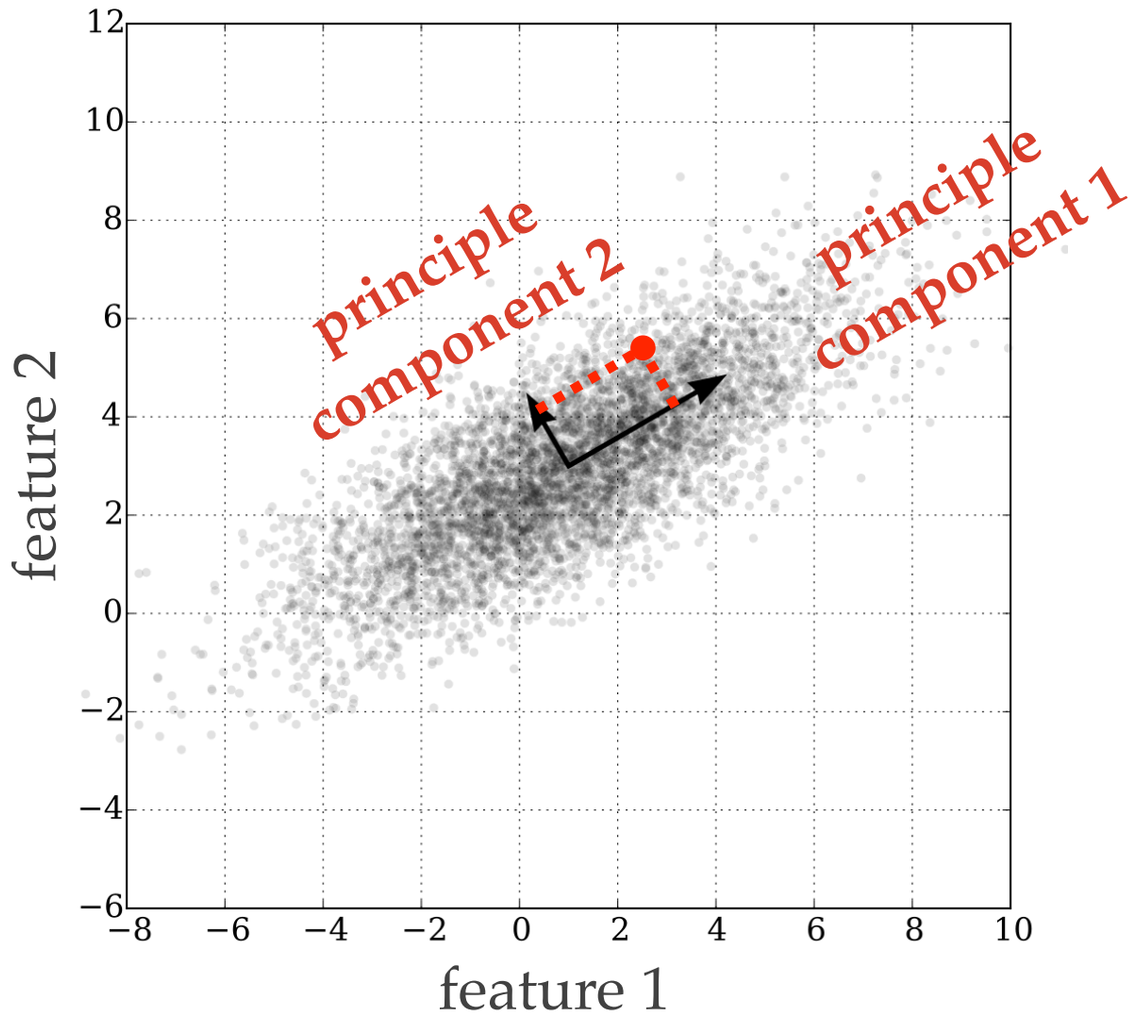
- PCA is one of the most popular dimensionality reduction algorithms in machine learning and data science.
- It is a linear transformation of the input data into a new coordinate system, defined by the *principle components*.



- 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)

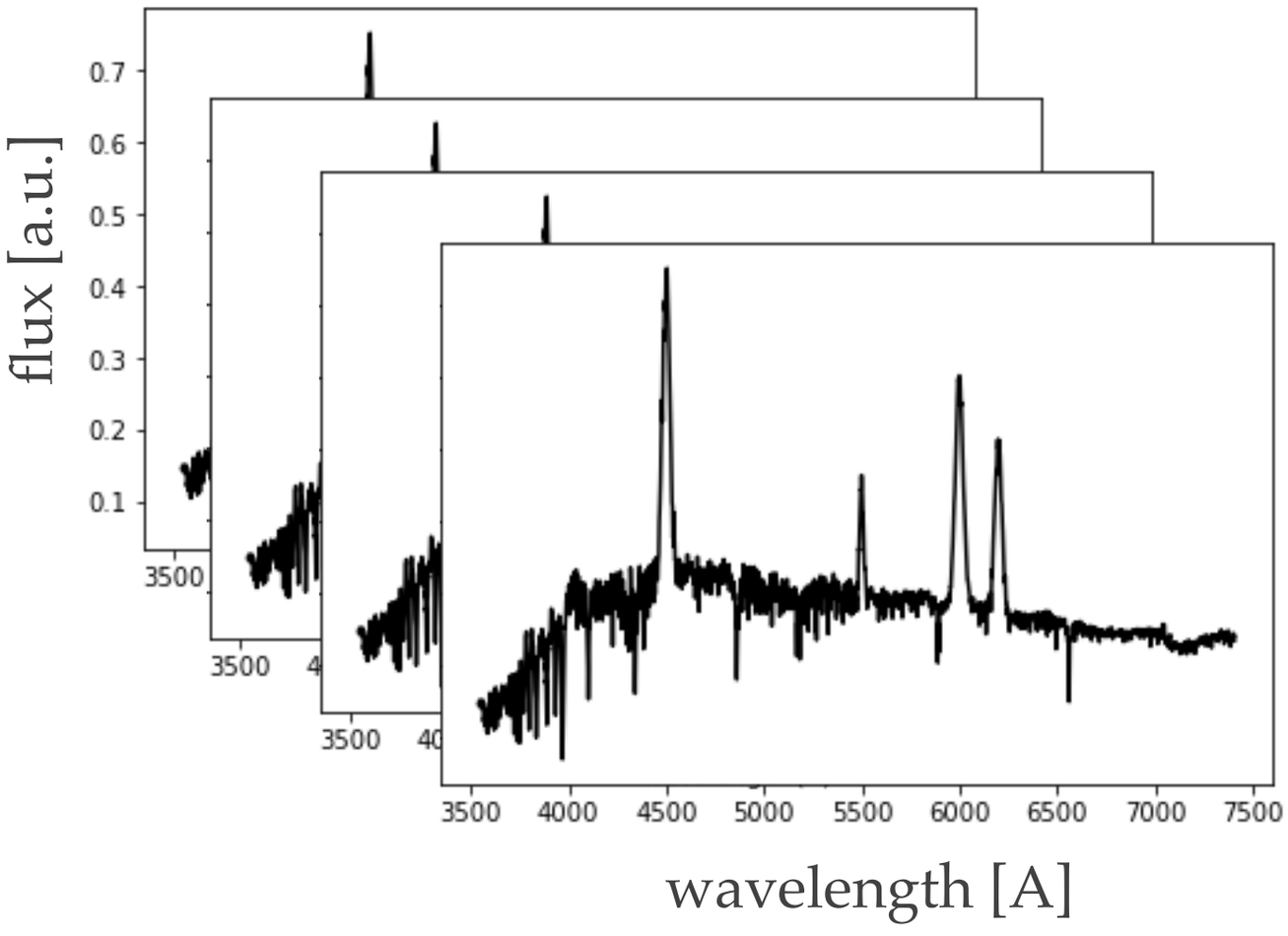
- PCA is one of the most popular dimensionality reduction algorithms in machine learning and data science.
- It is a linear transformation of the input data into a new coordinate system, defined by the *principle components*.



- 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.
- Every object in the input data is described as a n-dimensional point in this new coordinate system.

PCA - step by step

- We will go through the basic steps of PCA using an example of simulated galaxy spectra [see Jupyter notebook!]:



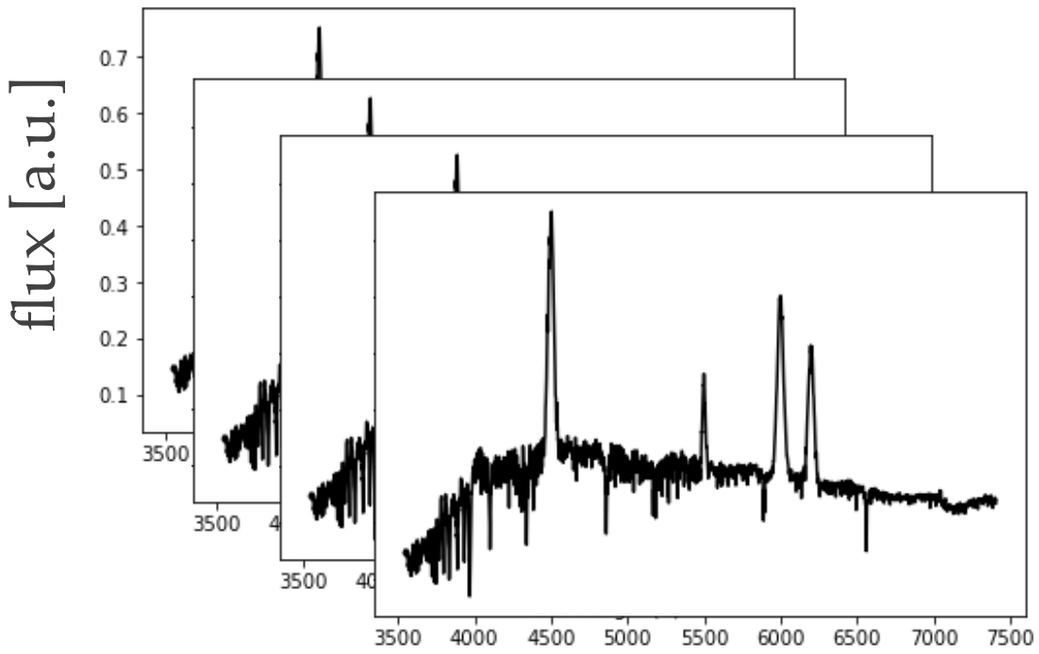
- There are 1,000 spectra.
- Each spectrum has 4300 flux measurements => each object in the input is a 4300-dimensional object.

Our goal is use PCA to reduce the dimensions of this high-dimensional dataset.

PCA - step by step

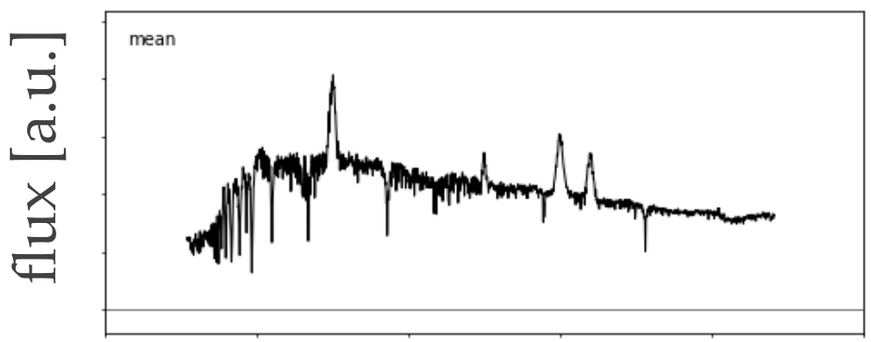
Step I - calculate and subtract the mean of the different objects.

Z =



-

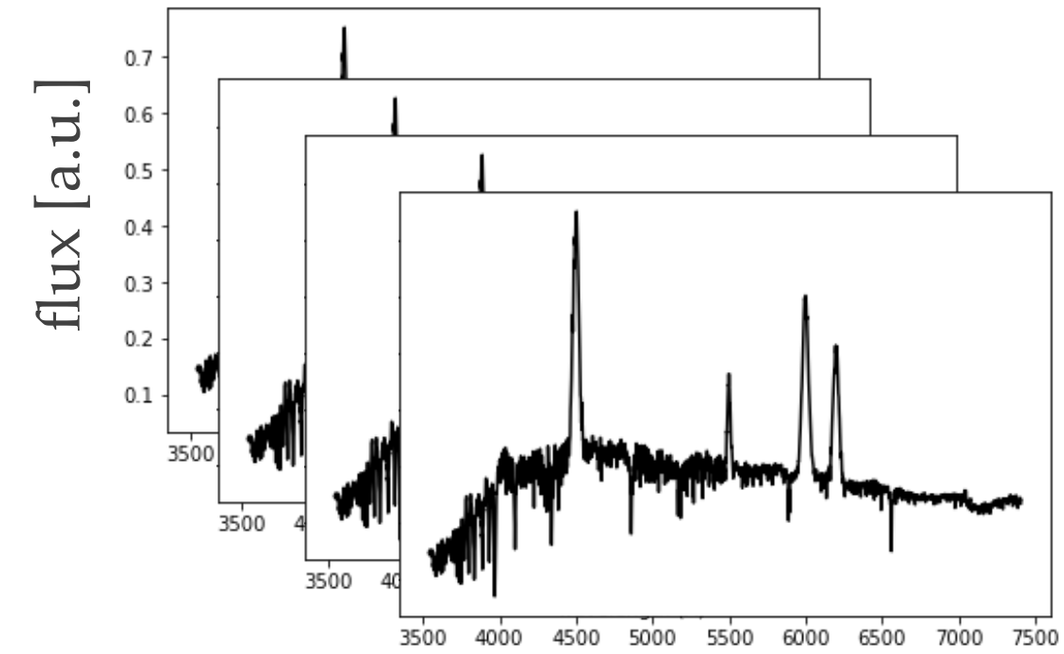
Mean spectrum



PCA - step by step

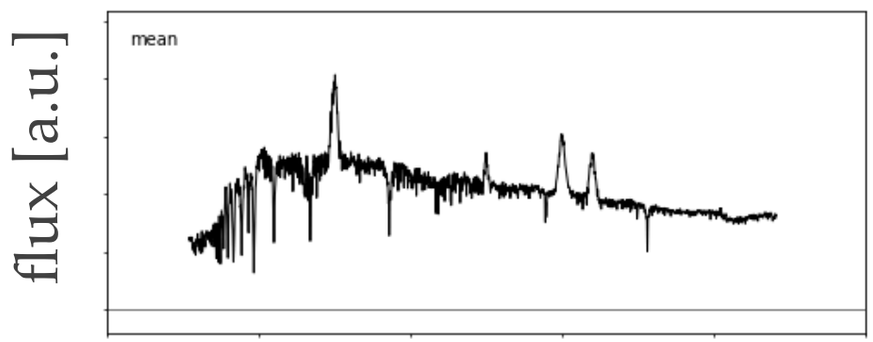
Step I - calculate and subtract the mean of the different objects.

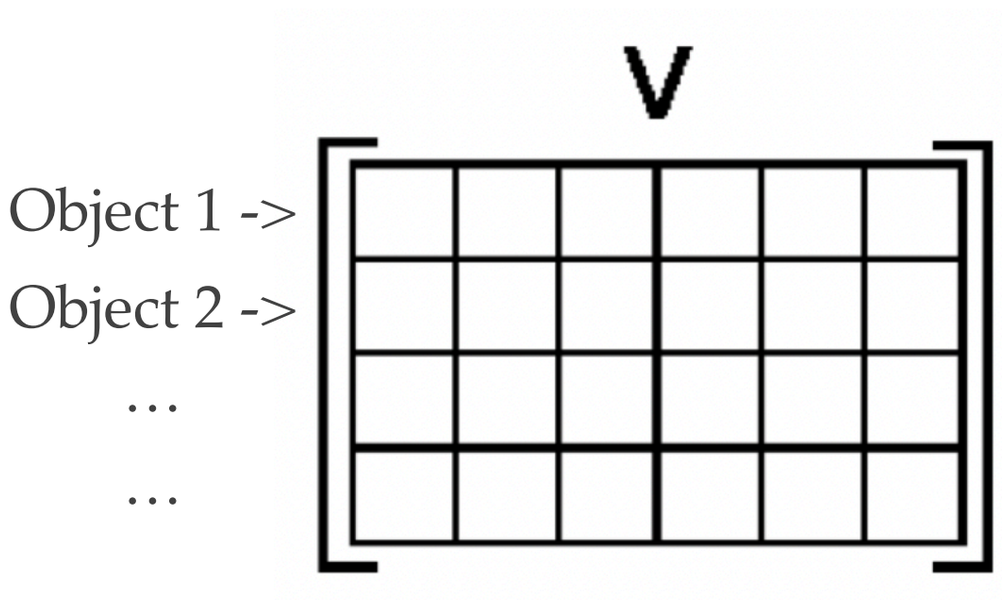
Z =



-

Mean spectrum





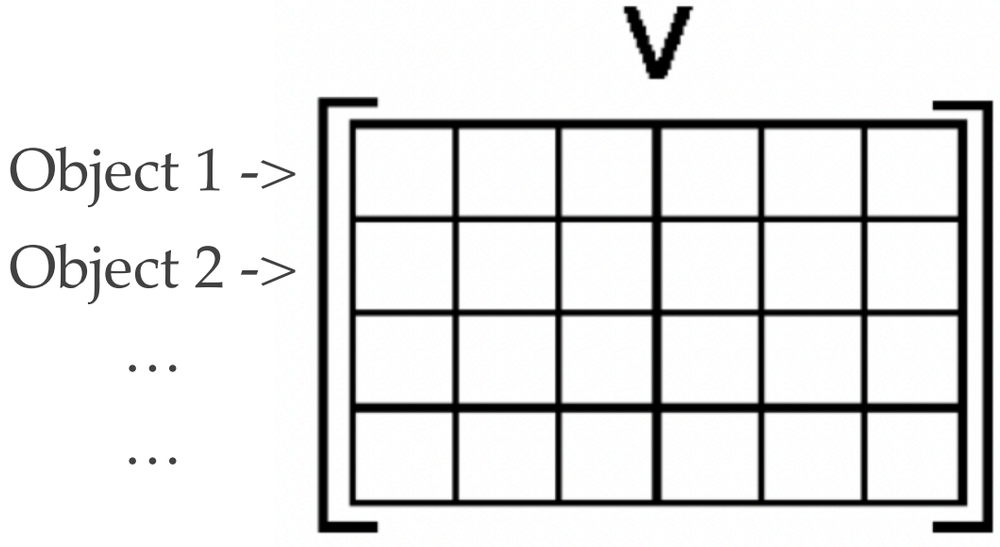
PCA - step by step

Step I - calculate and subtract the mean of the different objects.

Z =

-

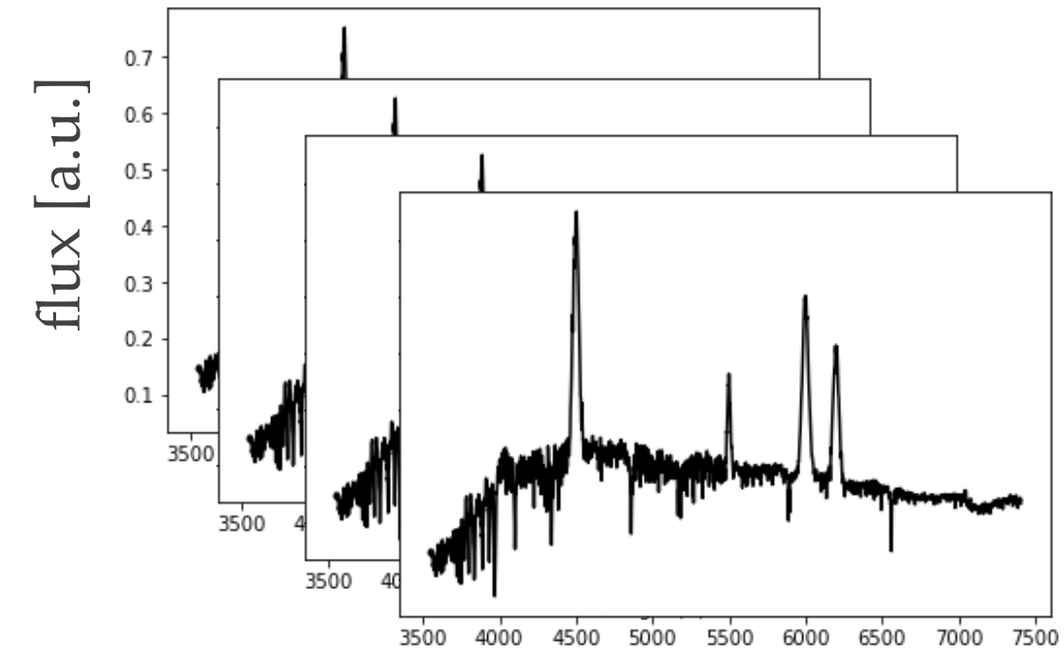
Mean image



PCA - step by step

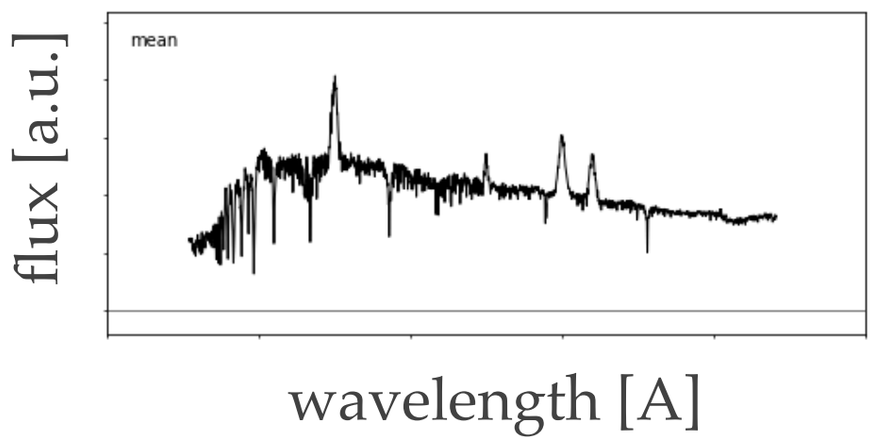
Step I - calculate and subtract the mean of the different objects.

$Z =$



$-$

Mean spectrum

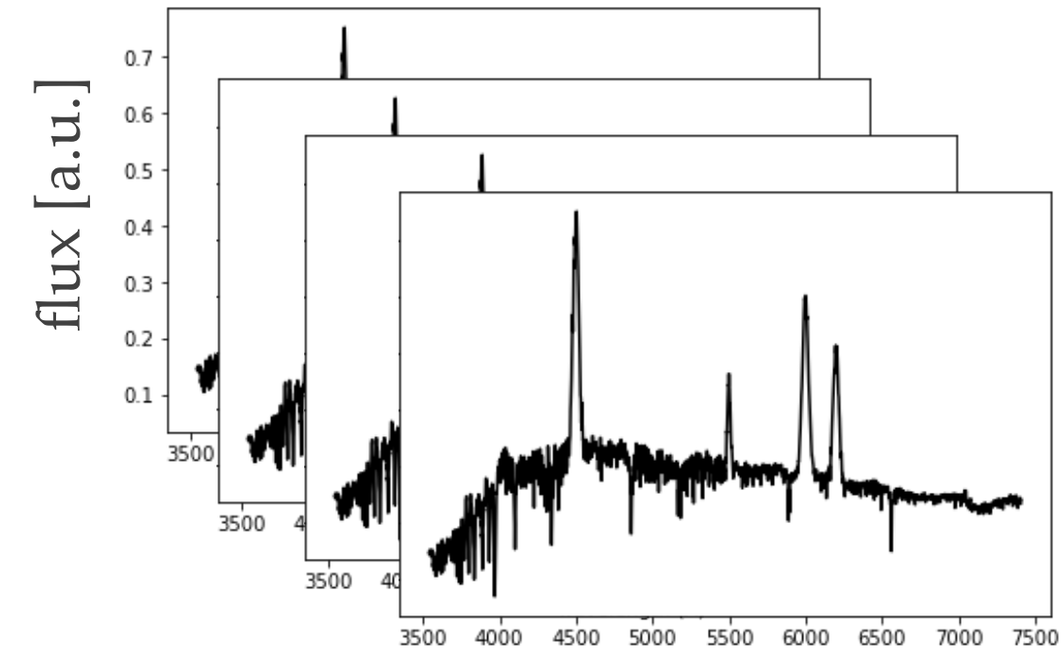


Step II - calculate the covariance matrix of Z: $C = Z^T Z$

PCA - step by step

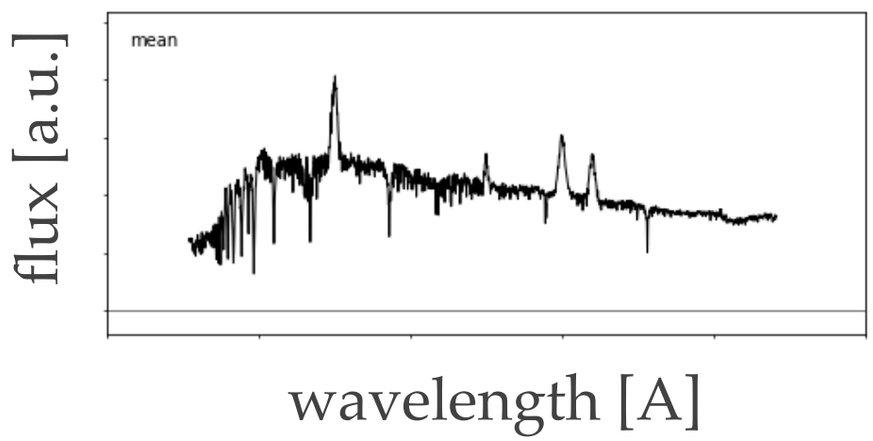
Step I - calculate and subtract the mean of the different objects.

$Z =$



$-$

Mean spectrum



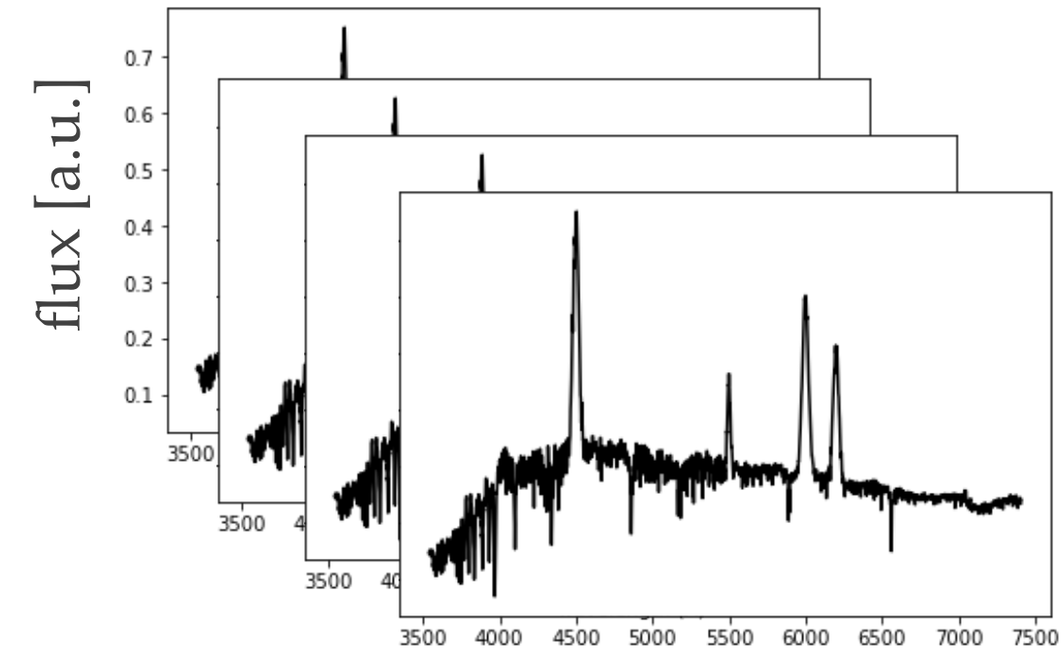
Step II - calculate the covariance matrix of Z: $C = Z^T Z$

Step III - calculate the eigenvectors and the corresponding eigenvalues of the covariance matrix: $Z^T Z = PDP^{-1}$

PCA - step by step

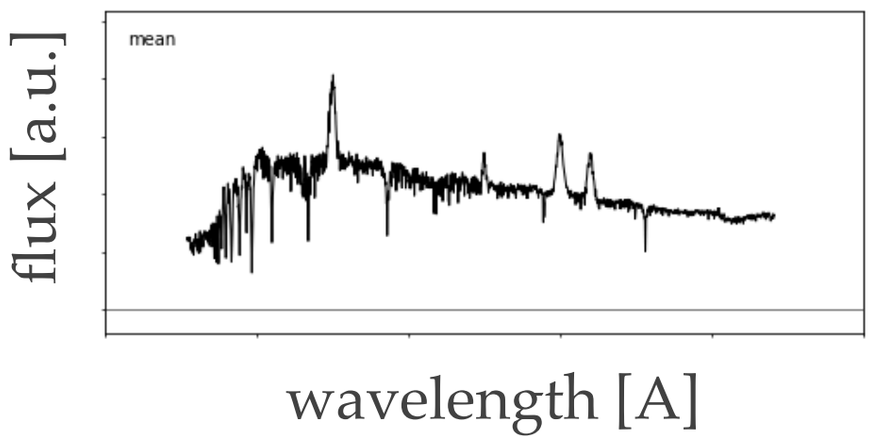
Step I - calculate and subtract the mean of the different objects.

$Z =$



$-$

Mean spectrum



Step II - calculate the covariance matrix of Z: $C = Z^T Z$

Step III - calculate the eigenvectors and the corresponding eigenvalues of the covariance matrix: $Z^T Z = PDP^{-1}$

Diagonal matrix containing eigenvalues ordered from the largest to the smallest

PCA - step by step

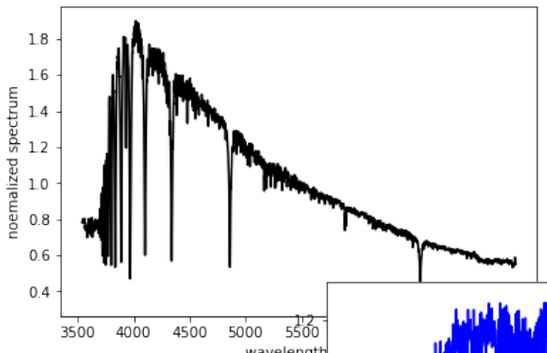
Step III - calculate the eigenvectors and the corresponding eigenvalues of the covariance matrix:

$$Z^T Z = PDP^{-1}$$

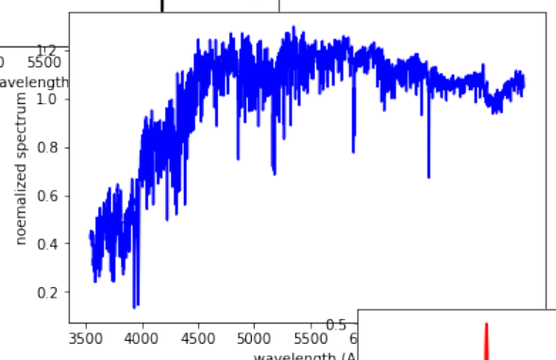
The matrix of the eigenvectors, or 'loadings'

Diagonal matrix containing eigenvalues ordered from the largest to the smallest

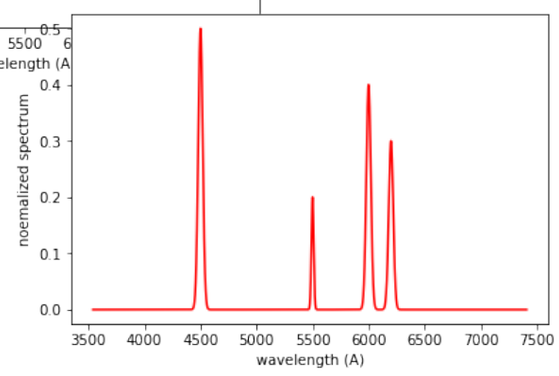
PC1



PC2



PC3



PCA - step by step

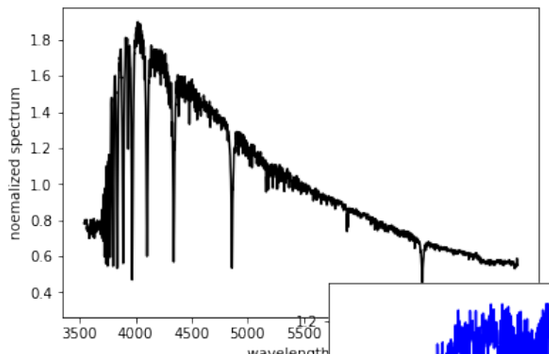
Step III - calculate the eigenvectors and the corresponding eigenvalues of the covariance matrix:

$$Z^T Z = PDP^{-1}$$

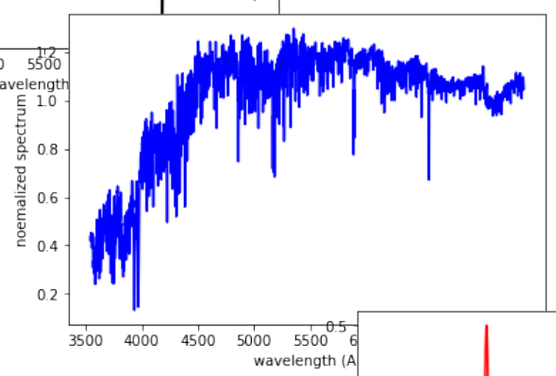
The matrix of the eigenvectors, or 'loadings'

Diagonal matrix containing eigenvalues ordered from the largest to the smallest

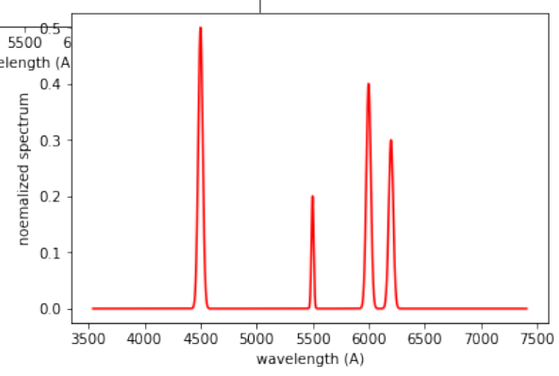
PC1

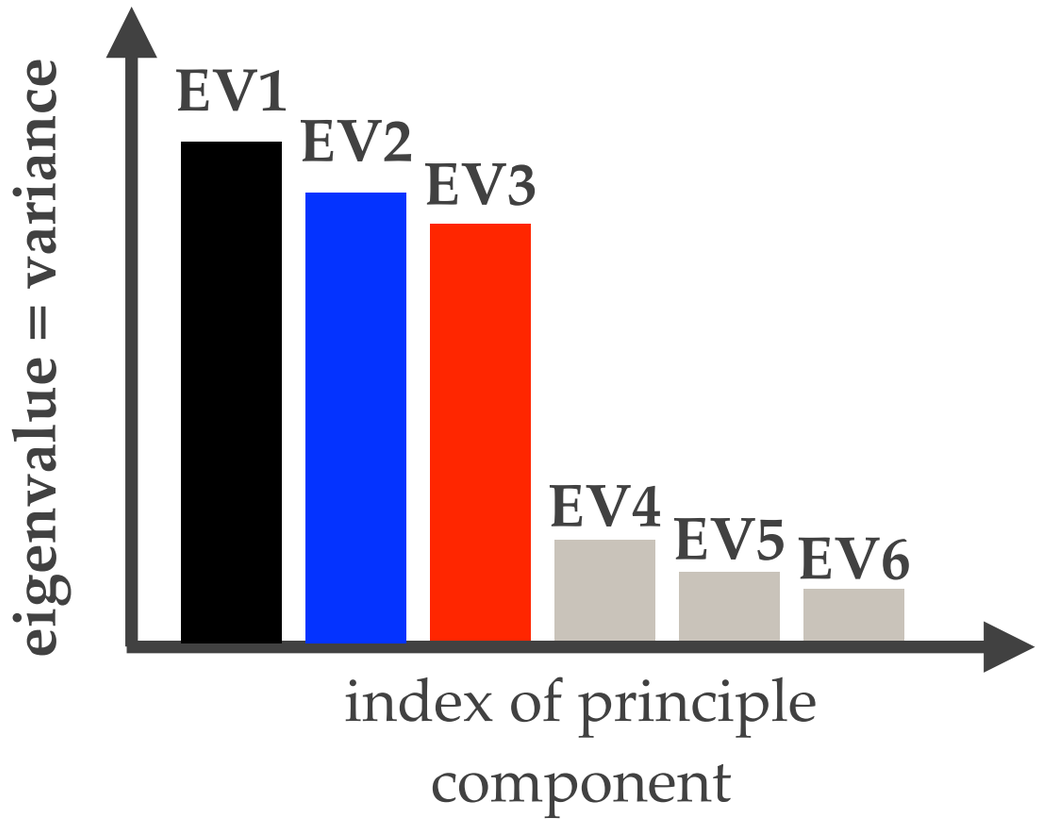


PC2



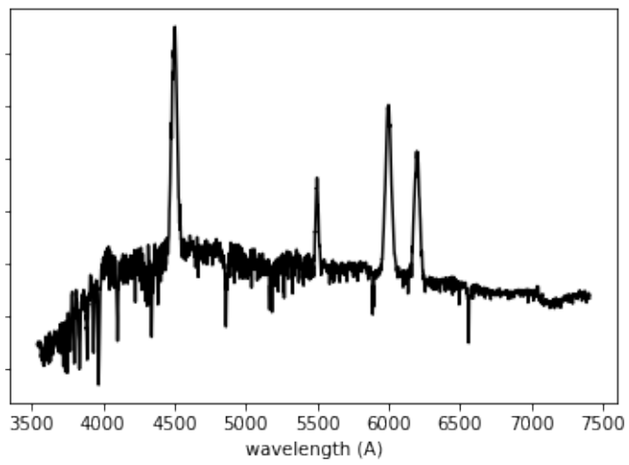
PC3



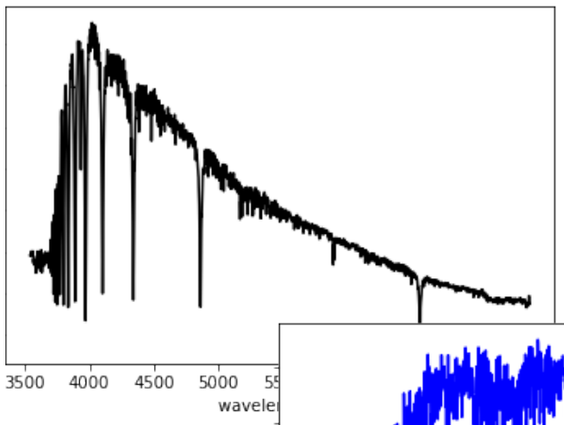


PCA - step by step

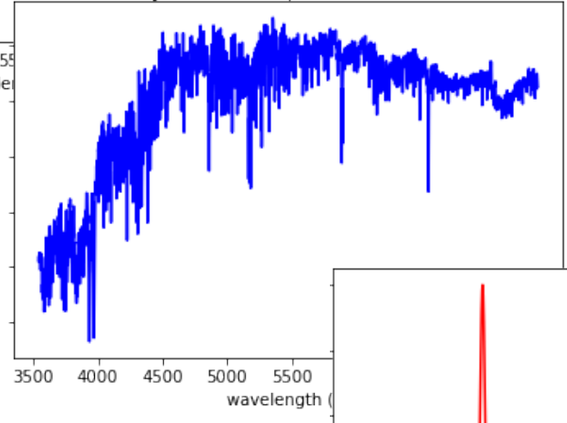
Step IV - transform the normalized input data to the new coordinate system: $V = ZP$. Each object is a linear combination of the principle components.



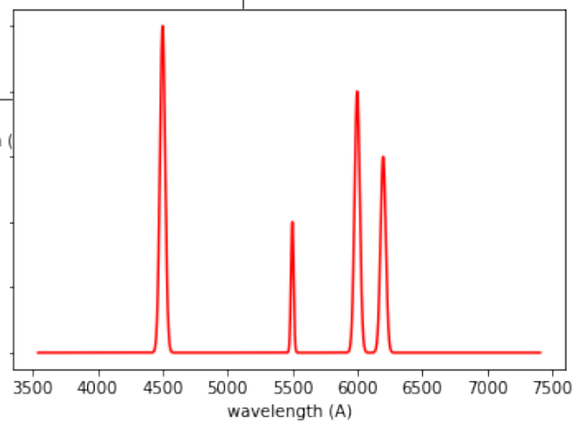
$= \{a_1 \times$



$, a_2 \times$



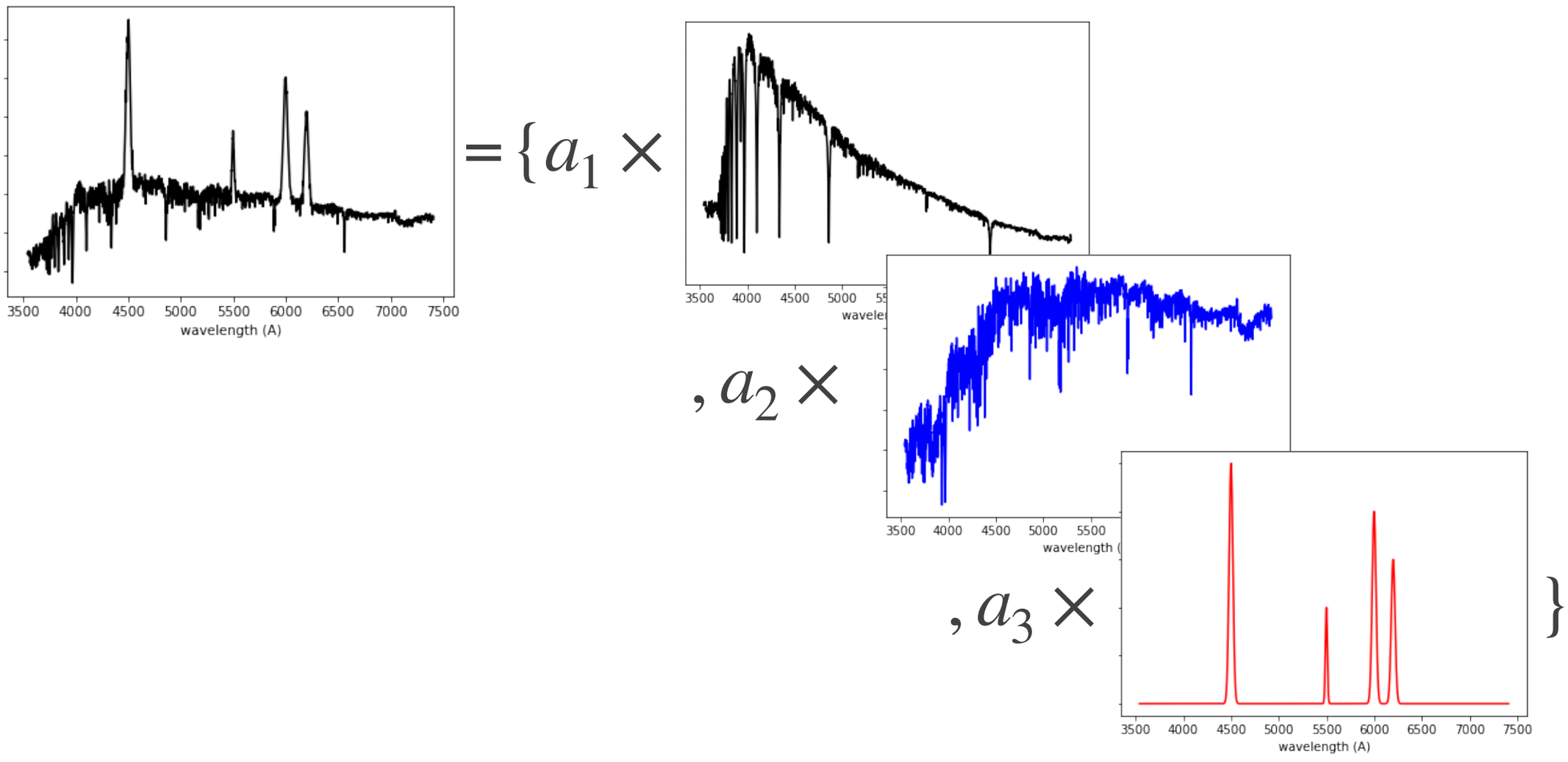
$, a_3 \times$



$\}$

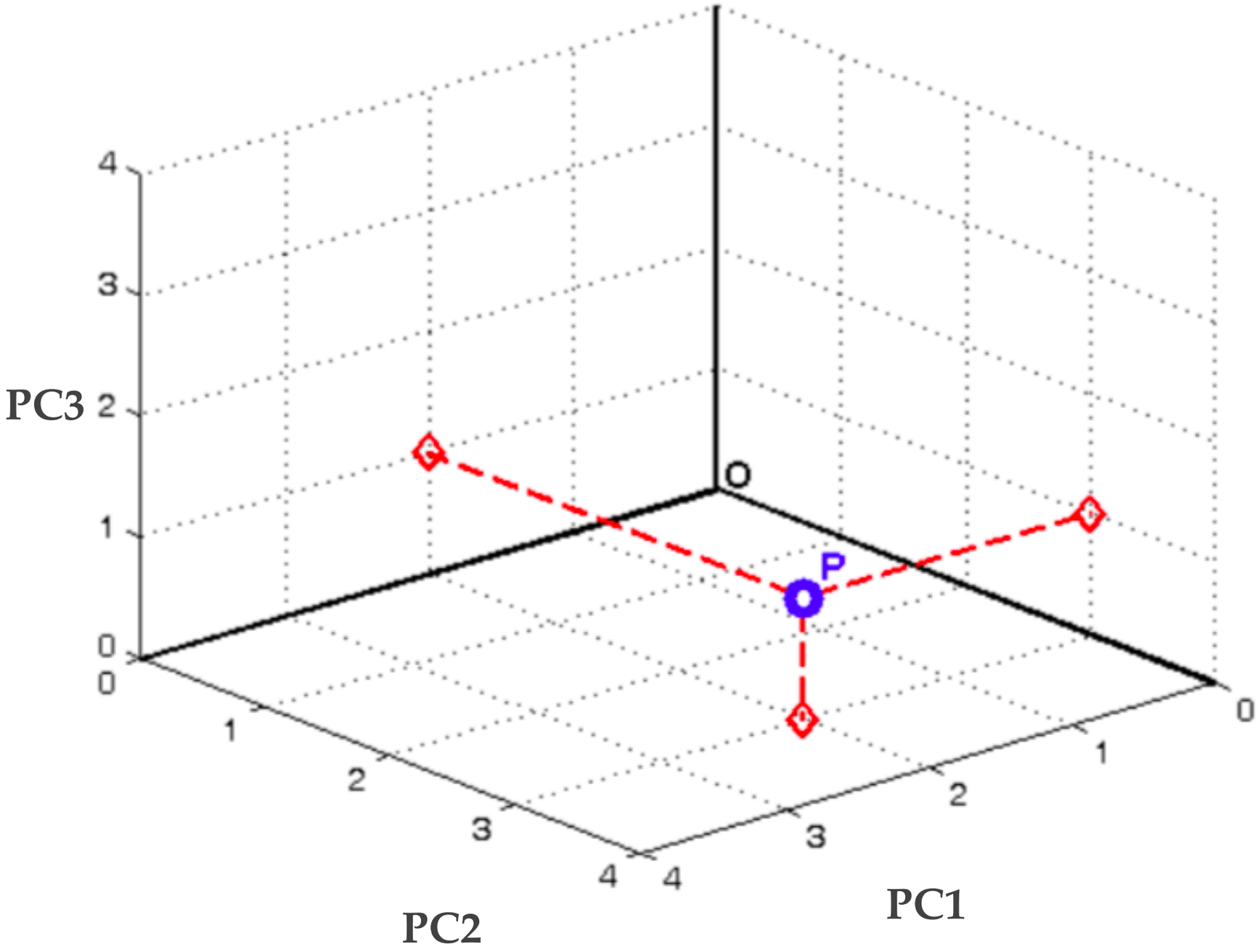
PCA - step by step

Step IV - transform the normalized input data to the new coordinate system: $V = ZP$. Each object is a linear combination of the principle components.



Now every spectrum is described using 3 numbers, rather than 4300.

PCA - step by step



Now every spectrum is described using 3 numbers, rather than 4300.

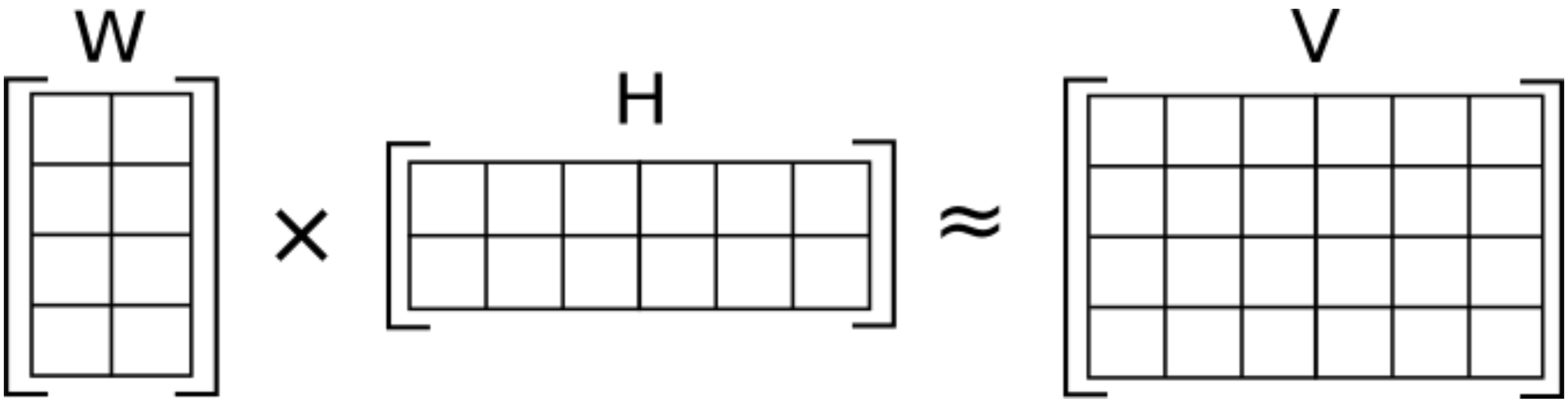
PCA - pros and cons

- **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.

- To the Jupyter notebook!

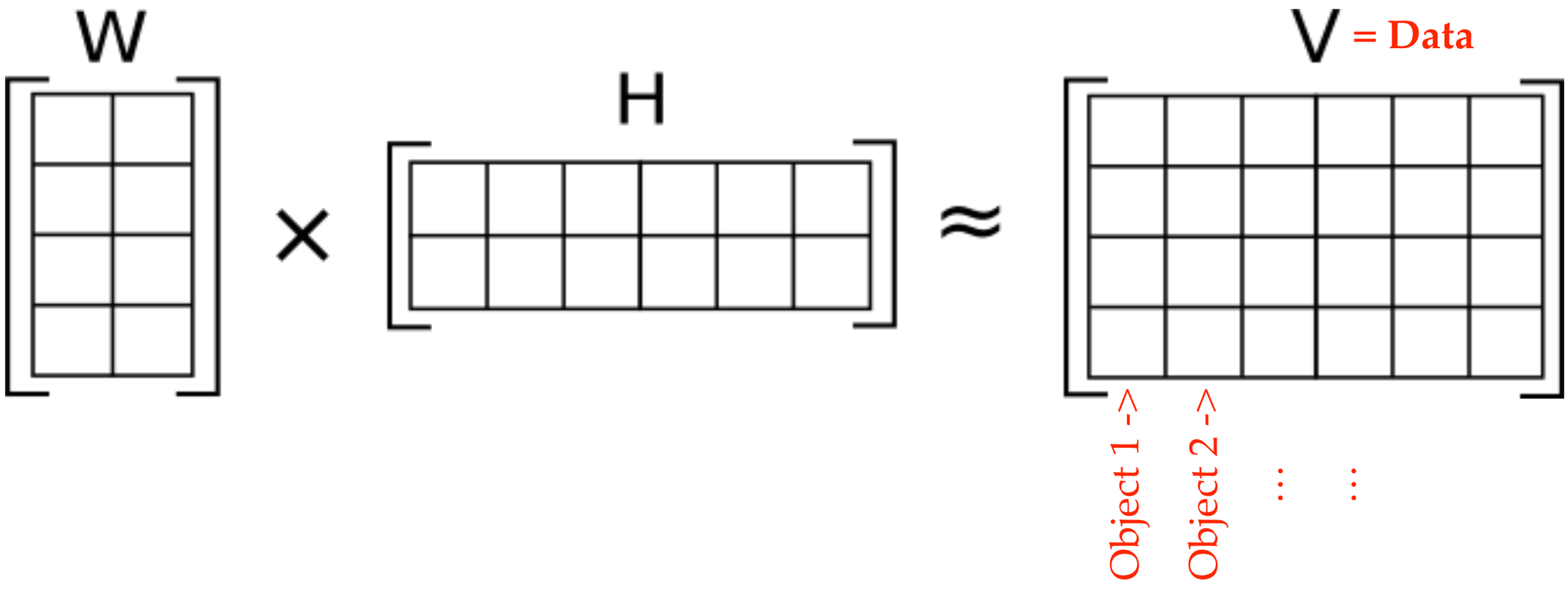
Non-Negative Matrix Factorization (NMF)

- In non-negative matrix factorization, a large matrix V into (usually much smaller) matrices W and H, where all three matrices have no negative elements.
  - By applying such a decomposition, we essentially assume that the (non-negative) signal is composed of a sum of positive contributions: combine parts to form a whole.



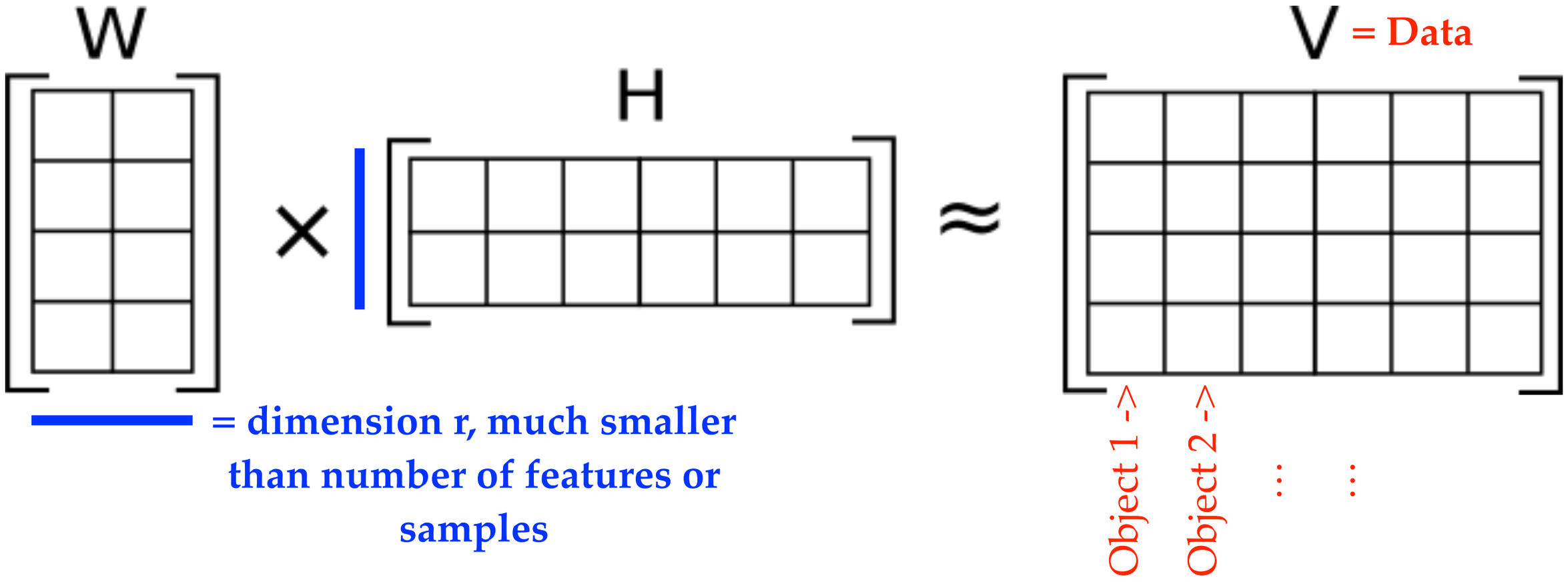
Non-Negative Matrix Factorization (NMF)

- In non-negative matrix factorization, a large matrix V into (usually much smaller) matrices W and H, where all three matrices have no negative elements.
  - By applying such a decomposition, we essentially assume that the (non-negative) signal is composed of a sum of positive contributions: combine parts to form a whole.



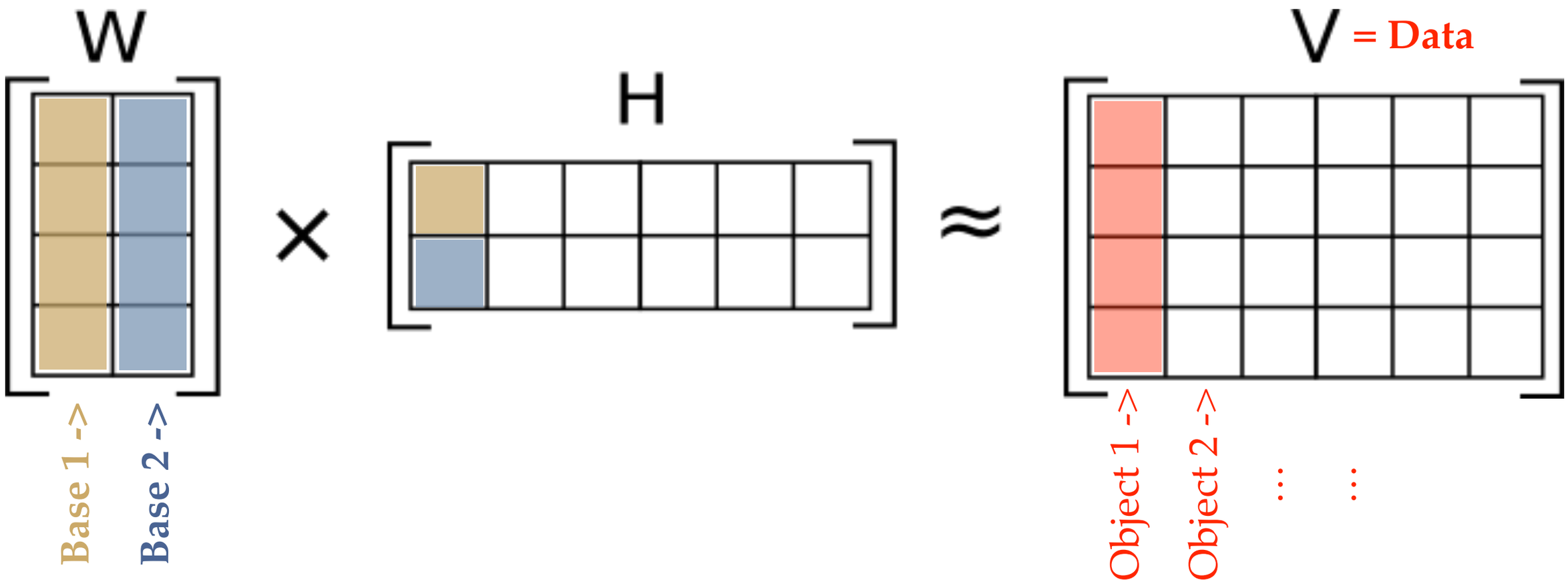
Non-Negative Matrix Factorization (NMF)

- In non-negative matrix factorization, a large matrix V into (usually much smaller) matrices W and H, where all three matrices have no negative elements.
  - By applying such a decomposition, we essentially assume that the (non-negative) signal is composed of a sum of positive contributions: combine parts to form a whole.



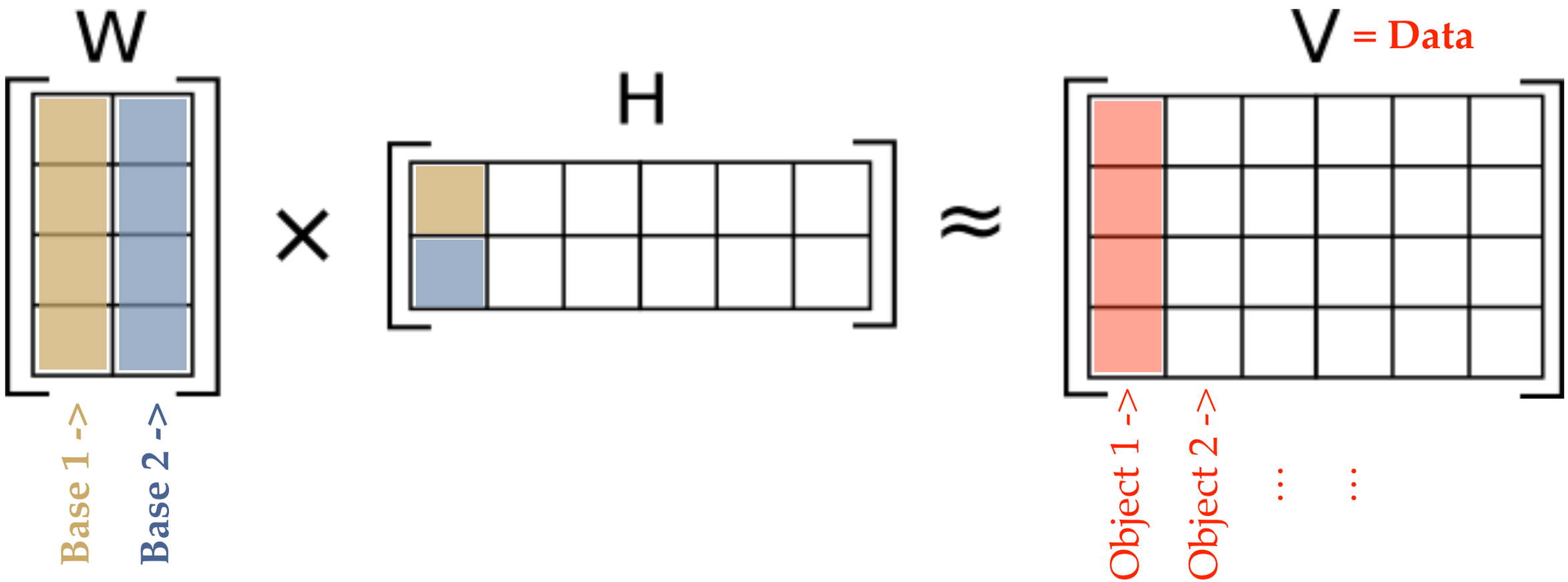
Non-Negative Matrix Factorization (NMF)

- In non-negative matrix factorization, a large matrix V into (usually much smaller) matrices W and H, where all three matrices have no negative elements.
  - By applying such a decomposition, we essentially assume that the (non-negative) signal is composed of a sum of positive contributions: combine parts to form a whole.



Non-Negative Matrix Factorization (NMF)

- In non-negative matrix factorization, a large matrix V into (usually much smaller) matrices W and H, where all three matrices have no negative elements.
  - By applying such a decomposition, we essentially assume that the (non-negative) signal is composed of a sum of positive contributions: combine parts to form a whole.



NMF: notes

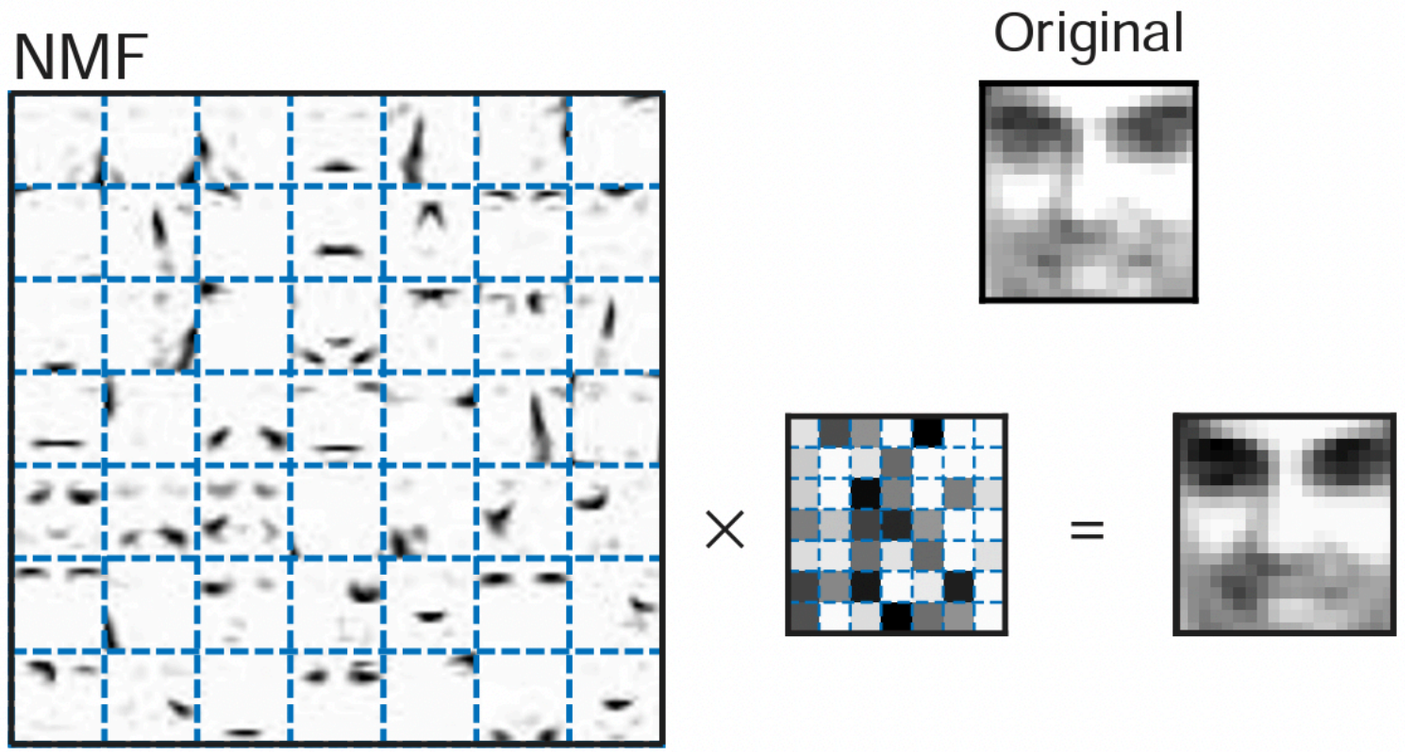
- There is no analytical solution for $W \geq 0, H \geq 0$ that satisfy $W \times H \approx V$, but there exist several numerical methods to do that.
- Assuming underlying structure to the data, NMF constructs sparse bases and sparse weightings.

Learning the parts of objects by nonnegative matrix factorization

1999

D. D. Lee*
*Bell Laboratories
Lucent Technologies
Murray Hill, NJ 07974

H. S. Seung*,†
†Dept. of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139



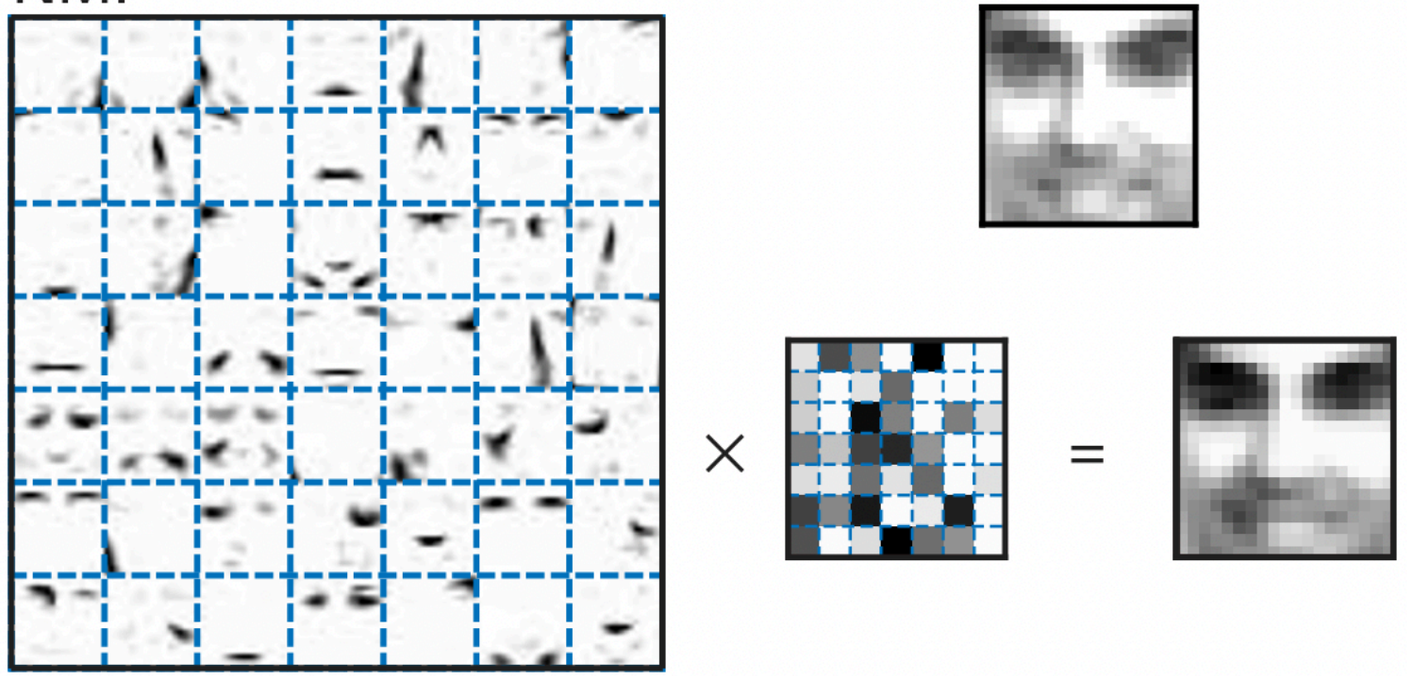
Learning the parts of objects by nonnegative matrix factorization

1999

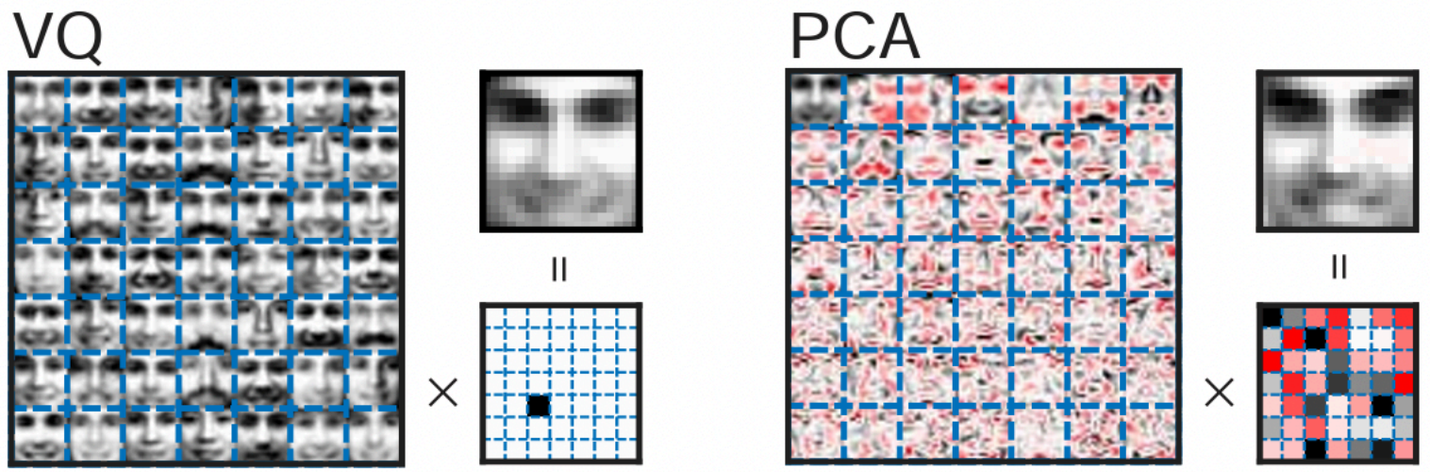
D. D. Lee*
*Bell Laboratories
Lucent Technologies
Murray Hill, NJ 07974

H. S. Seung*,†
†Dept. of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139

Parts of faces ->



For NMF, most of the bases contain numerous empty spaces, suggesting that this is a sparse representation

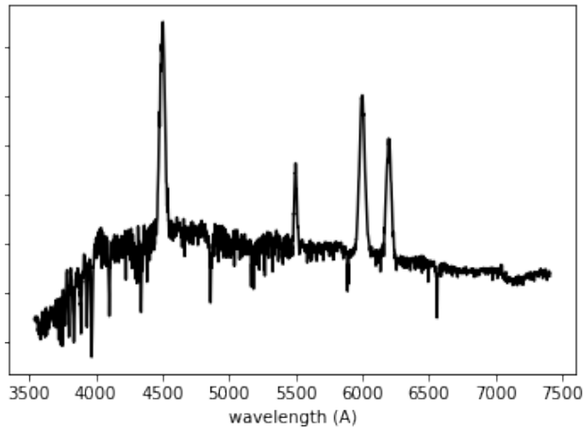


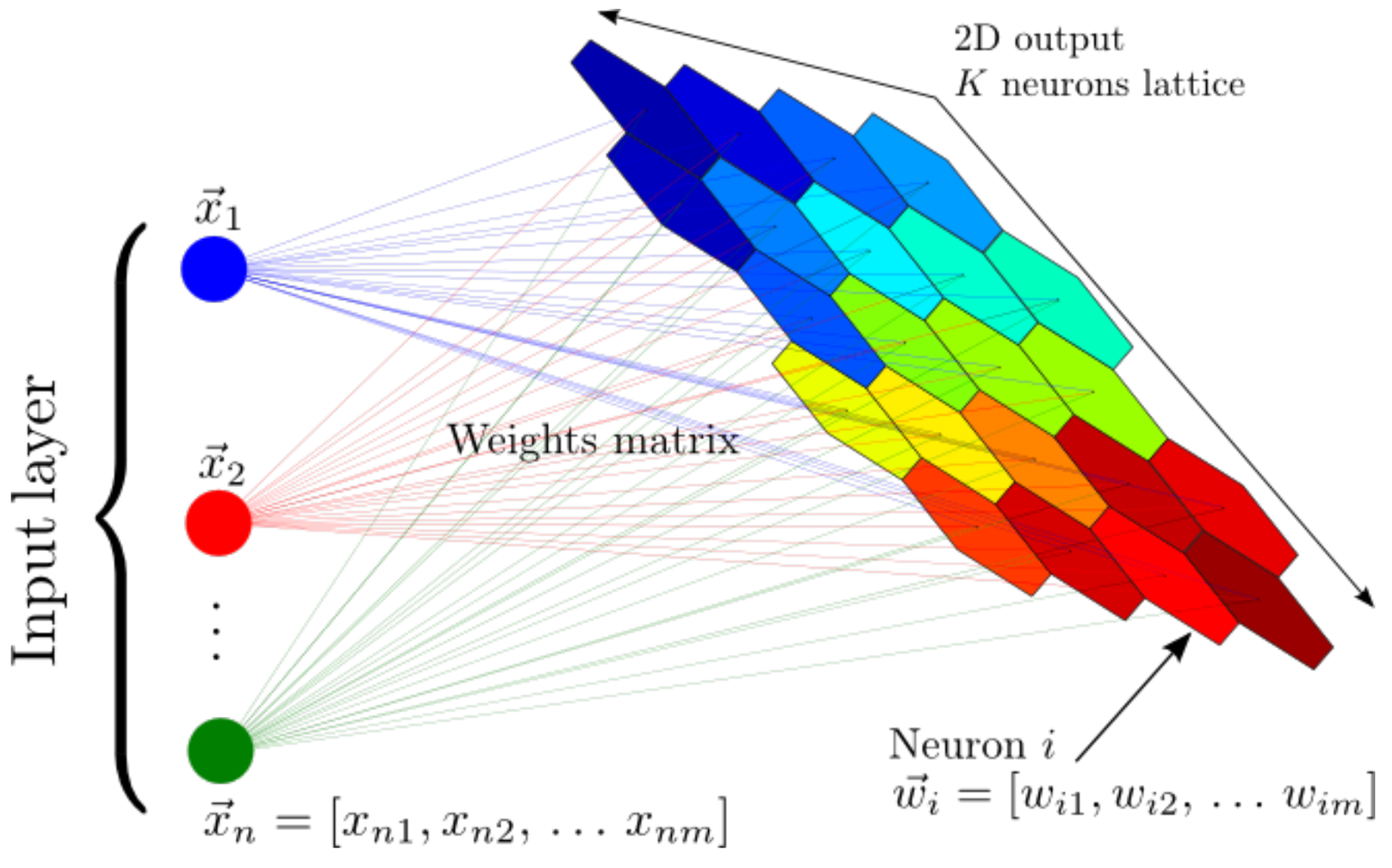
PCA has dense matrices containing positive and negative values.

- To the Jupyter notebook!

Self-Organizing Maps (SOM)

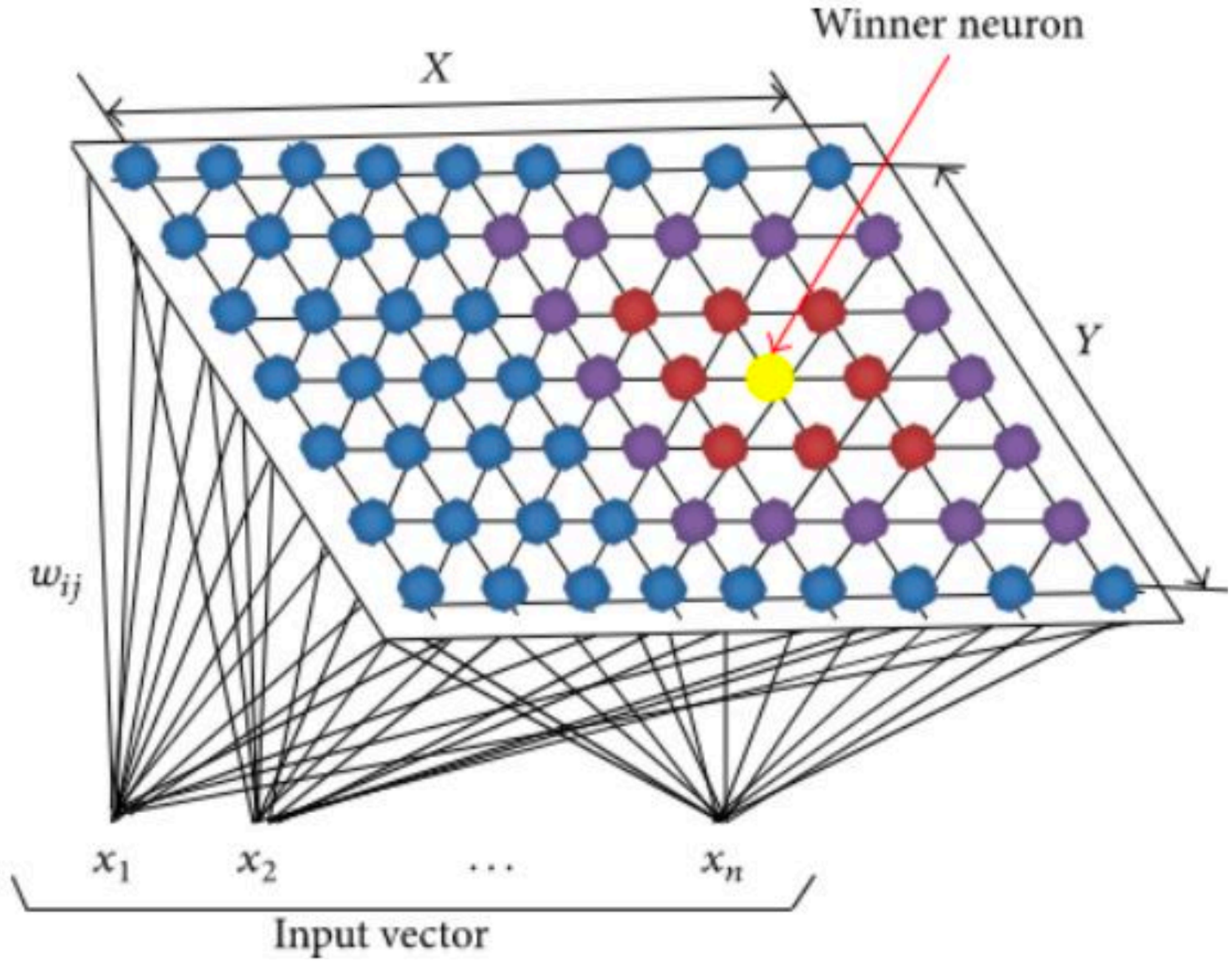
- SOM is an algorithm used to produce a low-dimensional representation of a higher-dimensional data. It belongs to the class of competitive learning algorithms.
- It is a neural network composed of two layers:
  - The input layer, with a dimension *d*, as our input data.
  - The output layer is a 2-dimensional lattice, composed of *k* neurons.
  - The input layer is fully-connected to the output layer using weight vectors of dimension *d*.





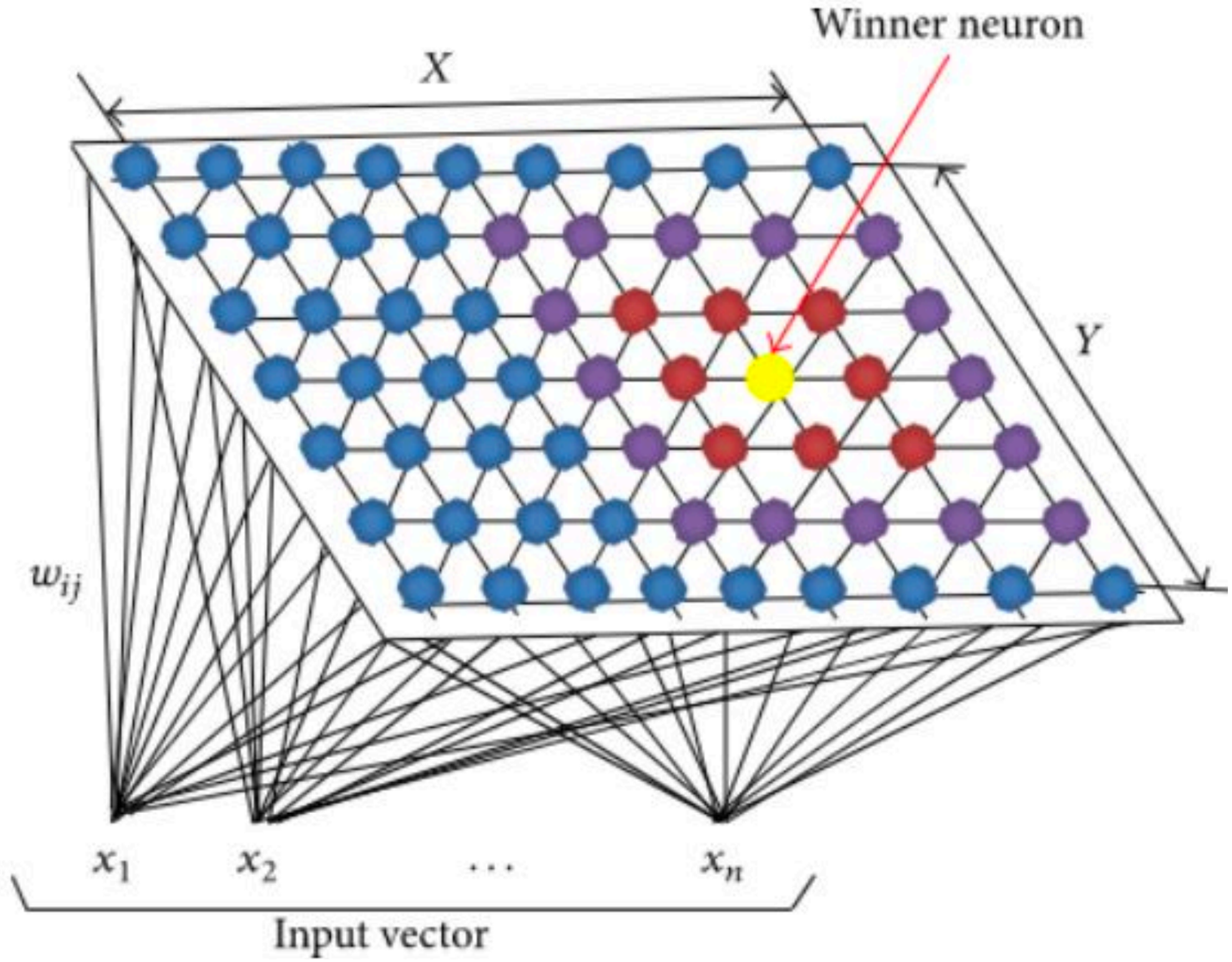
SOM: network training steps

1. The weight vectors, which have the same dimension d as our input objects, are assigned with random values. As the training progresses, the weights will become closer and closer to our input objects, and thus they are considered as the *prototypes* of our data.



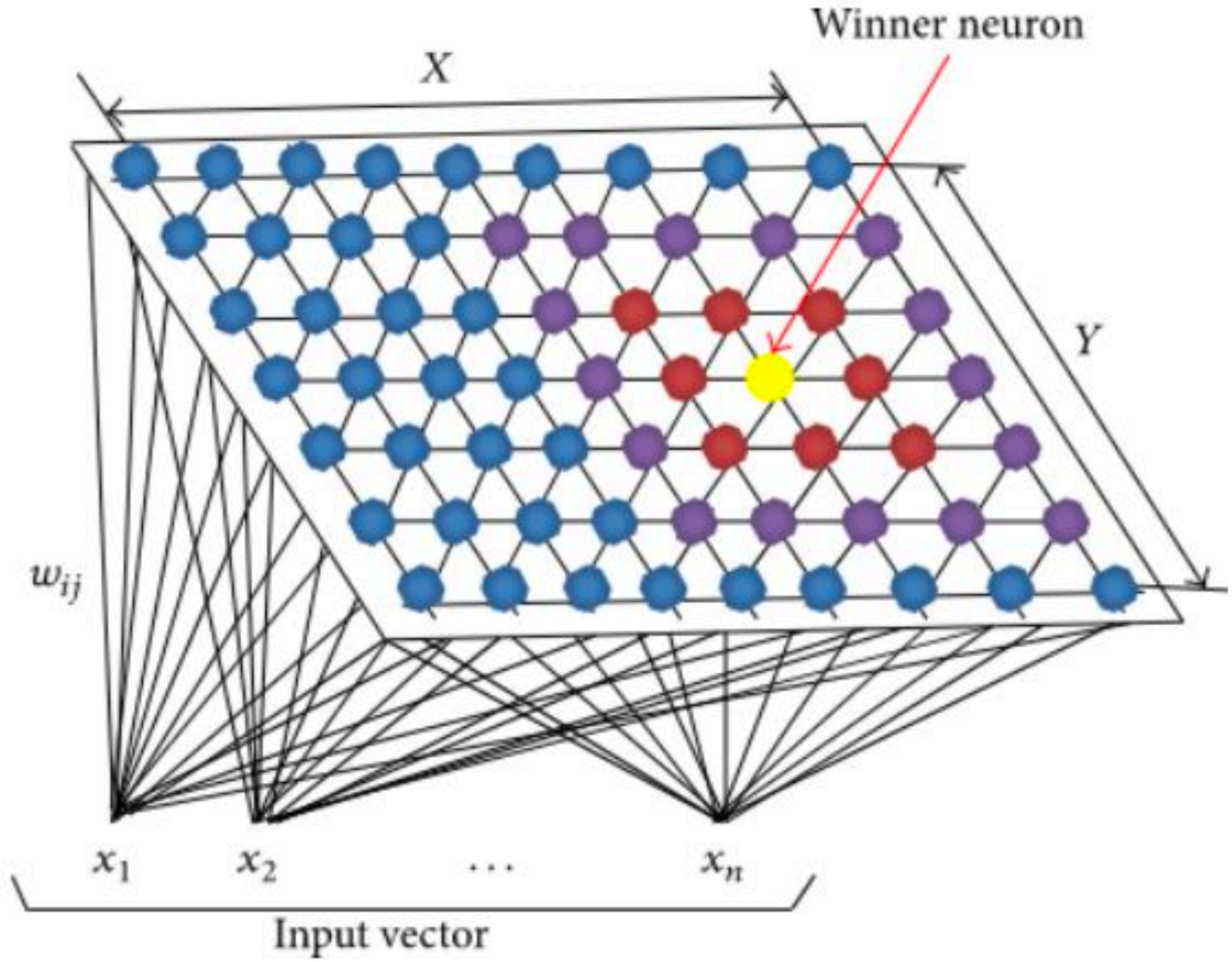
SOM: network training steps

2. For every new input vector, the Euclidean distance between the input vector and all the weight vectors is calculated: $D = ||W_i - X||$, where i is the number of neurons in the 2-d map.



SOM: network training steps

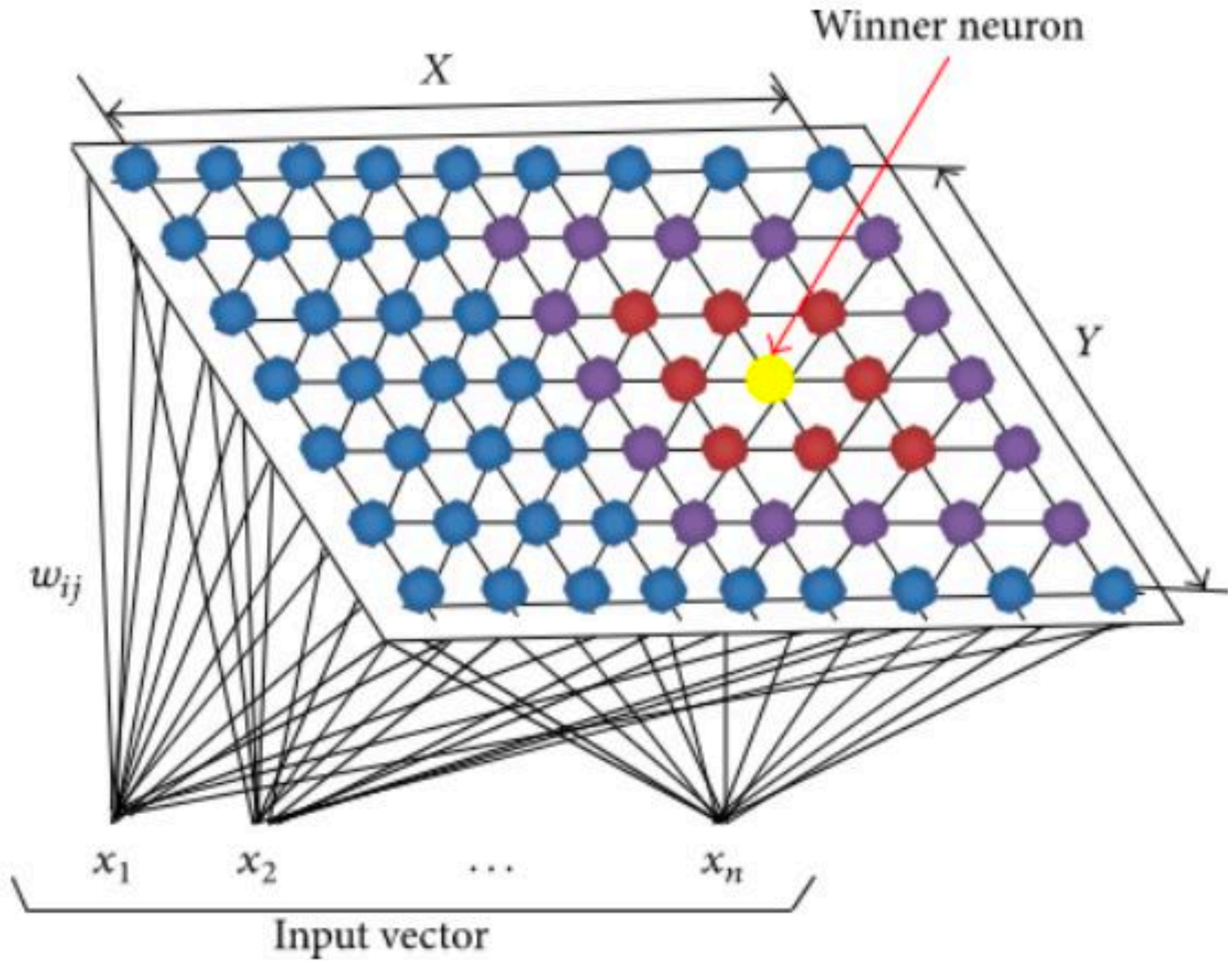
3. The p-th neuron that has the smallest Euclidean distance is declared as the winner. Its weight vector is then updated according to: $W_{p,t+1} \Leftarrow W_{p,t} + \alpha_t(X - W_{p,t})$



SOM: network training steps

3. The p-th neuron that has the smallest Euclidean distance is declared as the winner. Its weight vector is then updated according to: $W_{p,t+1} \Leftarrow W_{p,t} + \alpha_t(X - W_{p,t})$

Learning rate, decreases with time



SOM: network training steps

3. The p-th neuron that has the smallest Euclidean distance is declared as the winner. Its weight vector is then updated according to: $W_{p,t+1} \Leftarrow W_{p,t} + \alpha_t(X - W_{p,t})$

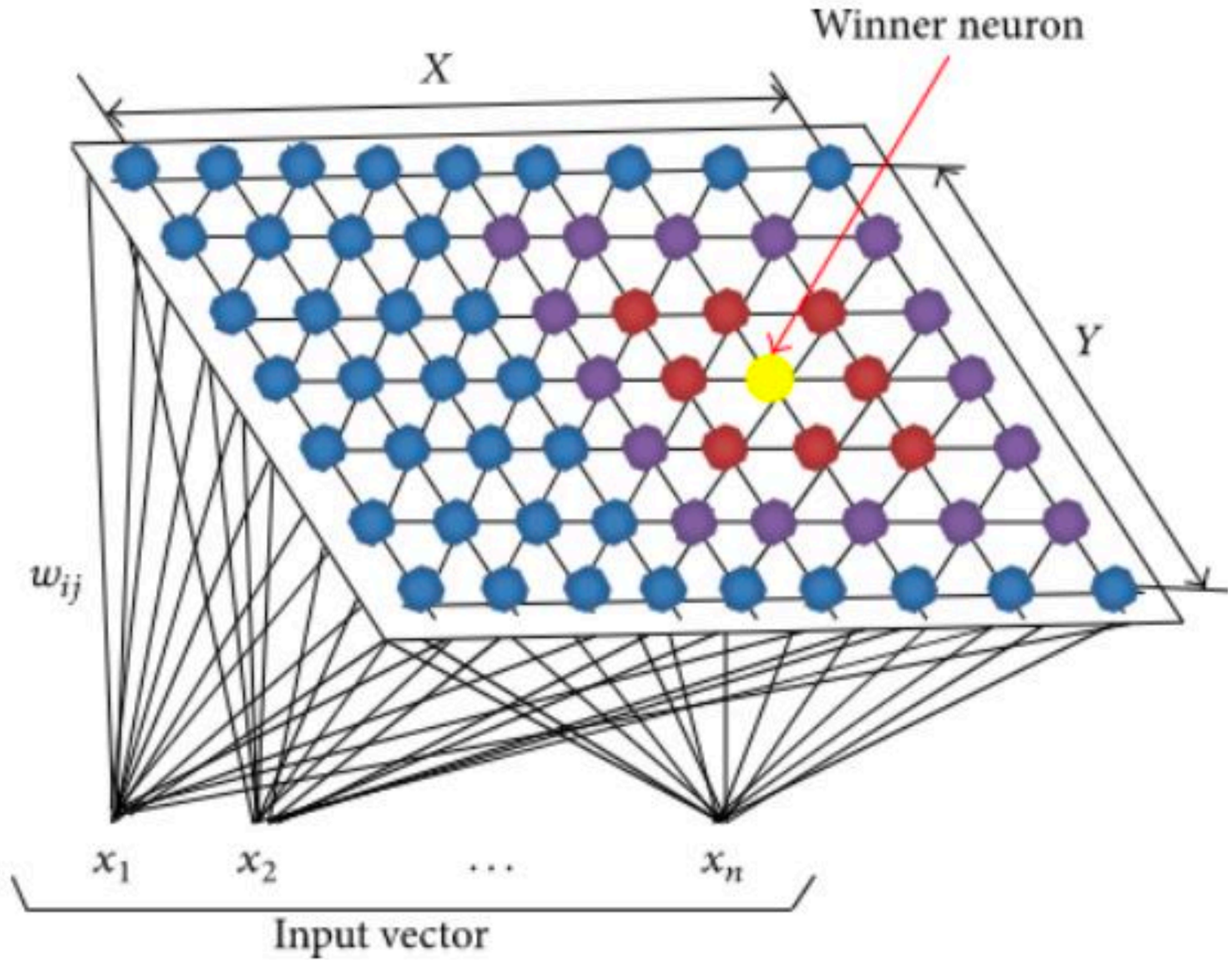
Learning rate, decreases with time

4. The neighboring neurons on the lattice will also get updated, but with a damping factor:

$$\delta_{ij} = \exp - \left(\frac{LD(i,j)^2}{2\sigma^2} \right)$$

The weights will then be updated with:

$$W_{i,t+1} \Leftarrow W_{i,t} + \alpha_t \delta_{ij}(X - W_{i,t})$$



SOM: network training steps

3. The p-th neuron that has the smallest Euclidean distance is declared as the winner. Its weight vector is then updated according to: $W_{p,t+1} \Leftarrow W_{p,t} + \alpha_t(X - W_{p,t})$

Learning rate, decreases with time

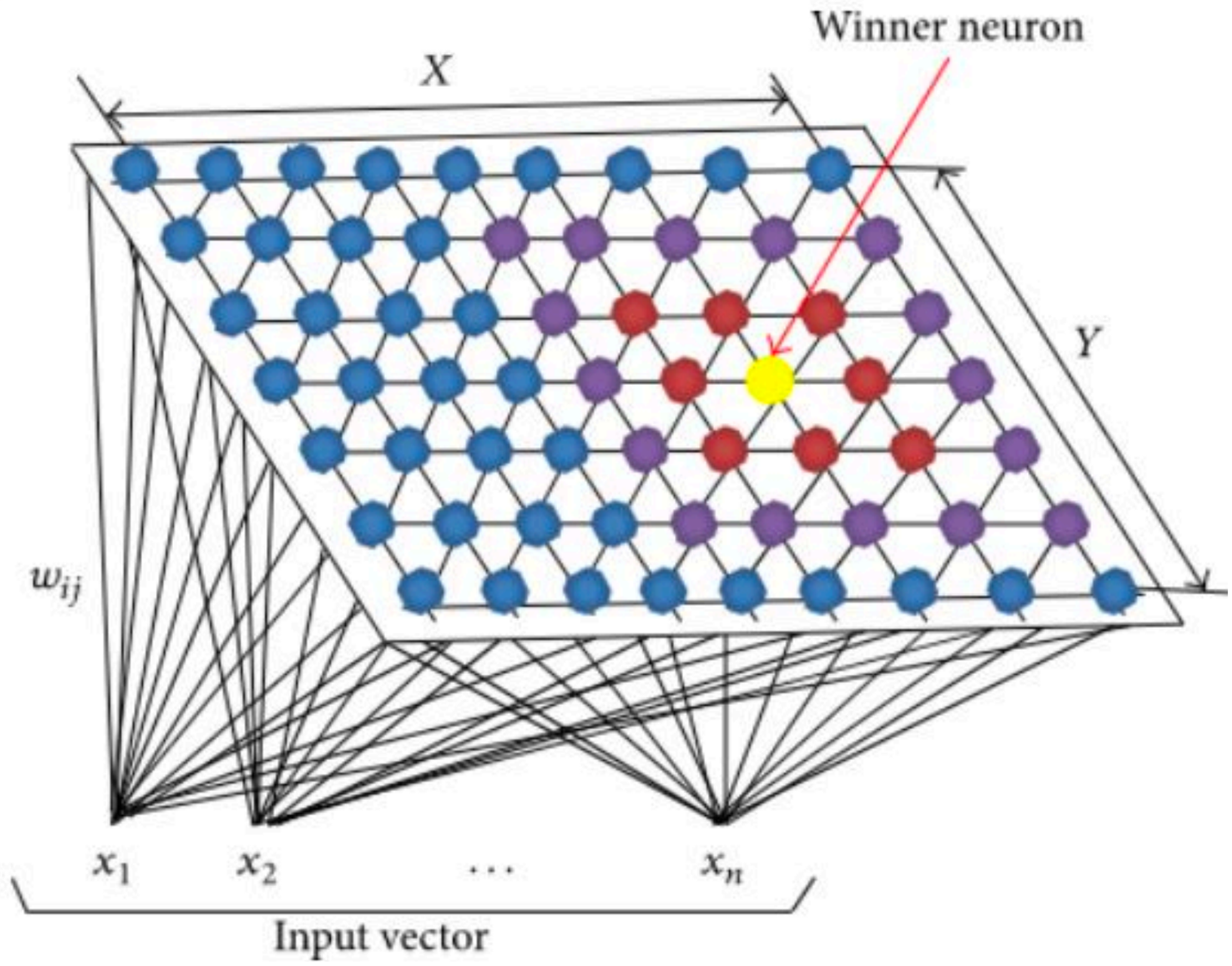
4. The neighboring neurons on the lattice will also get updated, but with a damping factor:

$$\delta_{ij} = \exp - \left(\frac{LD(i,j)^2}{2\sigma^2} \right)$$

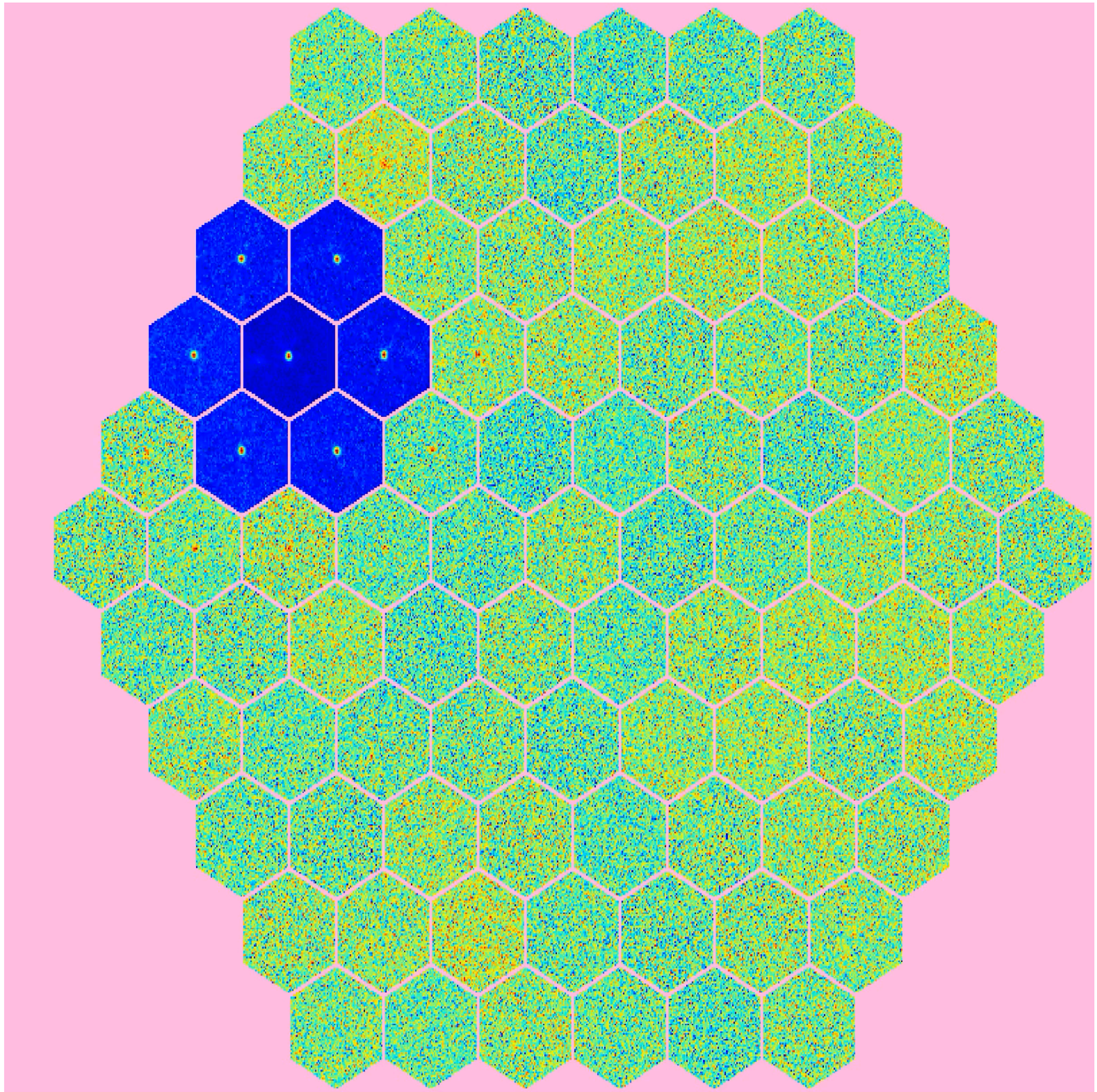
The weights will then be updated with:

$$W_{i,t+1} \Leftarrow W_{i,t} + \alpha_t \delta_{ij}(X - W_{i,t})$$

Closer neurons will be updated more!



SOM: visualization of training steps

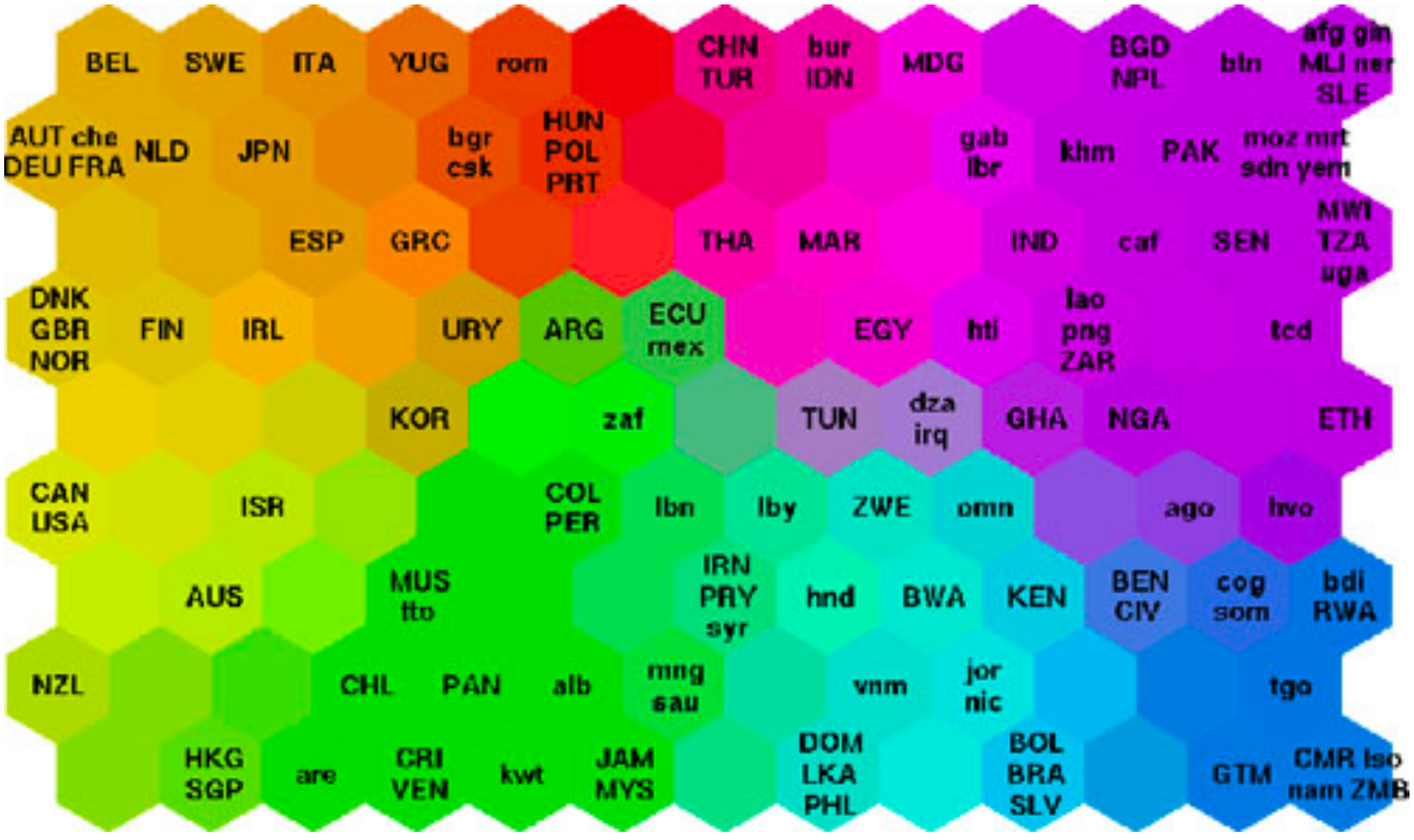


Many thanks to Erica Hopkins and Kai Polsterer for sharing with me the video!

SOM: inspecting the low-dimensional representation

Before training, we have to specify the number of neurons and the lattice structure (hexagonal or rectangular).

After training, the neurons in the 2-d lattice are essentially prototypes that represent clusters of objects which are similar to each other.



SOM: inspecting the low-dimensional representation

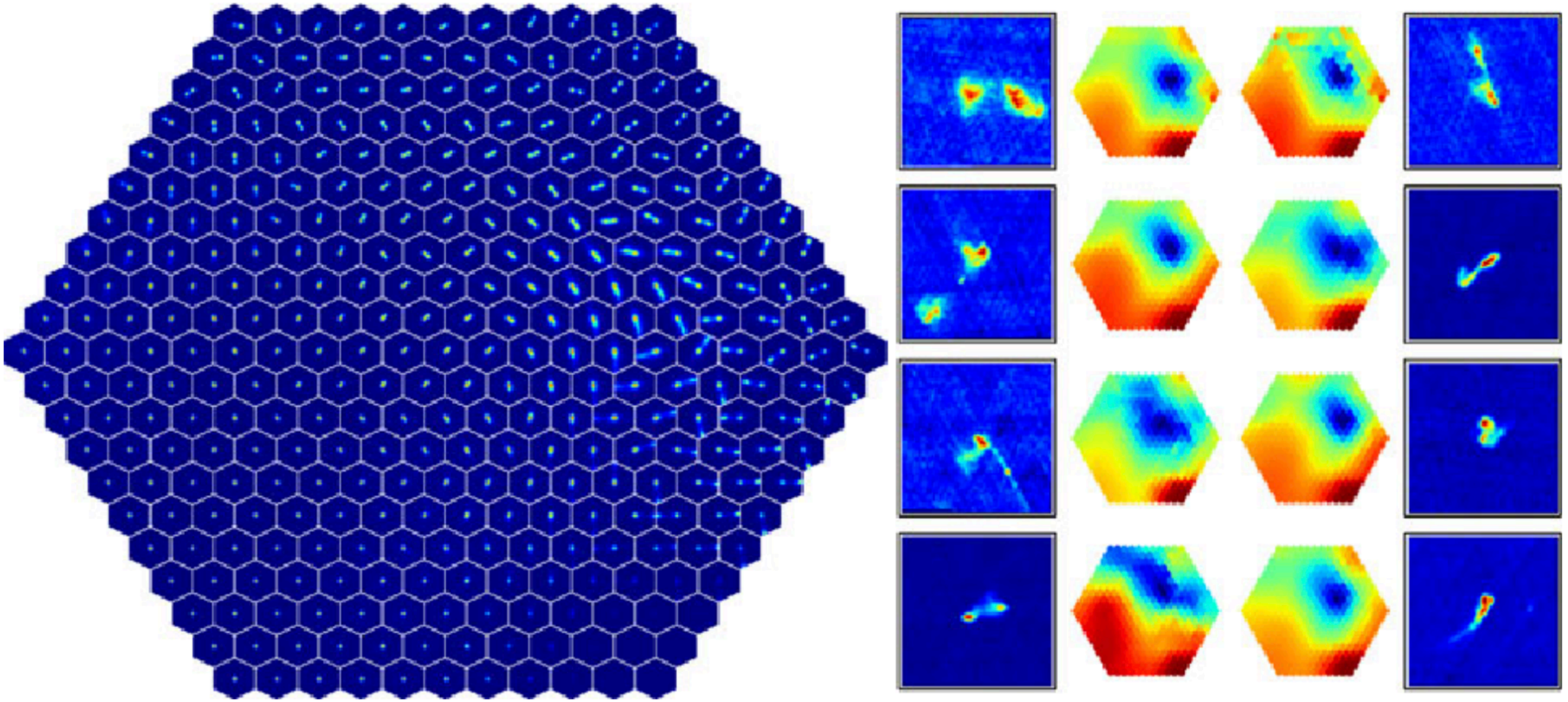


Figure 18. Application of PINK to 200 000 radio images from Radio Galaxy Zoo, taken from Polsterer et al. (2016). The left panel shows the resulting two-dimensional map containing the derived prototypes. The right panel shows eight outliers that were selected based on their dissimilarity with the prototypes, and heatmaps that indicate their distance to all the prototypes.