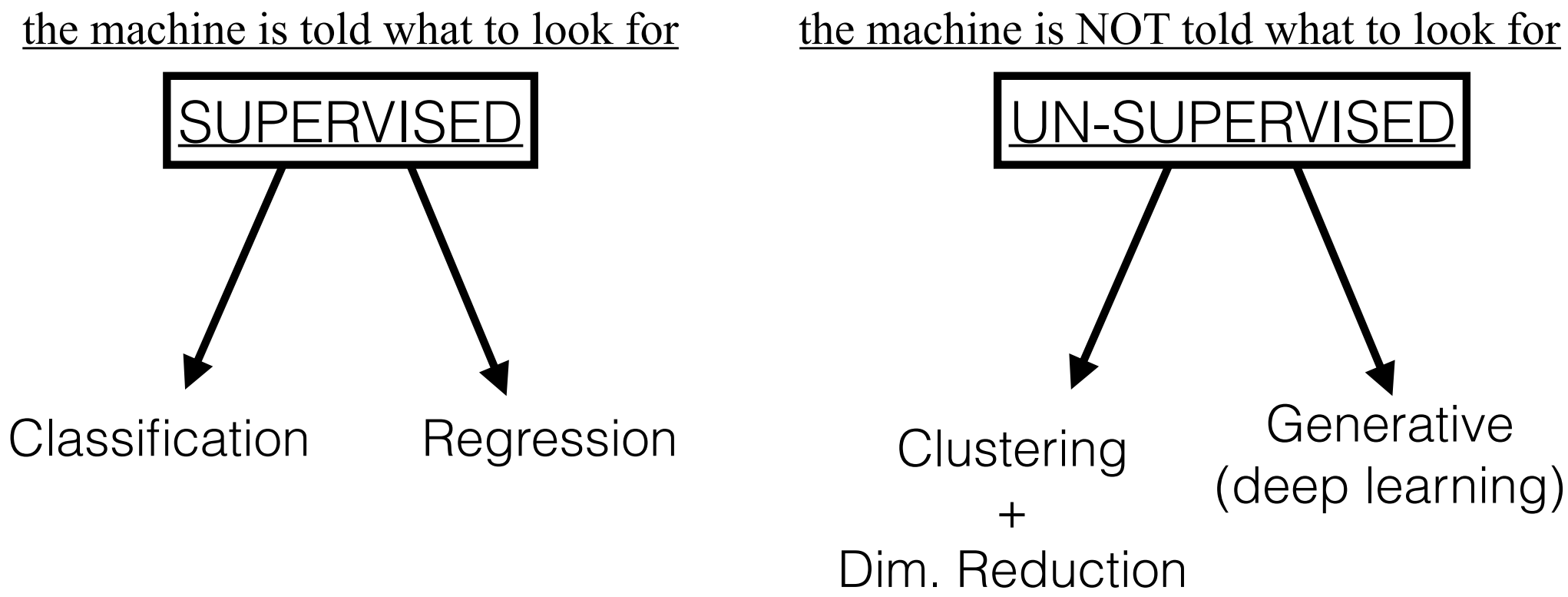
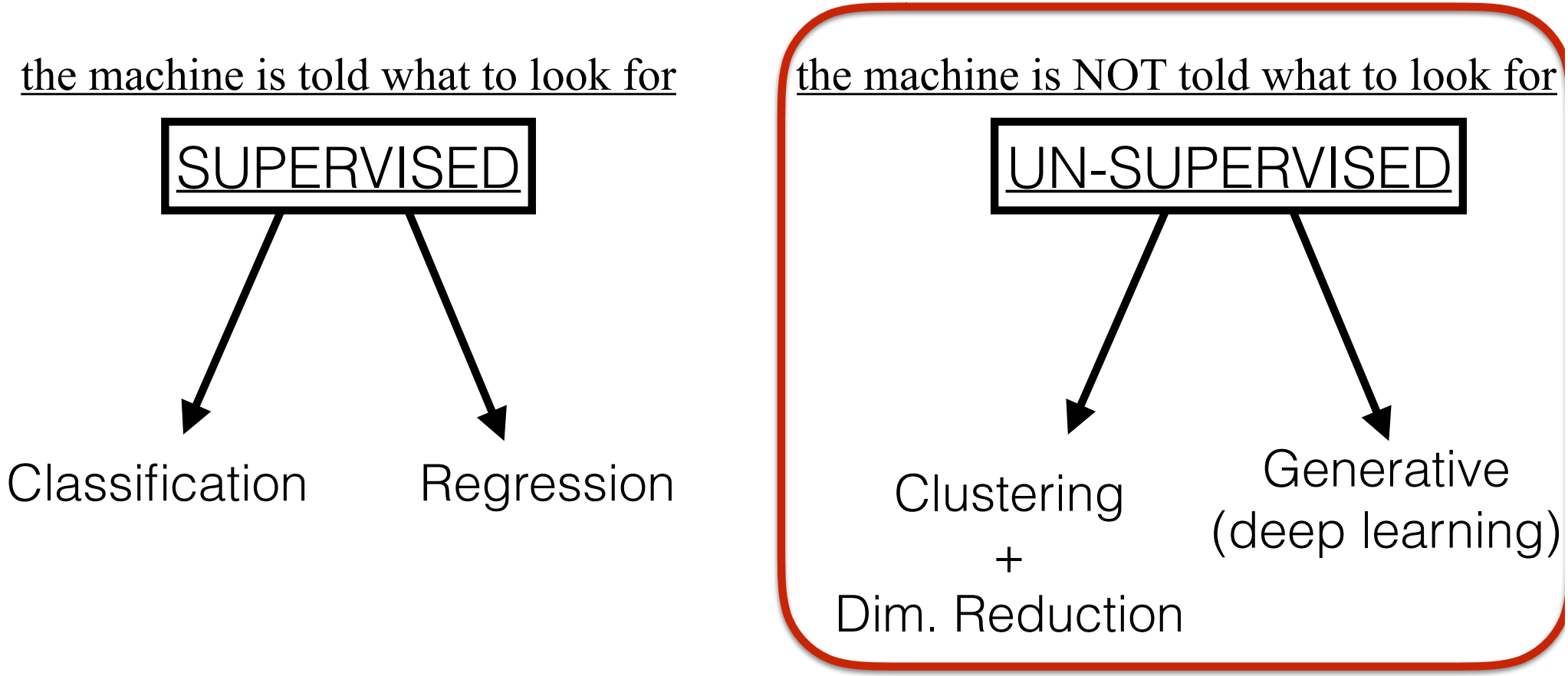
# PART IV: A VERY BRIEF INTRODUCTION TO UNSUPERVISED LEARNING

# WHAT DOES MACHINE LEARNING DO?

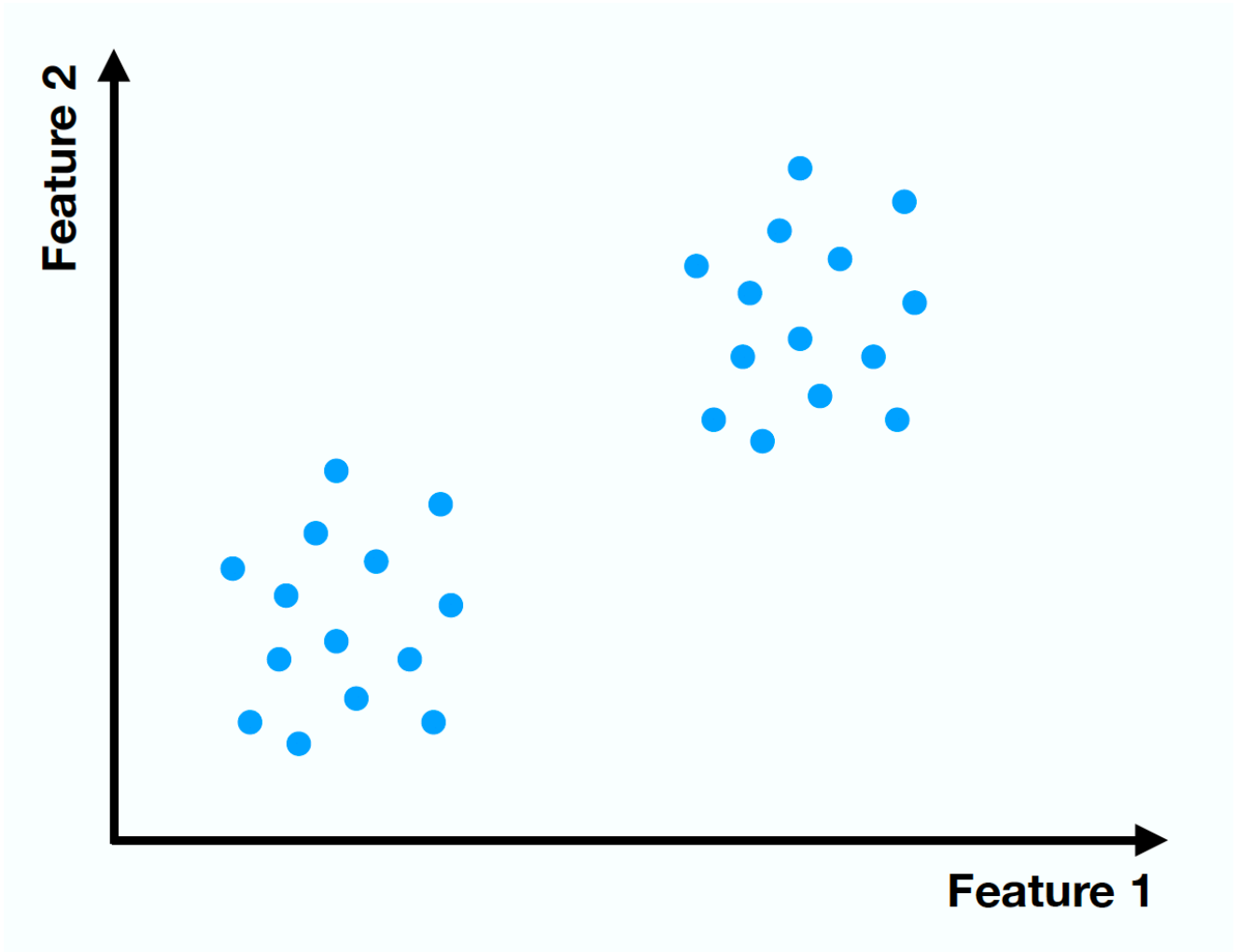


# WHAT DOES MACHINE LEARNING DO?



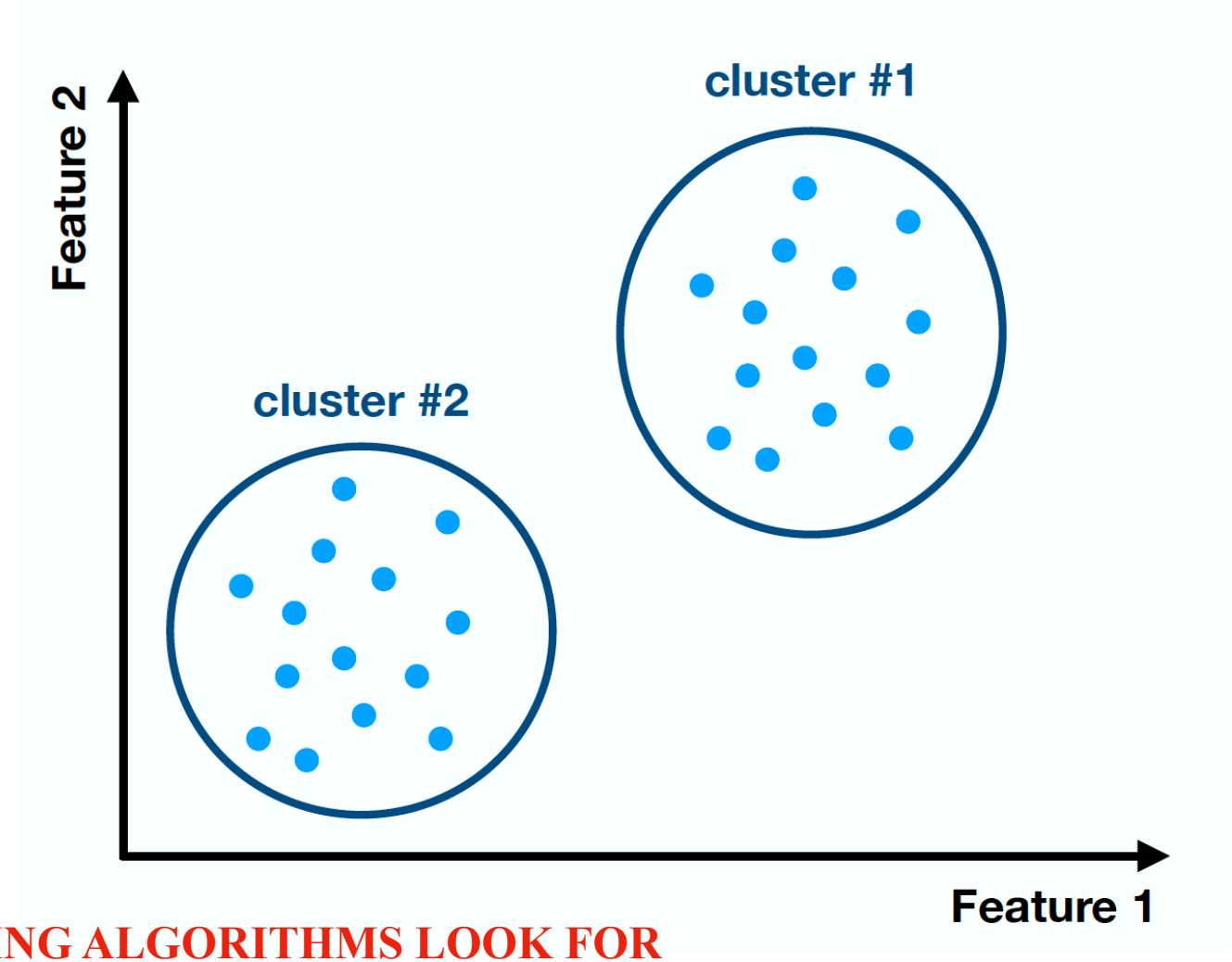
# CLUSTERING ALGORITHMS

# CLUSTERING ALGORITHMS




**Credit: D. Baron (IAC XXX Winter School)**

# CLUSTERING ALGORITHMS




**CLUSTERING ALGORITHMS LOOK FOR GROUPS OF DATA POINTS IN THE DATA SPACE**

**Credit: D. Baron (IAC XXX Winter School)**

# CLUSTERING ALGORITHMS

## WHAT ARE CLUSTERING ALGORITHMS USEFUL FOR?

**1. CLASSIFICATION [OBJECTS WITH SIMILAR FEATURES ARE CLUSTERED TOGETHER]**

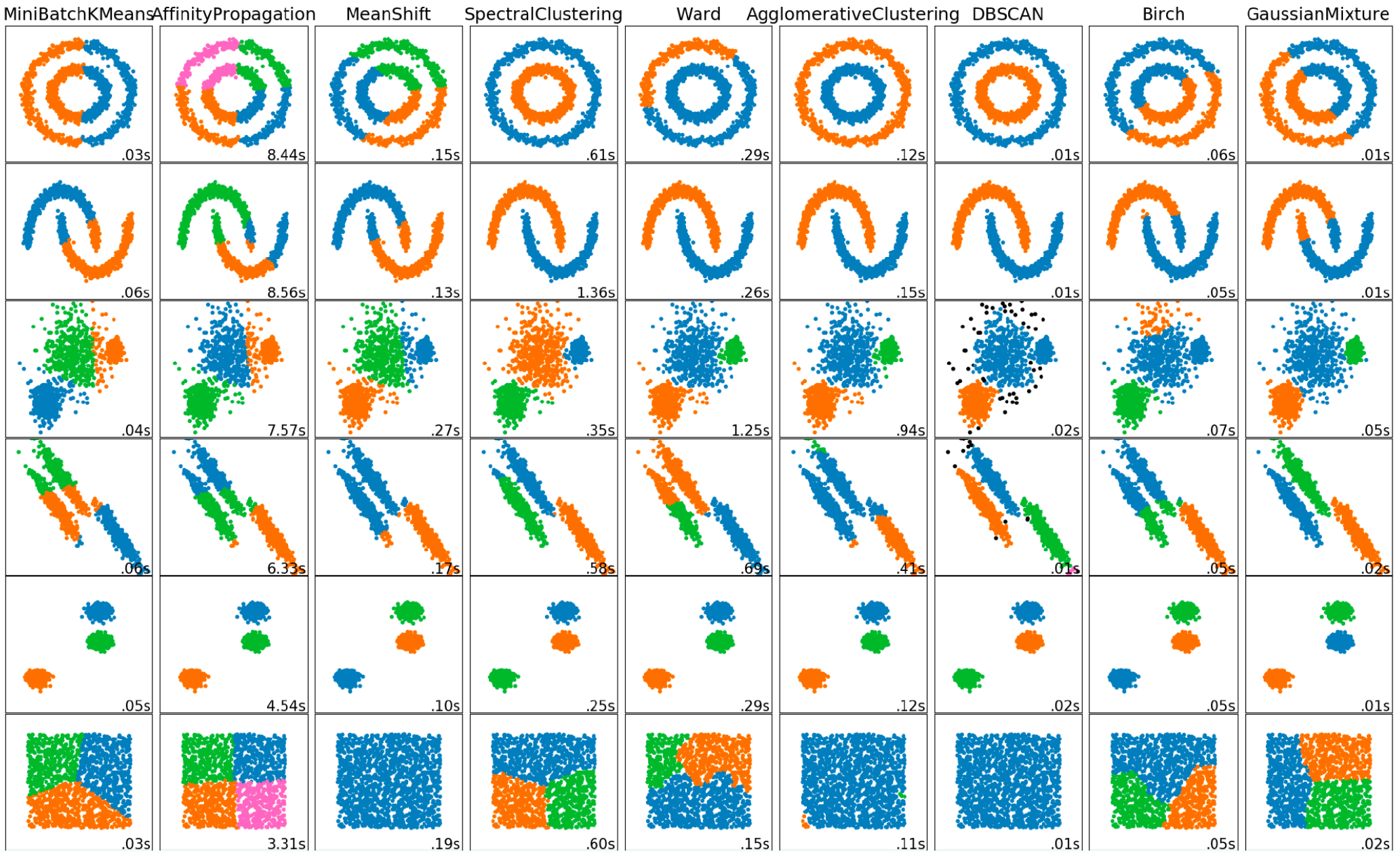
**2. VISUALIZATION OF LARGE DIMENSIONALITY SPACES**

**CLUSTERING ALGORITHMS LOOK FOR GROUPS OF DATA POINTS IN THE DATA SPACE**

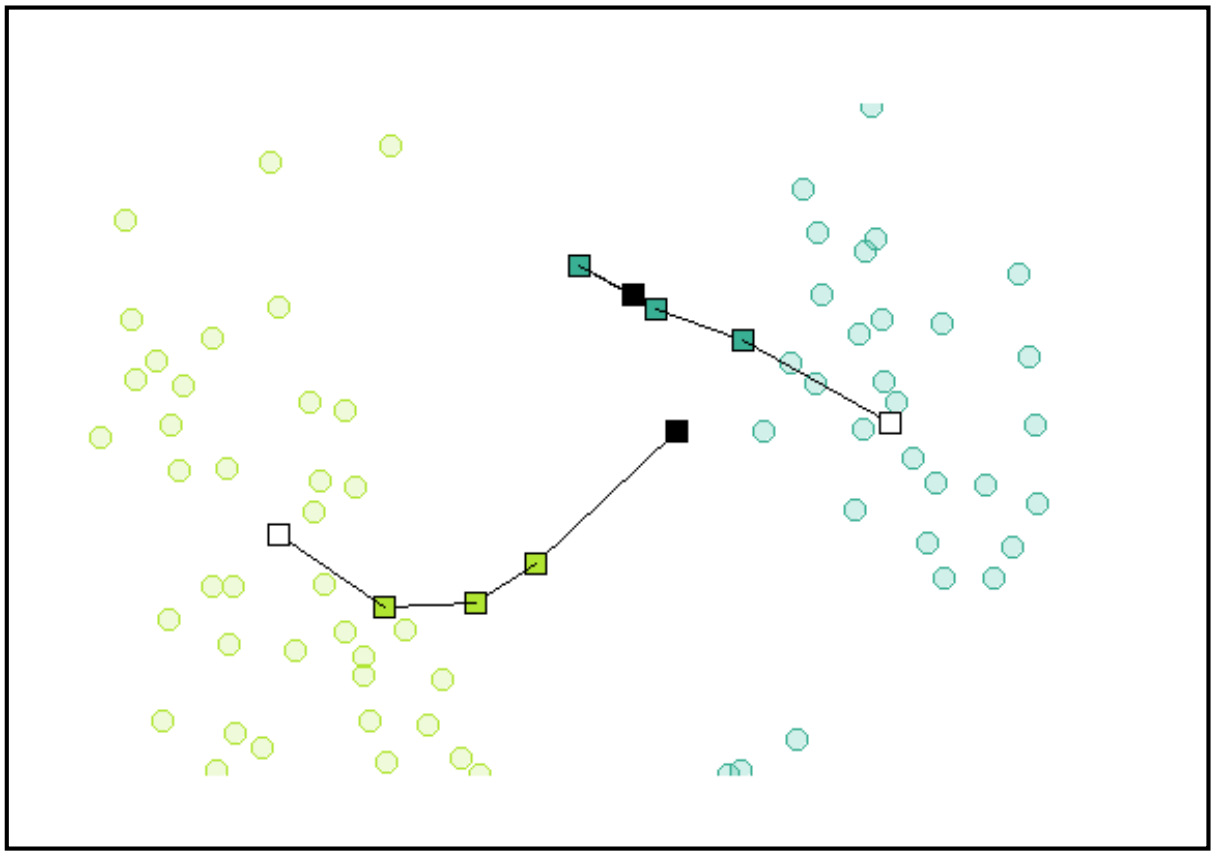
**Credit: D. Baron (IAC XXX Winter School)**

# THERE ARE A LARGE NUMBER OF CLUSTERING ALGORITHMS. THEI BEHAVIOR DIFFERS DEPENDING ON THE DATA STRUCTURE.

## [SCKIT LEARN]



# EXAMPLE: K-MEANS



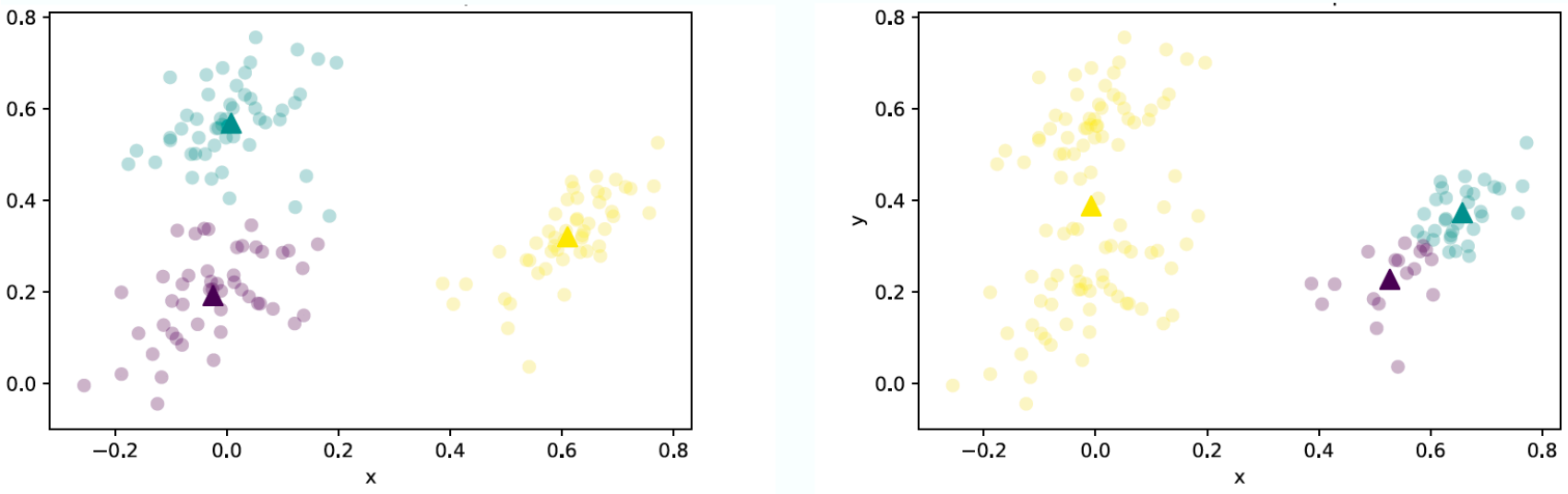
# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**Initial conditions:**

1. **Distance metric** ⟶ $L = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$
2. **Initial centroids**

k=3, and two different random placements of centroids

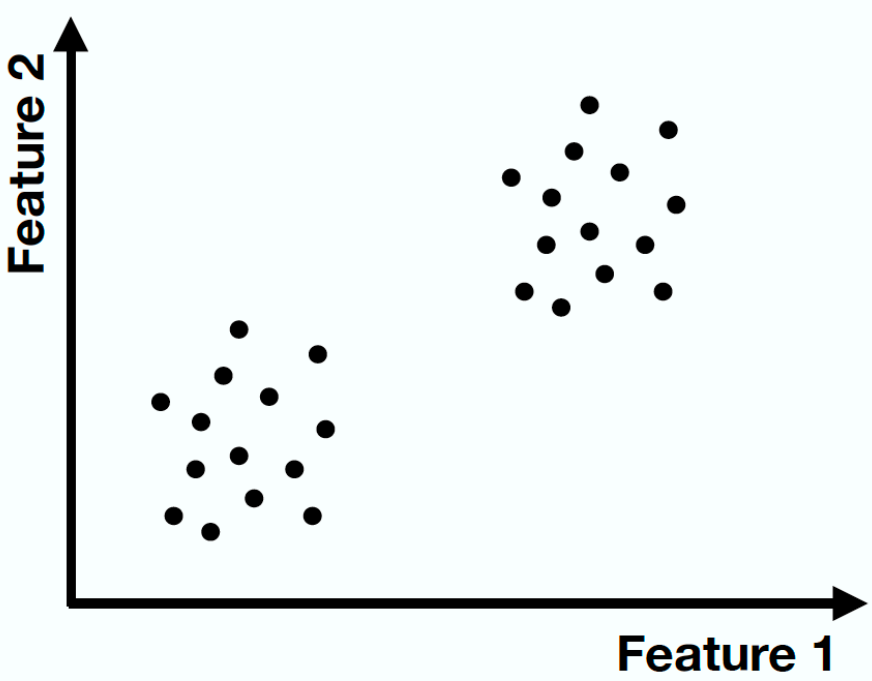


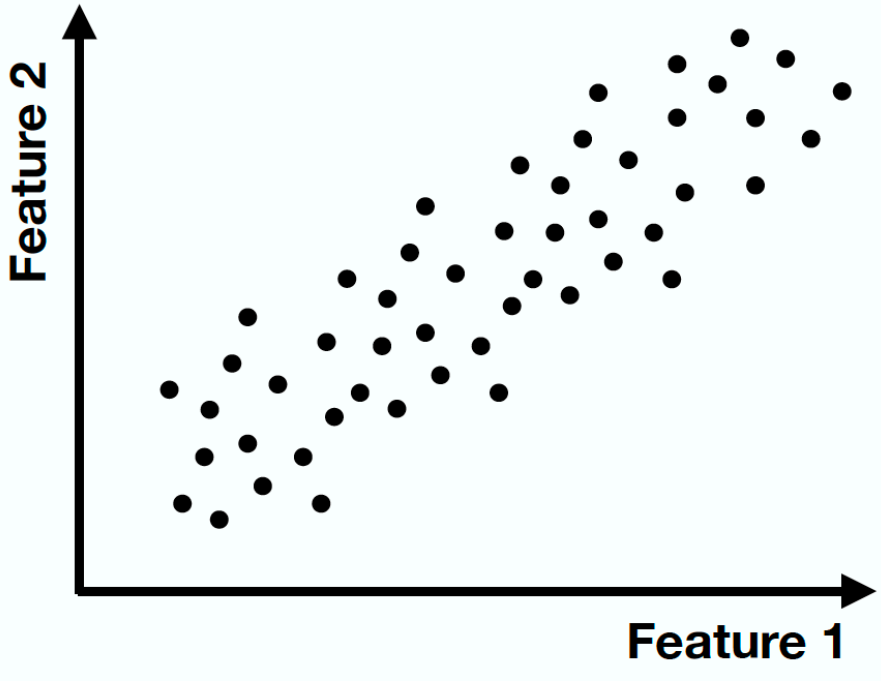

Credit: D. Baron (IAC XXX Winter School)

# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**The behavior will strongly depend on the structure of the input features**

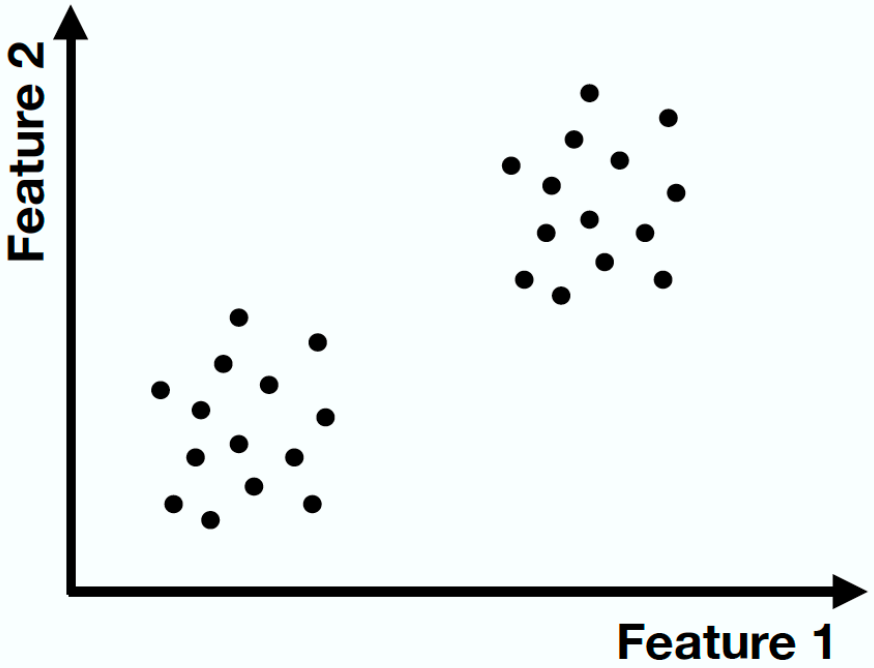


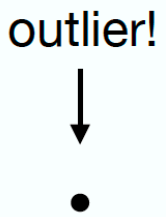
# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**The behavior will strongly depend on the structure of the input features**



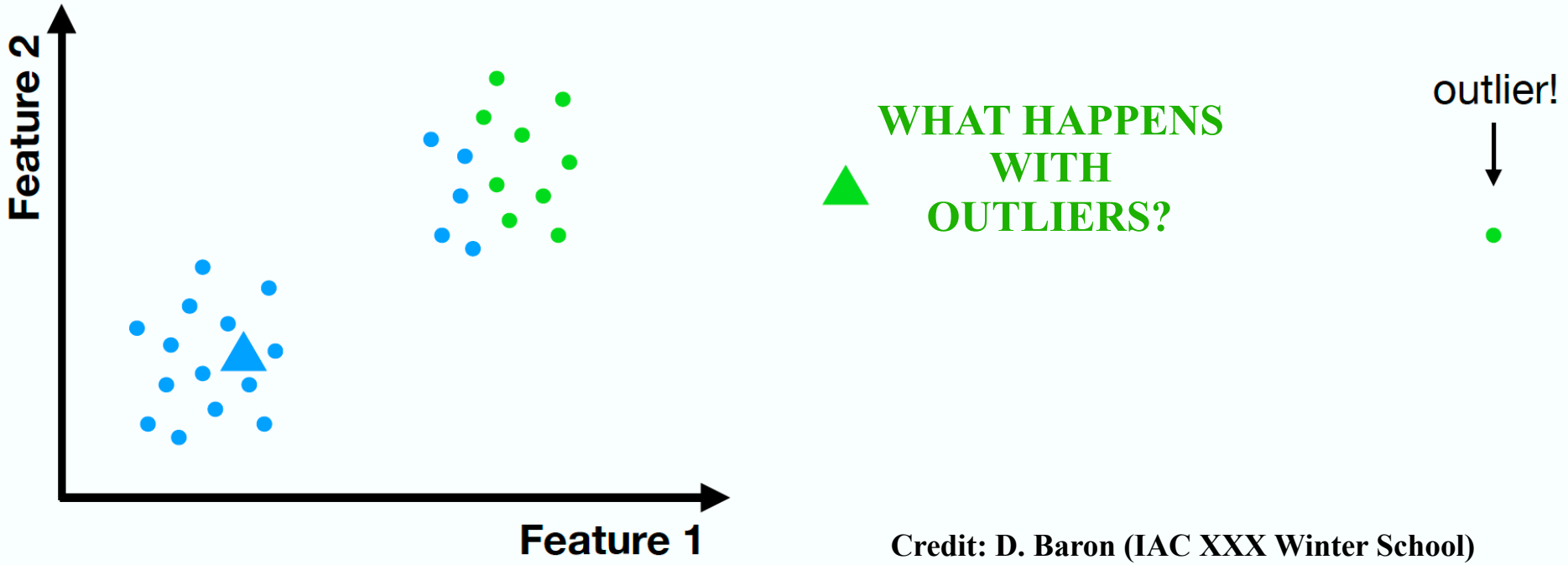

**WHAT HAPPENS WITH OUTLIERS?**



# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**The behavior will strongly depend on the structure of the input features**

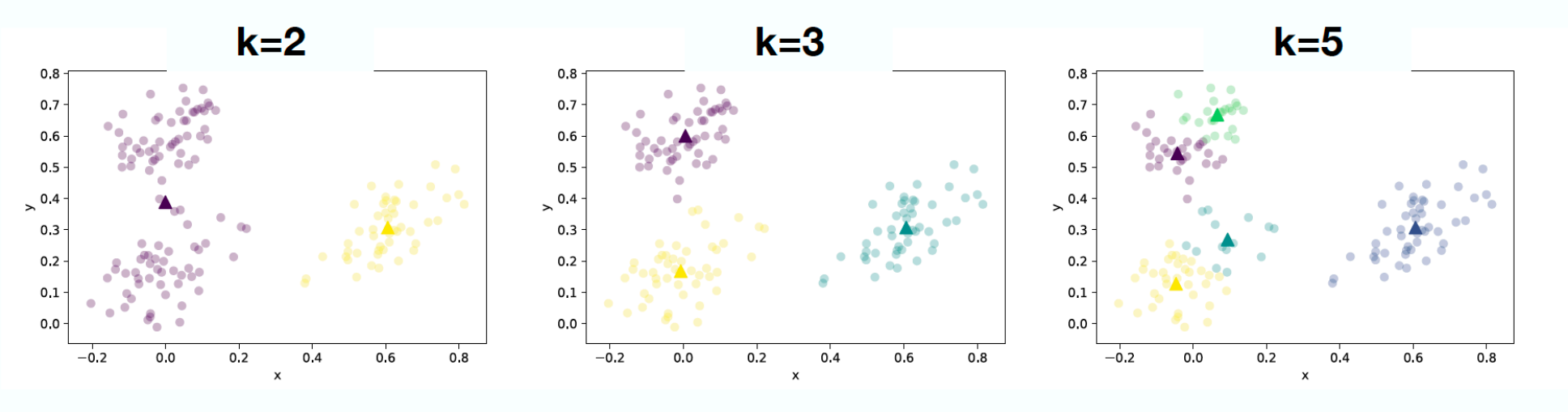



Credit: D. Baron (IAC XXX Winter School)

# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**Hyper-Parameters: The number of clusters needs to be fixed**



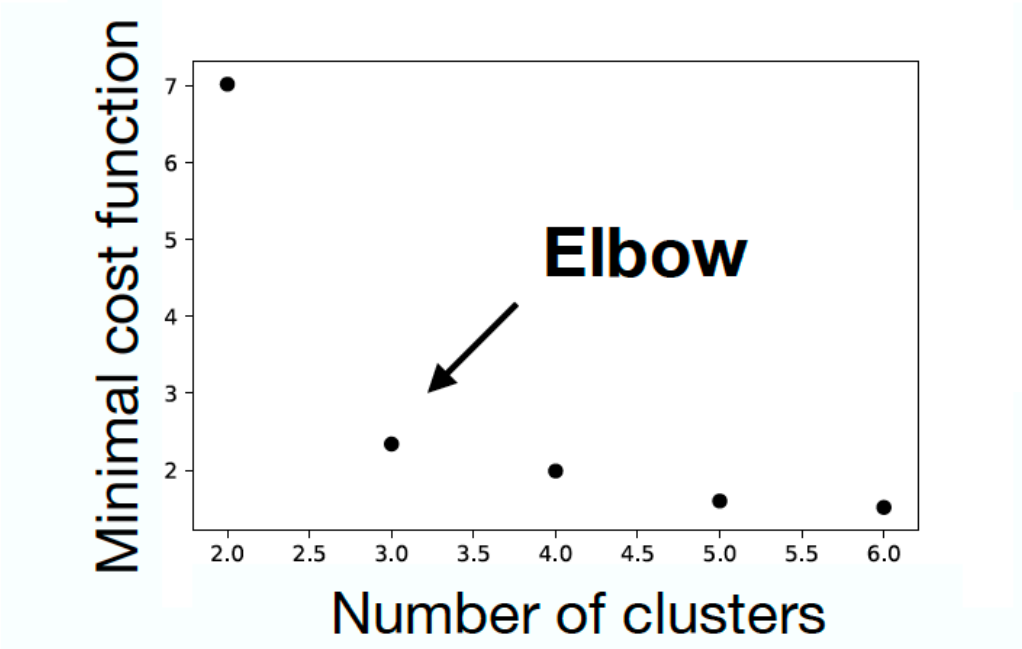

**Credit: D. Baron (IAC XXX Winter School)**

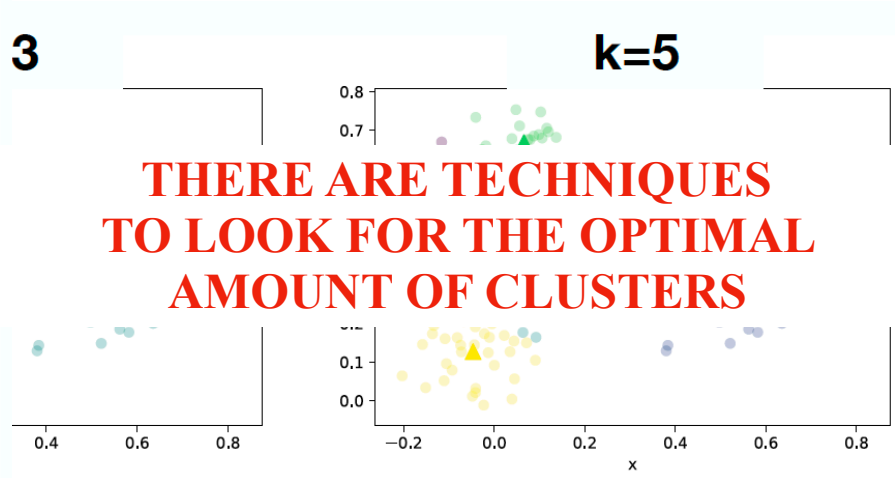
# K-MEANS IS VERY SENSITIVE TO SEVERAL CRITICAL CHOICES

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

**Hyper-Parameters: The number of clusters needs to be fixed**

98bis Boulevard Arago

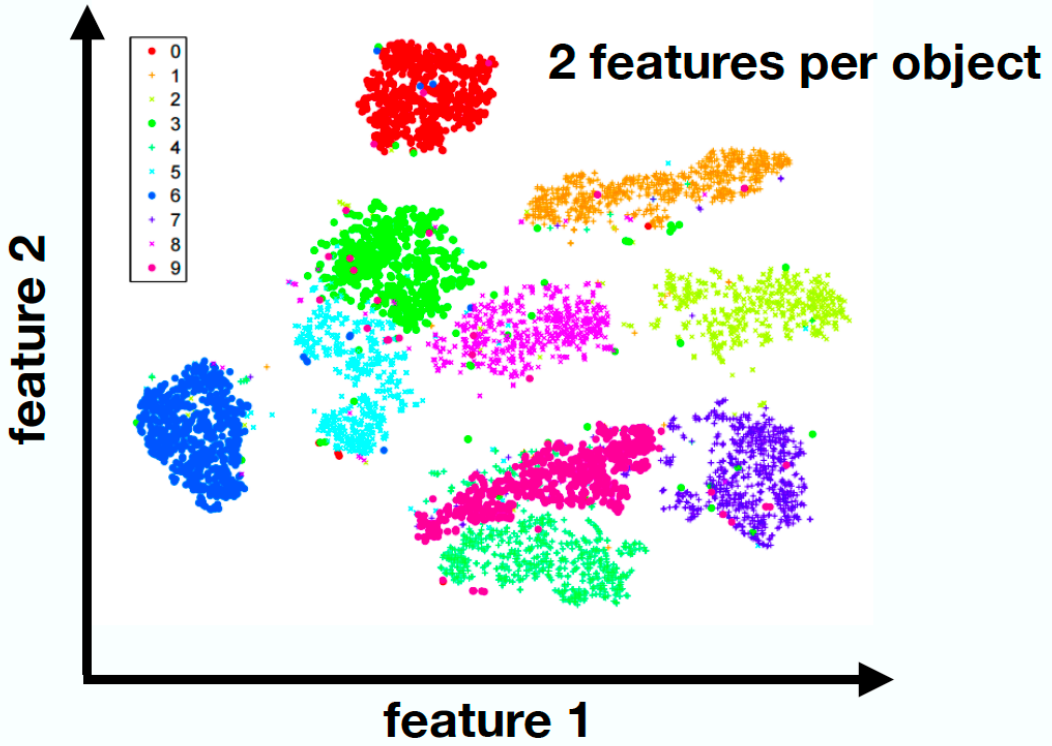



Credit: D. Baron (IAC XXX Winter School)

28 x 28 features per object

Dimensionality Reduction algorithm




# DIMENSIONALITY REDUCTION ALGORITHMS

Credit: Baron IAC XXX Winter School

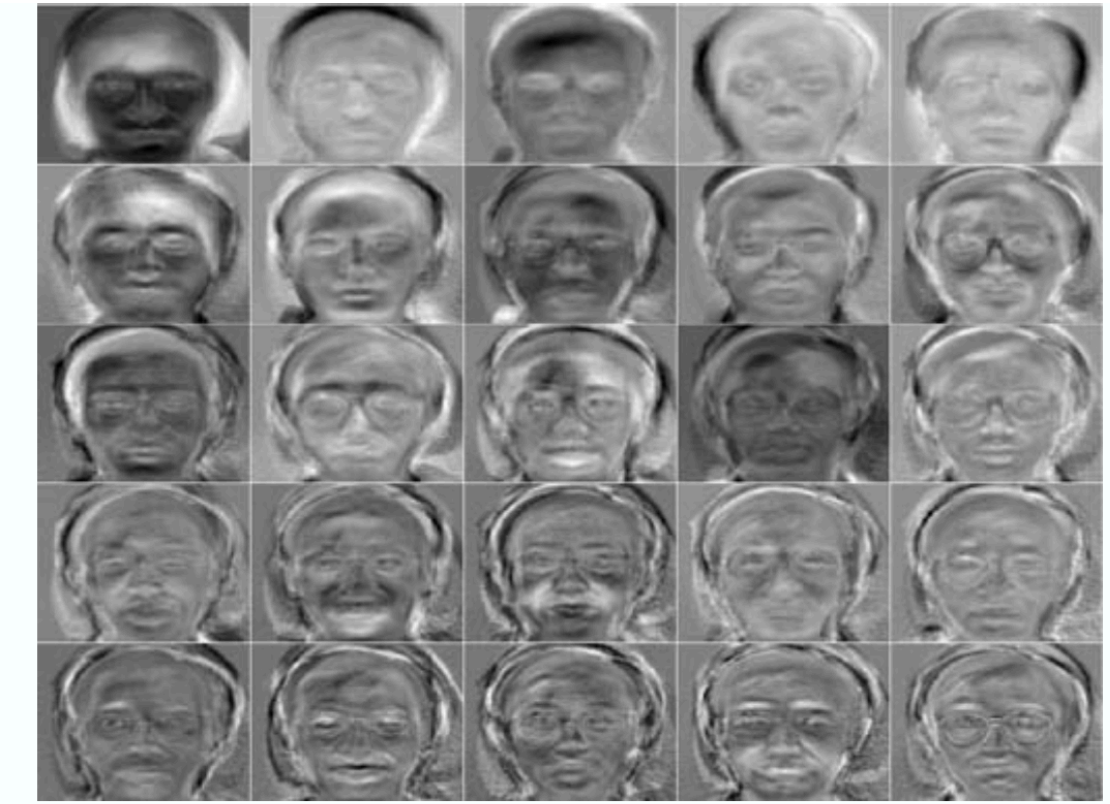
# WHY IS DIMENSIONALITY REDUCTION USEFUL?

- DATA COMPRESSION
- IMPROVE PERFORMANCE OF SUPERVISED LEARNING BY REDUCING REDUNDANCY IN ORIGINAL FEATURES
- DATA VISUALIZATION / INTERPRETATION
- UNCOVERING COMPLEX TRENDS
- OUTLIERS: UNKNOWN UNKNOWNS

# THERE ARE TYPICALLY TWO TYPES OF DIMENSIONALITY REDUCTION ALGORITHMS

## 1. PROTOTYPE BASED

**WE LOOK FOR PROTOTYPES THAN COMBINED, CAN REPRESENT MOST OF THE VARIANCE CONTAINED IN THE DATA**

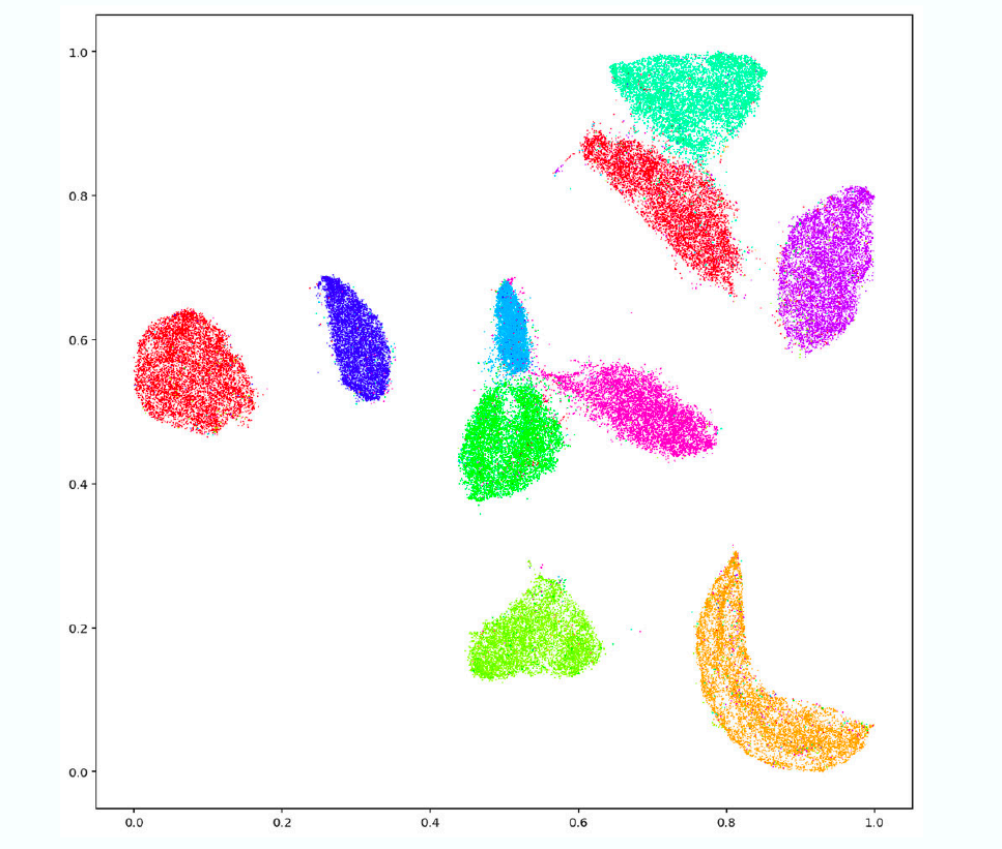


**EXAMPLES ARE: PCA, SOMS, SVD, ICA…**

# THERE ARE TYPICALLY TWO TYPES OF DIMENSIONALITY REDUCTION ALGORITHMS

## 1. EMBEDDING

**THEY EMBED A HIGH DIMENSIONAL DATASET INTO A LOWER DIMENSION**

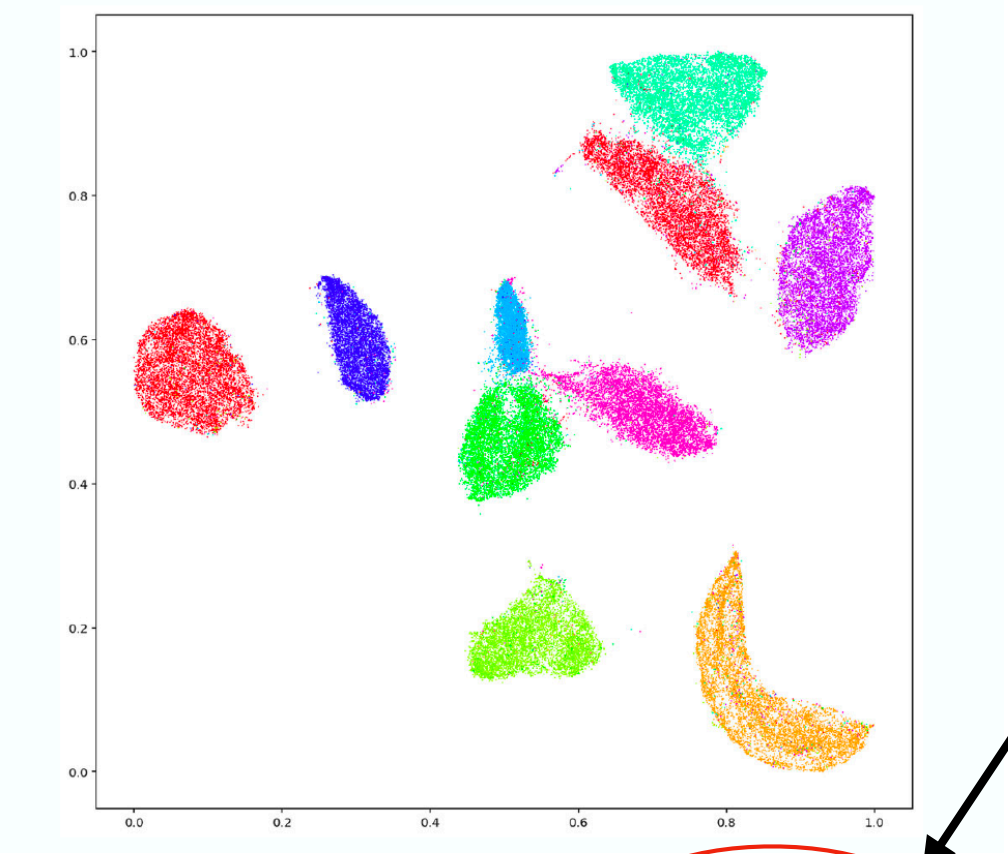



**EXAMPLES ARE: tSNE, *VAEs, GANs* ..**

# THERE ARE TYPICALLY TWO TYPES OF DIMENSIONALITY REDUCTION ALGORITHMS

## 1. EMBEDDING

**THEY EMBED A HIGH DIMENSIONAL DATASET INTO A LOWER DIMENSION**




**deep learning**

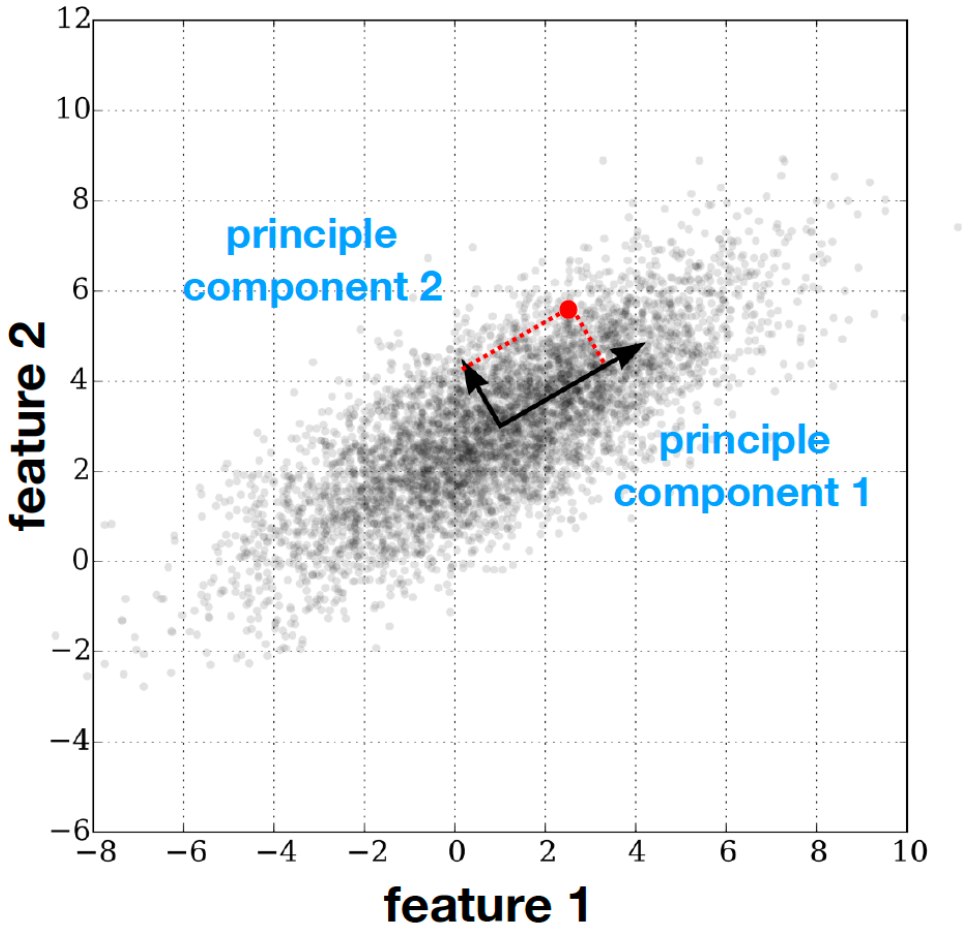
**EXAMPLES ARE: tSNE, *VAEs, GANs* ..**

# PRINCIPAL COMPONENT ANALYSIS

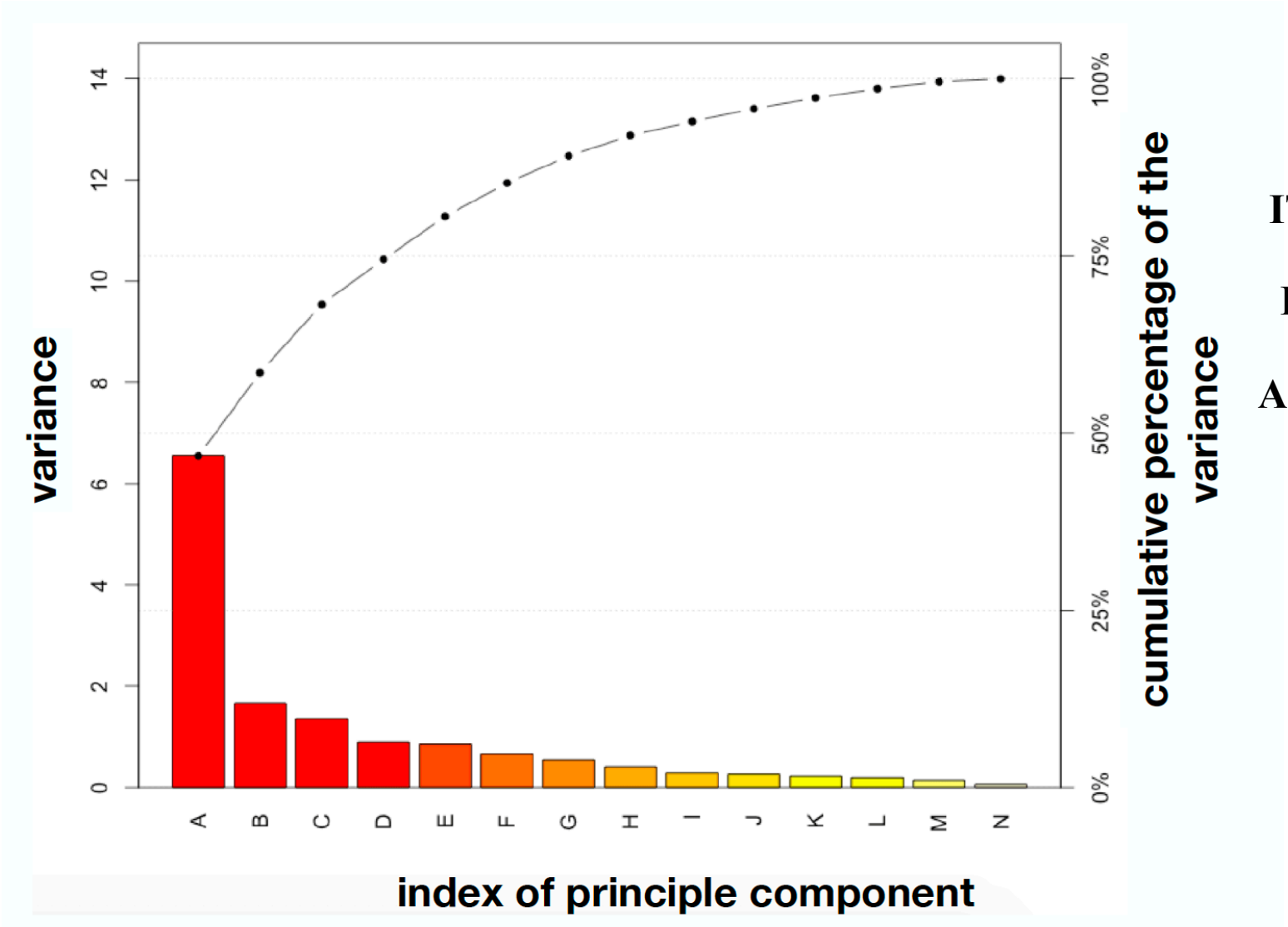
**PCA CONVERT A SET OF (CORRELATED) VARIABLES INTO A SET OF VALUES LINEARLY UNCORRELATED**

**1. THE FIRST PRINCIPLE COMPONENT ("PROTOTYPE"), HAS THE LARGEST POSSIBLE VARIANCE**

**2. THE FOLLOWING COMPONENTS HAVE THE LARGEST VARIABLES WITH THE ADDITIONAL CONSTRAIN THAT THEY ARE ORTHOGONAL TO THE PRECEDING COMPONENTS**



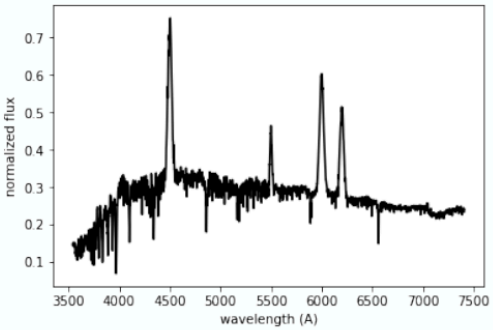
# PRINCIPAL COMPONENT ANALYSIS




**IT RESULTS IN DATA COMPRESSION, BY REPRESENTING EACH OBJECT AS A PROJECTION OF THE FIRST PRINCIPLE COMPONENTS**

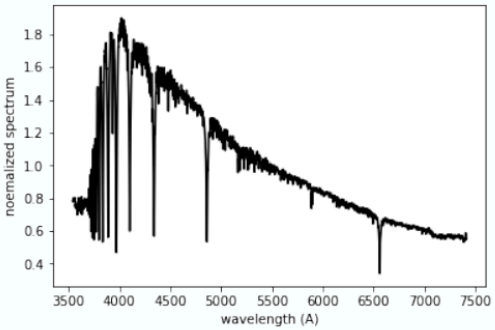
# PRINCIPAL COMPONENT ANALYSIS

**observed object**



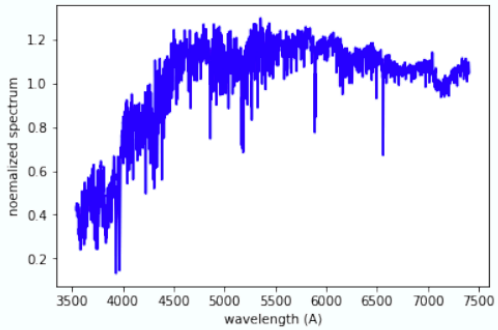

**= A ***

**principle comp. 1**



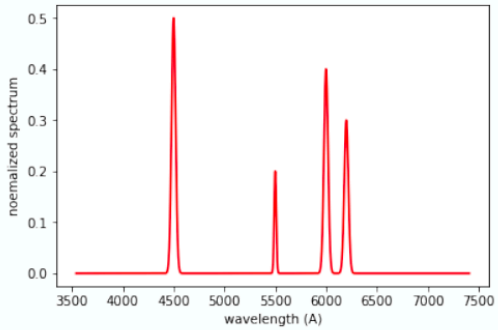

**+ B ***

**principle comp. 2**



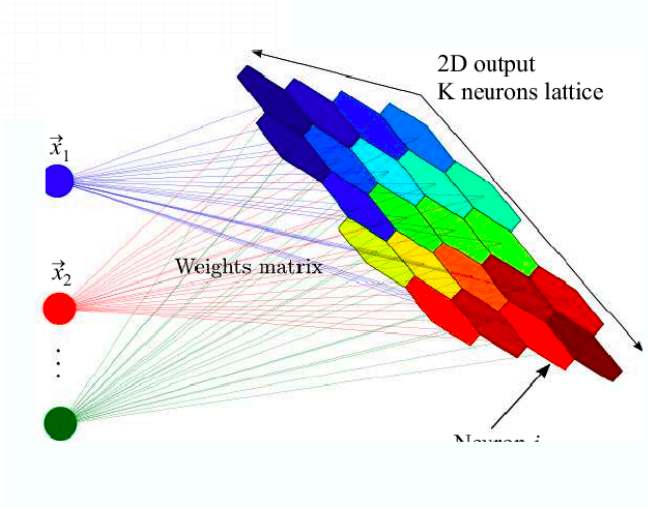

**+ C ***

**principle comp. 3**

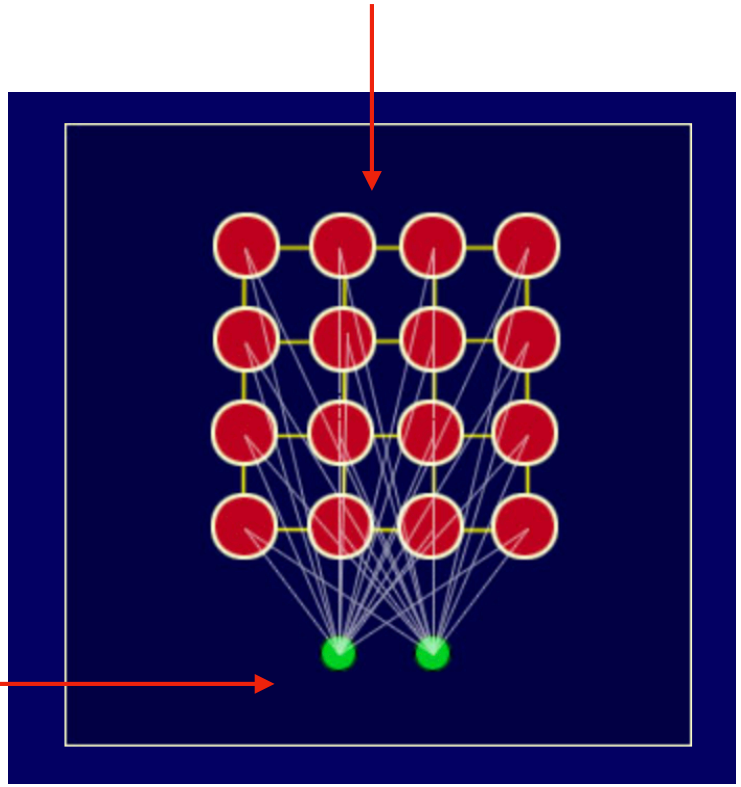



**OBJECTS ARE DIVIDED INTO MAJOR PROTOTYPES AND ALL OBJECTS CAN BE OBTAINED AS LINEAR COMBINATIONS**

# SELF ORGANIZING MAPS




**COMPLETION LAYER**
**("NEURONS" ARRANGE GEOMETRICALLY)**



**INPUT LAYER**
**(FEATURES)**

**EACH "NODE" IN THE SOM HAS THE SAME NUMBER OF WEIGHTS THAN INPUT VECTORS**

# SELF ORGANIZING MAPS

**1. EACH NODE'S WEIGHTS ARE INITIALIZED RANDOMLY**

# SELF ORGANIZING MAPS

**2. A VECTOR IS CHOSEN AT RANDOM FROM THE SET OF TRAINING DATA AND PRESENTED TO THE LATTICE**

$$D = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2}$$

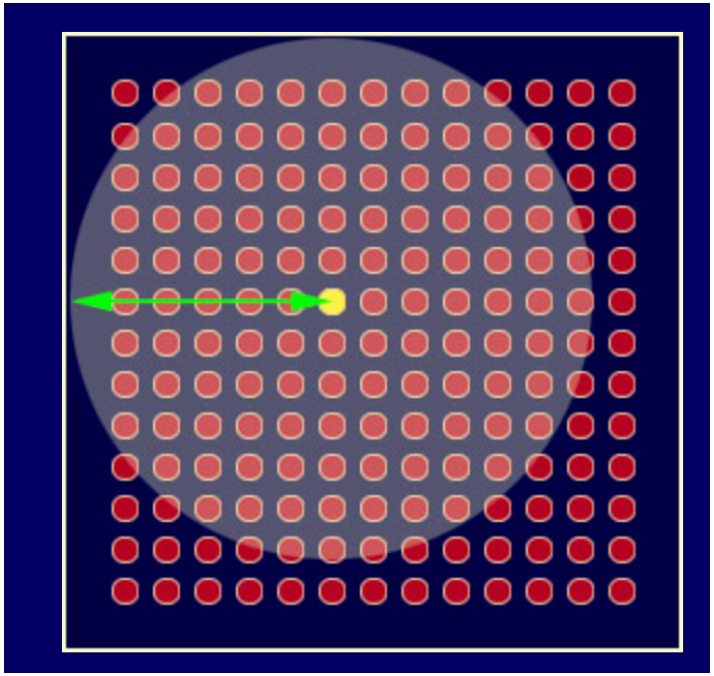
**3. EVERY NODE IS EXAMINED TO CALCULATE WHICH ONE'S WEIGHTS ARE MOST LIKE THE INPUT VECTOR. THE WINNING NODE IS COMMONLY KNOWN AS THE BEST MATCHING UNIT (BMU).**

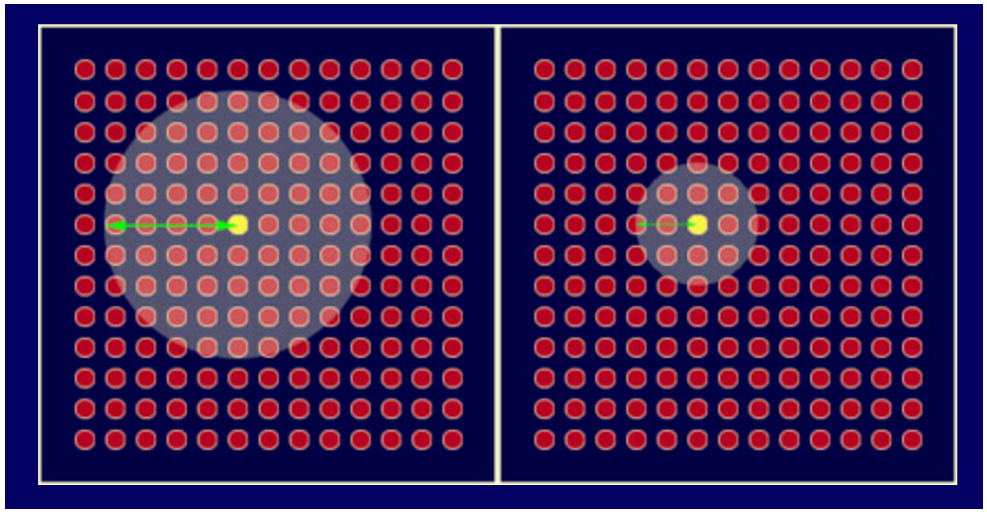
# SELF ORGANIZING MAPS

**4. THE RADIUS OF THE NEIGHBORHOOD OF THE BMU IS NOW CALCULATED. THIS IS A VALUE THAT STARTS LARGE, TYPICALLY SET TO THE 'RADIUS' OF THE LATTICE, BUT DIMINISHES EACH TIME-STEP.**

$$\sigma(t) = \sigma_0 \times e^{-t/\lambda}$$

**THE RADIUS DECREASES OVER TIME**

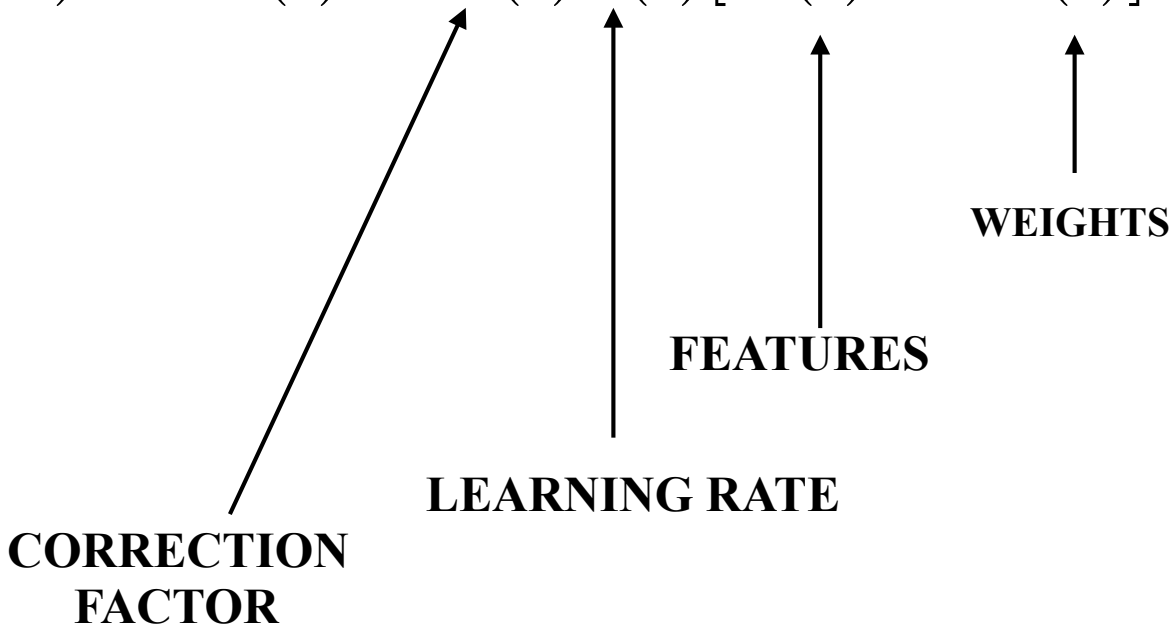




# SELF ORGANIZING MAPS

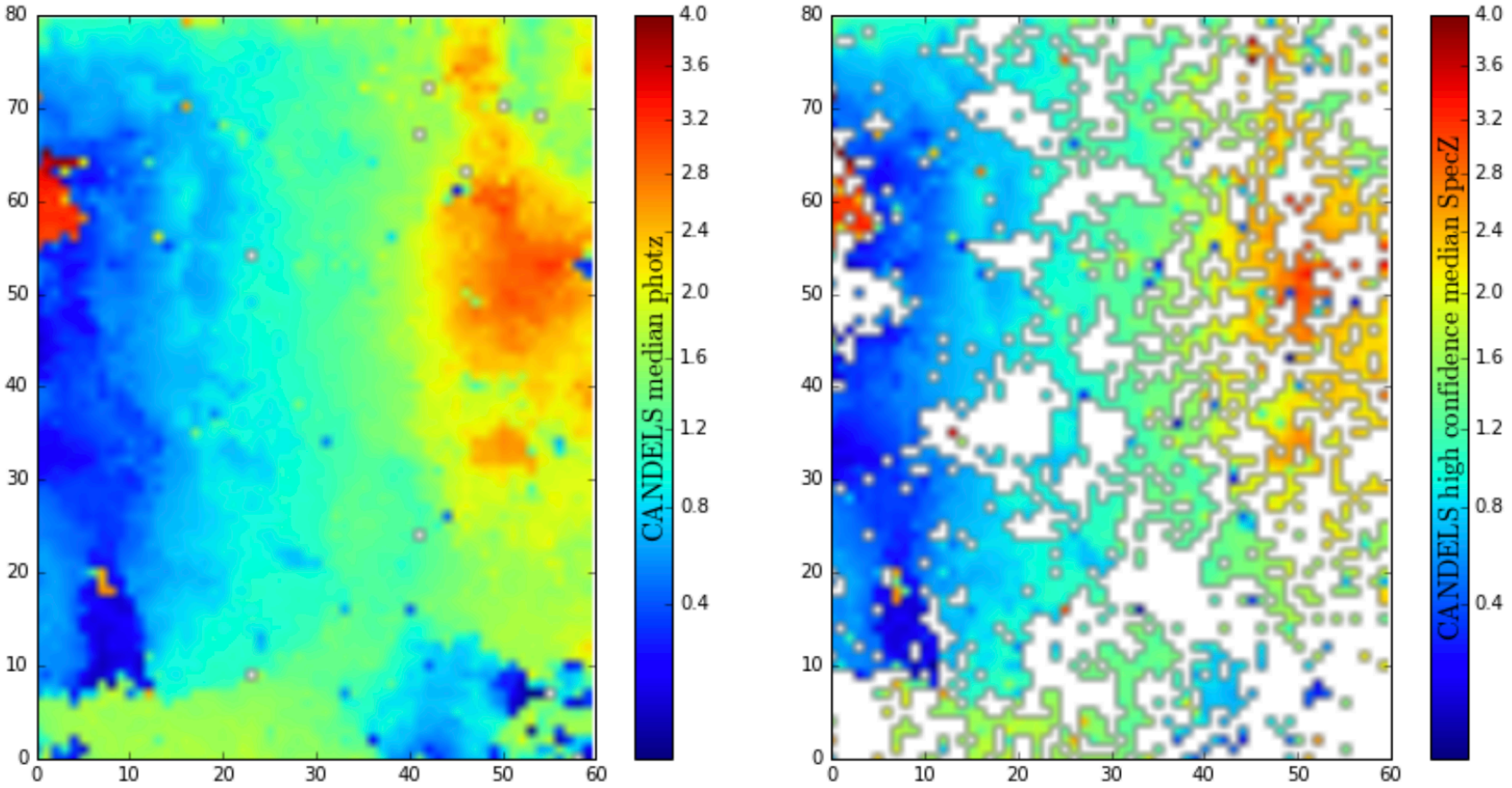
**5. EACH NEIGHBORING NODE'S (THE NODES FOUND IN STEP 4) WEIGHTS ARE ADJUSTED TO MAKE THEM MORE LIKE THE INPUT VECTOR. THE CLOSER A NODE IS TO THE BMU, THE MORE ITS WEIGHTS GET ALTERED.**

$$W(t+1) = W(t) + \Psi(t)\lambda(t)[V(t) - W(t)]$$



# SELF ORGANIZING MAPS

**6. REPEAT FROM 2 FOR N ITERATIONS**

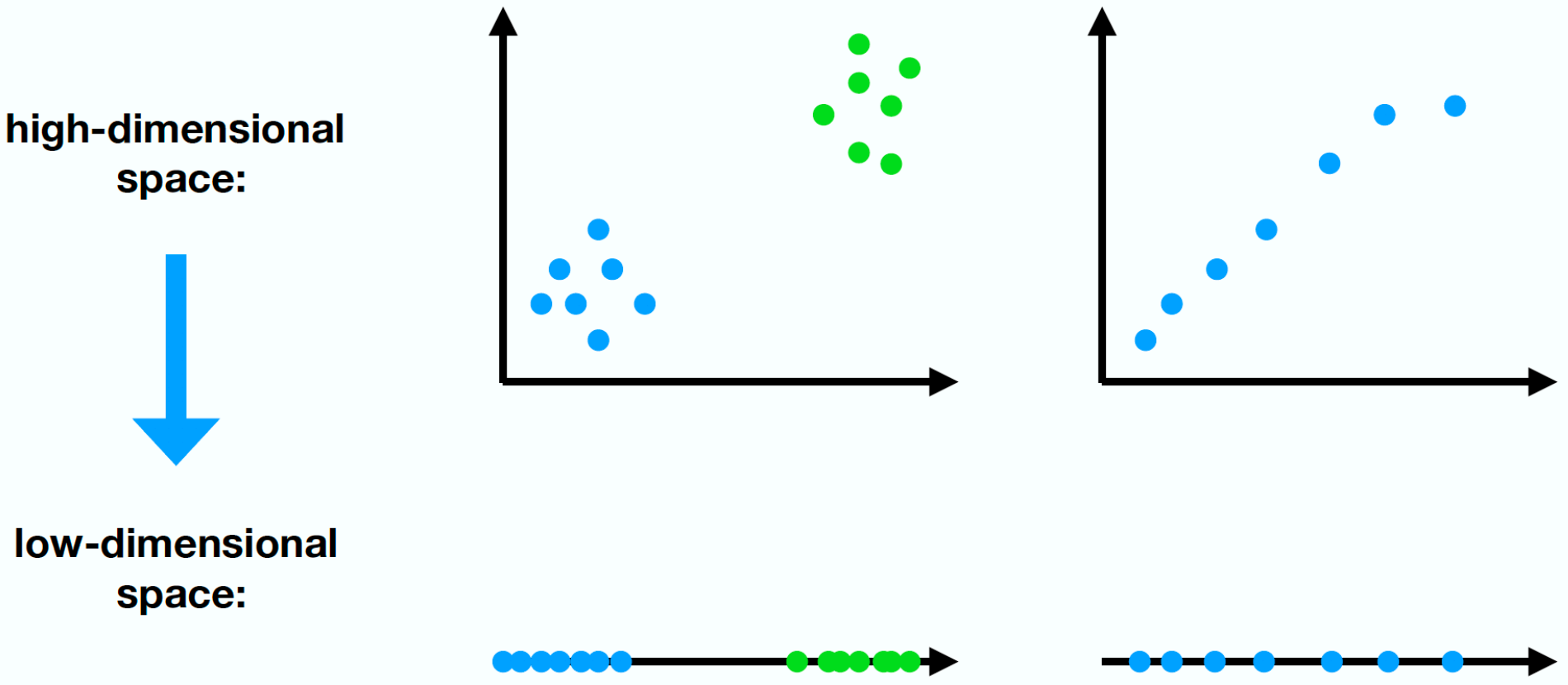



**HEMMATI ET AL. 2018**

**<u>SOMs FOR PHOTOMETRIC REDSHIFTS</u>**

# tSNE
# t-distributed stochastic neighbor embedding

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**

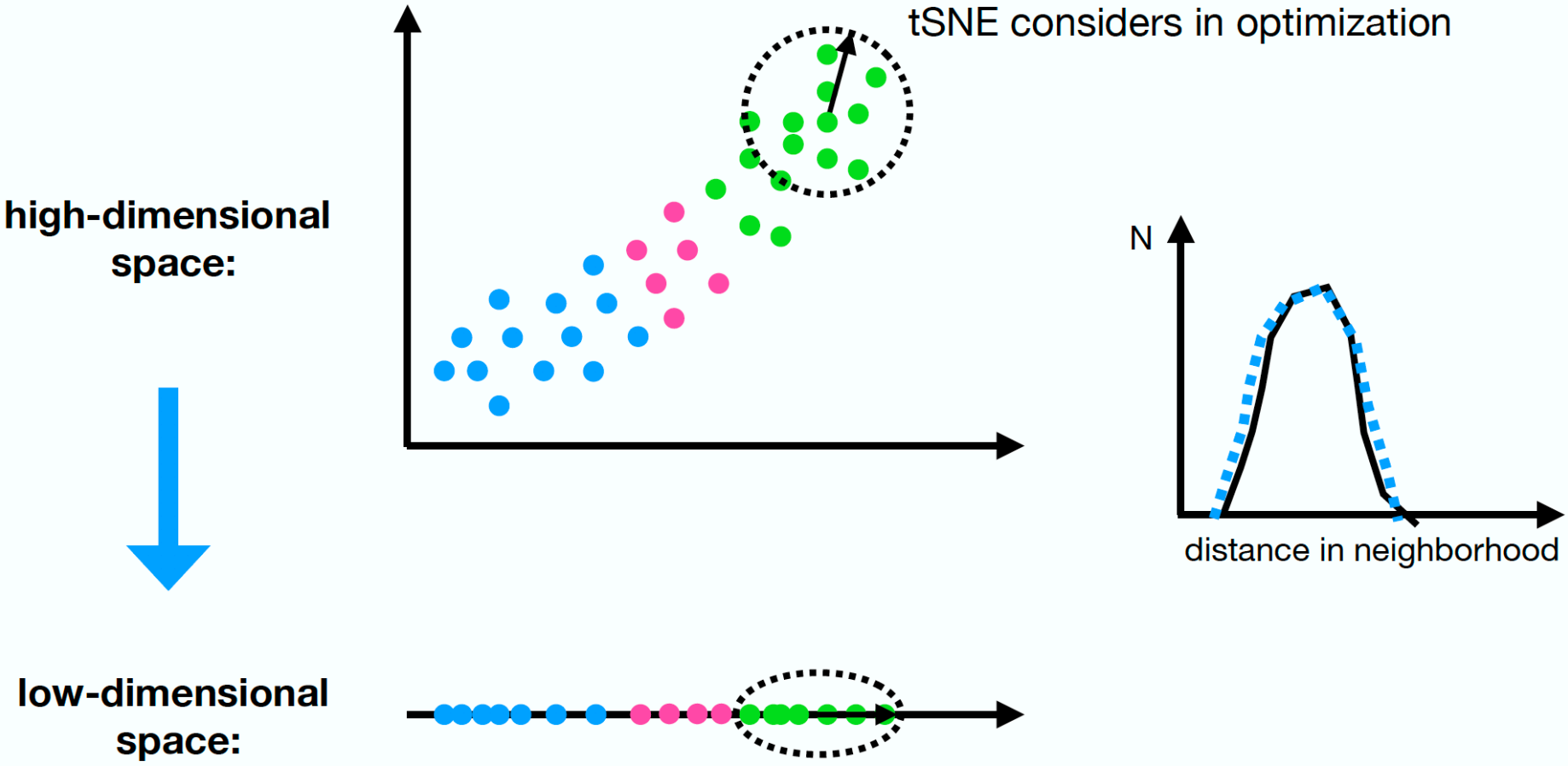



Credit: Baron IAC XXX Winter

# tSNE
# t-distributed stochastic neighbor embedding

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

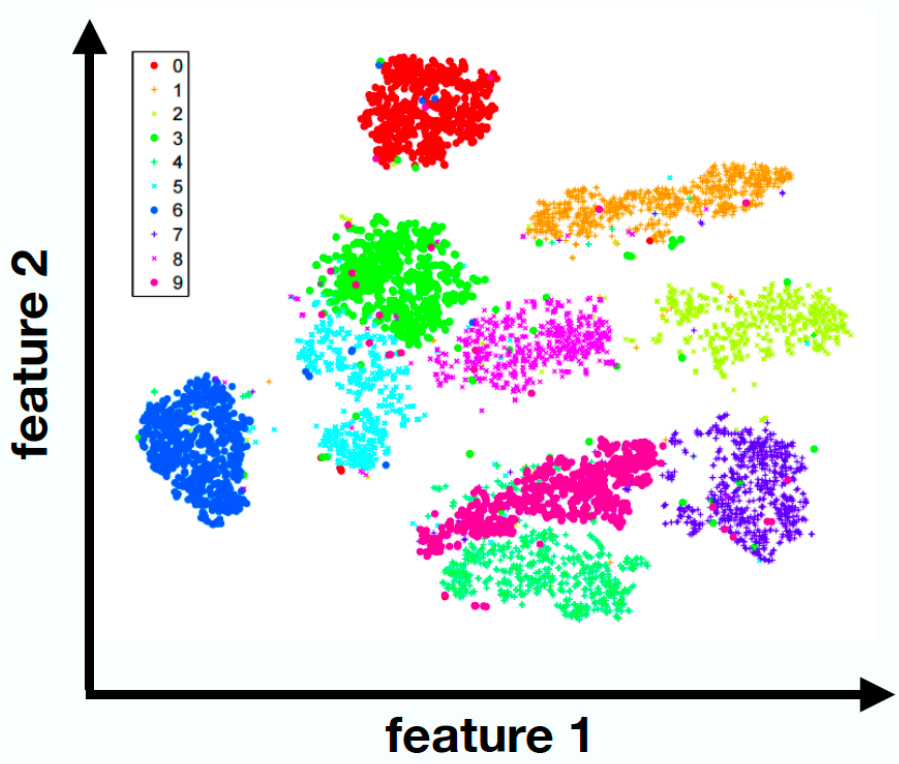
**perplexity:** the neighborhood that tSNE considers in optimization




**Credit: Baron IAC XXX Winter**

# tSNE
# t-distributed stochastic neighbor embedding

28 x 28 features per object




Credit: Baron IAC XXX Winter