

Clustering

CS545 - Machine Learning for Signal Processing

Today's lecture

- Clustering
- Hierarchical models
- K-means et al.
- Spectral clustering
- Gaussian mixtures and EM

Back to unsupervised

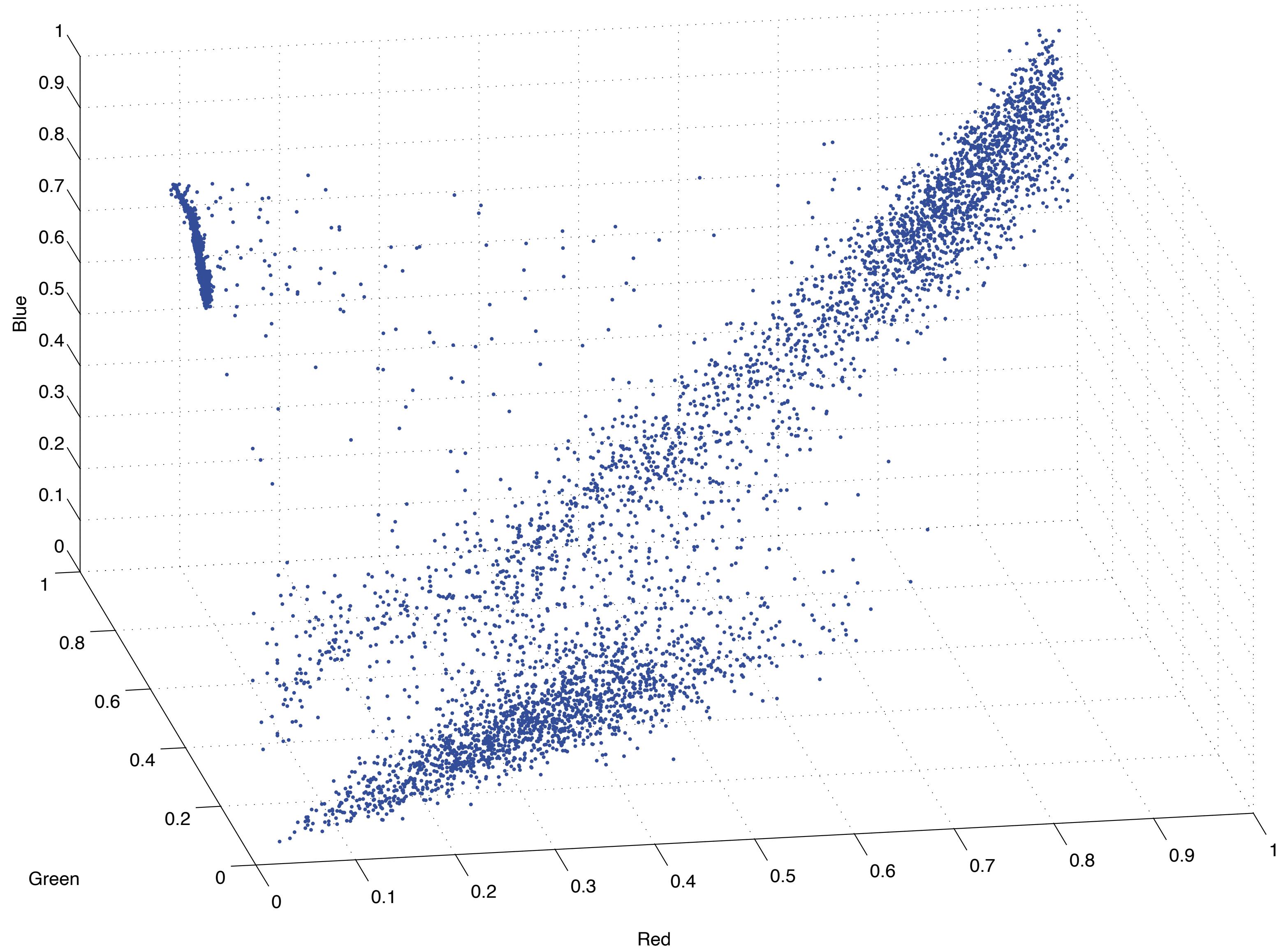
- Supervised learning
 - Use labeled data to do something smart
- What if the labels don't exist?

Some inspiration



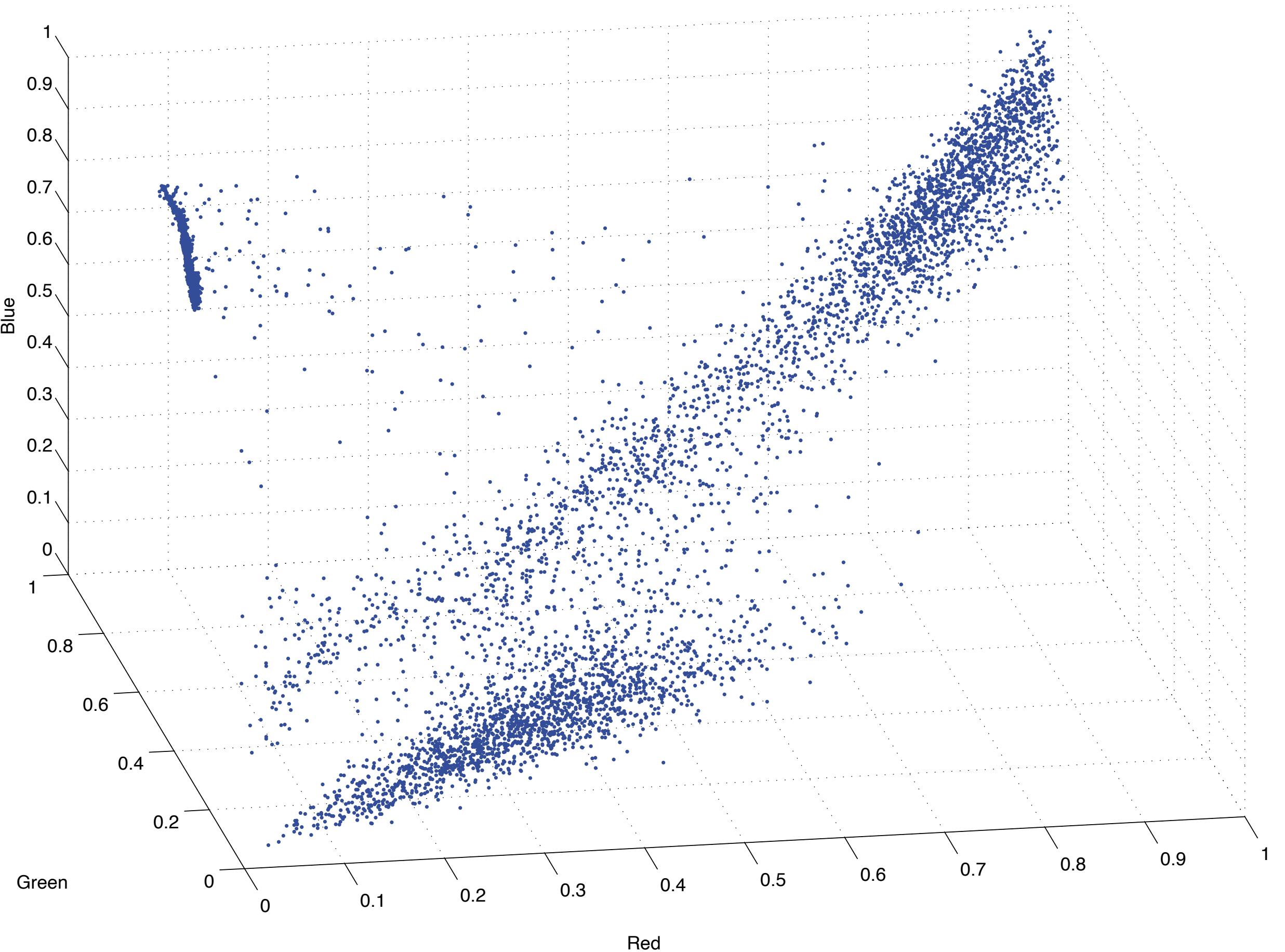
El Capitan, Yosemite National Park

The way we'll see it



A new question

- I see classes, but ...
- How do I find them?
 - Can I automate this?
 - How many are there?
- Answer: Clustering



Clustering

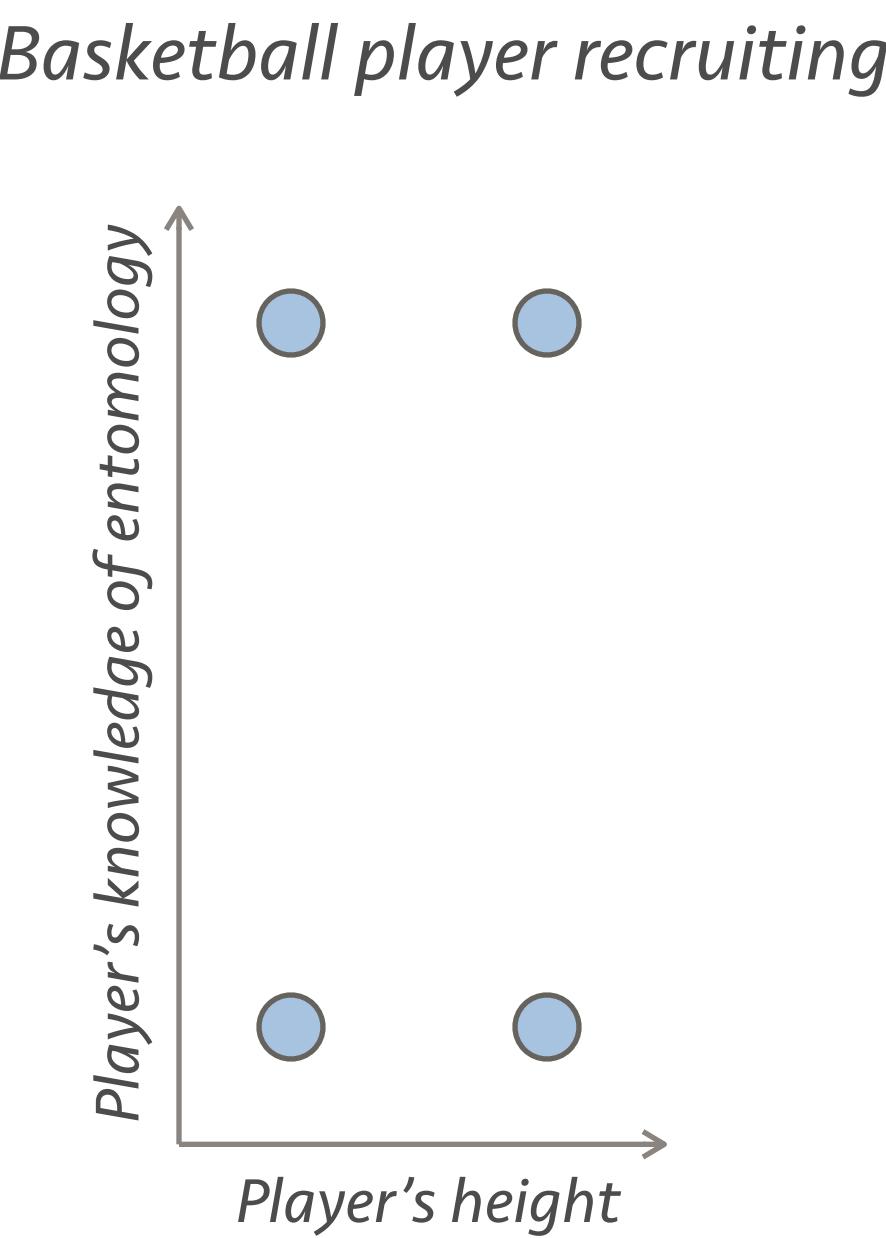
- Discover classes in data
 - Divide data in sensible clusters
- Fundamentally ill-defined problem
 - There is often no correct solution
- Relies on many user choices

Clustering process

- Describe your data using features
 - What's your objective?
- Define a proximity measure
 - How is the feature space shaped?
- Define a clustering criterion
 - When do samples make a cluster?

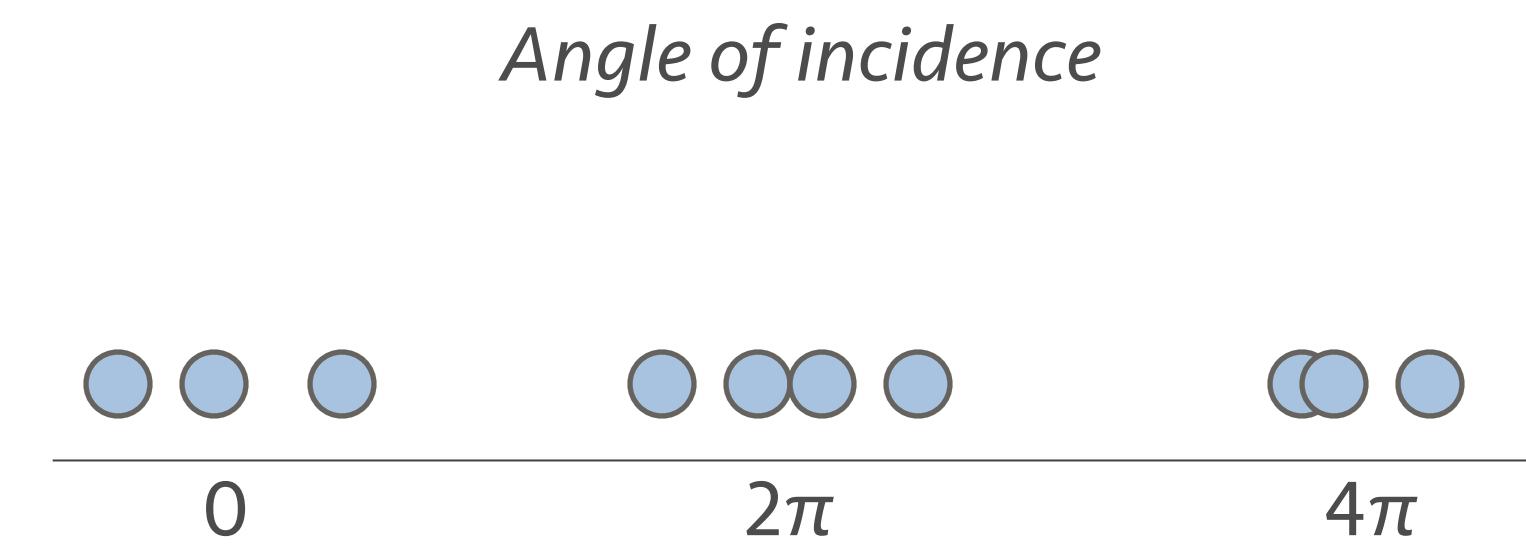
Know what you want

- Features & desired objective matter
 - Which are the two classes?



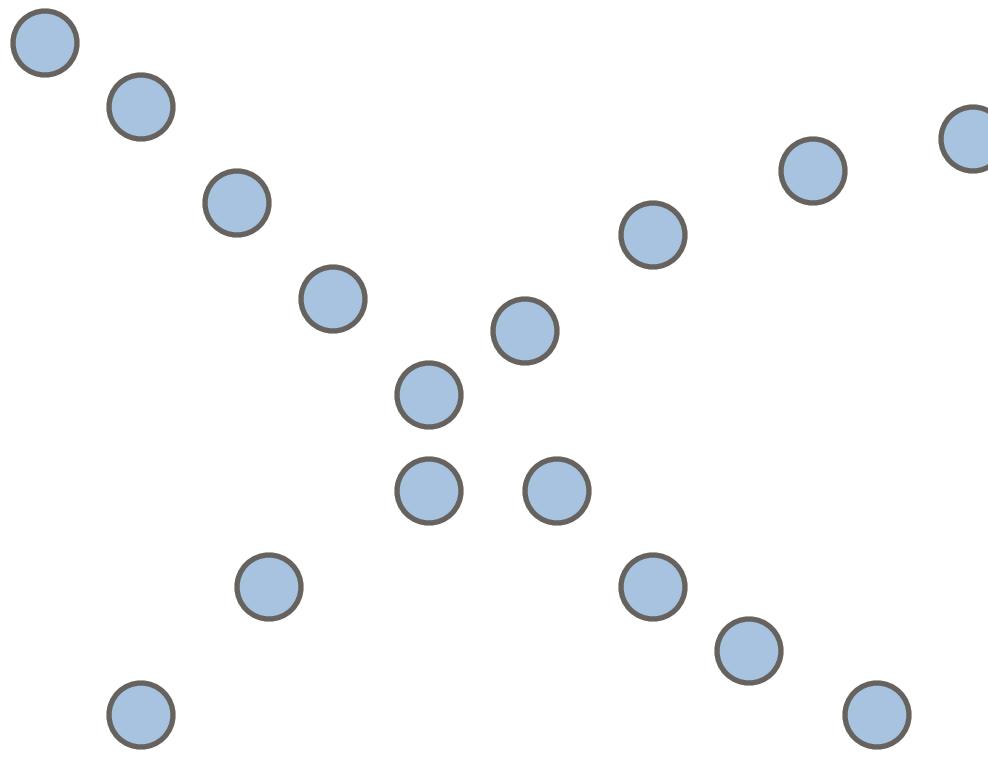
Know your space

- Define a sensible proximity measure



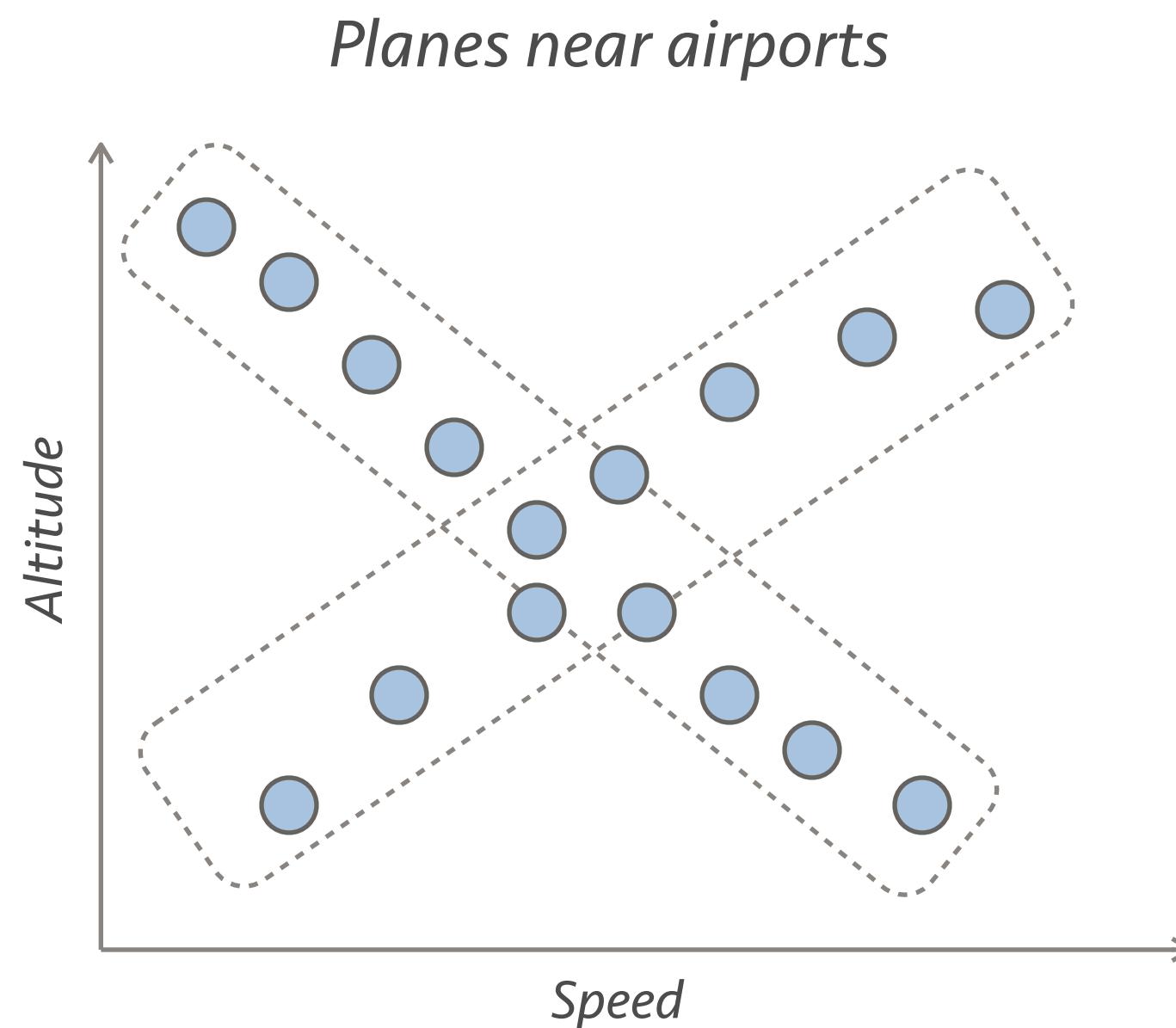
Know your cluster type

- What forms a cluster in your space?



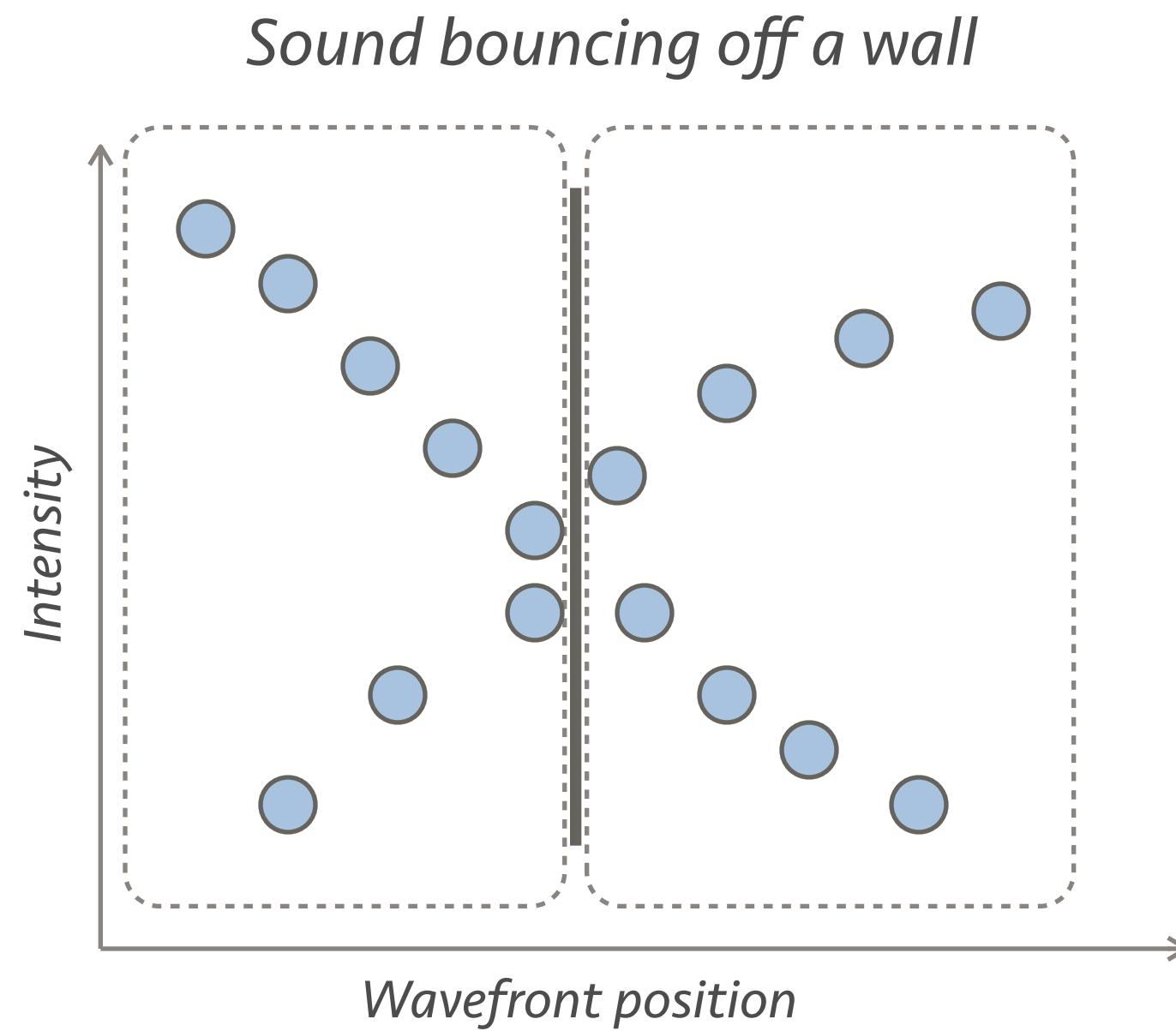
Know your cluster type

- What forms a cluster in your space?



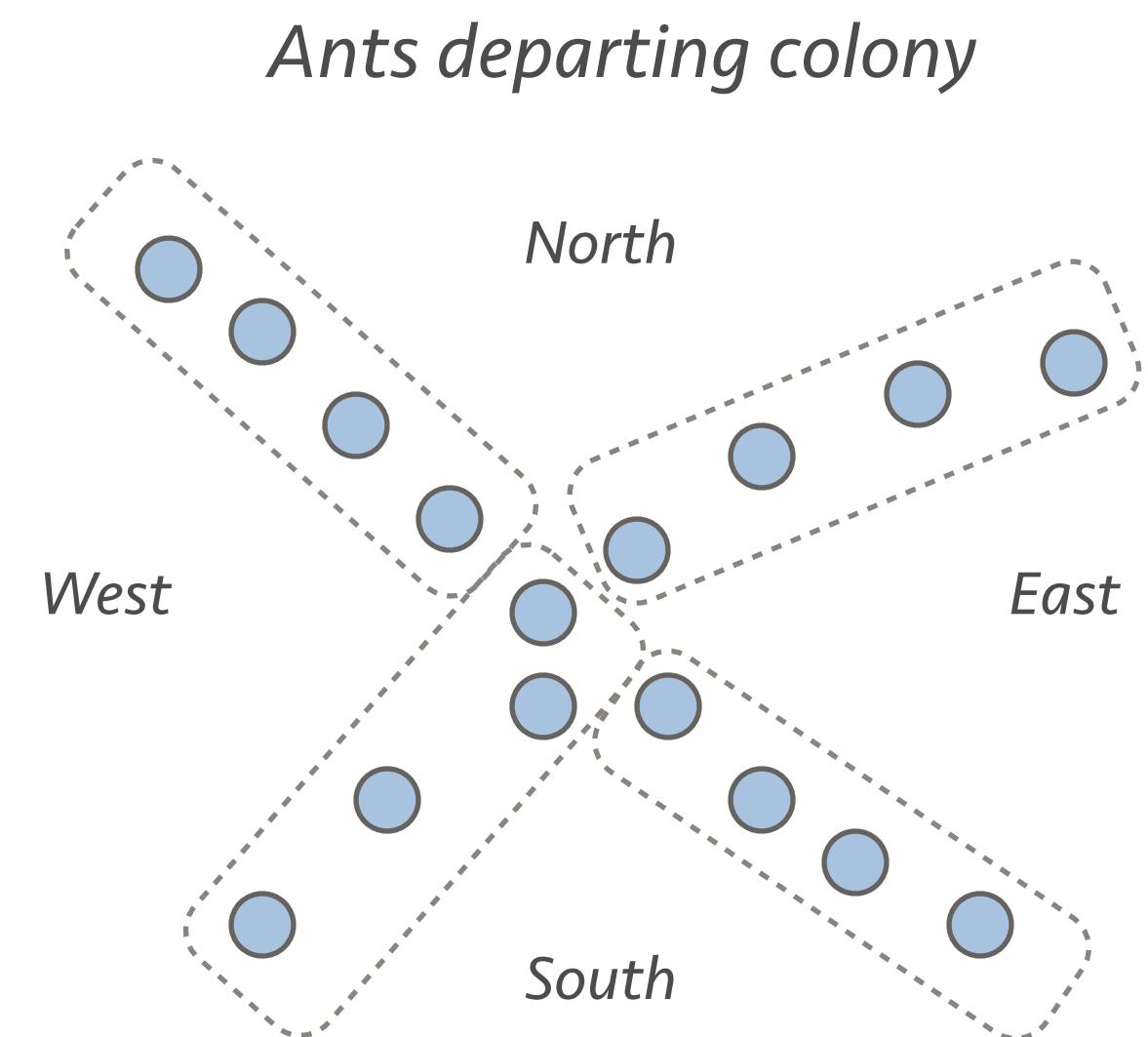
Know your cluster type

- What forms a cluster in your space?



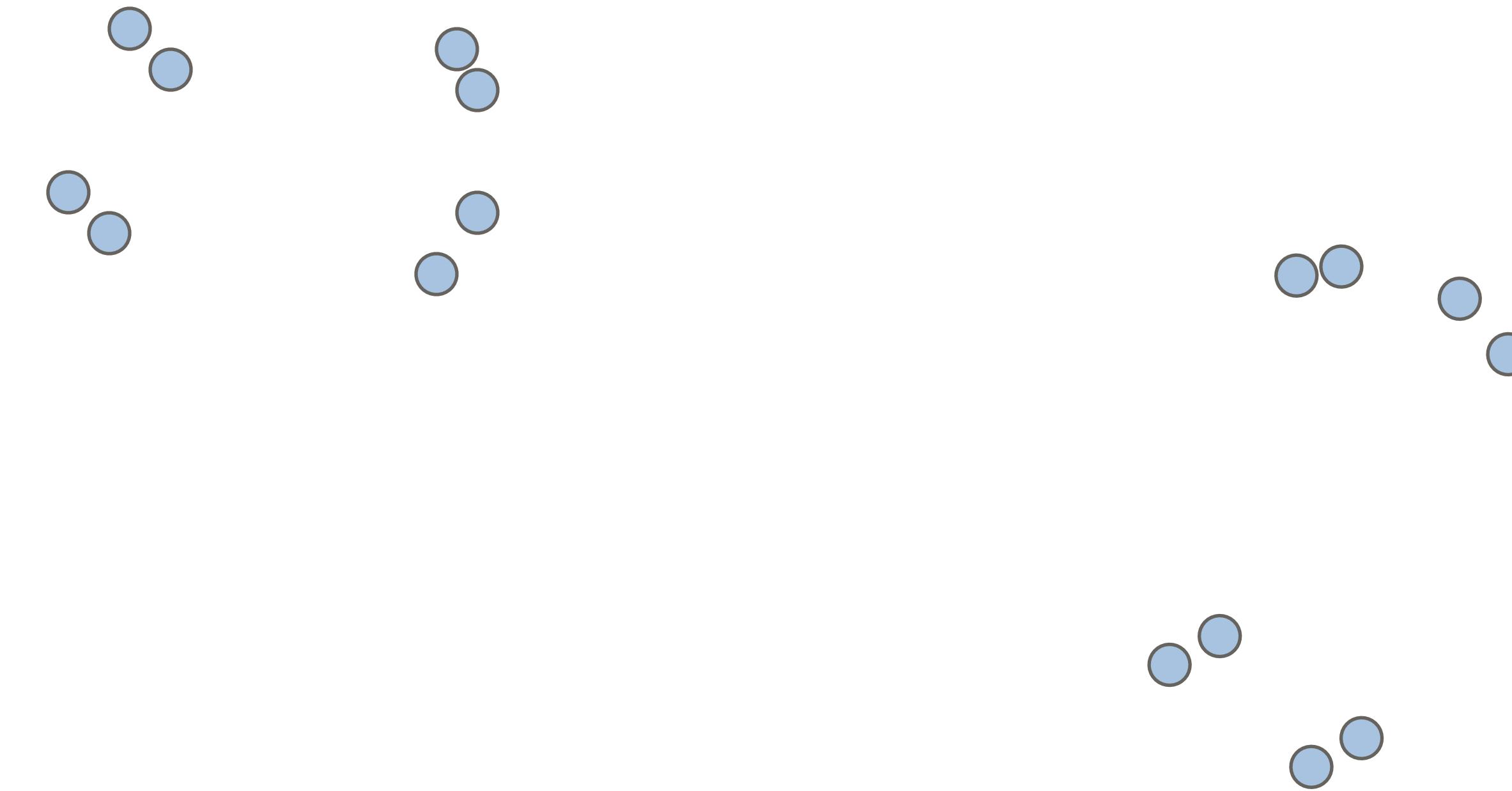
Know your cluster type

- What forms a cluster in your space?



How many clusters?

- The deeper you look the more you'll get



There are no right answers!

- Part of clustering is an art
- You need to experiment to get there
- But some good starting points exist

How to cluster

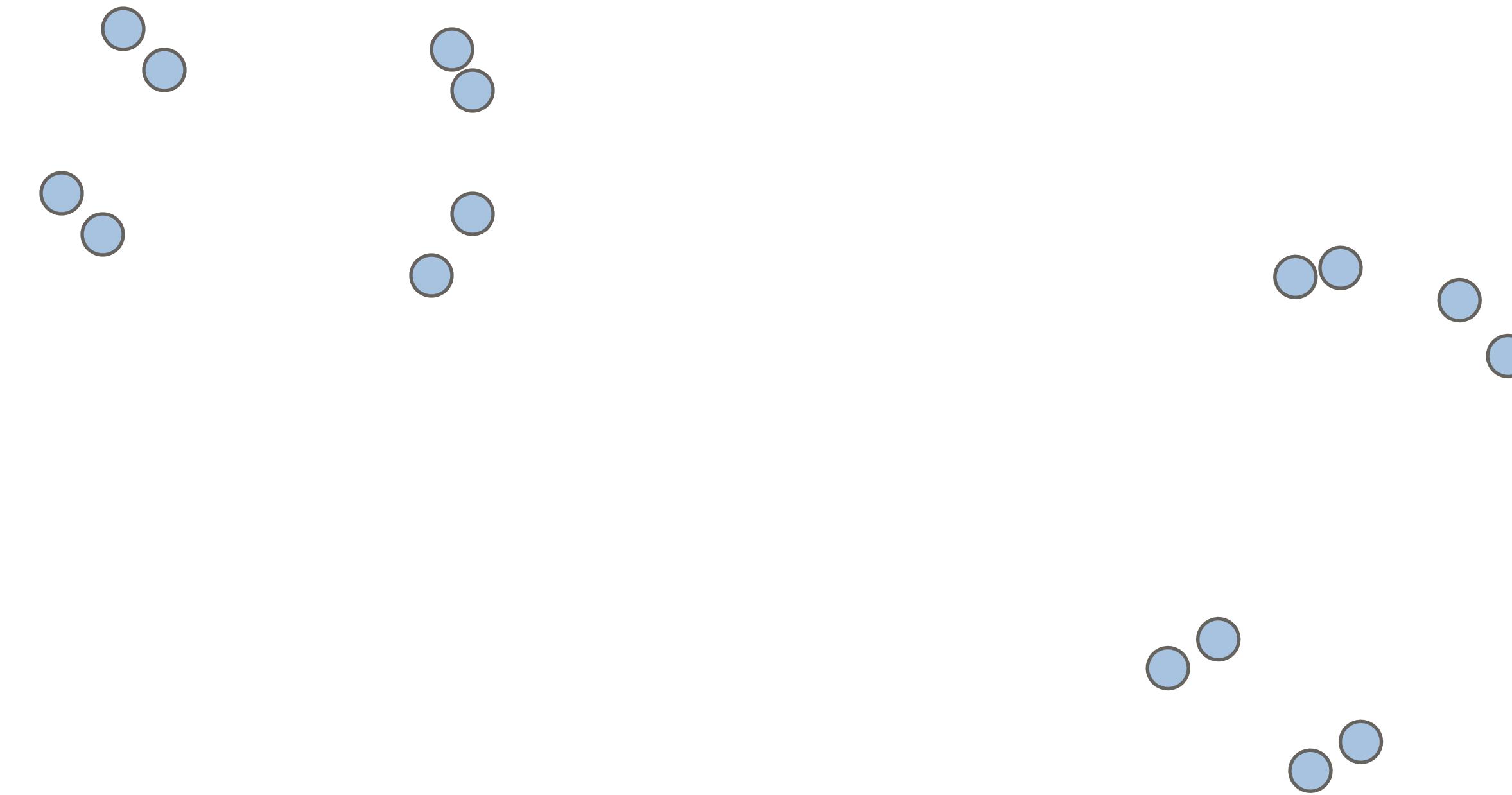
- Tons of methods
- We can use step-based logical steps
 - e.g., find two closest point and merge, repeat
- Or formulate a global criterion

Hierarchical methods

- Agglomerative algorithms
 - Keep pairing up your data
- Divisive algorithms
 - Keep breaking up your data

Agglomerative Approach

- Look at your data points and form pairs
 - Keep at it



More formally

- Represent data as vectors:

$$\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, N\}$$

- Represent clusters by: C_j
- Represent the clustering by:

$$R = \{C_j, j = 1, \dots, m\}$$

$$\text{e.g. } R = \{\{\mathbf{x}_1, \mathbf{x}_3\}, \{\mathbf{x}_2\}, \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}\}$$

- Define a distance measure: $d(C_j, C_i)$

Agglomerative clustering

- Choose: $R_0 = \{C_i = \{\mathbf{x}_i\}, i = 1, \dots, N\}$
- For $t = 1, \dots$
 - Among all clusters in R_{t-1} , find cluster pair $\{C_i, C_j\}$ such that:

$$\arg \min_{i,j} d(C_i, C_j)$$

- Form new cluster and replace pair:

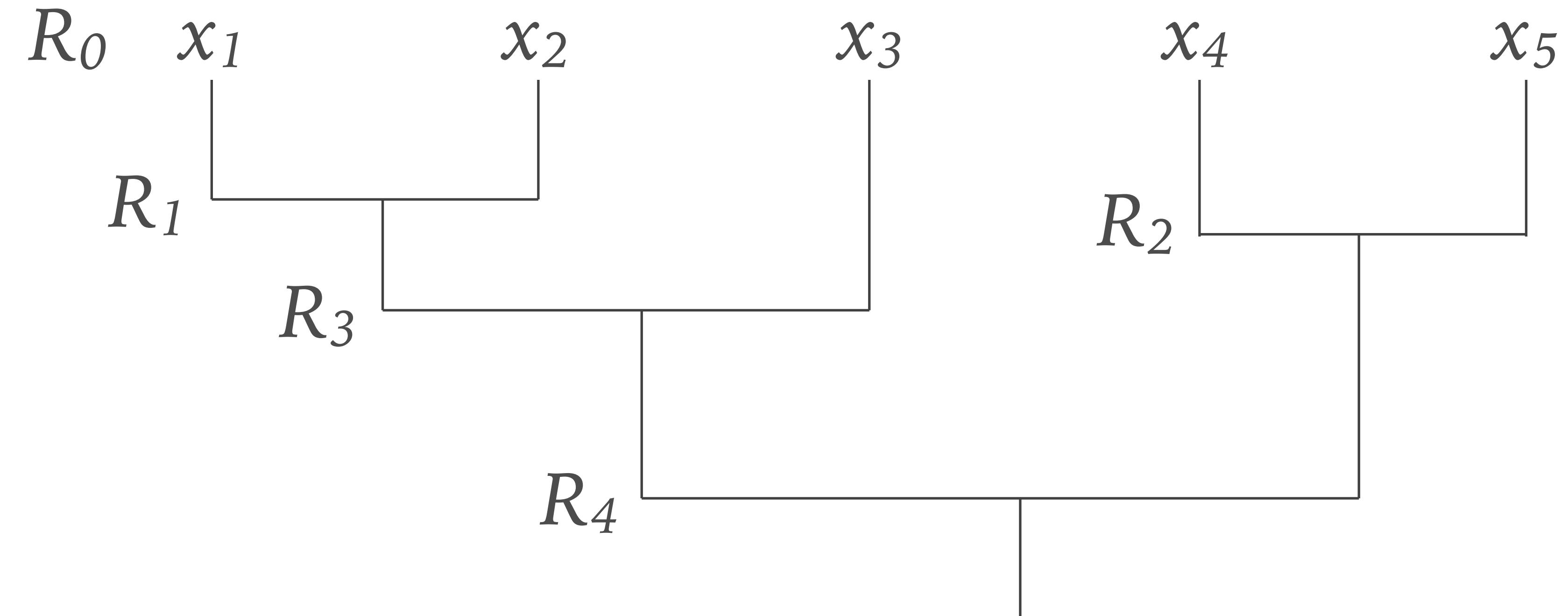
$$C_q = C_i \cup C_j$$

$$R_t = \left(R_{t-1} - \{C_i, C_j\} \right) \cup \{C_q\}$$

- Until we only have one cluster

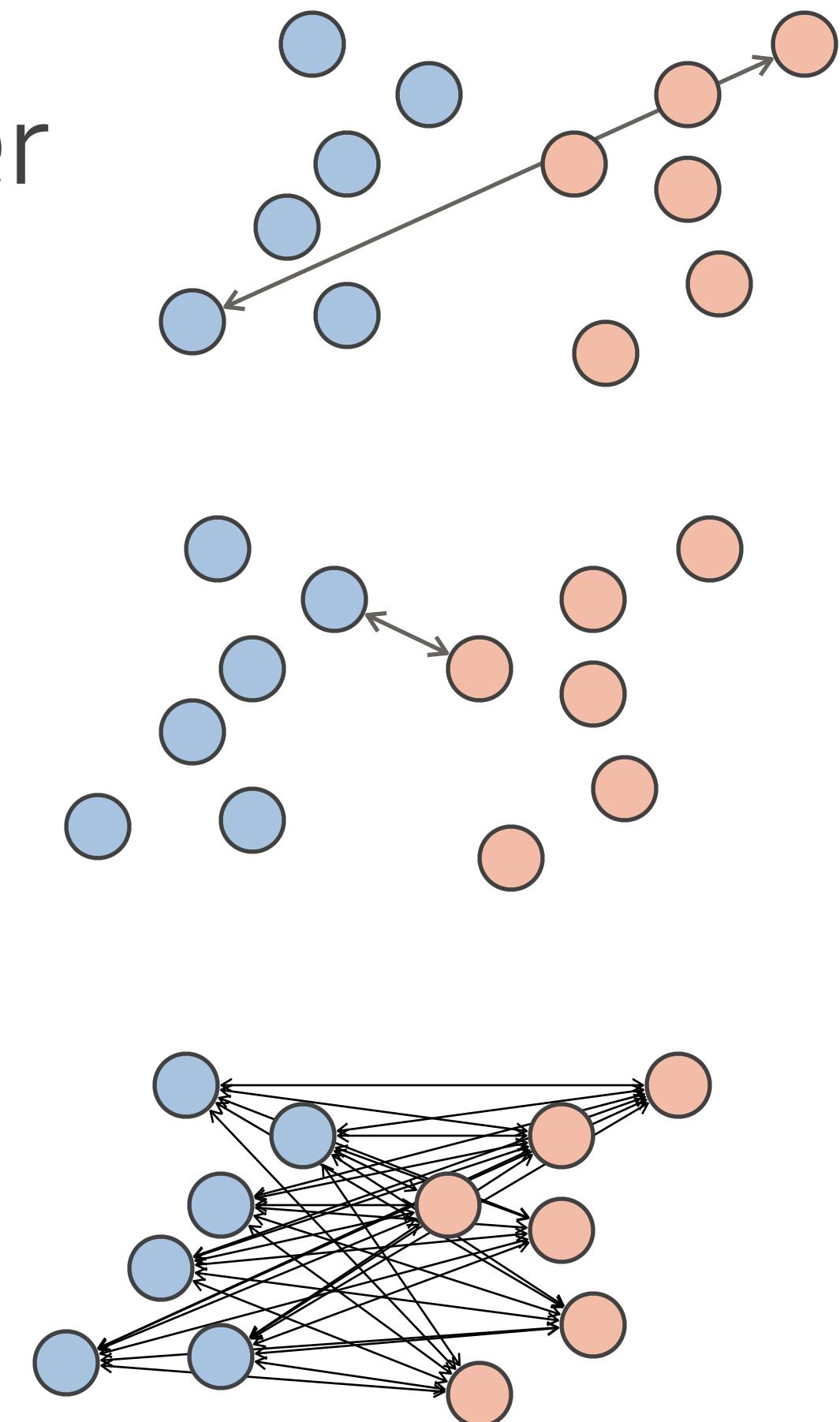
Representing this process

- The Dendrogram



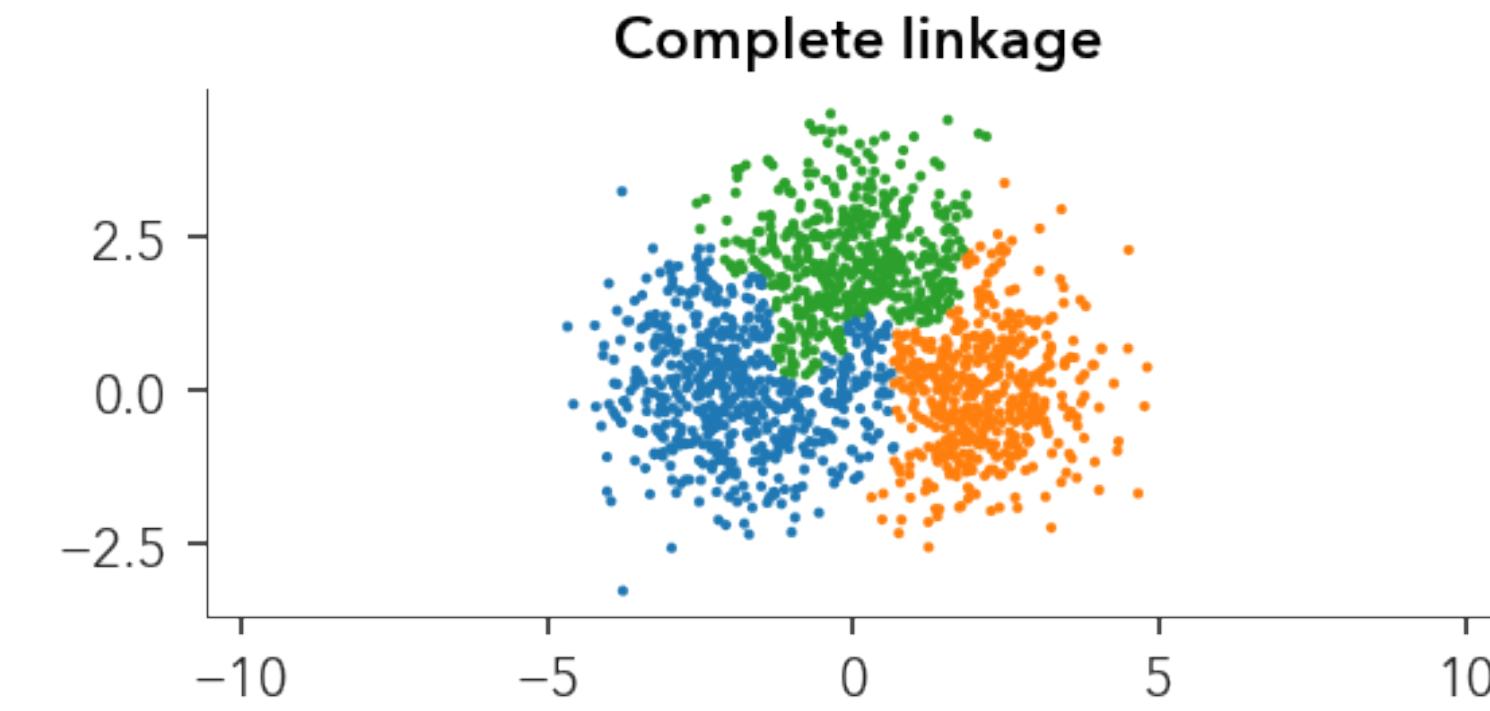
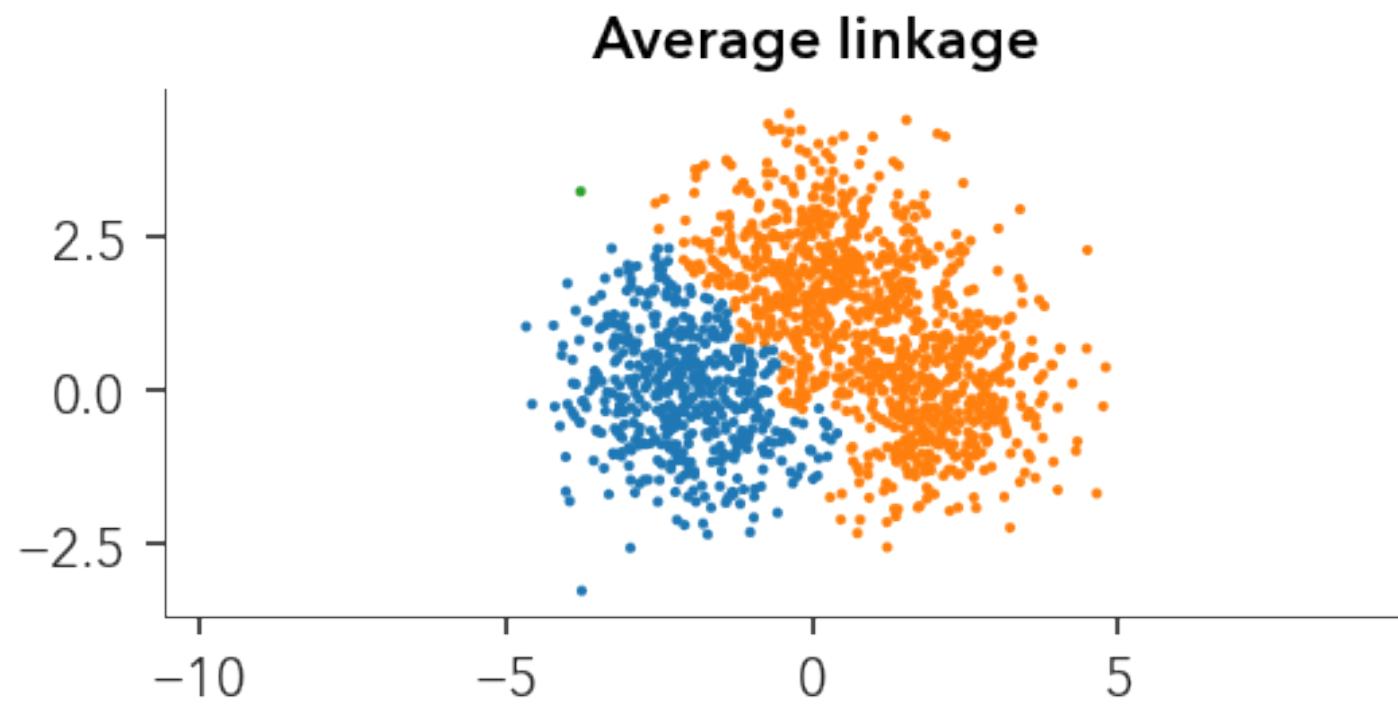
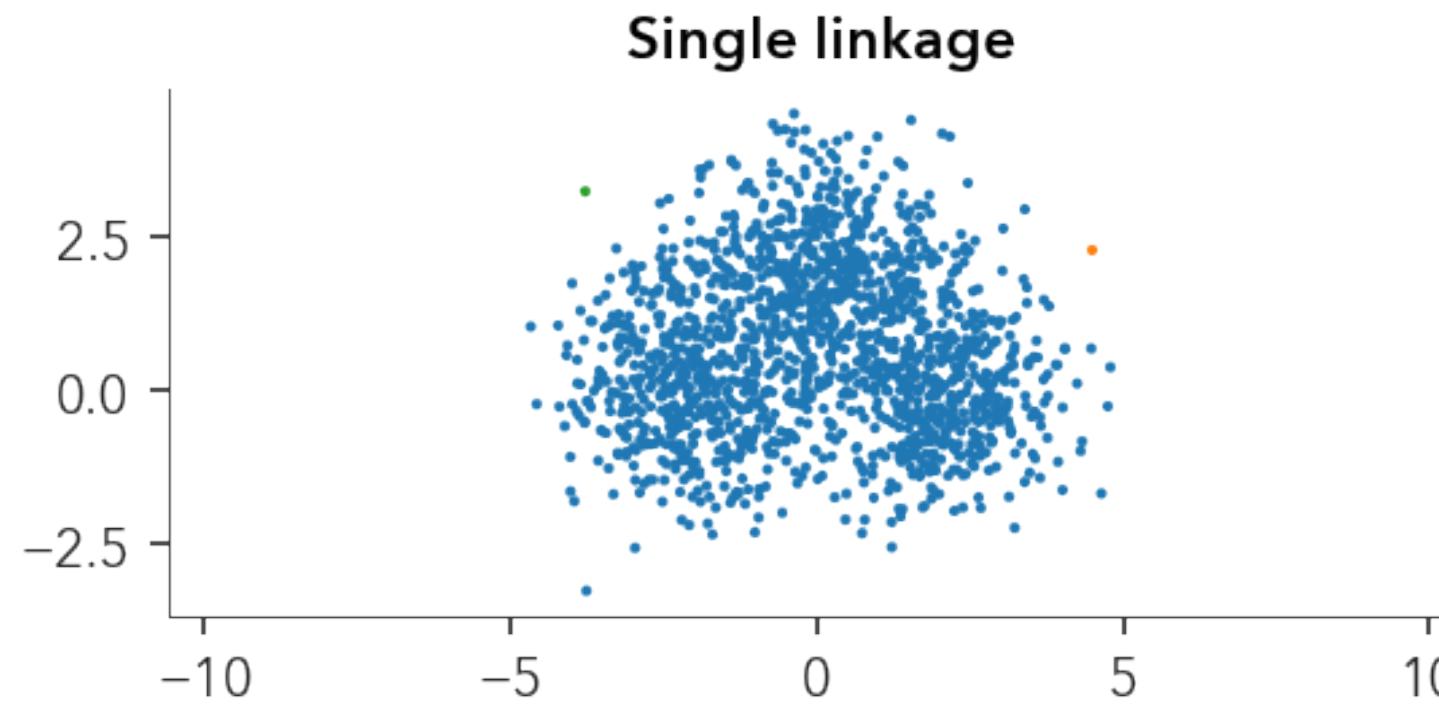
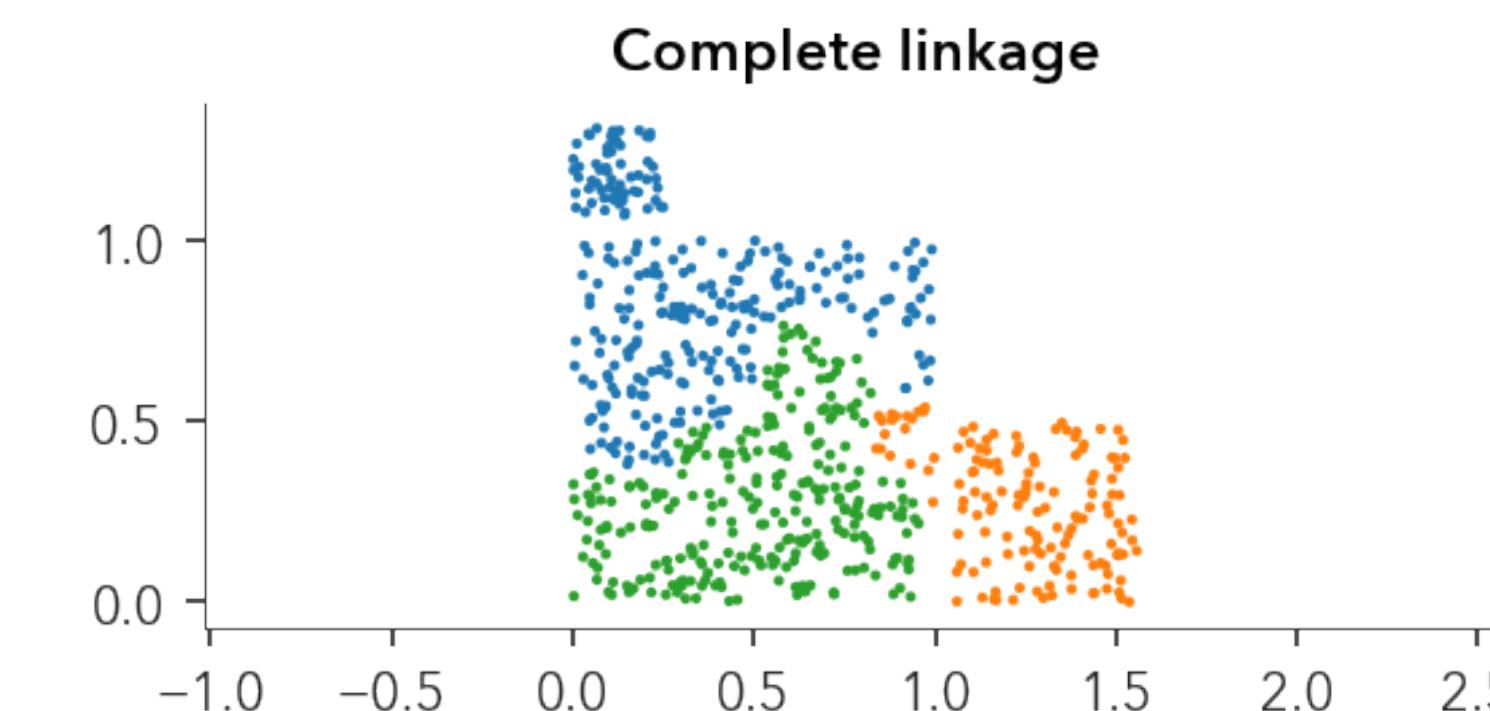
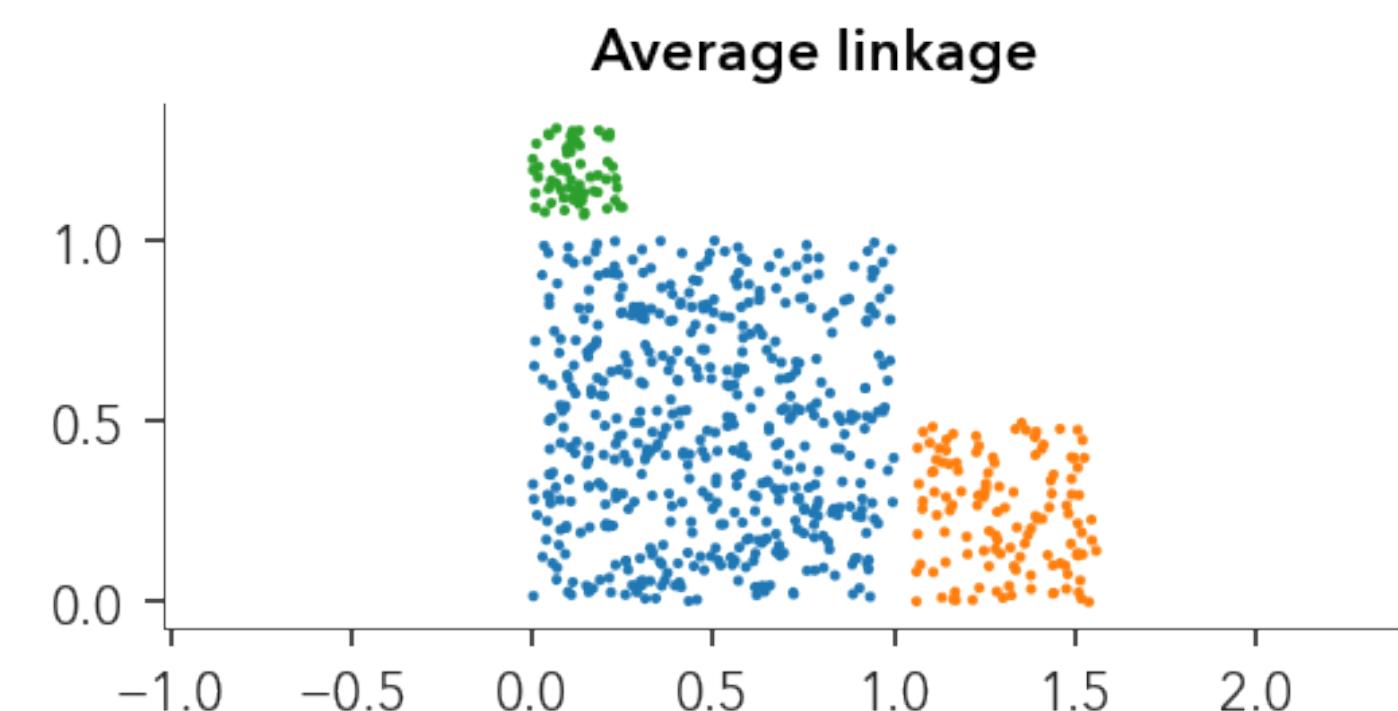
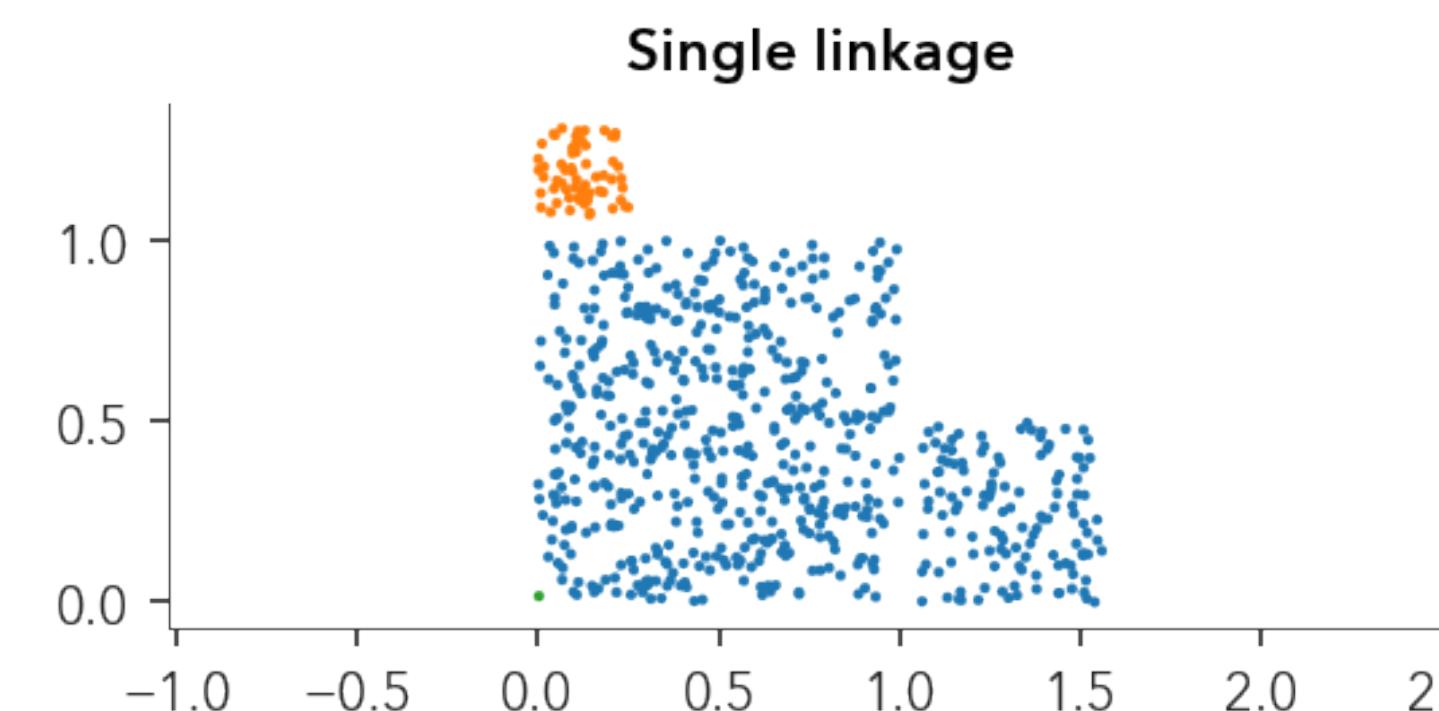
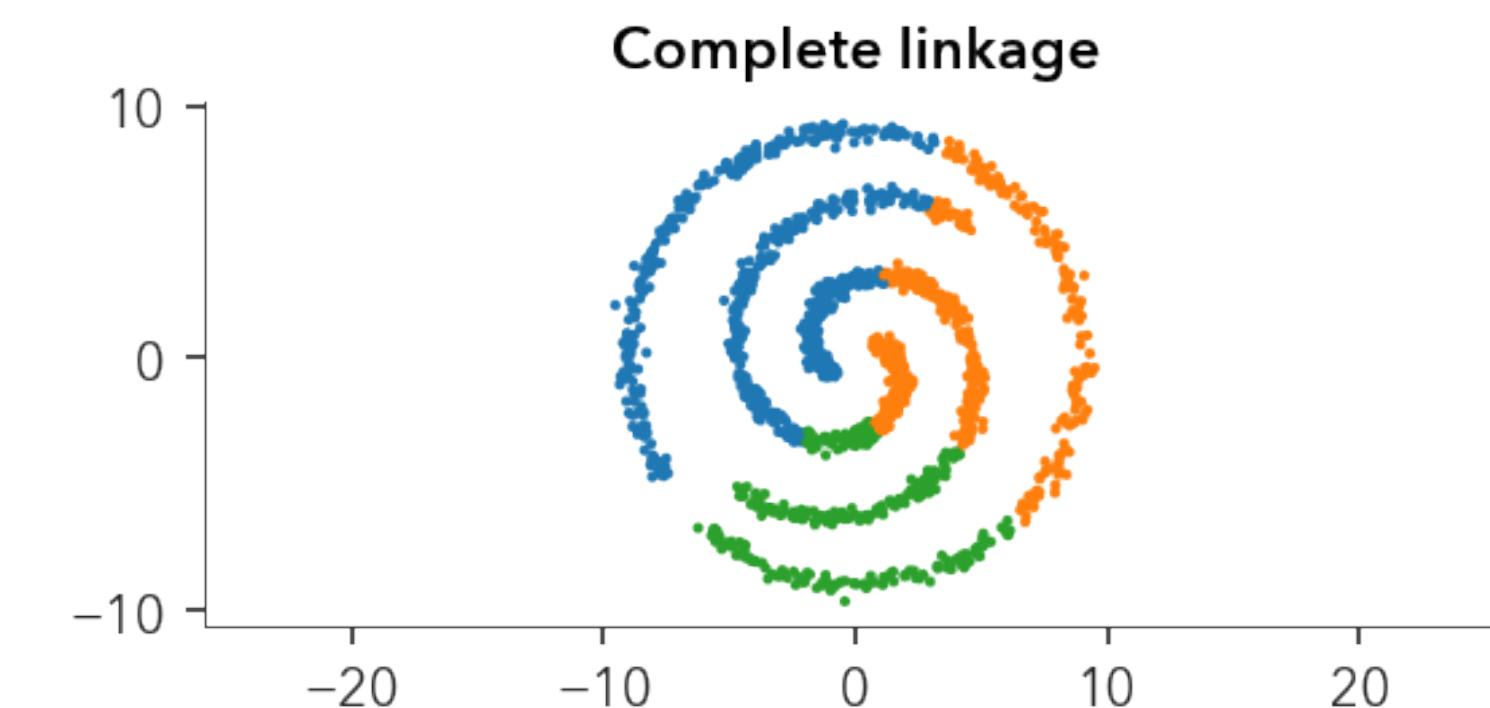
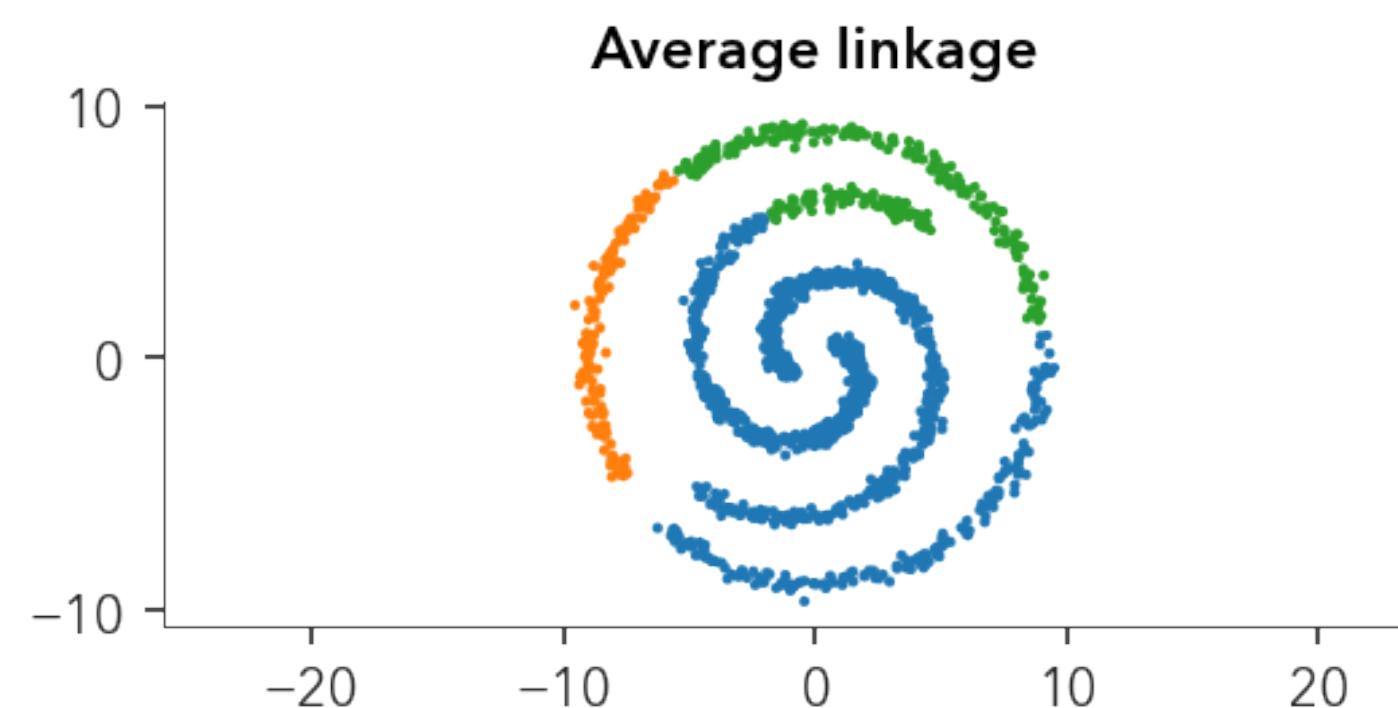
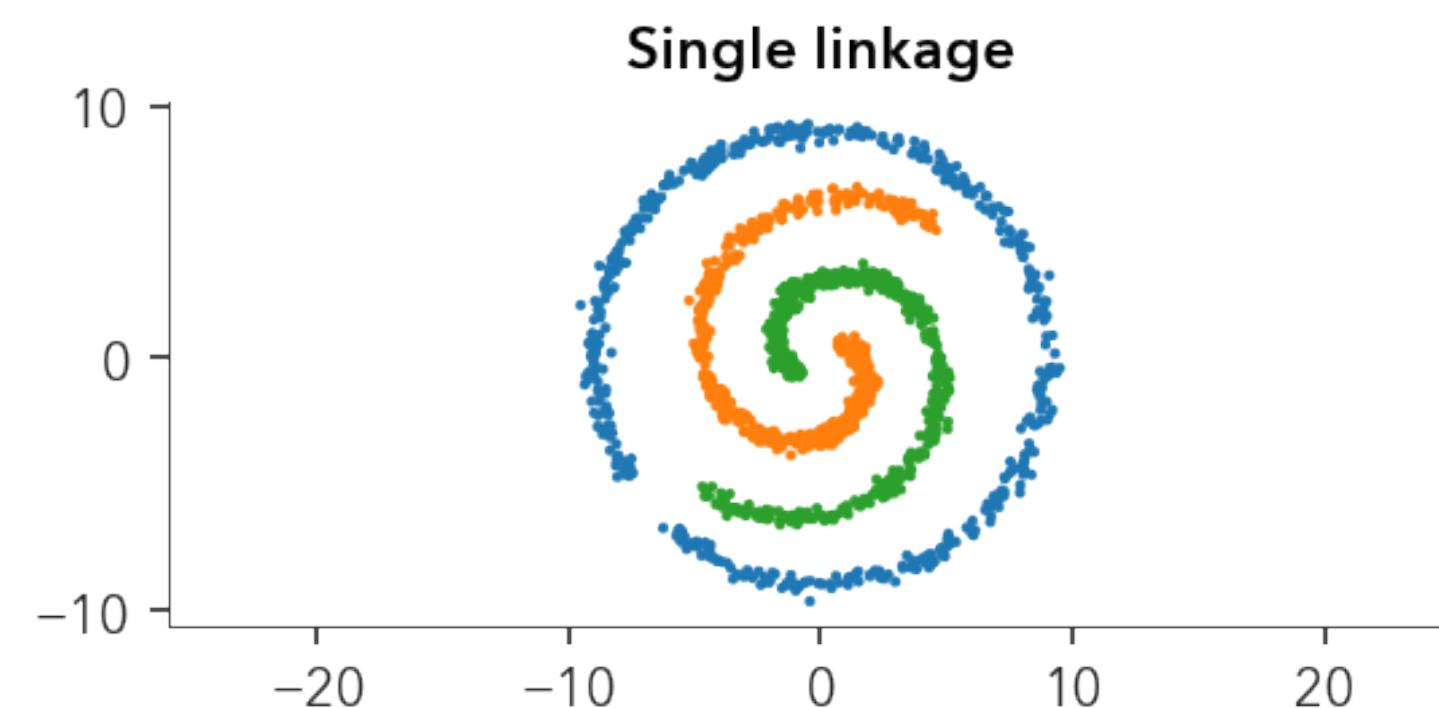
Cluster distance?

- Complete linkage
 - Merge clusters that result in the smallest diameter
- Single linkage
 - Merge clusters with two closest data points
- Group average
 - Use average of distances



And many more other types of linkage ...

Linkage behavior



What's involved

- At level t we have $N - t$ clusters
- At level $t + 1$ the pairs we consider are:

$$\binom{N-t}{2} = \frac{(N-t)(N-t-1)}{2}$$

- Overall comparisons:

$$\sum_{t=0}^{N-1} \binom{N-t}{2} = \frac{(N-1)N(N+1)}{6}$$

Not good for our problem

- El Capitan picture has 63,140 pixels
- How many cluster comparisons is that?

$$\sum_{t=0}^{N-3} \binom{N-t}{2} = 41,946,968,141,536$$

- Thanks, but no thanks ...

Divisive Clustering

- Works the other way around
- Start with all data in one cluster
- Start dividing into sub-clusters

Divisive Clustering

- Choose: $R_0 = \{ X \}$
- For $t = 1, \dots$
 - For $k = 1, \dots, t$
 - Find least similar sub-clusters in each cluster
 - Pick the least similar from that set:
$$\arg \max_{k,i,j} d(C_{k,i}, C_{k,j})$$
 - New clustering is now:
$$R_t = (R_{t-1} - \{C_k\}) \cup \{C_{k,i}, C_{k,j}\}$$
 - Repeat until each point is a cluster

Comparison

- Which one is faster?
 - Agglomerative
 - Divisive has a complicated search step
- Which one gives “better” results?
 - Divisive
 - Agglomerative makes only local observations
 - Seeing the global picture is very important!

Using a global criterion

- Given a set of data x_i
- Define a cost function:

$$J(\theta, \mathbf{U}) = \sum_i \sum_j u_{ij} d(\mathbf{x}_i, \theta_j)$$

- θ are the cluster parameters (e.g. cluster mean)
- $\mathbf{U} \in \{0,1\}^{samples \times clusters}$ is an assignment matrix
 - If $u_{ij} = 1$, then i -th point belongs to j -th cluster
- $d()$ is a distance function

An iterative solution

- We can't use a gradient method
 - The assignment matrix is binary-valued
- We have two parameters to find: θ , \mathbf{U}
 - Fix one and find the other, repeat flip case
 - Iterate until we're content

Overall process

- Randomly initialize θ and iterate:

- Estimate \mathbf{U}

$$u_{i,j} = \begin{cases} 1, & \text{if } d(\mathbf{x}_i, \theta_j) = \min_k d(\mathbf{x}_i, \theta_k) \\ 0, & \text{otherwise} \end{cases}$$

- Estimate θ

$$\theta_j = \arg \min_{\theta_j} \sum_{\forall i \in j} d(\mathbf{x}_i, \theta_j)$$

- Repeat until satisfied

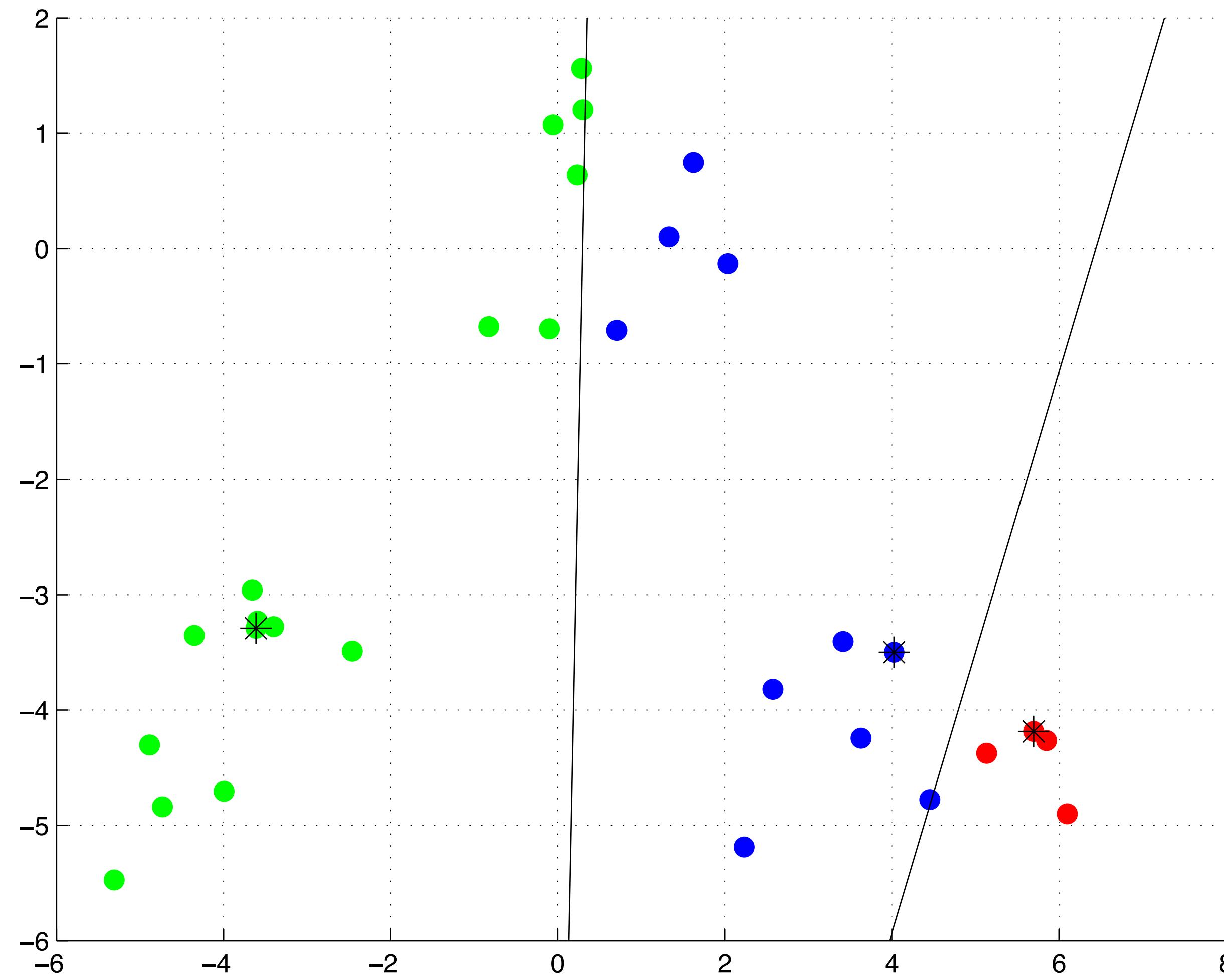
K-means

- Standard and extremely-popular algorithm
- Defines clusters in terms of their means
- Optimizes squared Euclidean distance from cluster means $d(\mathbf{x}, \boldsymbol{\mu}) = \|\mathbf{x} - \boldsymbol{\mu}\|^2$

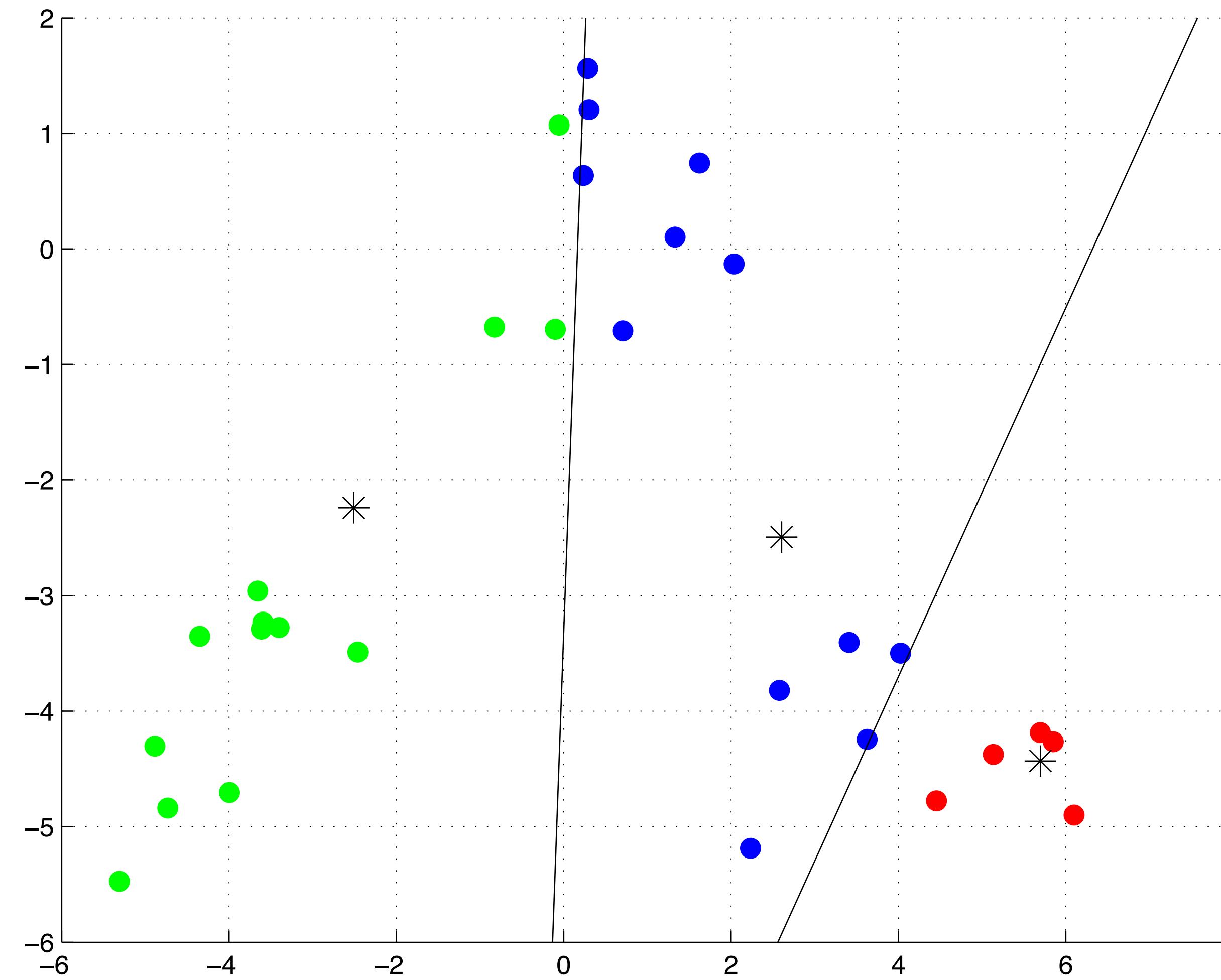
K-means algorithm

- Initialize K cluster means μ_k
- Iterate
 - Assign samples \mathbf{x}_i to closest mean μ_k
 - Estimate new μ_k from assigned samples \mathbf{x}_i
- Repeat until convergence

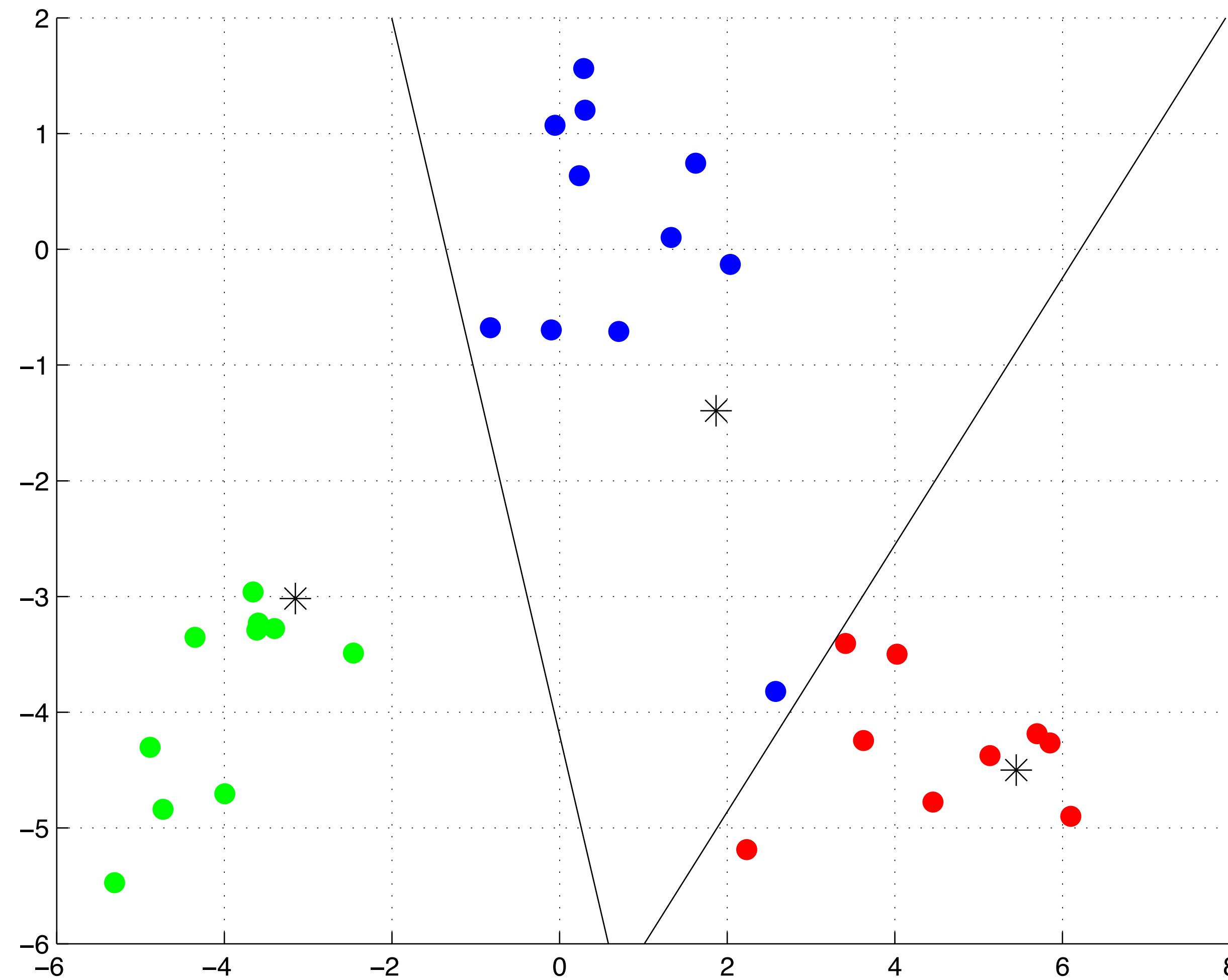
Example run – step 1



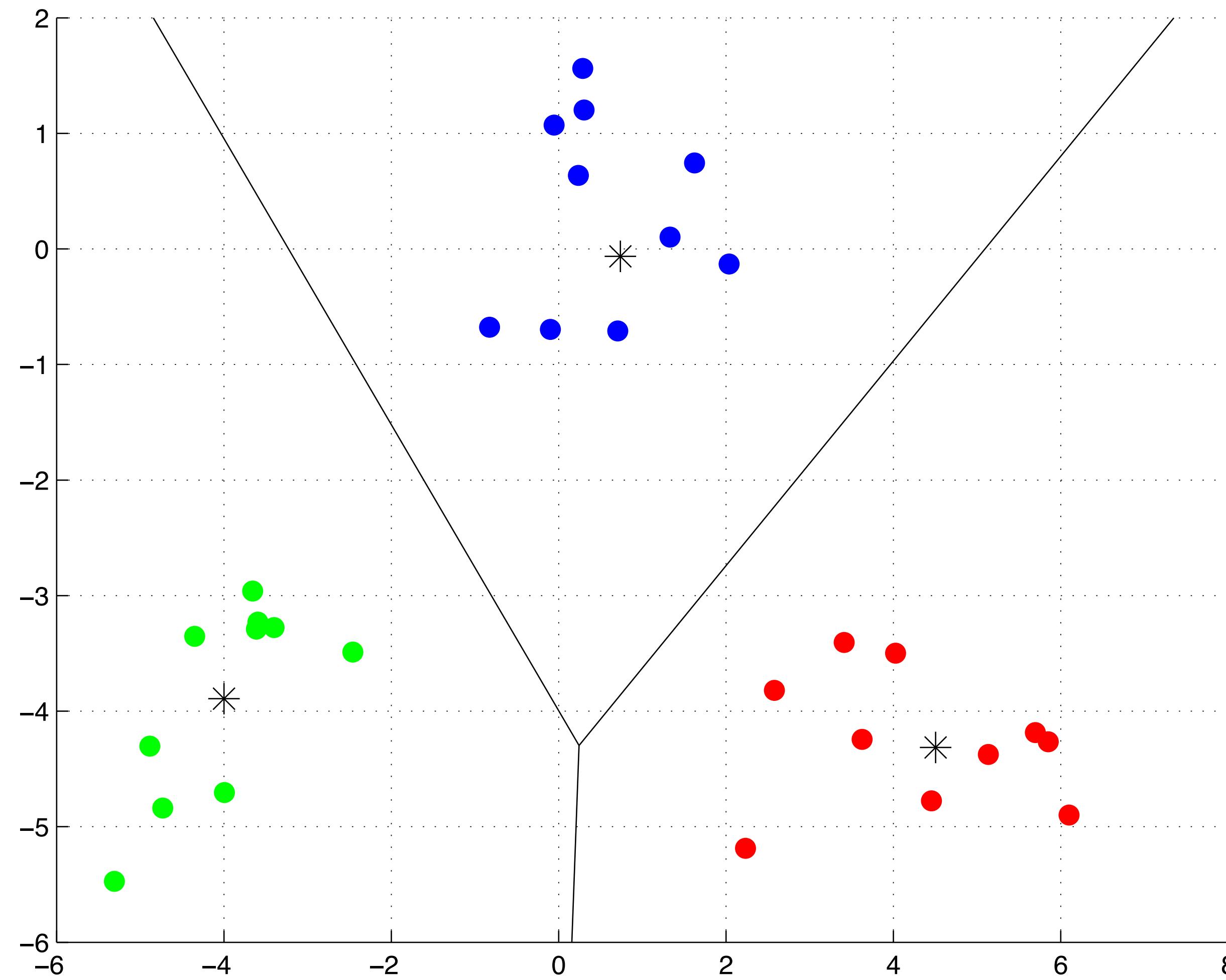
Example run – step 2



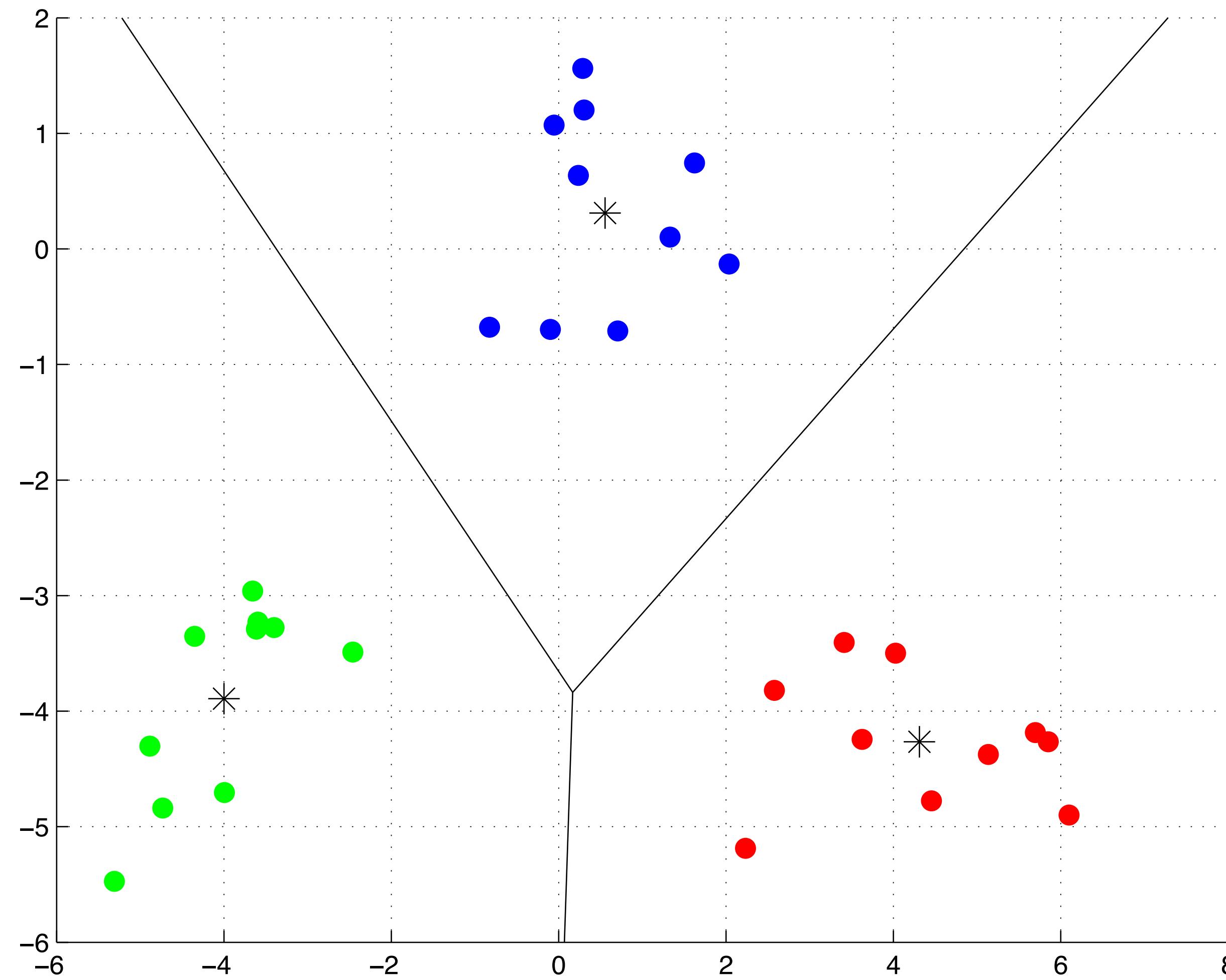
Example run – step 3



Example run – step 4



Example run – step 5

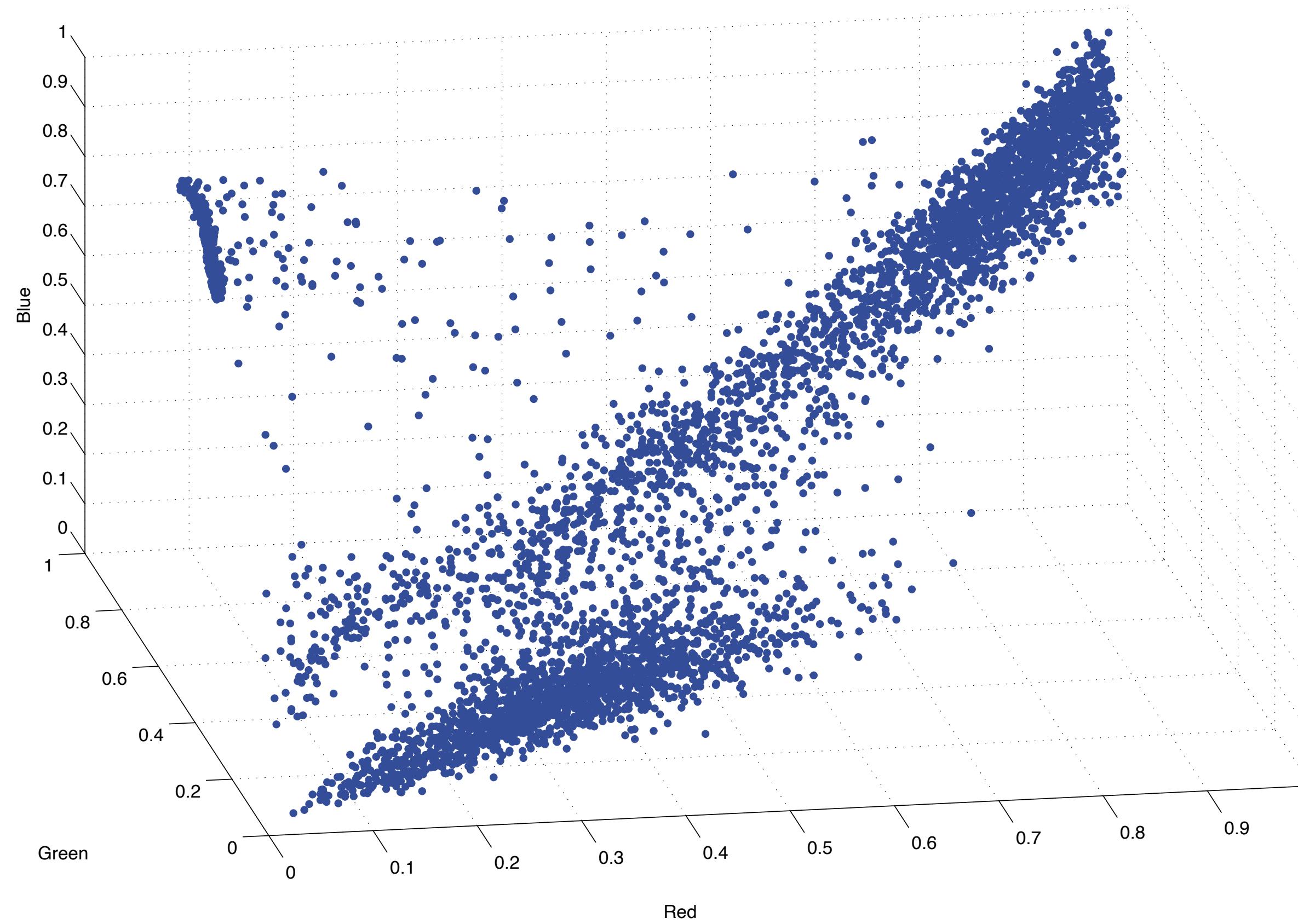


How well does it work?

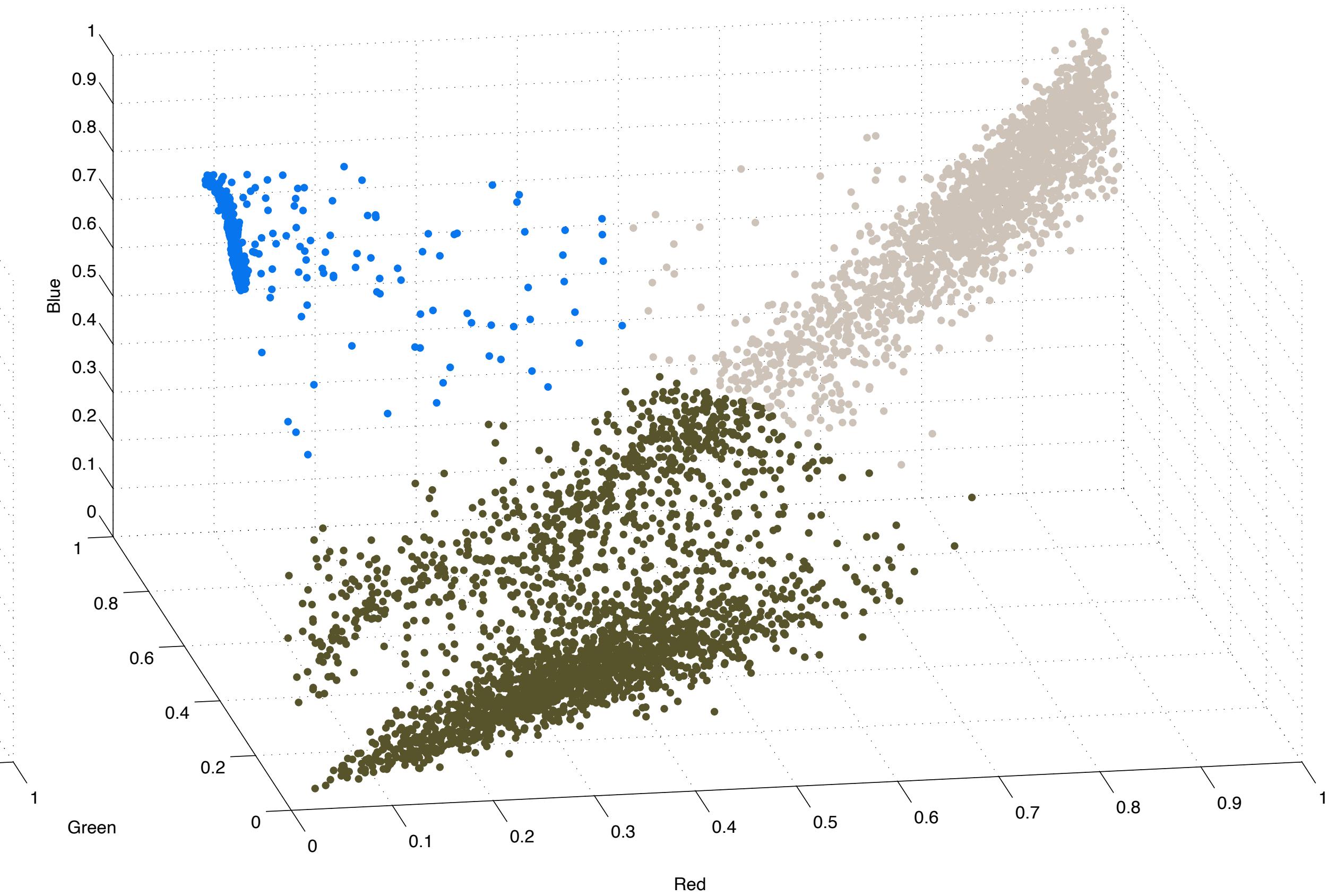
- Converges to a minimum of cost function
 - Not for all types of distances though!
- Is heavily biased by starting positions
 - Many tricks to alleviate that
- Sensitive to outliers

K-means on El Capitan

Input data



Clustered data



K-means on El Capitan

Input data

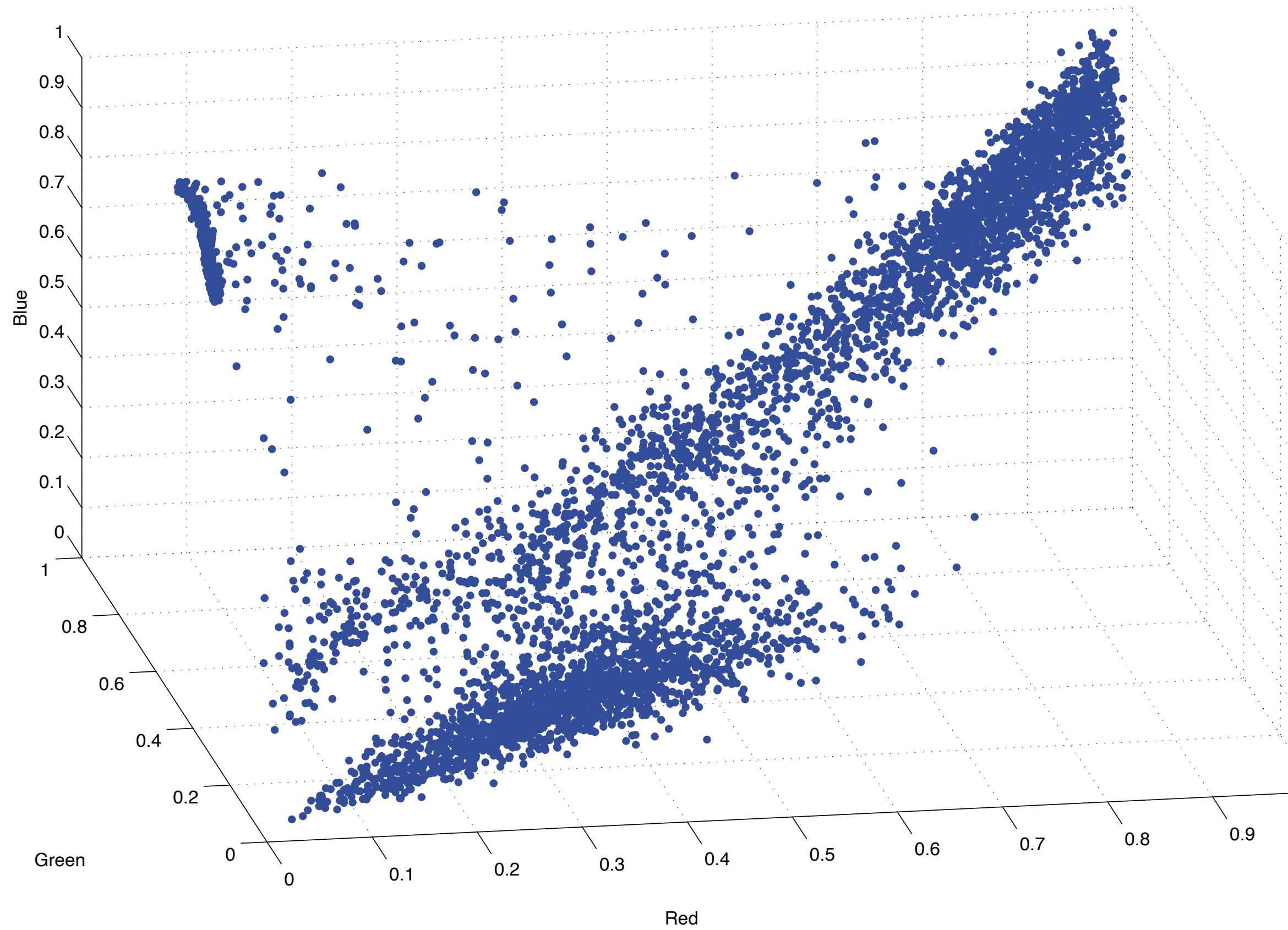


Pixels colored according to their cluster's mean

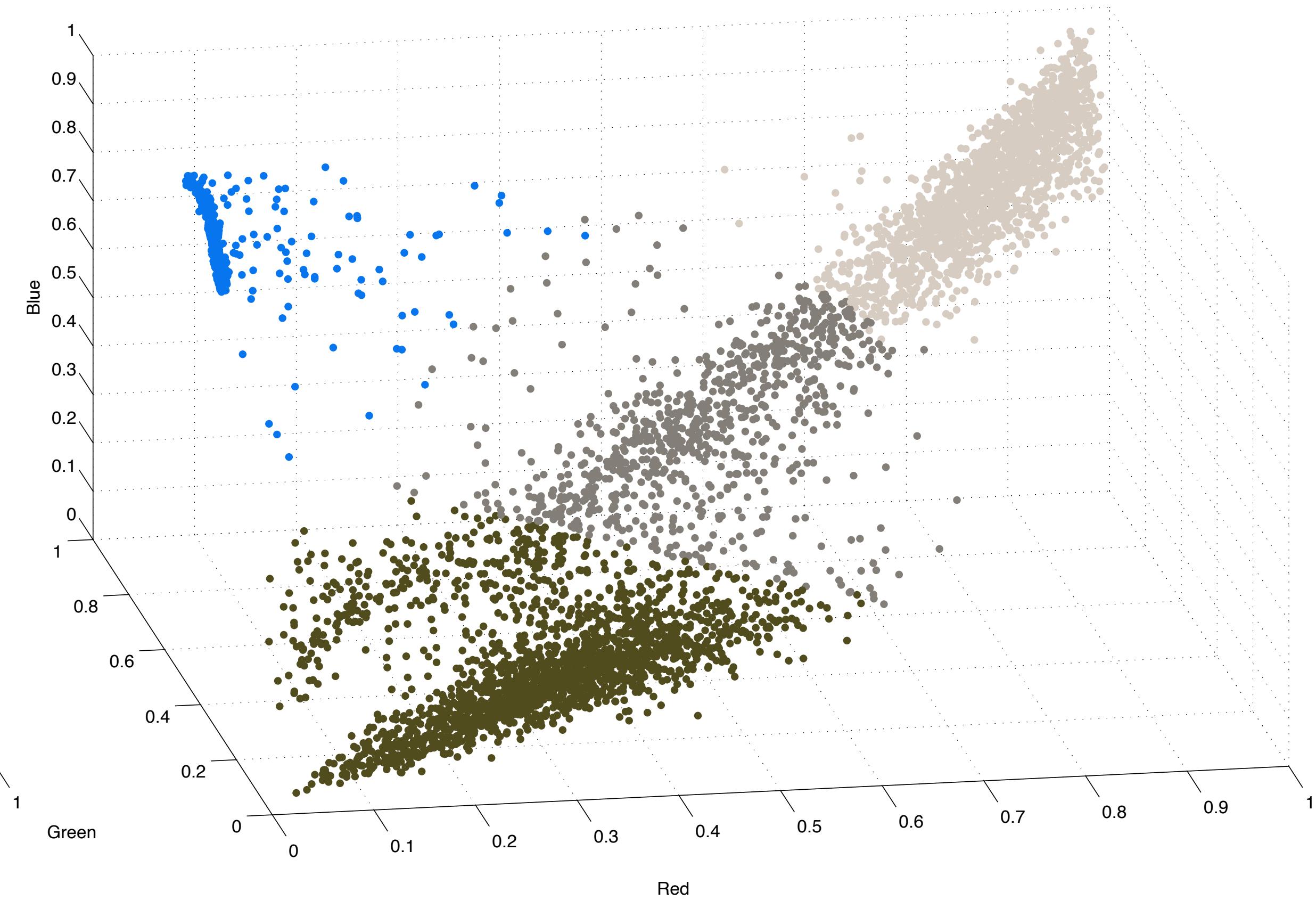


K-means on El Capitan

Input data



Clustered data



K-means on El Capitan

Input data

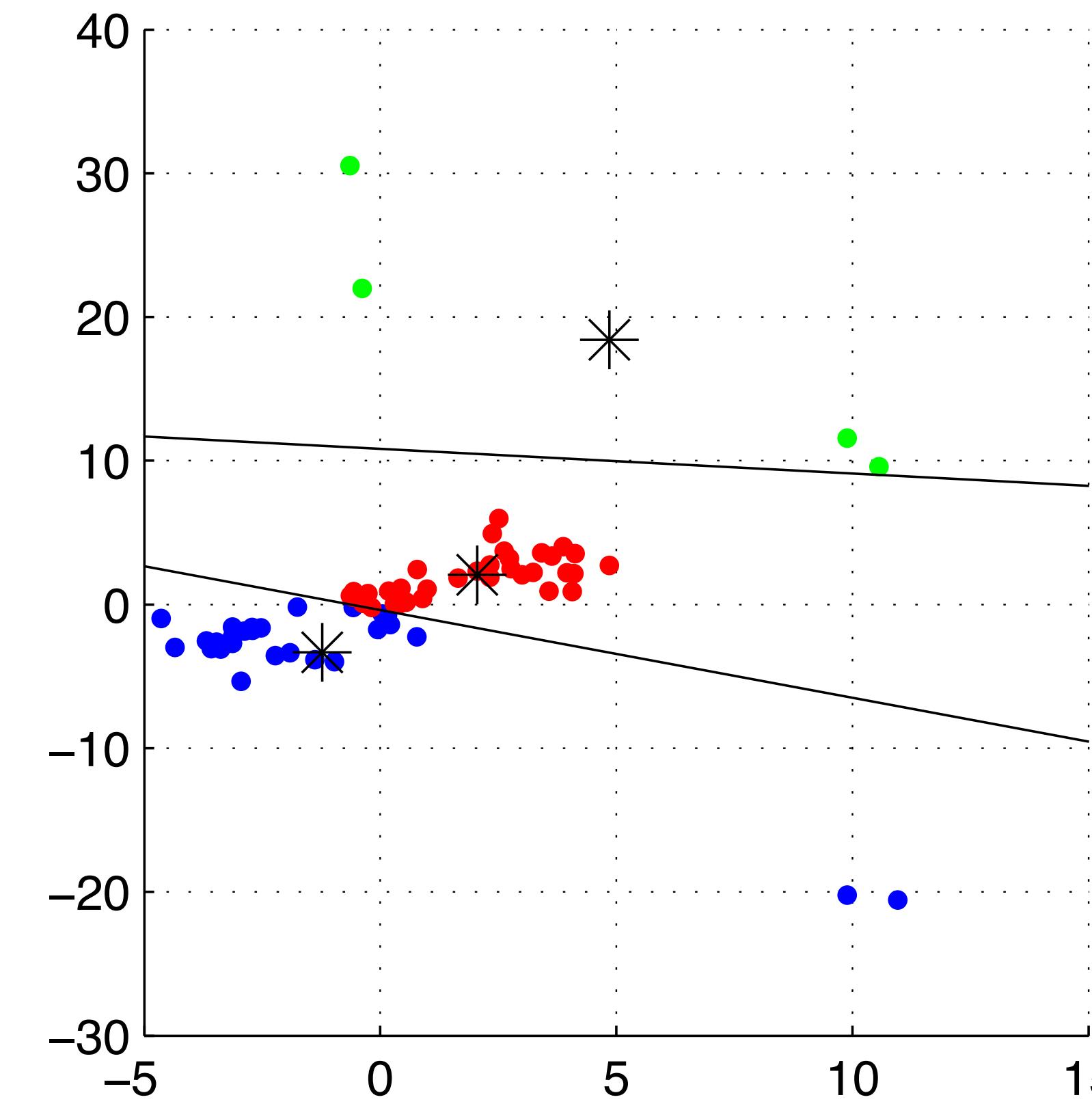
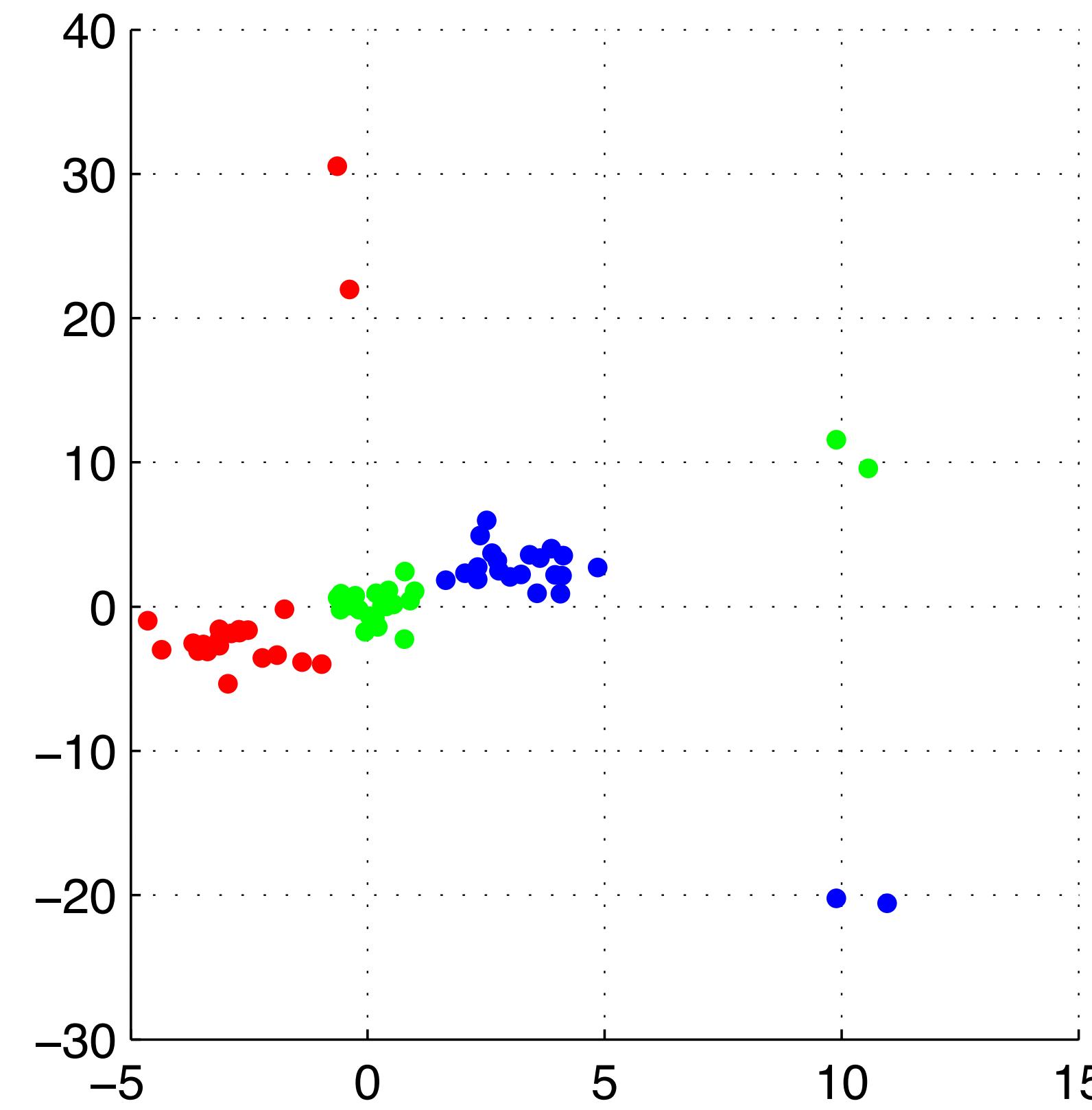


Pixels colored according to their cluster's mean



One problem with k-means

- K-means struggles with outliers



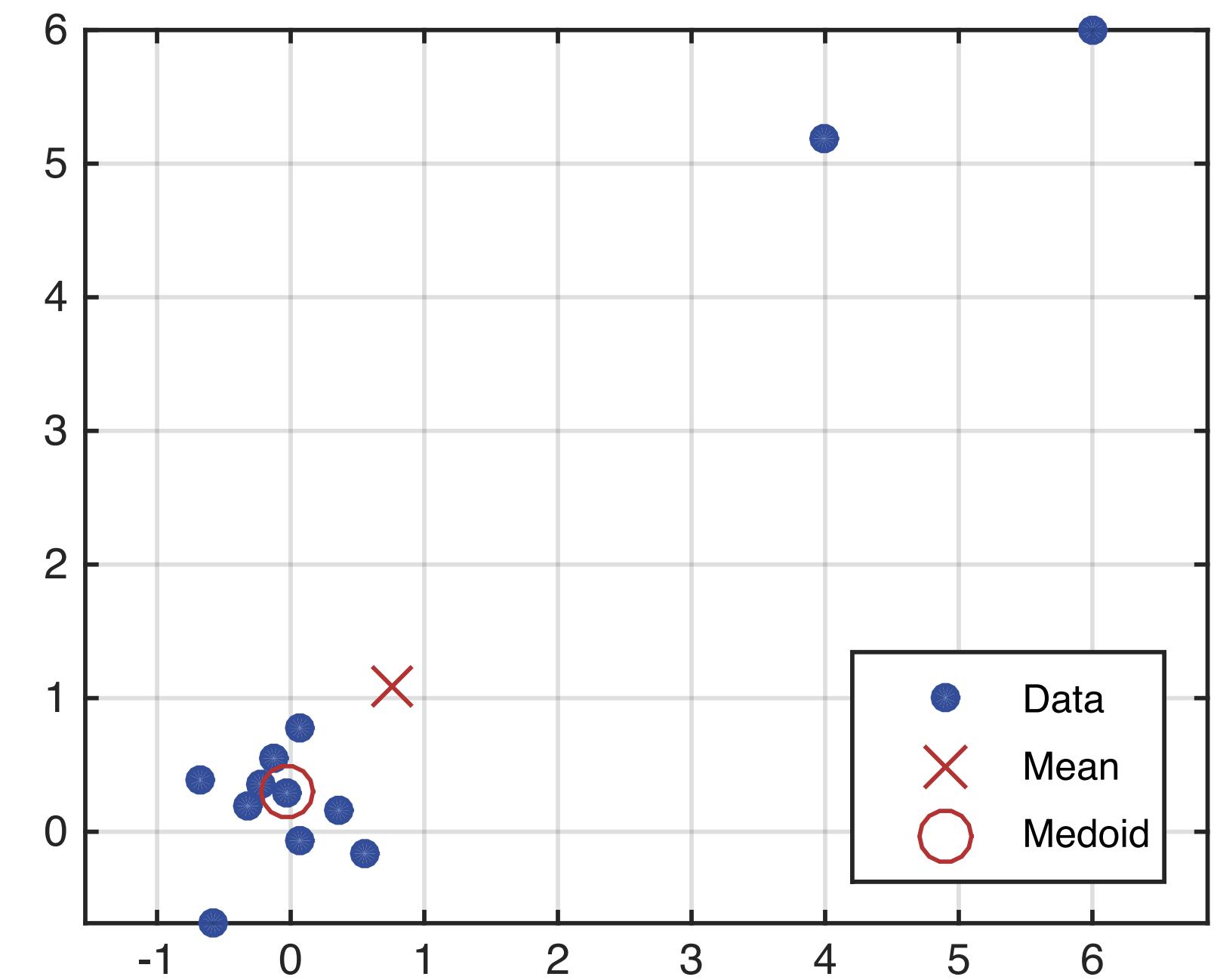
K-medoids

- **Medoid:**

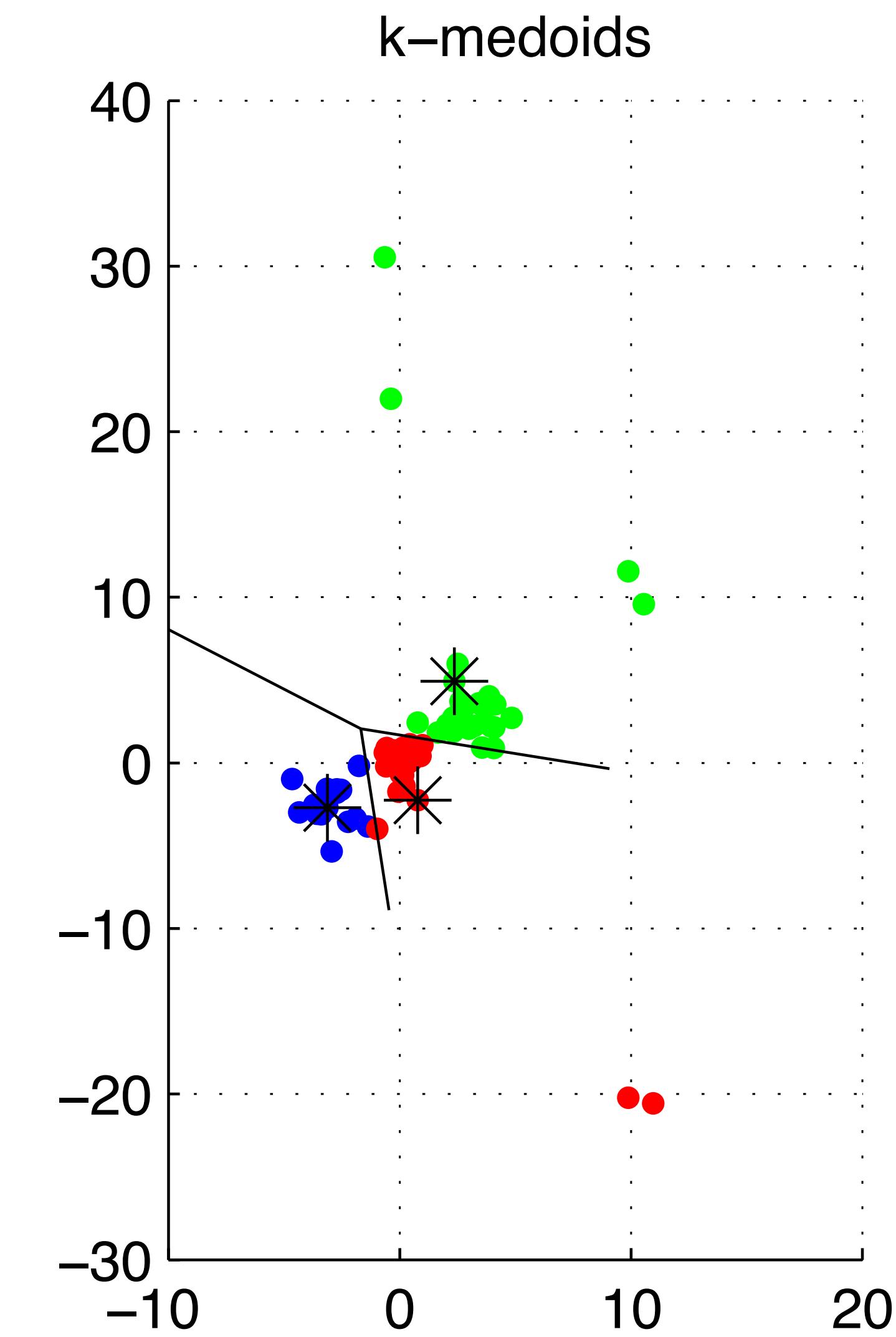
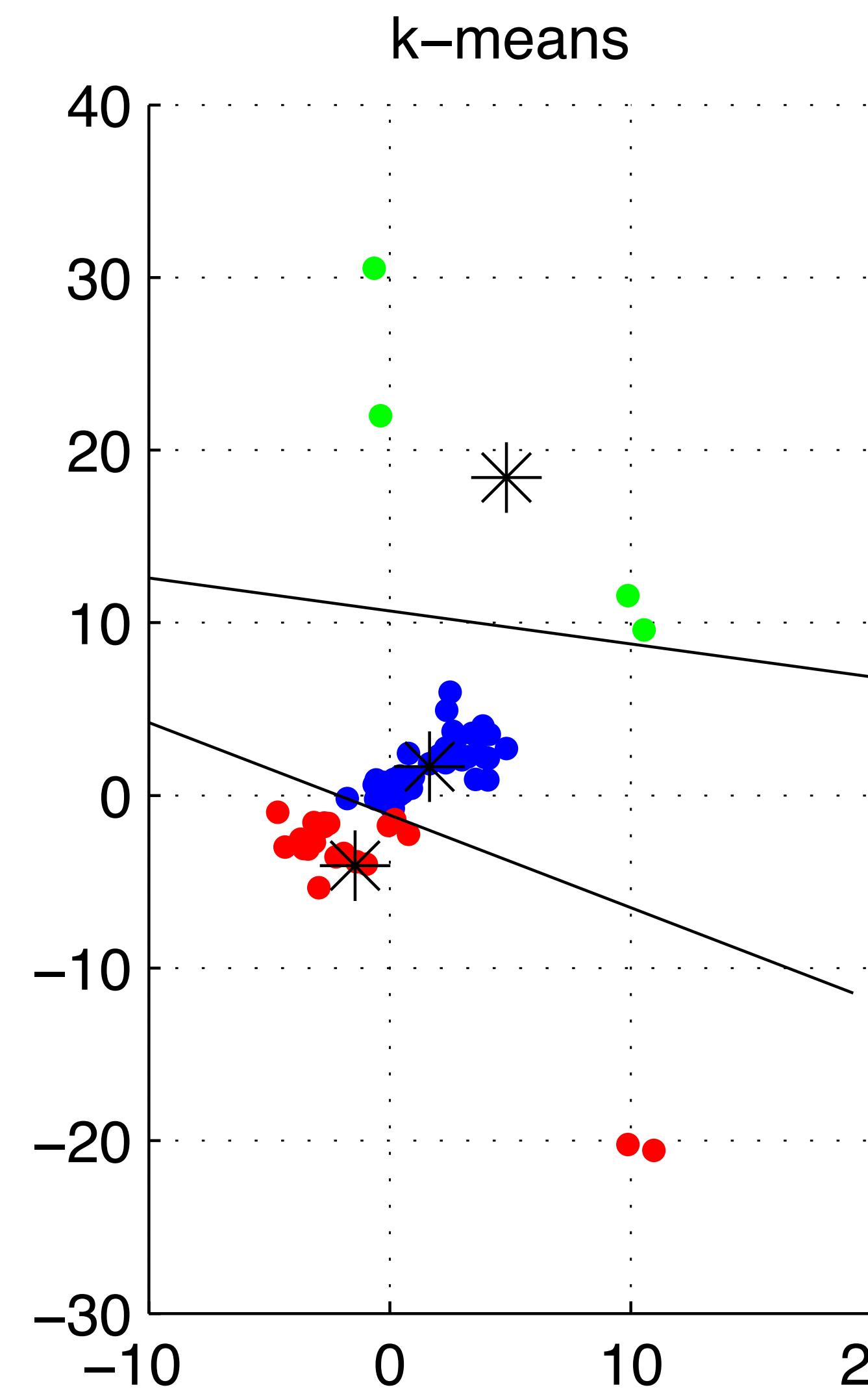
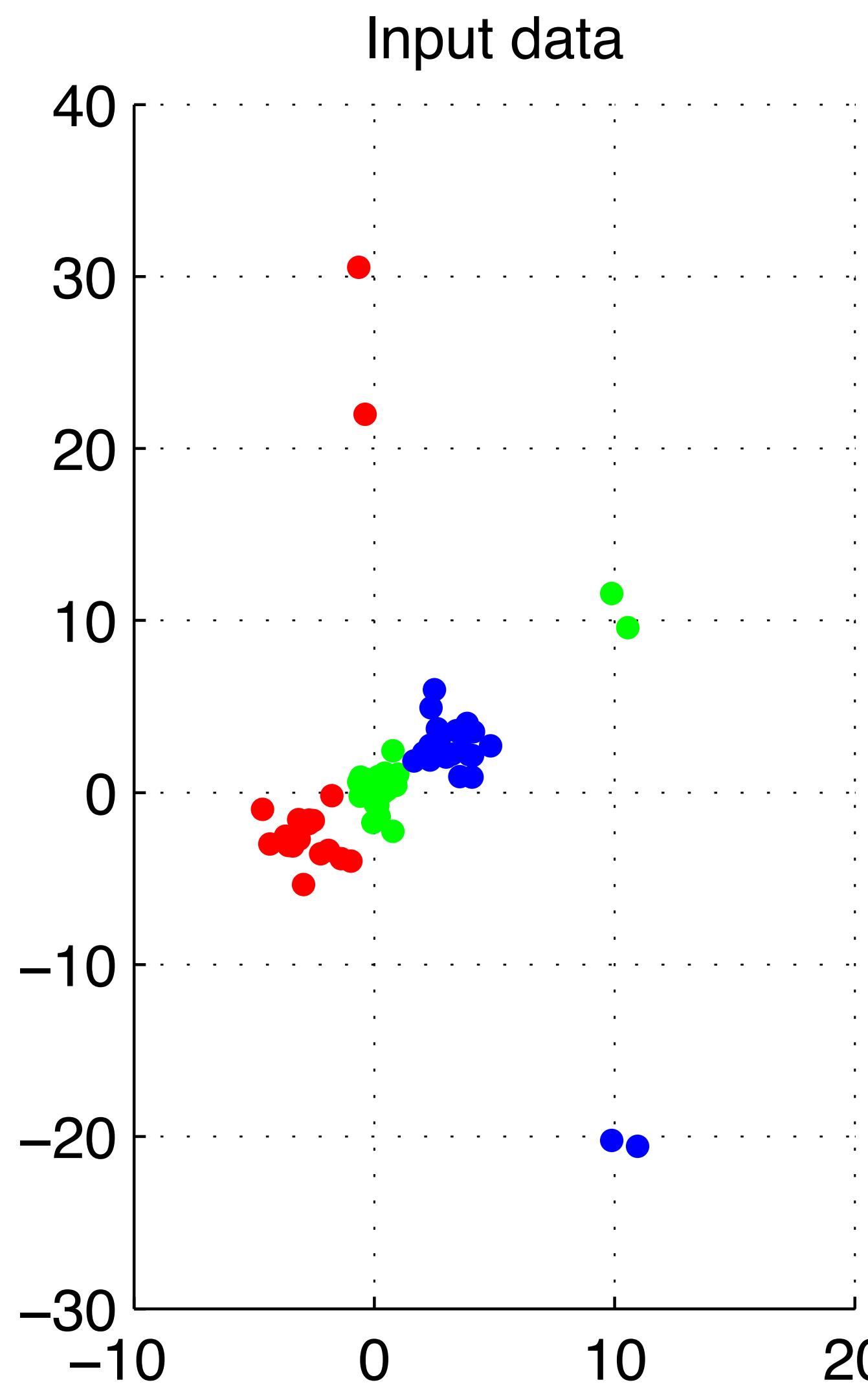
- Least dissimilar data point to all others

$$\mathbf{x}_{medoid} = \arg \min_{y \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}} \sum_{i=1}^N d(y, \mathbf{x}_i)$$

- Not as influenced by outliers as the mean
- Replace cluster means with medoids
 - Reformulate k-means as *k-medoids*

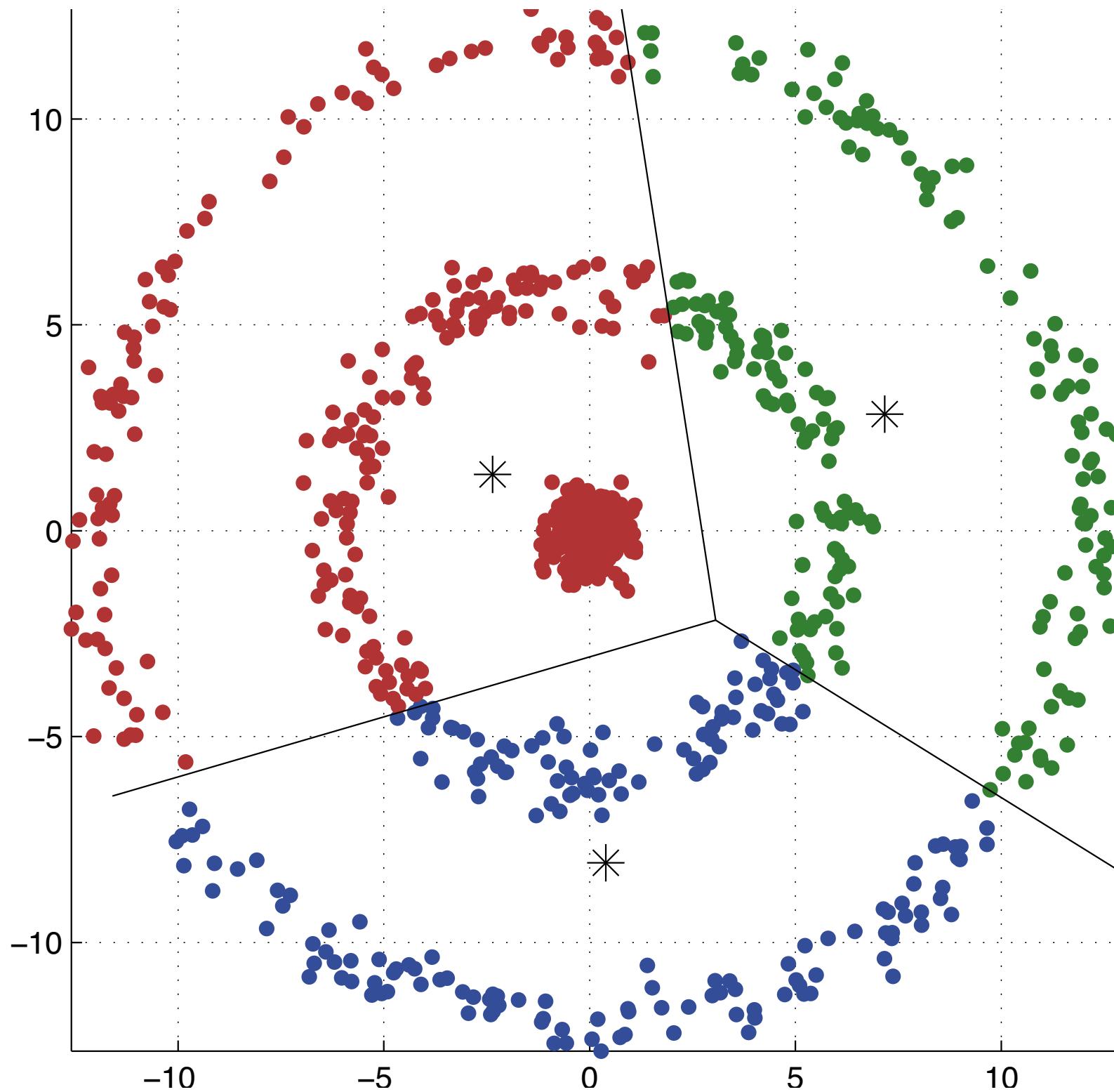


K-medoids



One more problem

- K-means finds compact clusters
 - And screws up cases like this one



Solution: Go to another space

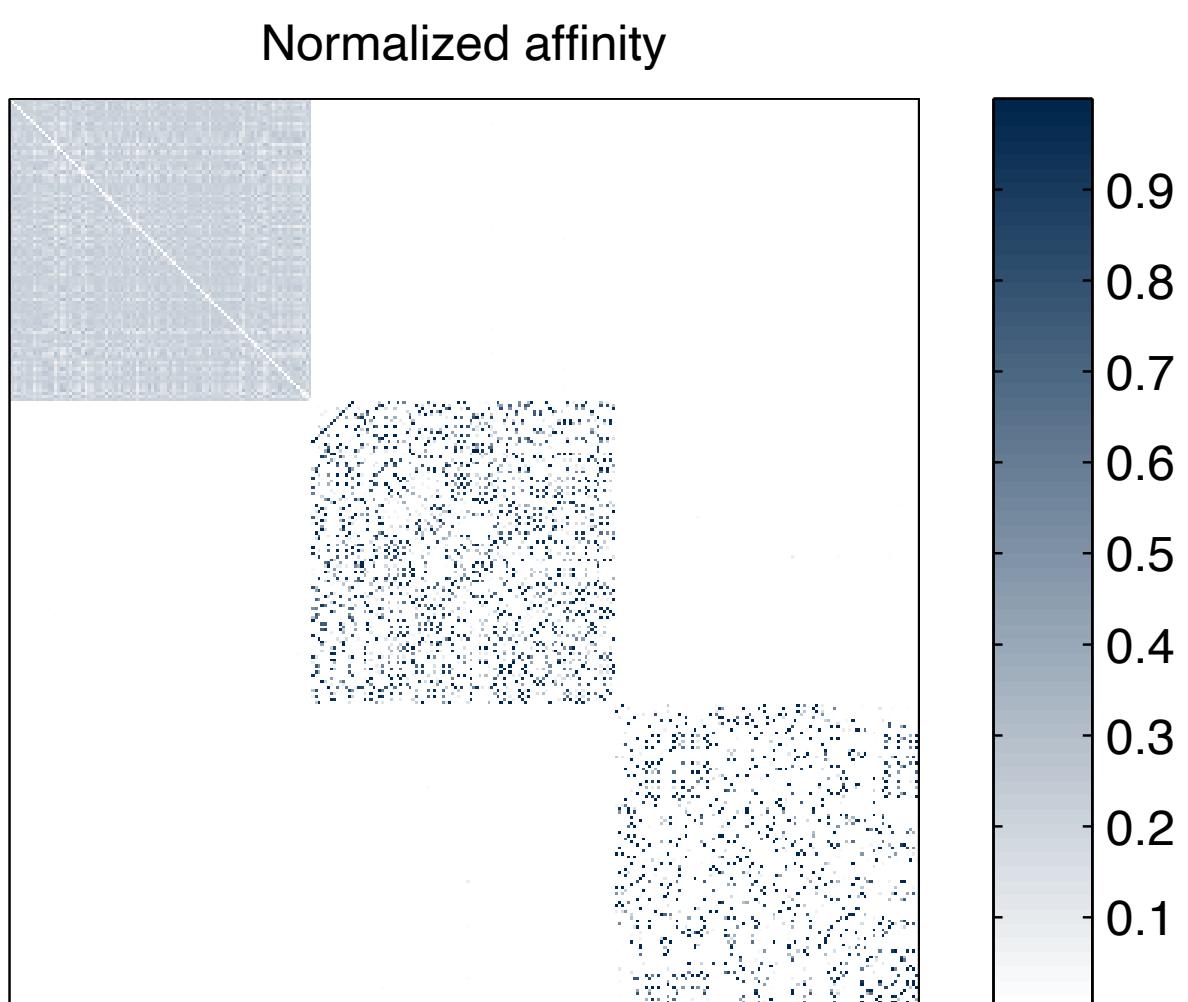
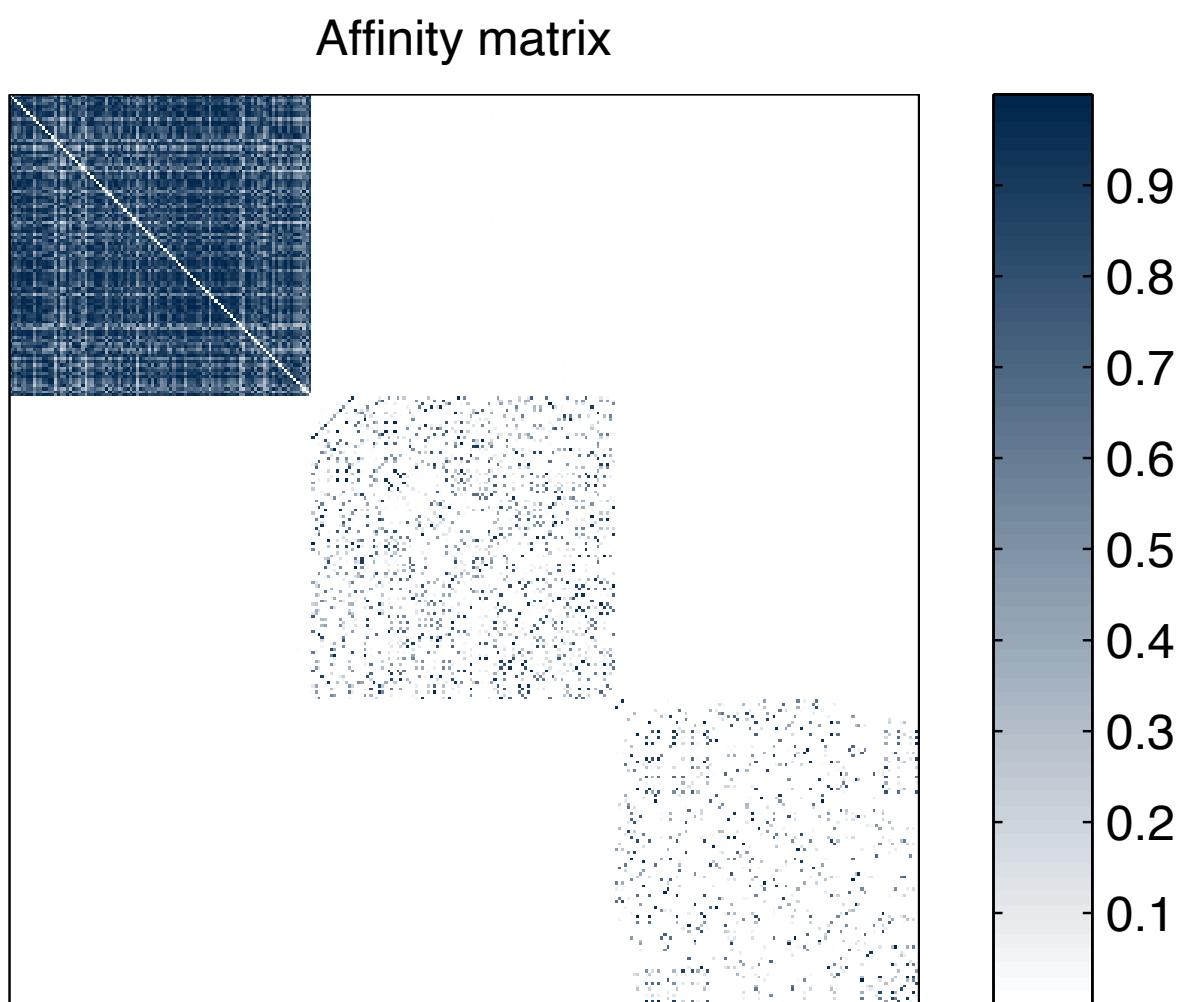
- E.g., *spectral clustering* approach
 - Define affinity matrix:

$$A_{i,j} = \begin{cases} e^{-\|x_i - x_j\|^2/\sigma}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}$$

- And “normalize” it:

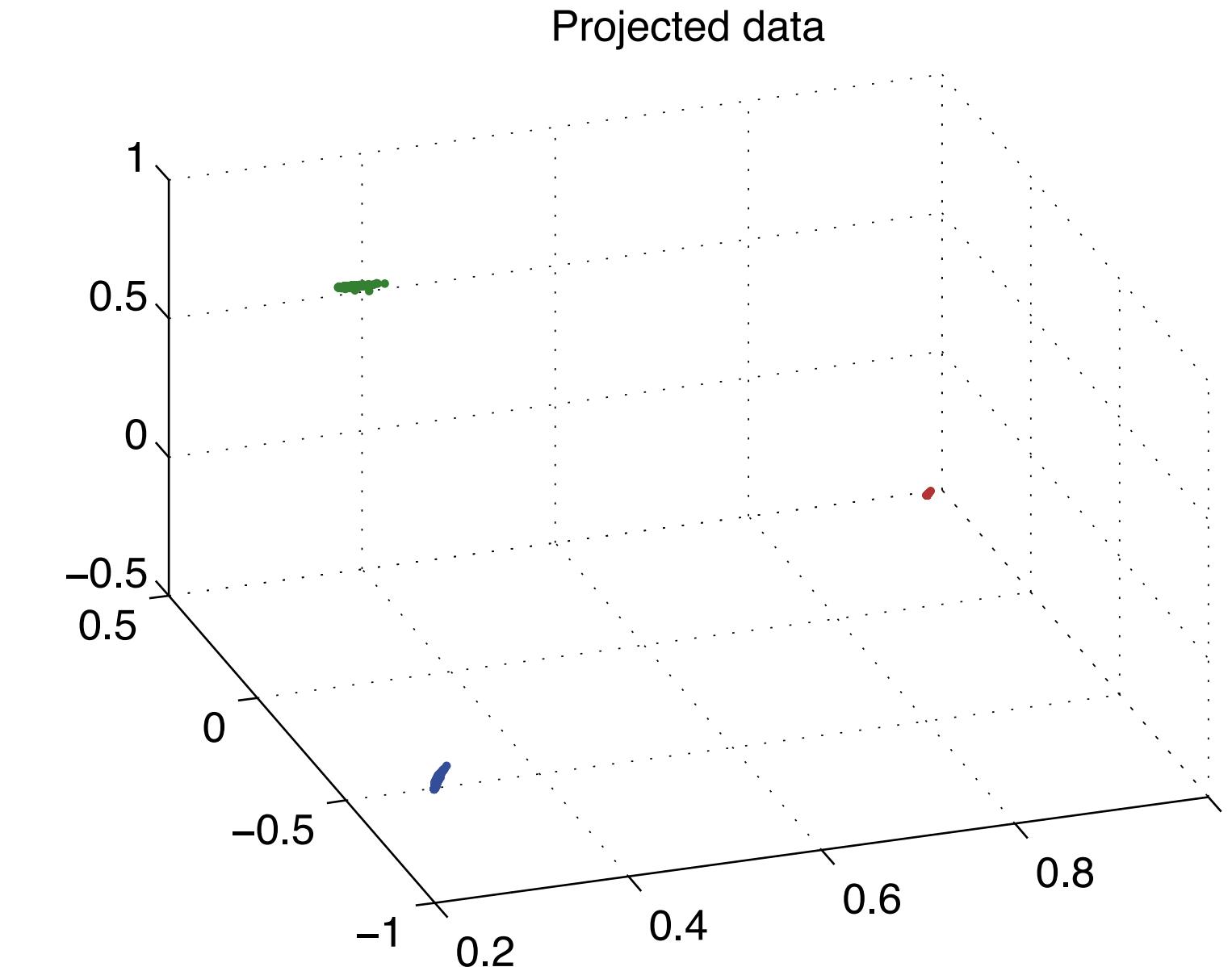
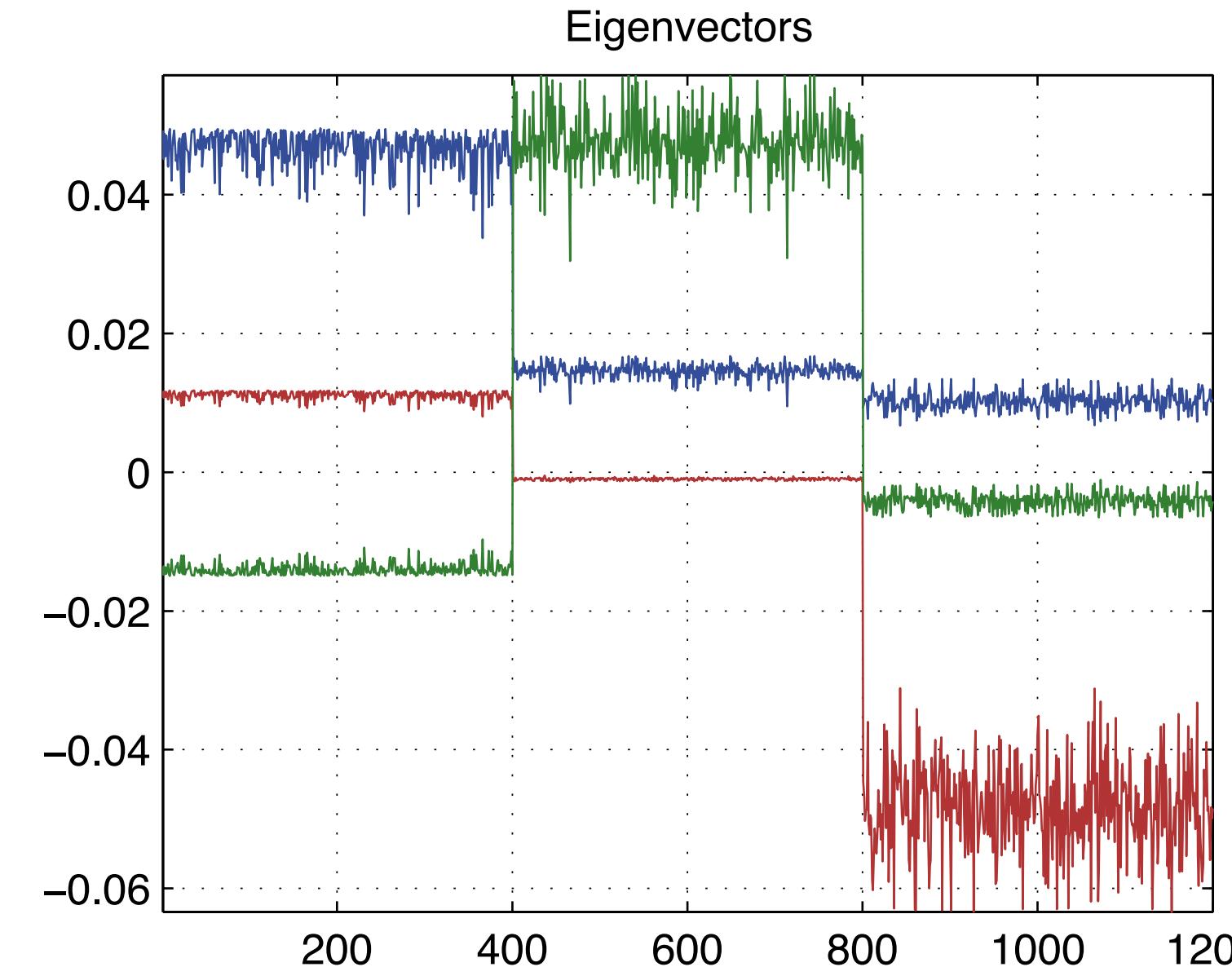
$$D_{i,j} = \sum_j A_{i,j}$$

$$L = D^{-1/2} \cdot A \cdot D^{-1/2}$$



Project and k-means it

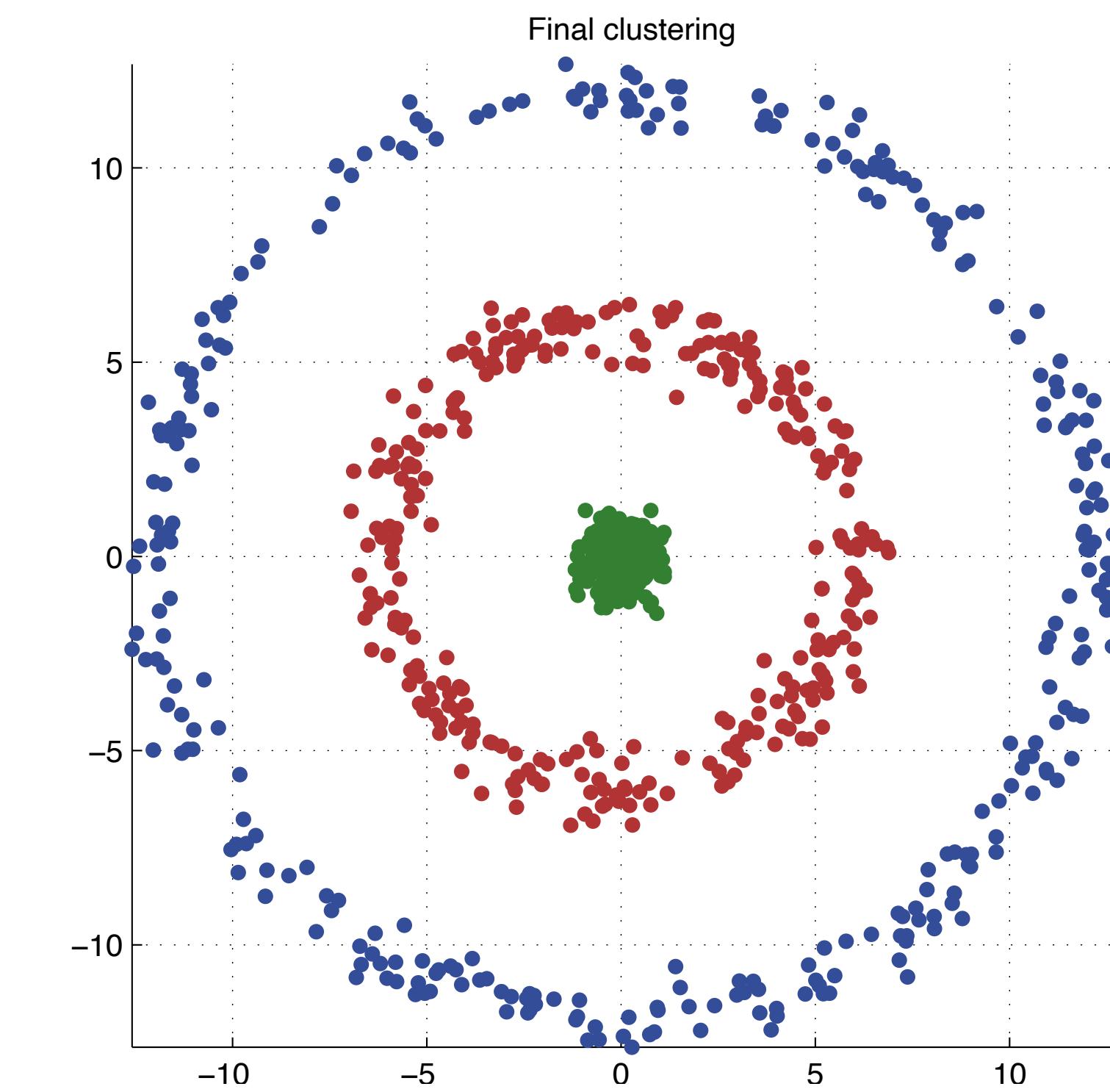
- Compute its K largest eigenvectors
 - Similar to performing dimensionality reduction or MDS



- And perform k-means on the result

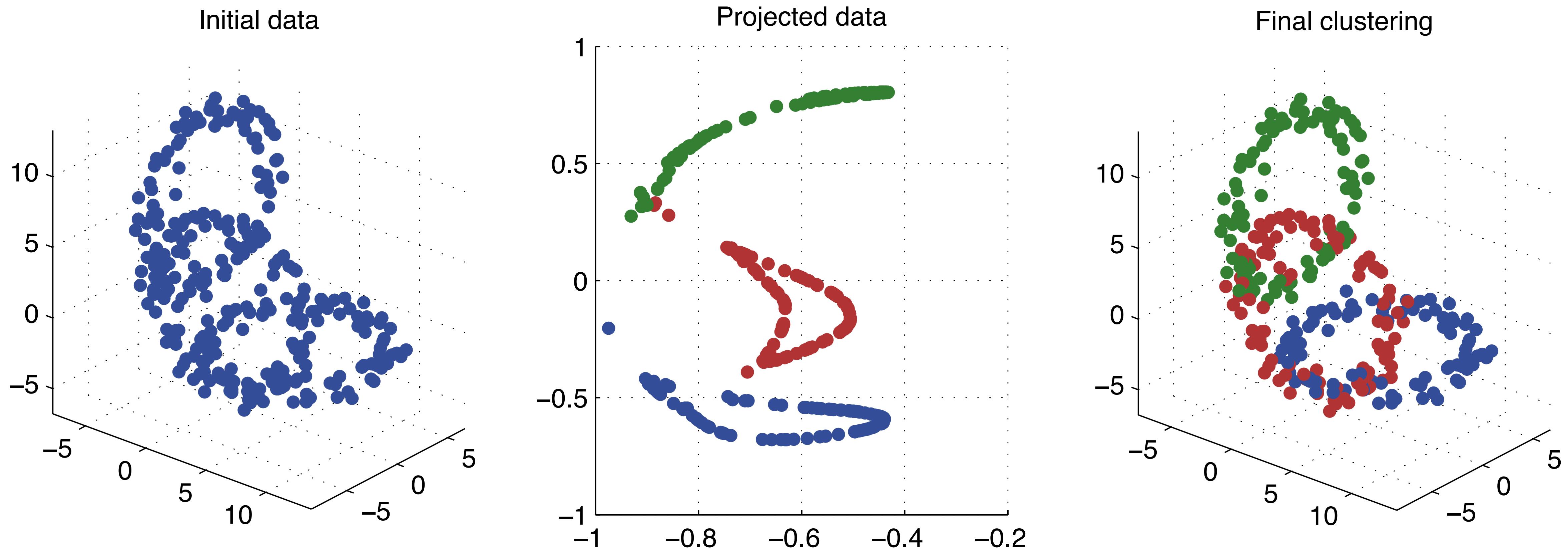
And voilà

- We can now discover non-compact clusters
- Does this sound familiar?
 - KPCA / Manifolds / MDS
- Spectral algorithms
 - Close relationship to graph theory



Works with really complex clusters

- E.g. three intertwined rings



Getting picky

- Any objection to k-means?

A familiar complaint

- Hard assignments are bad ...
 - Just like with kNN we make *hard* assignments
 - Which we should always avoid!
- Just as before, we will “soften up” the model

A new look for the means

- Instead of means, use Gaussians models

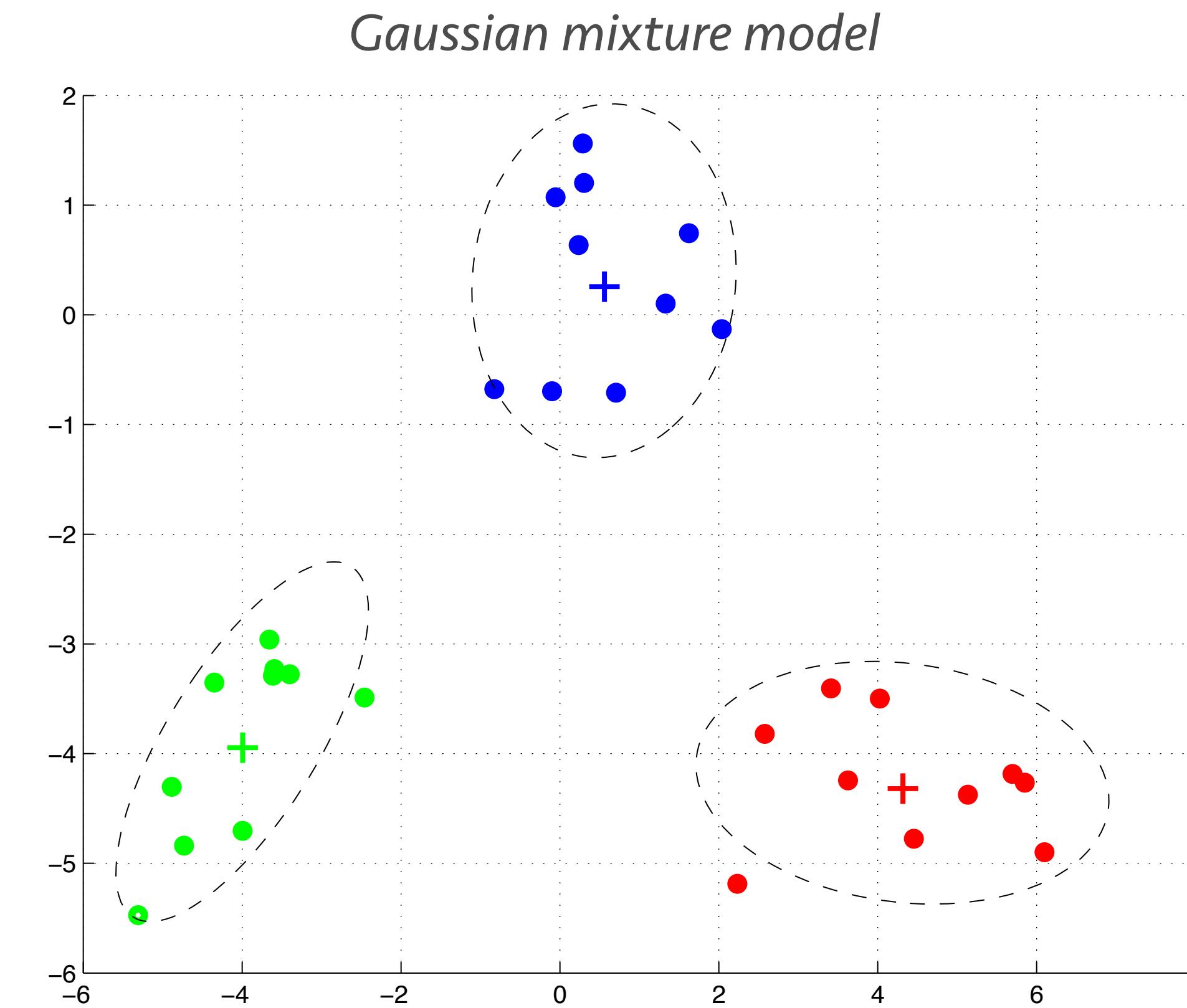
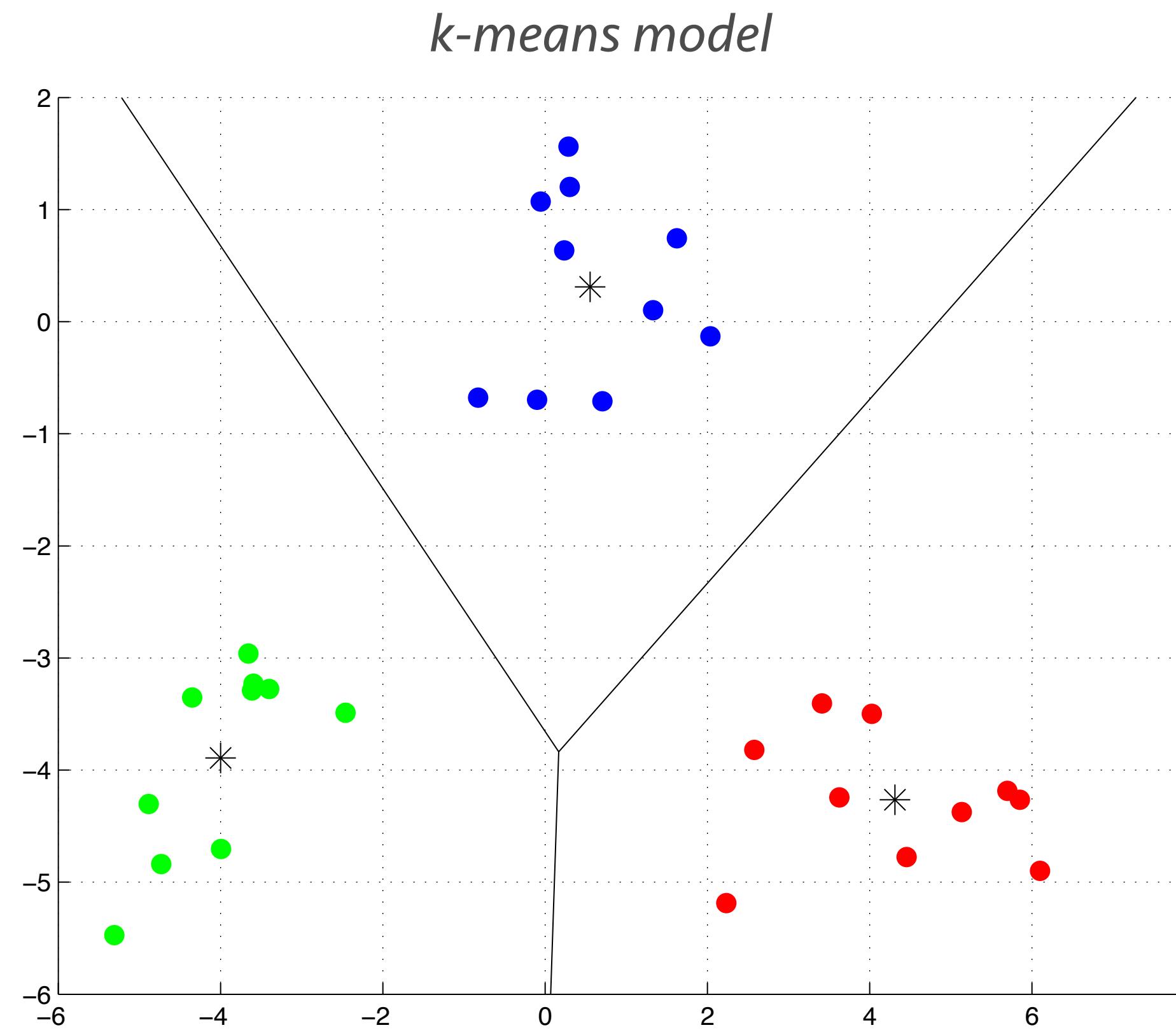
$$d(\mathbf{x}, \mu) = \|\mathbf{x} - \mu\|^2 \rightarrow \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)}$$

- Instead of a hard assignment use *soft* assignments

$$u_{ij} \in \{0,1\} \rightarrow u_{ij} \equiv P(z_j | \mathbf{x}_i, \mu_j, \Sigma_j)$$

A Gaussian Mixture Model

- This defines each cluster as a Gaussian distribution



Formal definition

- Maximize model likelihood given data

$$P(\mathbf{x}) = \sum_k \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- We can't solve this in a closed-form solution!
 - But we can use a process similar to the one we used to solve the k-means problem

“Softening” k-means

- You have multiple components
- Step 1:
 - Compute soft assignment of each data point to a cluster
- Step 2:
 - Re-estimate component models using the soft assignment as weights on the data points
- Repeat until convergence
 - Will always increase likelihood towards a local maximum

Expectation Maximization (EM) for GMMs

- Initialize μ_k, Σ_k, π_k
 - Can be random or based on a k-means result (or something else)
- E-step
 - Find how much each Gaussian explains every available data point

$$\gamma_{n,k} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}$$

Annotations for the equation:

- Cluster prior: π_k
- Cluster mean: μ_k
- Cluster covariance: Σ_k
- Sample: \mathbf{x}_n
- Cluster: k
- Cluster posterior: $\gamma_{n,k}$
- How much x_n belongs to cluster k : Numerator term
- Sum of how much x_n belongs to all clusters: Denominator term

Expectation Maximization (EM) for GMMs

- M-step
 - Weigh data and re-estimate parameters

$$N_k = \sum_n \gamma_{nk} \quad \longleftarrow \text{Cluster posterior sum over all samples}$$

$$\pi_k \leftarrow \frac{N_k}{N} \quad \longleftarrow \text{Cluster prior estimate}$$

$$\mu_k \leftarrow \frac{1}{N_k} \sum_n \gamma_{nk} \mathbf{x}_n \quad \longleftarrow \text{Cluster mean estimate (weigh samples by posterior)}$$

$$\Sigma_k \leftarrow \frac{1}{N_k} \sum_n \gamma_{nk} (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top \quad \longleftarrow \text{Ditto for covariance}$$

EM for GMMs

- Alternate between E and M steps until convergence
- Upon conclusion we have the model parameters

Expectation-Maximization in general

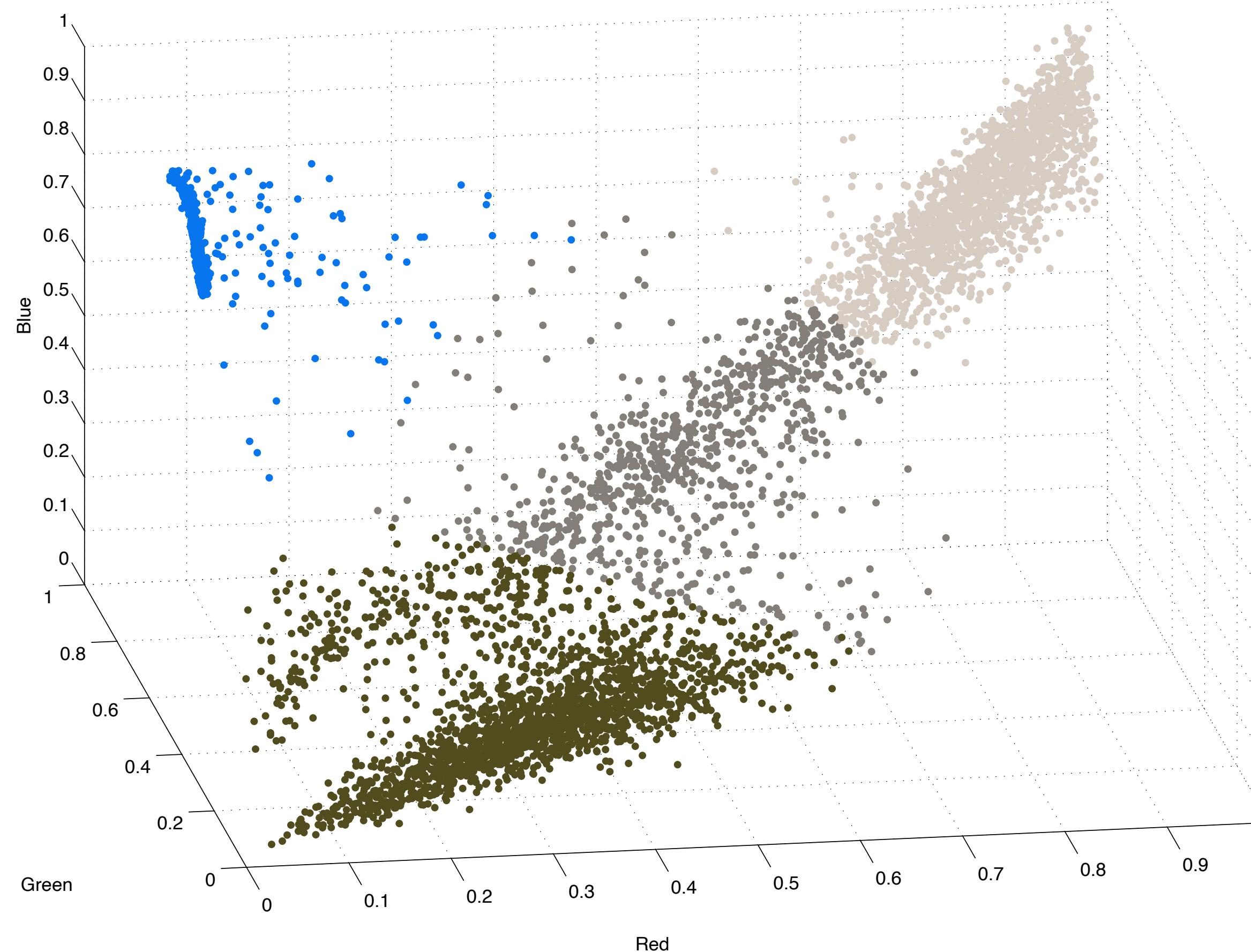
- For a latent model: $P(\mathbf{X}, \mathbf{Z} | \theta)$
 - Maximize $P(\mathbf{X} | \theta)$
- Make up some initial parameters θ
 - E-step: Evaluate $P(\mathbf{Z} | \mathbf{X}, \theta)$
 - M-step: Estimate updated parameters

$$\theta^{new} = \arg \max_{\theta} \sum_{\mathbf{Z}} P(\mathbf{Z} | \mathbf{X}, \theta^{old}) \log P(\mathbf{X}, \mathbf{Z} | \theta)$$

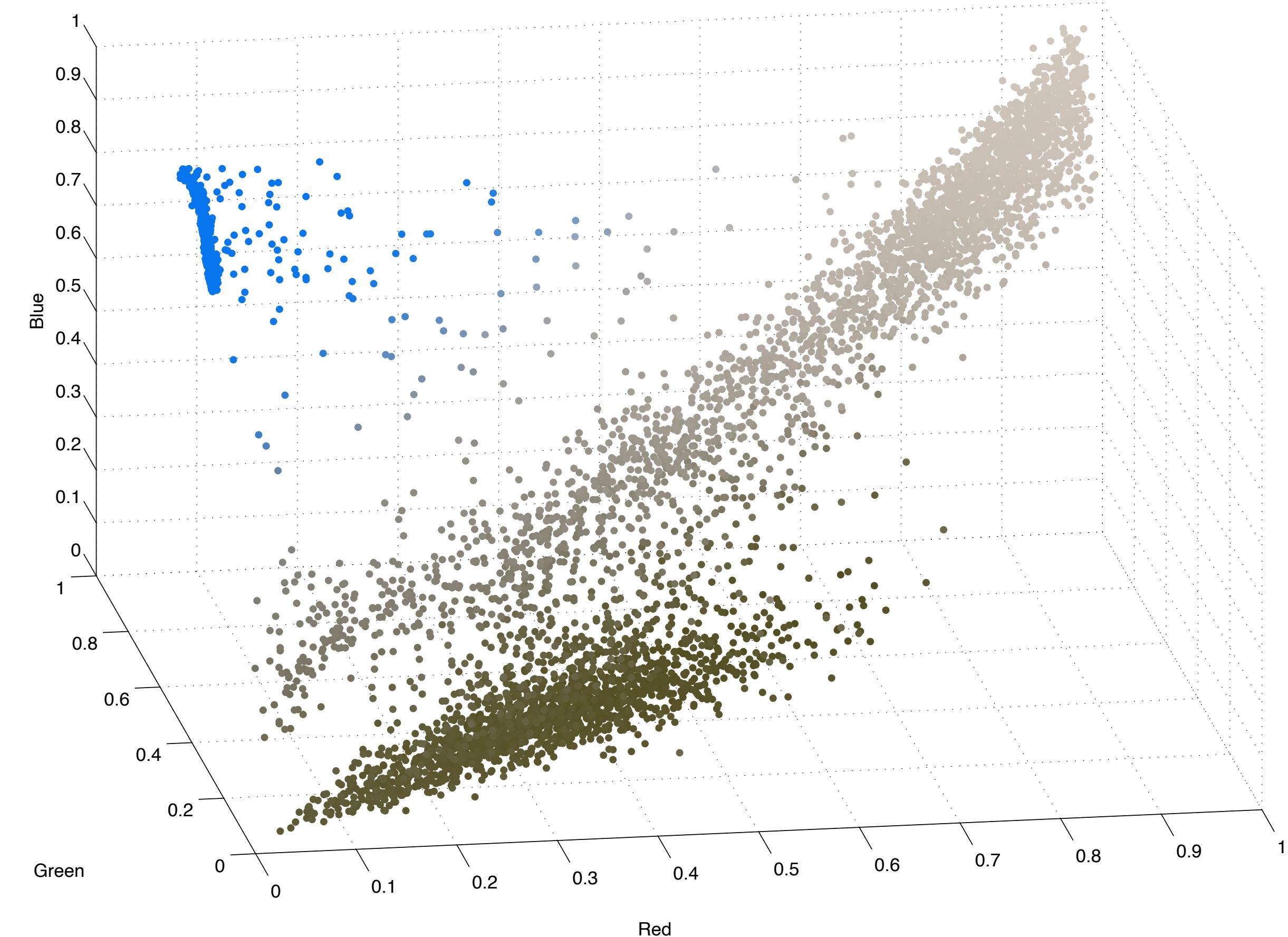
- Repeat until convergence

GMMs on El Capitan

k-means clustering

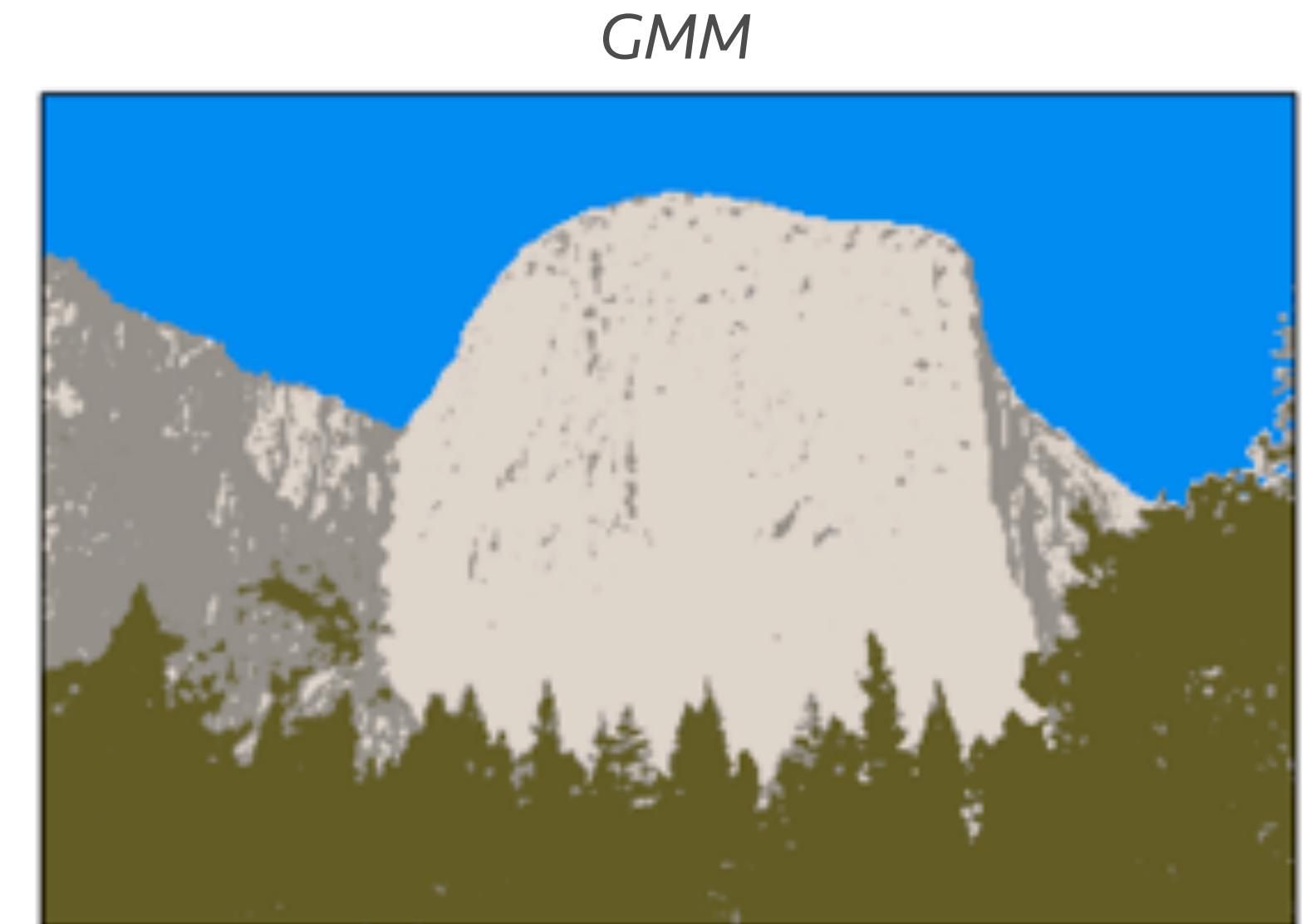
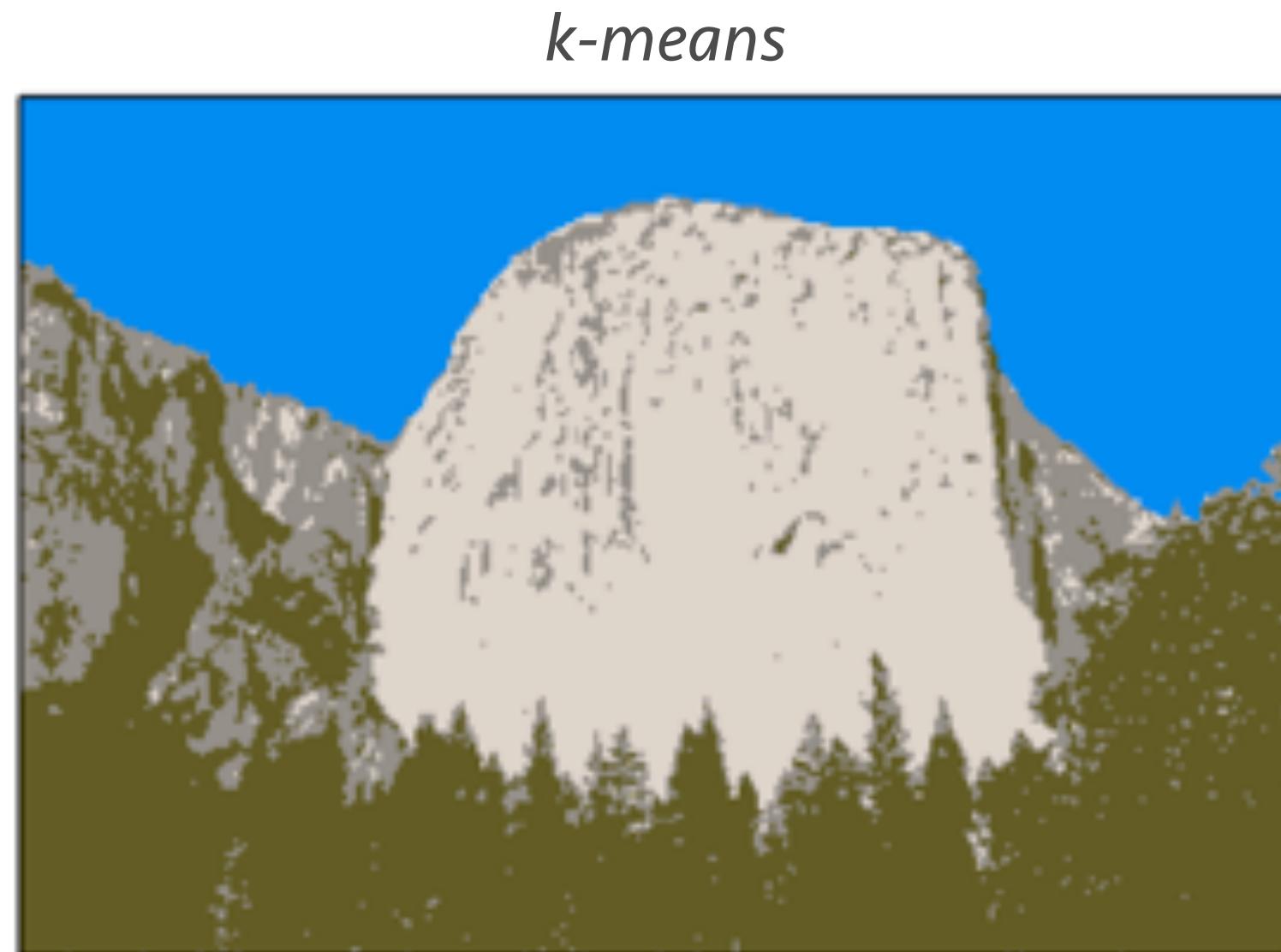


GMM clustering



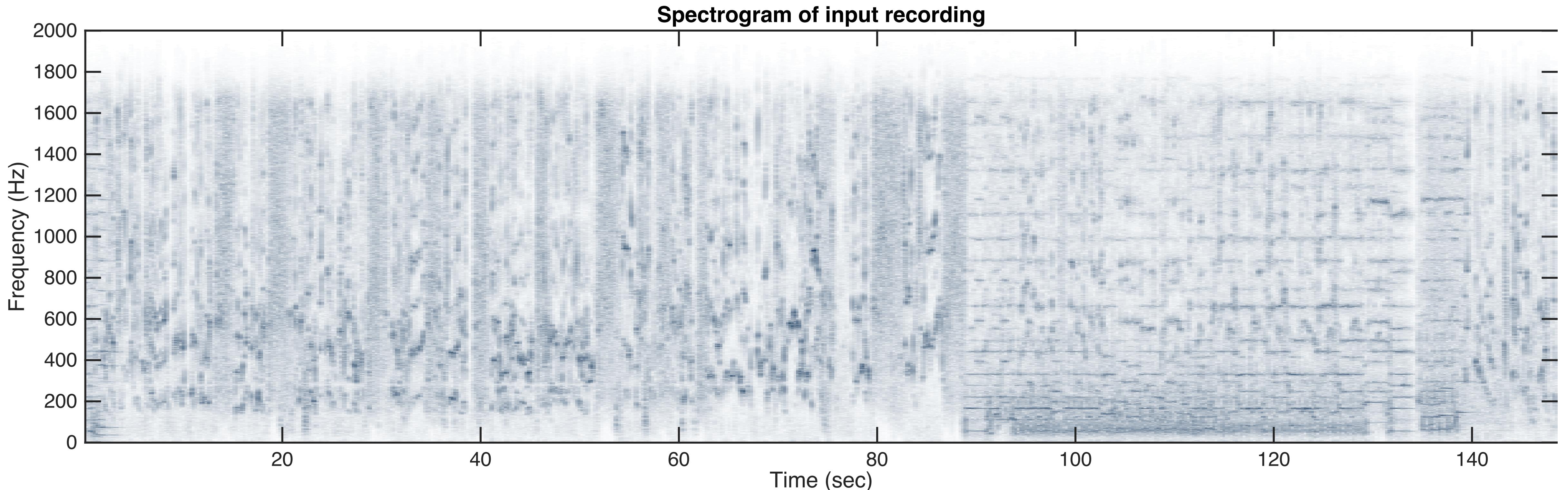
GMMs on El Capitan

- More accurate than k-means
 - Graceful handling of ambiguous regions
- Probabilistic interpretation!



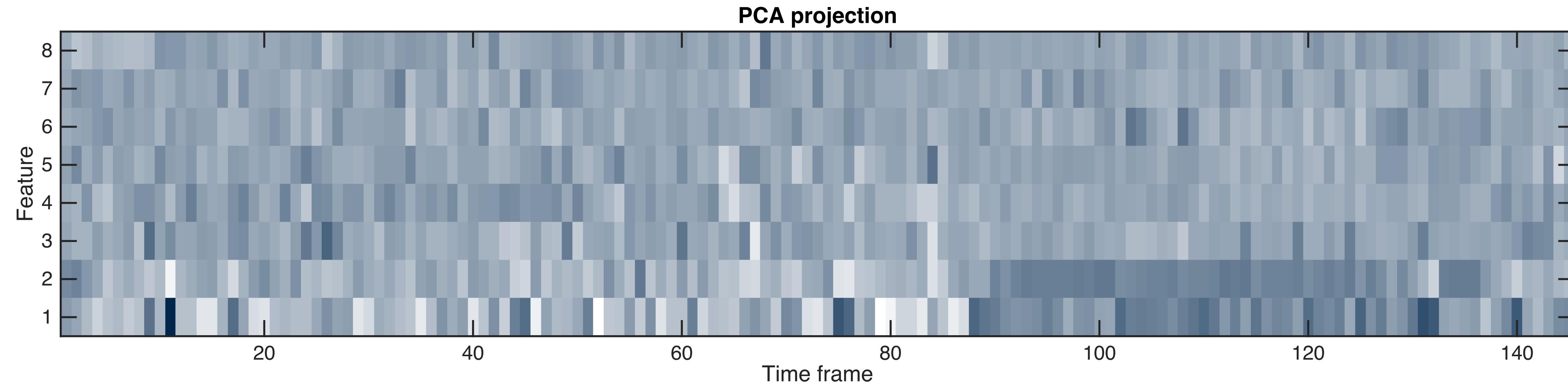
One more example

- Taking the spectrogram of a TV show
 - Let's see what the clusters are like



Drop the dimensions

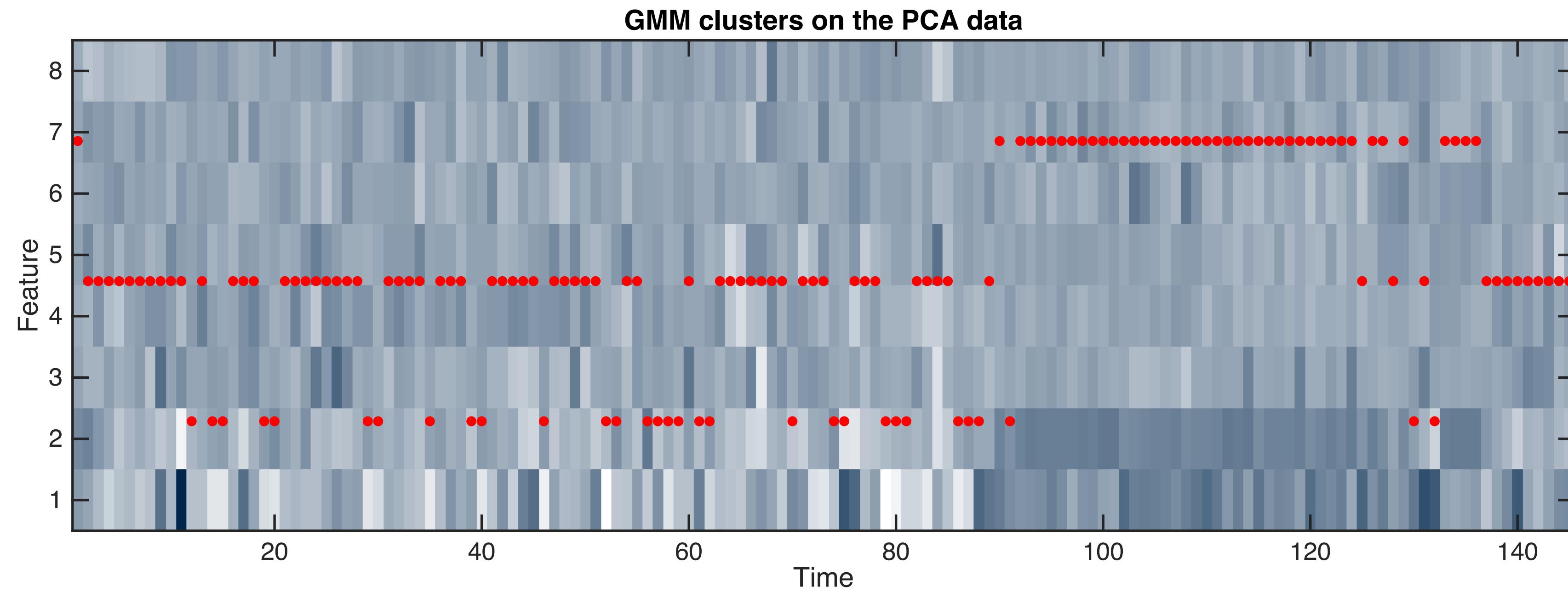
- We can do PCA
 - Dropping from 2049 dimensions down to 8



- Can you see any clusters?

Performing GMM

- Looking for three clusters
 - Finds frames with similar spectra (in the low dim PCA space)



What do these frames sound like?



Cluster 3 = Music



Cluster 2 = Speech



Cluster 1 = Audience laughter

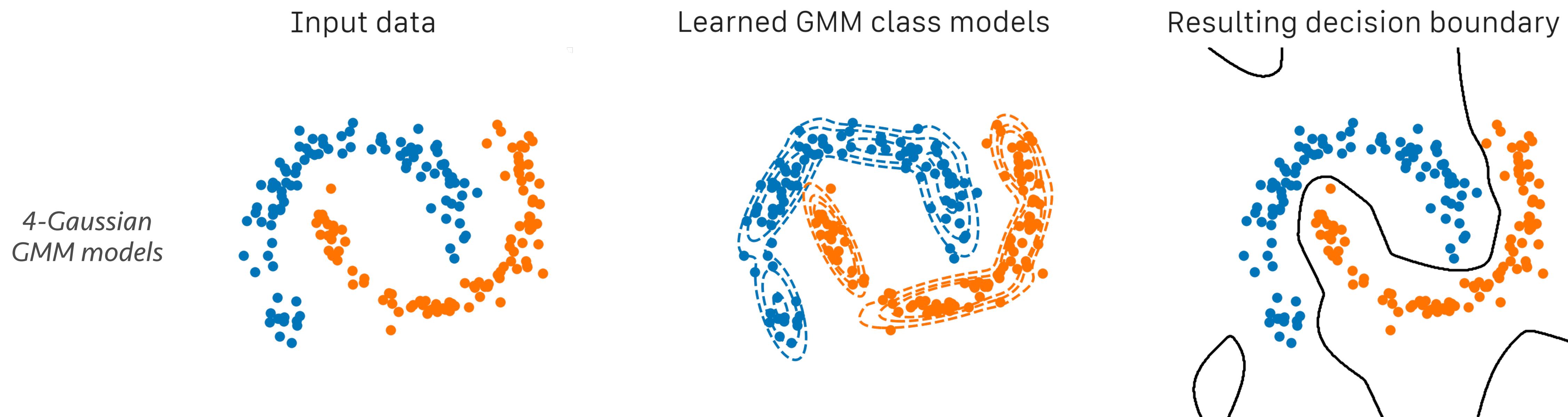
Some notes on GMMs

- In practice, full covariances can explode!
 - You can use diagonal covariance matrices instead
 - Use more Gaussians to make up for the fewer parameters
- Initial conditions matter a lot
 - You can initialize means using k-means
 - But that can also lead to poor starting conditions ...

GMMs for classification

- GMMs model the distribution of data
 - So we can also use as class models
- Learn one GMM per class
 - Which will give you the class likelihood
- Then evaluate new sample likelihood given GMMs
 - Can produce very flexible boundaries

Examples of GMM classification



Recap

- Hierarchical clustering
- K-means/medoids
 - Hard decisions, fast, reliable
- Spectral clustering
 - Finds quirky clusters
- Gaussian Mixture Models
 - Soft decisions, probabilistic, fantastic tool
- Expectation Maximization
 - Parameter estimation with mixture models

Next lecture

- Learning time series
- Hidden Markov models

Reading

- Textbook chapter 11, 14.5, 2.5.5
- Optional: all of the clustering chapters
- Spectral clustering:
 - <http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf>

Yay, one more problem set!

- Problem set 3 is out
 - It's all about detection/classification
 - Due Oct 18th
- It includes a final project proposal as well
 - This part can be submitted up to a week later (by Oct 25th)
 - Form teams, feel free to discuss your plans with me and Krishna
 - Teams should be no less than 2 and more than 3 people