CS598PS - Machine Learning for Signal Processing

Clustering

6 October 2017

Today's lecture

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

Back to unsupervised

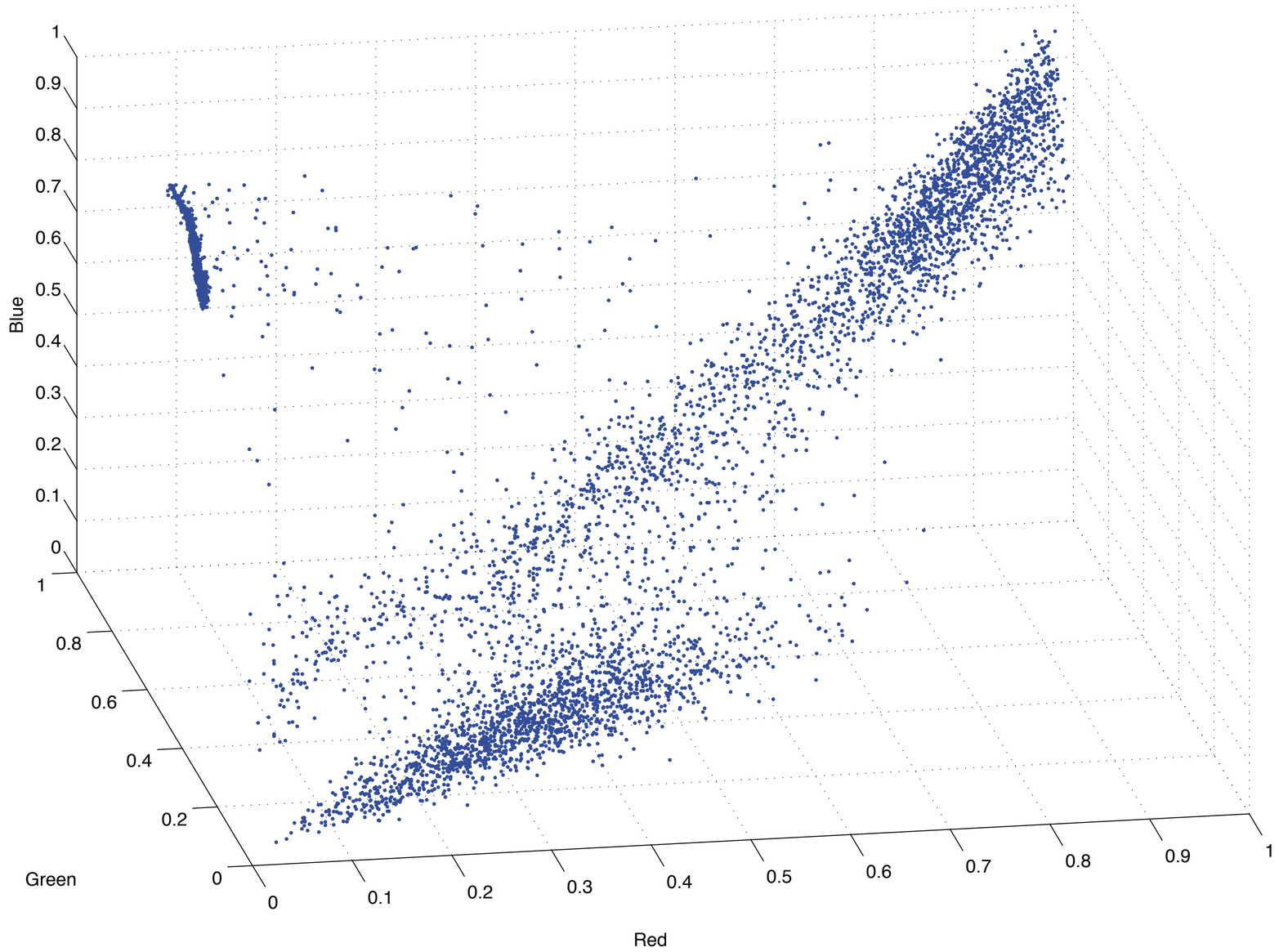
- Supervised learning
  - Use labeled data to do something smart

- What if the labels don't exist?

Some inspiration

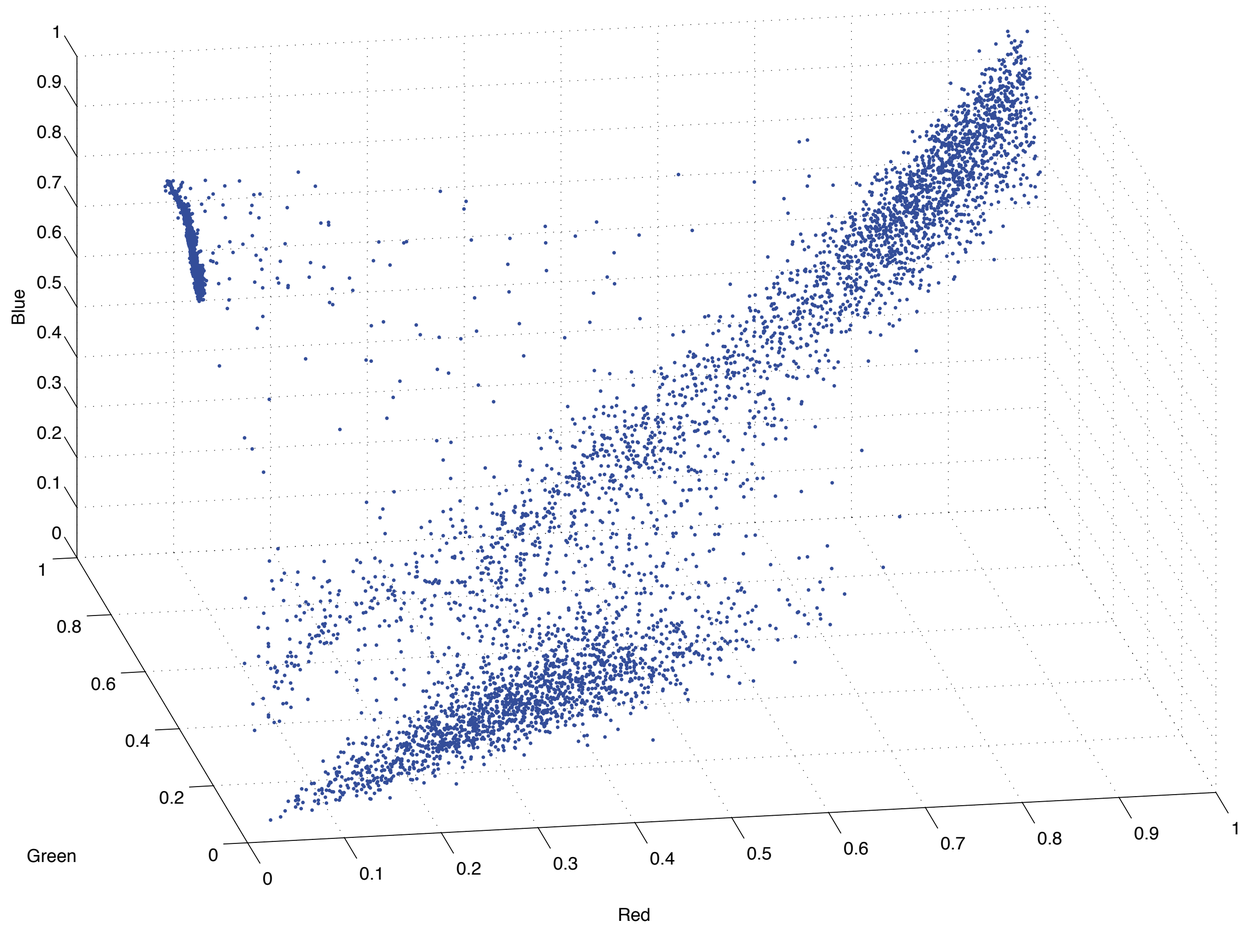
El Capitan, Yosemite National Park

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

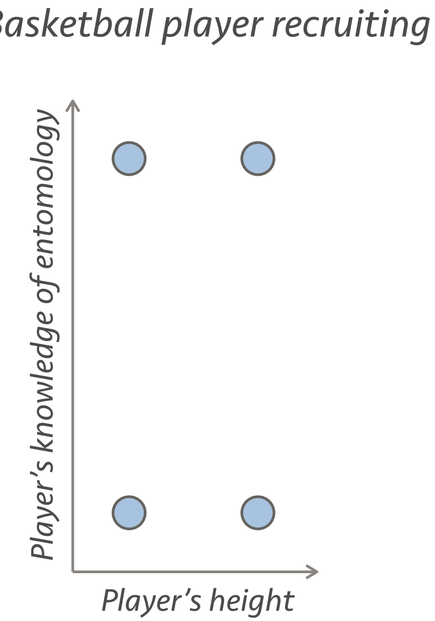
- Discover classes in data
  - Divide data in sensible clusters
- Fundamentally ill-defined problem
  - There is often no correct solution
- Relies on many user choices

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?

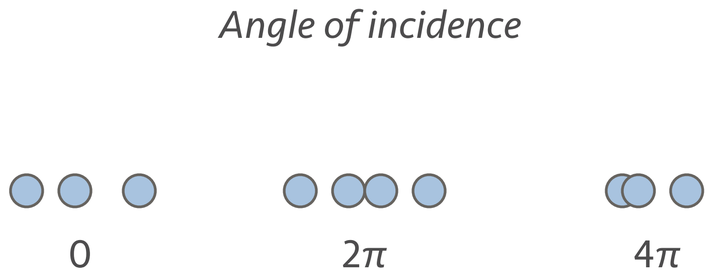
Know what you want

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



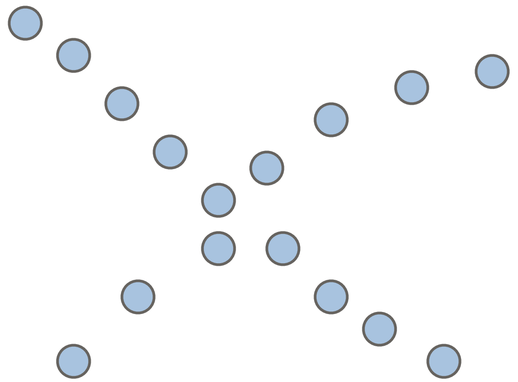
Know your space

- Define a sensible proximity measure



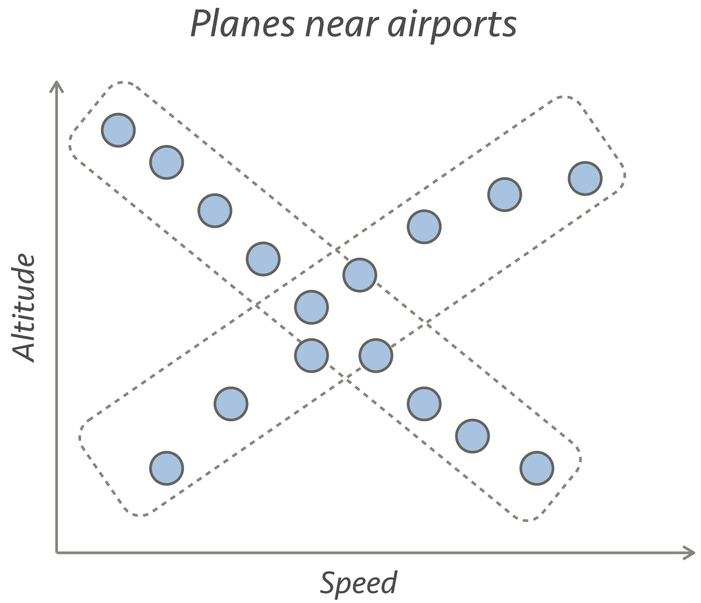
Know your cluster type

- What forms a cluster in your space?



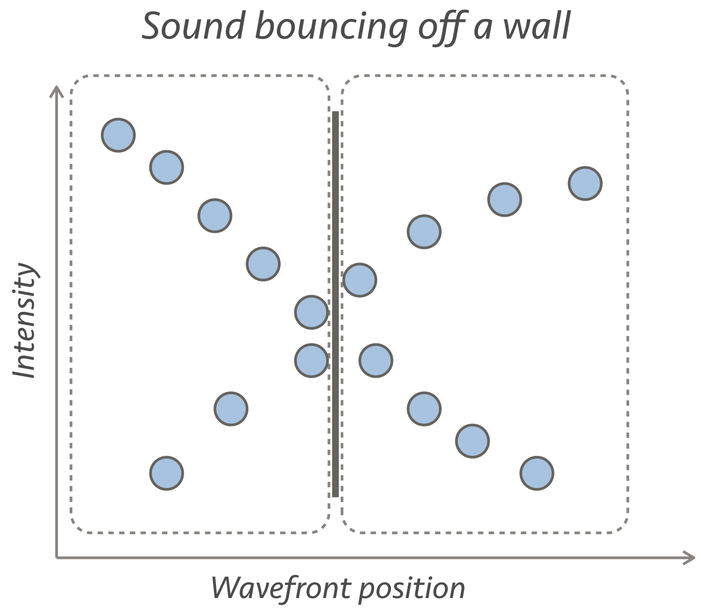
Know your cluster type

- What forms a cluster in your space?



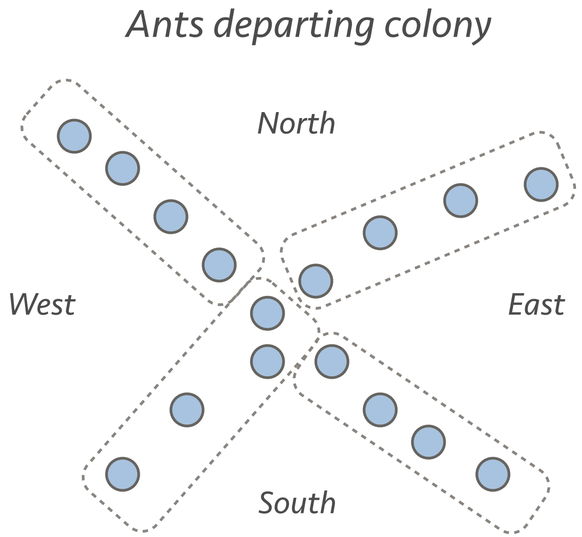
Know your cluster type

- What forms a cluster in your space?



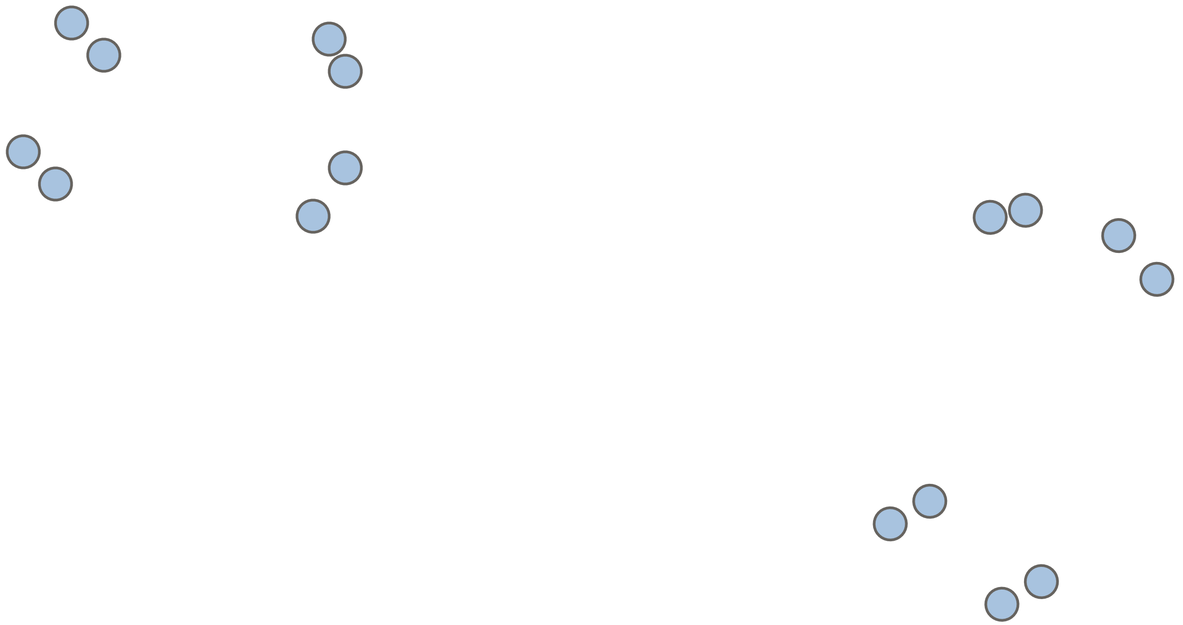
Know your cluster type

- 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

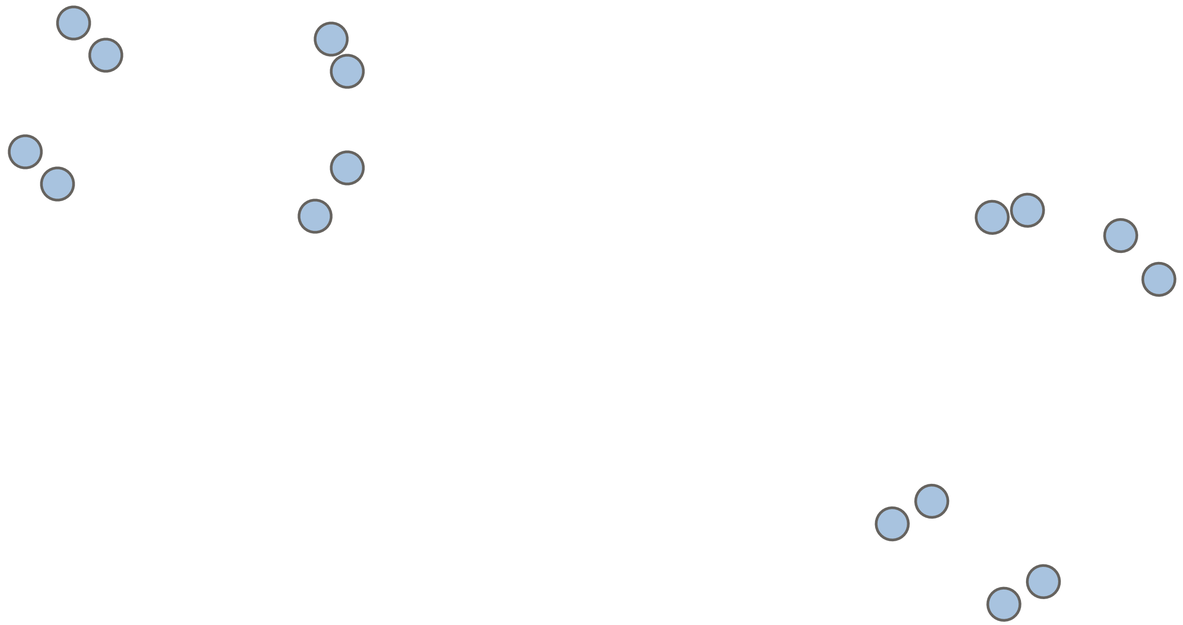
- Tons of methods
- We can use step-based logical steps
  - e.g., find two closest point and merge, repeat
- Or formulate a global criterion

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:

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

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

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

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

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

Agglomerative clustering

- Choose: $R_0 = \{C_i = \{\mathbf{x}_i\}, i = 1,\ldots,N\}$
- For $t = 1,\ldots$
  - 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:

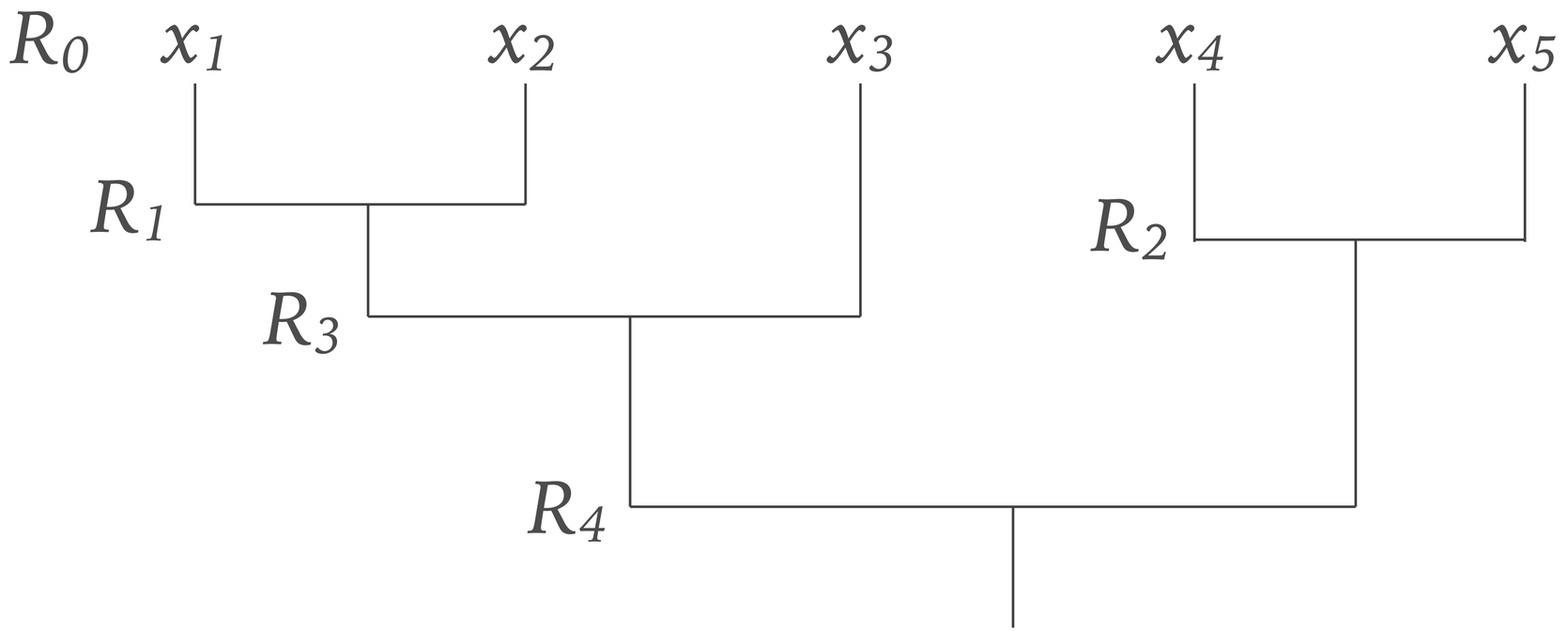
$$C_q = C_i \cup C_j$$

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

  - Until we only have one cluster

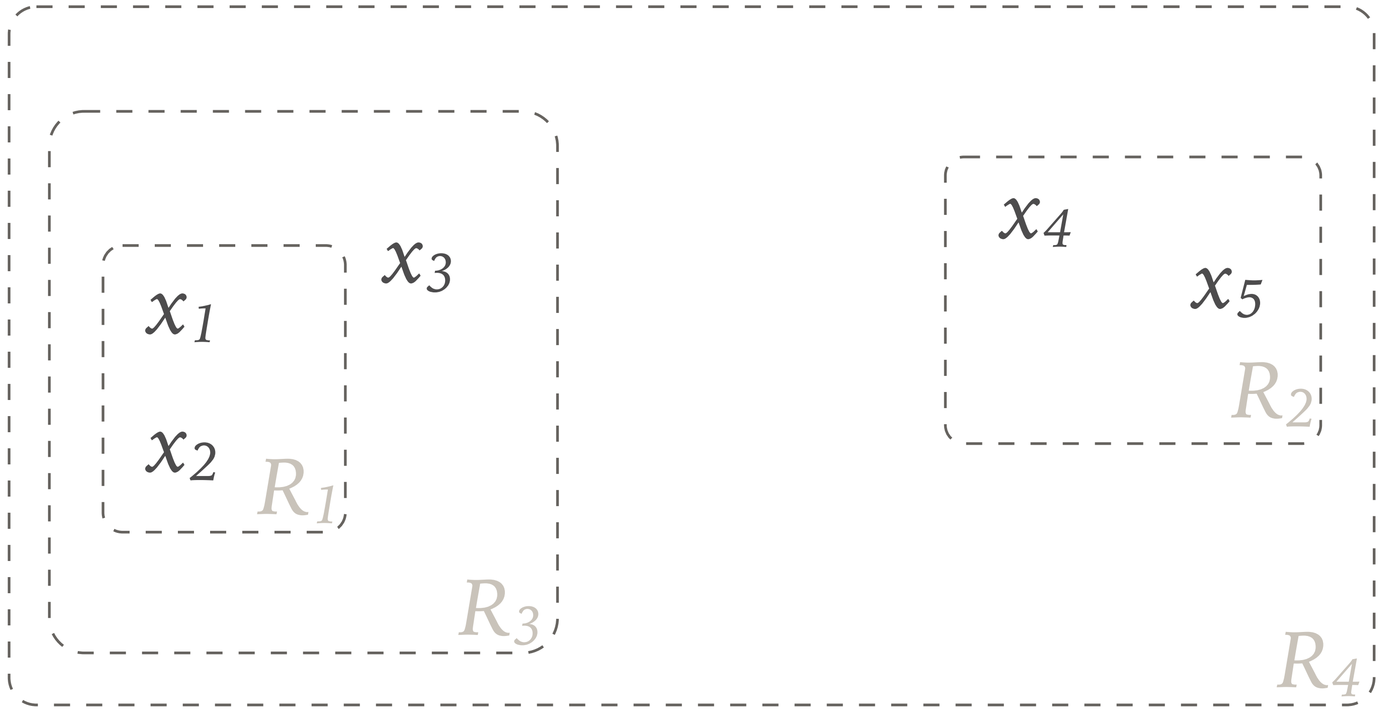
Representing this process

- The Dendrogram



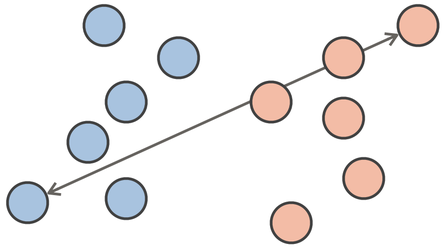
Another way

- Venn diagram

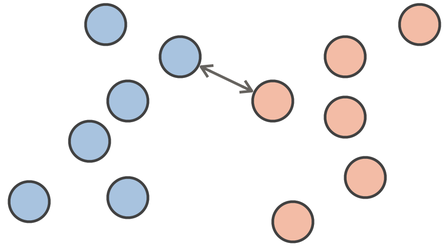


Cluster distance?

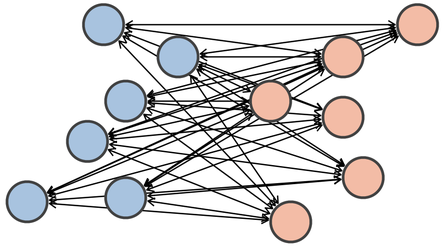
- Complete linkage
  - Merge clusters that result in the smallest diameter



- Single linkage
  - Merge clusters with two closest data points

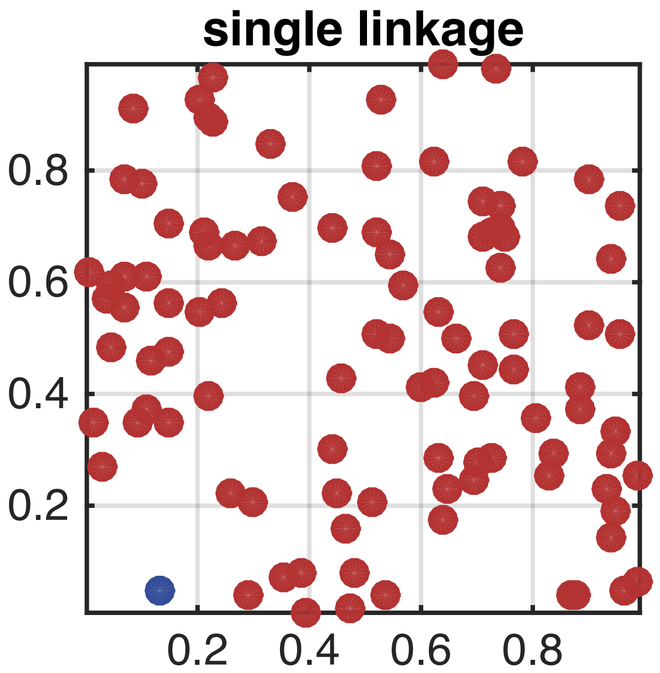


- Group average
  - Use average of distances

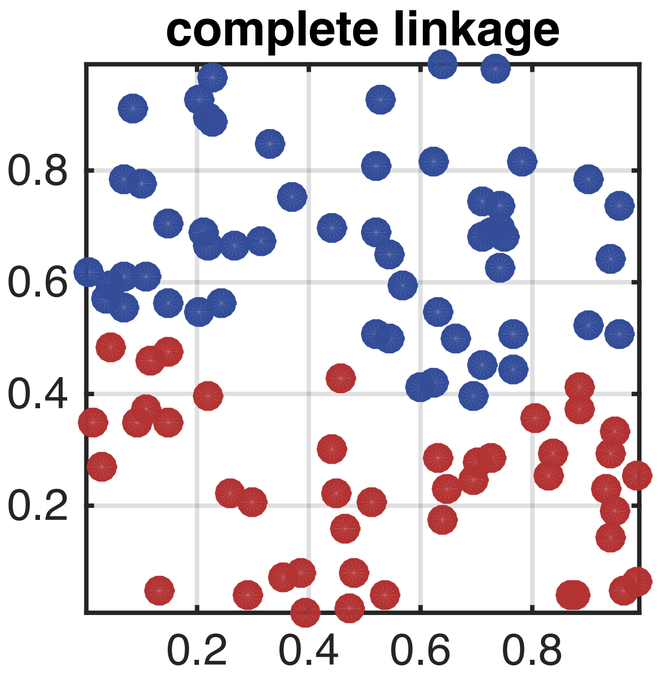


Linkage behavior

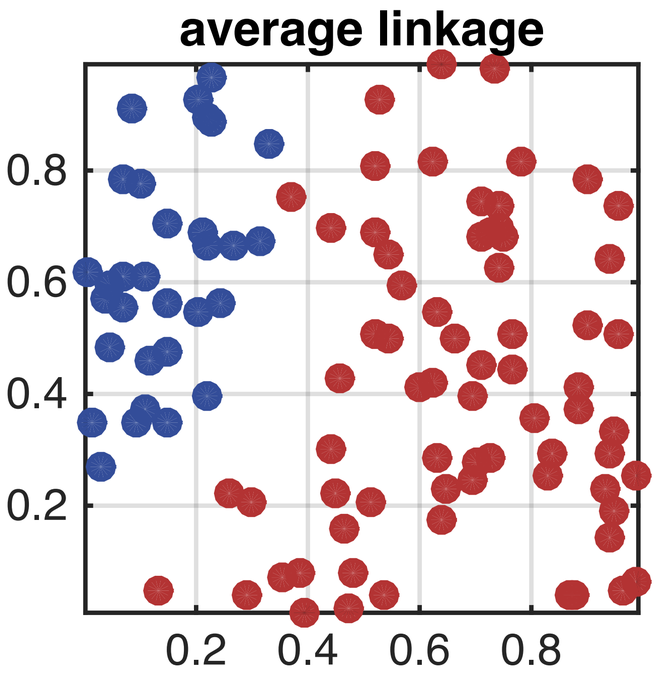
- Clustering uniformly distributed data
  - Where there aren't really any clusters



Links nearby points until the loneliness one is left out



Makes two compact clusters



Finds "roundest" clusters that are separated by a void

What's involved

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

$$\begin{pmatrix} N-t \\ 2 \end{pmatrix} = \frac{\big(N-t\big)\big(N-t-1\big)}{2}$$

- Overall comparisons:

$$\sum_{t=0}^{N-1} \begin{pmatrix} N-t \\ 2 \end{pmatrix} = \frac{\big(N-1\big)N\big(N+1\big)}{6}$$

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: $R_0 = \{ \mathbf{X} \}$
- For $t = 1,\ldots$
  - For $k = 1,\ldots,t$
    - Find least similar sub-clusters in each cluster
    - Pick the least similar from that set:

$$\arg\max_{k,i,j} d\left(C_{k,i}, C_{k,j}\right)$$

    - New clustering is now:

$$R_t = \left(R_{t-1} - \{C_k\}\right) \cup \left\{C_{k,i}, C_{k,j}\right\}$$

  - 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

Using a global criterion

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

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

  - θ are the cluster parameters (e.g. mean)
  - $\mathbf{U} \in \{0,1\}$ is an assignment matrix
  - $d()$ 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 $\mathbf{U}$

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

  - Estimate θ

$$\sum_i u_{ij} \frac{\partial d(\mathbf{x}_i, \theta_j)}{\partial \theta_j} = 0$$

  - Repeat until satisfied

K-means

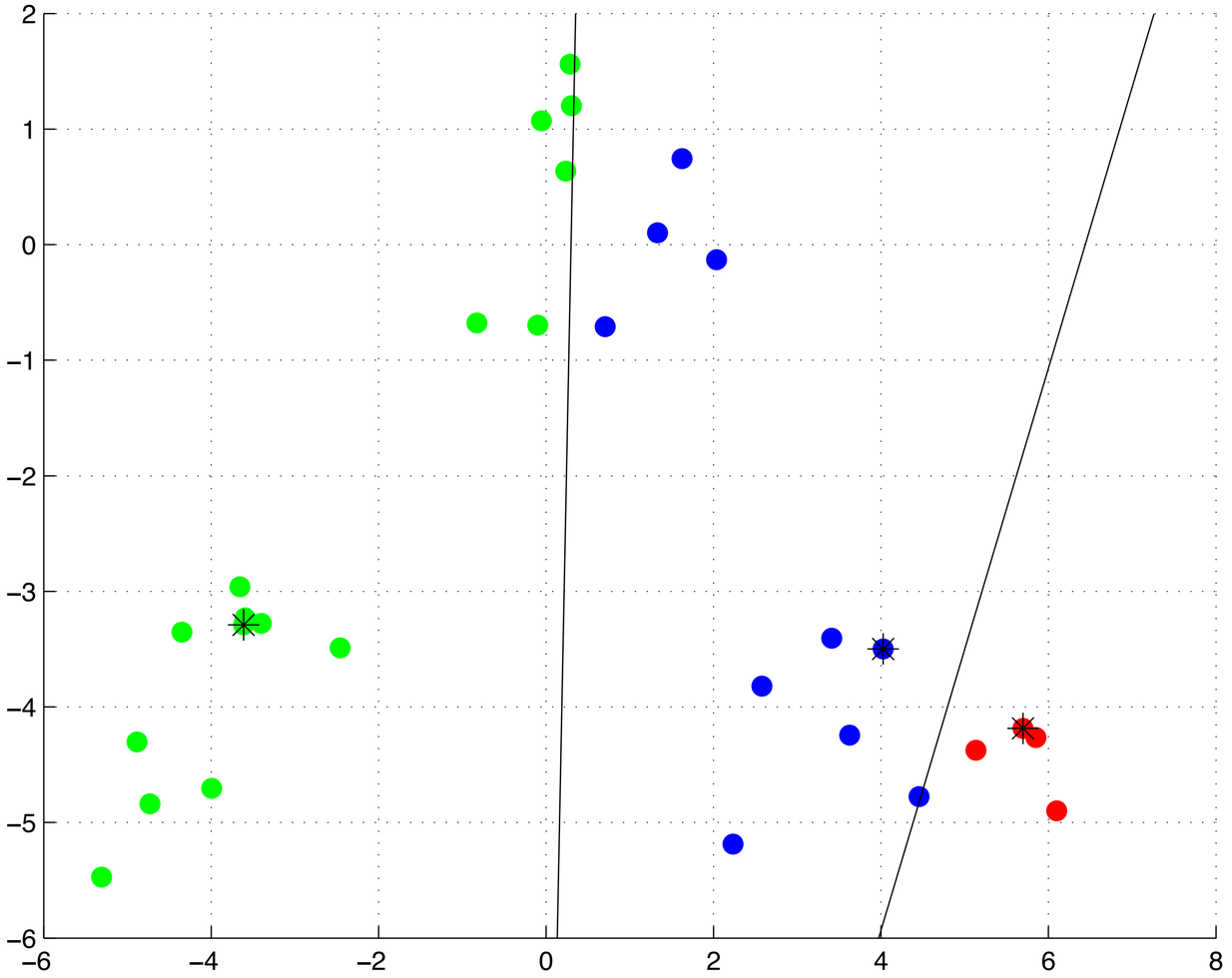
- Standard and extremely-popular algorithm
- Finds clusters in terms of region means
- Optimizes squared Euclidean distance from cluster means

$$d\left(\mathbf{x},\mu\right)=\left\|\mathbf{x}-\mu\right\|^2$$

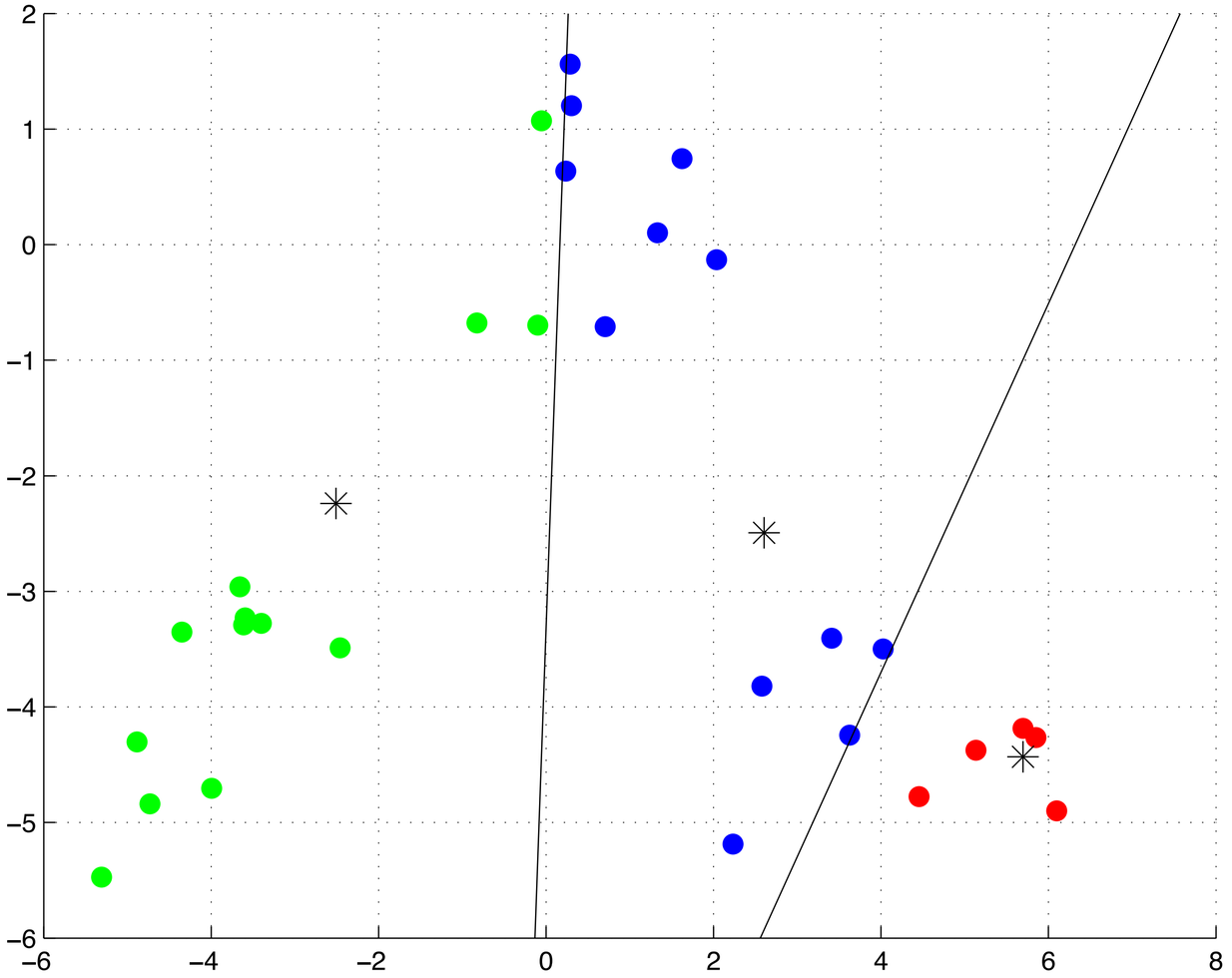
K-means algorithm

- Initialize k means μ_j
- Iterate
  - Assign samples x_i to closest mean μ_j
  - Estimate μ_j from assigned samples x_i
- Repeat until convergence

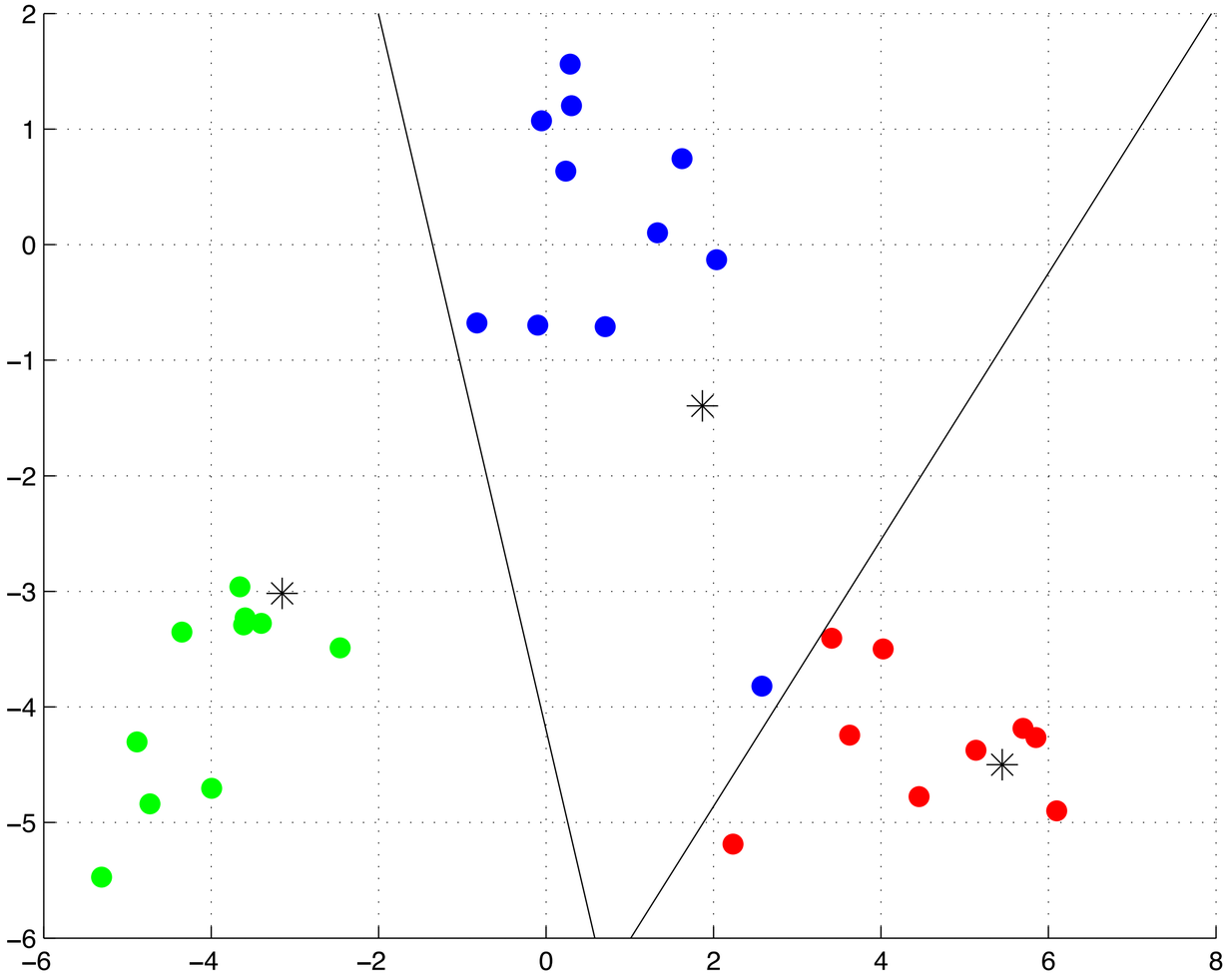
Example run – step 1



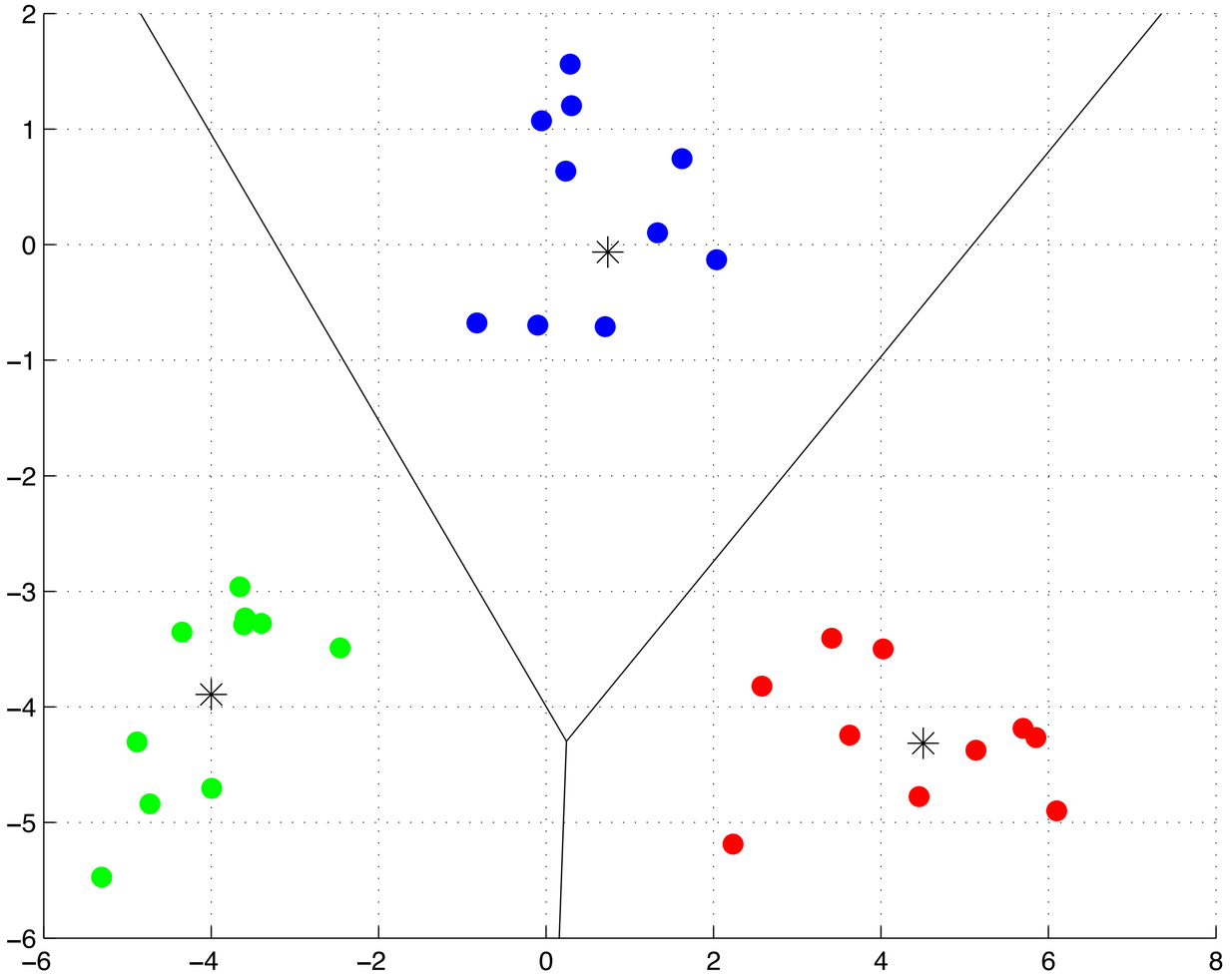
Example run – step 2



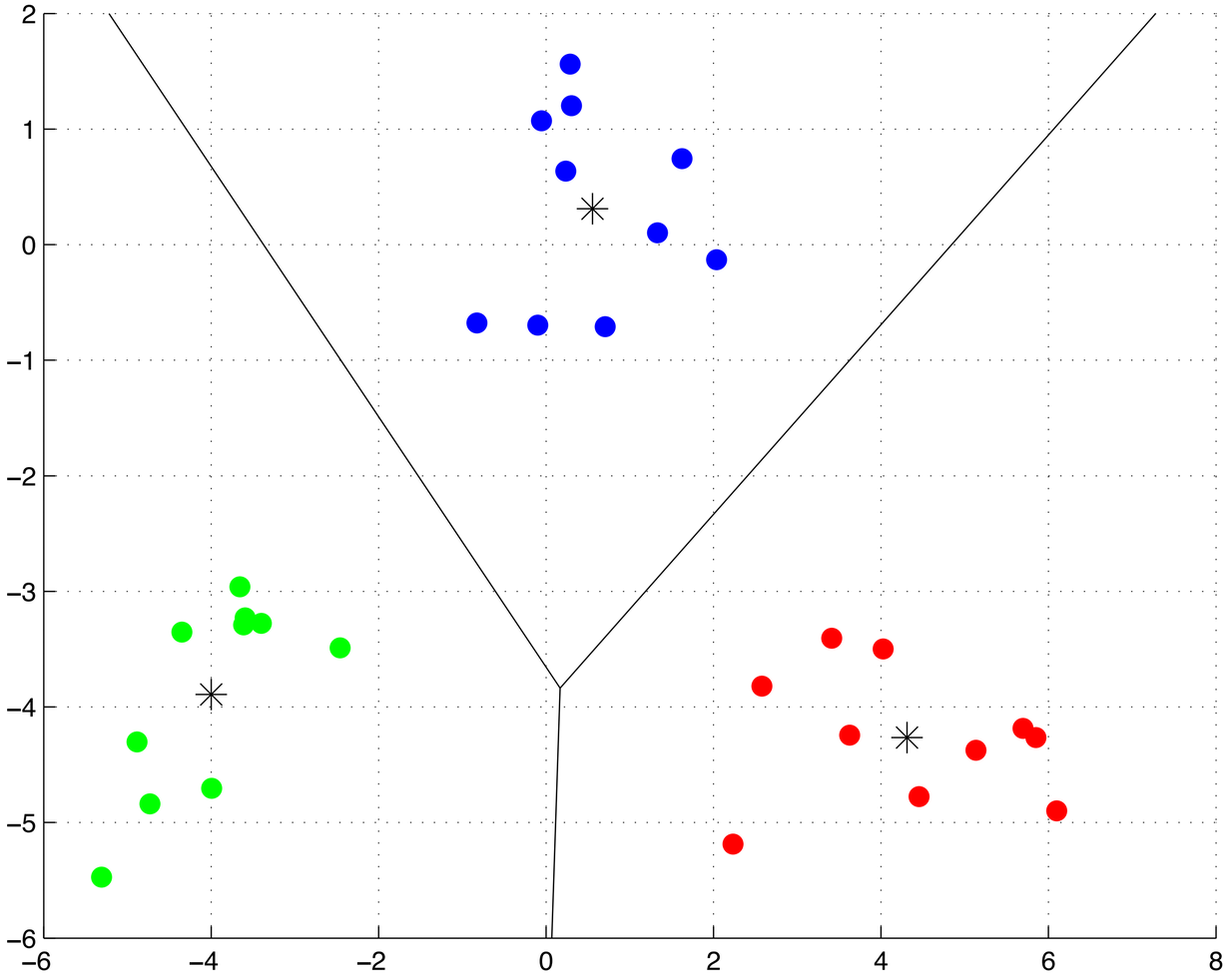
Example run – step 3



Example run – step 4



Example run – step 5

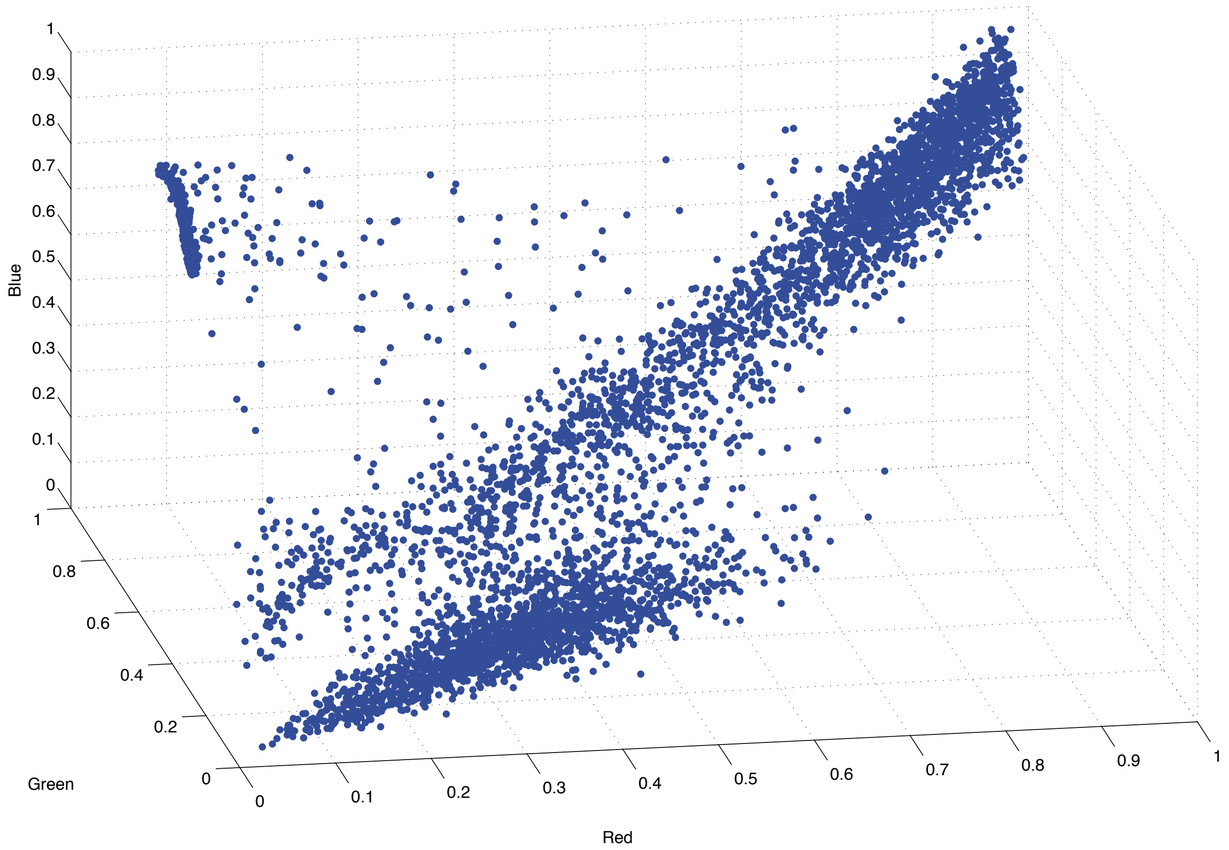


How well does it work?

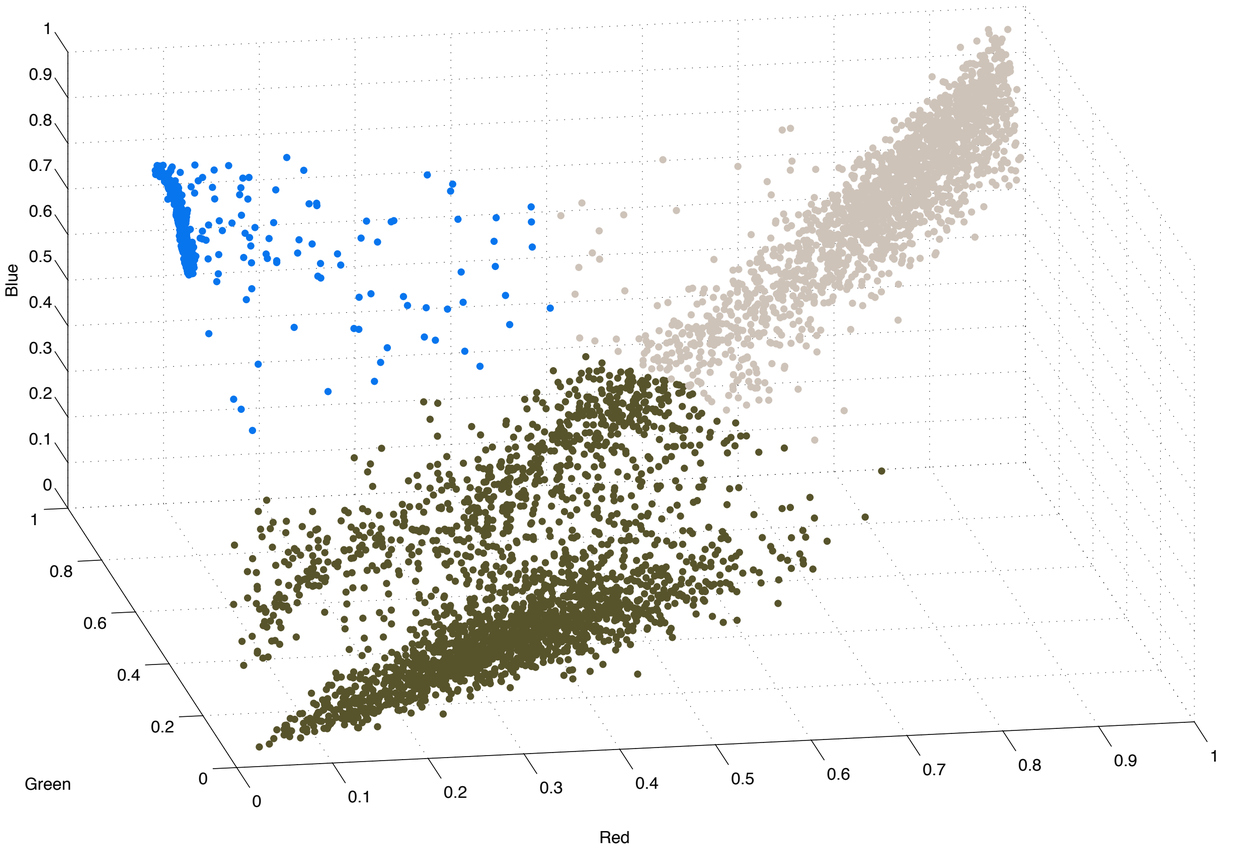
- Converges to a minimum of cost function
  - Not for all distances though!
- Is heavily biased by starting positions
  - Many tricks to alleviate that
- Sensitive to outliers

K-means on El Capitan

Input data



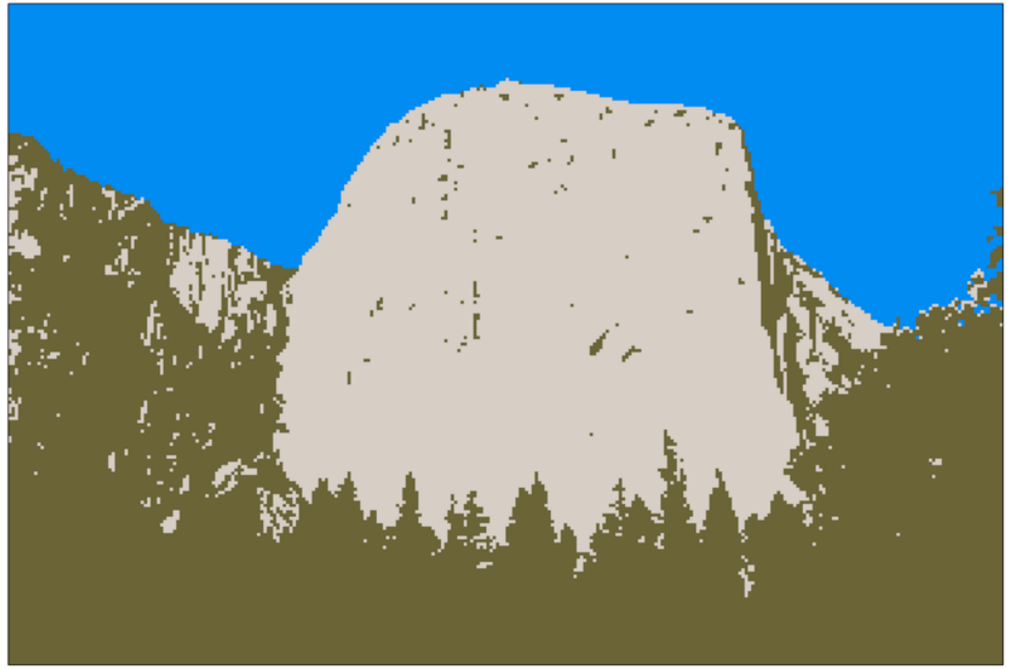
Clustered data



K-means on El Capitan

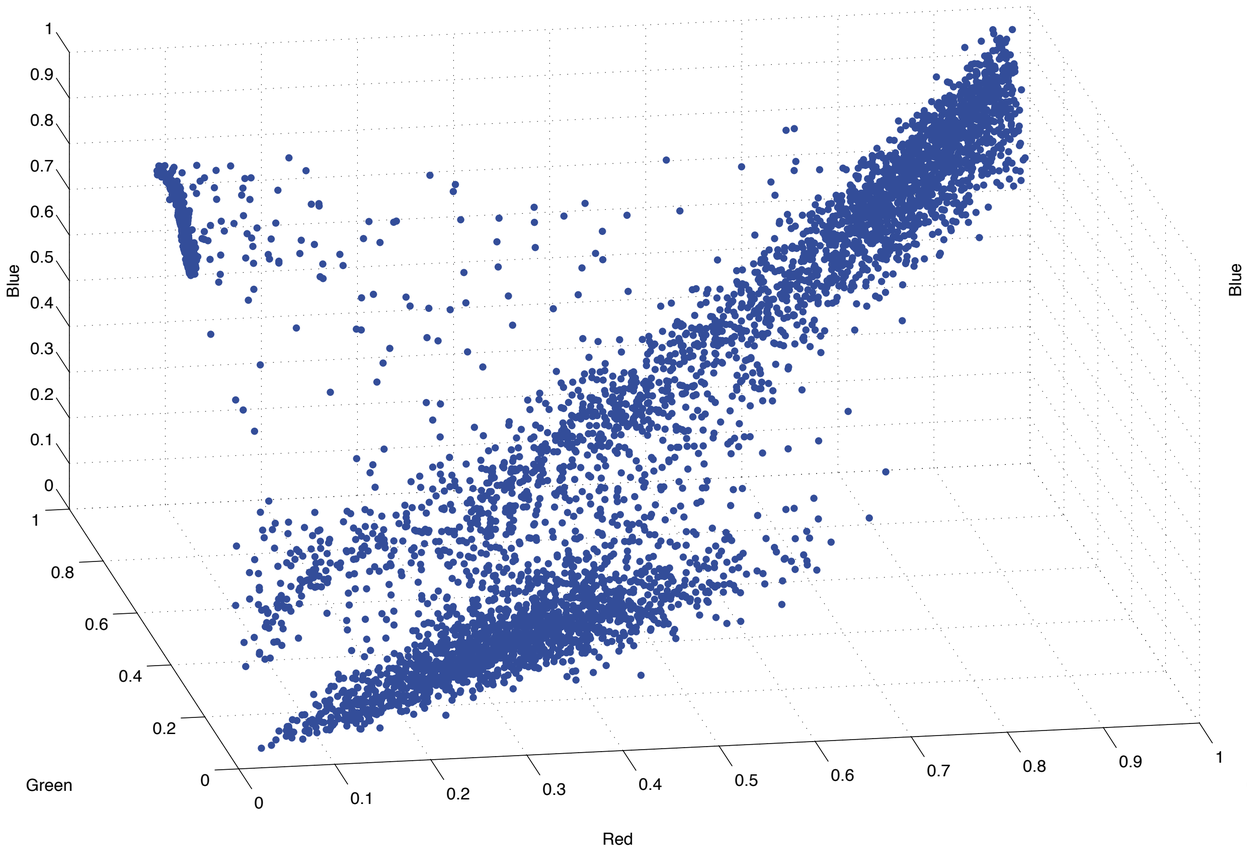
Input data

Clustered data

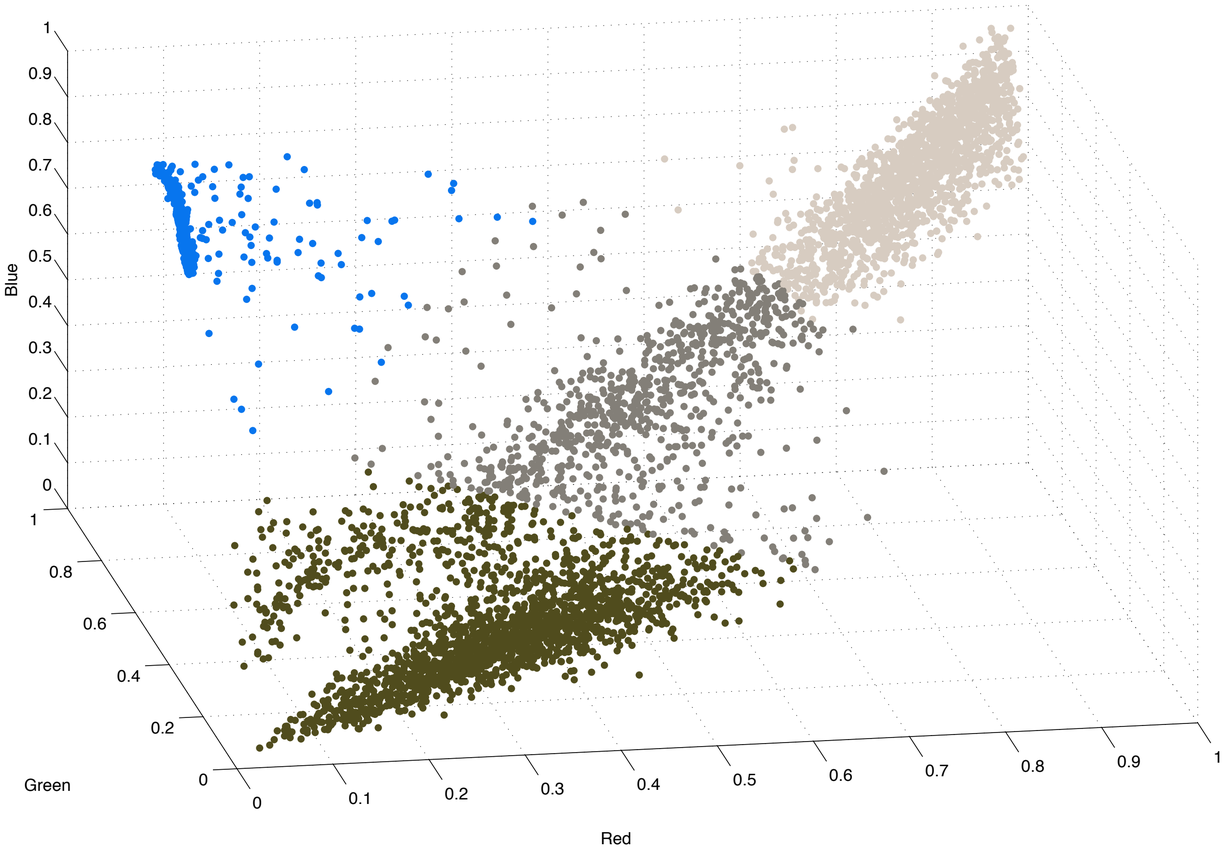


K-means on El Capitan

Input data



Clustered data



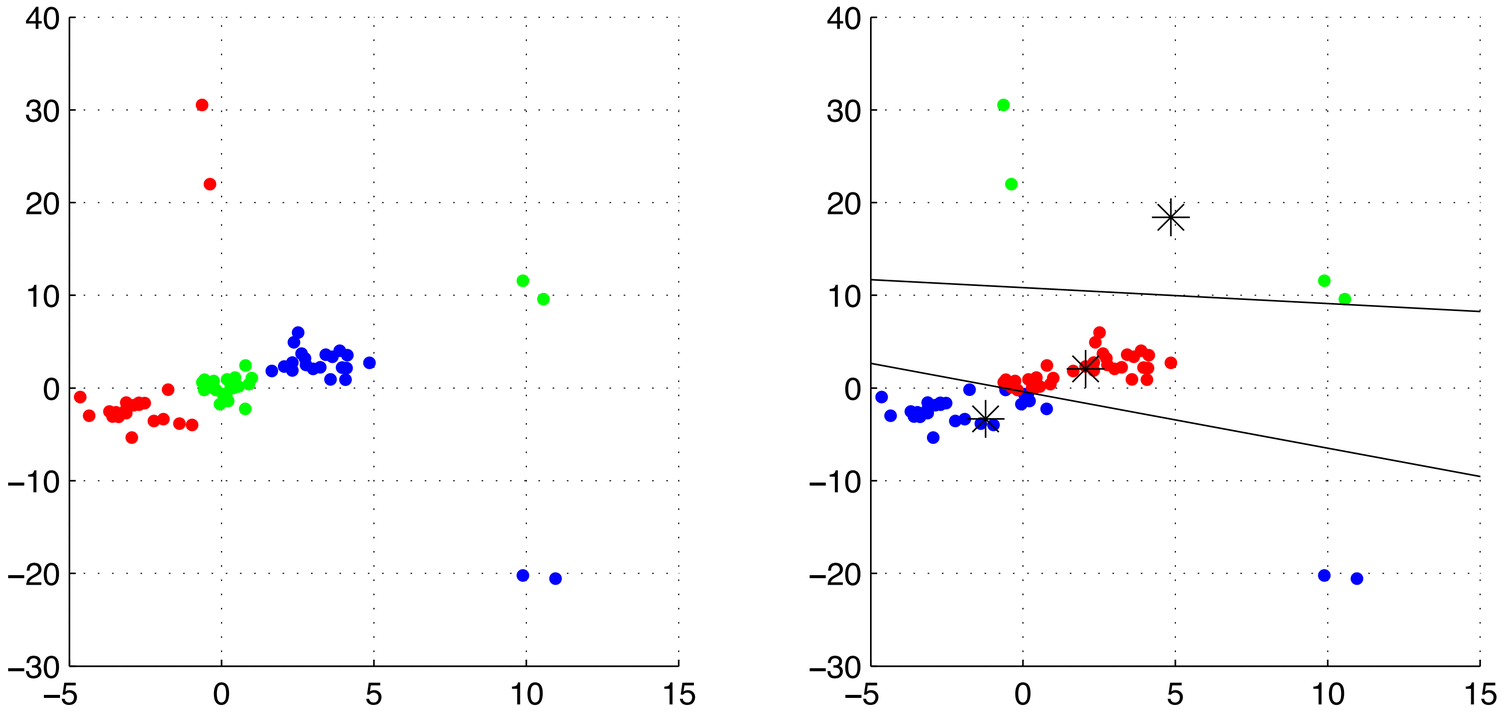
K-means on El Capitan

Input data

Clustered data

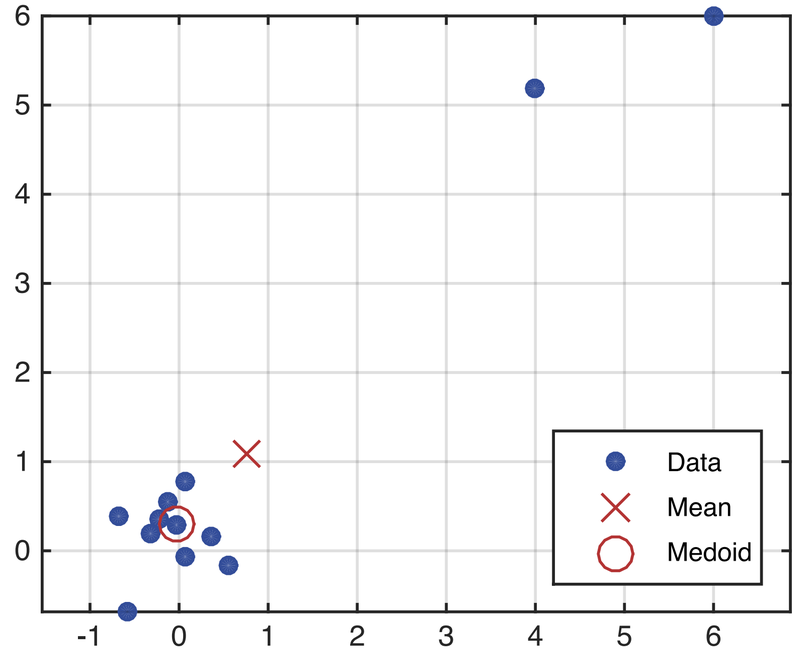
One problem

- K-means struggles with outliers

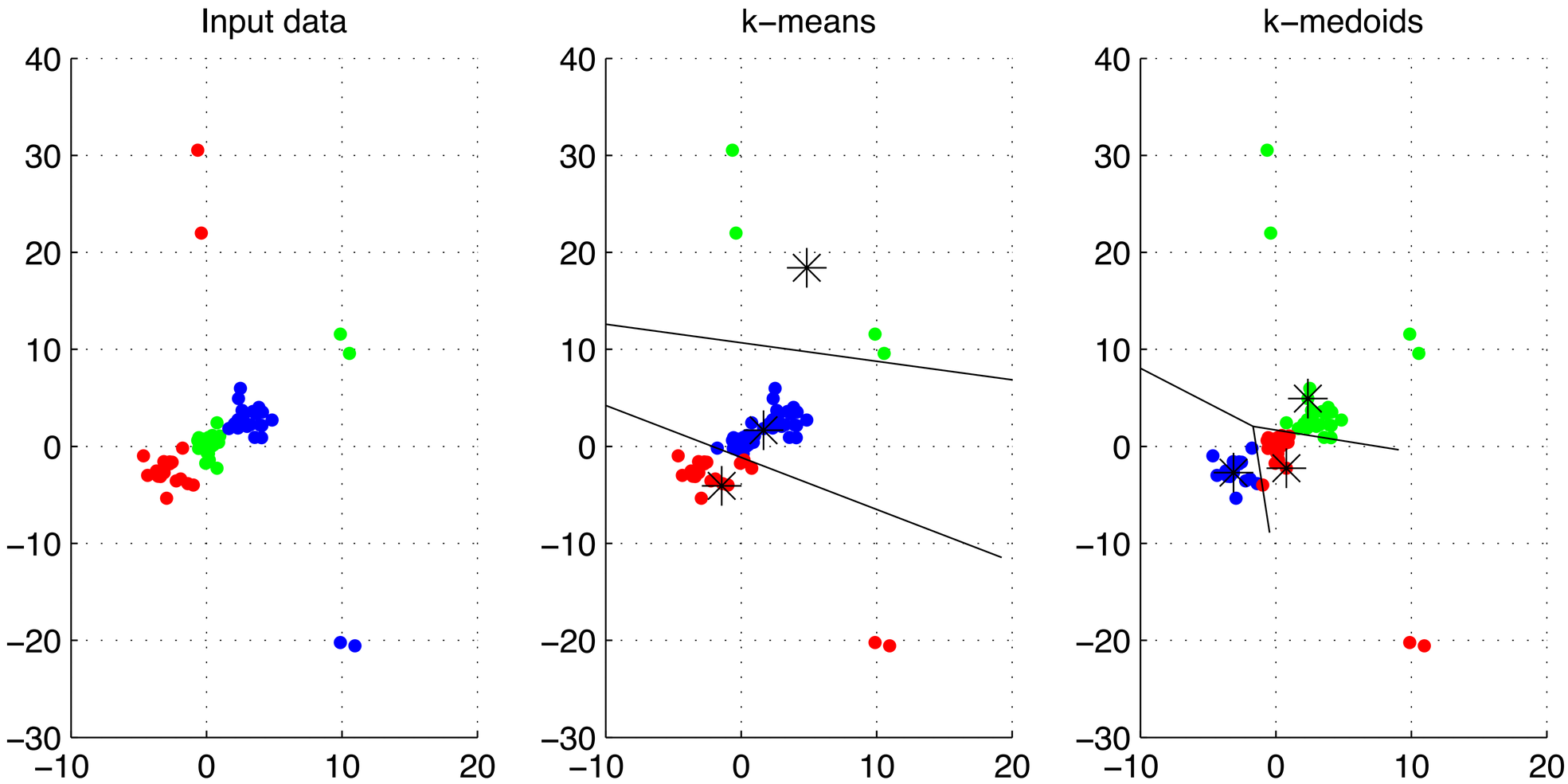


K-medoids

- Medoid:
  - Least dissimilar data point to all others
  - Not as influenced by outliers as the mean
- Replace means with medoids
  - Redesign k-means as k-medoids

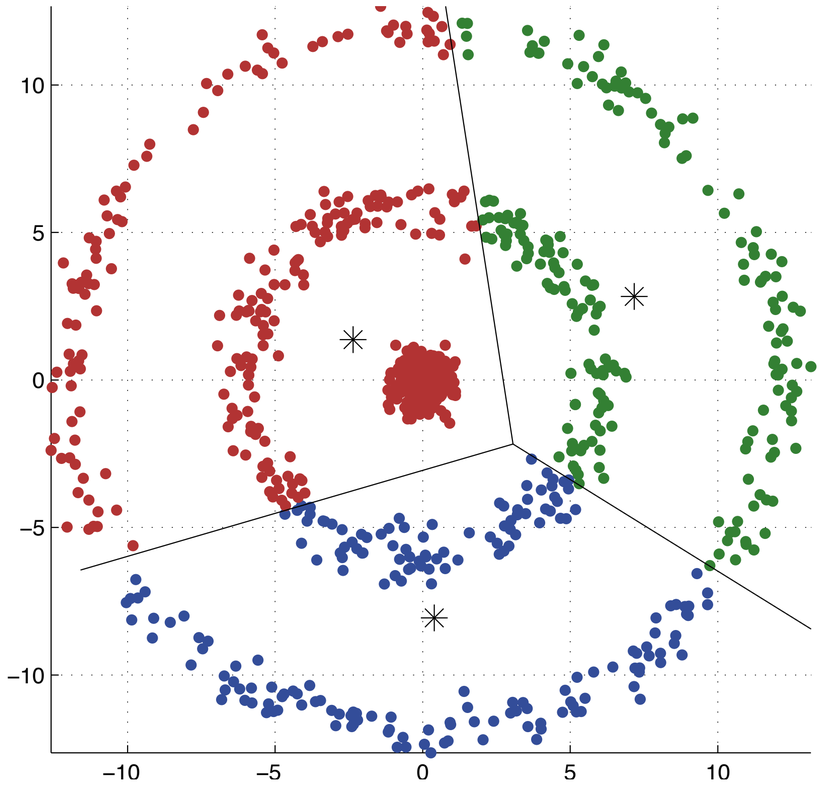


K-medoids



One more problem

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



Solution: Go to another space

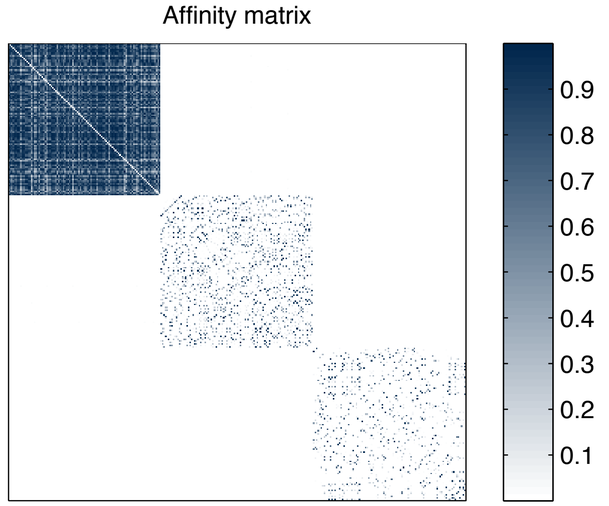
- Spectral clustering approach
  - Define affinity matrix:

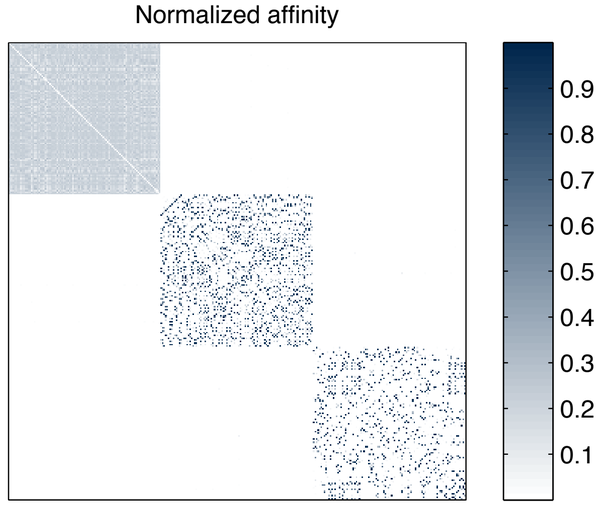
$$\mathbf{A}_{i,j} = \begin{cases} e^{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/\sigma} & if\ i \neq j \\ 0 & if\ i = j \end{cases}$$

  - And "normalize" it:

$$\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$$

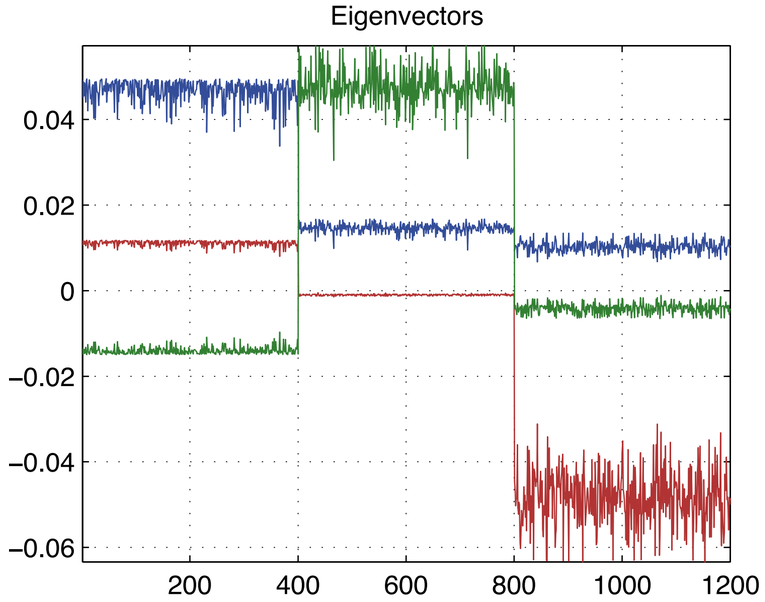
$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}} \cdot \mathbf{A} \cdot \mathbf{D}^{-\frac{1}{2}}$$

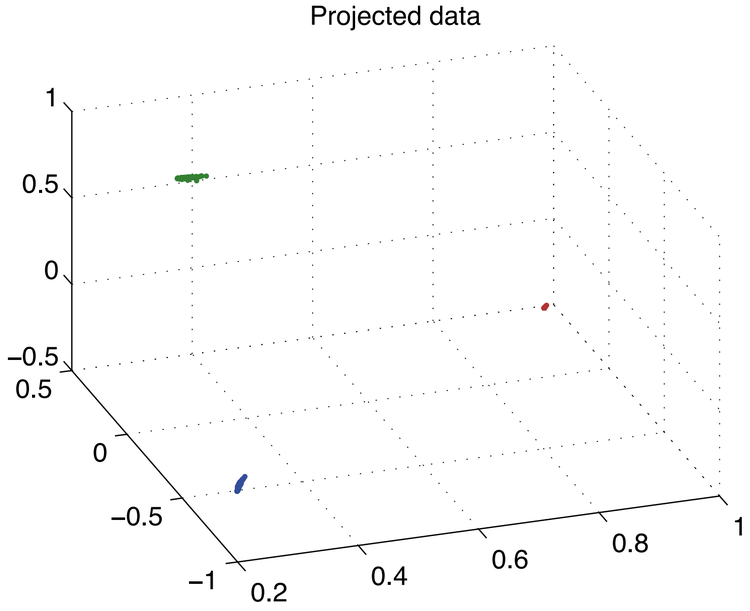




Project and k-means it

- Compute its k largest eigenvectors
  - Which is the same as dimensionality reduction

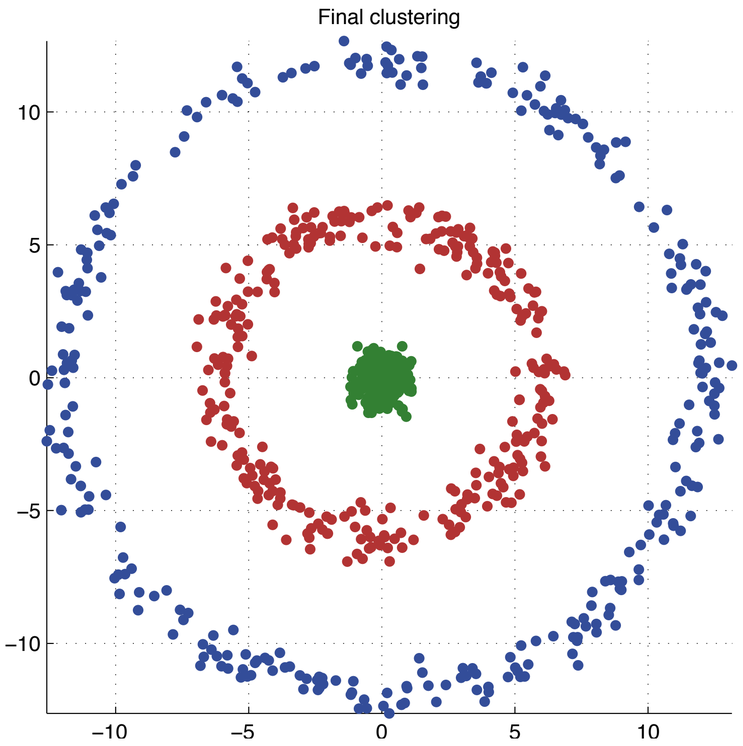




- And perform k-means on the result

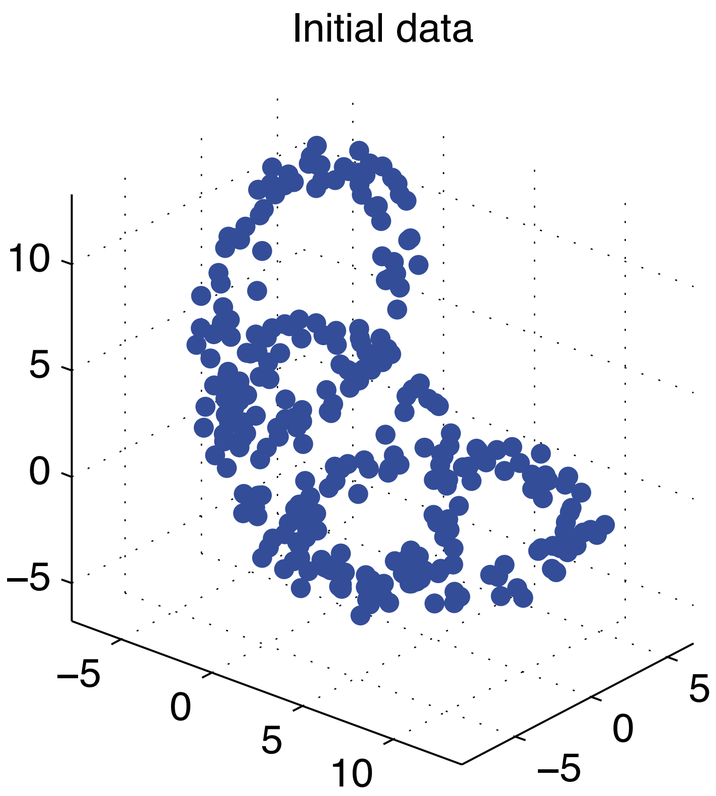
And voilà

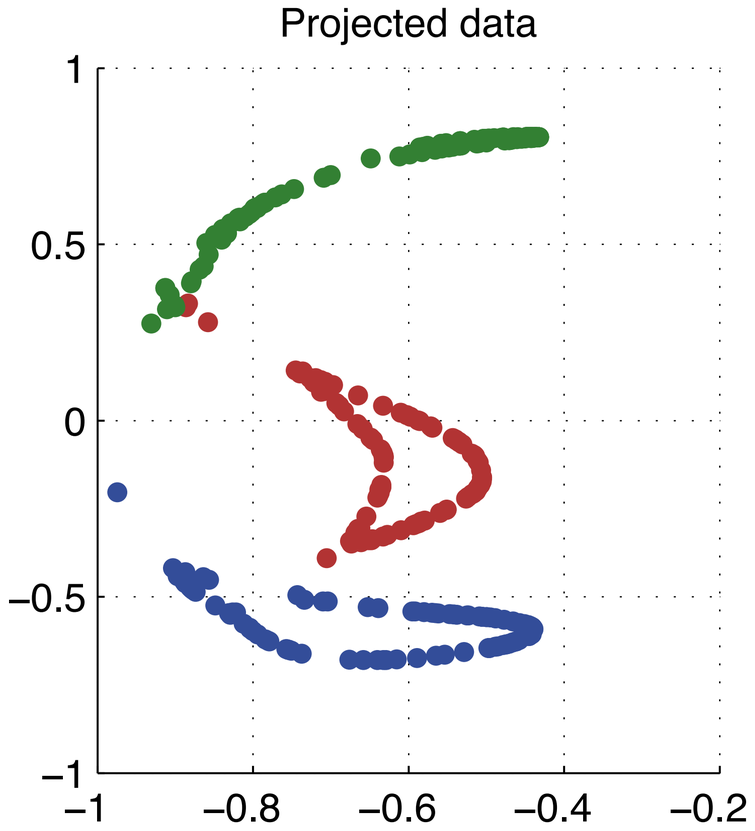
- We can now discover non-compact clusters
- Does this sound familiar?
  - KPCA / Manifolds
- Spectral algorithms
  - Close relationship to graph theory

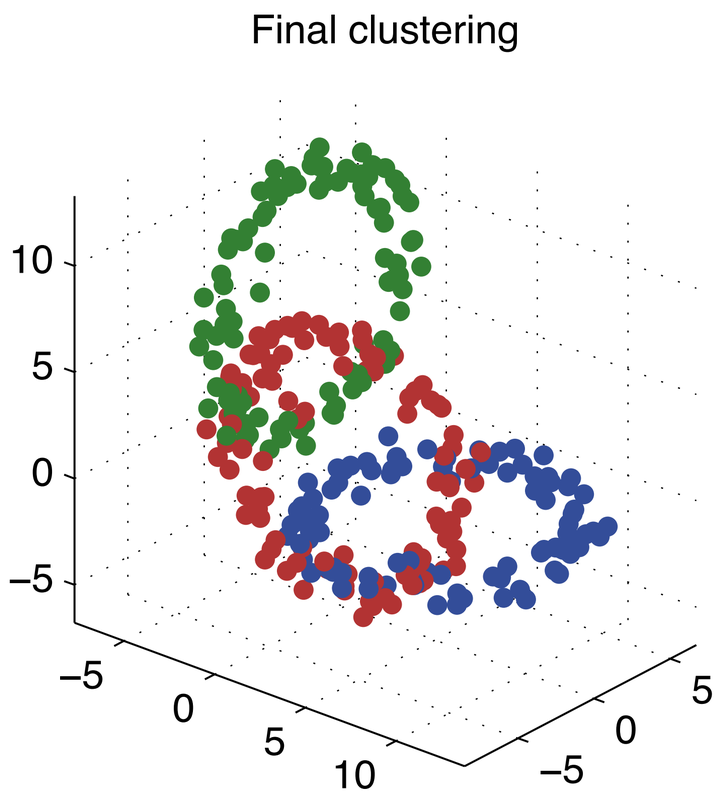


Works with really complex clusters

- 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 avoid!

- Like before we'll "soften up" the model

A new look towards the means

- Instead of the mean vectors use Gaussians

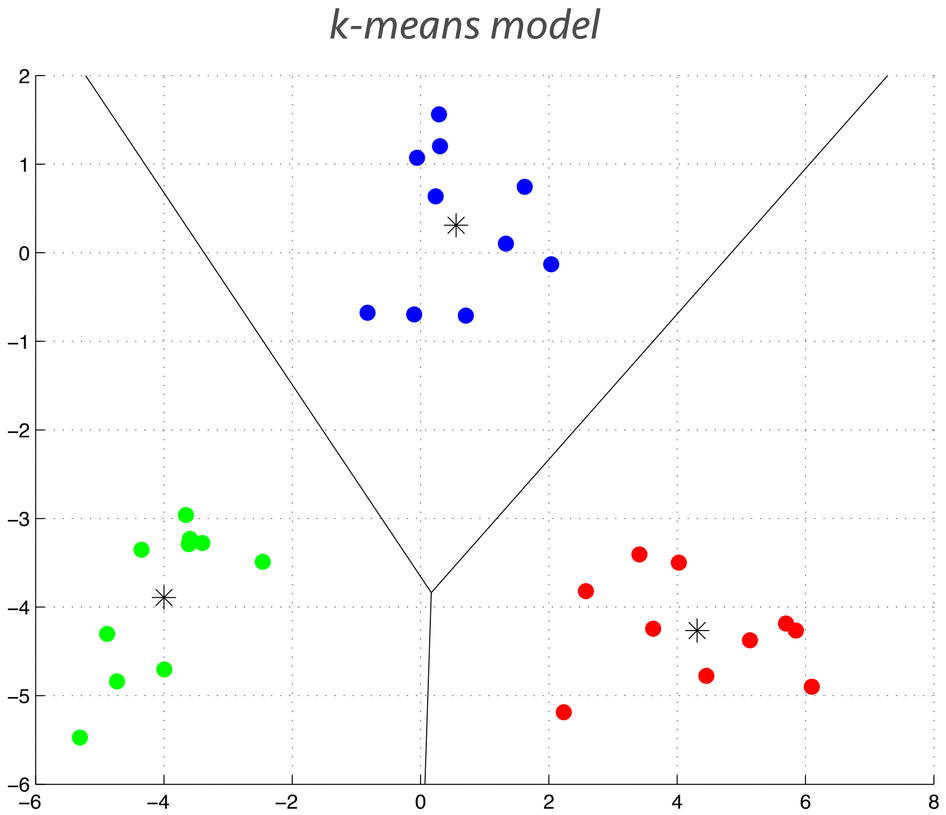
$$d(\mathbf{x},\mu) = \left\| \mathbf{x} - \mu \right\|^2 \rightarrow \frac{1}{\left(2\pi\right)^{d/2} \left|\Sigma\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)}$$

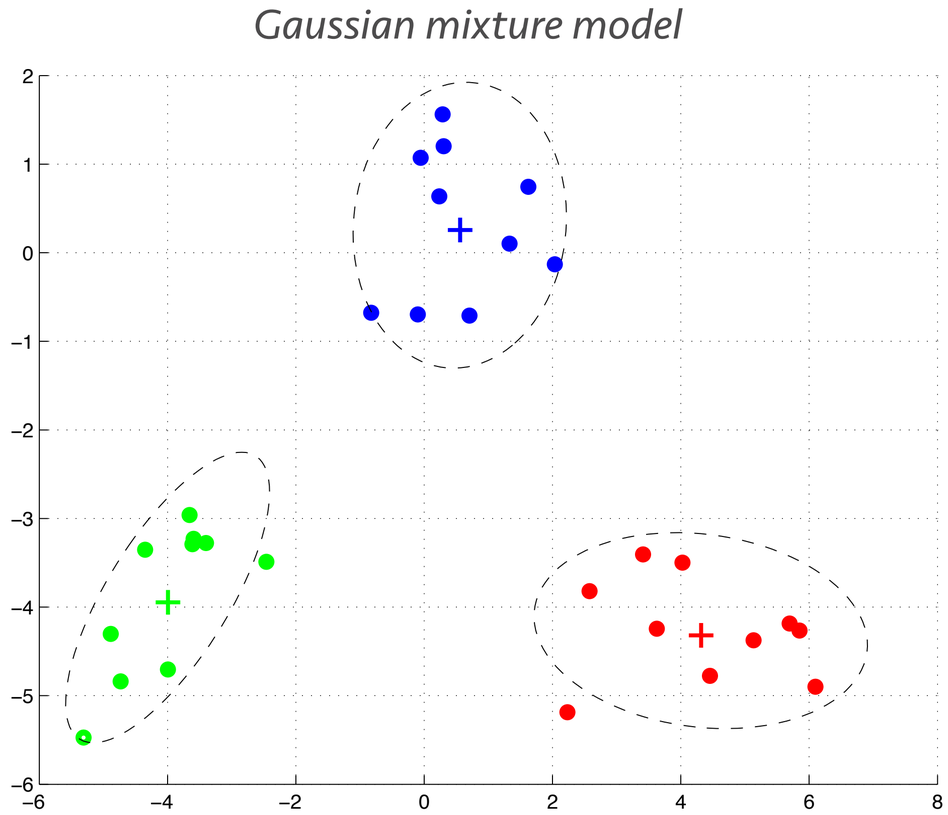
- Instead of a hard assignment use soft assignments

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

A Gaussian Mixture Model

- This effectively defines each cluster as a Gaussian distribution





Formal definition

- Maximize model likelihood given data

$$P(\mathbf{x}) = \sum_k \pi_k \mathcal{N}\left(\mathbf{x} \mid \mu_k, \Sigma_k\right)$$

- 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:
  - Find soft assignment of each component on data
- Step 2:
  - Re-estimate component models using soft assignment as weights on the given data points
- Repeat until convergence
  - Will always increase likelihood towards a local maximum

Expectation-Maximization

- For a latent model: $P(\mathbf{X}, \mathbf{Z} \mid \theta)$
  - Maximize $P(\mathbf{X} \mid \theta)$
- Choose initial parameters θ (may be random, or not)
  - E-step: Evaluate $P(\mathbf{Z} \mid \mathbf{X}, \theta)$
  - M-step: Estimate updated parameters

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

  - Repeat until convergence

EM for GMMs

- Initialize μ_k, Σ_k, π_k
  - Can be random or based on previous clustering results
- E-step
  - Find how much each Gaussian explains every available data point

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

EM for GMMs

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

$$N_k = \sum_n \gamma_{nk}$$

$$\mu_k \leftarrow \frac{1}{N_k} \sum_n \gamma_{nk} \mathbf{x}_n$$

$$\Sigma_k \leftarrow \frac{1}{N_k} \sum_n \gamma_{nk} (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top$$

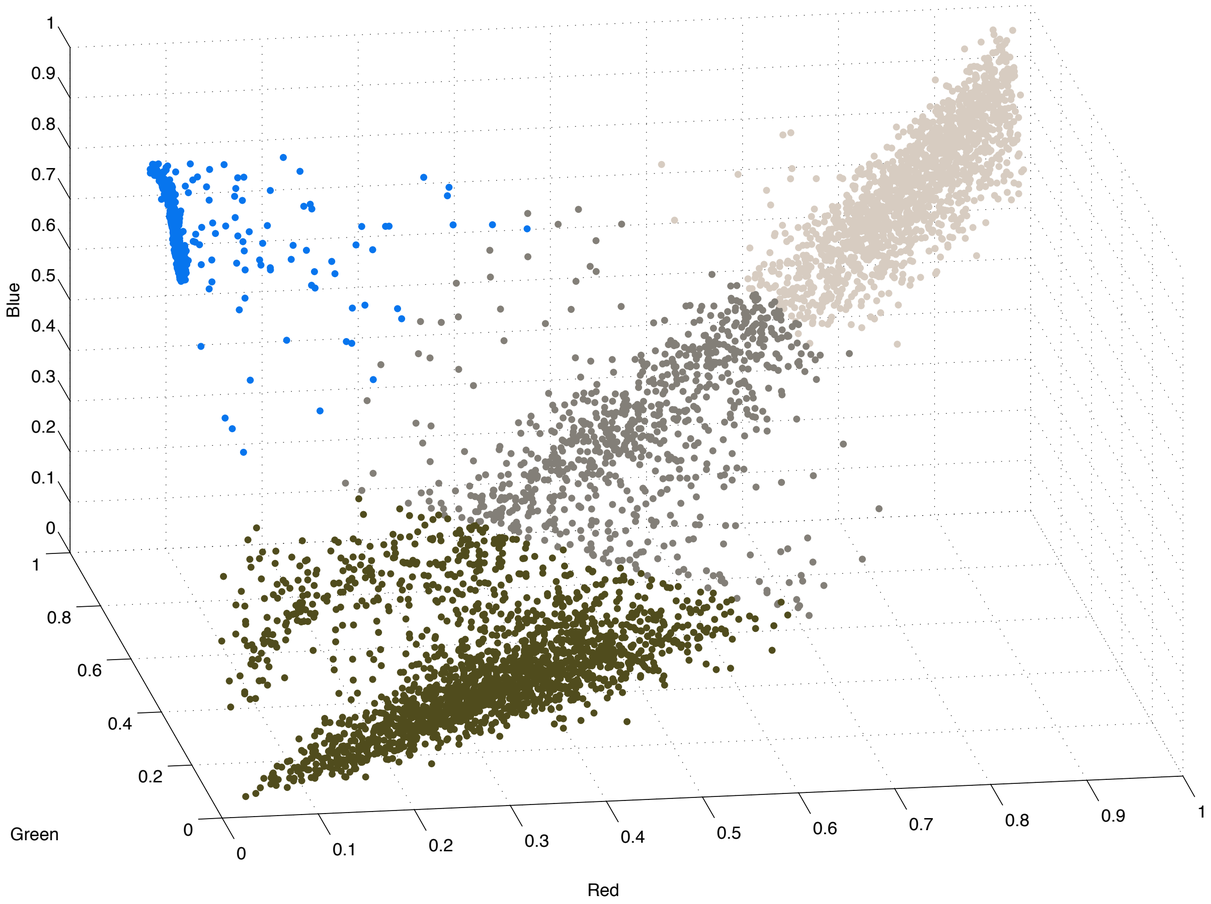
$$\pi_k \leftarrow \frac{N_k}{N}$$

EM for GMMs

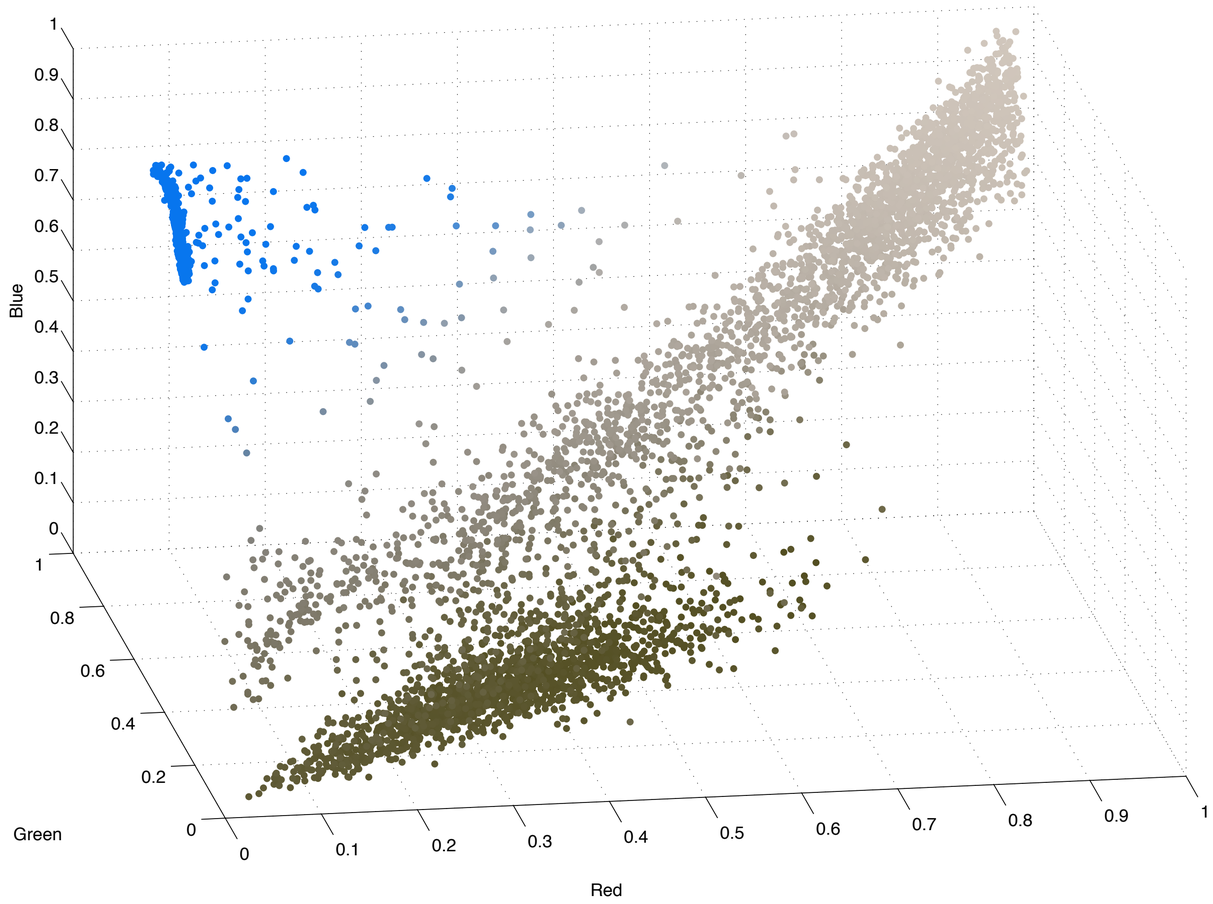
- Alternate between E and M steps until convergence
- Upon conclusion we have the model parameters

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!

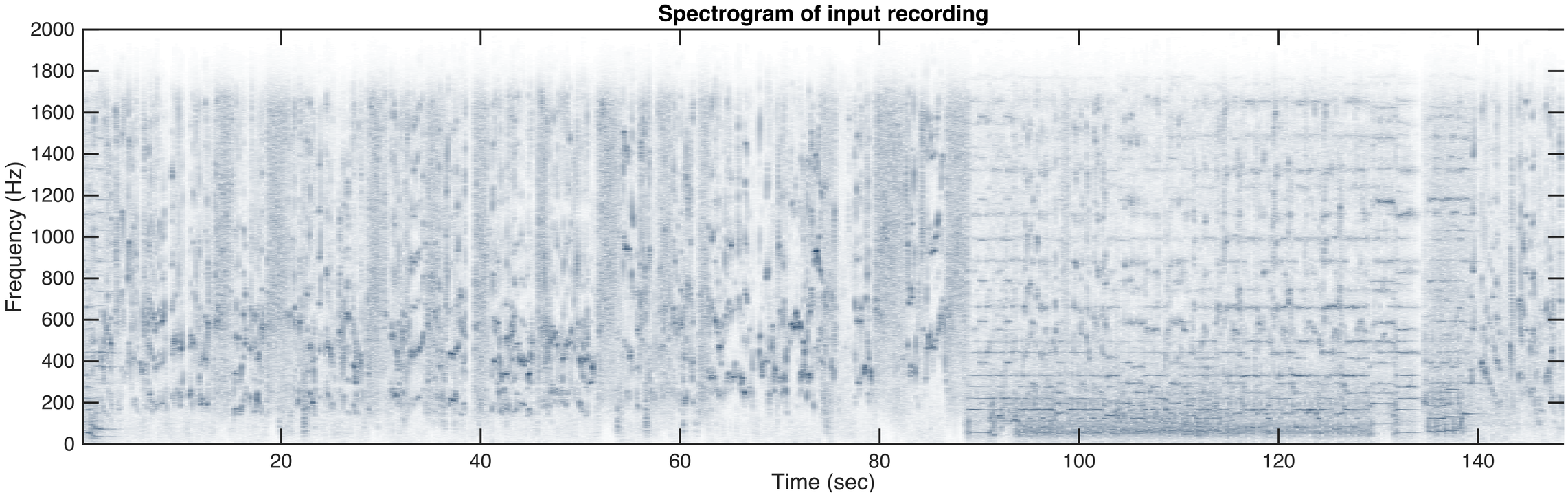
Input

k-means

GMM

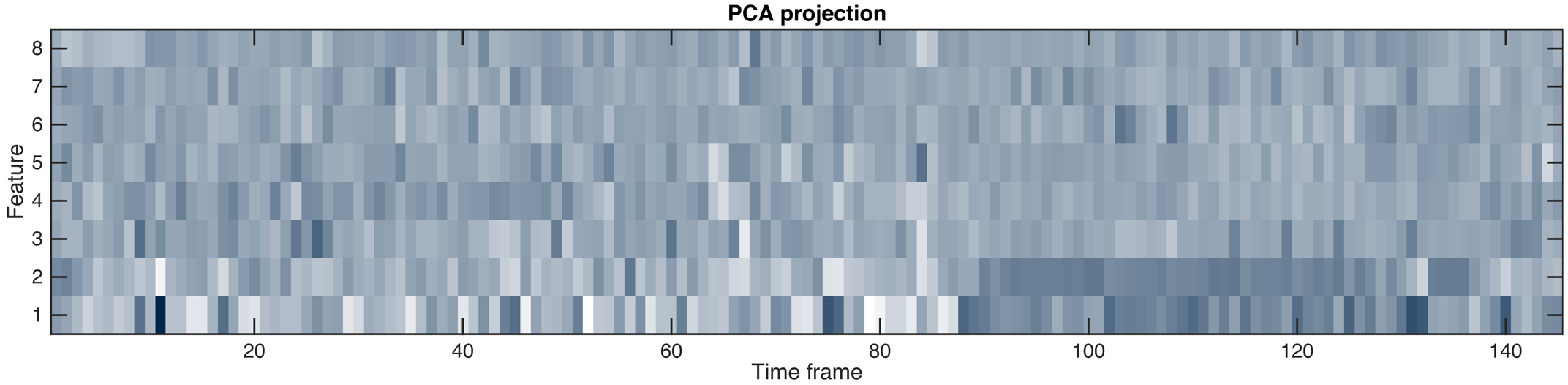
One more example

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



Drop the dimensions

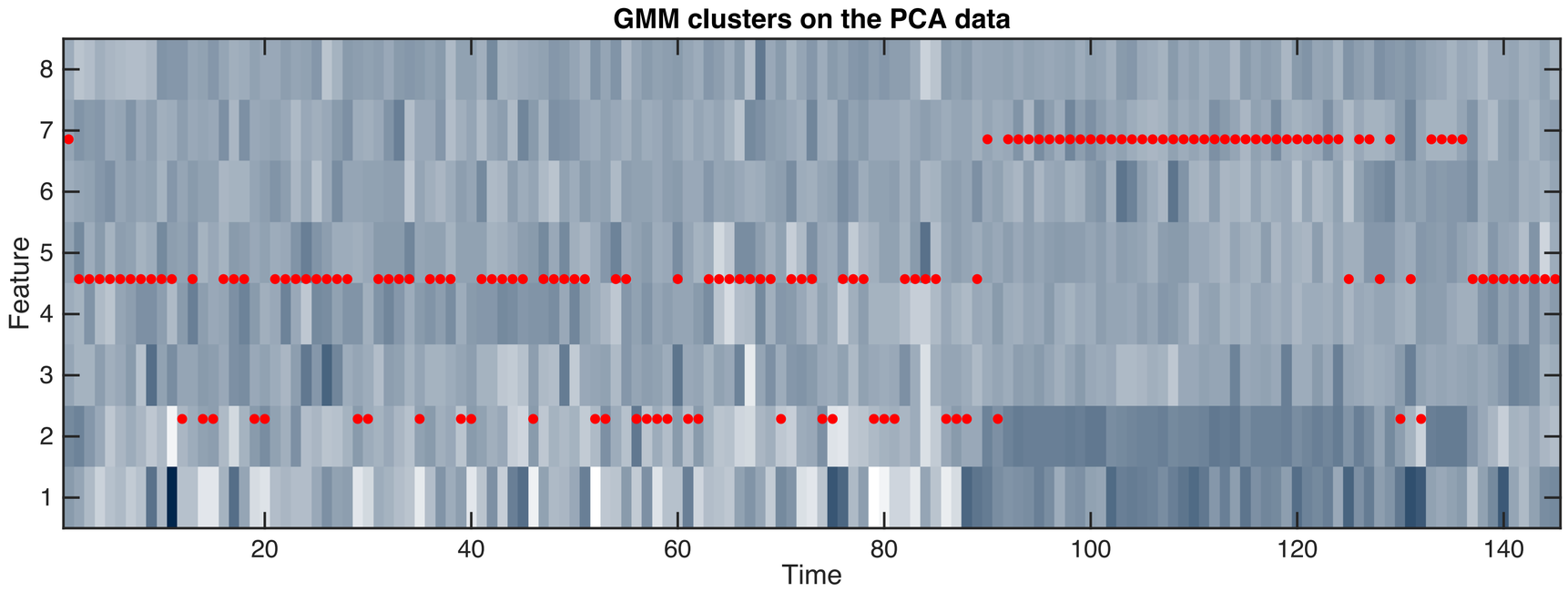
- We can do PCA
  - Dropping from 2049 dimensions down to 8



  - Can you see any clusters?

Performing GMM

- Looking for three clusters
  - Finds frames with similar spectra (in the low 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

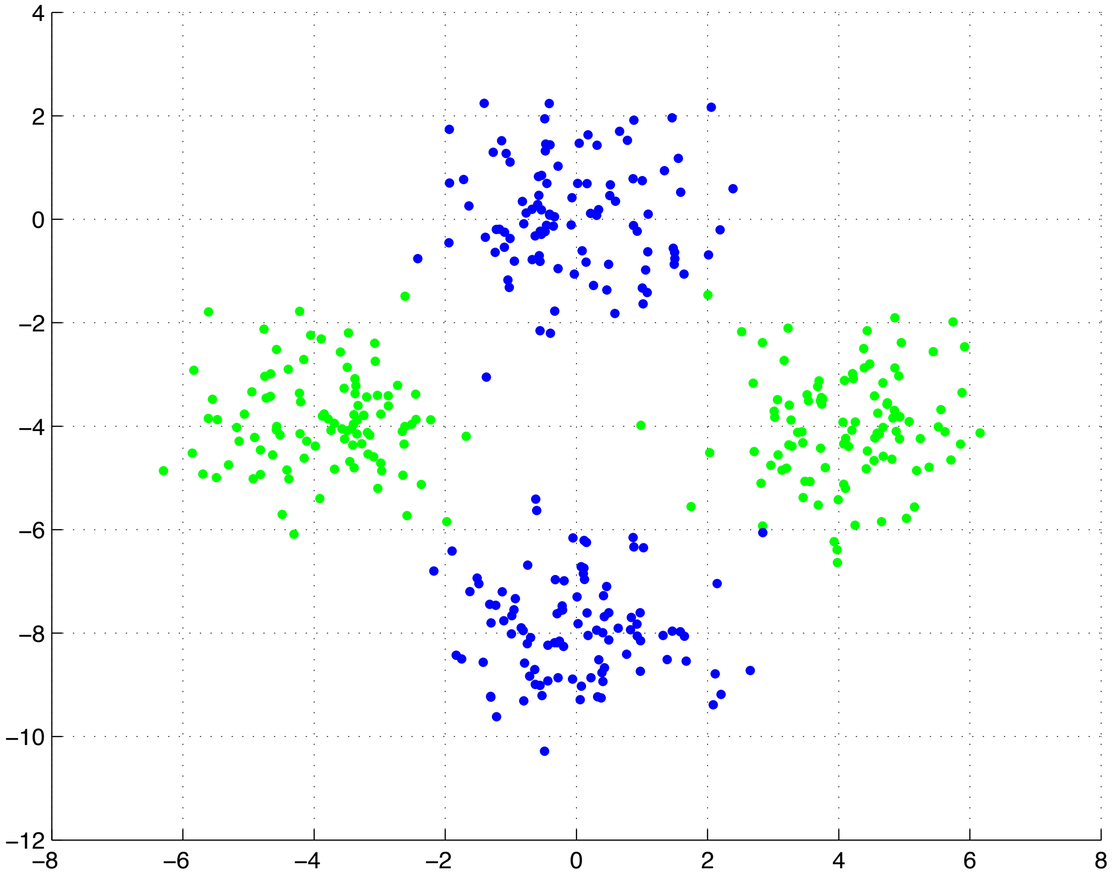
- Initial conditions matter a lot
  - Can initialize means using k-means

GMMs for classification

- GMMs model the distribution of data
- We can model multiple classes separately
  - One GMM per class
- Then evaluate likelihood given GMMs
  - Input belongs to most likely GMM

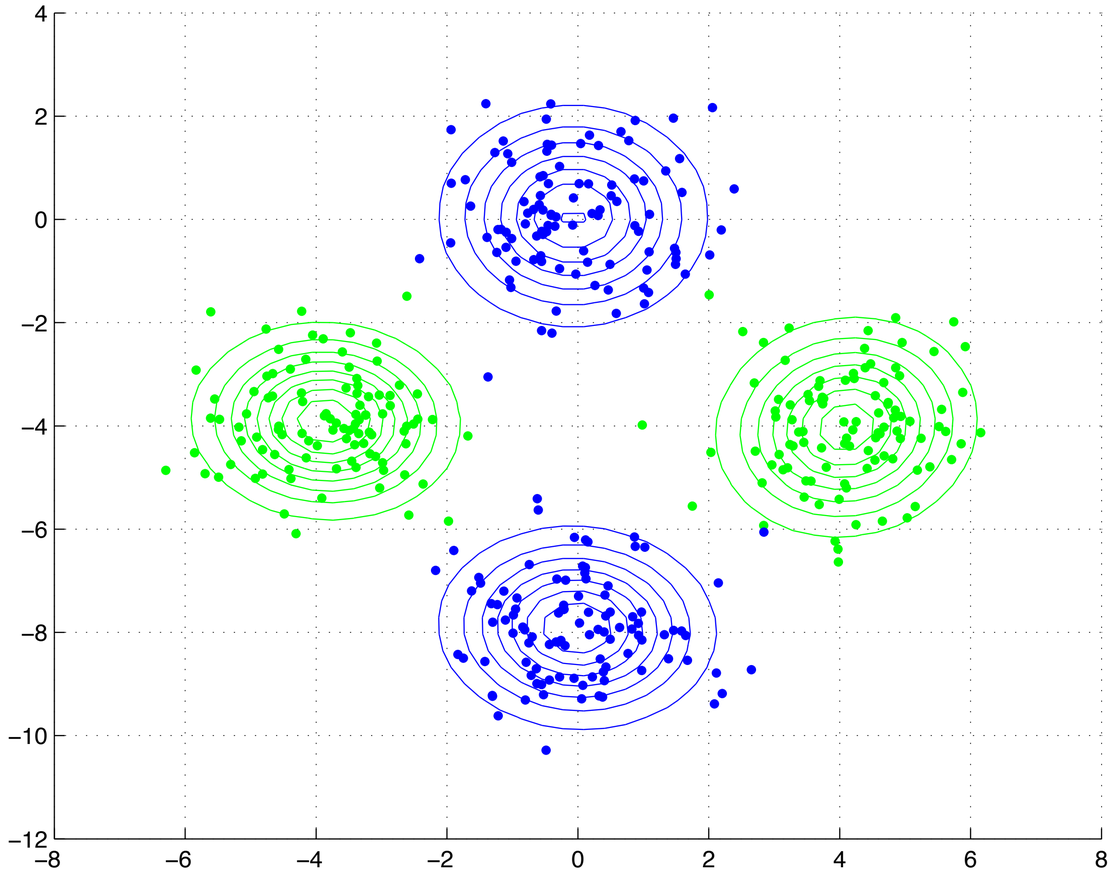
Example case

- Training data



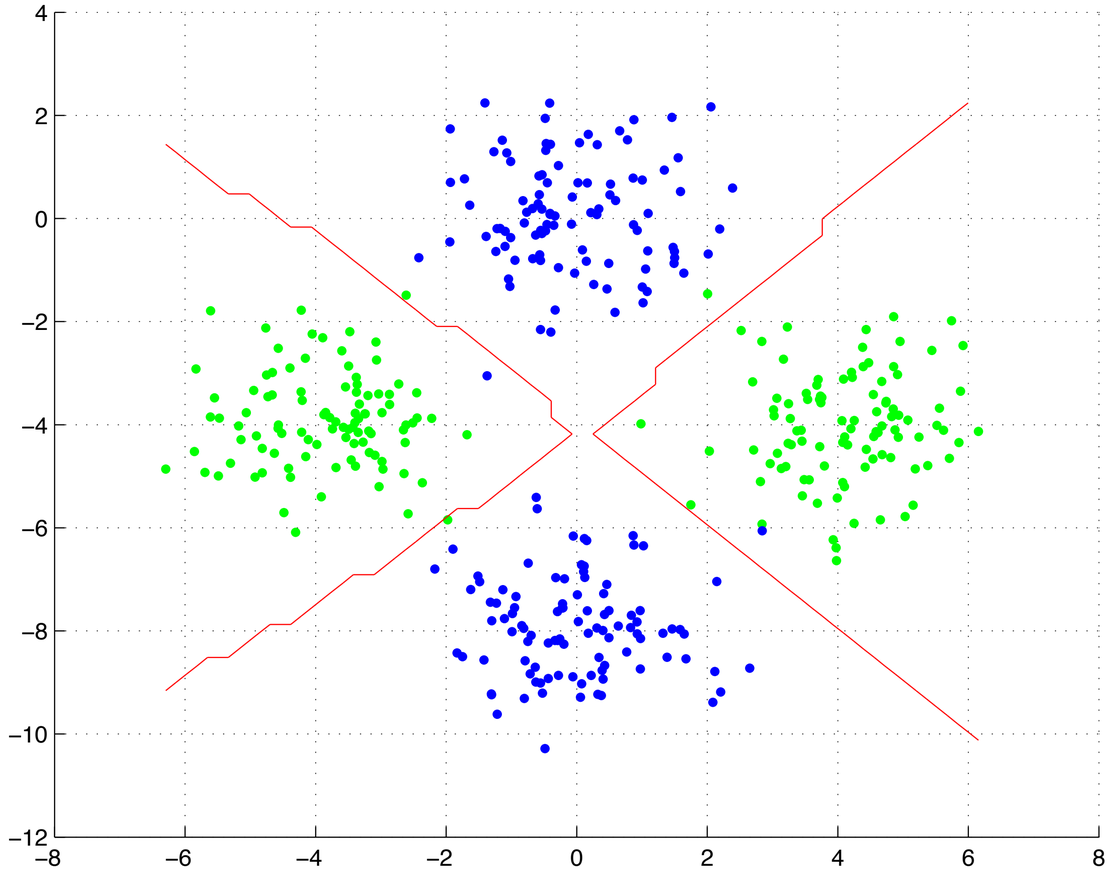
Learned GMMs

- Fitted class Gaussians (2 per class)



Making a classifier

- Implied decision boundaries from Gaussians



Close ties to unsupervised

- PCA/ICA/NMF are latent variable models
  - We can also learn them through EM
- K-means and PCA have a close connection
  - We can perform k-means through PCA
- Features and clusters are similar ideas

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

It's that time again

- The third problem set is out
  - This one is on classification
    - Hint: The problem sets have you write all the code you would need for your final project, thus making your life easier later
- Yes, this one is easier