Clustering

CS498

Today's lecture

- Clustering and unsupervised learning
- Hierarchical clustering
- K-means, K-medoids, VQ

Unsupervised learning

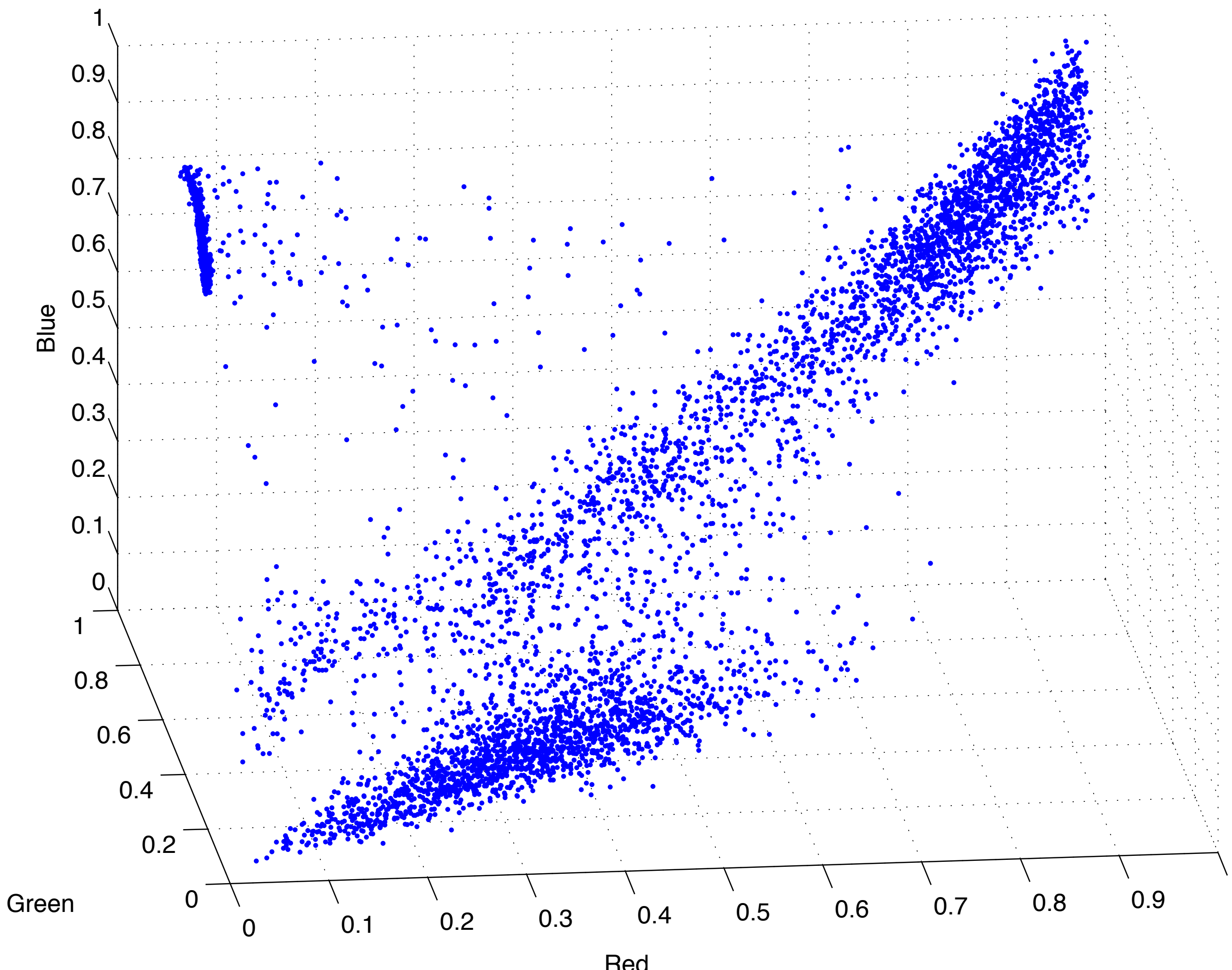
- Supervised learning
  - Use labeled data to do something smart
- What if the labels don't exist?

Some inspiration

El Capitan, Yosemite National Park

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

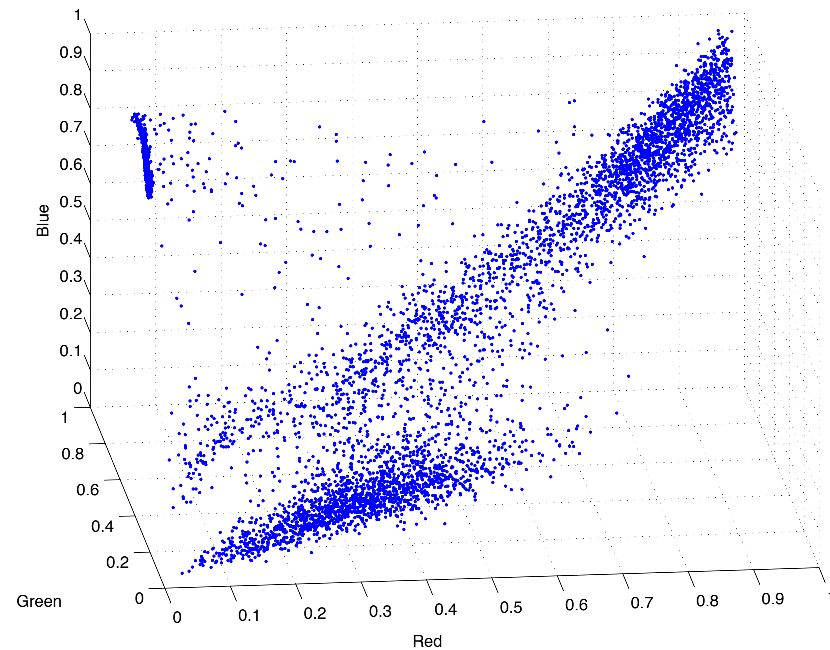
The way we'll see it



ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

A new question

- I see classes, but …
- How do I find them?
  - Can I automate this?
  - How any 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

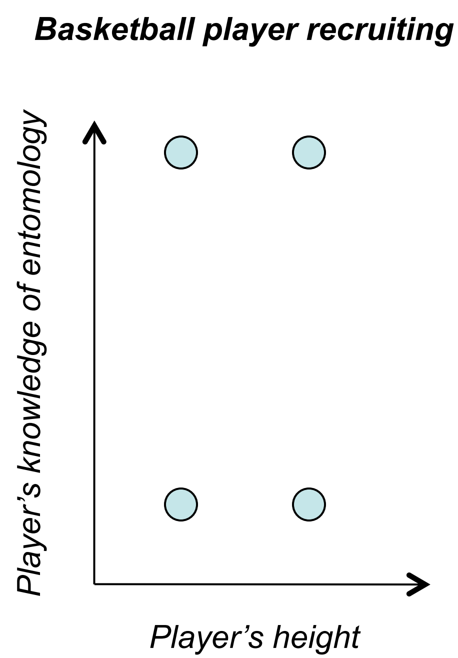
- Describe your data using features
  - What's your objective?
- Define a proximity measure
  - How is the feature space shaped?
- Define a clustering criterion
  - When do samples make a cluster?

Know what you want

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

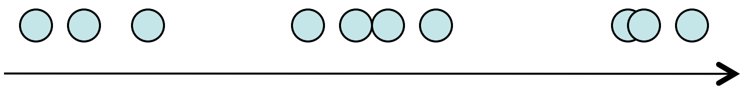
Know what you want

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



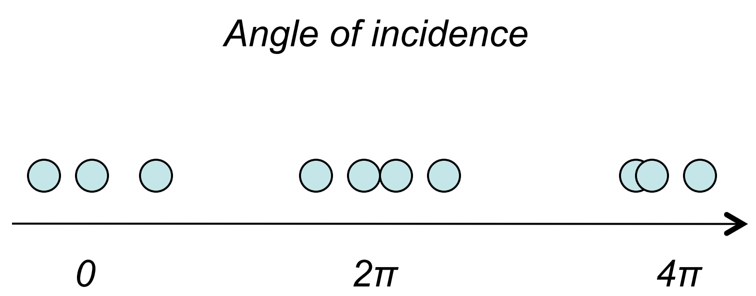
Know your space

- Define a sensible proximity measure



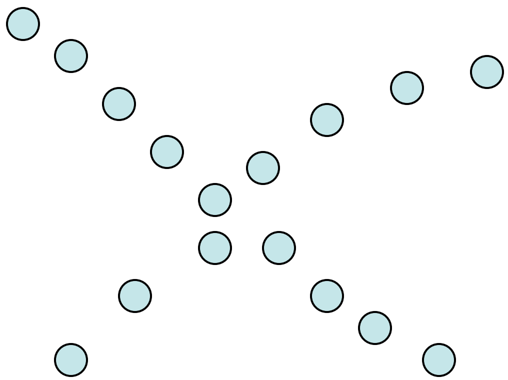
Know your space

- Define a sensible proximity measure



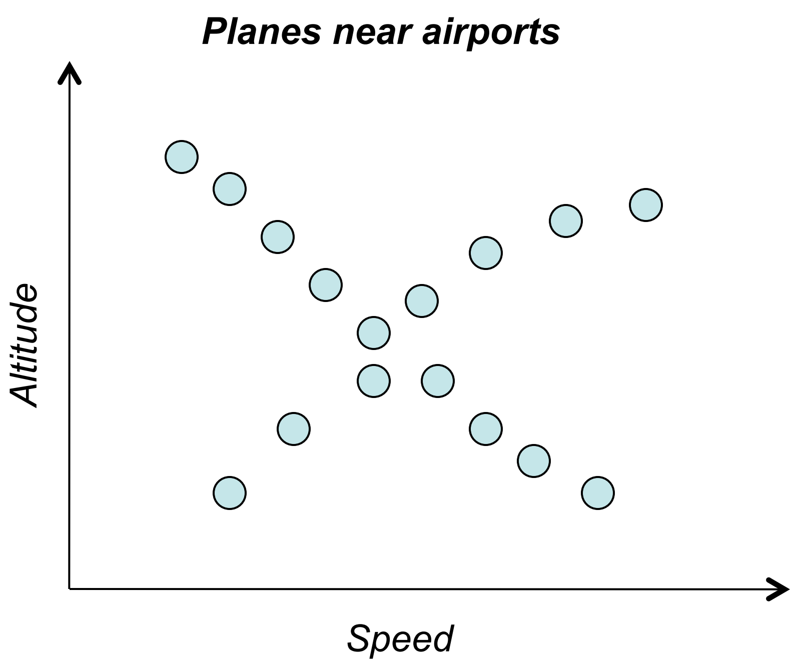
Know your cluster type

- What forms a cluster in your space?



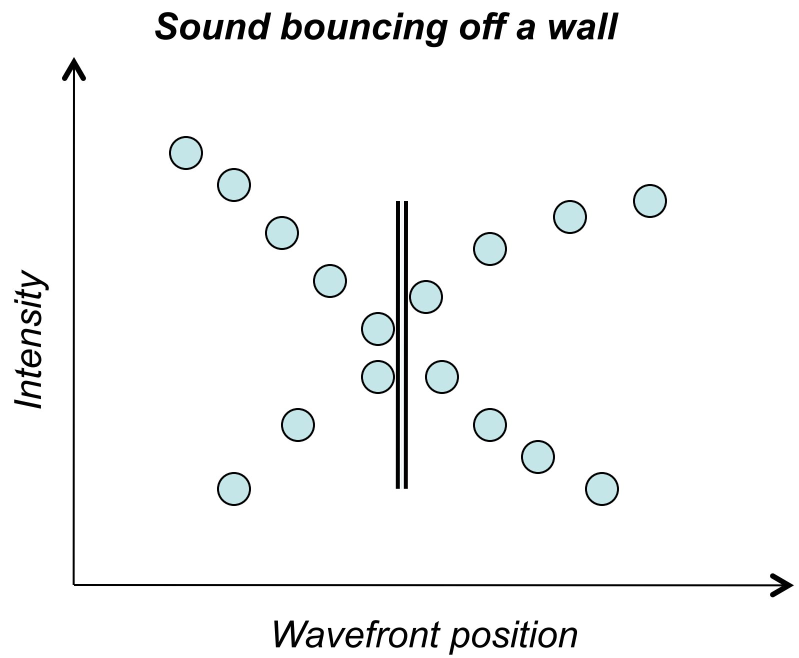
Know your cluster type

- What forms a cluster in your space?



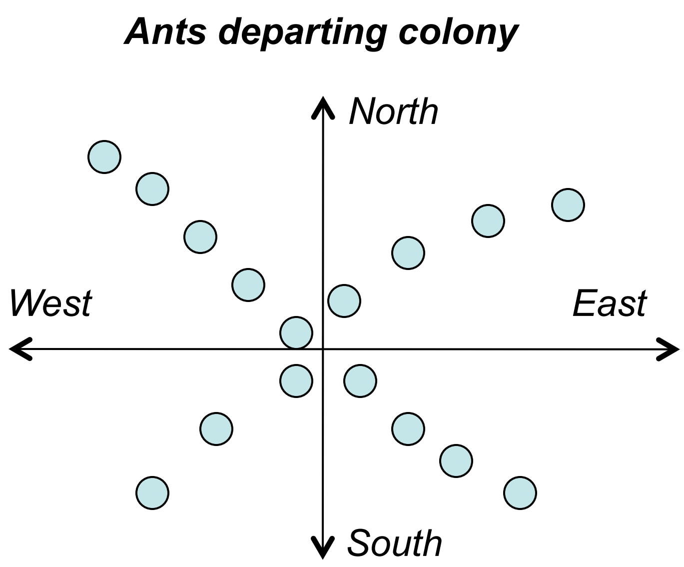
Know your cluster type

- What forms a cluster in your space?



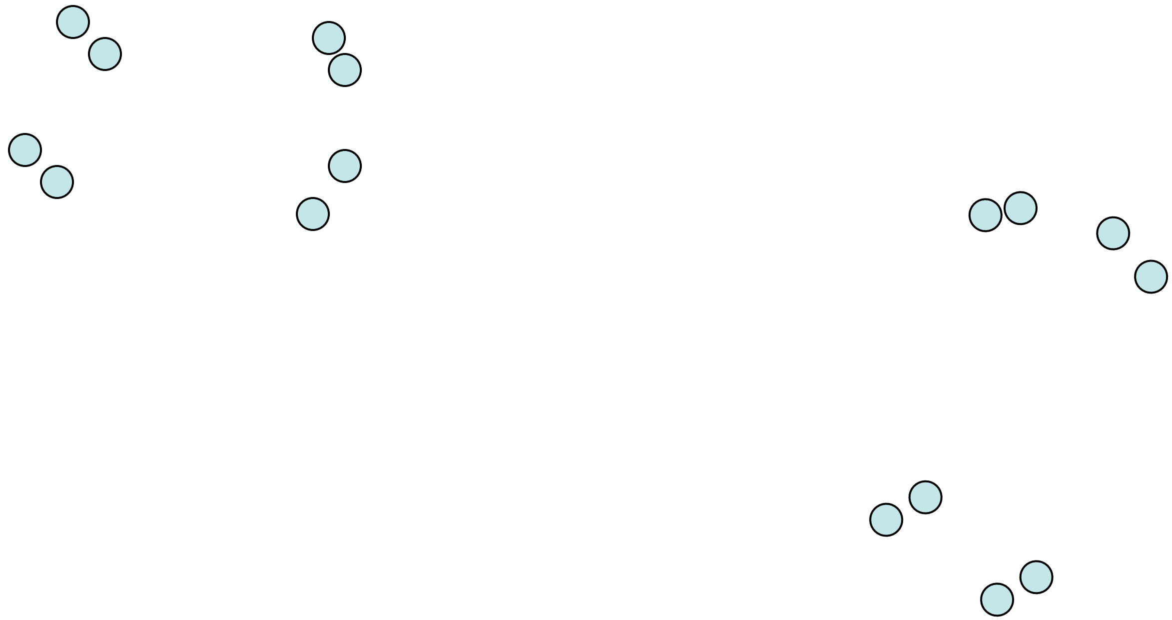
Know your cluster type

- 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

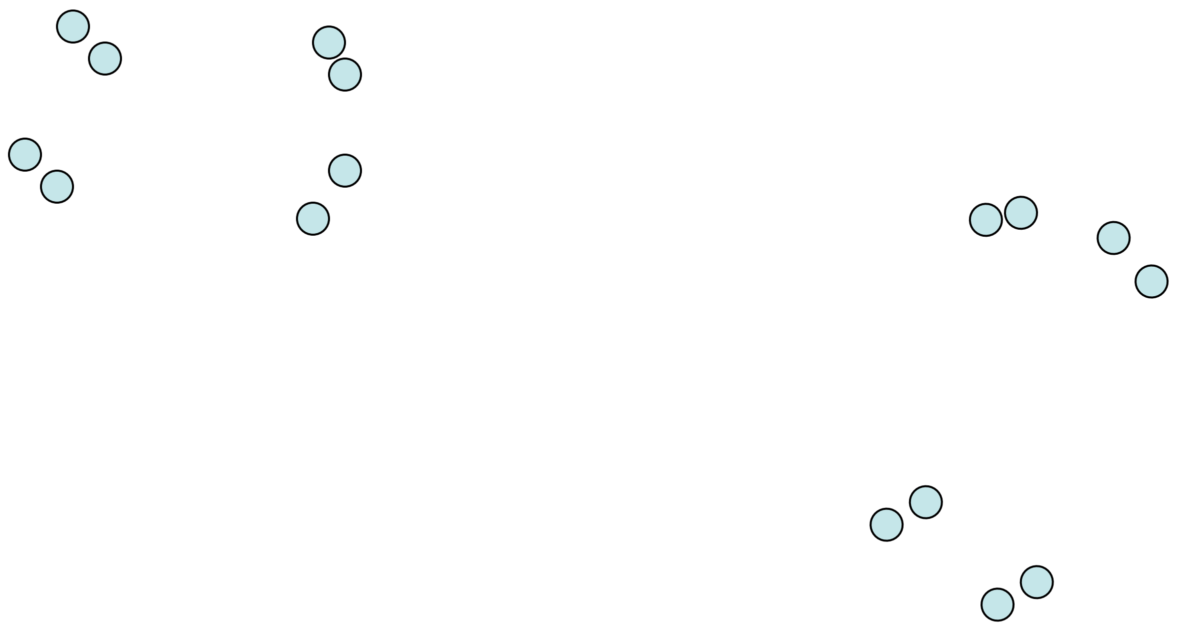
- Tons of methods
- We can use step-based logical steps
  - e.g., find two closest point and merge, repeat
- Or formulate a global criterion

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:

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

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

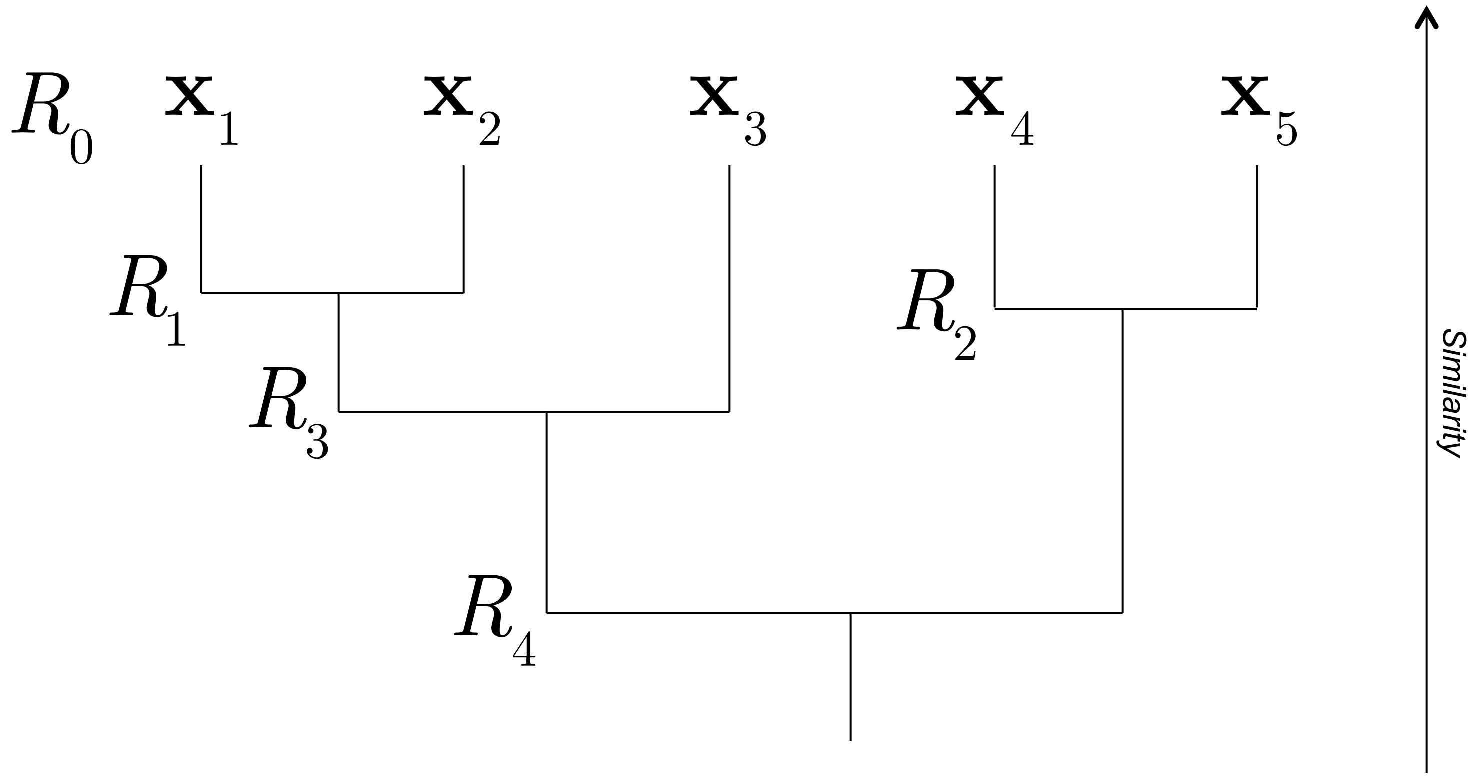
- Define a distance measure: $d(C_j, C_i)$

Agglomerative clustering

- Choose: $R_0 = \{C_i = \{\mathbf{x}_i\}, i = 1, \ldots, N\}$
- For $t = 1, \ldots$
  - Among all clusters in $R_{t\text{-}1}$, find cluster pair $\{C_i, C_j\}$ such that:
$$\arg\min_{i,j} d(C_i, C_j)$$
  - Form new cluster and replace pair:
$$C_q = C_i \cup C_j$$
$$R_t = (R_{t-1} - \{C_i, C_j\}) \cup \{\mathrm{C}_q\}$$
- Until we have only one cluster

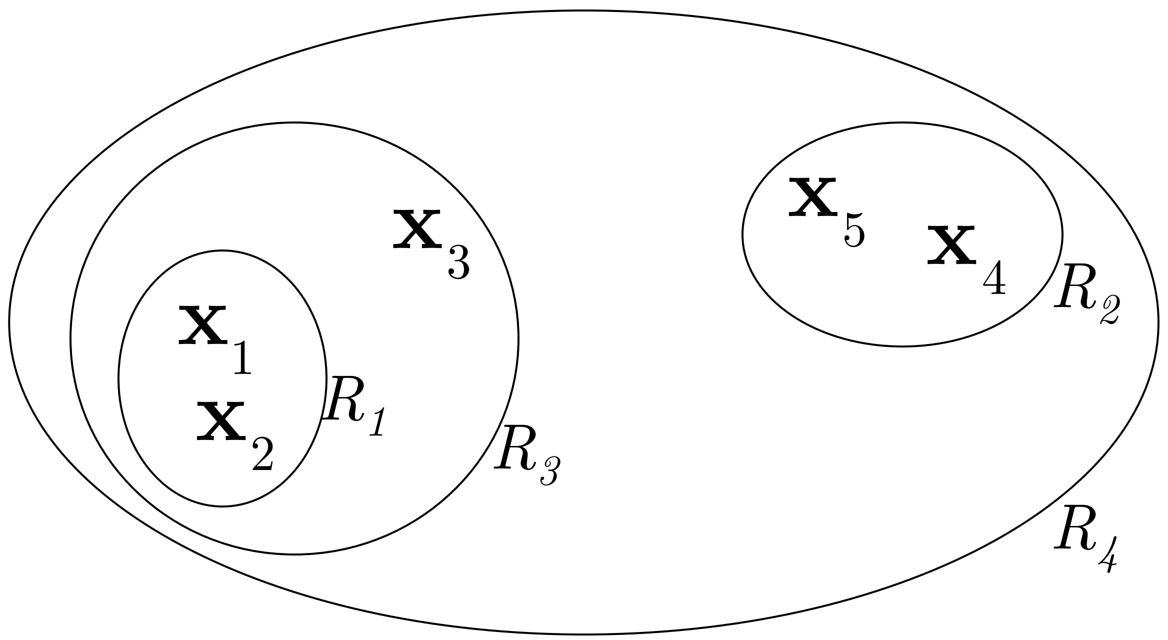
Pretty picture version

- Dendrogram



Pretty picture two

- Venn diagram



Cluster distance?

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

What's involved

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

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

- Overall comparisons:

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

Not good for our case

- 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 of all:

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

  - New clustering is now:

$$R_t = (R_{t-1} - \{C_t\}) \cup \{C_{t,i}, C_{t,j}\}$$

- 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 cost functions

- Given a set of data $\mathbf{x}_i$
- 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
  - $\mathbf{U} \in \{0,1\}$ 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 $\theta, \mathbf{U}$
  - Fix one and find the other, repeat flip case
  - Iterate until happy

Overall process

- Initialize θ and iterate:
  - Estimate $\mathbf{U}$

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

  - Estimate θ

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

  - Repeat until satisfied

K-means

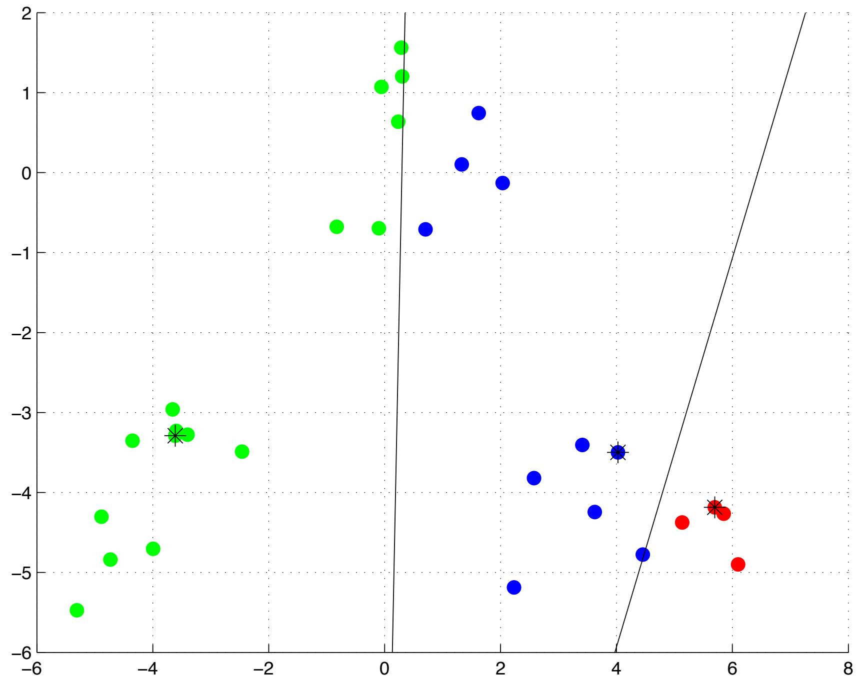
- Standard and super-popular algorithm
- Finds clusters in terms of region centers
- Optimizes squared Euclidean distance

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

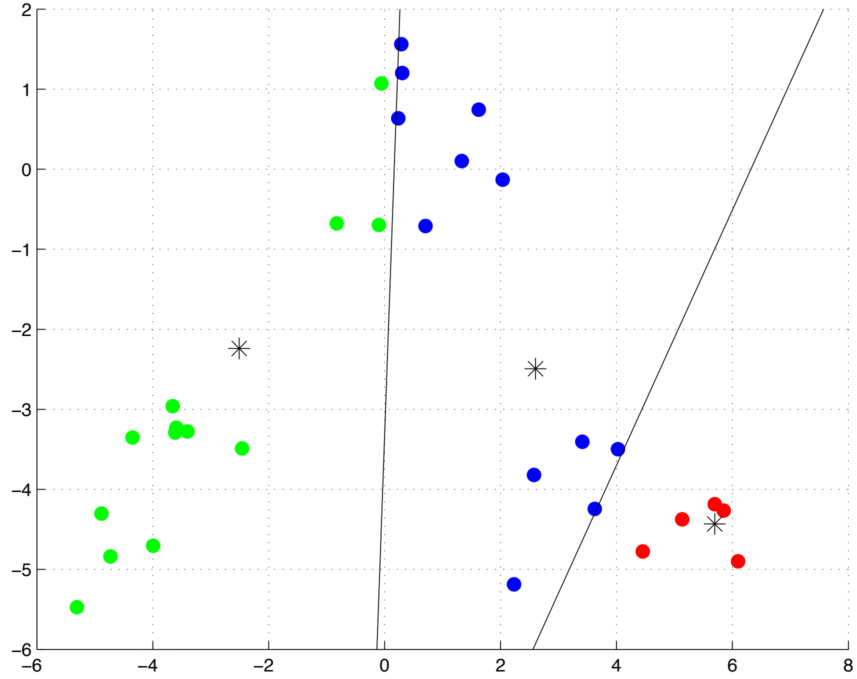
K-means algorithm

- Initialize k means $\boldsymbol{\mu}$
- Iterate
  - Assign samples $\mathbf{x}_i$ to closest mean $\boldsymbol{\mu}$
  - Estimate $\boldsymbol{\mu}$ from assigned samples $\mathbf{x}_i$
- Repeat until convergence

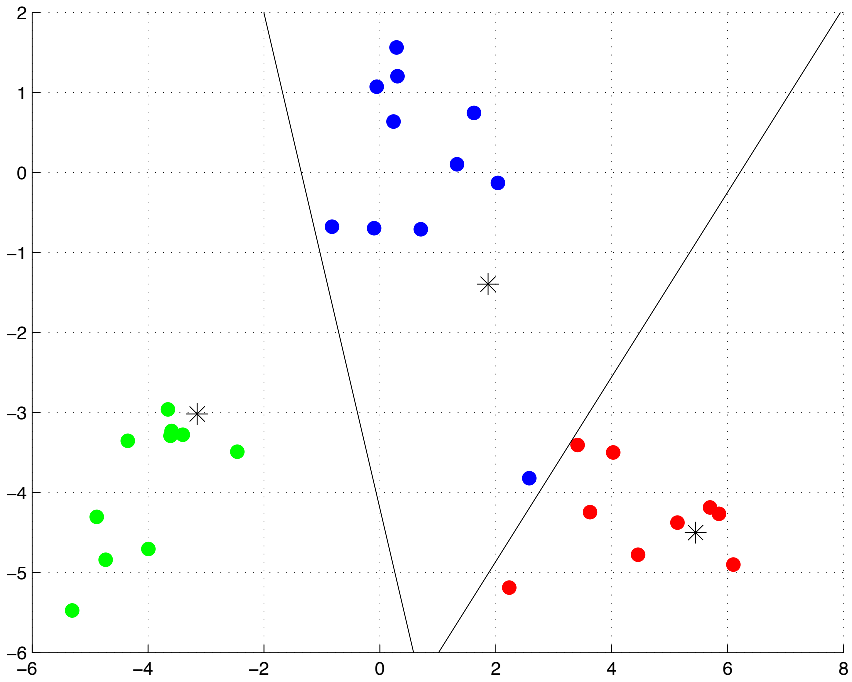
Example run – step 1



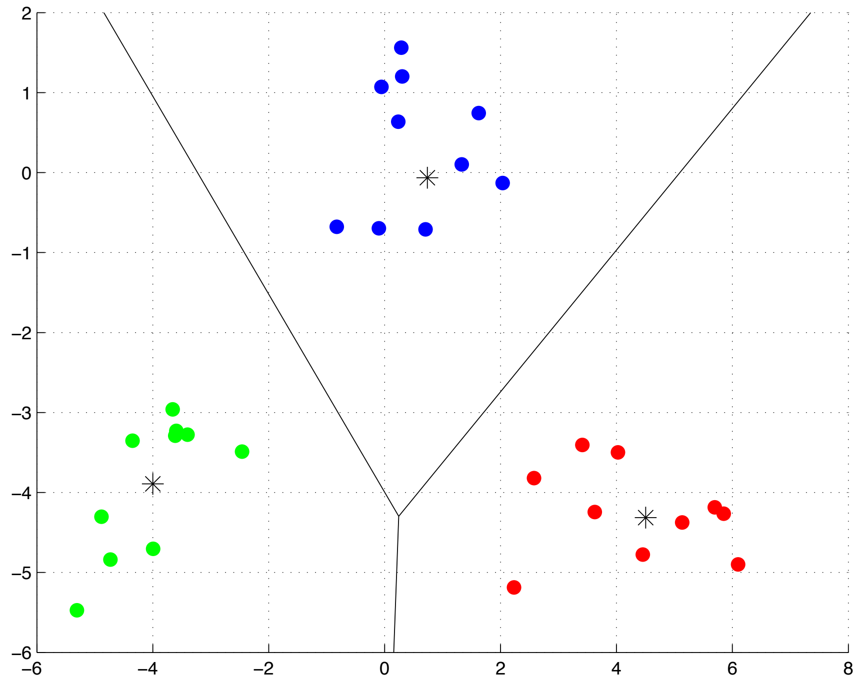
Example run – step 2



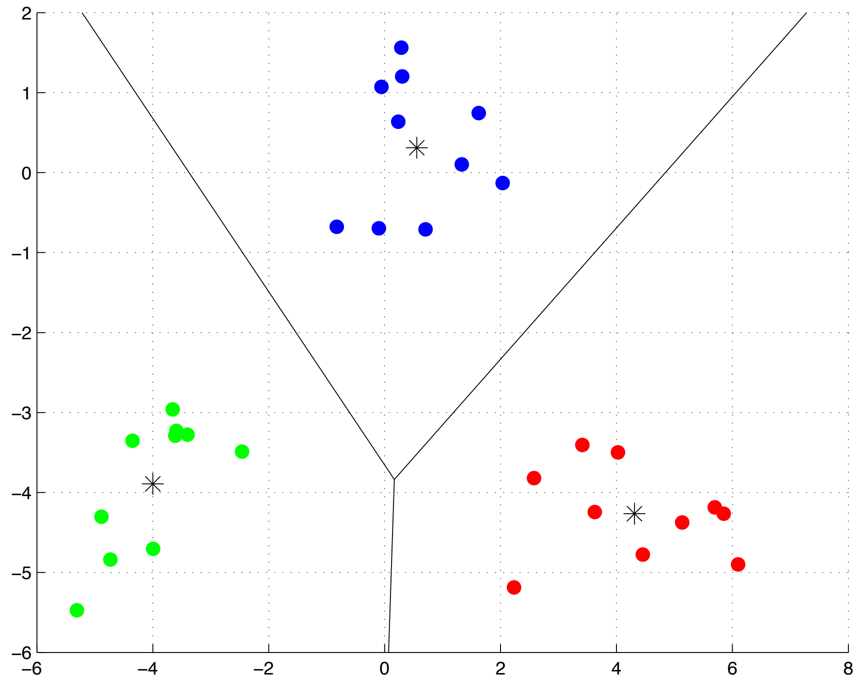
Example run – step 3



Example run – step 4



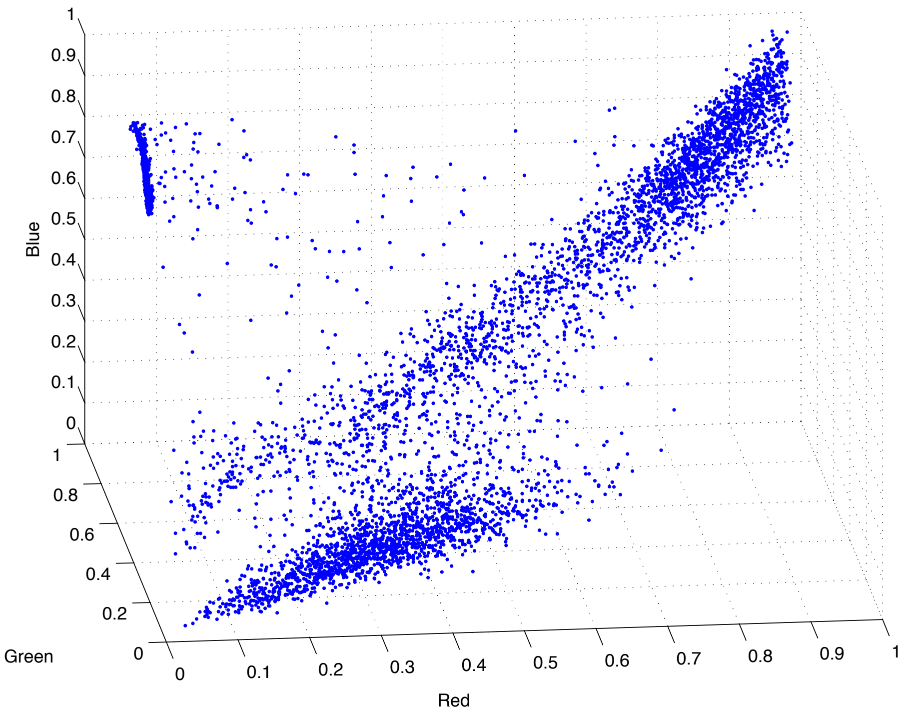
Example run – step 5

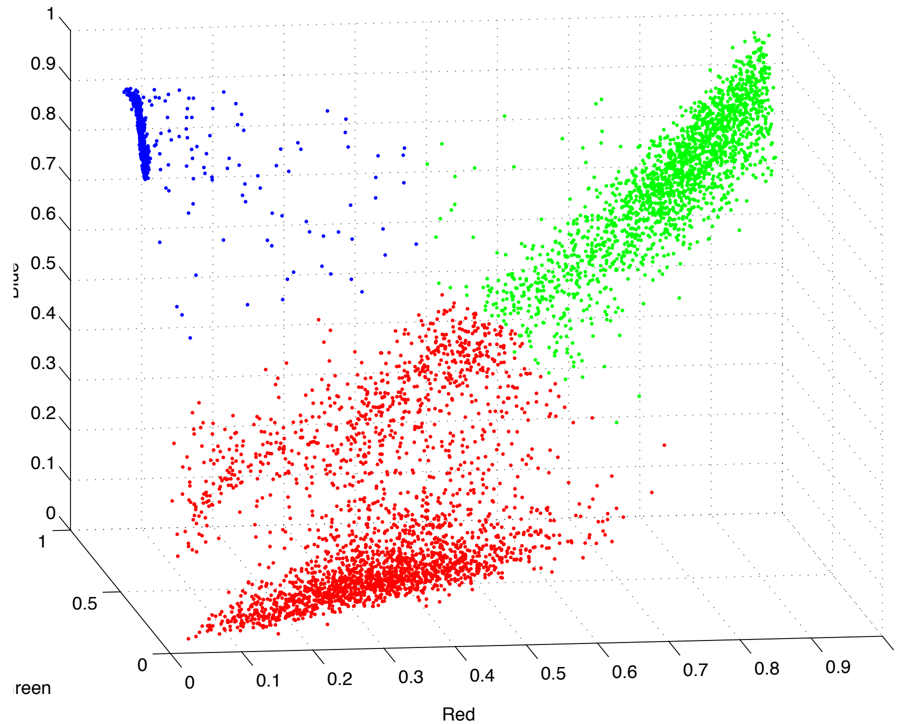


How well does it work?

- Converges to a minimum of cost function
  - Not for all distances though!
- Is heavily biased by starting positions
  - Various initialization tricks
- It’s pretty fast!

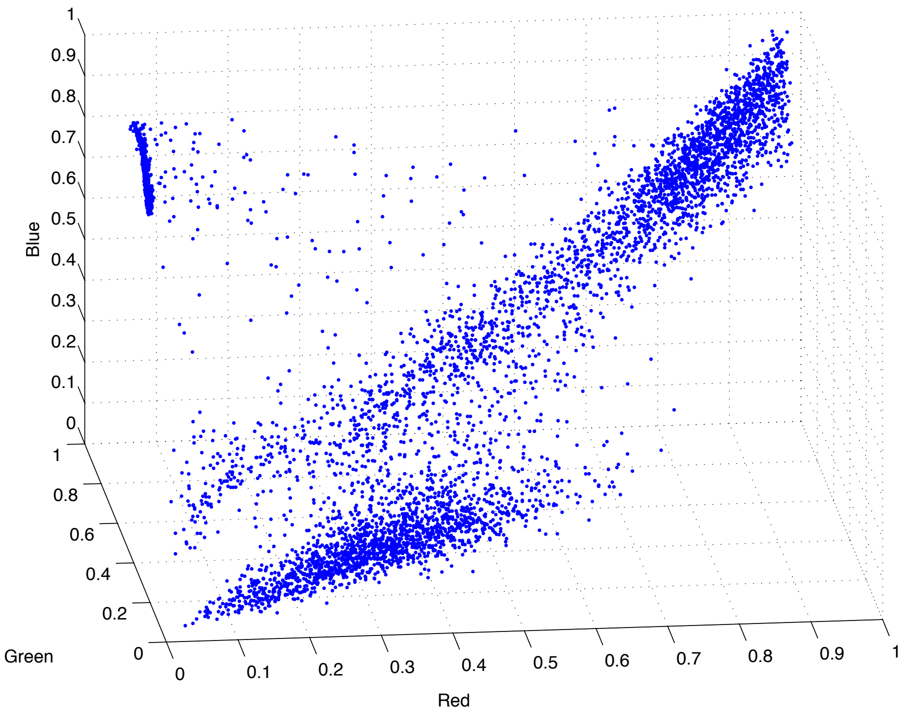
K-Means on El Capitan

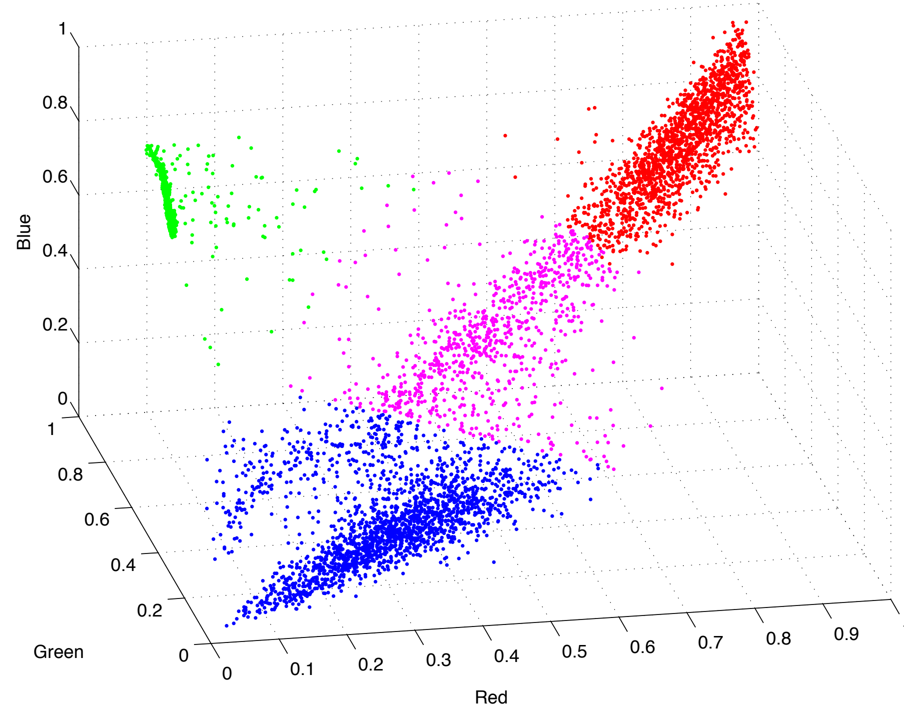




K-means on El Capitan

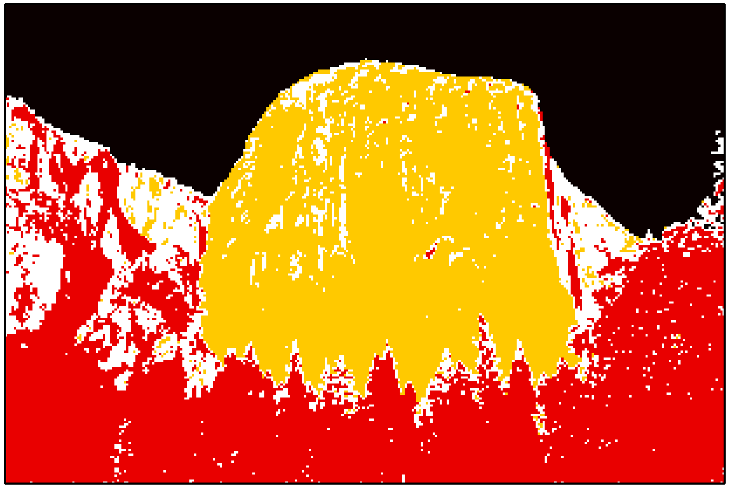
K-means on El Capitan





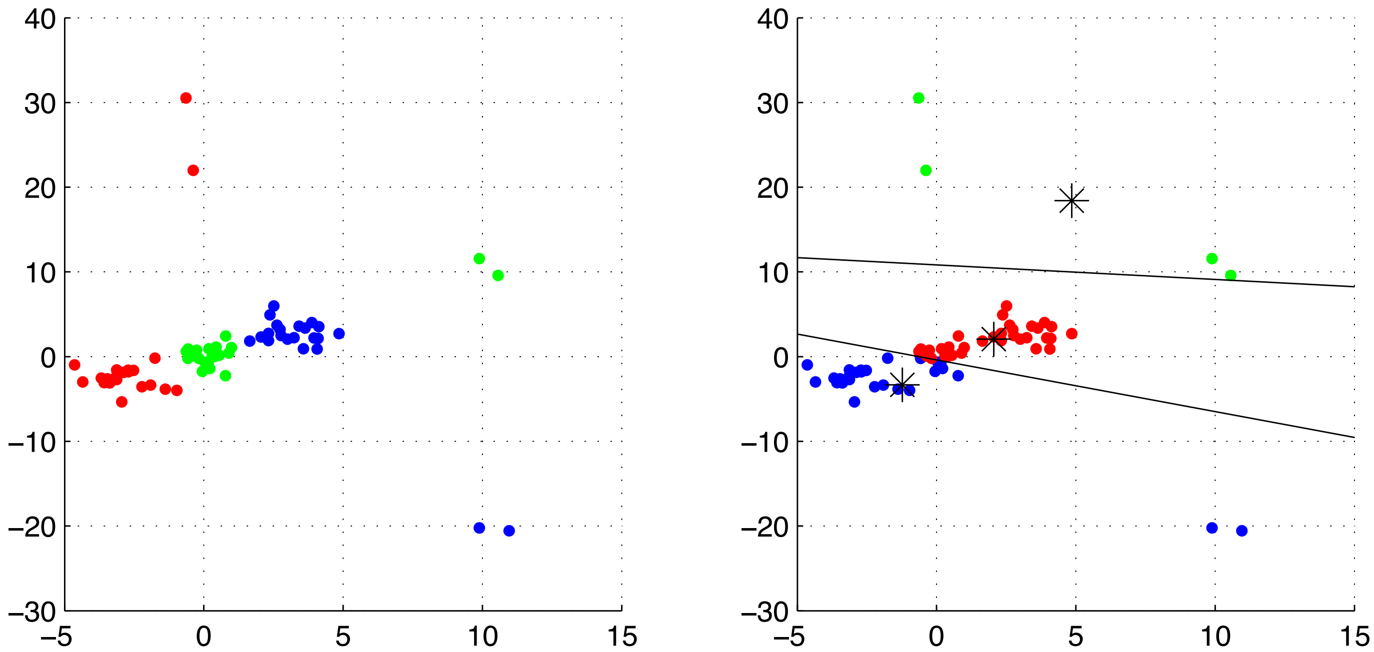
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

K-means on El Capitan



One problem

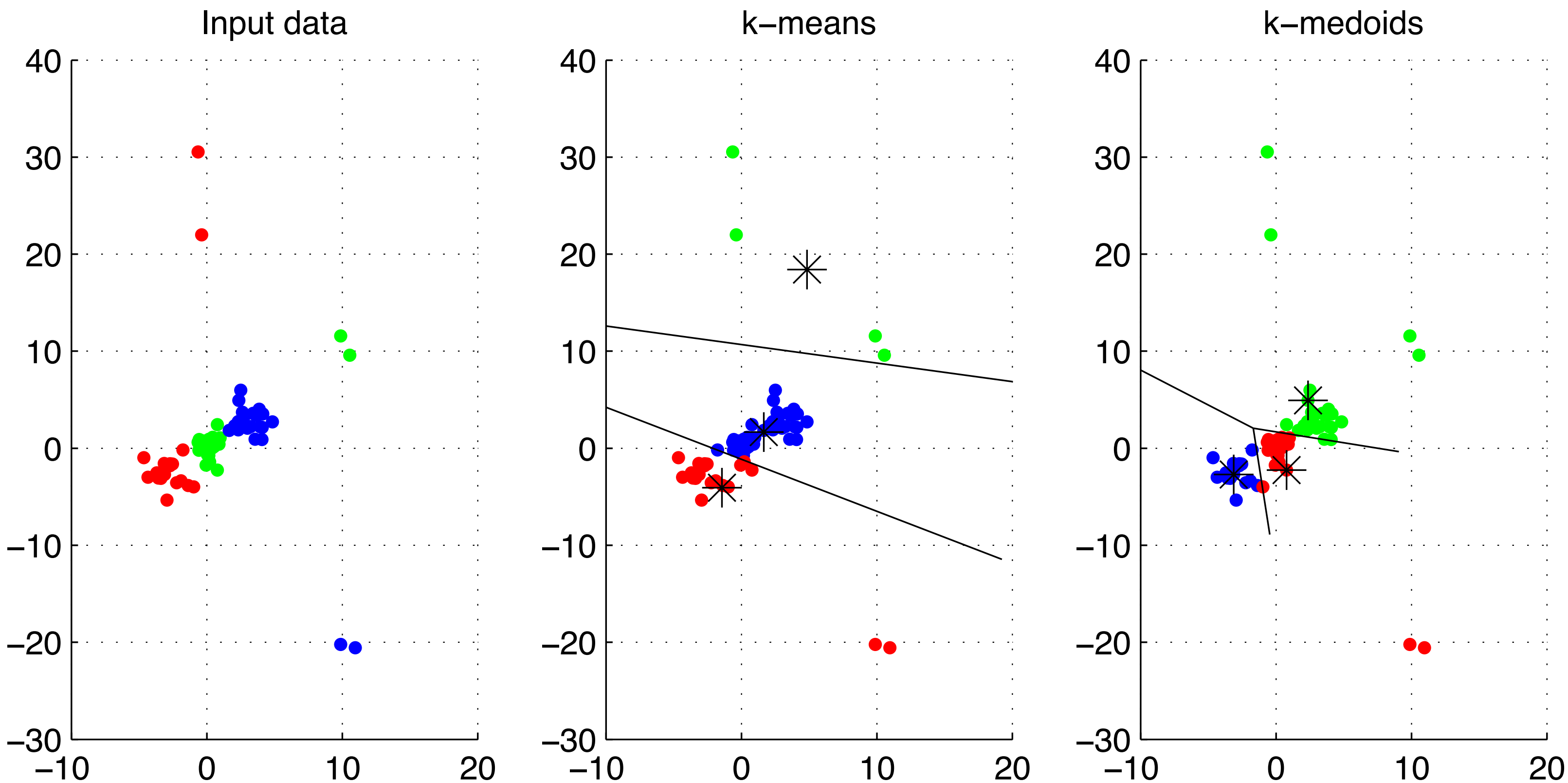
- K-means struggles with outliers



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

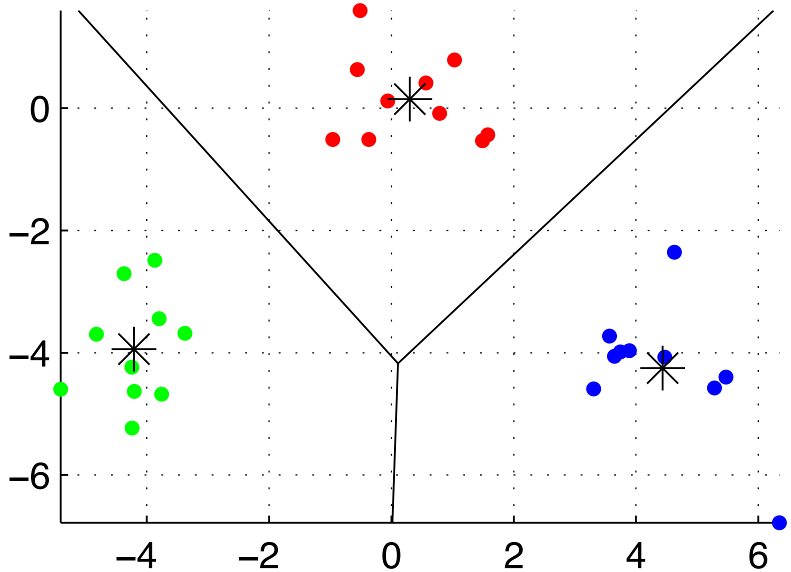


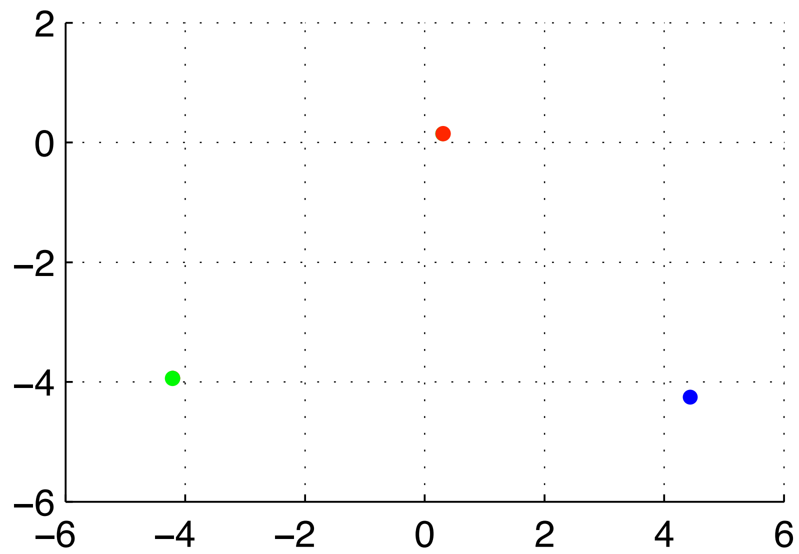
Vector Quantization

- Use of clustering for compression
  - Keep a codebook (≈ k-means)
  - Transmit nearest codebook vector instead of current sample

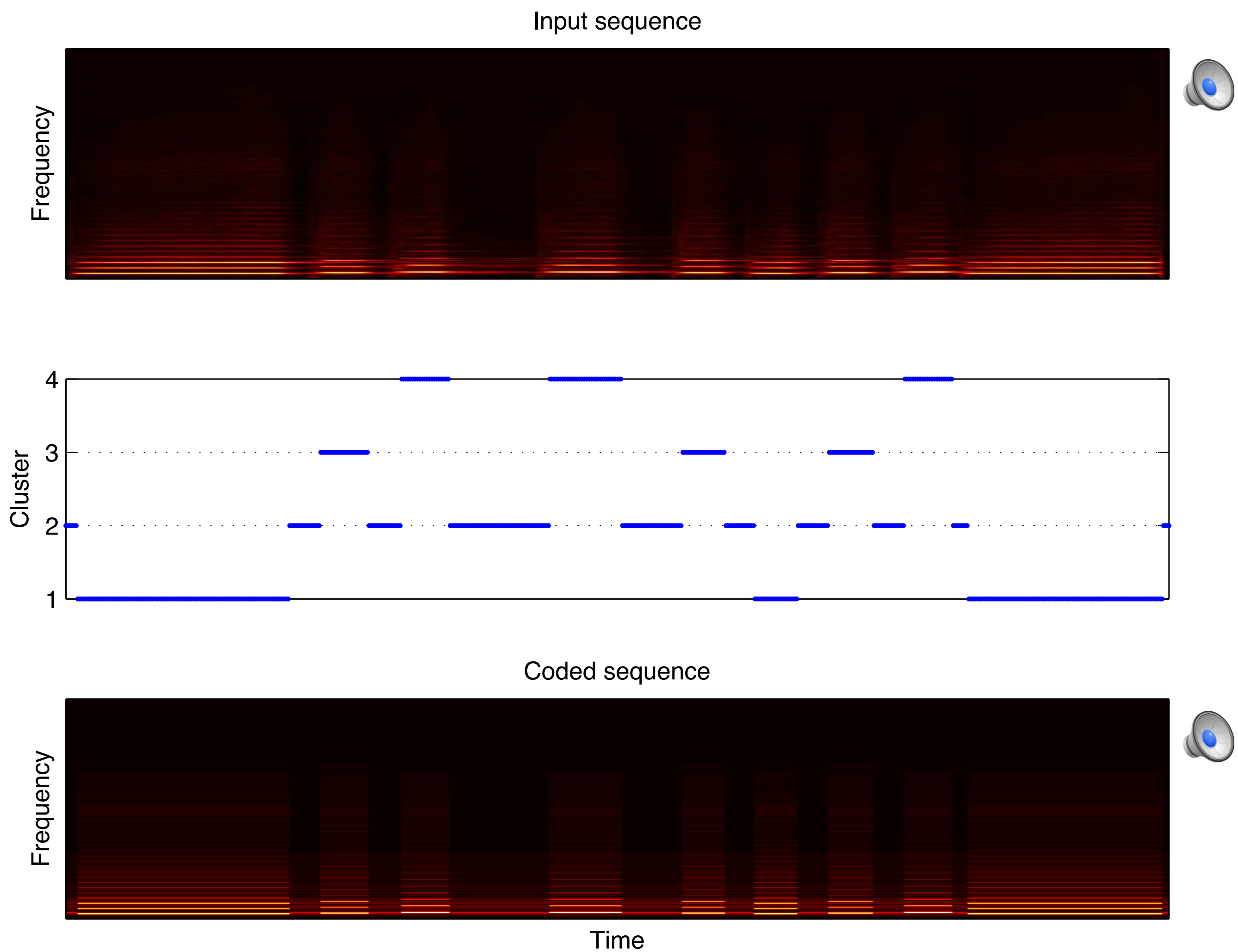
- We transmit only the cluster index, not the entire data for each sample

Simple example





Vector Quantization in Audio



Recap

- Hierarchical clustering
  - Agglomerative, Divisive
  - Issues with performance
- K-means
  - Fast and easy
  - K-medoids for more robustness (but slower)