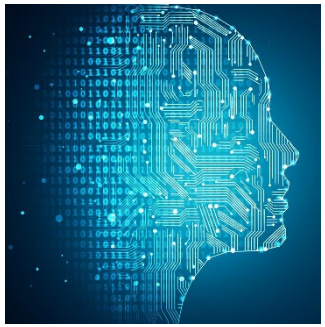
Machine Learning

k-Means Clustering

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Clustering

- Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
  - No labels ⇒ unsupervised learning
- **Goal**: Partition the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ into disjoint subsets called clusters
  - Points in the same cluster are more "similar" to each other than they are to points in different clusters
  - Can be represented as a clustering map $C: \{1, \ldots, n\} \to \{1, \ldots, k\}$
  - The unsupervised analog to classification

# of Clusters

- Different definitions of "similar" lead to different algorithms

Clustering Applications

- Market and image segmentation
- Social network analysis
- Organize computing clusters
- Astronomical data analysis
- Data reduction/compression

K-Means Criterion/Loss Function

- Choose C to minimize the following:

$$W(C) = \sum_{\ell=1}^{k} \sum_{i:C(i)=\ell} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_\ell\|^2$$

  - $\overline{\boldsymbol{x}}_\ell = \frac{1}{n_\ell} \sum_{j:C(j)=\ell} \boldsymbol{x}_j,\ n_\ell = \#\{i: C(i) = \ell\}$
- Note that k is fixed and assumed to be known
- $W(C)$ sometimes called the "within class scatter":

$$W(C) = \frac{1}{2} \sum_{\ell=1}^{k} \sum_{i:C(i)=\ell} \left[\frac{1}{n_\ell} \sum_{j:C(j)=\ell} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \right]$$

- Many clustering algorithms focus on minimizing some notion of dissimilarity

Proof

- On the board

Clustering algorithms

Three general types of algorithms

1. Combinatorial algorithms (e.g. k-means)
2. Mixture modeling (e.g. Guassian mixture model)
3. Mode seeking

The second typically assumes some probabilistic model while the first and third generally don't

K-Means Algorithm

- Minimizing $W(C)$ is a combinatorial optimization problem
- # of possible cluster maps C is

$$S(n,k) = \frac{1}{k!}\sum_{\ell=1}^{k}(-1)^{k-\ell}\binom{k}{\ell}\ell^n$$

- **Examples**: $S(10,4) = 34105$, $S(19,4) \approx 10^{10}$
  - Grows rapidly with n
- No known efficient search strategy
- **Goal**: try to find a good suboptimal partition

K-Means Algorithm

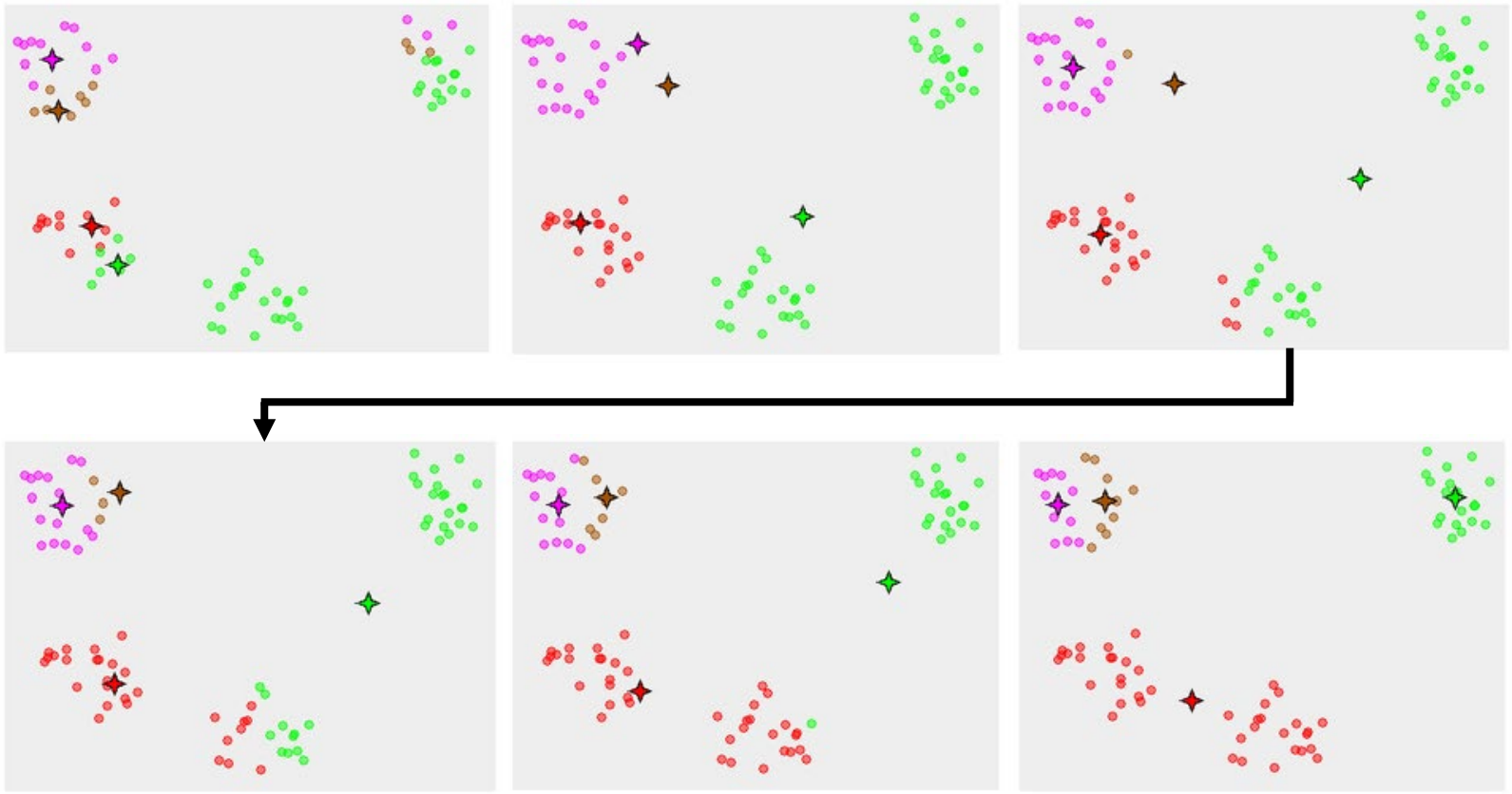
- Initialize $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k \in \mathbb{R}^d$
- Repeat
  - For $i = 1, \ldots, n$
    - $C(i) = \arg\min_{\ell} \|\boldsymbol{x}_i - \boldsymbol{m}_\ell\|^2$
  - End
  - For $\ell = 1, \ldots, k$
    - $\boldsymbol{m}_\ell = \frac{1}{n_\ell} \sum_{j:C(j)=\ell} \boldsymbol{x}_j$
  - End
- Until clusters don't change

K-Means Algorithm

- Also known as Lloyd's algorithm
- $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ are known as centroids
- **Complexity**: O(#iterations x #clusters x #instances x #dimensions)

Initialization

- The k-means result is highly dependent on the initialization
  - Converges to a local minimum
  - Thus the initial centroids determine where you end up



Initialization

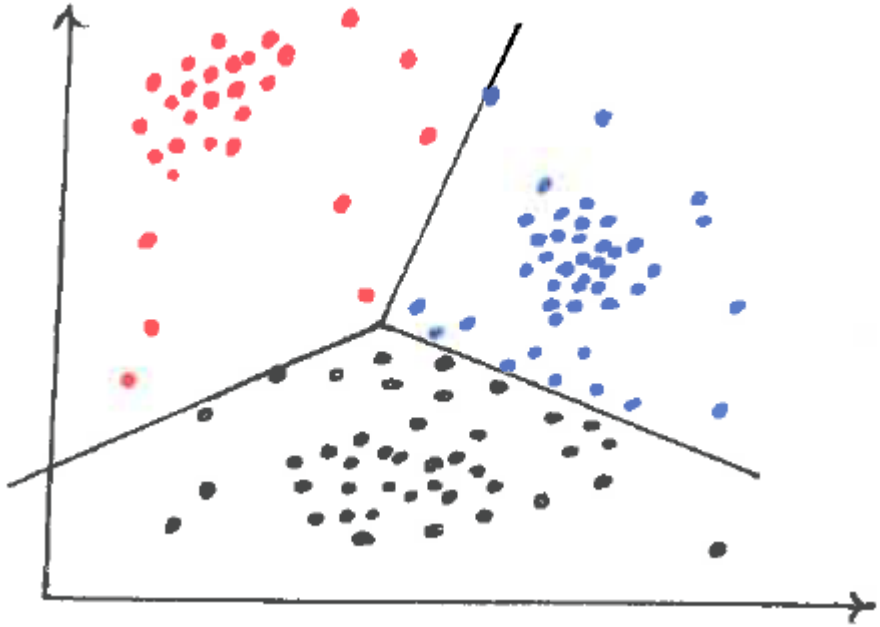
- Common strategy: initialize $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ to randomly chosen data points
- Also common to run the algorithm multiple times with different initial centroids and choose the result with the smallest $W(C)$
- Problems with random initialization:
  1. # of iterations can be quite large in the worst case
  2. Converged value of $W(C)$ can be quite far from optimal
- Better idea: choose initial $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ to be far apart
  - K-means++ is a particular implementation of this idea

K-means++ Algorithm

1. Choose $\boldsymbol{m}_1$ uniformly from $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$
2. For $\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, compute $D(\boldsymbol{x}) = \min_{j \in \{1,\ldots,i\}} \left\| \boldsymbol{x} - \boldsymbol{m}_j \right\|$ where i is the number of previously computed centroids
3. Select $\boldsymbol{m}_{i+1}$ by choosing $\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with probability $D^2(\boldsymbol{x}) / \sum_{\boldsymbol{x}} D^2(\boldsymbol{x})$
4. Repeat steps 2-3 until k centroids have been chosen
5. Proceed with standard k-means

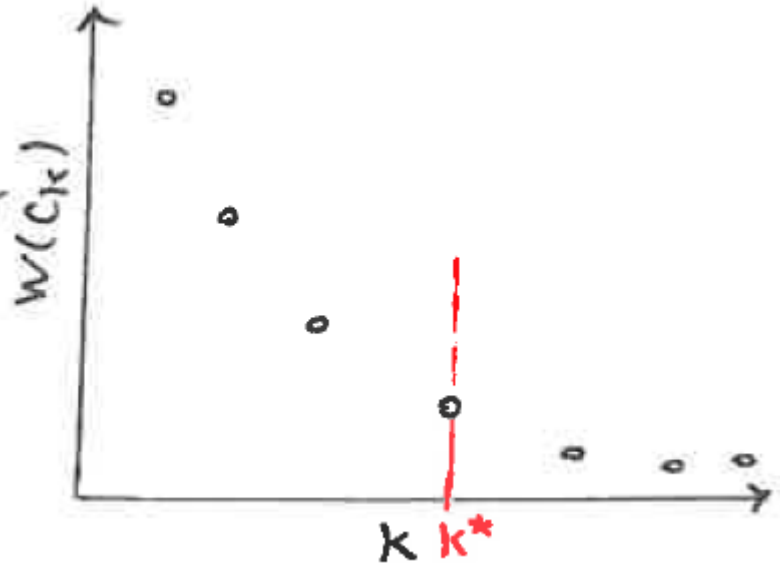
Cluster Geometry

- Clusters are “nearest neighbor” regions on Voronoi cells defined wrt cluster centroids
  - $\Rightarrow$ cluster boundaries are piecewise linear
- Thus the cluster regions are convex sets
- Therefore, k-means will fail to identify the true clusters if at least one of them is nonconvex
- K-means can be kernelized to accommodate nonconvex clusters



Model Selection

- How should k be chosen?
- Let $\hat{C}_k$ denote the output of k-means when k clusters are chosen
- Simple heuristic: plot $W(\hat{C}_k)$ as a function of k
- Basic idea: if k^* is the ideal cluster # then
  - If $k < k^*$, $W(\hat{C}_k) - W(\hat{C}_{k+1})$ will be relatively large
  - If $k > k^*$, $W(\hat{C}_k) - W(\hat{C}_{k+1})$ will be relatively small
  - Suggests choosing k close to the "knee"



Model Selection

- Another approach: add k as a regularization term to the objective function

$$\sum_{\ell=1}^{k} \sum_{i:C(i)=\ell} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_\ell\|^2 + \lambda k$$

  - λ is a regularization parameter
- Derived from a nonparametric Bayesian perspective (Kulis and Jordan, 2012)
- Optimized by a variant of k-means

Final Notes

- Many other clustering algorithms use k-means as a final step
  - A different representation of the data is learned first
  - Examples include spectral clustering, clustering on any other dimensionality reduction algorithm (e.g. diffusion maps, PHATE, t-SNE)
- Other algorithms use k-means as a preprocessing step
  - May be a useful starting configuration
  - Used in fast PHATE to determine landmarks

Further Reading

- Kulis and Jordan: https://icml.cc/2012/papers/291.pdf
- ISL Section 10.3.1
- ESL Sections 13.2.1 and 14.3.6