The K-means algorithm

D.A. Forsyth,

University of Illinois at Urbana Champaign

Setup

Write $\mathbf{x}_i$ for a set of N data items, which have been coerced to be vectors. Assume you know that there are k clusters. Write $\mathbf{c}_j$ for the center of the jth cluster. Write $\delta_{i,j}$ for a discrete variable that records which cluster a data item belongs to, so

$$\delta_{i,j} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases}$$

Every data item belongs to exactly one cluster, so $\sum_j \delta_{i,j} = 1$. Every cluster must contain at least one point, so $\sum_i \delta_{i,j} > 0$ for every j. The sum of squared distances from data points to cluster centers is then

$$\Phi(\delta, \mathbf{c}) = \sum_{i,j} \delta_{i,j} \left[(\mathbf{x}_i - \mathbf{c}_j)^T (\mathbf{x}_i - \mathbf{c}_j) \right].$$

Notice how the $\delta_{i,j}$ are acting as "switches". For the i'th data point, there is only one non-zero $\delta_{i,j}$ which selects the distance from that data point to the appropriate cluster center.

You could cluster the data by choosing the δ and $\mathbf{c}$ that minimizes $\Phi(\delta, \mathbf{c})$. This would yield the set of k clusters and their cluster centers such that the sum of distances from points to their cluster centers is minimized. There is no known algorithm that can minimize Φ exactly in reasonable time. The $\delta_{i,j}$ are the problem:

Approximation

Notice that if the $\mathbf{c}$'s are known, getting the δ's is easy – for the i'th data point, set the $\delta_{i,j}$ corresponding to the closest $\mathbf{c}_j$ to one and the others to zero. Similarly, if the $\delta_{i,j}$ are known, it is easy to compute the best center for each cluster – just average the points in the cluster. These observations yield a remarkably effective approximate algorithm. Iterate:

- Assume the cluster centers are known and allocate each point to the closest cluster center.
- Replace each center with the mean of the points allocated to that cluster; if there are no points in the cluster, restart the cluster by choosing some point uniformly and at random from the dataset and making that the cluster center.

Procedure: 9.3 *K-means clustering*

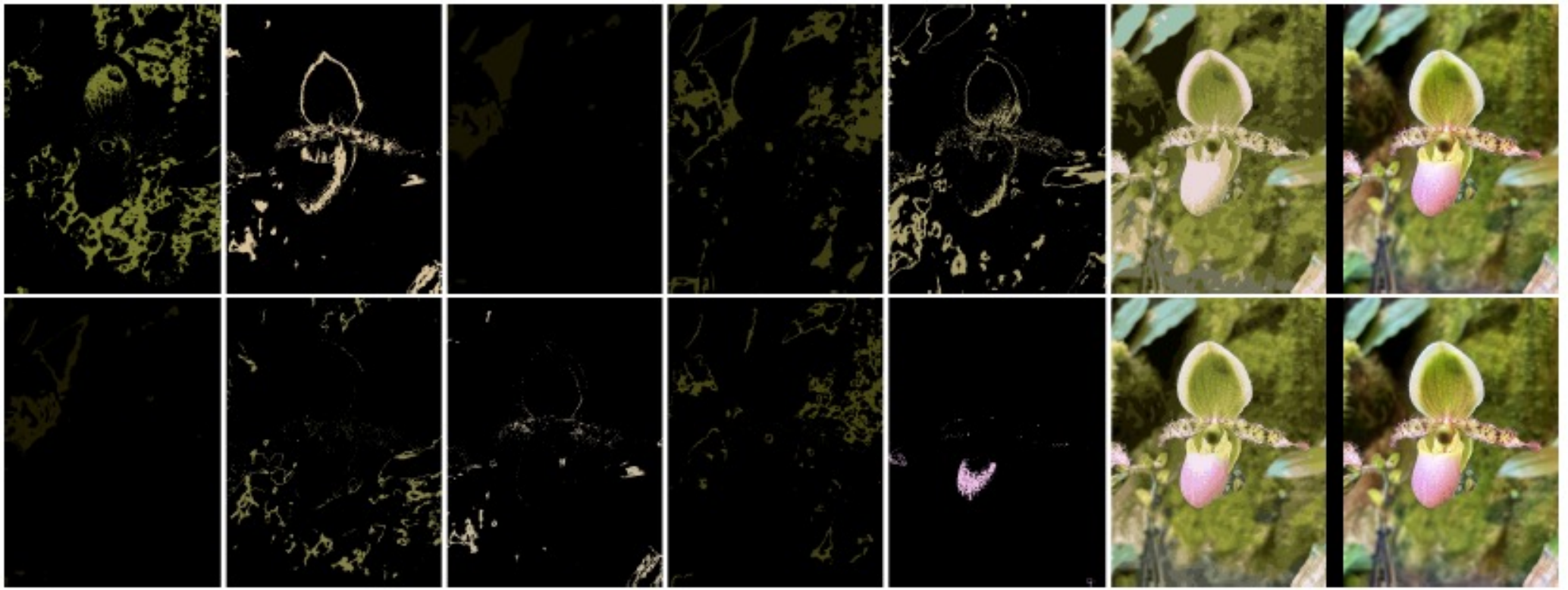
Initialize by choosing k and k initial cluster centers $\mathbf{c}_i$.
Iterate until convergence:

- Allocate each data item $\mathbf{x}_j$ to the closest cluster center.
- Replace each center with the mean of the points allocated to that cluster; if there are no points in the cluster, choose some point uniformly and at random from the dataset and make that the cluster center.

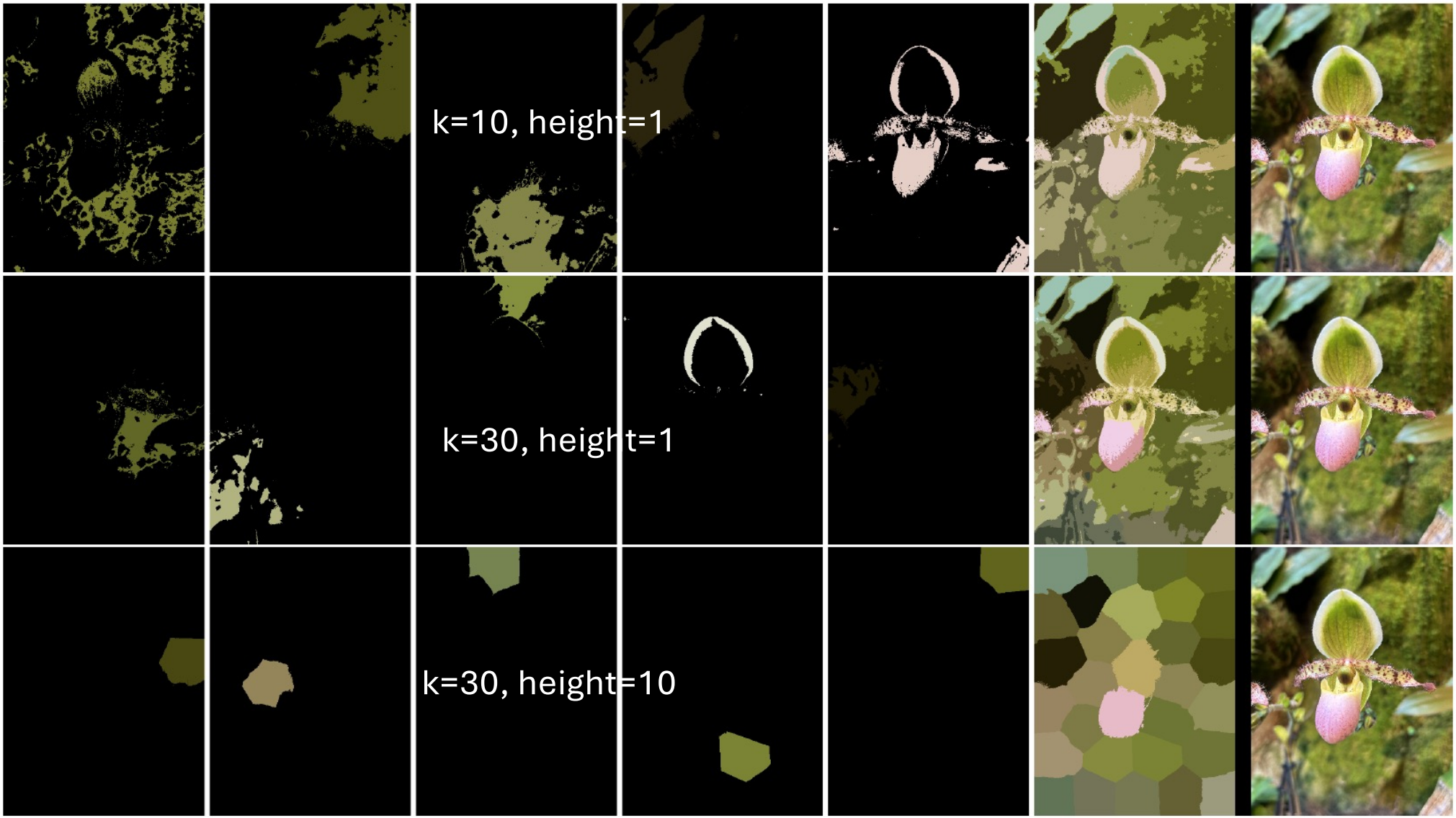
Test convergence by a combination of tests. Always stop when the number of iterations exceeds a threshold. You could stop if the cluster centers have moved by less than a threshold between iterations. Finally, if the allocation of data items to cluster centers has not changed between iterations, the method must have converged.

kmeans on color

- top k=10, bottom k=30



kmeans color and position



Issues

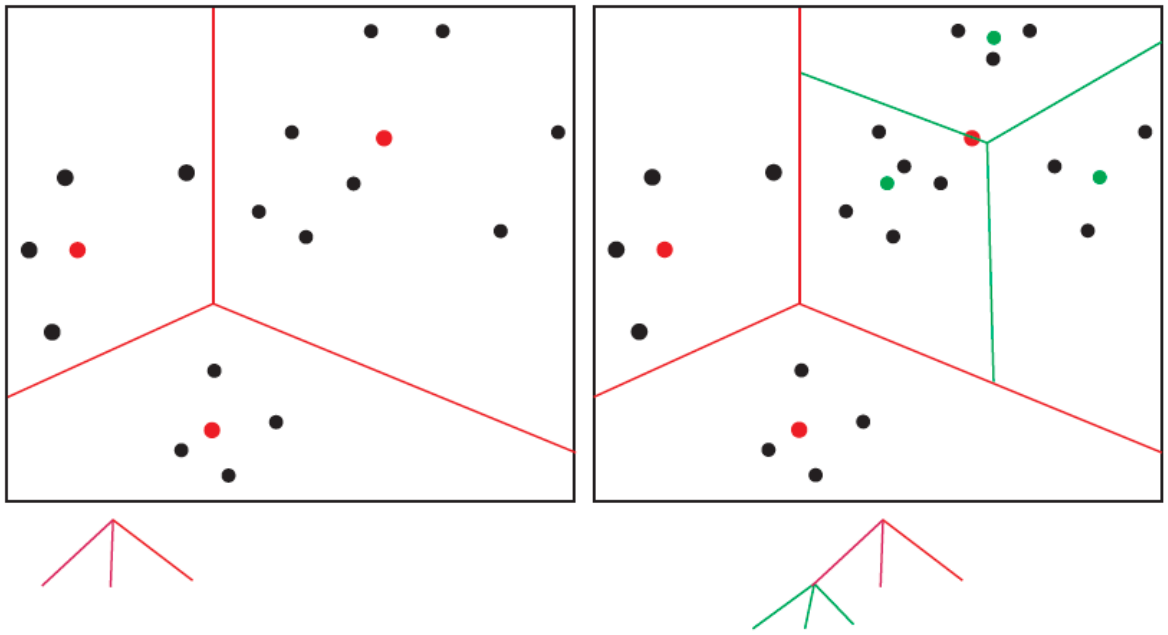
- Initializing
  - random choice of cluster centers
  - kmeans++, notes
- Empty clusters
  - random restart
- Scattered points
  - junk cluster
- What is k?
  - very hard from cluster metrics
  - application considerations

Efficiency

- What if the dataset is gigantic and k is huge?
  - subsample data; build a tree
- Hierarchical k-means

Hierarchical k-means

Remember this: *Build a tree using k-means by clustering a sample of the data, then allocating new data to the cluster with the closest center, and recurring. Stop when there is too little data in a cluster.*



Things to think about

9.5. Confirm that the cost function for k-means does not go up at each iteration, as long as no cluster required restarting.

9.6. Section 9.3.2 has "If there are more centers, each data point can find a center that is closer to it, so the value should go down as k goes up." Does this *always* happen? could the value go up?

9.7. Section 9.3.2 has "The best k is then the number of data points, which is not helpful." Explain.

9.8. You cluster 1e6 data points with hierarchical k-means, using random subsampling and $k = 100$; roughly how many leaves to you expect? How deep do you expect the tree to be?