Clustering 1: K-means, K-medoids

Optional reading: ISL 10.3, ESL 14.3

What is clustering? And why?

Clustering: task of dividing up data into groups (clusters), so that points in any one group are more "similar" to each other than to points outside the group

Why cluster? Two main uses

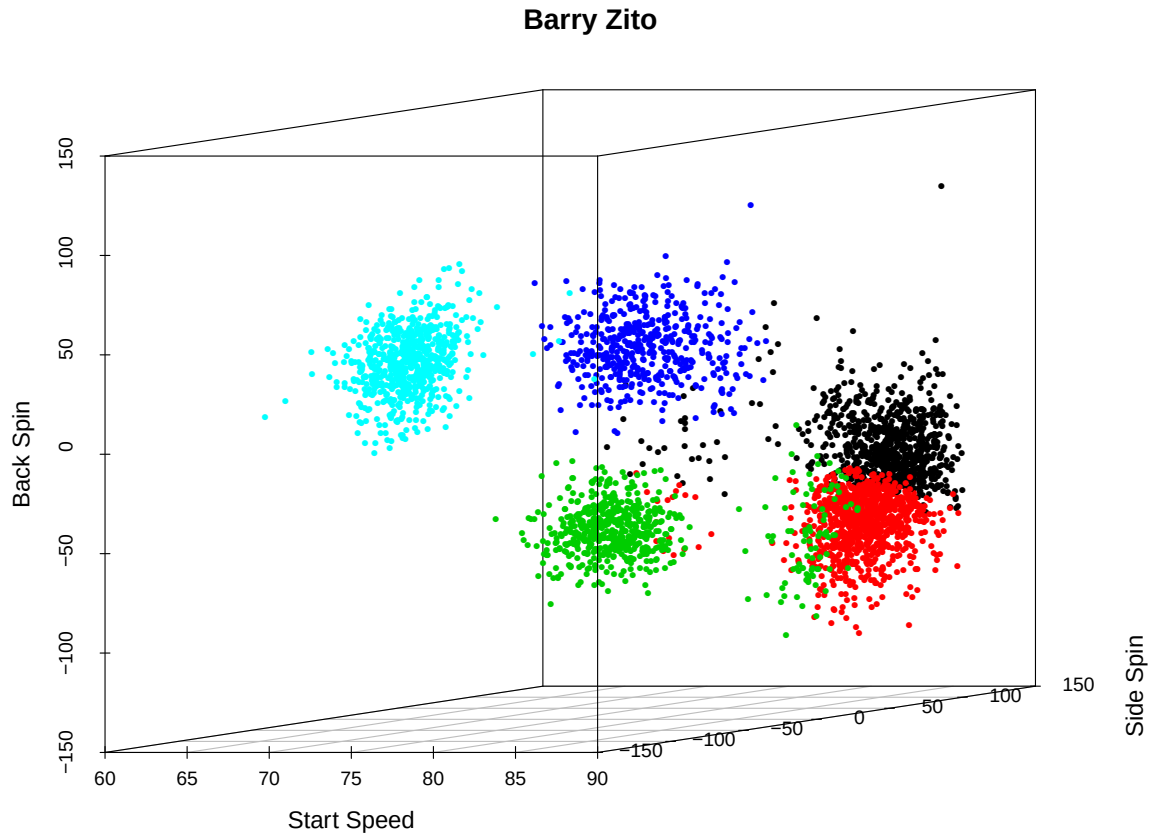
- Summary: deriving a reduced representation of the full data set. E.g., vector quantitization (we'll see this shortly)
- Discovery: looking for new insights into the structure of the data. E.g., finding groups of students that commit similar mistakes, or groups of 80s songs that sound alike

Other uses, e.g.,

- Checking up on someone else's work/decisions, investigating the validity of pre-existing group assignments
- Helping with prediction, i.e., in classification or regression

We won't study clustering for verification purposes, but we'll see clustering in prediction later

Clustering baseball pitches



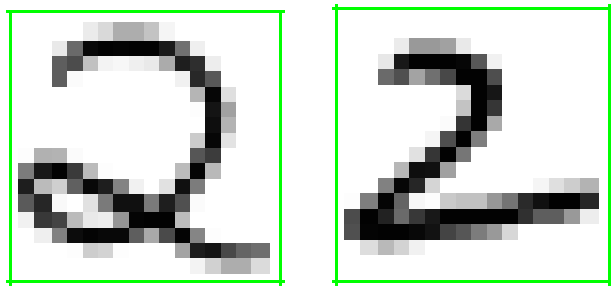
Inferred meaning of clusters: black – fastball, red – sinker, green – changeup, blue – slider, light blue – curveball

(Example from Mike Pane, former data mining student)

Don't confuse clustering and classification!

In classification, we have data for which the groups are known, and we try to learn what differentiates these groups (i.e., classification function) to properly classify future data

In clustering, we look at data for which groups are unknown and undefined, and try to learn the groups themselves, as well as what differentiates them



Dissimilarity and within-cluster scatter

Given observations $X_1, \ldots X_n$, and dissimilarites $d(X_i, X_j)$. (E.g., think of $X_i \in \mathbb{R}^p$ and $d(X_i, X_j) = \|X_i - X_j\|_2^2$)

Let K be the number of clusters (fixed). A clustering of points $X_1, \ldots X_n$ is a function C that assigns each observation X_i to a group $k \in \{1, \ldots K\}$

Notation: $C(i) = k$ means that X_i is assigned to group k, and n_k is the number of points in the group k. Also, let $d_{ij} = d(X_i, X_j)$

The within-cluster scatter is defined as

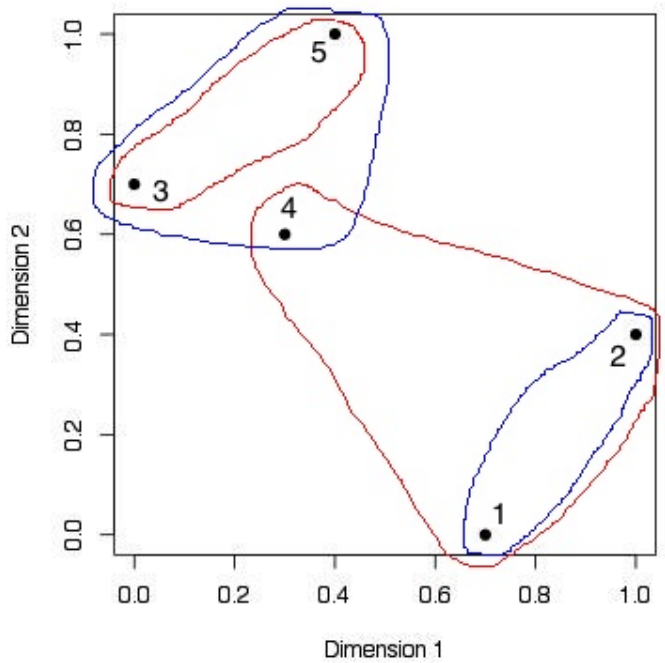
$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{C(i)=k, C(j)=k} d_{ij}$$

Smaller W is better

Simple example

Here $n = 5$ and $K = 2$,
$X_i \in \mathbb{R}^2$ and $d_{ij} = \|X_i - X_j\|_2^2$

	1	2	3	4	5
1	0	0.25	0.98	0.52	1.09
2	0.25	0	1.09	0.53	0.72
3	0.98	1.09	0	0.10	0.25
4	0.52	0.53	0.10	0	0.17
5	1.09	0.72	0.25	0.17	0



- Red clustering:
  $W_{\text{red}} = (0.25 + 0.53 + 0.52)/3 + 0.25/2 = 0.56$
- Blue clustering:
  $W_{\text{blue}} = 0.25/2 + (0.10 + 0.17 + 0.25)/3 = 0.30$

(Tip: `dist` function in R)

Finding the best group assignments

Smaller W is better, so why don't we just directly find the clustering C that minimizes W?

Problem: doing so requires trying all possible assignments of the n points into K groups. The number of possible assignments is

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

Note that $A(10, 4) = 34,105$, and $A(25, 4) \approx 5 \times 10^{13}$ ... huge

Most problems we look at are going to have way more than $n = 25$ observations, and potentially more than $K = 4$ clusters too (but $K = 4$ is not unrealistic)

So we'll have to settle for an approximation

Rewriting the within-cluster scatter

Focus on Euclidean space: now $X_i \in \mathbb{R}^p$ and dissimilarities are $d(X_i, X_j) = \|X_i - X_j\|_2^2$

Fact (Homework 1): within-cluster scatter can be rewritten as

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}\|X_i - X_j\|_2^2 = \sum_{k=1}^{K}\sum_{C(i)=k}\|X_i - \bar{X}_k\|_2^2$$

with $\bar{X}_k$ the average of points in group k, $\bar{X}_k = \frac{1}{n_k}\sum_{C(i)=k} X_i$. The right-hand side above is called within-cluster variation

Hence, equivalently we seek a clustering C that minimizes the within-cluster variation (approximately so)

Rewriting the minimization

Remember: we want to choose C to minimize

$$\sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - \bar{X}_k\|_2^2$$

Another fact (Homework 1): for any $Z_1, \ldots Z_m \in \mathbb{R}^p$, the quantity $\sum_{i=1}^m \|Z_i - c\|_2^2$ is minimized by taking $c = \bar{Z} = \frac{1}{m}\sum_{i=1}^m Z_i$, the average of the Z_i's

So our problem is the same as minimizing the enlarged criterion

$$\sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - c_k\|_2^2,$$

over both clusterings C and $c_1, \ldots c_K \in \mathbb{R}^p$

K-means algorithm

The K-means clustering algorithm approximately minimizes the enlarged criterion by alternately minimizing over C and $c_1, \ldots c_K$

We start with an initial guess for $c_1, \ldots c_K$ (e.g., pick K points at random over the range of $X_1, \ldots X_n$), then repeat:

1. Minimize over C: for each $i = 1, \ldots n$, find the cluster center c_k closest to X_i, and let $C(i) = k$
2. Minimize over $c_1, \ldots c_K$: for each $k = 1, \ldots K$, let $c_k = \bar{X}_k$, the average of points in group k

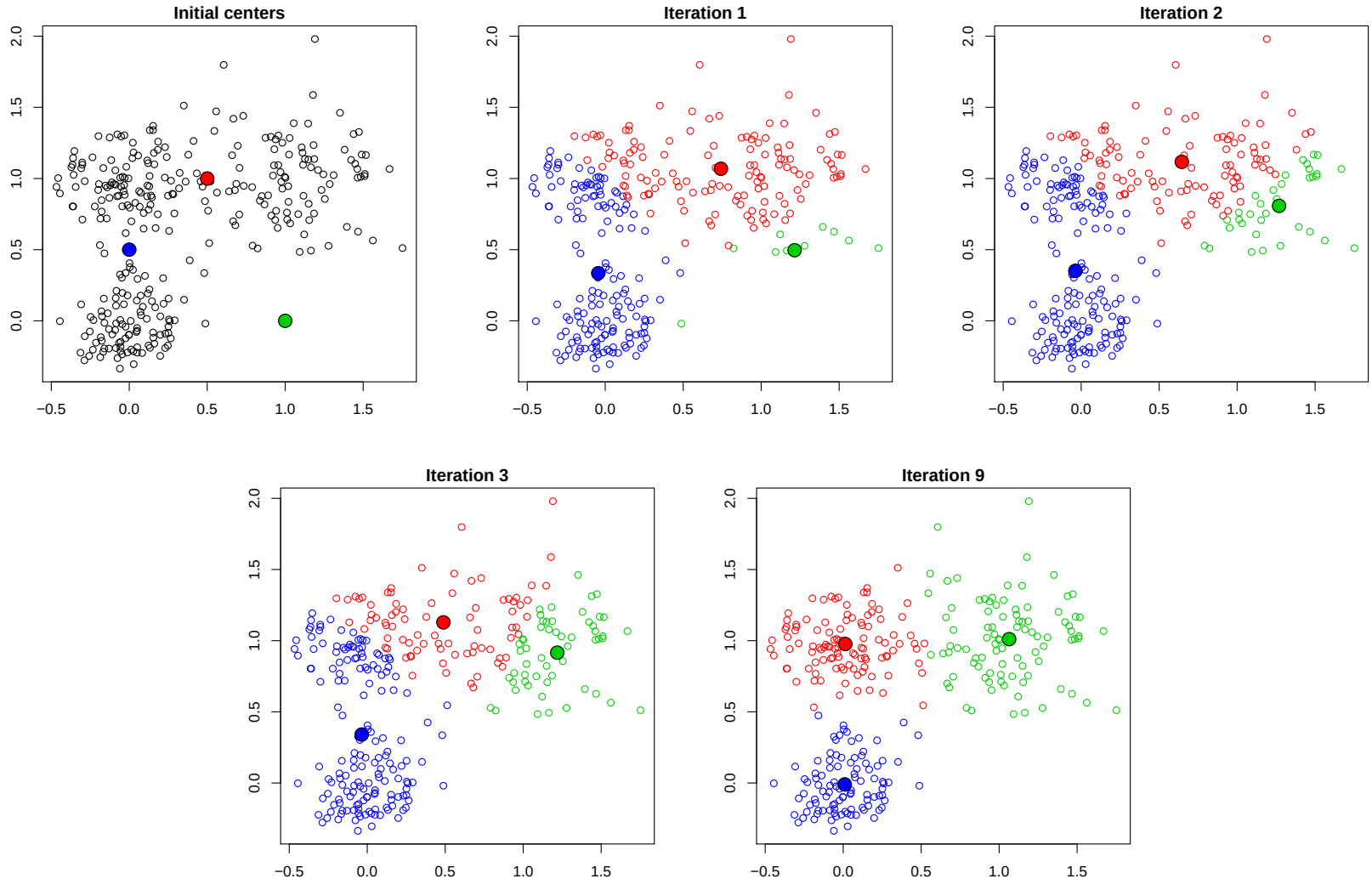
Stop when within-cluster variation doesn't change

In words:

1. Cluster (label) each point based the closest center
2. Replace each center by the average of points in its cluster

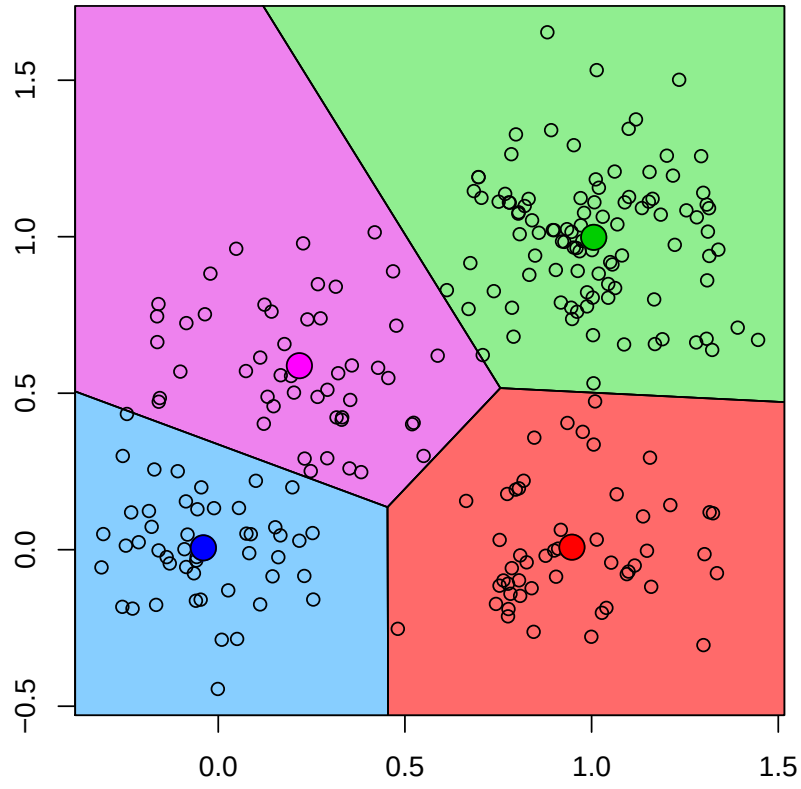
K-means example

Here $X_i \in \mathbb{R}^2$, $n = 300$, and $K = 3$



Voronoi tessellation

Given cluster centers, we identify each point to its nearest center. This defines a Voronoi tessellation of $\mathbb{R}^p$



Given $c_1, \ldots c_K \in \mathbb{R}^p$, we define the Voronoi sets

$$V_k = \{x \in \mathbb{R}^p : \|x - c_k\|_2^2 \leq \|x - c_j\|_2^2, \, j = 1, \ldots K\}, \;\; k = 1, \ldots K$$

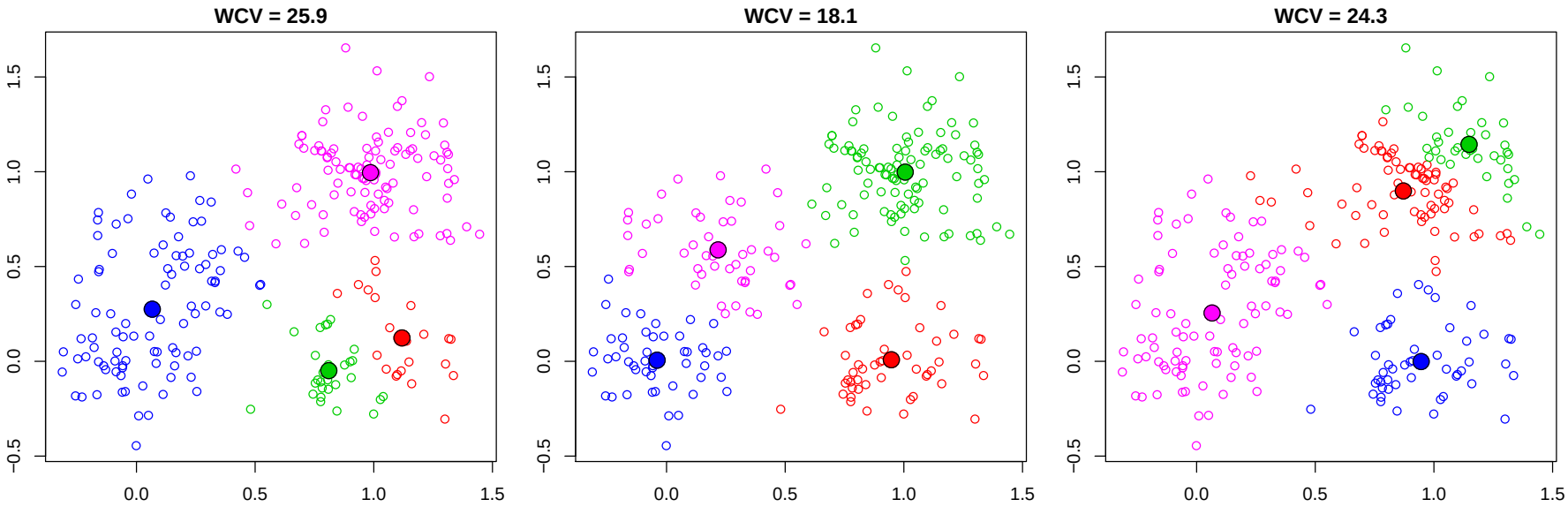
These are convex polyhedra (we'll see them again when we study classification)

Properties of K-means

- Within-cluster variation decreases with each iteration of the algorithm. I.e., if W_t is the within-cluster variation at iteration t, then $W_{t+1} \leq W_t$ (Homework 1)
- The algorithm always converges, no matter the initial cluster centers. In fact, it takes $\leq K^n$ iterations (why?)
- The final clustering depends on the initial cluster centers. Sometimes, different initial centers lead to very different final outputs. So we typically run K-means multiple times (e.g., 10 times), randomly initializing cluster centers for each run, then choose among from collection of centers based on which one gives the smallest within-cluster variation
- The algorithm is not guaranteed to deliver the clustering that globally minimizes within-cluster variation (recall: this would require looking through all possible assignments!)

K-means example, multiple runs

Here $X_i \in \mathbb{R}^2$, $n = 250$, and $K = 4$, the points are not as well-separated



These are results of result of running the K-means algorithm with different initial centers (chosen randomly over the range of the X_i's). We choose the second collection of centers because it yields the smallest within-cluster variation

Vector quantization

(Example from ESL p. 514.) Left: original image; middle: using 23.9% of the storage; right: using 6.25% of the storage

K-means is often called "Lloyd's algorithm" in computer science and engineering, and is used in vector quantization for compression

Basic idea: run K-means clustering on 4×4 squares of pixels in an image, and keep only the clusters and labels. Smaller K means more compression

In K-means, cluster centers are averages

A cluster center is representative for all points in a cluster, also called a prototype

In K-means, we simply take a cluster center to be the average of points in the cluster. Great for computational purposes—but how does it lend to interpretation?

This would be fine if we were clustering, e.g., houses in Pittsburgh based on features like price, square footage, number of bedrooms, distance to nearest bus stop, etc.

Not so if we were clustering faces of statistics professors (why?)

K-medoids algorithm

In some applications we want each center to be one of the points itself. This is where K-medoids comes in—an algorithm similar to the K-means algorithm, except when fitting the centers $c_1, \ldots c_K$, we restrict our attention to the points themselves

Initial guess for centers $c_1, \ldots c_K$ (e.g., randomly select K of the points $X_1, \ldots X_n$), then repeat:

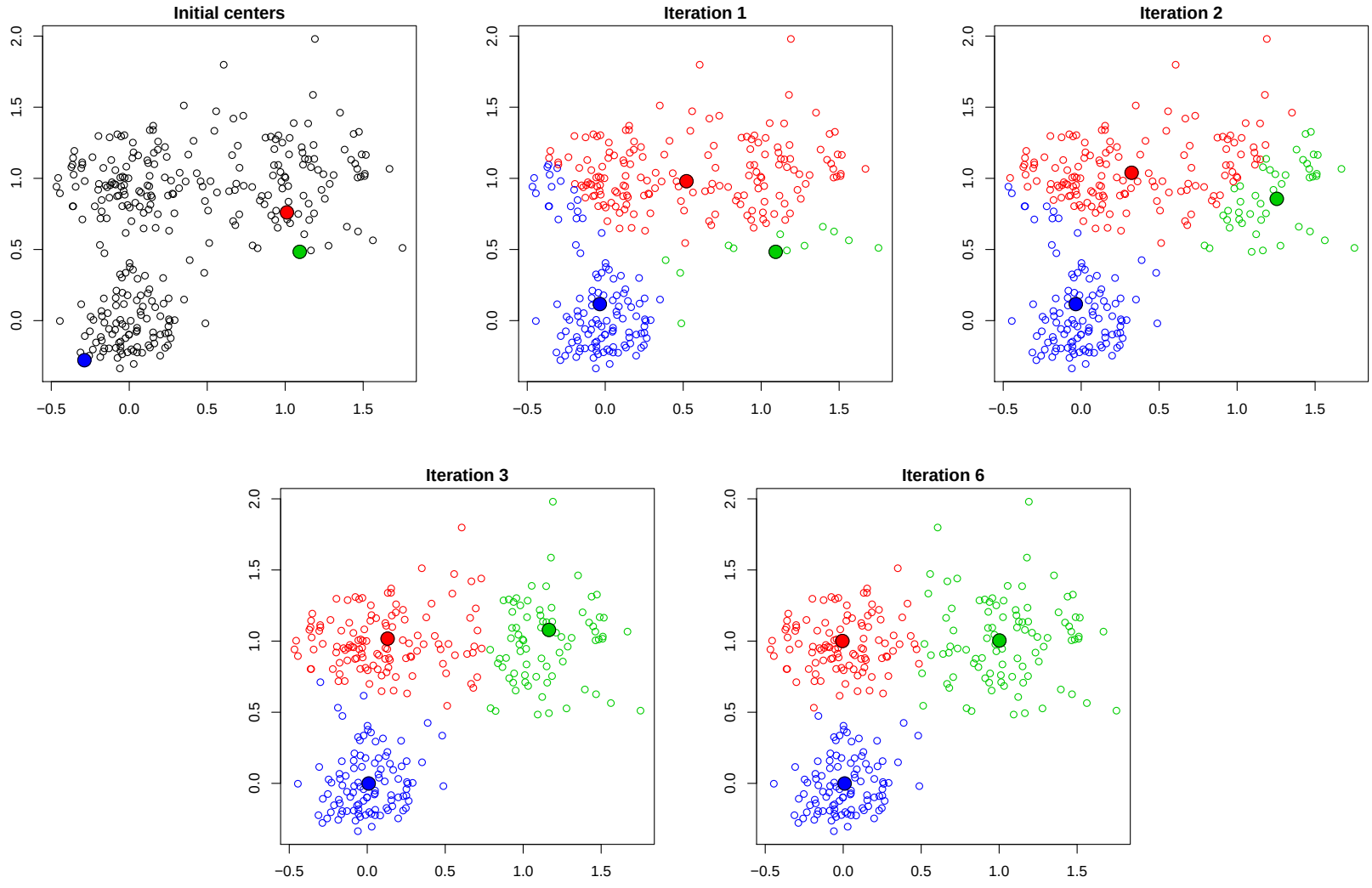
1. Minimize over C: for each $i = 1, \ldots n$, find the cluster center c_k closest to X_i, and let $C(i) = k$
2. Minimize over $c_1, \ldots c_K$: for each $k = 1, \ldots K$, let $c_k = X_k^*$, the medoid of points in cluster k, i.e., the point X_i in cluster k that minimizes $\sum_{C(j)=k} \|X_j - X_i\|_2^2$

Stop when within-cluster variation doesn't change

In words:

1. Cluster (label) each point based on the closest center
2. Replace each center by the medoid of points in its cluster

K-medoids example



Note: only 3 points had different labels under K-means

Properties of K-medoids

The K-medoids algorithm shares the properties of K-means that we discussed (each iteration decreases the criterion; the algorithm always converges; different starts gives different final answers; it does not achieve the global minimum)

K-medoids generally returns a higher value of

$$\sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - c_k\|_2^2$$

than does K-means (why?). Also, K-medoids is computationally harder than K-means (because of step 2: computing the medoid is harder than computing the average)

Remember, K-medoids has the (potentially important) property that the centers are located among the data points themselves

K-means and K-medoids in R

The K-means algorithm is part of the base distribution in R, given by the `kmeans` function (use `algorithm="Lloyd"`)

E.g.,

```
km = kmeans(x, centers=k, nstart=10, algorithm="Lloyd")
```

The K-medoids algorithm is implemented by the function `pam` (stands "for partitioning around medoids") in the package `cluster`

Recap: clustering

In clustering we divide up our data points into groups or clusters. We want points in any one group to be more similar to each other than to other points. All based on pairwise dissimilarities d_{ij}

Fixing the number of clusters K, the task of exactly minimizing the within-cluster variation (equivalently, within-cluster scatter) is not feasible. The K-means algorithm approximately minimizes this by iterating two simple steps

Though it always converges, the answer given by K-means depends on the initial centers. It also returns centers that are averages of data points. The K-medoids algorithm is an alternative where the centers are chosen among the points themselves. Its answer also depends on the starting configuration. Hence for either algorithm, one should run it several times with different starts

Next time: hierarchical clustering