Clustering 2

Nov 3 2008

HAC Algorithm

Start with all objects in their own cluster.
Until there is only one cluster:
    Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
    Replace c_i and c_i with a single cluster $c_i \cup c_i$

To compute the distance between two clusters

Single Link: distance of two closest members of clusters

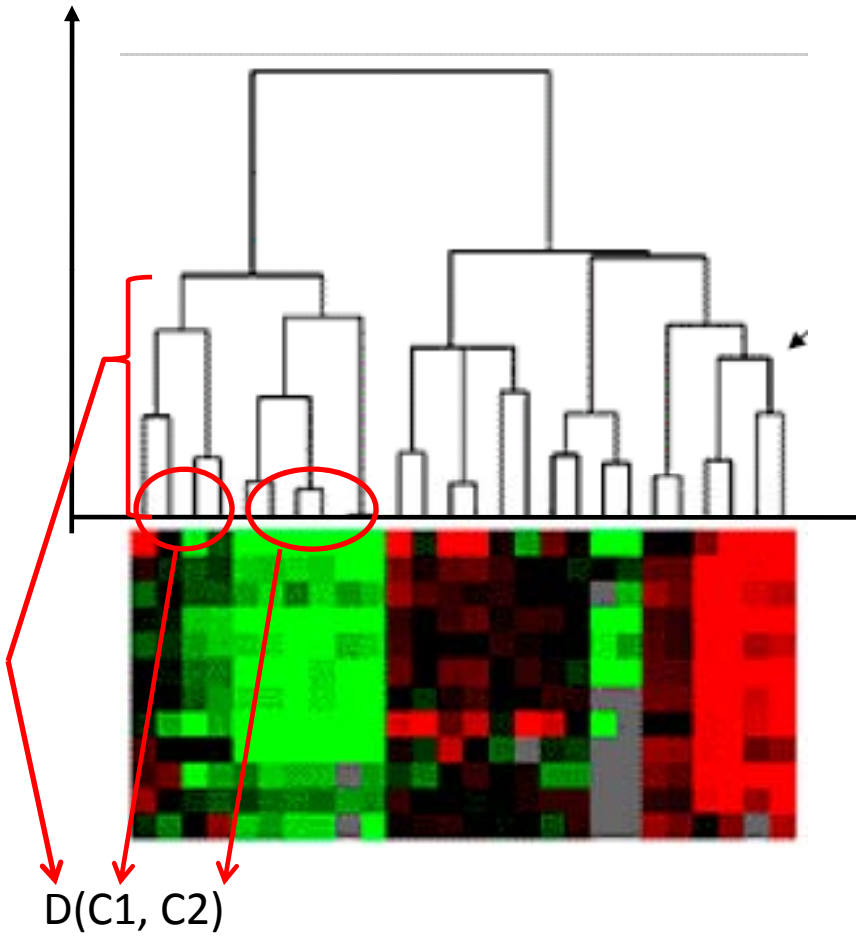
Complete Link: distance of two furthest members of clusters

Average Link: average distance

Comments on HAC

- HAC is a fast tool that often provides interesting views of a dataset
- Primarily HAC can be viewed as an intuitively appealing clustering procedure for data analysis/exploration
- We can create clusterings of different granularity by stopping at different levels of the dendrogram
- HAC often used together with visualization of the dendrogram to decide how many clusters exist in the data

HAC creates a Dendrogram



- Dendrogram draws the tree such that the height of a tree branch = the distance between the two merged clusters at that particular step
- The distances are always monotonically increasing
- This can provide some understanding about how many natural groups there are in the data
- A drastic height change indicates that we are merging two very different clusters together – maybe a good stopping point

Flat vs hierarchical clustering

- Hierarchical clustering generates nested clusterings
- Flat clustering generates a single partition of the data without resorting to the hierachical procedure
- Representative example: K-means clustering

Mean-Squared Error Objective

- Assume instances are real-valued vectors
- Given a clustering of the data into k clusters, we can compute the centroid of each cluster

$$\mu_c = \frac{1}{|c|} \sum_{\mathbf{x} \in c} \mathbf{x}$$

- If we use the mean of each cluster to represent all data points in the cluster, our mean squared error (MSE) is:

$$J_e = \sum_{i=1}^{k} \sum_{x \in c_i} ||x - \mu_i||^2$$

- One possible objective of clustering is to find a clustering that minimizes this error
- Note that this is a combinatorial optimization problem
  - Difficult to find the exact solution
  - Commonly solved using an iterative approach

Basic idea

- We assume that the number of desired clusters, *k*, is given
- Randomly choose *k* examples as *seeds*, one per cluster.
- Form initial clusters based on these seeds.
- Iterate by repeatedly reallocating instances to different clusters to improve the overall clustering.
- Stop when clustering converges or after a fixed number of iterations.

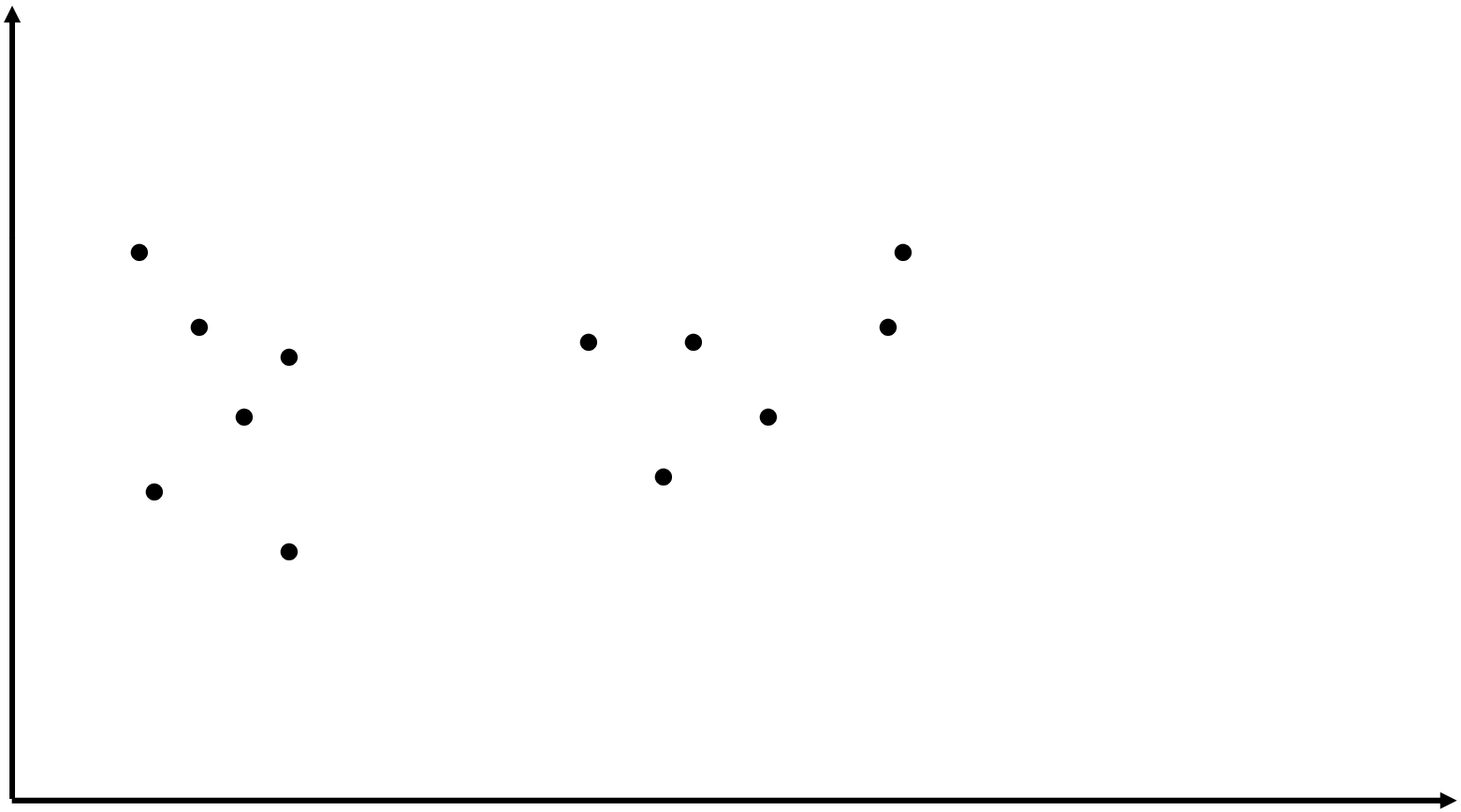
K-means algorithm (MacQueen 1967)

Input: data to be clustered and desired number of clusters k

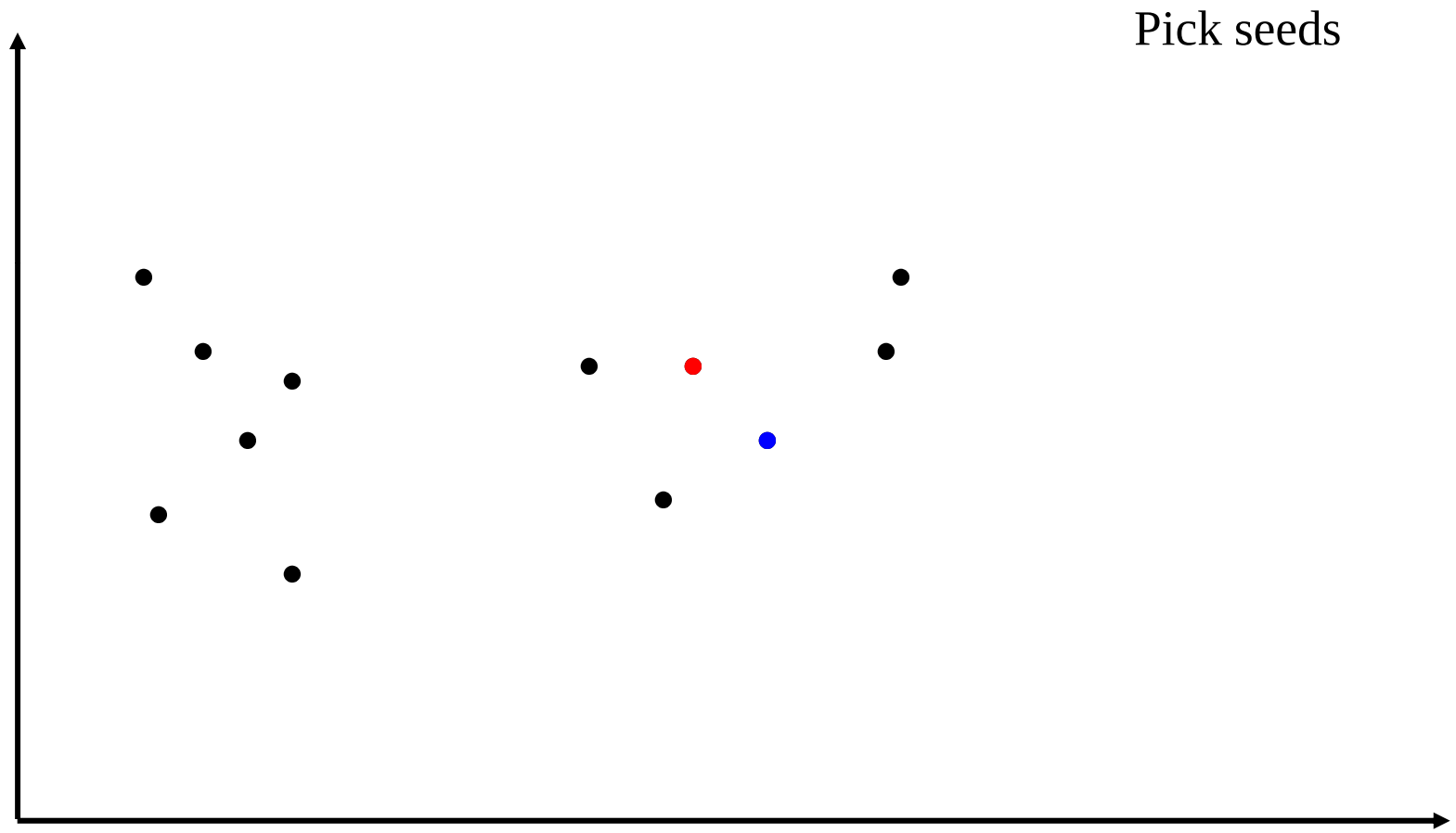
Let d be the distance measure between instances.
Select k random instances $\{s_1, s_2, \ldots s_k\}$ as seeds.
Until clustering converges or other stopping criterion:
    For each instance x_i:
        Assign x_i to the cluster c_j such that $d(x_i, \mathbf{s}_j)$ is minimal.
        (only change assignment if another cluster is strictly better than current cluster)
    For each cluster c_j
        $\mathbf{s}_j = \mu_j$ // (*Update the seeds to the centroid of each cluster*)

Output: k mutually exclusive clusters that cover all examples

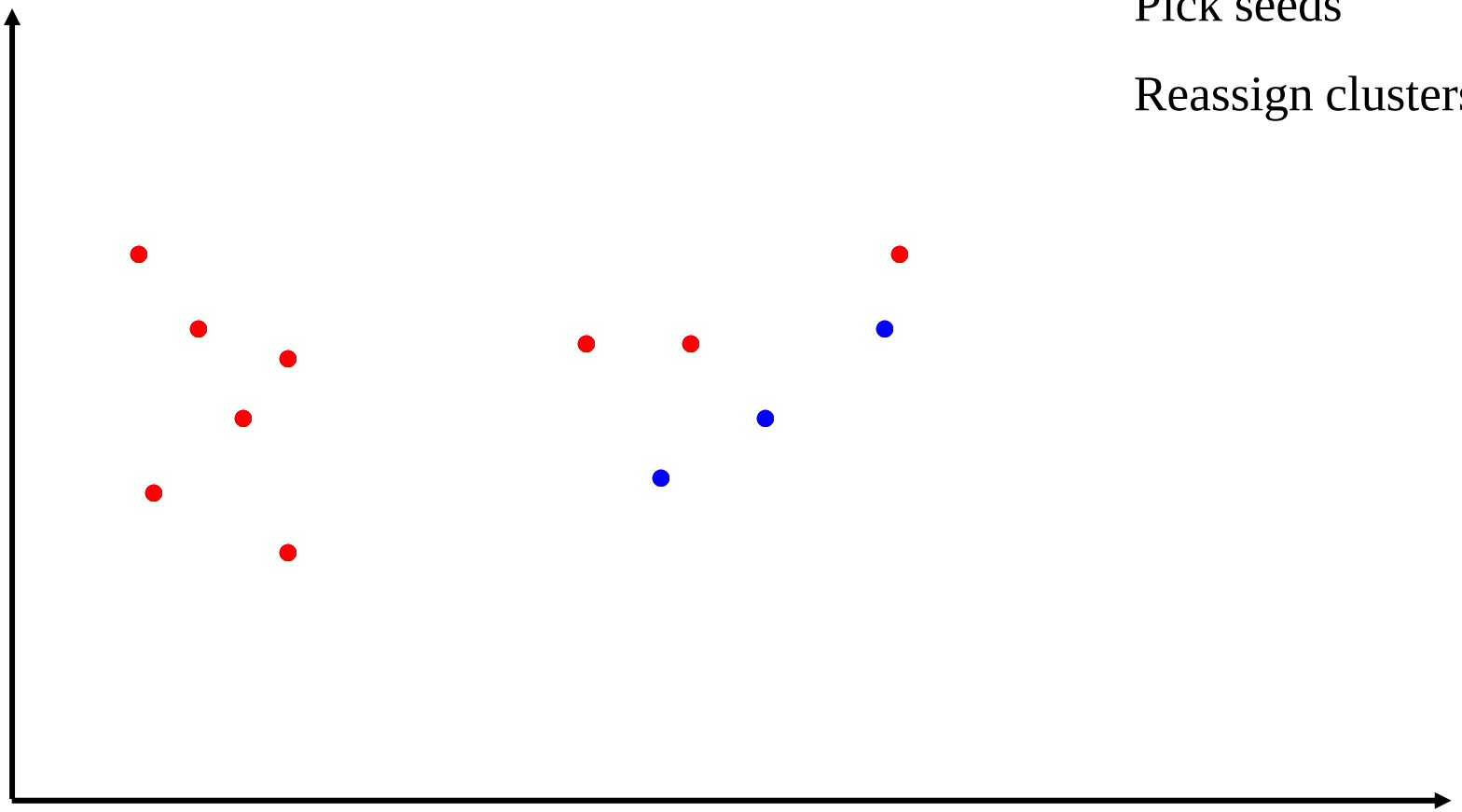
K-Means Example (K=2)



K-Means Example (K=2)



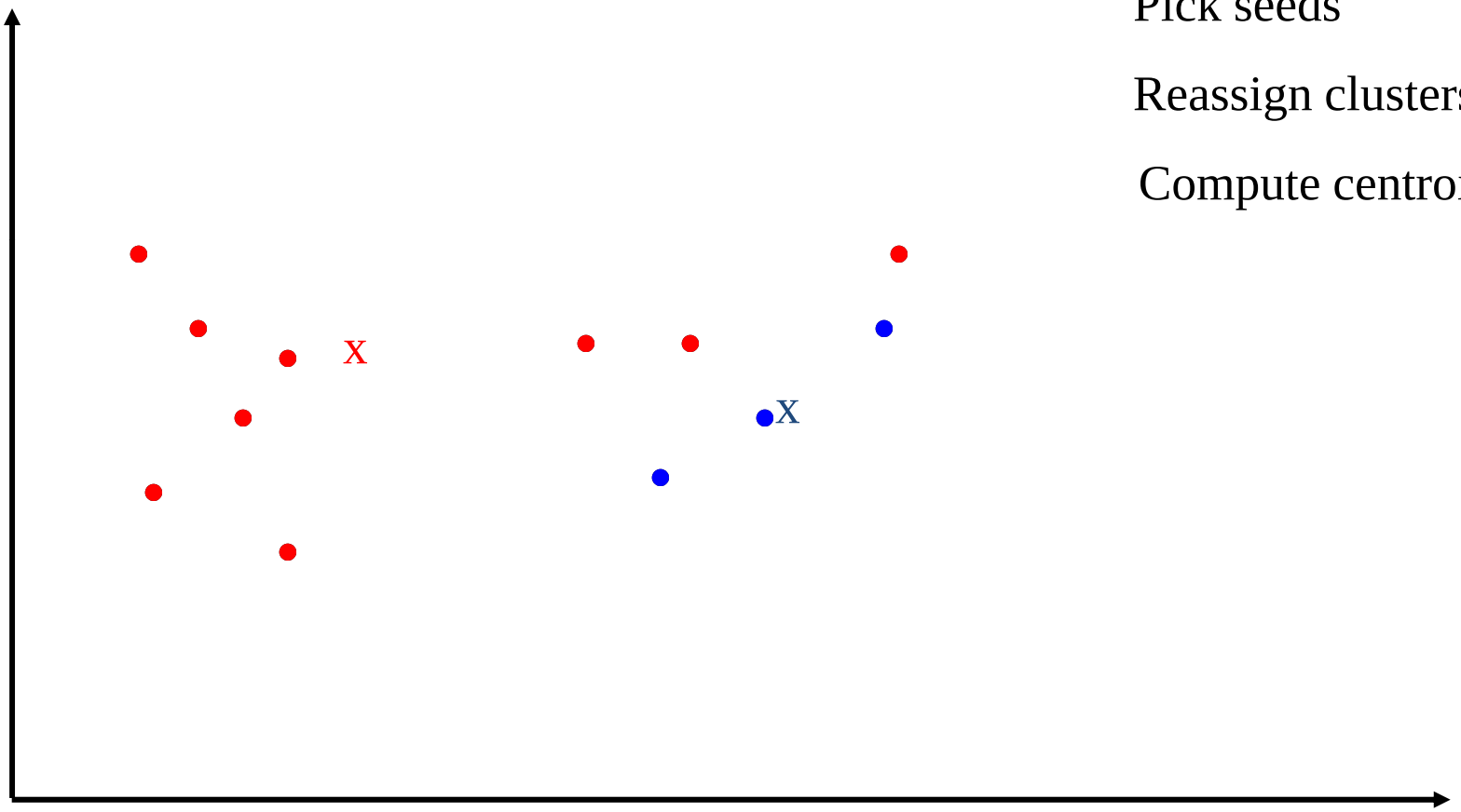
K-Means Example (K=2)



Pick seeds

Reassign clusters

K-Means Example (K=2)

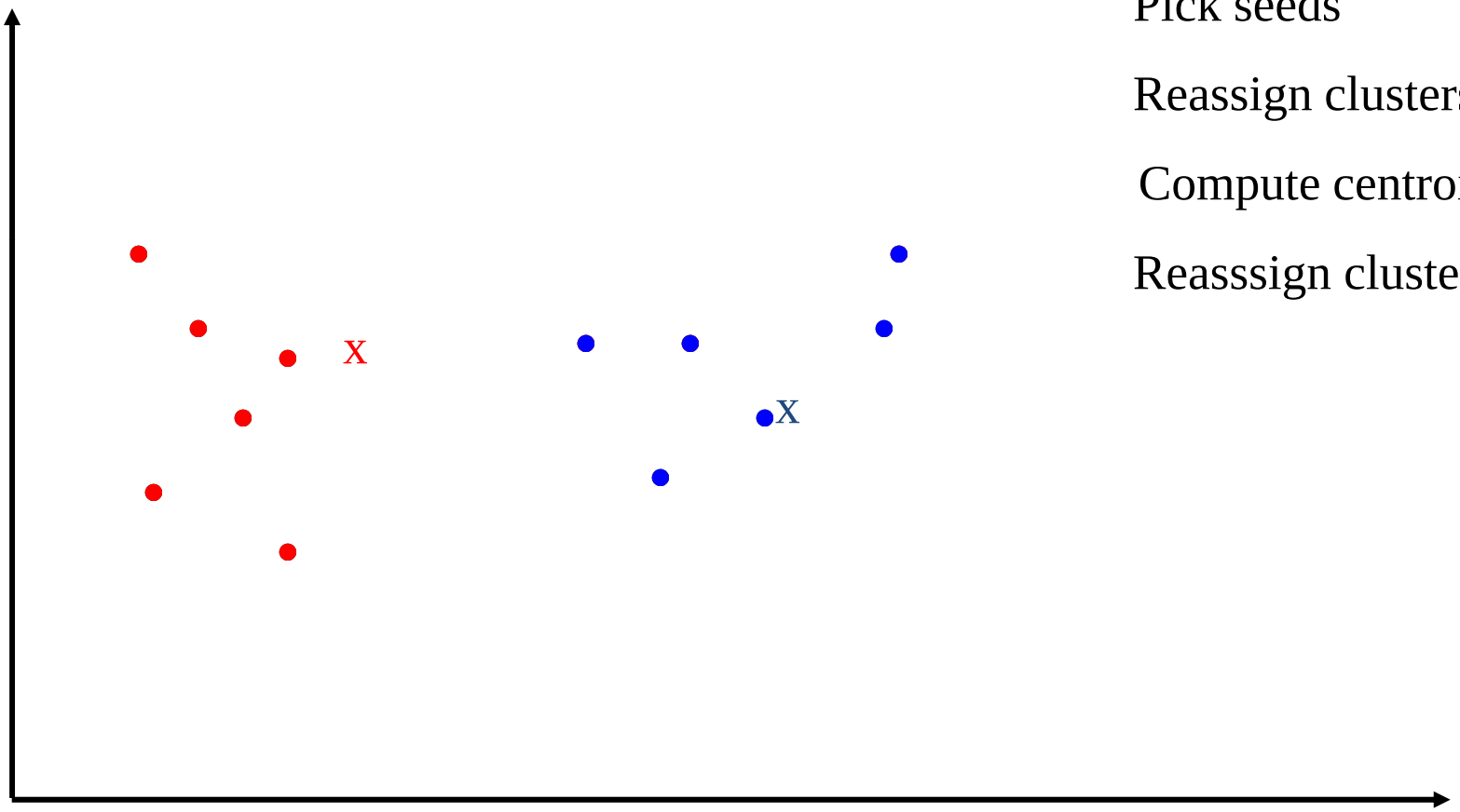


Pick seeds

Reassign clusters

Compute centroids

K-Means Example (K=2)



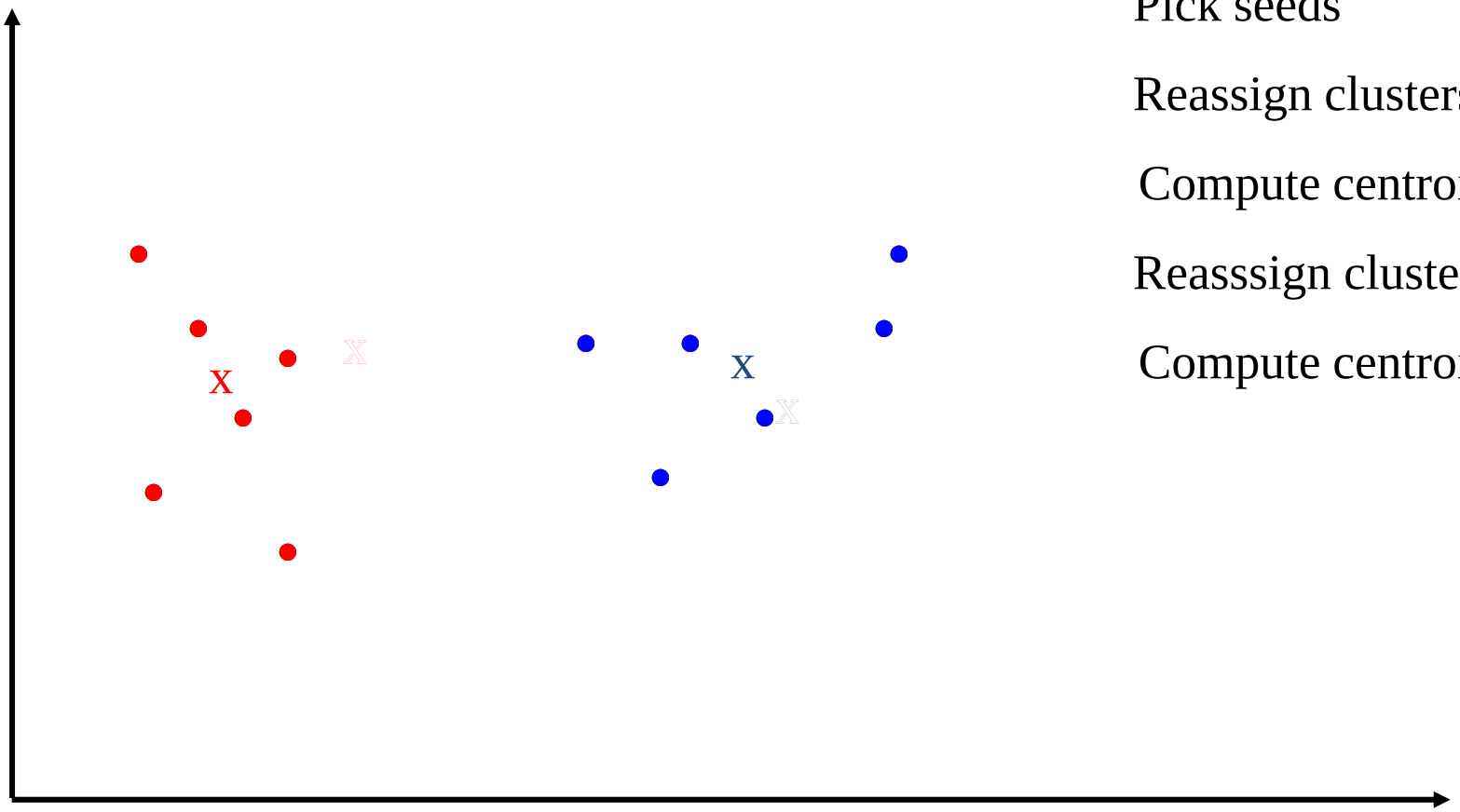
Pick seeds

Reassign clusters

Compute centroids

Reasssign clusters

K-Means Example (K=2)



Pick seeds

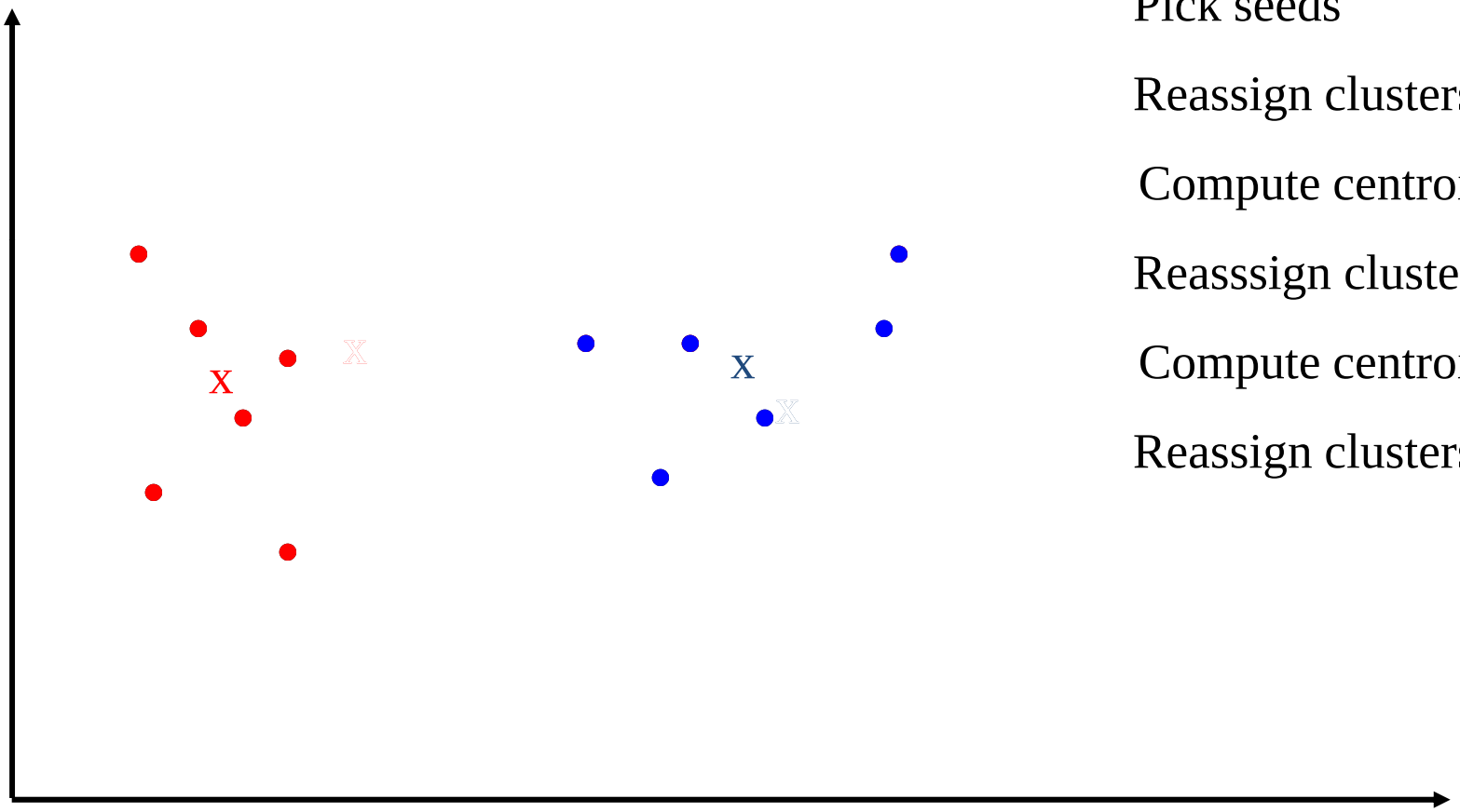
Reassign clusters

Compute centroids

Reasssign clusters

Compute centroids

K-Means Example (K=2)



Pick seeds

Reassign clusters

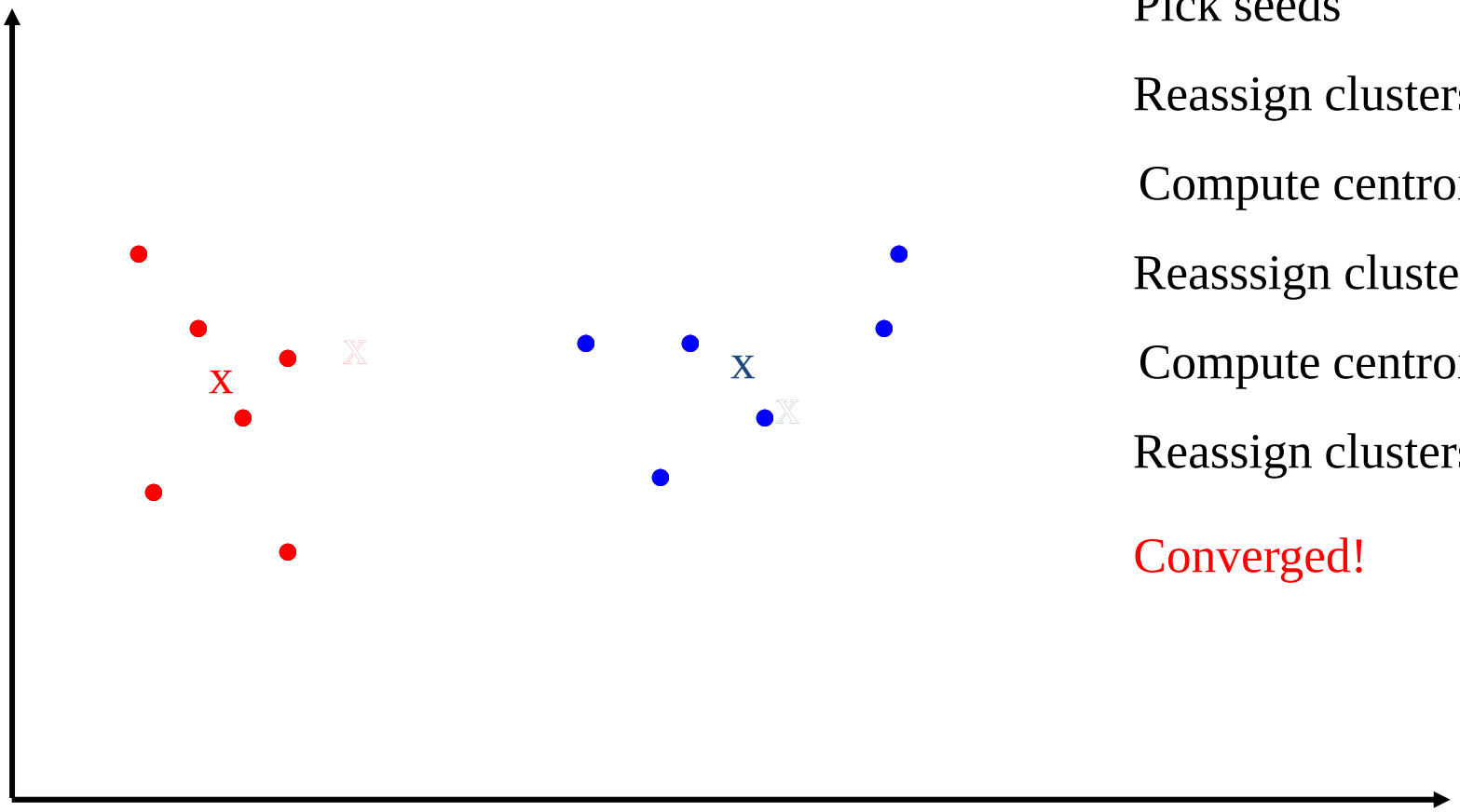
Compute centroids

Reasssign clusters

Compute centroids

Reassign clusters

K-Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reasssign clusters

Compute centroids

Reassign clusters

Converged!

Monotonicity of K-means

- **Monotonicity Property:** Each iteration of K-means strictly decreases the MSE until convergence
- The following lemma is key to the proof:
  - **Lemma:** Given a finite set C of data points the value of μ that minimizes the MSE:

is:

$$J = \sum_{x \in C} ||x - \mu||^2$$

$$\mu = \frac{1}{|C|} \sum_{\mathbf{x} \in C} \mathbf{x}$$

Proof of monoticity

- Given a current set of clusters with their means, the MSE is given by : $$J_e = \sum_{i=1}^{\kappa} \sum_{x \in c_i} ||x - \mu_i||^2$$
- Consider the reassignment step:
  - Since each point is only reassigned if it is closer to some other cluster than its current cluster, so we know the reassignment step will only decrease MSE
- Consider the re-center step:
  - From our lemma we know that μ_i' minimizes the distortion of c_i' which implies that the resulting MSE again is decreased.
- Combine the above two, we know Kmeans always decreases the MSE

Kmeans properties

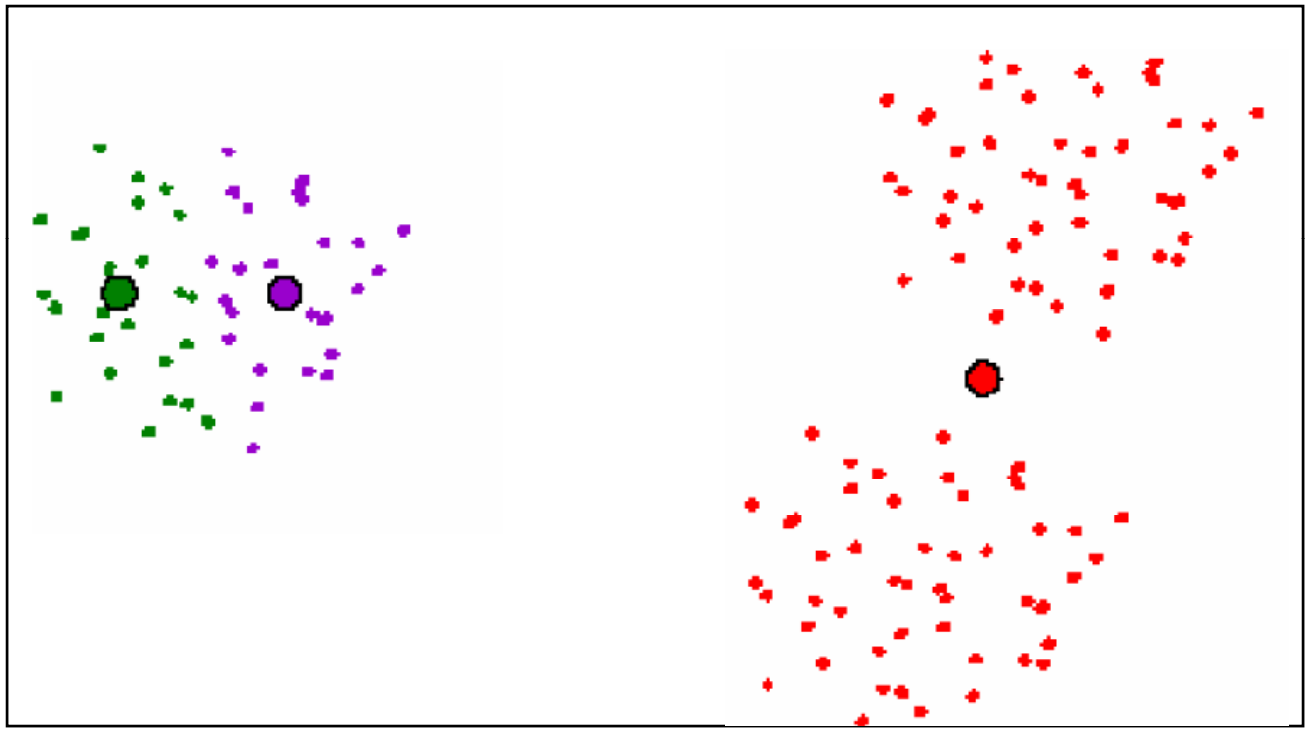
Good news!

- Kmeans always converges in a finite number of steps
  - Typically converges very fast (in fewer iterations than the number of points)
- Time complexity:
  - Assume computing distance between two instances is O(*d*) where *d* is the dimensionality of the vectors.
  - Reassigning clusters: O(*kn*) distance computations, or O(*knd*).
  - Computing centroids: Each instance vector gets added once to some centroid: O(*nd*).
  - Assume these two steps are each done once for *I* iterations: O(*Iknd*).
  - Linear in all relevant factors, assuming a fixed number of iterations, more efficient than O(n^2) HAC.

More Comments

Bad news!

- Highly sensitive to the initial seeds



- This is because MSE has many local minimal solutions, i.e., solutions that can not be improved by local reassignments of any particular points

Solutions

- Run multiple trials and choose the one with the best MSE
  - This is typically done in practice
- Heuristics: try to choose initial centers to be far apart
  - Using furthest first traversal
    - Start with a random initial center, set the second center to be furthest from the first center, the third center to be furthest from the first two centers, and son on
- One can also initialize with results of other clustering method, then apply kmeans
- s

Even more comments

- K-Means is exhaustive:
  - Cluster every data point, no notion of outlier
- Outliers may cause problems, why?
  - Outliers will strongly impact the cluster centers
  - Alternative: K-medoids – instead of computing the mean of each cluster, we find the medoid for each cluster, i.e., the data point that is on average closest to other objects in the cluster

Deciding ***k*** – a model selection problem

- What if we don't know how many clusters there are in the data?
- Can we use MSE to decide k by choosing k that gives the smallest MSE?
  - We will always favor larger k values
- Any quick solutions?
  - Find the knee
- We will see some other model selection techniques later