

A2 In-Class Discussion

Chris Kauffman

*Last Updated:
Tue Feb 24 05:40:50 PM EST 2026*

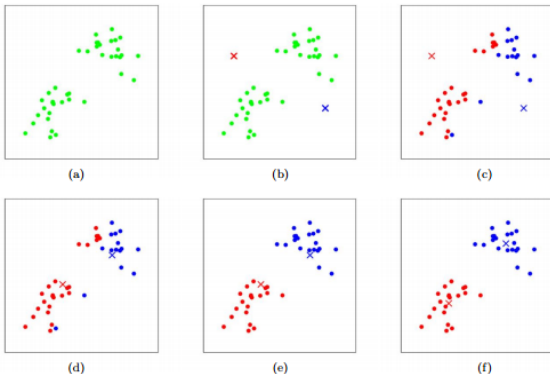
Two MPI Problems

1. Heat Simulation
2. K-means clustering

Previously discussed strategies to parallelize heat problem for distributed memory machine in lecture, will discuss K-Means now

K-Means Clustering

- ▶ A standard ML / Data Mining / Stats problem
- ▶ Input: data + #of clusters desired
- ▶ Output: assignment of each data to a cluster + cluster centers
- ▶ Algorithm: Iterates between
 1. Calculate cluster centers
 2. Calculate cluster assignments



Source: K-Means by Chris Piech. Based on a handout by Andrew Ng.

Overall

```
# ASSIGN RANDOM CLUSTER TO EACH DATA
...

maxiter = 100      # bounds the iterations
curiter = 1        # current iteration
nchanges = ndata   # count changes in assignment each iter

while nchanges > 0 and curiter <= maxiter: # loop until convergence
    # CALCULATE CLUSTER CENTERS
    ...

    # ASSIGN DATA TO CLUSTERS
    ...
```

Initial Assignment to Random Clusters

```
1  for i in range(data.ndata):           # random, regular initial cluster assignment
2      c = i % clust.nclust
3      data.assigns.append(c)
4
5  for c in range(clust.nclust):
6      icount = data.ndata / clust.nclust;
7      extra = 0
8      if c < (data.ndata % clust.nclust):
9          extra = 1                     # extras in earlier clusters
10     clust.counts[c] = icount + extra;
```

Calculate Cluster Centers

```
1 # DETERMINE NEW CLUSTER CENTERS
2 for c in range(clust.nclust):      # reset cluster centers to 0.0
3     for d in range(clust.dim):
4         clust.features[c][d] = 0.0
5
6 for i in range(data.ndata):        # sum up data in each cluster
7     c = data.assigned[i]
8     for d in range(clust.dim):
9         clust.features[c][d] += data.features[i][d]
10
11 for c in range(clust.nclust):      # divide by ndatas of data to
12     if clust.counts[c] > 0:         # get mean of cluster center
13         for d in range(clust.dim):
14             clust.features[c][d] = clust.features[c][d] / clust.counts[c]
```

Assign Data to Clusters

```
1 # DETERMINE NEW CLUSTER ASSIGNMENTS FOR EACH DATA
2 for c in range(clust.nclust):      # reset cluster counts to 0
3     clust.counts[c] = 0
4
5 nchanges = 0
6 for i in range(data.ndata):        # iterate over all data
7     best_clust = None
8     best_distsq = float("inf")
9     for c in range(clust.nclust):  # compare data clusters, assign closest
10         distsq = 0.0
11         for d in range(clust.dim): # calculate squared distance to each dim
12             diff = data.features[i][d] - clust.features[c][d]
13             distsq += diff*diff
14         if distsq < best_distsq:    # if closer to this cluster than
15             best_clust = c          # current best
16             best_distsq = distsq
17 clust.counts[best_clust] += 1
18 if best_clust != data.assigned[i]: # assigning data to a different cluster?
19     nchanges += 1                  # indicate cluster assignment has changed
20     data.assigned[i] = best_clust  # assign to new cluster
```

Distributed Memory Parallel Versions

- ▶ Algorithm deals with Data and Clusters, each a matrixy thing
- ▶ How would you divide up this data in a distributed parallel version?
- ▶ Would data redistribution be required in your scheme?
- ▶ What information needs to be exchanged at each iteration?
- ▶ Do processors need to communicate for the initial cluster assignment? Or can data be assigned to initial clusters without communication?