Region Growing Multiscale Geographical Model

Concept Review

Region Growing Multiscale Geographical Model

- Multiscale Geographical Model adress variability in bandwidth for each location. However fails in accuracy because;
  - The bandwidth is tested for only one sample in each location (case of 2D data). This can be addressed by clustering (adaptive bandwidth) or discrete region approach (fixed bandwidth).
  - The distance is same in each direction. Unfortunately, spatial non-stationarity represented by spatial surface is usually not linear, therefore Multiscale approach is not able to capture it.

Region Growing Multiscale Geographical Model

- Region Growing MultiScale Geographical Model (work title – *ReGrow MuScle*), solves these issues by iteratively looking for optimal region for each location.
- Algorithms consist of clustering, model tuning and prediction.
- The complexness of algorithm is higher, **the computational complexness lower**.
- The complexness of algorithm manifest in number of parameters, and clustering process, which requires visual exploration.

Algorithm

1. Tuning Process
    - **Aim**: For each cluster of training samples find optimal region.
2. Prediction

Algorithm - tuning

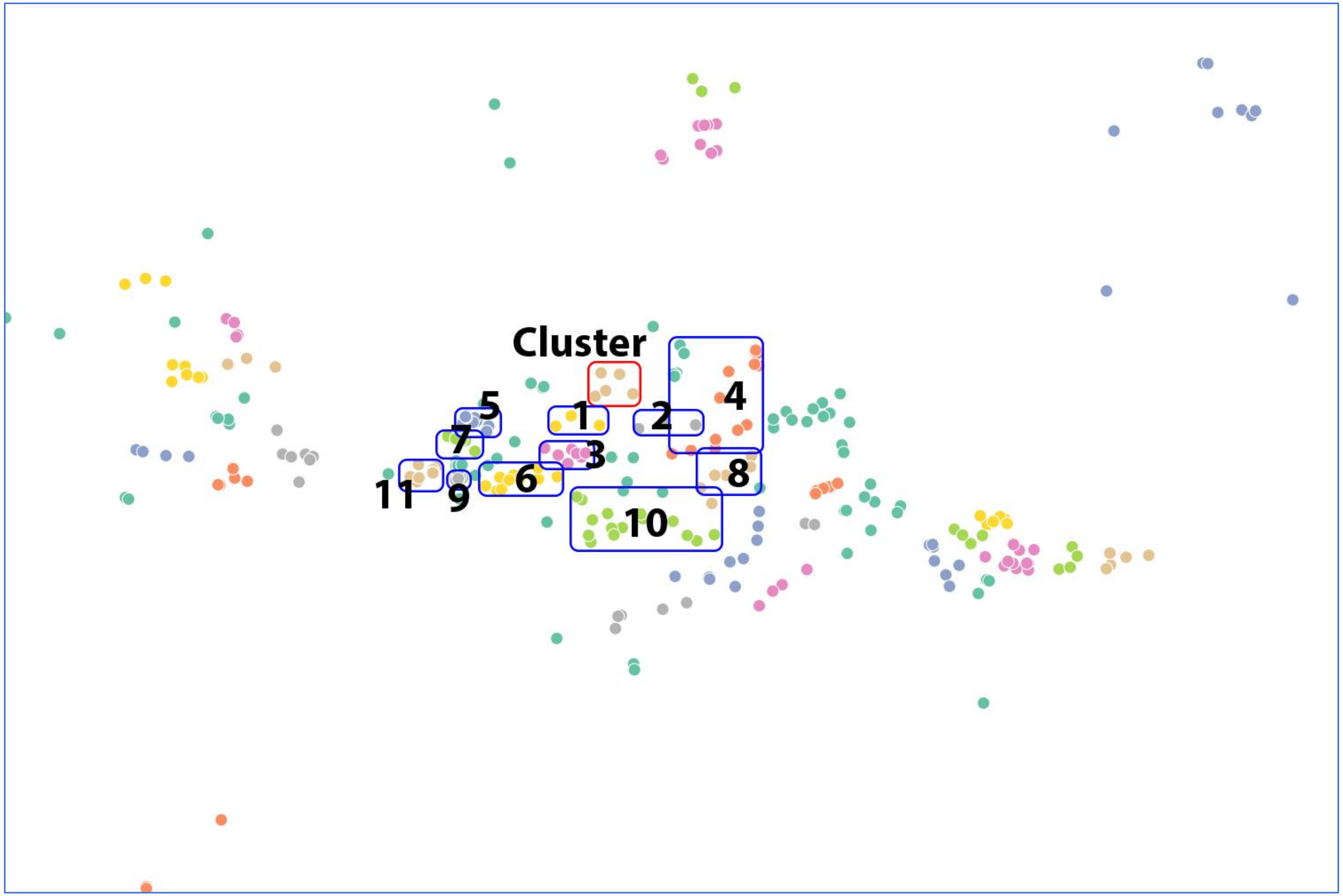
1. Cluster training sample according to their location.
   - Use algorithm DBSCAN or OPTICS or HDBSCAN
   - Important parameters;
     - *Epsilon* – manipulates the distance between points in one cluster
     - *Min cluster size* – minumum points in cluster

Algorithm - tuning

2. Cycle for each cluster label. This will be *core sector.*
3. If number of samples in *core sector* > *max samples*:
   - Sample *max samples* from cluster (sector)
   - These samples will be tested – *core sector test samples*
   - Rest of cluster samples will be used for training

Algorithm - tuning

4. Order other clusters according to their distance to the core sector.
    - Use Euclidean/Manhattan distance to the centroid/medoid/closest point.

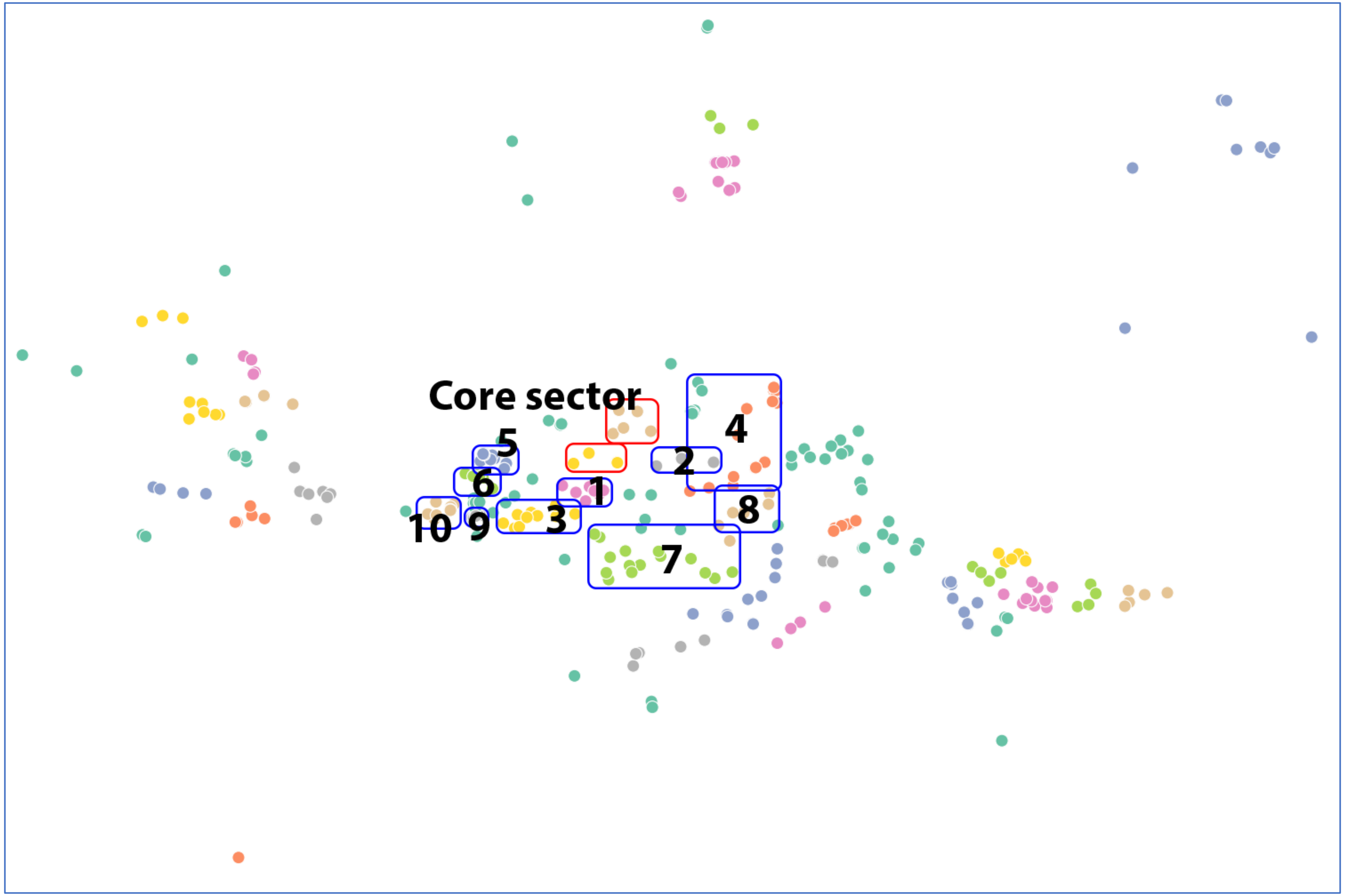

Cluster
1
2
3
4
5
6
7
8
9
10
11

Algorithm - tuning

5. All samples in closest cluster are assigned to the core sector and are used for training. Closest cluster merge into core sector, but its location (centroid) is saved.
6. Model is create and trained on training samples and test on *core sector test samples*. RMSE is calculated – *core RMSE*.

Algorithm - tuning

7. Order the remaining cluster according to the distance of core sector and closest first cluster.
8. Samples from new cluster are used for training of new model with other samples from core sector.



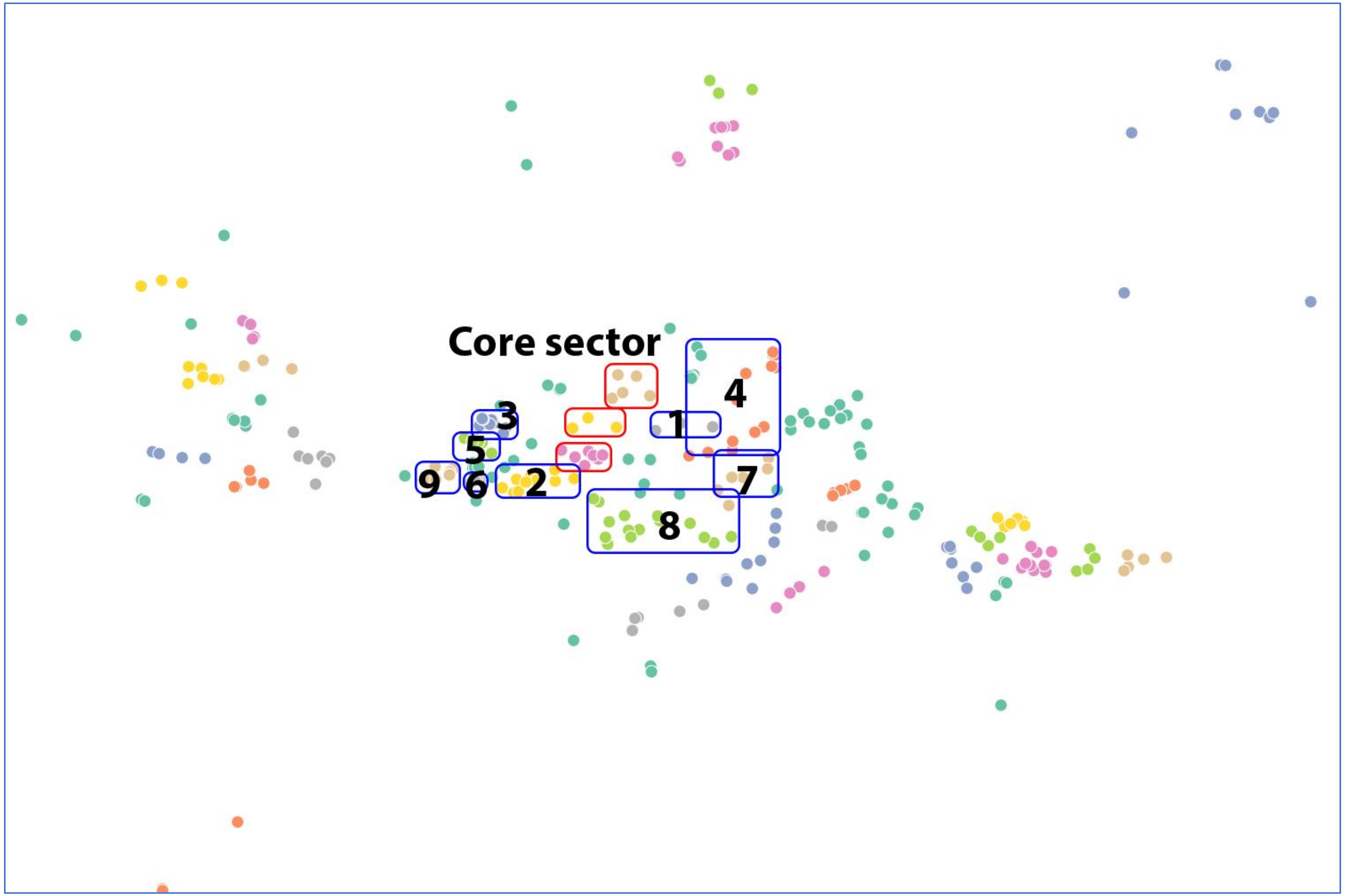
Core sector
1
2
3
4
5
6
7
8
9
10

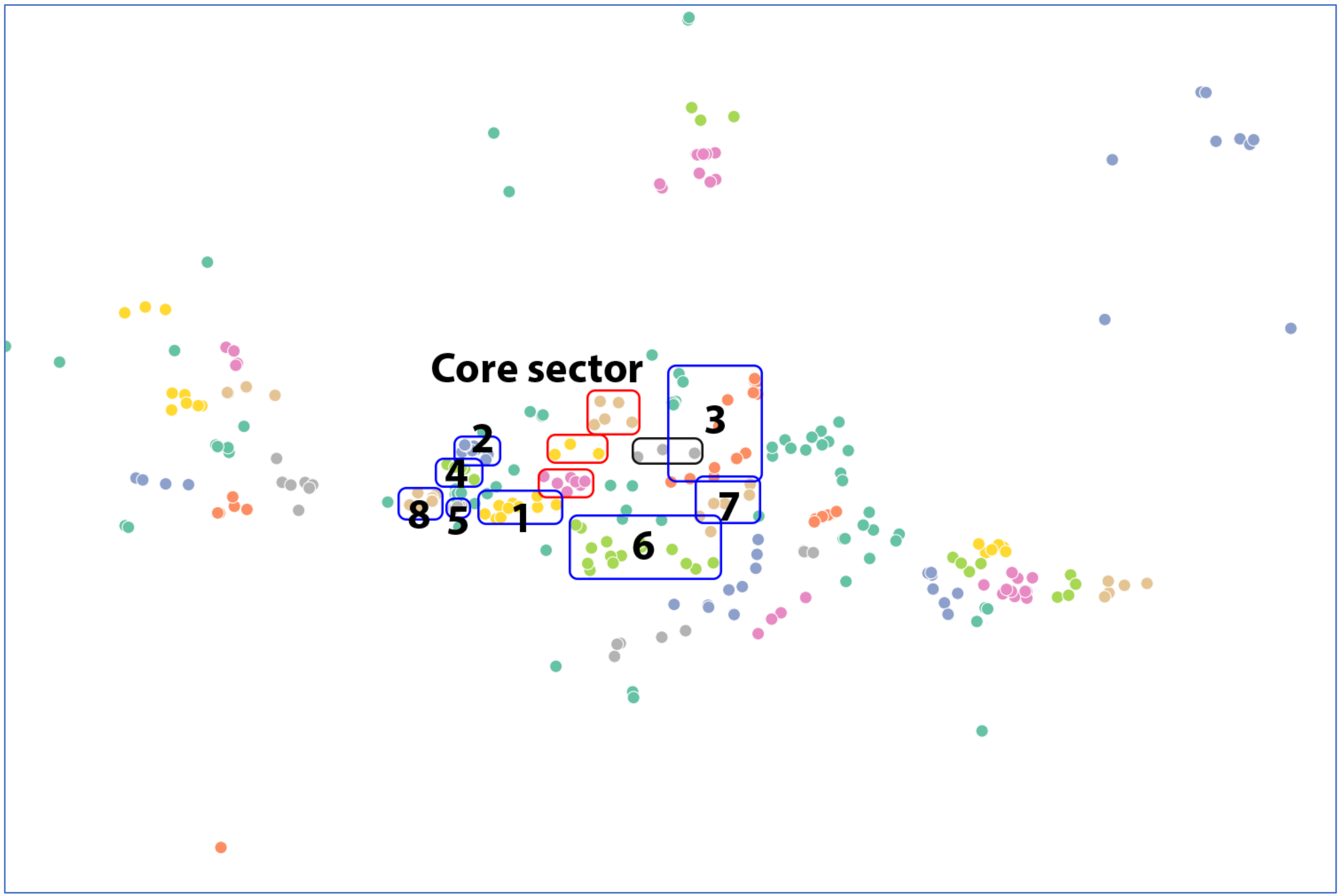
Algorithm - tuning

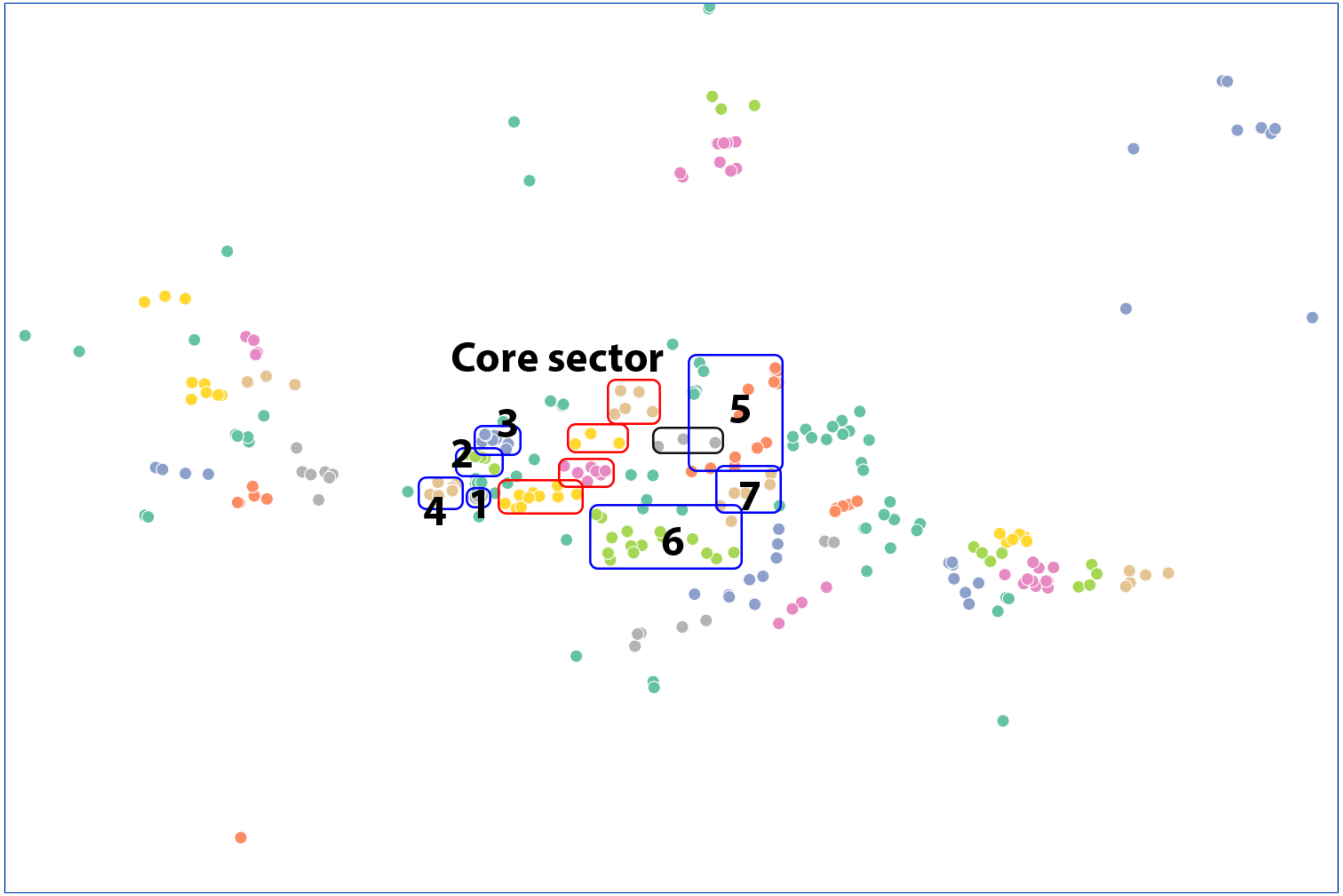
9. Calculate RMSE – *new RMSE*.
   - If new RMSE < core RMSE, assign cluster to core sector.
   - If new RMSE > core RMSE, discard cluster.
10. If new cluster was assigned to core sector, calculate distances and order clusters. If new cluster was not assigned, continue to second closest cluster.

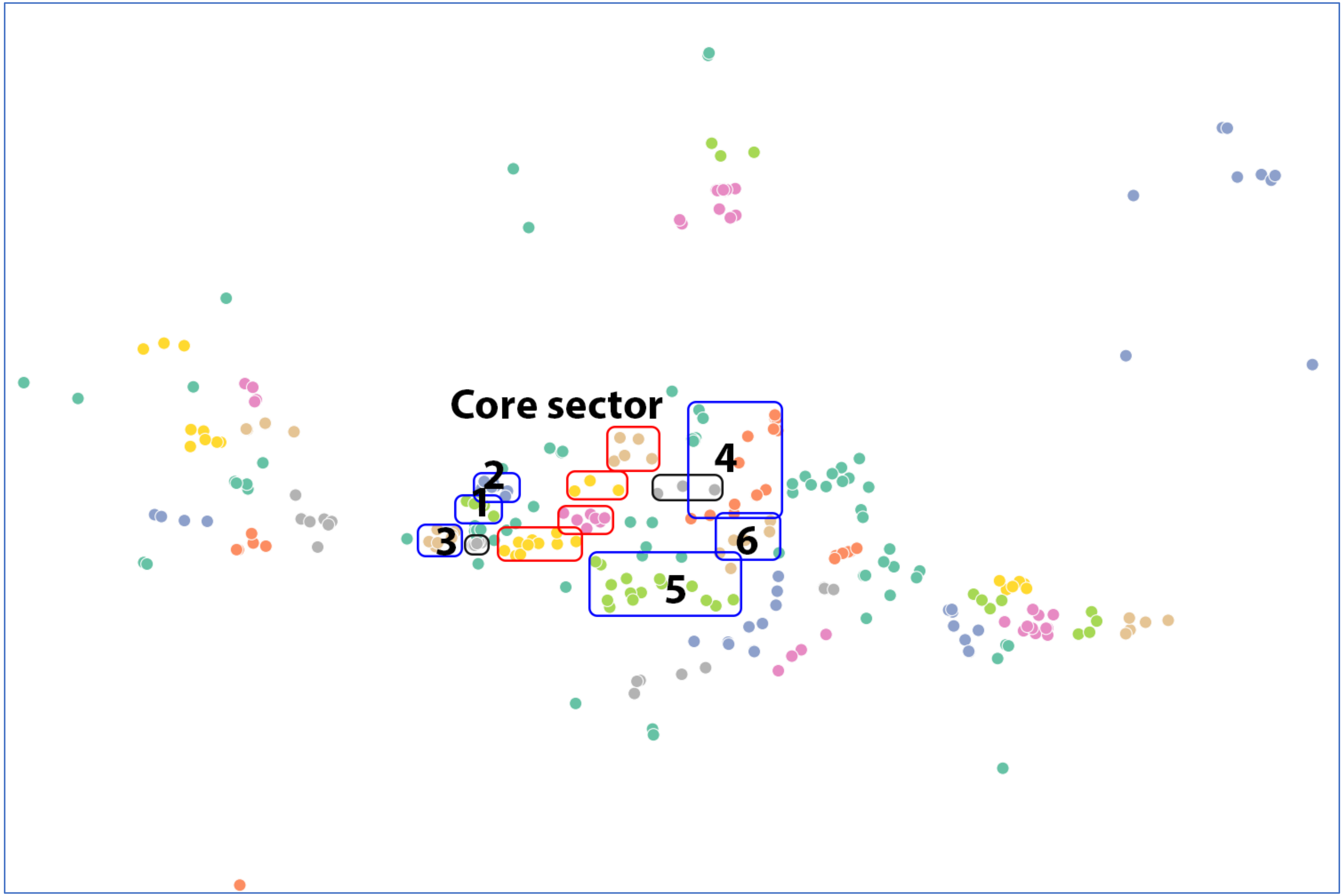
Algorithm - tuning

11. Iteratively continue and stop when;
    - RMSE is not improving.
    - Max iteration *max_iter* is met.
12. Save all cluster labels which incorporated into core sector.


Core sector
1
2
3
4
5
6
7
8
9


Core sector
1
2
3
4
5
6
7
8


Core sector
1
2
3
4
5
6
7



Core sector
1
2
3
4
5
6

Algorithm - tuning

OUTLIERS

- Outliers are trained and tested on a global model.
- Outliers can be included in tuning process for clusters, **not solved how**.
- Number of outliers should be **minimized** during clustering process.

Algorithm - prediction

Find cluster to which testing sample belongs.

- Clustering is unsupervised classification, we cant *predict* a new samples to existing cluster.
- Testing samples are assigned existing cluster according to their distance.

1. For each tested sample, calculate distance to centroid/mesoid of each existing cluster.
2. Closest cluster is assigned to the tested sample if;
    1. The distance is smaller than epsilon. Else;
    2. the sample is outlier.

Algorithm - prediction

3. For each cluster, which consist of tested samples, create a model;