1 Online SSL

- Complete `online_ssl_update_centroids` using the pseudocode 1.
- Complete `online_ssl_compute_solution` following the pseudocode 2

Algorithm 1 Incremental k-centers (simplified)

Input: an unlabeled x_t, a list of centroids C_{t-1}, a list of multiplicities v_{t-1}, taboo list b containing the labeled centroids.
if $(|C_{t-1}| = k)$ **then**
  $c_1, c_2 \leftarrow$ two closest centroids such that at least one of them is not in b.
  // Decide which centroid is c_{rep}, that will represent both c_1 and c_2, and which centroid is c_{add}, that will represent the new point x_t.
  if c_1 in b **then**
    $c_{\text{rep}} \leftarrow c_1$
    $c_{\text{add}} \leftarrow c_2$
  else if c_2 in b **then**
    $c_{\text{rep}} \leftarrow c_2$
    $c_{\text{add}} \leftarrow c_1$
  else if $v_{t-1}(c_2) \leq v_{t-1}(c_1)$ **then**
    $c_{\text{rep}} \leftarrow c_1$
    $c_{\text{add}} \leftarrow c_2$
  else
    $c_{\text{rep}} \leftarrow c_2$
    $c_{\text{add}} \leftarrow c_1$
  end if
  $v_t \leftarrow v_{t-1}$
  $v_t(c_{\text{rep}}) \leftarrow v_t(c_{\text{rep}}) + v_t(c_{\text{add}})$
  $c_{\text{add}} \leftarrow x_t$
  $v_t(c_{\text{add}}) = 1$
else
  $C_t \leftarrow C_{t-1}.\text{append}(x_t)$
  $v_t \leftarrow v_{t-1}.\text{append}(1)$
end if

Algorithm 2 Online HFS with Graph Quantization

Input: t, a list of centroids C_t, a list of multiplicities v_t and labels y.
$V \leftarrow \text{diag}(v_t)$
$[\widetilde{W}_q]_{ij} \leftarrow$ weight between centroids i and j.
Compute the Laplacian L of the graph represented by $W_q = V\widetilde{W}_q V$
// Infer labels using hard-HFS.
$\widehat{y}_t \leftarrow \text{hardHFS}(L, y)$
// Remark: with the preceding construction of the centroids, x_t is always present in the reduced graph and does not share the centroid with any other node.

Some practical considerations:

- The labeled nodes are fundamentally different from unlabeled ones. Because of this, it is always a good idea to keep them separate, and never merge them in a centroid. In the implementation this is accomplished with a taboo list b that keeps track of nodes that cannot be merged together.
- In streaming applications, it is not always possible to stop execution to partition the centroids, and it is often preferable to pay a small price at every step to keep execution smooth. In our case, the centroids are updated at every step.
- Whenever a new node arrives, and we have too many centroids, we choose the two closest centroids c_{add} and c_{rep}. c_{add} will forget the old centroid and will point to the new sample that just arrived, and c_{rep} will take care of representing all nodes that belonged to c_{add}.

References

[1] Moses CHARIKAR, Chandra CHEKURI, Tomas FEDER, and Rajeev MOTWANI. Incremental clustering and dynamic information retrieval. *SIAM journal on computing*, 33(6):1417–1440, 2004.