General Q&A

No questions.

Theory Tasks

Chapter 2: Fundamentals

For data aggregation, is there a relationship between distributive/algebraic/holistic aggregates and self-maintainable aggregates?

- self-maintainability is a property regarding change operations (like insertion) of data
  - basic definition: update aggregate after change only from change and previous value
  - aggregate might be self-maintainable regarding some operations, but not others
- distributive aggregates:
  - can be computed from aggregates of same type determined on data partitions
  - thus, self-maintainable regarding insertion
    - ∗ compute aggregate on set of new point(s)
    - ∗ combine with value of aggregate on existing points
    - ∗ e.g., min, max, count, and sum can be updated with new point(s)
  - sum / count also self-maintainable regarding deletion
    - ∗ subtract values of deleted points / subtract number of deleted points
  - min / max not self-maintainable regarding deletion
    - ∗ no problem if deleted value higher than min / lower than max → aggregate same
    - ∗ problem if deleted value equal to min / max → new value of aggregate unclear
- algebraic aggregates:
  - can be computed from set of distributive aggregates
  - self-maintainability in narrow sense violated (need multiple sub-aggregates)
  - self-maintainable in broad sense (if access to all necessary sub-aggregates)
  - e.g., if sum and number of points stored, can still compute mean after insert/delete
- holistic aggregates:
  - cannot be computed from fixed amount of sub-aggregates (e.g., median, mode)
  - thus, not self-maintainable regarding insertion/deletion

Chapter 7: Clustering

What are ‘convex space partitions’ in the context of k-Means?

- all clustering algorithms typically partition space (at least implicitly)
  - for each point, you can decide to which cluster it belongs
  - for ‘soft’ clustering methods like mixture models, membership is probabilistic
  - for k-Means, points are assigned to closest centroid
  - partitioning can be visualized as Voronoi diagram
- partitions in k-Means are convex, as
  - all points on a hypersphere around centroids have same distance to centroids
  - hyperspheres are convex
  - k-Means attempts to minimize distances to centroids

How can we improve upon the complexity of k-Medoid?

- complexity according to lecture: $O(t \cdot k \cdot (n-k)^2)$
- can reduce number of iterations t, but that might just prevent proper convergence
- main problem: in each iteration,
  - consider each pair of medoid and non-medoid $\rightarrow O(k \cdot (n-k))$
  - compute change in objective function $\rightarrow O(n-k)$
- improvement 1: CLARA
  - naming: 'Clustering LARge Applications'
  - run PAM on sample of size $n' < n$
  - replaces n with n' in complexity estimate
- improvement 2: CLARANS
  - naming: 'Clustering LARge Applications based upon RANdomized Search'
  - use full dataset (so not an extension of CLARA!)
  - randomly select medoid *numlocal* times (instead $t \cdot k$)
  - for medoid, randomly select non-medoid at most *maxneighbor* times (instead $n-k$)
  - overall complexity: $O(numlocal \cdot maxneighbor \cdot (n-k))$
- improvement 3 (not in lecture): FasterPAM
  - same results as PAM, but $O(k)$ speedup due to improved medoid swapping

How does OPTICS work?

- density-based clustering algorithm
- naming: 'Ordering Points To Identify the Clustering Structure'
- usually for numeric data, but actually combinable with arbitrary dissimilarity measure
- hyperparameters:
  - $MinPts$: min number of points in neighborhood to make point *core* (as in DBSCAN)
  - ϵ: max neighborhood radius (different to DBSCAN, despite same name)
- output: no clustering, but ordered list of points with core/reachability distances
  - can be used to create reachability plot
  - can be used to create fixed, DBSCAN-like clustering (by choosing some $\epsilon^* < \epsilon$)
  - in particular, can extract clusterings with different density thresholds
  - in particular, can use reachability plot to select a sensible ϵ^*
- important definitions:
  - core distance: min neighborhood radius of a point to make it *core*
  - reachability distance: min distance such point directly density reachable from other
    - at least actual distance between the two points
    - at least core distance of second point
- procedure:
  - process each point exactly once
    - retrieve neighbors and compute core distance
    - write point to output (it is processed now)
    - check if point *core*; if yes, update reachability distances of unprocessed neighbors
  - maintain priority list
    - points ordered by minimum reachability distance to all points in output
    - process points in this order (other than DBSCAN, which uses arbitrary order)
- complexity: $O(n \cdot \log n)$ with spatial index structure for neighborhood queries
  - also, ϵ must not be not too high for efficient neighborhood queries
  - $O(n^2)$ with naive implementation (neighborhood queries in $O(n)$)

How does k-Mode work?

- partitioning-clustering algorithm for categorical data
- hyperparameter: number of clusters k
- output: cluster centroids, cluster assignments (latter can be computed with former)
- complexity: $O(t \cdot k \cdot n)$ with t iterations and n points
- procedure similar to k-Means, i.e., repeatedly
  - assign points to their closest centroids
  - re-compute centroids from points assigned to them
- adaptation 1: Hamming distance instead of Euclidean distance
  - reason: data is categorical instead of numeric
  - thus, cannot say how different points are, only if attribute values are different or not
  - Hamming distance sums (binary) differences over all attributes
- adaptation 2: centroid computation
  - for categorical data, cannot compute mean
  - instead, take the mode (most frequent value) of each attribute separately
  - centroid might not be an actual point from dataset (but applies to k-Means as well)

Chapter 8: Outlier Detection

Why should outlier detection be considered unsupervised?

- outlier-detection problems (datasets) typically don't have class labels
  - by definition, that's unsupervised learning
  - in literature, also outlier detection with partially or fully labeled datasets
- outliers don't form a clearly defined target
  - are rare
  - are diverse: usually not only distinct from 'normal' data, but also other outliers
  - i.e., no unique concept/distribution behind them, so characterization difficult

For k-distance-based outliers, what are the dangers of setting k very low or very high?

- k very low: doesn't recognize outliers if close to a few other outliers
  - e.g., for $k = 1$, two outliers close to each other won't be recognized
- k very high: small clusters in data (distant from remaining data) recognized as outliers
  - e.g., imagine $k = 50$ and a cluster of 40 points well-separated from remaining points
- in both examples, depends on use case if you want to label mentioned points as outliers

How do Restricted Boltzmann Machines work for outlier detection?

- two-layer neural networks
  - 'visible' and 'hidden' neurons connected to each other, but not among themselves
- try to learn probability distribution of training data
  - various hyperparameters control training, e.g., number of iterations
  - model (parameters): weight matrix between neurons, biases of neurons
- to score a point as outlier, put it in visible layer and compute 'free energy'
  - says how unlikely point is, given the learned probability distribution (weights)
- compared to other neural-network approaches (Autoencoders, Self-Organizing Maps):
  - architecture different
  - scoring idea (difference to some learned representation of data) roughly similar