1

Clustering

CPS 296.3: Information Management and Mining

Jun Yang
Duke University
February 10, 2008

† Thanks to contents borrowed from Ullman (http://infolab.stanford.edu/~ullman/mining/mining.html), Han (http://www.cs.uiuc.edu/homes/hanj/bk2/slidesindex.html), and Kumar (http://www-users.cs.umn.edu/~kumar/dmbook/)

2

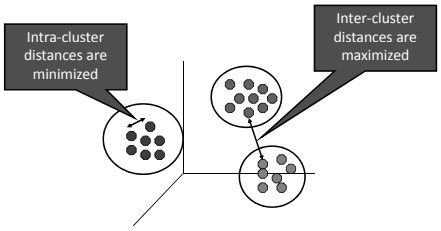
Announcements

- Email me suggestions of papers/topics for the seminar-style portion of the course
  - To begin in two weeks
- Selection of papers/topics by next lecture
- Assignment of discussion leaders next week

3

The problem of clustering

- Finding groups of objects such that objects in one group are "similar" to one another and "dissimilar" from objects in other groups



4

Example: city planning

- Describe each house by a list of attributes
  - Type: nominal (categorical)
  - Location: numeric, 2 dimensions
  - Value: numeric
  - Mortgage: numeric
  - Flood zone: binary
  - etc.
- Cluster houses based on similarity among their attribute values

5

Example: SkyCat

- A catalog of 2 billion "sky objects," each represented by radiation levels in 7 frequency bands
  - Sloan Sky Survey is a newer, better version
- Cluster objects into groups that may (or may not) correspond to their types, e.g., galaxies, nearby stars, quasars, etc.

6

Example: clustering CD's

- Intuitively: music divides into categories, and customers prefer a few categories
  - But what are categories really?
- Similar CD's have similar sets of customers who bought them, and vice versa $\rightarrow$ collaborative filtering
- Represent each CD by the set of purchasers
  - A space with one dimension for each customer, with values 0 (did not buy) and 1 (bought)
- To many dimensions?
  - ➲Minhashing (LSH)

Example: clustering docs

- Represent each doc by a binary vector, where the i-th component is 1 iff the doc contains the i-th word
- Docs with similar sets of words belong to the same cluster (and may be about the same topic)
- Again, use LSH to reduce # of dimension

Example: clustering DNA's

- Each DNA sequence is a sequence of characters from {C, A, T, G}
- Measure similarity between sequences by edit distance—the minimum # of inserts and deletes needed to turn one into the other

➲Note there is a "distance," but no convenient space in which points "live"

Input data

- n objects, p dimesions
- Option 1:
  $n \times p$ data matrix +
  distance function
- Option 2:
  $n \times n$ distance matrix

Distance measures

- d is a distance measure (metric) if it is a function from pairs of objects to real numbers such that
  - $d(x, y) \geq 0$
  - $d(x, y) = 0$ iff $x = y$
  - $d(x, y) = d(y, x)$
  - $d(x, y) \leq d(x, z) + d(z, y)$ ➲ triangle inequality

Edit distance

- The edit distance between two strings is the # of inserts and deletes of characters needed to turn one into the other
  - Equivalently: $d(x, y) = |x|+|y|-2|\text{LCS}(x, y)|$
    - LCS = longest common subsequence = (any) longest string obtained both by deleting from x and deleting from y
  - E.g., if x = *abcde*, y = *bcduve*, then LCS(x, y) = *bcde*, $d(x, y) = 3$

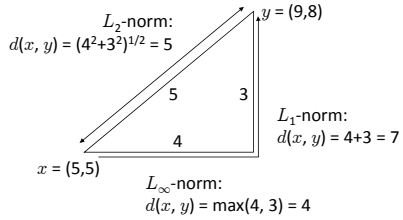
➲Why does edit distance satisfy triangle inequality?

- Extensions: allow mutate, substring reversal, etc.

L_p norms

- $d(\mathbf{x}, \mathbf{y}) = (\sum_f |x_f - y_f|^p)^{1/p}$
  - A.k.a. Minkowski distance
- L_2 norm (Euclidean distance): square root of the sum of the squares of the differences between $\mathbf{x}$ and $\mathbf{y}$ in each dimension
  - The most common notion of "distance"
- L_1 norm (Manhattan distance): sum of the differences in each dimension
  - I.e., distance if you have to travel parallel to axes
- L_∞ norm: max. of the differences in any dimension
  - I.e., limit when $p \to \infty$

L_p norms illustrated



Unit circles in 2d (source: Wikipedia):

Standardization

Necessary when dimensions have different scales

For each dimension f:

- Calculate the mean absolute deviation
  - $s = (\sum_i |x_i - m|)/n$, where $m = (\sum_i x_i)/n$
  - ➲How about the standard deviation?
- Calculate the standardized measurement
  - $z_i = (x_i - m)/s$
- Then, use z_i's instead of x_i's for dimension f

Points or vectors?

- Recall TF-IDF
  - Each doc is a vector of weights, each corresponding to the TF-IDF product for a term
  - Text similarity between two docs = normalized dot product of their vectors
  - A variation: similarity = angle between the vectors = arc cos $(\mathbf{v}_1 \cdot \mathbf{v}_2 / (|\mathbf{v}_1||\mathbf{v}_2|))$
    - ➲Does this satisfy triangle inequality?
    - ➲Is this a metric?

➲How does this metric compare with L_p norm by treating vectors as points?

Metrics for binary vectors

Is the binary variable symmetric?

- Symmetric: male vs. female
  - Simple matching coefficient: $(n_{01}+n_{10})/n$
  - ➲Why does this satisfy triangle inequality?
- Asymmetric: containing term i vs. not containing term i; HIV positive vs. HIV negative
  - Jaccard distance = 1 – Jaccard similarity = $(n_{01}+n_{10})/(n_{01}+n_{10}+n_{11})$
  - ➲Why does this satisfy triangle inequality?

Nominal variables

- Consider vectors of nominal (categorical) variables
  - E.g., a variable's value is one of {red, yellow, green}
- Option 1: simple matching distance = 1 – (# of matches)/(# of variables)
- Option 2: replace each nominal variable with a bunch of binary variables, one for each possible value

Ordinal variables

- Consider an ordinal variable
  - I.e., the ordering is important, not the actual value
  - E.g.: ranks (1, 2, 3, …), medals (gold, silver, bronze)
- Say there are m possible values v_1, v_2, …, v_m in ranked order
- Map value v_j to number $(j - 1)/(m - 1)$, in [0, 1]
- Then treat the dimension as one in an Euclidean space

Combining different types

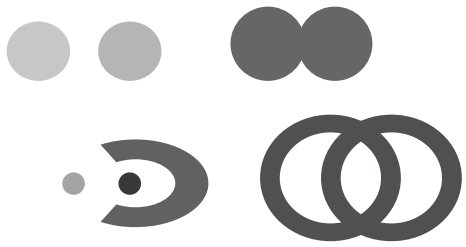
- What if the dimensions have different types?
  - Numeric, symmetric/asymmetric binary, nominal, ordinal, etc. *Trick Q: what type is a minhash value?*
- One could use a weighted formula:
  $d(i, j) = (\sum_f \delta_{ij}^{(f)} d_{ij}^{(f)}) / (\sum_f \delta_{ij}^{(f)})$
  - $\delta_{ij}^{(f)} = 0$ if f is asymmetric binary and $x_{(i, f)} = x_{(j, f)} = 0$, or 1 otherwise
  - If f is binary/nominal: $d_{ij}^{(f)} = 1$ if $x_{(i, f)} = x_{(j, f)}$, or 0 otherwise
  - For other types of f: $d_{ij}^{(f)}$ = standardized or normalized quantity

Clustering as optimization

- Partition a set of points such that a certain objective function is minimized
  - Objective functions vary depending on the apps
- Generally tough
  - E.g.: given a set of points in R^d, can you find k balls of radius r that cover all of them (k-center)?
    - NP-hard for $d \geq 2$ if either d or k is part of the input
    - Approximable within 2 but no better

➲Many approaches, with different objectives and heuristics

In the eyes of beholder



Roadmap for this lecture

- Partitioning approach
  - k-means, k-medoids/PAM, CLARA
- Hierarchical approach
  - BIRCH, CURE, Chameleon
- Density-based approach
  - DBSCAN
- Connection to machine learning
  - BFR

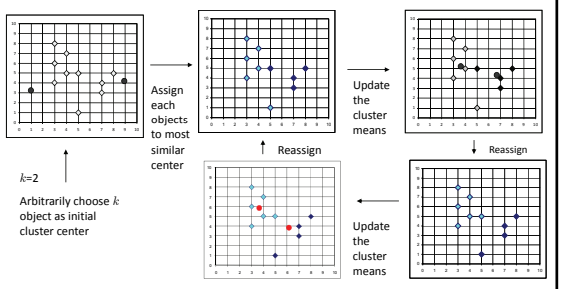
k-means

- Objective: place k centers to minimize the variance between points and the centers they assigned to
- Key insight
  - Given the partitioning of points, we can easily pick the optimal center location for each subset
    - The centroid: $(\sum_i \mathbf{x}_i)/n$
  - Given the centers, we can easily find the optimal partitioning of points
    - The closest center

The k-means algorithm

1. Pick k (cluster) centers
   - E.g.: pick one at random, then each other point as far away as possible from the previous points
2. Assign points to clusters—a point always goes with the closest cluster center
3. Compute the new centers as centroids of the clusters
4. Go back to 2, or stop when the partitioning no longer changes

k-means example

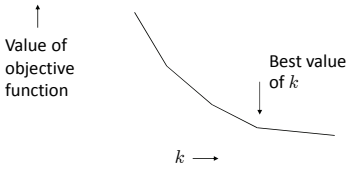


k-means discussion

- Biggest strength: efficiency
  - $O(kn)$ per iteration
  - # of iteration low in practice, although may be exponential in the worst case
- Biggest weakness: possibility to stop at local minimum
- Other issues
  - Applicable only when "mean" is defined, but what about nominal data?
  - How do you know k in advance?
  - Unable to handle noisy data and outliers
  - Unable to find clusters with non-convex shapes

A method for picking k

- Try different k, looking at the change in the value of the objective function as k increases
- This value falls rapidly until the right choice of k, and then changes less



k-medoids

- Instead of using the centroid to represent the cluster, use the medoid—the most centrally located object in the cluster
  - A medoid serves as the reference object from which we measure distance to other objects in its cluster ➲ change in problem definition
- Addresses two issues
  - No longer requires the notion of "mean"
  - Less sensitive to outliers

PAM

Partitioning Around Medoids

[Kaufman & Rousseeuw, 1987]

1. Select k objects (as candidate medoids)
   - Assign an object to the closest selected object
2. For each pair of selected and non-selected objects, calculate the benefit of swapping them
3. Pick the pair that offers the biggest benefit, do the swap, and go back to 3; or stop when no pair offers positive benefit

From PAM to CLARA

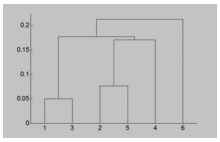
- Scalability of PAM is an issue
  - $O(k\,(n-k)^2)$ per iteration
- CLARA (Clustering LARge Applications) [same authors, 1990]
  - Draw multiple samples of the data set
  - Apply PAM to each sample
  - Give the best clustering as the output

➲Quality depends on the samples

➲Efficiency depends on sample size

Hierarchical clustering

- Produces a set of nested clusters organized in a hierarchy (tree)
- Can be visualized as a dendrogram
  - A tree-like diagram that records the sequences of merges or splits



Strengths

- No need to guess k
  - Desired # of clusters can be obtained by "cutting" the dendrogram
- The hierarchy may actually correspond to something meaningful
  - E.g., biological classification, phylogeny

Approaches

- Agglomerative (bottom-up)
  - Start with the objects as individual clusters
  - In each step, merge the "nearest" pair of clusters until only one is left (or until k are left)
- Divisive (top-down)
  - Start with one cluster with everything
  - In each step, split a cluster until each cluster contains one object (or until there are k clusters)

➲Key question: what is the "nearness" of clusters, if they each contain multiple objects?

Inter-cluster similarity

- Distance between centroids
- Distance between medoids
- Single link: min. distance between a pair of objects (one from each cluster)
- Complete link: max. dist. between a pair of objects
- Average: avg. dist. between a pair of objects
- Ward's method: increase in sum-of-squares error when two clusters care combined
  - Intuitively, want to reduce variance within a cluster
- Etc., etc.

Weakness

- Scalability is an issue—in many cases, $O(n^3)$: n steps, each looking at all pairs
- Once a decision is made to combine two clusters (or to split one), it cannot be undone
- What is the right choice of inter-cluster similarity?
- No objective function is directly minimized

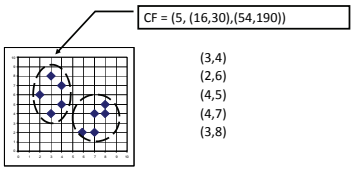
BIRCH [Zhang, Ramakrishnan, Livny, SIGMOD '96]

Balanced Iterative Reducing & Clustering Using Hierarchies

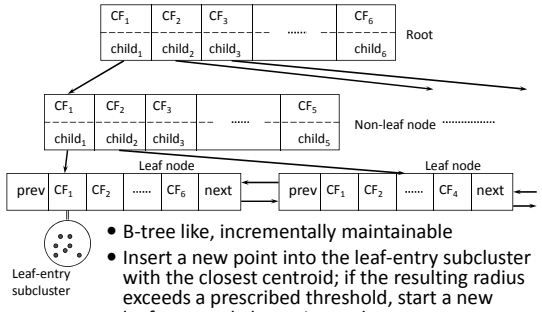
- Goal: deal with data that does not fit in memory
  - Also good for incremental clustering of new objects
- Works with p-dimensional numeric data
- Phase 1: scan db to construct a CF (Clustering Feature) tree—a multi-level compression of the data that tries to capture the clustering of data
- Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

CF in BIRCH

- CF: summary statistics for a subset of points
  - N (scalar): # of points
  - $\mathbf{LS}$ (p-vector): component-wise sum of all points
  - $\mathbf{SS}$ (p-vector): component-wise square sum of all points



CF-tree in BIRCH



- B-tree like, incrementally maintainable
- Insert a new point into the leaf-entry subcluster with the closest centroid; if the resulting radius exceeds a prescribed threshold, start a new leaf-entry subcluster instead

Computing radius from CF

- Centroid $\mathbf{c} = \mathbf{LS} / N$
- Radius = sqrt($\sum_i \|\mathbf{x}_i - \mathbf{c}\|^2/N$)
- Radius^2 = ($\sum_i \sum_d (x_{i,d} - c_d)^2$) / N
  = $\sum_d$ [($\sum_i (x_{i,d} - c_d)^2 / N$)] — Variance of coordinates for dimension d
  = $\sum_d (SS_d/N - c_d^2)$
  = ($\sum_d SS_d$)/N – ($\sum_d c_d^2$)
  - Could have done it by just keeping the sum of all squares instead of the p-vector $\mathbf{SS}$

➲Other things needed for clustering calculation can be derived from N, $\mathbf{LS}$, and $\mathbf{SS}$ as well

More BIRCH discussion

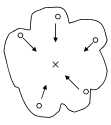
- What if the CF-tree is too big for memory?
  - Raise the radius threshold
  - Build a new tree from the old tree's leaf entries, merging them when appropriate
    - No need to read detailed data again
- Very efficient: one scan is enough
  - Optionally, additional scans can be used to further improve clustering
- Do you trust CF-tree leaf entries to be indivisible?
- Result depends on insertion order

CURE [Guha, Rastogi, Shim, SIGMOD '98]

Clustering Using REpresentatives

- Single link is susceptible to noise/outliers
- Complete link/average/centroid/medoid tend to favor spherical shapes

➲Key idea: use multiple points to represent a cluster—pick well separated ones, and then "shrink" them towards the cluster center



- Cluster similarity = similarity of the closest pair of representative points from different clusters

CURE algorithm

- Draw a random sample S to cluster
- Run the improved hierarchical clustering algorithm (with the new representative scheme) on S
- With the clusters, label data on disk
  - Each cluster again has multiple representatives
  - Assign a data point to the cluster with the closest representative

➲For more efficiency, partition S, cluster each partition, and then merge

Chameleon [Karypis, Han, Kumar, IEEE Computer '99]

- Preprocessing: distance matrix $\rightarrow$ graph
  - There is an edge between two objects iff one is among the top-k most similar objects of the other
  - ➲Sparser region means longer edges
    - ➲Assign smaller weights to longer edges
- Phase 1: use a multilevel graph partitioning algorithm to find many well-connected clusters
  - They are "indivisible" (contained in "true" clusters)
- Phase 2: use agglomerative clustering to merge subclusters

Chameleon cluster similarity

Two key properties

- Relative interconnectivity = absolute interconnectivity between clusters normalized by the internal connectivity of the clusters
  - Interconnectivity $\approx$ edge cut
- Relative closeness = absolute closeness between two clusters normalized by the internal closeness of the clusters
  - Closeness $\approx$ average weight of edges in the cut

Density-based clustering

- Change in perspective: clusters are dense regions of objects in the data space, separated by regions of low density
- Features
  - Can find clusters of arbitrary shape
  - Handle noise gracefully
  - Need density parameters as termination condition

DBSCAN [Ester, Kriegel, Sander, Xui, KDD '96]

- Two parameters
  - Eps: radius that defines neighborhood of interest
  - MinPts: if an object's Eps-neighborhood size $\geq$ MinPts, then the object is a core object
- A point p is directly density-reachable from q if
  - q is a core object
  - p is in the Eps-neighborhood of q
- A point p is density-reachable from q if there exists a chain of points between them where each point is directly density-reachable from the previous
- Points p, q are density-connected if they are both density-reachable from some point
- Cluster = maximal set of density-connected points

Clustering as model fitting

- Clustering = identifying dense regions of data
- Represent the probability density function using a mixture model with k component density functions f_1, …, f_k
  - $p(x) = \sum_h w_h f_h(x \mid \phi_h)$, where $\sum_h w_h = 1$
  - Think of each component as describing a cluster
  - w_h: fraction of points in cluster h
  - ϕ_h: parameters of cluster h
    - For Gaussian, they would include the mean (center location) and the covariance matrix (cluster shape)

➲Assignment of data to clusters is now probabilistic

Learning the clusters

- Quality of clusters = log-likelihood of the data given the mixture model
- EM (Expectation Maximization) algorithm
  - Start with an initial estimation of parameters $\mathbf{\Phi}$
  - Expectation-step: for each point x, compute the membership probability of x in each cluster h
  - Maximization-step: update parameters $\mathbf{\Phi}$
  - Go back to the E-step, unless log-likelihood is not improving

➲Looks familiar (to k-means)?

BFR [Bradley, Fayyad, Reina, AAAI '98]

- Goal: scale to data larger than memory
- Key idea: not all points are equally important to clustering
  - Read points in one memory-full at a time
    - Assume ordering is random
  - From the initial batch, find k initial clusters
  - Process subsequent batches one at a time
  - Most points from previous memory loads are summarized by simple statistics

➲Idea applies to k-means as well as EM clustering

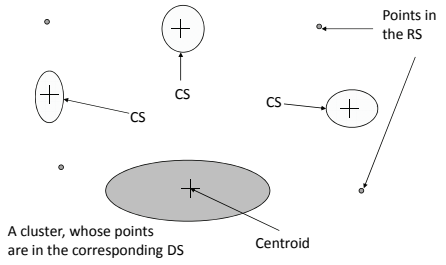
- We discuss in the context of k-means next

Three classes of points

- Discard set (DS): points close enough to a centroid to be summarized
- Compression set (CS): groups of points that are close together but not close to any centroid; summarized, but not assigned to a cluster
- Retained set (RS): isolated points

- Each DS/CS is summarized by (N, SUM, SUMSQ)
  - Exactly the same as BIRCH (N, $\mathbf{LS}$, $\mathbf{SS}$)

DS, CS, RS illustrated



Processing a memory-load

- Among this new batch of points, find those that are "sufficiently" close to a cluster centroid
- Use any main-memory clustering algorithm to cluster the remaining new points and the old RS
  - Clusters becomes CS's; outliers go into RS
- Consider merging CS's
- If this is the last round, merge all CS's and RS into their nearest cluster

➲How does this compare with BIRCH?

A few details

- When should a point be added to a cluster (DS)?
  - E.g., when Mahalanobis distance $\leq$ some threshold
    - sqrt($(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x}-\boldsymbol{\mu})$), where $\boldsymbol{\mu}$ is the centroid and $\mathbf{S}$ is the covariance matrix
      - When $\mathbf{S}$ is diagonal, same as normalized Euclidean
        - Normalize in each dimension d: $z_d = (x_d - \mu_d) / \sigma_d$
        - Take square root of the sum of the squares of the z_d's
- When should two CS's be combined into one?
  - E.g., when variance of combined cluster is below some threshold

Concluding remark

The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.

— Jain and Dubes, *Algorithms for Clustering Data*