Clustering at Large Scales

Veronika Strnadová
CSC Lab
What is clustering?

Finding groups of highly similar objects in unlabeled datasets

or:

Finding groups of highly connected vertices in a graph
Applications

• Image Segmentation
  • *Normalized Cuts and Image Segmentation.*
    Jianbo Shi and Jitendra Malik, 2001

• Load Balancing in Parallel Computing
  • *Multilevel k-way Partitioning Scheme for Irregular Graphs.*
    George Karypis and Vipin Kumar. 1998

• Genetic Mapping
  • *Computational approaches and software tools for genetic linkage map estimation in plants.*
    Jitendra Cheema and Jo Dicks. 2009

• Community Detection
  • *Modularity and community structure in networks.*
    Mark E.J. Newman. 2006
Clustering in Genetic Mapping

“The problem of genetic mapping can essentially be divided into three parts: grouping, ordering, and spacing”

Computational approaches and software tools for genetic linkage map estimation in plants. Cheema 2009
Challenges in the large scale setting

Big Data: Many popular clustering algorithms are inherently difficult to parallelize, and inefficient at large scales
Challenges in the large scale setting

Big Data: Many popular clustering algorithms are inherently difficult to parallelize, and inefficient at large scales

“The computational issue becomes critical when the application involves large-scale data” Jain, *Data Clustering: 50 Years Beyond k-means*
Challenges in the large scale setting

Big Data: Many popular clustering algorithms are inherently difficult to parallelize, and inefficient at large scales

“The computational issue becomes critical when the application involves large-scale data”  Jain, *Data Clustering: 50 Years Beyond k-means*

“There is no clustering algorithm that can be universally used to solve all problems”  Xu & Wunsch, *A Comprehensive Overview of Basic Clustering Algorithms*
Popular Clustering Techniques

• *k*-means
  • K-means/K-medians/K-medoids
  • Better, bigger k-means

• Spectral
  • Optimizing *N*Cut
  • PIC

• Hierarchical
  • Single Linkage
  • BIRCH
  • CURE

• Density-based
  • DBSCAN
  • II DBSCAN

• Multilevel
  • Metis
  • ParMetis
  • Multithreaded Metis

• Modularity
  • Community Detection
\( k \)-means

- “...simpler methods such as \( k \)-means remain the preferred choice of
$k$-means

• “...simpler methods such as $k$-means remain the preferred choice of in many large-scale applications.” Kulis and Jordan, *Revisiting $k$-means: New Algorithms via Bayesian Nonparametrics*. 2012

• “The $k$-means algorithm is very simple and can be easily implemented in solving many practical problems.” Xu and Wunsch, *Survey of Clustering Algorithms*. 2005
$k$-means

• “...simpler methods such as $k$-means remain the preferred choice of in many large-scale applications.” Kulis and Jordan, Revisiting $k$-means: New Algorithms via Bayesian Nonparametrics. 2012

• “The $k$-means algorithm is very simple and can be easily implemented in solving many practical problems.” Xu and Wunsch, Survey of Clustering Algorithms. 2005

• “In spite of the fact that $k$-means was proposed over 50 years ago and thousands of clustering algorithms have been published since then, $k$-means is still widely used.” Jain, Data Clustering: 50 Years Beyond $k$-means.
Given: $x_1, x_2, ..., x_n \in \mathbb{R}^d, k$

Objective: minimize $\sum_{j=1}^{k} \sum_{i=1}^{n} r_{ij} ||x_i - \mu_j||_2^2$ where $r_{ij}$ is an indicator function

Initialize $\mu_1, \mu_2, ..., \mu_k$

Repeat:

1. Find optimal assignments of $x_i$ to $\mu_j$
2. Re-evaluate $\mu_j$ given assignments $x_i$

Until $\sum_{j=1}^{k} \sum_{i=1}^{n} r_{ij} ||x_i - \mu_j||_2^2$ converges to a local minimum

Example: $k = 2$
Convergence of $k$ means

Let $J = \sum_{j=1}^{k} \sum_{i=1}^{n} r_{ij} ||x_i - \mu_j||_2^2$

- Minimize $J$ with respect to $r$: (discrete optimization)
  - Simply assigning each $x_i$ to its closest $\mu$ ensures that $||x_i - \mu_j||$ is minimized for each $x_i$, so let:
    - $r_{ij} = \begin{cases} 1, & \text{if } j = \text{argmin}_{1,...,k} ||x_i - \mu_j||_2^2 \\ 0, & \text{otherwise} \end{cases}$

- Minimize $J$ with respect to $\mu$: (set derivative of $J$ wrt $\mu_j$ to 0)
  - $-2 \sum_{i=1}^{n} r_{nj} (x_i - \mu_j) = 0 \rightarrow \mu_j = \frac{1}{n_j} \sum_{i=1}^{n} r_{nj} x_i$
Distributed K-Means

Given: \( x_1, x_2, \ldots, x_n \in \mathbb{R}^d, k \)

Objective: minimize \( \sum_{j=1}^{k} \sum_{i=1}^{n} r_{ij} ||x_i - \mu_j||^2 \) where \( r_{ij} \) is an indicator function

Initialize \( \mu_1, \mu_2, \ldots, \mu_k \) and keep as a global variable

Repeat:

1. Map: Find optimal assignments of \( x_i \) to \( \mu_j \)
   (map: outputs key=j, value = \( x_i \))
2. Reduce: Re-evaluate \( \mu_j \) given assignments of \( x_i \)
   (combine: partial sums of \( x_i \) with same key)
   (reduce: takes j (key) and list of values \( x_i \), computes a new \( \mu_j \))

Until \( \sum_{j=1}^{k} \sum_{i=1}^{n} r_{ij} ||x_i - \mu_j||^2 \) converges to a local minimum

Parallel k-Means Using MapReduce. Zhao 2009
Divide-and-conquer $k$-Medians

Divide $x_i$ into $l$ groups $\chi_1, \ldots, \chi_l$
For each $i$, find $O(k)$ centers
Assign each $x_i$ to its closest center
Weigh each center by the number of points assigned to it
Let $\chi'$ be the set of weighted centers
Cluster $\chi'$ to find exactly $k$ centers

*Clustering Data Streams: Theory and Practice* by Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani. 2001
Disadvantages of k-means

• Assumes clusters are spherical in shape

• Assumes clusters are of approximately equal size

• Assumes we know $k$ ahead of time

• Sensitivity to Outliers

• Converges to a local optimum of the objective
Disadvantages of k-means

• Assumes clusters are spherical in shape

• Assumes clusters are of approximately equal size

• Assumes we know $k$ ahead of time

• Sensitivity to Outliers

• Converges to a *local* optimum of the objective
Disadvantages of k-means

• Assumes clusters are spherical in shape

• Assumes clusters are of approximately equal size

• Assumes we know $k$ ahead of time

• Sensitivity to Outliers

• Converges to a *local* optimum of the objective
Disadvantages of k-means

• Assumes clusters are spherical in shape

• Assumes clusters are of approximately equal size

• Assumes we know $k$ ahead of time

• Sensitivity to Outliers

• Converges to a *local* optimum of the objective
Addressing the $k$ problem

• Assuming we know $k$ ahead of time:
A parametric, probabilistic view of k-means

Assume that the points $x_i$ were generated from a mixture of $k$ (multivariate) Gaussian distributions:

$$N(x | \mu, \Sigma) = \frac{1}{2\pi^{D/2}} \frac{1}{\sqrt{\det{\Sigma}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

$$p(x) = \sum_{k=1}^{K} \pi_k N(x | \mu_k, \Sigma) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1$$

*Pattern Recognition and Machine Learning*. Christopher M. Bishop. 2006
What is the *most likely* set of parameters $\theta = \{\{\pi_k\}, \{\mu_k\}\}$ that generated the points $x_i$?

We want to maximize the likelihood $p(x|\theta)$

$\Rightarrow$ Solve by *Expectation Maximization*

Xu and Wunsch, *Survey of Clustering Algorithms*
DP-Means

• A probabilistic, nonparametric view of $k$-means:

• Assumes a “countably infinite” mixture of Gaussians, attempt to estimate the parameters of these Gaussians and the way they are combined

• Leads to an intuitive, $k$-means-like algorithm

DP-Means

Given: $x_1, \ldots, x_n, \lambda$

Goal: minimize $\sum_{j=1}^{k} \sum_i r_{ij} \|x_i - \mu_j\| + \lambda k$

Algorithm:

Initialize $k = 1$, $\mu = \text{global mean}$

Initialize $r_{ij} = 1$ for all $i = 1 \ldots n$

Repeat:

• For each $x_i$:
  • compute $d_{ij} = \|x_i - \mu_j\|_2^2, j = 1 \ldots k$
  • If $\min_j d_{ij} > \lambda$, start a new cluster: set $k = k + 1$;
  • Otherwise, set $r_{ij} = \arg\min_j d_{ij}$

• Generate clusters $c_1, \ldots, c_k$ based on $r_{ij}$

• For each cluster compute $\mu_j$

Optimizing a different Global Objective: Spectral Clustering and Graph Cuts
Spectral Clustering

Key idea: use the eigenvectors of a graph Laplacian matrix \( L(G) \) to cluster a graph

\[
L(G) = D - W = \begin{pmatrix}
3 & -1 & -1 & -1 & 0 \\
-1 & 2 & 0 & -1 & 0 \\
-1 & 0 & 3 & -1 & -1 \\
-1 & -1 & -1 & 4 & -1 \\
0 & 0 & -1 & -1 & 2
\end{pmatrix}
\]


*Normalized Cuts and Image Segmentation.* Jianbo Shi and Jitendra Malik. 2001

*A Tutorial on Spectral Clustering.* Ulrike von Luxburg. 2007
Spectral Clustering Algorithm

Given: data points $x_i$ and a similarity function $s(x_i, x_i)$ between them, $k$

1. Construct (normalized) graph Laplacian $L(G(V, E)) = D - W$
2. Find the $k$ eigenvectors corresponding to the $k$ smallest eigenvalues of $L$
3. Let $U$ be the $n \times k$ matrix of eigenvectors
4. Use $k$-means to find $k$ clusters $C'$ letting $x'_i$ be the rows of $U$
5. Assign data point $x_i$ to the $j$th cluster if $x'_i$ was assigned to cluster $j$

*A Tutorial on Spectral Clustering.* Ulrike von Luxburg. 2007
Spectral Clustering Algorithm

Given: data points $x_i$ and a similarity function $s(x_i, x_i)$ between them, $k$

Objective: minimize the ratio cut (or normalized cut)

1. Construct (normalized) graph Laplacian $L(G(V, E)) = D - W$
2. Find the $k$ eigenvectors corresponding to the $k$ smallest eigenvalues of $L$
3. Let $U$ be the $n \times k$ matrix of eigenvectors
4. Use $k$-means to find $k$ clusters $C'$ letting $x'_i$ be the rows of $U$
5. Assign data point $x_i$ to the $j$th cluster if $x'_i$ was assigned to cluster $j$

A Tutorial on Spectral Clustering. Ulrike von Luxburg. 2007
Graph Cuts

A graph cut of $G(V, E)$ is the sum of the weights on a set of edges $S_E \subseteq E$ which cross partitions $A_i$ of the graph

$$W(A_i, \overline{A_i}) = \sum_{x_i \in A_i, x_j \in \overline{A_i}} w_{ij}$$

$$cut(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1,\ldots,k} W(A_i, \overline{A_i})$$

Examples:

$cut(A_1, A_2) = 7$

$cut(A_1, A_2) = 2$
Normalized Cuts

\[ \text{cut}(A_1, A_2) = 2 \]

\[ \text{Ratiocut}(A_1, A_2, \ldots, A_k) = \sum_{i=1,\ldots,k} \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|} \]

\[ \text{Ncut}(A_1, A_2, \ldots, A_k) = \sum_{i=1,\ldots,k} \frac{\text{cut}(A_i, \overline{A_i})}{\text{vol}(A_i)} \]

\[ \text{Ratiocut}(A_1, A_2) = 0.226 \]

\[ \text{Ncut}(A_1, A_2) = 0.063 \]
Relationship between Eigenvectors and Graph Cuts

For some subset of vertices $A$, let:

$$f^T = (f_1, ..., f_n) \in R^n,$$

such that:

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\overline{A}|} & \text{if } v_i \in \overline{A} \end{cases}$$

Then:

$$f^T L f = |V| \cdot RatioCut(A, \overline{A}), \quad f \perp 1 = 0, \quad \|f\| = \sqrt{n}$$

Therefore, minimizing RatioCut is equivalent to minimizing $f^T L f$, subject to:

$$f \perp 1 = 0, \quad \|f\| = \sqrt{n}, \text{ and } f_i \text{ as defined above}$$

_A Tutorial on Spectral Clustering_. Ulrike von Luxburg. 2007
Discrete Optimization:
\[
\min_{f \in \mathbb{R}^n} f^T L f \quad \text{subject to } f \perp 1, \|f\| = \sqrt{n}, \text{ and } f_i \text{ as defined previously}
\]

Relaxed Optimization:
\[
\min_{f \in \mathbb{R}^n} f^T L f \quad \text{subject to } f \perp 1, \|f\| = \sqrt{n}
\]

Since \(1\) is the first eigenvector of \(L\), the solution to this is given by the Rayleigh-Ritz theorem, and we have that the optimal solution to the relaxed optimization problem is the second eigenvector of \(L\)

But this solution is real-valued, so we need to transform the values of \(f\) back into discrete values

A Tutorial on Spectral Clustering. Ulrike von Luxburg. 2007
Spectral Clustering: key ideas

• The $k$ real eigenvectors of the (normalized) graph Laplacian are the optimal solutions to a relaxed version of the ratio (or normalized) cut problem

• An approximate solution is found by mapping the values of the indices in the $k$ real eigenvectors back to the discrete set $(1, \ldots, k)$

• Advantage: works well in practice

• Disadvantages:
  • computationally expensive (need to solve for eigenvectors)
  • approximate solution is not guaranteed to be close to the true solution
Power Iteration Clustering (PIC)

• Large-scale extension to spectral clustering
• Key idea: Use power iteration on \( I - D^{-1}L = D^{-1}W \) until convergence to a linear combination of the \( k \) smallest eigenvectors

\[
\text{Applicable to large-scale text-document classification, works well for small values of } k
\]

\[
\text{If } FF^T = W \text{ then } (D^{-1}W)v = D^{-1}FF^Tv
\]

*Power Iteration Clustering.* Lin and Cohen. 2010,
*A Very Fast Method for Clustering Big Text Datasets.* Lin and Cohen. 2010
Addressing the shape problem

Assuming clusters are spherical in shape
Hierarchical Clustering

- Produces a hierarchy of clusterings
  - Agglomerative or Divisive

Density-based Clustering

- key idea: clusters are sets of data points in regions of high density
Hierarchical Clustering

- Single Linkage
  [Sibson72], [Gower69]
Density-Based Clustering
DBSCAN – Algorithm

Given: $x_i$ in dataset $\mathcal{X}$, $\mathcal{E}$, $\text{MinPts}$

For each $x_i$ in dataset $\mathcal{X}$ do:

If $x_i$ is note yet assigned to a cluster then:

Build a cluster around $x_i$ using $\mathcal{E}$ and $\text{MinPts}$

Building clusters:

All points that are *density-reachable* from $x_i$ are added to the cluster containing $x_i$

*A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Ester et al. KDD96
Example of density reachability:

Suppose \( \text{MinPts} = 4 \)
Parallel DBSCAN

• DBSCAN using the disjoint set data structure:
  • Initially each point is its own “disjoint set”
  • For each point not yet assigned to a cluster, merge its disjoint set with the disjoint sets of all clusters in its $\varepsilon$-neighborhood

• In Parallel:
  • Merge all local disjoint sets that satisfy density reachability, keeping a list of nonlocal vertices that should be merged with local vertices
  • Then, merge in parallel across threads using a union-lock (shared memory)

BIRCH Algorithm

• Goal: cluster large datasets efficiently and handle outliers
• Dynamically build a tree in which leaf nodes are clusters with all points within distance $t$, and all non-leaf nodes are unions of their child nodes
• Then, cluster the leaf nodes using an agglomerative clustering algorithm
BIRCH Algorithm

• Dynamically build a tree in which leaf nodes are clusters with all points within distance \( t \), and all non-leaf nodes are unions of their child nodes
• Then, cluster the leaf nodes using an agglomerative clustering algorithm
BIRCH Algorithm

• Dynamically build a tree in which leaf nodes are clusters with all points within distance $t$, and all non-leaf nodes are unions of their child nodes.
• Then, cluster the leaf nodes using an agglomerative clustering algorithm.
BIRCH Algorithm

• Dynamically build a tree in which leaf nodes are clusters with all points within distance $t$, and all non-leaf nodes are unions of their child nodes
• Then, cluster the leaf nodes using an agglomerative clustering algorithm
BIRCH Algorithm

- Dynamically build a tree in which leaf nodes are clusters with all points within distance $t$, and all non-leaf nodes are unions of their child nodes.
- Then, cluster the leaf nodes using an agglomerative clustering algorithm.
BIRCH

• Key idea: store important cluster information in *cluster feature vectors* $CF_i$ for each cluster $i$

$$CF_i = (N_i, LS_i, SS_i)$$

- $N_i =$ number of points in cluster $i$,
- $LS_i = \sum_{i=1}^{N} x_i$, $SS_i = \sum_{i=1}^{N} x_i^2$
The CF-tree

\[ CF_1 = CF_1 + CF_2 + CF_3 + CF_4 \]

\[ CF_2 = CF_1 + CF_2 + CF_3 + CF_4 \]

Leaf Nodes: \([CF_1], [CF_2], ..., [CF_L]\)
CURE

• Attempts to address the inability of k-means to find clusters of arbitrary shape and size and the computational complexity of hierarchical algorithms

• Key idea: store a “well-scattered” set of points to capture the size and shape of the cluster

_Cure: An Efficient Clustering Algorithm for Large Databases_. Sudipto Guha, Rajeev Rastogi and Kyusek Shim. 2001
CURE Algorithm

• Input: $k$, the number of clusters
• Draw a random sample of the points
• Each point is its own cluster initially
• Each cluster stores a point of representative points and a mean
• Build a $kd$-tree of all clusters
• Use the tree to build a heap that stores $u$.closest for each cluster $u$
• While size(heap) > $k$:
  • Merge together the two closest clusters in the heap
  • Update the representative points in each cluster
  • Update the tree and the heap
• Merging step:
  • New mean is a mean of the means of the clusters being merged
  • Select $c$ well-scattered points based on their distance from the new mean
  • Shrink each representative point closer to the mean
CURE Algorithm: Initial Data
CURE Algorithm: Random Sample
CURE Algorithm: Nearest neighbors
Cure Algorithm: Nearest neighbor heap
CURE: merge
Advantages & Disadvantages of CURE

**Advantages:**
- Finds arbitrarily shaped clusters
- Sample size relative to the dataset size is relatively small if clusters are well-separated
- Reduces sensitivity to outliers due to the shrinkage of representative points

**Disadvantages:**
- $O(n^2 \log_2 n)$
- “a middle ground between centroid-based and all-point extremes”
Highly Application-Specific Clustering Methods

• Multilevel Schemes for load balancing

• Modularity Clustering for community detection
Multilevel $k$-way partitioning for load balancing

- Goal: Partition a graph into $k$ clusters of approximately equal size
- Key Idea: First, condense the graph into a smaller graph, partition/cluster the condensed graph, then expand the small graph back out and refine it

Multilevel partitioning example
Multilevel partitioning example

find a maximal matching
Multilevel partitioning example

find a maximal matching

coarsen
Multilevel partitioning example

find a maximal matching

partition

coarsen
Multilevel partitioning example

find a maximal matching

refine and expand

partition

coarsen
Distributed Multilevel Partitioning

• Coarsening Stage:
  • first, compute a graph coloring using Luby’s algorithm
  • next, for all vertices of the same color in parallel:
    • all vertices with the same color select a match using the heavy edge heuristic
    • after all vertices select a neighbor, synchronize
  • Coarsening ends when there are $O(p)$ vertices

• Partitioning Stage:
  • Recursive bisection, based on nested dissection and greedy partitioning

• Refinement Stage:
  • greedy refinement is used on all vertices of the same color in parallel, based on the gain of each vertex

*Parallel Multilevel k-Way Partitioning Scheme for Irregular Graphs. George Karypis and Vipin Kumar. 1999*
Multi-Threaded Multilevel Partitioning

• Each thread owns a subset of vertices
• Coarsening Stage
  • “unprotected matching”: let $M$ be a shared vector of “matchings”
    • Each thread populates $M$ as it finds a match for its local vertex, writes are done unprotected
    • After all threads finish, each thread corrects the piece of $M$ corresponding to its slice of vertices
• Partitioning Stage
  • “parallel $k$-sectioning”: each thread computes a $k$-partitioning and the one with best edge cut is selected
• Refinement Stage
  • “coarse-grained refinement”: each thread keeps a local queue of vertices sorted by gain values
    • After each thread has moved a fixed number of vertices from the priority queue, all threads communicate and moves are undone until the potential partition weights would result in a balanced partitioning

Multi-Threaded Graph Partitioning. Dominique LaSalle and George Karypis. 2013
Modularity Clustering

- Key idea: find groups of vertices in a graph that have more edges than expected
Modularity

• The expected number of edges between vertices $v_i$ and $v_j$ with degrees $d_i$ and $d_j$ is:
  \[
  \frac{d_i d_j}{2m} \text{ where } m = \frac{1}{2} \sum_i d_i
  \]

• Suppose we have divided the network into clusters $C_1, \ldots, C_l$

• The modularity can be expressed as:

• $Q = \sum_{v_i, v_j \in C_k} \left( A_{ij} - \frac{d_i d_j}{2m} \right)$ where $A$ is the adjacency matrix and the sum is over all $v_i$ and $v_j$ that fall in the cluster $C_k$
Modularity Clustering Algorithm

- Maximizing modularity for two clusters is equivalent to finding the principal eigenvalue/eigenvector pair of a “modularity matrix”
- Proceed by recursive bisection:
  - First, find the principal eigenvalue and eigenvector by power iteration
  - Next, in each successive split:
    - find the principal eigenvector of the “generalized” modularity matrix
  - Stop when modularity no is no longer positive
Advantages of Modularity Clustering

• A “natural” stopping criterion: we don’t need to know $k$ ahead of time

• Seems to be a suitable objective for community detection
Parallel Modularity Clustering

• Agglomerative clustering scheme:
  • Each vertex starts as its own "community"
  • Communities are merged based on their potential contribution to the modularity score
  • Merging is done in parallel using a maximal matching on potential contributions of merging particular edges

*Parallel community detection for massive graphs. Riedy, E. Jason, Henning Meyerhenke, David Ediger, and David A. Bader. 2012*
Frameworks

• CombBLAS
  • *The Combinatorial BLAS: Design, Implementation, and applications*. Aydin Buluc and John Gilbert. 2011
  • A set of primitives for graph computations based on linear algebra
  • Applicable to:
    • K-means clustering
    • Power iteration clustering
    • Markov clustering

• GraphLab
  • *GraphLab: A New Parallel Framework for Machine Learning*. Low, Yucheng, et al. 2010
  • A large-scale library for machine learning
  • Smart scheduling and relaxed data/computational consistency
    • a graph-based data model which simultaneously represents data and computational dependencies
  • Good for:
    • Gibbs Sampling => “fuzzy” DP Means

• ParMetis
  • *ParMetis: Parallel Graph Partitioning and Sparse Matrix Ordering Library*. Karypis, George, Kirk Schloegel and Vipin Kumar. 2003
  • an MPI-based parallel library that implements a partitioning algorithms for unstructured graphs and meshes
    • Multilevel partitioning schemes
Summary: Popular Clustering Techniques

- $k$-means
  - K-means/K-medians/K-medoids
  - Better, bigger $k$-means

- Spectral
  - Optimizing $N$Cut
  - PIC

- Hierarchical
  - Single Linkage
  - BIRCH
  - CURE

- Density-based
  - DBSCAN
  - II DBSCAN

- Multilevel
  - Metis
  - ParMetis
  - Multithreaded Metis

- Modularity
  - Community Detection
Conclusion and Future work

“The tradeoff among different criteria and methods is still dependent on the applications themselves.” A Comprehensive Overview of Basic Clustering Algorithms. Xu & Wunsch. 2005

Future Work:

• Large-scale Nonparametric methods – don’t specify $k$ to start with
  • Decomposing the similarity function in genetic mapping into an inner product in order to use it for PIC
  • Parallel DP-Means
Thank you

John Gilbert, Linda Petzold, Xifeng Yan

Aydin Buluc, Stefanie Jegelka, Joseph Gonzalez

Adam Lugowski, Kevin Deweese
References


Backup Slides
Power Iteration Clustering (PIC)

- Large-scale extension to spectral clustering
- Key ideas: power iteration and the spectral gap
  - Power iteration: $D^{-1}L = I - D^{-1}S$
    \[
    v^{t+1} = \frac{(I - D^{-1}L)v^t}{\| (I - D^{-1}L)v^t \|} \]
  - Let:
    \[
    \delta^{t+1} = \| v^{t+1} - v^t \| \]
  - Stop when:
    \[
    \delta^{t+1} - \delta^t \approx 0 \]
- Applicable to large-scale text-document classification, works well for small values of $k$
  - If $FF^T = S$ then $(I - D^{-1}S)v = v - D^{-1}FF^Tv$

*Power Iteration Clustering.* Lin and Cohen. 2010,
*A Very Fast Method for Clustering Big Text Datasets.* Lin and Cohen. 2010
Utility of CF vectors in BIRCH

Because we have the additivity property of CF vectors:

\[ \text{CF}_i + \text{CF}_j = (N_i + N_j, LS_i + LS_j, SS_i + SS_j) \]

We can easily compute updates to the mean, radius, or diameter of a CF vector, and we can use CF vectors to compute distances between centroids and incoming points.

Example: distance between centroid \( \mu_i \) of cluster \( i \) and centroid \( \mu_j \) of cluster \( j \)

\[ \| \mu_i - \mu_j \| = \left( \left( \frac{LS_i}{N_i} - \frac{LS_j}{N_j} \right)^T \left( \frac{LS_i}{N_i} - \frac{LS_j}{N_j} \right) \right)^{1/2} \]
Multilevel $k$-Way: Matching & Coarsening Step

Randomly select a vertex, and match it with the neighbor that has the heaviest edge ("heavy edge matching")

Multilevel $k$-way Partitioning Scheme for Irregular Graphs. George Karypis and Vipin Kumar. 1998
Multilevel $k$-Way: Partition Step

Use spectral clustering to find $k$ clusters
Multilevel $k$-Way: Uncoarsening and Refinement Step

- Expand the vertices and edges back out to the finer graph
- Use the $gain$ heuristic to move vertices one at a time, and keep the configuration that gives the best balanced partitioning

The gain of a vertex $v$ is the net reduction in the weight of edges that would result in moving $v$ from partition A to partition B

Example: gain of moving vertex 3 to partition B

$gain(v_3) = 2 - 4 = -2$
Generative Process specification

- A convention for specifying a probability distribution in a more compact way, by leaving out certain conditional independence properties in the specification; these properties are implied.
- At each step, the distribution depends only on the previously defined random variables
- Example of a generative process specification:

\[
X_1, X_2 \sim \text{Bernoulli}\left(\frac{1}{2}\right) \\
X_3 \sim N(X_1 + X_2, \sigma^2) \\
X_4 \sim N(aX_2 + b, 1) \\
X_5 \sim \begin{cases} 
1 & \text{if } X_4 \geq 0 \\
0 & \text{else}
\end{cases}
\]

This specification means that: If \( X \) is a vector such that: \( X = (X_1, \ldots, X_5) \) then it respects this graphical model:

In other words, the multivariate distribution on \( X \) respects the following conditional independence probabilities:

\[
p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_2)p(x_5|x_4)
\]
A Gaussian Process

We can equivalently look at a mixture of Gaussians as a generative model:

- Choose a cluster, then choose a point from that cluster

A real-valued stochastic process \( \{X_t, t \in T\} \) where \( T \) is an index set, is a Gaussian process if for any choice of distinct values \( t_1, ..., t_k \in T \), the random vector \( \mathbf{X} = (X_{t_1}, ..., X_{t_k})' \) has a multivariate normal distribution with mean \( \mu = E[\mathbf{X}] \) and covariance matrix \( \Sigma = \text{cov}(\mathbf{X}, \mathbf{X}) \).

Example: \( T = \{1, 2, 3, 4, 5, 6\} \), \( X = (X_1, X_2, X_3, X_4) \)

\[
p(X = x) = p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_2) = N(E[X], \text{cov}(X, X))
\]

\[
X_1, X_2 \sim N\left(\frac{1}{2}, \sigma^2\right)
\]

\[
X_3 \sim N(X_1 + X_2, \sigma^2)
\]

\[
X_4 \sim N(aX_2 + b, 1)
\]
A Dirichlet Process

• In Bayesian modelling, we assume we have an underlying and unknown distribution that we wish to infer given some observed data. We can do this by assuming a prior, and then computing the posterior given some data using that: posterior $\propto$ likelihood $\times$ prior. But constraining the prior to be parametric limits the types of inferences we can make. The nonparametric approach solves this problem, but we have to choose a prior that makes the posterior calculation tractable.

• The Dirichlet Process is a stochastic process used in Bayesian nonparametric models of data. Each draw from a Dirichlet process is itself a distribution. The Dirichlet process has Dirichlet distributed finite dimensional marginal distributions. Distributions drawn from a Dirichlet process are discrete, but cannot be described using a finite number of parameters, “hence the classification as a nonparametric model.”
A Dirichlet Mixture

Assume $G_0$ is a prior distribution over the means of the Gaussians:

$$\mu_1, \mu_2, \ldots, \mu_k \sim G_0$$

The mixing coefficients are distributed according to a Dirichlet distribution:

$$\pi \sim Dir(k, \pi_0)$$

We choose one of the clusters from a discrete distribution over $\pi$, such as a multinomial distribution:

$$z_1, \ldots, z_n \sim Discrete(\pi)$$

Finally, we choose $x_i$ according to the Gaussian distribution with mean $\mu_z$

$$x_1, \ldots, x_n \sim N(\mu_z, \sigma I)$$

“A way to view the DP mixture model is to take the limit of the above model as $k \to \infty$, when choosing $\pi_0$ to be $(\alpha/k)1$”
• The Dirichlet distribution is a commonly used as a prior distribution over $\theta$ when $p(x|\theta)$ is a multinomial distribution over k values (it’s the conjugate prior to the multinomial)

• Computing $E(X)$ from p is difficult/intractable/inefficient
  • Instead, approximate $E(f(X))$ using an MCMC method such as Gibbs Sampling
  • Construct a markov chain that has p as its stationary distribution
    • Ergodic theorem says: if we run chain long enough, it will converge to the st. dist. And so the end state is approximately what it truly is
    • Ergodic theorem + irreducibility: just use sample mean over all states that markov chain takes to approximate sample mean
A clustering function

A clustering function takes a distance function $d$ on a set $S$ of $n \geq 2$ points and outputs a clustering $\Gamma$.

Example: $S = \{1,2,3,4,5,6,7\}$

$f(d) = \Gamma = \{\{1,2,3\}, \{4,5,6,7\}\}$

An Impossibility Theorem for Clustering. Jon Kleinberg. 2003
Scale Invariance: \( f(d) = f(\alpha \ast d) \)

\[
\begin{align*}
  f(d_1) &= \Gamma = \{\{1,2,3\}, \{4,5,6,7\}\} \\
  f(\alpha \ast d_1) &= \Gamma = \{\{1,2,3\}, \{4,5,6,7\}\}
\end{align*}
\]
Consistency

\[ f(d) = \Gamma = \{\{1,2,3\}, \{4,5,6,7\}\} \]

\[ f(d') = \Gamma = \{\{1,2,3\}, \{4,5,6,7\}\} \]
Richness:

Range($f$) is equal to the set of all partitions of $S$. etc...
Impossibility Theorem for Clustering

A clustering function should have the three following properties:

\textit{scale invariance, consistency, and richness}

Theorem: \textit{For each }\( n \geq 2 \), \textit{there is no clustering function that satisfies scale invariance, consistency, and richness}

“...there is no solution that simultaneously satisfies a small collection of simple properties”

− Jon Kleinberg, \textit{An Impossibility Theorem for Clustering}. 2003
Possibility of Clustering

Instead of thinking about clustering functions, let’s think about clustering results – what makes a good clustering?

There exist *Clustering Quality Measures* that satisfy consistency, richness, scalability

\[ m(\{\{1,2,3\}, \{4,5,6,7\}\}) = \alpha \in R \]

*Measures of Clustering Quality: A Working Set of Axioms for Clustering*. Margareta Ackerman and Shai Ben-David. 2009
Modularity

• Suppose we have divided the network into two groups
• The expected number of edges between vertices $v_i$ and $v_j$ with degrees $d_i$ and $d_j$ is:
  \[ \frac{d_id_j}{2m} \text{ where } m = \frac{1}{2} \sum_i d_i \]
• The modularity can be expressed a modularity is defined as:
  \[ Q = \sum_{i,j} (A_{ij} - \frac{d_id_j}{2m}) \text{ where } A \text{ is the adjacency matrix and the sum is over all } \]
  \[ u_i^T s \frac{1}{4m} s^T B s \text{ where } s_i = 1 \text{ if } v_i \text{ is in group 1 and } s_i = -1 \text{ if } v_i \text{ is in group 2, and } \]
  \[ B_{ij} = A_{ij} - \frac{d_id_j}{2m} \]
• We can therefore write the modularity as:
  \[ Q = \frac{1}{4m} \sum_{i=1,...,n} (u_i^T s)^2 \lambda_i \]
Gibbs Sampling

• An MCMC method:
  • tool for sampling from and computing expectations with respect to very complicated and high dimensional probability distribution
  • MCMC constructs a markov chain whose stationary distribution is the complicated distribution
  • Can be used in DP means for expectation maximization